Ministère de l'enseignement supérieur et de la recherche scientifique

Université Saâd Dahleb

Mémoire de fin d'étude pour obtention du diplôme d'ingénieur d'état en informatique

THEME

Exclusion mutuelle en environnements mobiles

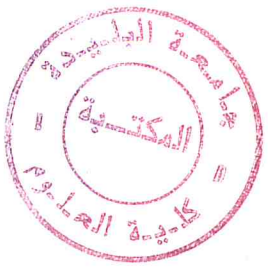
Réalisé par

Melle BENMAHDJOUB Radia
Melle TIFOURA Amina Rahima

Proposé par

Mr DAHAMNI Foudil

Promotion 2002-2003



Sommaire

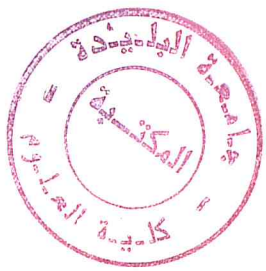
Introduction Générale

Chapitre I : Etude de l'environnement mobile

Chapitre II : Exclusion mutuelle en environnement mobile

Chapitre III : Etude de l'outil de simulation NS-2

Chapitre IV : Mise en œuvre



Liste des figures

INTRODUCTION GENERALE :

L'essor des technologies sans fil offre aujourd'hui de nouvelles perspectives dans le domaine des télécommunications. L'évolution récente des moyens de la communication sans fil a permis la manipulation de l'information à travers des unités de calculs portables (ordinateurs portables) qui ont des caractéristiques particulières (une faible capacité de stockage, une source d'énergie autonome…) et accède au réseau à travers une interface de communication sans fil. Comparant avec l'ancien environnement (environnement statique), le nouvel environnement résultant, appelé environnement mobile permet aux unités de calculs une libre mobilité et il ne pose aucune restriction sur la localisation des usagers.

La mobilité ou le nomadisme et le nouveau mode de communication utilisé, engendrant de nouvelles caractéristiques propres à l'environnement mobile : une fréquente déconnexion, un débit de communication des ressources modeste, et des ressources d'énergie limitées.

Dans notre travail, on se concentre sur la mobilité des machines, bien sûr, le matériel seul n'est pas suffisant, même en supposant un réseau sans fil fiable, performant et omniprésent, il est nécessaire de fournir un support logiciel important pour prendre en compte les trois facteurs essentiels des environnements mobiles :

- La mobilité des machines, ce qui nécessite des mécanismes étendus de routage, d'adressage et nomage.
- Les aléas des communications sans fil, avec de fréquentes déconnexions et une bande passante très réduite et très variable.
- Les ressources limitées des machines mobiles, telles que l'espace disque ou encore la durée de vie des batteries.

En dépit des problèmes cités, un utilisateur souhaite se déplacer librement et continuer à travailler le plus normalement possible, il est donc indispensable de fournir une continuité de service malgré les relocalisations, les déconnexions, ou les perturbations réseau.

Nous nous somme penchés sur l'impact de la mobilité des machines afin de recevoir certains algorithmes du système d'exploitation (algorithmes d'exclusion mutuelle) en reportant les coûts de calculs, de communications et de stockage sur le réseau statique, de sorte à solliciter le moins possible les faibles ressources des unités mobiles et les aléas de la communication sans fil ; pour cela, il nous faut développer un outil de simulation d'algorithmes adapté à un tel environnement afin d'évaluer les éventuels coûts. On se propose de le faire sous le logiciel «NS » NETWORK SIMULATOR.

Notre mémoire regroupera quatre chapitres, dans le premier chapitre nous décrirons l'environnement mobile et ses caractéristiques (spécificité, architecture, …etc.). Au second chapitre, nous présenterons l'exclusion mutuelle et ses différents algorithmes avec une adaptation de l'un d'eux à l'environnement mobile. Le troisième chapitre comportera une étude de l'outil de simulation (NS), que nous utiliserons pour définir une plate-forme de simulation pour implémentation en C++ et OTCL de l'algorithme d'exclusion mutuelle. Au quatrième et dernier chapitre, nous présenterons une proposition de

résolution basée sur l'algorithme proposé et d'autres algorithmes d'exclusion mutuelle ainsi que le calcul du coût induit.

I. Introduction :

Les environnements mobiles offrent une grande flexibilité d'emploi ; en particulier ils permettent la mise en réseau des sites dont le câblage serait trop onéreux à réaliser dans leur totalité, voir même impossible (par exemple en présence d'une composante mobile). On cite l'exemple du projet NAFIN (Netherlands Armed Forces Integrated Network), qui a visé l'amélioration des performances des forces militaires de l'aire et marine en intégrant la technologie des réseaux sans fil, plus les environnements mobiles offrent beaucoup d'avantages par rapport à l'environnement habituel (statique), mais en contre partie, des problèmes peuvent apparaître causés par les nouvelles caractéristiques du système c'est pour cela que de nouvelles solutions doivent êtres trouvés pour s'adapter aux limitations qui existent, ainsi qu'aux facteurs qui rentrent en jeu lors de la conception.

II. Définition de l'environnement mobile :

Un environnement mobile est un système composé de sites mobiles et qui permet à ses utilisateurs d'accéder à l'information indépendamment du facteur temps de leurs positions géographiques ; ces unités communiquent à travers leurs interfaces sans fil [IMI 94]et peuvent être de configurations diverses : avec ou sans disque, des capacités de sauvegarde et de traitement plus ou moins modestes ; un tel environnement doit contenir les éléments suivants :

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

- Une liaison de communication sans fil permet la liberté de mouvement aux utilisateurs.
- Des mécanismes systèmes nécessaires à sa gestion.
- Un réseau pouvant supporter la mobilité (architecture).

Pour établir un tel environnement il est important de revoir les algorithmes classiques (exclusion mutuelle, élection, diffusion…) et cela dans le but de pouvoir :

- Accéder aux données ou aux fichiers partagés à distance.
- Faire suivre les messages des machines qui se déplacent.
- Définir une architecture pour les systèmes des ordinateurs mobiles.
- Résoudre les problèmes de connexion/déconnexion.
- Gérer la consommation de l'énergie.

III. Caractéristiques des environnements mobiles :

Contrairement à l'environnement statique, l'environnement mobile se caractérise par :

III .1. Les connexions sans fil :

La liberté de mouvement qu'offre l'environnement mobile est dû essentiellement à la connexion sans fil ; d'ailleurs on distingue deux grandes catégories de connexions sans fil :

- Les ondes lumineuses.
- Les ondes radiofréquences.

III .1.1. Les ondes lumineuses :

Elles regroupent : - les ondes infrarouges.
- les ondes laser.

Les infrarouges offrent un bon débit avec toutefois les contraintes de visibilités car il est nécessaire que l'émetteur et le récepteur soit en vue l'un et l'autre sans oublier que cette technique obéit aux lois de réflexions et d'absorption limitant leurs modes d'utilisation ; ajoutant à cela la consommation très élevée de l'énergie, ce qui épuise la source qui alimente le mobile.

III .1.2. Les ondes radiofréquences :

Elles regroupent deux classes : - les ondes radio.
- les micro ondes.

Les ondes radiofréquences traversent les différents obstacles sauf les charpentes métalliques et peuvent atteindre de grandes distances. Leur seul problème c'est qu'elles sont limitées par la législation des télécommunications, car la bande magnétique réservée pour une application mobile est relativement réduite, cependant cette technique demeure la plus souple d'application.

III .1.3. La fiabilité des connexions sans fil :

La communication sans fil est moins fiable que la communication dans les réseaux filaire car la propagation du signal subit des perturbations (erreurs de transfert, microcoupure, timeout) dues à l'environnement, qui altèrent l'information transférée. Il s'ensuit alors un accroissement du délai alloué à la

transition du message à cause de l'augmentation du nombre de retransmissions. La connexion peut être aussi rompue ou altérée par la mobilité des sites.

On peut aussi observer les problèmes dus à la connexion sans fil quand par exemple lors d'un rassemblement populaire il peut y avoir une surcharge du réseau liée au nombre d'unités mobiles dans une même cellule (dans le cas des réseaux cellulaires). L'une aussi des limites de la communication sans fil vient de la relative faiblesse de la bande passante des technologies utilisées, et cette bande passante est partagée entre les utilisateurs d'une même cellule.

Pour augmenter la capacité de service d'un réseau, deux techniques sont utilisées : la technique de recouvrement des cellules sur différentes longueurs d'ondes et celle qui réduit la portée du signal pour avoir plus de cellules mais de rayon moindre couvrant une région donnée.

III .2. Les machines mobiles :

La flexibilité qu'offre l'environnement mobile nécessite l'utilisation de machines mobiles mais celles-ci entraînent des problèmes qui ne sont pas rencontrés en informatique classique, car les machines mobiles se caractérisent essentiellement par la portabilité et qui dit portabilité dit limitation d'énergie et miniaturisation des composants.

L'augmentation de la durée de vie des batteries se caractérise [FOR, 94] par les mécanismes suivants :

- L'utilisation de processeurs équipés de fonctions de gestion de la consommation d'énergie.

- Permettre des opérations de conservation d'énergie telles que l'arrêt du disque, l'extinction de l'écran, et les opérations de réduction en besoin de calcul et en communication.
- L'introduction de l'opération de mise en veille qui permet à un portable en attente de suspendre toutes ses exécutions et de réduire la vitesse de son horloge. La réalisation d'un tel mécanisme nécessite de reprendre les protocoles de communication(contrôle des erreurs, compression, cryptage, …etc.), et les algorithmes constituants les systèmes d'exploitation gérant ces mobiles (exclusion mutuelle, élection, interblocage, … etc.).

Pour des raisons de portabilité, les éléments constituants un portable sont miniaturisés : le poids des machines actuelles est en moyenne de 2KG et leur taille ne dépasse pas généralement le format A4, cette miniaturisation limite les capacités de calcul et de stockage des machines portables [FOR, 94].

III .2.1. Modes de fonctionnement des mobiles : [BAG, 96]

Pour les sites mobiles on peut distinguer trois modes de fonctionnement différents :

III .2.1.1. Le mode connecté :

Le mobile dispose d'une connexion normale au réseau à la manière d'une station classique.

III .2.1.2. Le mode déconnecté :

Dans ce cas le mobile n'est plus physiquement relié au réseau et cela est du aux :

- Fortes perturbations.
- Surcharges momentanées.
- Ruptures volontaires ou non (pannes).

En informatique mobile, les déconnexions sont prévisibles la plupart du temps, il est donc possible de les préparer (par exemple : charger des données sur le mobile, et les envoyer sur le réseau fixe).

III .2.1.3. Le mode veille :

On utilise le mode veille pour la préservation des ressources énergétiques en envisageant :

- Une vitesse d'horloge réduite.
- Des exécutions d'applications suspendues.

La particularité de ce mode c'est qu'une unité mobile reste accessible au système, contrairement au mode déconnecté, et elle peut reprendre son mode normal à la réception d'un message et repasser le plus simplement en mode connecté.

III .3. Les réseaux mobiles :

III .3.1. Définition :

Les réseaux mobiles sans fil, peuvent être classés en deux classes : les réseaux avec infrastructure qui utilisent généralement le modèle de la

communication cellulaire, et les réseaux sans infrastructure ou les réseaux « ad hoc ».

Plusieurs systèmes utilisent déjà le modèle cellulaire et connaissent une très forte expansion à l'heure actuelle (le réseau GSM par exemple) mais requièrent une importante infrastructure logistique et matérielle fixe.

III .3.2. Les réseaux sans infrastructure (ad hoc) :

L'évolution récente de la technologie dans le domaine de la communication sans fil pousse aujourd'hui les chercheurs et les spécialistes à faire des efforts énormes pour aboutir au but primordial des réseaux qui est : « Le pouvoir d'accéder à une information n'importe quand et à n'importe quel endroit ».

Le concept des réseaux *ad hoc* essai d'étendre les notions de la mobilité à toutes les composantes de l'environnement. Ici, contrairement aux réseaux basés sur la communication cellulaire, aucune administration centralisée n'est possible, ce sont les hôtes mobiles elles-mêmes qui forment d'une manière *ad hoc* une infrastructure du réseau. Aucune position ou limitation n'est faite sur la taille du réseau *ad hoc*, le réseau peut contenir des centaines ou des milliers d'unités mobiles.

Les réseaux *ad hoc* sont idéals pour les applications caractérisées par une absence (ou la non-fiabilité) d'une infrastructure préexistante, tel que les applications militaires et les autres applications de tactiques comme les opérations de secours (incendies, tremblement de terre…) et les missions d'explorations.

III .3.2.1. Définition :

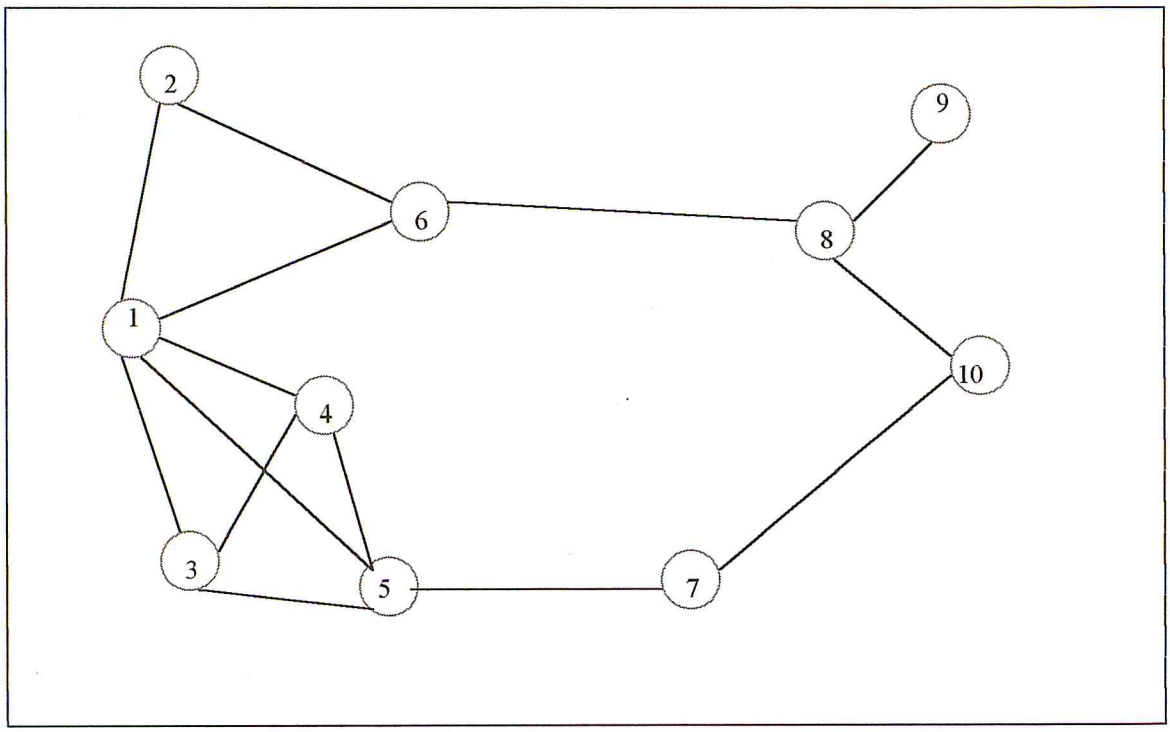
Le réseau mobile *ad hoc,* appelé généralement MANET (Mobile Ad hoc NETwork), consiste en une grande population, relativement dense, d'unités mobiles qui se déplacent dans un territoire quelconque et dont le seul moyen de communication est l'utilisation des interfaces sans fil, sans l'aide d'une infrastructure préexistante ou administration centralisée.

III .3.2.2. Caractéristiques des réseaux ad hoc :

Les réseaux mobiles *ad hoc* sont caractérisés par :

- Une bande passante limitée : Les réseaux basés sur une communication sans fil se caractérisent par l'utilisation d'un médium de communication partagé, ce partage fait que la bande passante réservée à un hôte soit modeste.
- Des contraintes d'énergie : Le paramètre d'énergie doit être pris en considération dans tout contrôle fait par le système, car les hôtes mobiles sont alimentés par des sources d'énergies autonomes comme les batteries.
- Une sécurité physique limitée : Les réseaux *ad hoc* sont plus touchés par le paramètre de sécurité que les réseaux filaires, et cela se justifie par les limitations physiques qui font que le contrôle des données transférées doit être minimisé.
- L'absence d'infrastructure : Vu que les réseaux *ad hoc* se distinguent des autres réseaux par l'absence d'infrastructure, c'est les hôtes qui sont responsables d'établir et de maintenir la connectivité du réseau d'une manière continue.

- Une topologie dynamique : Les unités mobiles du réseau, se déplacent d'une façon libre et arbitraire (voir **Figure I-1**). Par conséquent la topologie du réseau peut changer à des instants imprévisibles d'une manière rapide et aléatoire (voir **Figure I-2**)et cela induit des déconnexions des unités mobiles très fréquentes.



◯ :nœud (ou unité mobile). _____ : lien de communication.

Figure I -1 : La modélisation d'un réseau *ad hoc.*

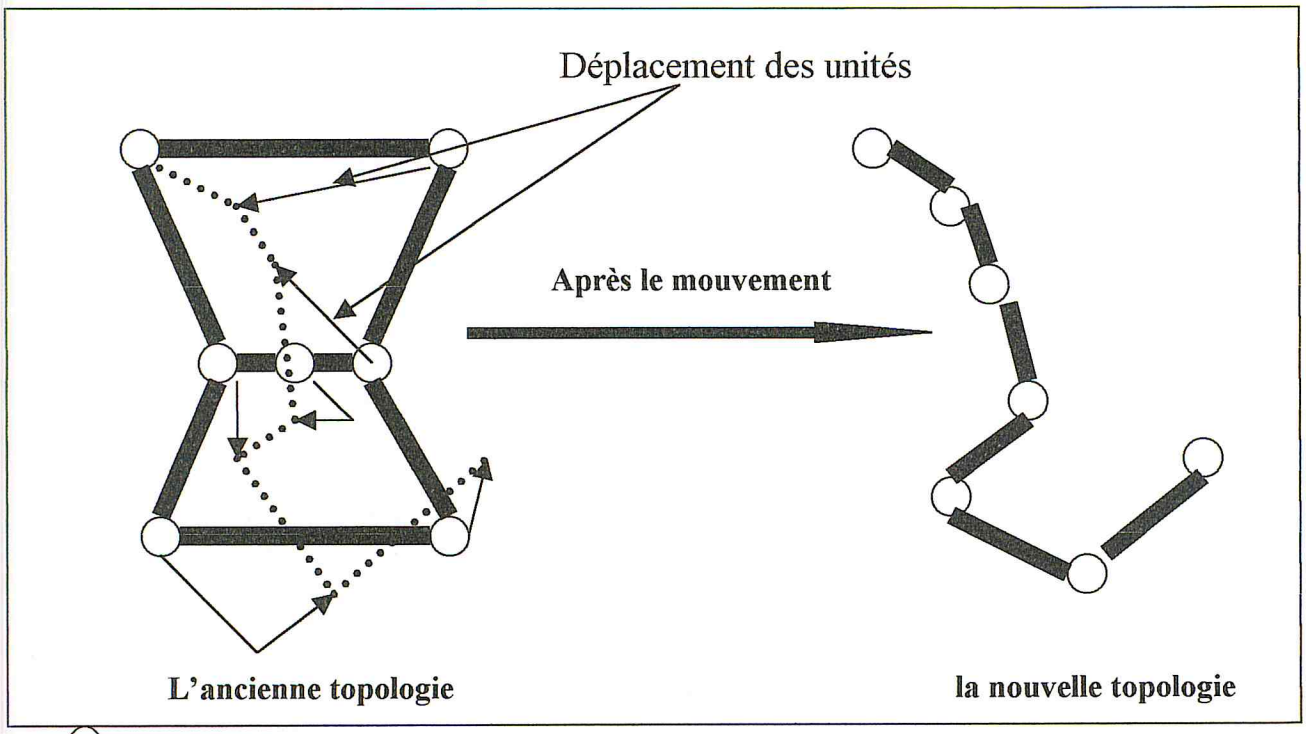
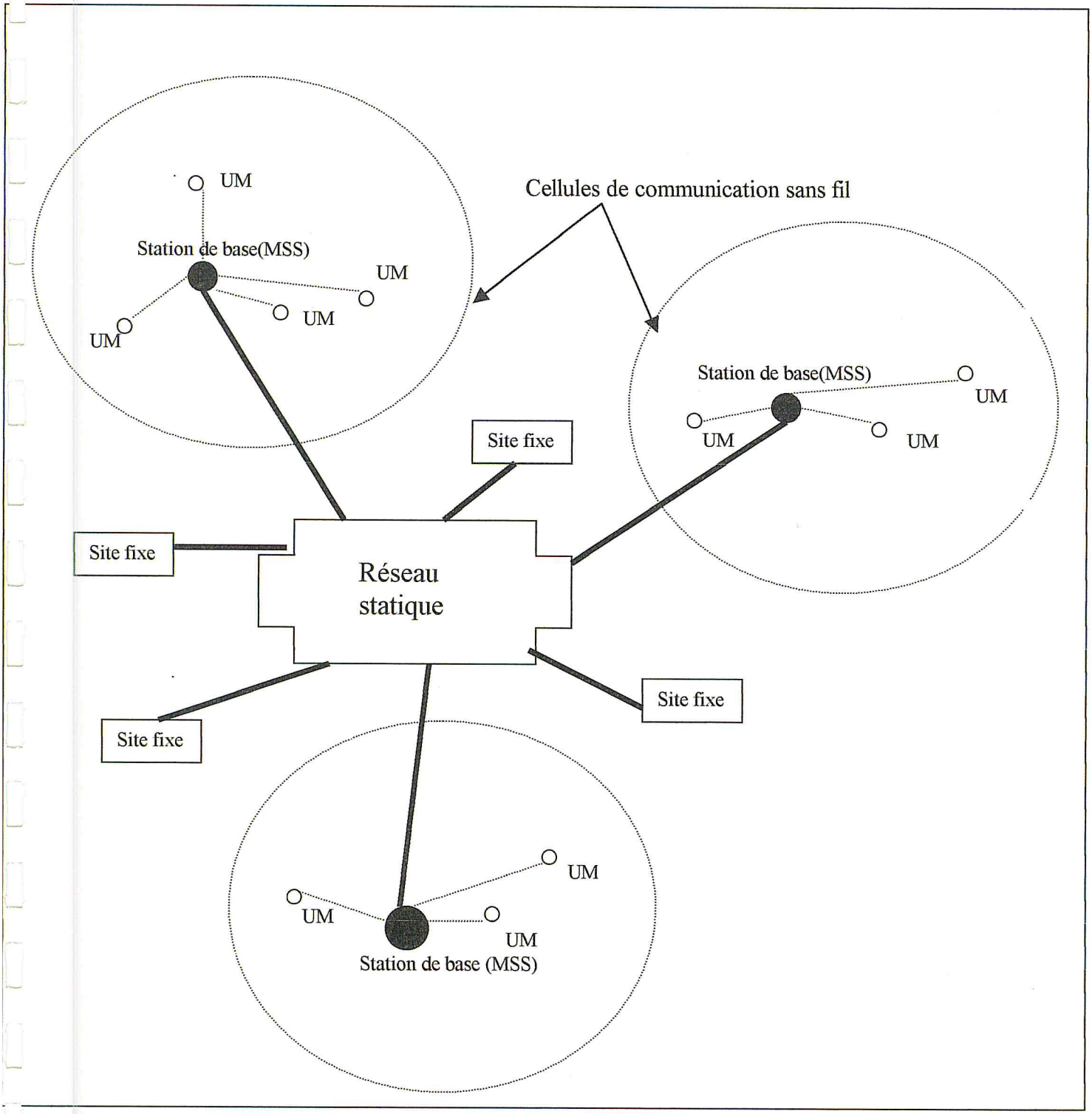


Figure I -2 : Le changement de la topologie *ad hoc.*

D'une manière générale les réseaux *ad hoc* sont utilisés dans toute application où le développement d'une infrastructure réseau filaire est trop contraignant, ou bien parce qu'il est difficile à mettre en place, alors que la durée d'installation du réseau ne justifie pas le câblage à demeure.

III .3.3. Les réseaux avec infrastructure :

Ce type de réseaux utilise généralement le modèle de la communication cellulaire qui est basée essentiellement sur l'utilisation des réseaux filaire (Internet, ATM) et la présence des stations de bases qui couvrent les différentes unités mobiles du système ; donc on peut dire que le réseau avec infrastructure est constitué de deux parties [IMI, 94] comme le montre la **Figure I -3**.



UM : unité mobile.

Figure I -3. Le modèle des réseaux avec infrastructure.

La partie fixe est représentée par des stations fixes reliées par des liaisons filaires et / ou satellites, certaines de ces stations fixes sont équipées d'interfaces de communication sans fil pour servir de point d'accès aux stations mobiles. Ces stations fixes sont appelées MSS (Mobile Service Station) et sont réparties sur l'ensemble des cellules pour gérer l'espace géographique couvert par le réseau ; quant à la partie mobile elle est représentée par un ensemble d'hôtes mobiles (ordinateurs portables) qui sont équipés eux aussi d'interfaces de communications sans fil.

Il faut savoir qu'à un instant donné, un hôte mobile ne peut appartenir qu'à une seule cellule et ne peut accéder au réseau qu'à travers la MSS gérant la cellule où il se trouve.

On peut schématiser la cellule de communication comme suit : **Figure I -4**.

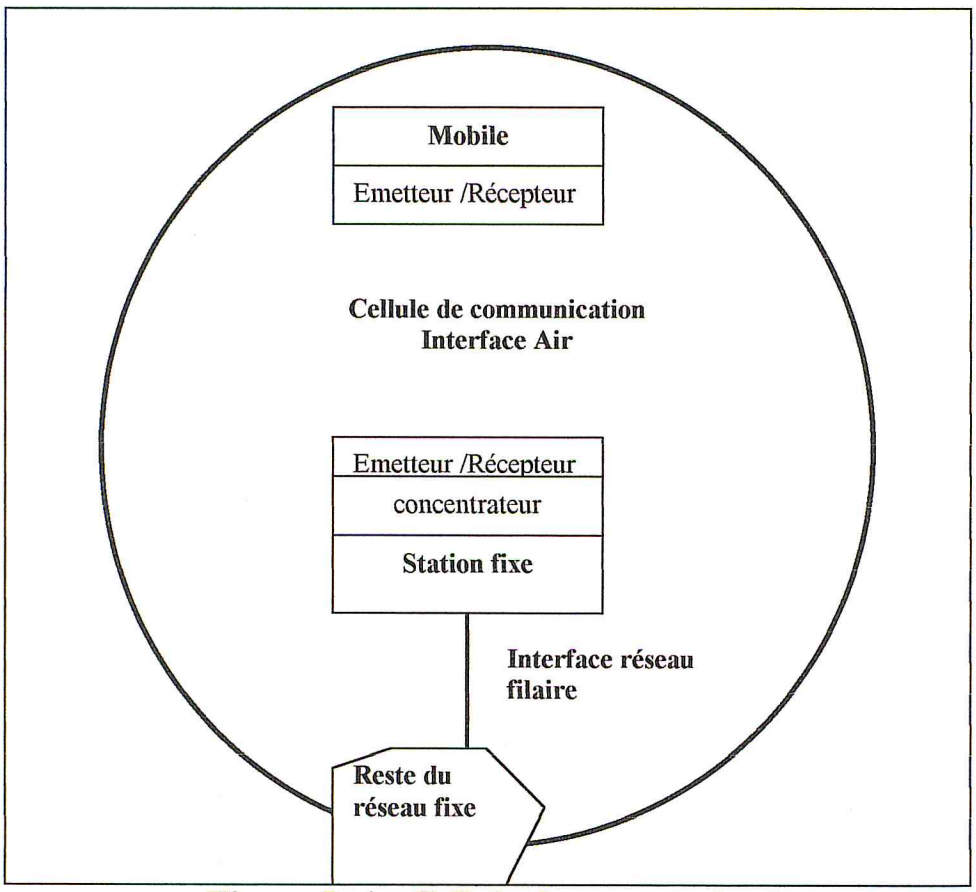


Figure I -4 : Cellule de communication

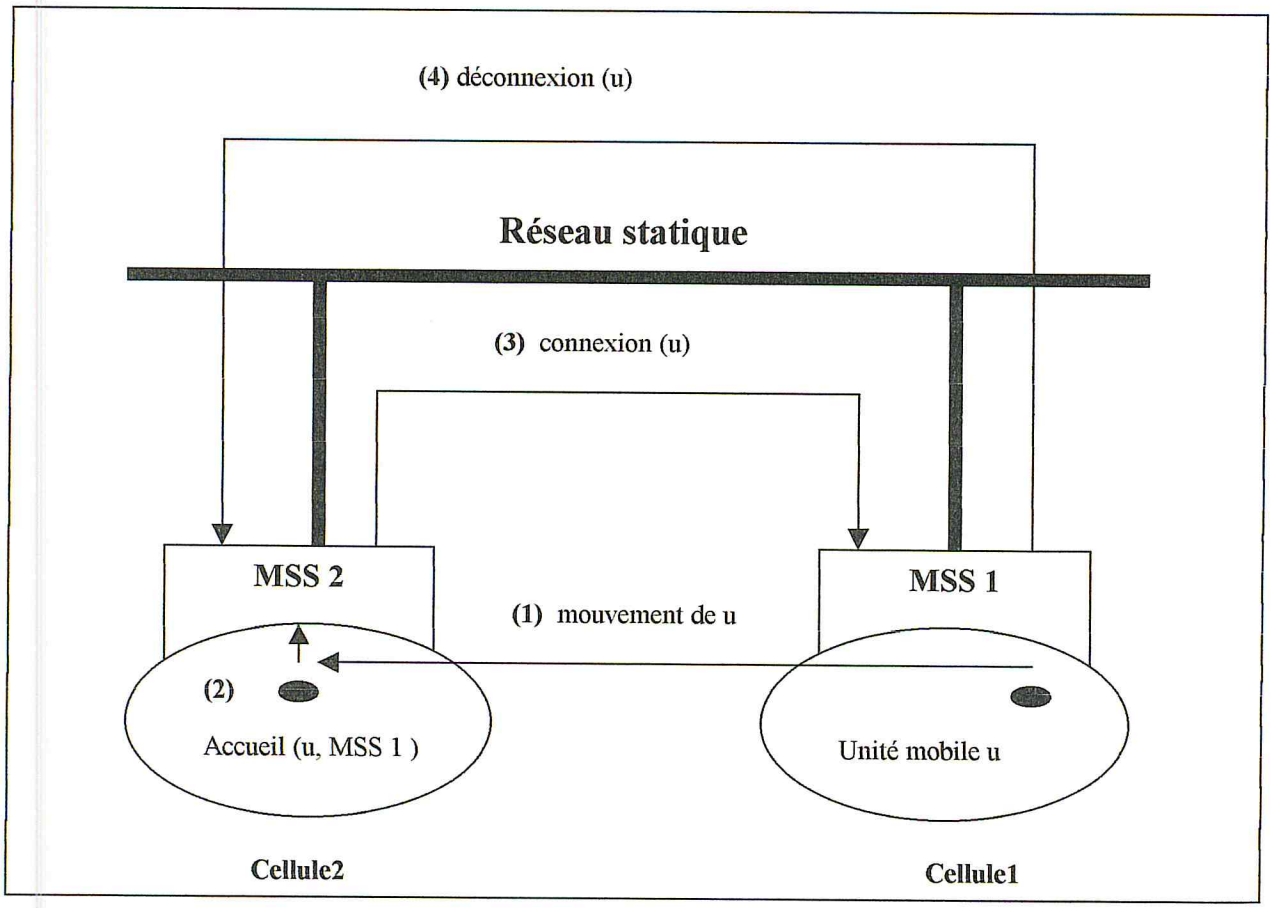


Figure I -5 : Schéma de la procédure *hand-off*

III .3.3.1. L'allocation des canaux :

L'allocation de l'ensemble des canaux disponibles entre les différentes cellules, sans qu'il y ait d'interférences dans les cellules adjacentes minimise sérieusement le nombre de sessions de communication ; on peut maximiser ce nombre de sessions en appliquant le principe de réutilisation de la même fréquence dans des cellules adjacentes [SIN, 95].

L'affectation des canaux se fait selon deux méthodes :

a) La méthode statique :

Dans cette méthode le spectre de communication est divisé en canaux affectés aux différentes cellules de manière définitive pour éviter les interférences ; un mobile se trouve dans une cellule utilise un canal réservé à cette cellule pour communiquer avec sa station de base.

L'inconvénient de cette méthode c'est qu'en cas de surcharge de mobile dans la même, le nombre de canaux ne suffit plus ce qui entraîne une saturation de la cellule alors que les canaux des cellules voisine peuvent être libres ; son avantage réside dans sa simplicité.

b) La méthode dynamique :

Dans cette stratégie l'ensemble des canaux réservés à une cellule varie avec le temps ; si celle-ci est surchargée, elle emprunte des canaux à ses voisines immédiates de sorte que l'emprunt ne cause pas des interférences entre les cellules adjacentes ; on peut dire que c'est la méthode la plus utilisée.

IV. Les problèmes liés à la mobilité :

Malgré tous les avantages que présente l'environnement mobile on peut néant moins remarquer de nombreux problèmes qu'il faut prendre en compte. Parmi ces problèmes on cite :

IV .1. Le nommage et l'adressage : [BAG, 95], [PAU, 99]

Dans les réseaux mobiles, les sites mobiles sont en mouvement continu, et utilise différents points d'accès au réseau ce qui induit un fréquent changement

d'adresse, donc pour identifier un site on doit séparer le nom du site mobile (qui est permanent) de son adresse (qui est temporaire), ce qui fait que l'on ne peut pas nommer un site par son adresse car elle varie dans le temps.

IV .2. Le routage : [BAG, 95]

Dans un réseau mobile le routage des paquets vers les sites mobiles a besoins que les routes des messages changent de manière à refléter la localisation courante des mobiles.

Aujourd'hui beaucoup de schémas utilisent un serveur de routage de paquets qui est localisé à une adresse connue, mais les inconvénients de ce serveur c'est que les messages peuvent parcourir deux fois le même chemin ou un chemin inutile car contraint de passer par le serveur.

IV .3. La dépendance de localisation :

La localisation des mobiles doit être très précise, car cette localisation sera utile pour la recherche d'informations liées à la localisation comme faire appel, ou encore répondre à des requêtes spécifiques.

IV .4. La gestion des données :

En ce qui concerne la gestion des données l'informatique mobile induit beaucoup de problèmes qui sont :

IV.4.1. La récupération de données : [GRA, 00], [GRA, 01]

La récupération des données nécessite l'adaptation des protocoles afin de permettre aux mobiles de rester connecté au réseau pendant ses déplacements, et en particulier la gestion du *hand-off* et il est nécessaire aussi de supprimer les données résiduelles dans les stations fixes occupées par des mobiles à un instant donné pour éviter la saturation de l'espace mémoire.

IV .4.2. La diffusion des données :

La mobilité engendre la redirection des messages et le maintient des routes ; une station de base peut émettre un message à toutes les unités mobiles se trouvant dans sa cellule en une seule opération de transfert. En cas de diffusion sélective (émission multi destinataire d'un message à un ensemble d'unités mobiles) ; de cela peut résulter que le message n'arrive pas à certaines unités mobiles destinataires car leurs entrées dans la cellule à eu lieu après la diffusion du message ou bien elles ont quitté la cellule avant la diffusion du message.

IV .4.3. La gestion des déconnexions :

L'un des inconvénients de la mobilité est les déconnexions fréquentes que les chercheurs tentent de résoudre, car la déconnexion d'une unité mobile au cours de l'exécution d'un programme au quel elle participe engendre sa suspension jusqu'au rétablissement d'une nouvelle connexion, et de cela résulte des problèmes de récupération des données et diffusion de données.

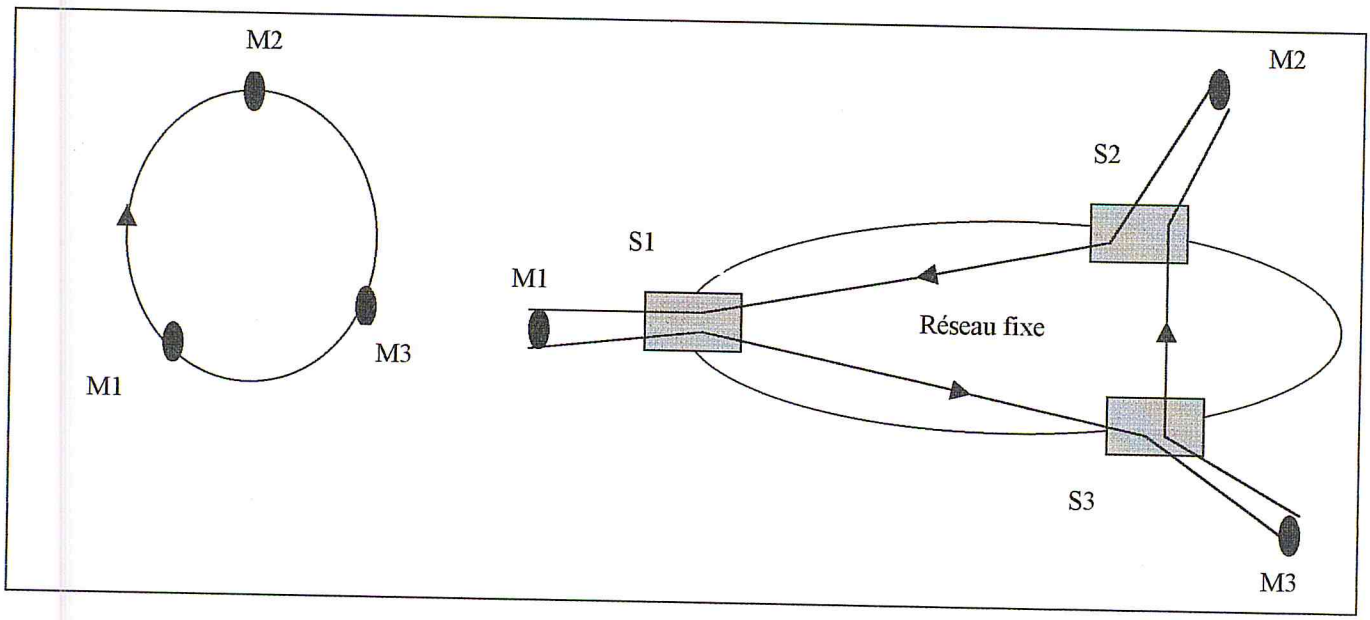
IV .5. Architecture :

Plus l'hôte mobile se déplace plus on constate de points d'accès au réseau différents ; ce qui engendre la modification de son architecture. Dans les réseaux distribués statiques, certains algorithmes reposent sur les structures logiques de type arbre, graphe ou anneau indépendamment de la structure physique du réseau. La mobilité d'une machine implique que sa localisation par rapport au reste du réseau évolue au cours du temps et la connectique de tout le réseau sera modifiée.

IV .5.1. Les anneaux logiques :

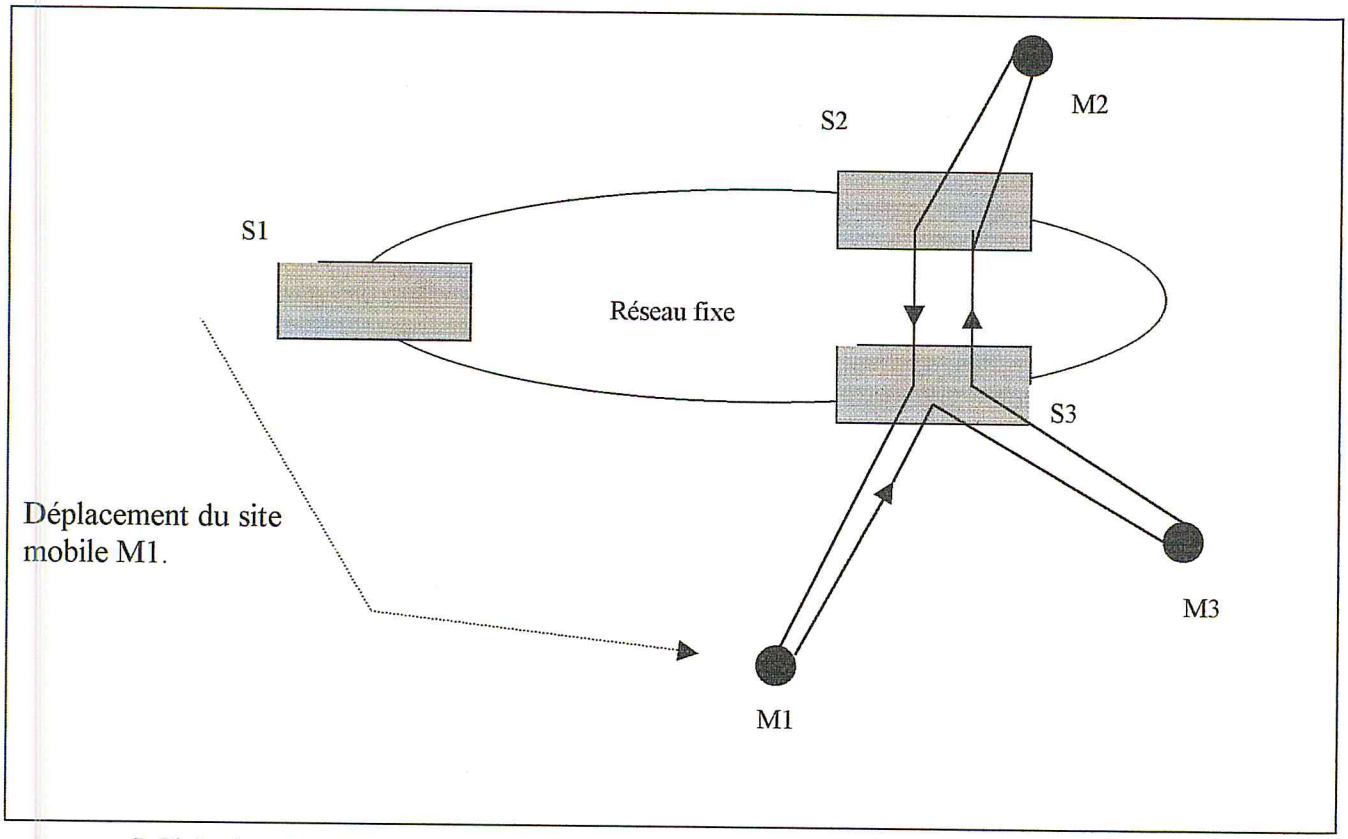
On peut voir dans la **Figure I-6** que la mobilité des machines induisent des changements dans l'anneau à travers le temps selon leur emplacement dans le réseau.

La **Figure I-7** nous montre que si un mobile change de position, sa station de support change aussi.



M1,M2,M3 : sites mobiles. S1,S2,S3 :stations support (fixes).

Figure I -6 : Anneau logique unidirectionnel avec sites mobiles [BAG,95].



M1,M2,M3 : sites mobiles. S1,S2,S3 : station support(fixes)

Figure I .7 : Effet de la mobilité sur l'anneau logique unidirectionnel [BAG,95]

V. Conclusion :

Ce chapitre a été axé sur le concept des environnements mobile et l'utilisation de la technologie sans fil. L'évolution rapide qu'a connu la technologie sans fil récemment, a permit l'application de nouveaux systèmes de communication qui offrent plus d'avantages que les systèmes classiques. Ces nouveaux systèmes n'astreignent plus l'usager à être fixé à un endroit, car elle permet une certaine liberté de mouvement.

Le but de ce chapitre a été de donner un aperçu général sur la technologie mobile qui ne cesse d'avancer de jour en jour.

La mobilité induit des propriétés qu'un concepteur doit prendre en compte ; ces propriétés se résument en :

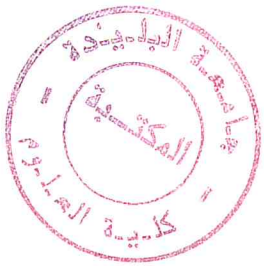
- Une configuration mobile du réseau à cause de la mobilité des hôtes et leurs déconnexions incessantes.
- Une énergie limitée sur les mobiles.
- Une faible puissance de calculs et de stockage des machines.
- Une bande passante limitée des médiums sans fil.
- Une introduction des coûts de recherche et de localisation dans l'évaluation des algorithmes élaborés.
- Une rupture fréquente des médiums sans fil.
- Une diffusion naturelle des médiums sans fil.

L'introduction de ces propriétés dans un système distribué a besoin que l'on revoie l'ensemble des problèmes et des structures de contrôle pour les adapter aux environnements mobiles.

Nous allons dans ce qui suit aborder l'un des problèmes de base qui est «l'exclusion mutuelle » ; après une présentation et une critique des solutions existantes dans la littérature, nous présenterons une solution adaptée à l'environnement mobile.

chapitre II

Exclusion mutuelle en environnement mobile



I. Introduction :

Au cours de ce dernier siècle l'informatique et plus précisément l'utilisation des ordinateurs a connu une véritable révolution. De 1945 aux années 80 tous les ordinateurs étaient gros et chers et les mini ordinateurs coûtaient plusieurs milliers de dollars donc les entreprises possédaient que quelques ordinateurs travaillants localement car il n'y avait pas de possibilités d'interconnexion.

Au milieu des années 80 le monde technologique vie deux évolutions qui vont changer la situation. La première fut le développement de microprocesseurs puissants ; car au départ c'était des machines à 8 bits puis rapidement on a pu observer celles à 16 bits, 32 bits et 64 bits avec une puissance de calcul comparable à celle d'un ordinateur et un prix abordable. La seconde évolution fut l'invention des réseaux locaux à haut débit qu'on connaît aussi sous le nom de LAN (local area network). Ces systèmes permettent de connecter des centaines d'ordinateurs pour échanger des informations en une milliseconde environ. Le résultat de ces deux évolutions c'est qu'on peut faire facilement coopérer un grand nombre d'UC reliées par un réseau haut débit ; c'est ce qu'on appelle un système distribué.

Les systèmes distribués constituent donc un grand avancement dans le domaine informatique et comme tout avancée ils présentent des avantages comme le partage des données et des périphériques, une facilité de communication interpersonnelle et une souplesse de travail. Mais aussi ils ont des inconvénients tel que le piratage de données, la saturation du réseau, mais l'inconvénient le plus pesant en programmation distribuée demeure le

phénomène d'exclusion mutuelle car les systèmes qui mettent en œuvre plusieurs processus sont le plus souvent programmés en utilisant les sections critiques. Quand un processus doit lire ou mettre à jour de façon sure des structures de données partagées, il commence par entrer en section critique pour réaliser une exclusion mutuelle, c'est à dire s'assurer que d'autres processus n'utiliseront pas la même structure de données au même moment.

On peut exposer notre problème en disant qu'il consiste à considérer un ensemble de n processus reliés par un réseau complètement maillé et fiable. Il s'agit d'offrir à chaque processus un protocole lui permettant d'enter et de sortir d'une section critique. Le protocole doit remédier aux problèmes d'interblocages, de famine et doit garantir un ordre de satisfaction entre les requêtes d'enter en section critique des processus candidats.

Ce deuxième chapitre mettra en évidence les différents algorithmes existants résolvant le problème de l'exclusion mutuelle, ainsi que leur adaptation aux conditions de l'environnement mobile. Enfin, une proposition d'algorithme résolvant l'exclusion mutuelle dans l'environnement mobile sera exposée.

II. Principes de base de l'algorithmique distribuée :

Un algorithme distribué est reconnu comme étant un ensemble de processus qui communiquent entre eux par échange de messages [RAY, 85].

Dans cet ensemble on peut distinguer deux sous ensembles :

- L'ensemble des processus qui peuvent émettre, recevoir et traiter l'information.

- Les liaisons par lesquelles les processus s'échangent des messages.

Ces deux sous-ensembles définissent un réseau de communication muni d'une topologie déterminant les relations entre processus. Les différents processus sont autorisés à exécuter simultanément du fait de l'indépendance de traitement de chaque processus et cela engendre des problèmes de coopération et de compétition entre processus.

En algorithmique distribuée on distingue trois concepts de bases : le calcul diffusant, le jeton circulant, et l'estampillage.

II.1. Le calcul diffusant :

Le calcul diffusant est un concept proposé par Dijkstra [DIJ, 80], il se caractérise par la définition d'un processus particulier n'ayant pas de prédécesseurs et où tous les autres processus sont à l'état passif.

Le calcul diffusant consiste pour le processus particulier, d'envoyer un ou plusieurs messages à un ou plusieurs de ses successeurs. A la réception du premier message, tout processus passif devient actif et a la possibilité d'émettre des messages à ses successeurs [MAD, 87]. L'un des buts du calcul diffusant est d'effectuer des contrôles particuliers tel que la terminaison.

II.2. Le jeton circulant :

Le jeton circulant est une technique qui consiste à faire circuler un privilège qu'on appel jeton entre un ensemble de processus connectés en structure d'anneau ; l'anneau peut être défini de manière statique ou reconfiguré au cours

de l'exécution. Cette technique est utilisée pour gérer l'accès en section critique (exclusion mutuelle).

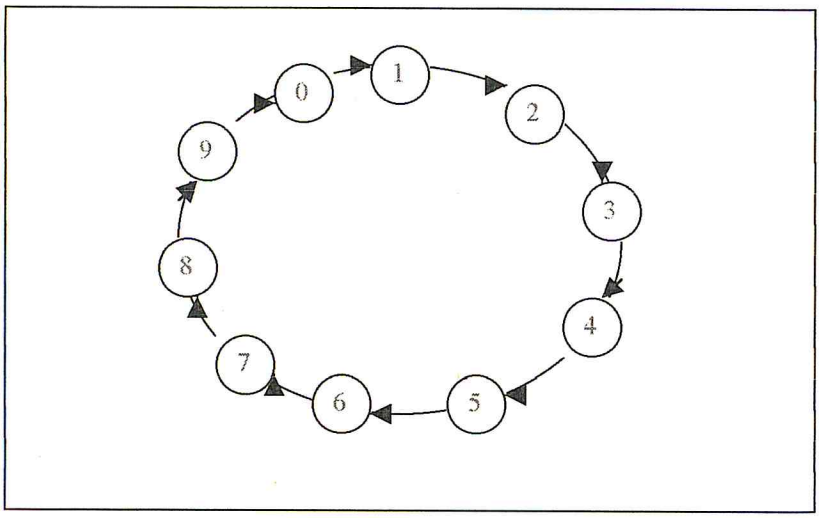


Figure II-1 : Processus en structure d'anneau.

Remarque :

- L'ordre des processus importe peu mais chaque processus doit savoir qui le suit.

II.3. L'estampillage :

L'estampillage appelé aussi horloge logique ; technique proposée par Lamport [LAM, 78], sont but est d'ordonner les événements qui se produisent par l'exécution des processus réalisant un algorithme distribué.

Chaque processus pi a pour charge de gérer un compteur local hi qui est appelé communément horloge de pi, chaque message émis par pi est marqué par son horloge hi ; tel que : message (m, hi, i)

Quand un processus un processus pi reçoit un message (m, hj, j) de pj, son horloge prend la valeur [MAX (hi, hj)+1], et cette valeur sera considérée comme étant la date de réception du message.

Quand un processus pi émet un message (m, hi, i) à un processus pj, il incrémente son horloge hi avant de l'inclure dans le message.

Un message (m1, hi, i) est dit antérieur à un message (m2, hj, j) si (hi<hj) ou (hi=hj et i<j).

La technique d'estampillage est utilisée le plus souvent dans les algorithmes d'exclusion mutuelle pour ordonner une file d'attente distribuée dans les différents sites.

III. L'exclusion mutuelle :

L'avancée technologique que fut la programmation parallèle a engendré quelques problèmes, mais le problème le plus pesant par son importance demeure le problème d'exclusion mutuelle. Ce phénomène, essentiel pour la conception des systèmes d'exploitation, son but est d'établir des opérations de base permettant de résoudre les conflits résultants du partage des ressources dans un système informatique par un ensemble de processus[RAY, 84] ; et plus exactement lors d'un accès à une ressource partagée.

Au niveau de la programmation, toutes les instructions relatives à l'utilisation de la ressource crique, sont appelées sections critiques. Pour contrôler l'accès à la section critique nous disposons d'un protocole caractérisé par la demande de la ressource et sa libération, et il est utilisé comme suit :

1) Demande d'entrer en section critique.
2) Accès à la section critique.
3) Libération de la section critique.

Pour qu'un algorithme d'exclusion mutuelle soit opérationnel, il doit contenir un minimum de propriétés dues à Dijkstra [DIJ, 74] et qui se résume dans ce qui suit :

- A tout instant un processus au plus peut se trouver en section critique.
- Si plusieurs processus se retrouvent bloqués en attente de la section critique, alors qu'aucun ne s'y trouve, l'un d'entre eux doit y accéder au bout d'un temps fini.
- Si un processus est bloqué hors d'une section critique son blocage ne doit pas empêcher un autre processus d'y accéder.
- La solution doit être la même pour tous les processus et aucun d'eux ne jouent un rôle privilégié.

Les algorithmes d'exclusion mutuelle dans un environnement parallèle reposent sur le principe d'échange de messages c'est pour cela qu'ils doivent être équitables et exempts d'interblocages

III.1. Les algorithmes d'exclusion mutuelle : .

Dans un contexte distribué on constate que les algorithmes ne reposent sur aucun dispositif centralisé tel que le mot mémoire et ou horloge.

A la lumière de cette définition on peut distinguer les algorithmes distribués sous la forme de deux grandes familles, en effet il existe deux modes d'expressions pour les algorithmes parallèles d'exclusion mutuelle. Le premier

utilise les variables d'état, et le second utilise la communication de messages pour véhiculer les informations nécessaires entre les processus.

III.1.1. Les algorithmes distribués basés sur les variables d'états :

L'état d'un processus est définit par ses variables qui lui sont propres et locales.

Une variable est dite propre à un processus si seulement ce dernier est le seul a avoir le pouvoir de la lire et de l'écrire à la foie, alors que tous les autres processus ne sont autorisés qu'a la lire.

Une variable est dite locale à un processus si elle est inaccessible aux autres processus [RAY, 84]. Le comportement de ces variables se résume comme suit : La lecture d'une variable se fait par l'envoie d'un message au propriétaire, qui a son tour répond par un message comportant la valeur de cette variable.

L'inconvénient de ce type d'algorithmes est le nombre élevé des messages nécessaires pour gérer l'accès à la section critique ; de plus, on peut noter que les processus testent en permanence les variables d'état qui les intéressent jusqu'à la satisfaction de leurs conditions d'attente. Ces tests aveugles et lourds augmentent énormément le nombre de messages échangés au sein du réseau.

Les algorithmes basés sur les variables d'état conviennent plus aux systèmes centralisés, qu'aux systèmes distribués de part l'indépendance offerte aux différents processus et vu le nombre faramineux de messages qu'ils utilisent ; on peut tout de même observer que ces algorithmes résistent généralement bien, car la panne d'un processus n'affecte pas le comportement global de l'algorithme. on peut citer à titre d'exemples des algorithmes faisant partie de cette classe :

- Algorithme de la Boulangerie [LAM, 74] qui utilise la technique d'estampillage pour ordonner les requêtes.
- Algorithme auto-stabilisateur [DIJ, 74] qui utilise la technique du jeton circulant.
- Algorithme de Hehner et Shyamasundar [HEH, 81] qui utilise la technique d'estampillage avec un comportement non symétrique pour les processus.

III.1.2. Les algorithmes distribués basés sur la communication de messages :

A la différence des algorithmes basés sur les variables d'état, un processus ne demande plus les informations des autres processus, mais il en reçoit ; chaque fois qu'un processus change d'état, il diffuse cette modification aux autres ; de cette manière on peut observer une réduction dans le nombre de messages échangés entre processus, en éliminant les messages de tests inutiles dans le cas où les variables testées ne changeraient pas d'état. Ce type d'algorithmes repose sur le principe d'estampillage pour gérer les messages transmis et ils présentent l'avantage d'être totalement indépendants de la topologie du réseau où ils s'exécutent ; et cela grâce à leur mode d'expression qui est la communication de messages.

Nous allons aborder dans ce qui suit un nombre d'algorithmes appartenant à cette classe d'algorithme distribuée.

III.1.2.1. L'algorithme de distribution d'une file d'attente [LAM, 78] :

L'algorithme de Lamport [LAM, 78] utilise le principe de l'horloge logique pour ordonner les demandes d'entrée en section critique des différents processus ; ce qui caractérise cet algorithme c'est son utilisation d'une file d'attente distribuée, dont il existe une représentation partielle sur chaque site.

Un processus qui veut entrer en section critique (SC), transmet sa demande datée de son horloge à tous les autres processus. Un processus ne peut entrer en section critique que si sa demande est la plus antérieure (horloge) que toutes les autres demandes, et qu'il a reçu un accord de tous les autres processus, et cela pour garantir qu'il n'y a pas eu un message antérieur en transit. Le protocole d'accès à la section critique (SC) se déroule comme suit :

- Diffuser une requête datée de son horloge locale
- Attendre qu'elle soit la plus ancienne
- Entrer en section critique
- Diffuser une libération de la section critique

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

- (n-1) messages pour diffuser la demande.
- (n-1) messages pour recevoir les accords.
- (n-1) messages pour diffuser la libération.

L'algorithme de distribution d'une file d'attente garantit l'exclusion mutuelle, tout en étant équitable et pauvre d'interblocages. Cet algorithme résiste aux pannes en ajoutant dans la file un champ ***absent*** pour chaque site ; ce

champ est modifié en fonction du message (absent, k, i) provoqué par le réseau de transport indiquant la panne de ce site, il est consulté par chaque processus avant de comparer son horloge avec une autre, s'il est à vrai, alors la comparaison est annulée.

III.1.2.2. L'algorithme de Ricart et Agrawala [RIC, 81] :

Cet algorithme est considéré comme une optimisation de celui de Lamport et cela en éliminant les messages accusés de réception. Ce qui ramène le nombre de messages véhiculés pour espérer entrer en section critique à 2(n-1) qui se résument en :

- (n-1) messages pour diffuser la demande.
- (n-1) message pour recevoir les réponses à la demande.

Le protocole d'accès à la section critique se déroule comme suit :

- Diffuser une requête datée de son horloge
- Attendre des réponses favorables de tous les autres sites tout en répondant aux requêtes les plus anciennes et en différant les nouvelles.
- Accéder à la section critique
- Répondre aux requêtes différées

L'algorithme de Ricart et Agrawala repose sur un réseau de transport considéré sans erreurs, où les temps de transit sont variables et les messages peuvent se doubler.

III.1.2.3. L'algorithme de Le Lann [LEL, 77] :

L'algorithme repose sur le principe de circulation d'une information particulière appelée jeton ou privilège entre un ensemble de sites organisé en anneau virtuel, et défini par une numérotation des sites choisis par convention de 0 à n-1 (avec n le nombre de sites constituant l'anneau).

Le jeton ou privilège circule dans le sens des numéros croissants. A l'initialisation du système un site et un seul possède le jeton, de ce fait il a l'autorisation d'accéder à la section critique.

L'implémentation de cet algorithme connaît un certain nombre de problèmes tel que la panne d'un site au moment où il détient le jeton ou celle du système de communication qui causent la perte du jeton privilège. La solution à ces problèmes nécessite de nouveaux protocoles qui peuvent détecter la perte du jeton et sa régénération.

La panne d'un processus pose le problème de la reconfiguration de l'anneau logique. Cette tâche est à la charge du système de transport qui pour cela fournit à chaque processus deux variables d'état : voisin gauche et voisin droit qui varie de 0 à n-1 et assure leur mise à jour lors de la panne d'un site.

On peut comptabiliser le nombre de messages nécessaires pour accéder à la section critique, en disant qu'il varie de 0 dans le cas où tous ces processus effectueraient leur demande, à une infinité de messages dans le cas où aucun processus ne désirerait y accéder.

III.1.2.4. L'algorithme de Suzuki et Kasami [SUZ, 82] :

L'algorithme de Sazuki et Kasami repose sur les bases de l'algorithme de Le Lann [LEL, 77] avec une optimisation consistant à faire accompagner le jeton d'un tableau de n entiers (avec n le nombre de sites) contenant des informations sur les dernières requêtes satisfaites des différents sites.

Un processus qui demande d'entrer en section critique diffuse une demande avec un numéro supérieur à celui circulant avec le jeton. Un processus pi venant de sortir de la région critique transmet le jeton au premier de ses voisins, dans l'ordre pi+1,..., pn, p0,...,pi-1 ayant le numéro de sa requête supérieur à celui dans le jeton. Ce procédé évite l'envoi du jeton aux processus qui ne demandent pas d'entrer en section critique.

Cette technique permet une optimisation du nombre de messages nécessaires pour une exclusion mutuelle tel qu'on constate qu'il y a diffusion de :

- n messages dans le cas ou plus d'un processus demanderaient la ressource.
- 0 message si le processus demandant la ressource possède déjà le jeton.

Bien que garantissant l'exclusion mutuelle l'algorithme soit imperméable aux interblocages seulement dans le cas où les messages seraient délivrés dans un temps finis. L'inconvénient majeur de l'algorithme réside dans la taille du message du jeton qui est constituée de l'information particulière concernant le jeton et d'un tableau de n éléments, de ce fait le jeton n'est plus une simple information connue par tous les sites, comme dans le cas de l'algorithme précèdent, mais un ensemble d'informations liées à tous les sites et dépandants de l'évolution des demandes dans ces sites ; par conséquent les techniques de

régénération du jeton après sa perte, utilisées dans l'algorithme de Le Lann ne peuvent être utilisées dans cet algorithme d'où la nécessité de nouvelles techniques.

Dans ce qui suit nous allons vous en présenter l'algorithme :

Processus pi

Var osn : 0. +(;
Jeton présent, dedans : booléen ;
Jeton, requête : tableau [1..n] de 0..+(;

Protocole d'accès

- Si non (jeton présent) alors
  Début
  - Osn := osn +1 ;
  - Diffuser (request, osn, i) ;
  - Attendre (jtn, jeton) ;

  Fin
- Dedans :=vrai ;
- Jeton présent :=vrai ;

Section critique

- Jeton [i] := osn ;
- Dedans := faux ;
- Pour j de i+1..n, de 1..i-1 faire
  Début
  - Si requete [j]>jeton [j] Et jeton présent alors

Début

- Jeton présent := faux ;
- Envoyer (jtn, jeton) à j ;
- Exit ;

Fin

Fin

Gestion des messages

- A la réception de (REQUEST,k, j) de j faire

Début

- Requete [j] := max (requete [j], k) ;
- Si jeton présent et non (dedans) alors

Début

- Pour j de i+1..n, de 1..i-1 faire

Début

- Si requete [j]> jeton[j] et Jeton présent alors

Début

- Jeton présent :=faux ;
- Envoyer(jtn, jeton) à j ;
- Exit ;

Fin

Fin

Fin

Fin

IV. Exclusion mutuelle en environnement mobile :

Le but primaire de l'informatique mobile est de surmonter les problèmes résultants de la mobilité des utilisateurs tout en offrant les mêmes possibilités de services que ceux des systèmes statiques, tel que le partage des ressources réparties sur les différents hôtes mobiles ou fixes. L'accès à de telles ressources comme les fichiers, les mémoires, et les imprimantes entraîne des problèmes de cohérence.

Des algorithmes existent pour permettre l'accès en exclusion mutuelle à ces ressources dans les systèmes répartis mais ne traitent pas les contraintes des mobiles.

IV.1. Le modèle du système entrepris :

Le modèle utilisé pour répondre aux besoins de la mobilité, est le modèle cellulaire dans un réseau constitué d'une partie fixe avec des stations de base réparties sur l'ensemble des cellules pour l'espace géographique, et d'une partie mobile représentée par l'ensemble des hôtes mobiles (voir la figure I.3).

Dans ce modèle chaque station de base (MSS) dispose d'une table comportant 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 même MSS. A un instant donné, un mobile ne peut appartenir qu'à une seule cellule, qui représente sa position actuelle. Le réseau statique, supposé fiable, délivre les messages, séquentiellement entre les stations de base (MSS) avec une latence arbitraire. Le réseau sans fil assure une liaison

FIFO (First In First Out) des messages entre la MSS et les hôtes mobiles (MHS) locaux. Tant qu'un mobile n'a pas quitté une MSS, il reçoit les messages, séquentiellement, qui lui sont envoyés par cette MSS. Un mobile 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 supprime de sa liste des mobiles locaux et met à jour un indicateur pour informer ceux qui le cherchent de sa déconnexion.

Quand un mobile veut se connecter à une nouvelle MSS, il envoi un message *connect (Mh-id)* à cette dernière, où Mh-id représente l'identité du mobile, qui l'ajoute à sa liste de mobiles locaux et envoie un message *Let (Mh-id)* à son ex MSS, laquelle lui retourne ses paramètres à partir du message numéro 'r', et met à jour l'indicateur de déconnexion (volontaire ou non).

On peut observer que dans l'environnement mobile, l'évaluation des algorithmes repose sur de nouvelles mesures [BAD, 94] telle que :

- C_{fixe} : qui est le coût induit par l'envoi d'un message via le réseau statique.
- $C_{wireless}$: qui est le coût induit par l'envoi d'un message via une liaison sans fil.
- C_{seach} : qui est le coût de localisation d'un mobile.

IV.2. L'utilisation de l'exclusion mutuelle classique :

Les hôtes mobiles, tout comme les fixes peuvent enter en concurrence pour accéder à des ressources critiques ou partagées telles que les fichiers, les variables, les imprimantes, et autres. Pour pouvoir remédier et contrôler les conflits résultants d'une telle concurrence, nous allons prendre pour cela deux

algorithmes classiques qui sont l'algorithme de Lamport [LAM, 78] et celui de Le Lann [LEL, 77] qu'on appliquera à l'environnement mobile.

L'algorithme de Lamport [LAM, 78] est basé sur la communication de messages ; son application directe consiste à l'exécuter au niveau de chaque mobile. Le nouvel algorithme a le même principe que celui qu'on a décrit précédemment [III.1.2.1]dans ce chapitre sauf que l'on fait précéder 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 un coût estimé à :

$$2\,C_{wireless} + C_{fixe} + C_{seach}$$

Comme l'algorithme classique nécessite un échange de $3\,(N_{MH} - 1)$ messages pour satisfaire une demande, le nouvel algorithme par conséquent un coût de :

$$3\,(N_{MH}-1)\,(2\,C_{wireless}+C_{fixe}+C_{seach})$$

Avec N_{MH} = nombre de mobiles.

Chaque mobile a 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 donc considérable cela engendre d'une part l'épuisement de la batterie du mobile émetteur et celles 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_{seach}) d'acheminement des messages entre les MSS et de localisation des mobiles surcharge le réseau statique. Les instructions exécutées sur le mobile de gestion de la file d'attente consomment aussi de l'énergie. L'accès d'un mobile à la section critique nécessite le réveil de tous les mobiles qui peuvent opérer en mode veille pour répondre à sa requête.

L'algorithme de Le Lann [LEL, 77] est basé sur le principe du jeton ou du privilège ; on estime le coût induit par cet algorithme à :

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

On peut observer dans l'application de l'algorithme de Le Lann les mêmes inconvénients induits dans l'algorithme vu précédemment.

Pour remédier à ces inconvénients on se propose d'appliquer le principe des deux- tiers :

IV.2.1. Principe des deux-tiers :

Le principe des deux-tiers [ACH, 94] comme son nom l'indique propose que les deux-tiers d'un algorithme pour l'environnement mobile soient exécutées sur la partie statique du réseau (soit les MSS) [BAD, 94] ; cela permet de décharger les mobiles des deux-tiers du coût d'une application distribuée, et de libérer le médium sans fil de la plupart des messages de contrôle liés à cette application.

IV.2.1.1. L'application du principe des deux-tiers à l'algorithme de Lamport :

L'application du principe des deux-tiers à l'algorithme de Lamport vu précédemment signifie que la file d'attente est distribuée dans toutes les MSS, et elle contient toutes les demandes d'accès à la section critique du réseau envoyées par les mobiles à leur MSS locale par conséquent il n'y aura pas de recherche de mobile pour lui transmettre l'autorisation d'accès, car 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 pour le placer dans la file. 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

Actions exécutées par une MSS M

Var f : file d'attente de (identite-mobile, horloge, servi : booléen) init (;

/*ordonnée selon l'ordre croissant du champs horloge.
Les éléments servis à faux sont les moins prioritaires*/.

- A la réception de (DEMANDE, Mhi) de Mhi
  - Enfiler (f, Mhi, false) ;
  - Diffuser (REQUEST, Mhi) aux MSS ;
  - Attendre la réceptionde (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 MSS ;

- 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 MSS ;

- A la réception de (RELEASE, MHj) de M '
  - Défiler (f, MHj) ;

Une MSS M envoie (ACCES, M) à un mobile Mhi si seulement si :

- Mhi est local à M ;
- MHi est à la tête de f ;

- f (Mhi). Servi = true ;

le coût induit par ce nouvel algorithme pour satisfaire une demande est :

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

où :

- 3 $C_{wireless}$: coût d'envoi d'une demande par un mobile + le coût de réception de l'autorisation d'accès + le coût d'envoi 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.

Les avantages relevés dans cet algorithme sont :

- Il décharge les mobiles de l'envoi 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.
- Il décharge des mobiles des (CPU, mémoire, …) de gestion des files d'attente.
- Il diminue le nombre de messages circulant dans le réseau statique car ($N_{MSS} << N_{MH}$).
- Il ne nécessite aucun coût de localisation des mobiles parce que les informations concernant leurs demandes sont dans toutes les MSS.

IV.2.1.2. L'application du principe des deux-tiers à l'algorithme de Le Lann [LEL, 77] :

L'application du principe des deux-tiers sur l'algorithme [LEL, 77] consiste à faire circuler le jeton entre les MSS au lieu des mobiles. Chaque mobile n'a qu'a envoyer une demande à sa MSS courante. Lorsque le jeton atteint cette dernière, elle recherche le mobile pour lui délivrer une autorisation d'accès en section critique. Nous allons illustrer tout cela par l'algorithme suivant :

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 un MSS Mi

Var file-servi, file-demande : file

- A la réception de (DEMANDDE, Mh)
  - Enfiler (file-demande, Mh)

- A la réception du jeton de son prédécesseur Mi-1
  - File-servi := file-demande
  - File-demande := vide
  - Si non (file vide (file-servie)) alors

Début

- Défiler (file-servi, Mh)
- Chercher (Mh)
- Envoyer (ACCORD, M) à Mh

Fin

- A la réception de (LIBERER de Mh)
  - Si non (file vide (file-servie)) alors

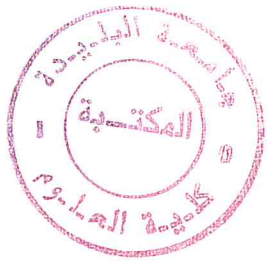
Début

- Défiler (file-servie, Mh)
- Chercher (Mh)
- Envoyer (ACCORD, M) à Mh

Fin

Sinon

- Envoyer (jeton) à son successeur dans l'anneau.



Par conséquent le coût se résume à :

- N_{MSS}. C_{fixe} : Coût pour faire circuler le jeton.
- $N_{demande}$. (3 $C_{wireless}$ + C_{seach} + 2 C_{fixe}) : Coût pour satisfaire une demande.

On peut observer que le coût est réduit à de simples demandes faites par les mobiles, et est lié au nombre de demandes d'accès à la section critique.

IV.2.2. Les problèmes propres à l'environnement mobile :

Bien que le principe des deux-tiers soit un bon outil pour l'adaptation des algorithmes dits classiques à l'environnement mobile, sans pour autant résoudre les problèmes liés à la mobilité proprement dite, tels que les déconnexions, les reconnexions, le hand-off,...etc.

Le problème posé pour l'application faite sur l'algorithme de Lamport se traduit comme suit :

Si un mobile possédant l'autorisation d'accès se déplace vers une autre MSS que celle qui l'a lui a délivré, alors les conditions de service seront satisfaites aussi dans cette nouvelle MSS et par conséquent, une autorisation d'accès lui est envoyée une seconde fois, car la MSS d'accueil ne sait pas que le mobile a déjà reçu l'autorisation d'accès pour les mêmes conditions.

Pour la deuxième application effectuée sur l'algorithme de Le Lann, le problème posé est le problème d'équité dans le cas où un mobile se déplacerait devant le jeton, l'attend dans chaque cellule, accède à la région critique et se déplace vers la prochaine MSS dans l'anneau ; il pourra ainsi, accéder (NMSS-1) fois à la région critique dans un seul tour du jeton, et priver un autre mobile qui lui attendra jusqu'à la satisfaction de (NMSS-1) (NMH-1) demandes avant que le jeton ne lui soit délivré.

V. Proposition d'algorithme en environnement mobile :

La résolution du problème d'exclusion mutuelle en environnement mobile implique la prise en compte d'un ensemble de critères afin d'adapter

l'algorithme à un tel environnement. On peut résumer ces critères dans ce qui suit :

- Minimiser les opérations effectuées sur les mobiles afin de conserver leur énergie.
- Minimiser l'utilisation du médium sans fil vu la limitation de sa bande passante et son coût élevé.
- Permettre aux mobiles de fonctionner dans leurs différents modes.
- Minimiser la charge sur le réseau fixe

De plus, le principe des deux-tiers permet de transférer la charge des mobiles aux stations de base, les quelles sont plus puissantes et dont le coût de communication est relativement faible, de sorte que le mobile se limite à envoyer sa demande et c'est à sa MSS d'accorder l'autorisation d'accès à la section critique à ce mobile où à un autre ; de ce fait les critères du choix ne concernent plus que la partie fixe du réseau, tel que : le nombre de messages échangés, la résistance aux pannes,...etc.

V.1. choix de l'algorithme :

Dans ce qui va suivre nous allons procéder à la critique des différents algorithmes pour en synthétiser le meilleur à choisir.

Les algorithmes basés sur les variables d'état nécessitent un nombre élevé de messages pour assurer l'accès en exclusion mutuelle à la section critique, alors que ceux basés sur la communication des messages sont mieux adaptés à l'environnement mobile. Ces derniers se divisent en deux classes distinctes selon la technique utilisée :

- **Distribution d'une file d'attente :** cette classe englobe les algorithmes qui distribuent une file, de type FIFO, concernant les informations relatives aux différentes demandes dans toutes les MSS, pour permettre une vue globale à chaque site ([LAM, 78], [RIC, 81]). Pour garantir la cohérence de cette file, les 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 MSS, si une MSS tombe en panne les mobiles locaux peuvent se déplacer vers d'autres stations où ils pourront recevoir l'autorisation.

- **Circulation d'un jeton :** cette classe englobe les algorithmes utilisant la technique du jeton (Le Lann [LEL, 77], Suzuki & Kasami [SUZ, 82]). Ces derniers nécessitent un nombre de messages relativement inférieur, mais connaissant un certain nombre de problèmes tels que la perte du jeton et sa régénération. Contrairement à ceux de la classe précédante, 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.

  On distingue deux algorithmes : Le Lann et Suzuki & Kasami. Le premier fait circuler le jeton, lequel représente une information particulière (fixe et connue de toutes les MSS et à tout instant), entre toutes les MSS dans un anneau fixe qui n'est pas reconfiguré qu'en cas de panne d'une station de base. Cette solution pose le problème de nombre infini de messages si aucun mobile ne demande l'accès à la section critique, le jeton continue à circuler dans l'anneau. Ce qui constitue une surcharge inutile pour le réseau fixe. De plus, si les demandes de la section critique sont concentrées au niveau d'un ensemble de MSS, alors que les autres n'en contiennent pas, 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 (Suzuki & Kasami), en accompagnant le jeton d'informations concernant l'évolution des demandes au niveau des différentes stations de base. Cela permet d'éviter l'envoi du jeton aux MSS qui ne le demandent pas.

Cependant, les techniques *inform* et *proxy* [ACH, 94] (qu'on définira plus loin) peuvent diminuer le coût de localisation d'un mobile, lequel est au maximum de $N_{MSS}.C_{fixe}$: coût de diffusion de la requête de recherche et de réception de la réponse. De plus, dans les réseaux mobiles, les stations de base sont souvent dupliquées au sein des mêmes cellules pour tolérer leurs pannes, ce qui rend rare la panne d'une MSS et la considère comme un cas particulier.

Le choix a porté sur l'adaptation de l'algorithme de Suzuki & Kasami pour la résolution du problème de l'accès en exclusion mutuelle à une ressource classique dans l'environnement mobile.

V.1.1. Principe de l'algorithme :

Le jeton circule entre les MSS dans le sens croissant de leur indice, il contient les numéros des dernières requêtes satisfaites sur chaque MSS. Chaque mobile désirant entrer en section critique envoi une et une seule fois sa demande à la MSS qui le gère. Chaque MSS contient une file FIFO des demandes d'accès en section critique. A chaque fois qu'une MSS reçoit sa demande, après le dernier passage du jeton, elle diffuse une requête accompagnée de son horloge (incrémentée juste avant son envoi) à toutes les autres MSS (ne sachant pas où se trouve le jeton) pour réclamer le privilège. Ainsi, cette requête est enregistrée. 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 sur après leur localisation.

Après satisfaction des demandes locales, une MSS Mi met à jour son champ dans le jeton pour annuler sa requête. Elle envoie, ensuite, le jeton à la première de ses voisines Mj, ayant réclamé le jeton, dans l'ordre Mi+1,…,Mn, 1,…Mi-1 avec jeton[j] > requête [j]. L'ordre i+1,…,n, 1 …,i-1 est très important pour éviter la famine des MSS au cas où leurs prédécesseurs demanderaient toujours le jeton. Les demandes qui arrivent, au 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.

V.1.2. L'algorithme proposé [DAH, 98] :

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 ;
Jeton présent, dedans : boolaen init false ;
Requete, jeton : tableau [1..NMSS] de 0..+∞ init 0 ;
File demandes, fileservice : file FIFO de MH init ∅ ;

- A la réception de (DEMANDE_SC, MH) d'un mobile MH
  - Enfiler (filedemande, MH) ;
  - Si ¬jetondemande alors

    Début

      - Jetondemande := true ;
      - Si ¬jetonprésent alors

        Début

          - Hor := Hor+1 ;
          - Diffuser (REQ,Hor,M) à toutes les autres MSS ;

        Fin

        Sinon

        Début

          - Dedans := true ;
          - Jetonprésent := true ;
          - Fileservice := filedemande ;
          - Filedemande := ∅ ;
          - Défiler (fileservice, MH) ;
          - Chercher (MH) ;
          - Envoyer (AUTORISATION, M) à MH ;

        Fin

    Fin

- A la réception du jeton
  - Dedans := true ;
  - Jetonprésent := true ;
  - Fileservice := filedemande ;

- Filedemande :=∅ ;
- 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 (jetonprésent) & (¬dedans) alors

    Début
    - Pour j de M+1..NMSS, 1..M-1 faire

      Début
      - Si (Requete[j]>jeton[j])&(jetonprésent) alors

        Début
        - Jetonprésent := false ;
        - Envoyer(jeton) à j ;
        - Exit ;

        Fin

      Fin

    Fin

- A la réception de (LIBERER, MH) de MH
  - Si ¬(filevide(fileservice)) alors

    Début
    - Défiler(fileservice, MH') ;
    - Chercher(MH') ;
    - Envoyer(AUTORISATION, M) à MH' ;

    Fin

Sinon

Début

- Dedans := false ;
- Jeton[M] := Hor ;
- Pour j de M+1..NMSS, 1..M-1 faire

  Début

  - Si (Requete[j]>jeton [j]& (jetonpresent) alors

    Début

    - Jetonpresent := false ;
    - Envoyer(jeton) à j ;
    - Exit ;

    Fin

  Fin

- Si ¬(filevide(filedemandes)) alors

  Début

  - Si jetonpresent alors

    Début

    - Dedans := true ;
    - Fileservice :=filedemandes ;
    - Filedemandes :=∅ ;
    - Défiler (fileservice, MH) ;
    - Chercher (MH) ;
    - Envoyer (AUTORISATION, M) à MH

    Fin

    Sinon

Début

- Hor++ ;
- Diffuser(REQ,Hor,M) à toutes les MSS ;

Fin

Fin

Sinon jetondemandé= false ;

Fin ;

V.1.3. Propriétés de l'algorithme :

L'exclusion mutuelle est 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 durant sa possession du privilège. Par conséquent, à un instant donné, un seul mobile au plus peut avoir une autorisation d'accès à la section critique.

D'une autre part l'utilisation de l'algorithme proposé ne peut conduire à des cas de famine, car aucune demande n'attend 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 MSS ; et comme le jeton est envoyé à la première MSS suivante dans l'anneau l'ayant demandé, alors toute MSS ayant réclamé le jeton, 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 sans pour cela y avoir un changement pour l'algorithme.

Afin de remédier à la possibilité de croissance des variables Hor, et par conséquent, la croissance des champs des vecteurs requête et jeton ; si une MSS M, qui reçoit le jeton, s'aperçoit que sa variable Hor est devenue assez grande, elle exécute les instructions suivantes :

Jeton[M] := 0

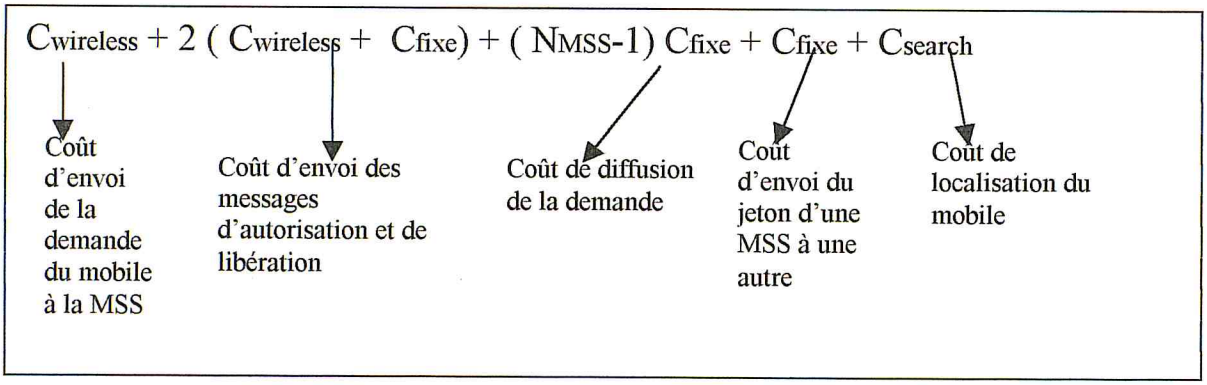
Diffuser (INITREQ, M) ;

Une MSS qui reçoit ce message exécute l'instruction suivante :

Requête [M] := 0 ;

V.1.4. Le coût induit par l'algorithme proposé :

Pour la satisfaction d'une demande l'algorithme induit un coût estimé à :



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

- On peut généraliser ce coût pour K demandes qui sera dans le pire des cas où chaque MSS contient une seule demande au moment de l'arrivé du jeton :

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

- Le coût de satisfaction de K demandes dans la même MSS est estimé à :

$$3\,KC_{wireless} + (2\,K + N_{MSS})\,C_{fixe} + KC_{search}.$$

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

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

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

V.1.5. Comparaison de l'algorithme proposé avec les autres algorithmes :

On a vu précédemment que les coûts de satisfaction d'une demande pour les algorithmes de Lamport et Le lann adaptés à l'environnement mobile étaient respectivement de :

$3\,C_{wireless} + 4\,(N_{MSS})\,C_{fixe}$. Et de $3C_{wireless} + (N_{MSS} + 2)\,C_{fixe}$.

On observe aussi que le coût pour l'algorithme proposé était de :

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

On constate que pour l'algorithme propose on gagne, au pire des cas, (2NMSS – 6) messages dans le réseau fixe, par rapport à celui de Lamport ; ce qui signifie que dès que le réseau contient plus de 3 MSS, l'algorithme proposé devient moins coûteux.

Par contre, par rapport à l'algorithme de Le Lann on perd un coût de (NMSS + 1) Cfixe messages, cela pour éviter le cas où aucun mobile ne demande d'entrer en section critique avec un nombre de messages infini, alors qu'il est nul dans l'algorithme proposé.

V.2. Le problème de perte du jeton :

Comme vue précédemment, les algorithmes distribués adaptés aux environnements mobiles posent des problèmes, et l'un des problèmes les plus virulents est celui de la perte du jeton due à la panne d'une MSS au moment où elle détenait le jeton ou encore à la panne d'un mobile ayant l'autorisation d'accès.

V.2.1. La panne d'une MSS :

A la panne d'une MSS M, le réseau de transport informe les autres MSS en leur délivrant le message (PANNE, M). A la réception de ce dernier, il faut tout de suite déterminer si M détenait le jeton lors de sa panne ; si oui, le jeton est régénéré par celle qui le lui a délivré, et cela pour garder les informations circulant avec le jeton. Pour la définition de 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 définir en même temps si M avait le jeton au moment de sa panne et également définir celle qui a pour rôle de le régénérer. Il est à noter que chaque MSS contient un tableau de booléens PANNE tel que PANNE[i] prend la valeur vrai si la MSSi est en panne.

V.2.1.1. Protocole de détection de panne et de régénération :

Action exécutées par une MSS M

- A la réception de (PANNE, M')
  - Panne [M'] := True ;
  - Si M= PPNEP (M') alors

    Début
    - Si ¬ jeton présent alors
    - Envoyer (ELECTION, version_jeton, M,M) au PSNEP (M) ;

    Fin

- A la réception de (REGENERER)
  - jeton présent := True ;
  - pour j de M+1..NMSS , 1..M-1 faire

    Début
    - si requete [j] > jeton[j] & (jeton présent) alors

      Début
      - jeton présent := False ;
      - envoyer (jeton) à j
      - Exit ;

Fin

Fin

- A la réception de (ELECTION, version jeton', M',M'')
  - Si M= M'' alors /*le jeton est perdu*/

Début

- Si M= M' alors /* M est régénératrice*/

Début

- Jeton présent := true ;
- Pour j de M+1..NMSS, 1..M-1 faire

Début

- Si (requete [j]< jeton[j] & (jeton présent) alors

Début

- Jeton présent := False ;
- Envoyer (jeton) à j ;
- Exit ;

Fin

Fin

Fin

Sinon envoyer (REGENERER) à M' ;

Sinon

- Si ¬ jeton présent alors

Début

- Si version jeton > version jeton' alors
  - Envoyer (ELECTION, version jeton, M,M'') au PSNEP (M)

Sinon

- Envoyer (ELECTION, version jeton', M',M'') au PSNEP (M)

Fin

Fin

La fonction PSNEP (M) premier successeur non en panne (respectivement PPNEP (M) premier prédécesseur non en panne) fournit l'identité de la première MSS successeur (respectivement prédécesseur) de M qui n'est pas en panne et elle est définit par l'algorithme suivant :

Fonction PPNEP (M : MSS)

Début

- J := M-1
- Fin := false
- Tant que ¬ fin faire

Début

- Si ¬ panne [j] alors

Début

- Fin := true
- PPNEP := j

Fin

Fin

Fin

Fonction PSNEP (M : MSS)

Début

- J := M+1 ;
- Fin := false ;
- Tant que ¬ fin faire

Début

- Si ¬ panne [j] alors

Début

- Fin := true ;
- PSNEP := j

Fin

- J := (j+1) mod NMSS

Fin

Fin.

La panne d'une MSS entraîne un blocage de tous les mobiles demandeurs d'accès, faute de réponse. 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'. S'ils existent, elle leur envoie le message (PANNE, M'). A la réception de ce message, un mobile peut se trouver dans l'un des trois cas suivants :

- Il attend dans l'autorisation : alors il doit faire une nouvelle demande,
- Il l'a déjà reçu : alors il doit abandonner la section critique, ses modifications sont annulées et il doit effectuer une nouvelle demande ;
- Il a libéré la section critique et a envoyé le message de libération : alors il ne fait rien.

V.2.2. La panne d'un mobile :

A tout moment, un mobile qui est connecté au réseau peut tomber en panne ou être inaccessible car il a quitté le périmètre de couverture assuré par le réseau.

La panne d'un mobile détenant l'autorisation d'accès à la section critique, provoquera le blocage de l'algorithme avec l'attente infinie de la libération par la MSS ayant le jeton, par conséquent aucun autre mobile ne pourrait accéder à la section critique. Pour corriger et surmonter ce problème, l'autorisation d'accès est accompagnée d'un temps maximum d'attente de la réponse. Au-delà de ce temps, si le mobile n'a toujours pas libéré la ressource, il recevra un message l'informant du retrait du privilège, et cela immédiatement s'il est toujours connecté ou après sa reconnexion.

Dans les conditions normales un mobile ayant reçu l'autorisation d'accès à la section critique ne peut se déconnecter qu'après sa libération.

V.3. La localisation d'un mobile :

Comme nous l'avons déjà vue au paravent, l'application de la technique du jeton nécessite la localisation du mobile pour lui envoyer l'autorisation d'accès à la section critique, car un mobile peut bien déposer une demande d'entrée en

section critique à une MSS et se déplacer vers une autre où il pourra recevoir son autorisation d'accès.

La fonction rechercher (Mh : mobile) que nous allons illustrer dans ce qui suit permet de retourner l'identité de la dernière MSS fréquentée par Mh et l'état du mobile : connecté, déconnecté, panne, ou bien veille. La fonction Rechercher (Mh : mobile) est réalisée à l'aide de trois méthodes (search, inform, proxy [ACH,94]).

V.3.1. Méthode Search :

La méthode search a pour objectif de rechercher le mobile sollicité et cela se fait au niveau de toutes les MSS ; son cout est indépendant des mouvements du mobile après avoir déposé sa demande et il est égal à :

$$N_{MSS} \cdot C_{fixe}$$

Fonction Rechercher (Mhld: mobile)

Début

- Si Mhld ∈ locallist alors

Début

- M' := M
- Etat := Etat (Mhld)

Sinon

- Diffuser (CHERCHER, M, Mhld)
- Attendre la réception de (DECLARE, M', Mhld, Etat) de M'

Fin

Retourner (M', Etat)

Fin .

- A la réception de (CHERCHER, M', Mhld') de M'
  - Si Mlhd' ∈ locallist alors

Envoyer (DECLARE, M, Mhld', Etat(Mh')) à M'.

V.3.2. Méthode Inform :

A tout moment chaque MSS connaît la localité des mobiles déposant une demande à son niveau. Un mobile qui a changé de cellule (hand-off) doit informer la MSS où il a fait sa demande, de sa nouvelle MSS. La méthode inform est intéressante si le nombre de déplacements intercellulaire de chaque mobile est inférieur au nombre total de MSS ($N_{MOVE} < N_{MSS}$).

Fonction Rechercher (Mhld : Mobile)

Début

- Retourner (Localité (Mhld), Etat(Mhld))

Fin

Lors du Hand-off

- A la réception de (DEMANDEACCUEIL, Mhld, M', M'')
  - Si M''≠ Nil alors
    - Envoyer (ilEstLa, Mhld, M, connecté) a M''

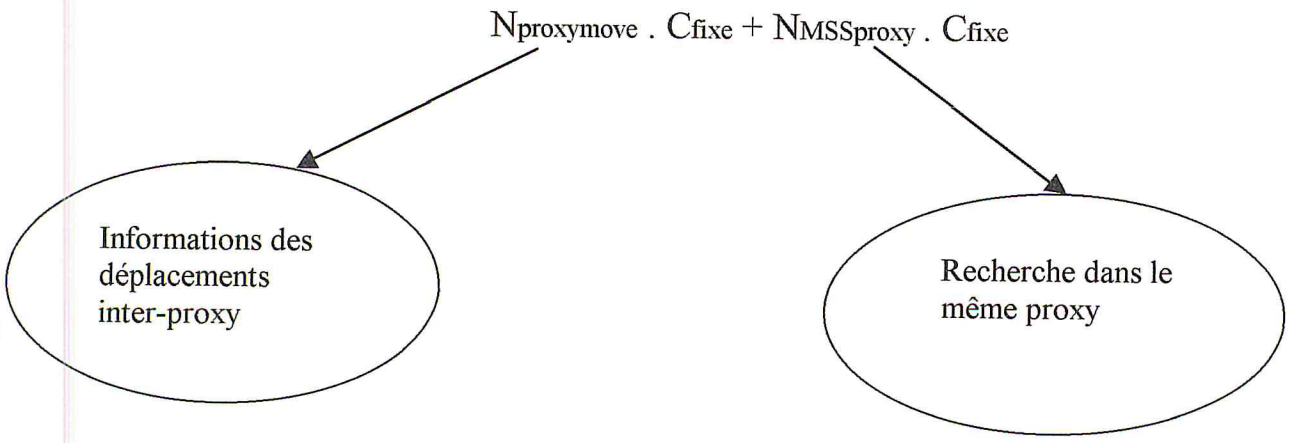
- A la réception de (ilEstLa, Mhld, M', Etat) de M'
  - Localité [Mh] := M'
  - Etat [Mhld] := Etat.

V.3.3. Méthode Proxy :

La méthode proxy s'applique dans le cas où pour chaque mobile on a un nombre de déplacements intercellulaire très inportant, mais limité à un certain ensemble de cellules ; de ce fait on regroupe chacun de ces ensembles pour assurer leur gestion par une MSS principale appelée *proxy*. Le mobile n'informe sa station proxy que s'il s 'agit d'un déplacement inter-proxy, par conséquent il sera recherché au niveau des cellules gérées par ce proxy.

Quand un mobile dépose une demande d'accès en section critique, il le fait dans une MSS M qui en informe sa proxy, et cette dernière gère une variable indiquant la proxy où ce dernier évolue.

La fonction de recherche s'exécute au niveau de la MSS où le mobile a déposé sa demande d'accès en section critique afin de lui transmettre l'autorisation d'accès ; on peut observer le coût induit par la méthode proxy pour un mobile faisant $N_{proxymove}$ déplacemaents inter-proxy de :



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

Lors du Hand-off

- A la réception de DEMANDEACCUEIL(Mhld, M', M'') de Mhld

  /*M' : la MSSoù Mhld se trouve actuellement.

  /*M'' : la MSS où Mhld a déposé sa demande de section critique.

  Si proxy (M)≠ proxy (M') alors

  - Envoyer (CHANGEPROXY, Mhld, Proxy(M)) à proxy(M'')

  Fin

Fonction Rechercher(Mhld : mobile)

Début

- Si Mhld ∈ LocalList Alors

Début

- M' := M
- Etat := Etat(Mhld)

Fin

Sinon

Début

- Envoyer (CHERCHER, M, Mhld) à la MSS proxy (M)
- Attendre la réception de (DECLARE, M', Mhld, Etat) de M'

Fin

- Retourner (M', Etat)

Actions exécutées par une MSS proxy M

- A la réception de (CHERCHER, M', Mhld) de M'
  - Si proxy [Mhld] = M alors

  Début
    - Si Mhld ∈ LocalList alors
      - Envoyer (DECLARE, M, Mhld, Etat (Mhld))

    Sinon
      - Envoyer (CHERCHER, M', Mhld) à toutes les MSS locales

  Fin

  Sinon
    - Envoyer (LOCALISE, M', Mhld) à proxy [Mhld]

- A la réception de (LOCALISE, M', Mhld)
  - Envoyer (CHERCHER, M', Mhld) à toutes les MSS locales

Actions exécutées par toute MSS M

- A la réception de CHERCHER (M4, Mhld) du proxy
  - Si Mhld ∈ LocalList alors
    - Envoyer (DECLARE, M, Mhld, Etat (Mhld)) à M' ;

Le choix de la méthode de localisation se base sur les critères suivant:

La méthode Inform s'avère meilleure que proxy si le nombre de mouvements intercellulaires effectué par un mobile est inférieur au nombre des mouvements inter-proxy additionné au nombre des MSS dans ce proxy :

$$N_{move} \cdot C_{fixe} < N_{proxymove} \cdot C_{fixe} + N_{MSSproxy} \cdot C_{fixe}$$

$$\Rightarrow \quad N_{move} < N_{proxymove} + N_{MSSproxy}$$

$$\Rightarrow \quad N_{move} < N_{proxymove} + N_{MSS} / N_{proxy}$$

on observe aussi que la méthode Search semble meilleure que Proxy dans le cas où le nombre de MSS est inférieur au nombre des mouvements inter-proxy ajouté au nombre des MSS dans ce proxy :

$$N_{MSS} < N_{proxymove} + N_{MSSproxy}$$

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

$$\Rightarrow N_{MSS} < N_{proxymove} \Big/ 1 - (1/ N_{proxy})$$

V.4. Les sites fixes:

Le réseau fixe est un réseau distribué classique, avec un ensemble de MSS. Cela implique qu'il contient des sites fixes qui utilisent déjà les services du réseau. Lors de la conception des algorithmes de l'environnement mobile, il faut prendre en compte ces sites car ils peuvent utiliser les mêmes ressources que les hôtes mobiles. Dans l'algorithme que nous avons proposé, les sites fixes ont été considéré comme des MSS qui ne seront jamais visitées par les mobiles. Ils ne demanderont le jeton que 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 ;

Jetonprésent, dedans : booléen init false ;

Requete, jeton : tableau[1..$N_{MSS+sitesfixes}$] de 0..+∞ init 0 ;

- Pour accéder à la ressource
  - Si ¬ jeton présent alors

Début

- Hor := Hor+1
- Diffuser (REQ, Hor, M) sur tout le réseau fixe
- Attendre la réception du jeton

Fin

- Dedans := true
- Jetonprésent := true

Section critique

- Jeton[M] := Hor
- Dedans := false
- **Section critique**
  - Pour j de M+1,..$N_{MSS+sitesfixes}$, 1..M-1 faire

Début

- Si (Requete[j]> jeton[j]) & (jetonprésent) alors

Début

- Jetonprésent := false
- Envoyer (jeton) à j
- Exit

Fin

Fin

- A la réception de (REQ, Hor', M') de M'
  - Requete[M'] := Max(Hor', Requete[M']
  - Si (jetonprésent) & (¬ Dedans) alors

Début

- Pour j de M+1,..$N_{MSS+sitesfixes}$,1..M-1 faire

Début

- Si (Requete[j]> jeton[j]) & (jetonprésent) alors

Début

- Jetonprésent := false
- Envoyer(jeton) à j
- Exit

Fin

Fin

Fin.

VI. Conclusion :

Ce chapitre a essayé d'englober les différents algorithmes classiques d'exclusion mutuelle avec une adaptation de certains à l'environnement mobile.

Le choix de l'algorithme s'est porté sur la classe des algorithmes qui sont basés sur la communication par messages en utilisant la technique du jeton circulant (Algorithme de Suzuki et Kasami [SUZ,82] pour son efficacité) et cela par rapport aux nouveaux besoins basés essentiellement sur la communication. Dans un premier temps on applique à l'algorithme le principe des deux-tiers et cela pour permettre le transfère de la charge la plus importante sur le réseau fixe,

ensuite on a traité les problèmes liés à la mobilité (déconnexion des mobiles, perte de l'autorisation d'accès à la section critique, localisation des mobiles,…) ou liés à l'algorithme lui-même (perte du jeton et sa régénération).

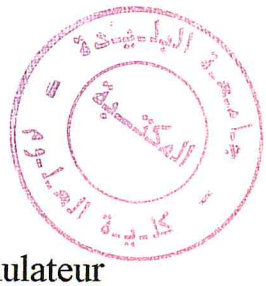
chapitre III

Etude de l'outil de simulation NS-2

I. Introduction :

Dans le domaine de la recherche informatique il ne suffit pas de concevoir et mètre au point des algorithmes sur des bases théoriques mais il faut aussi en avoir la démonstration pour en évaluer les performances et anticiper les problèmes induits par ces applications et comme pour la plupart du temps il s'agit de systèmes complexes cela reviendrait à des coûts faramineux, c'est pour cela que la simulation est considérée comme un outil d'évaluation a moindre coût. Beaucoup de plates formes de simulation sont implémentées pour répondre aux besoins cités plus hauts mais pour ce qui nous concerne nous allons nous attarder sur l'outil *Network simulator* (NS).

Le simulateur réseau NS est un simulateur à événements discrets orienté objet. Au début NS été disponible sous la version 1.0 qui a était développée au sein du laboratoire de lawerence berckley (LBNL) par le groupe de recherche réseau, par la suite le développement de NS a était rattaché au projet VINT sous lequel la version 2.0 est apparue.



I.1. Le projet VINT [ISI, 96] :

Le projet VINT est un projet qui a pour but la construction d'un simulateur réseau qui offre des outils et des méthodes innovatrices dans un environnement aussi proche que possible de la réalité. NS a pour mission de répondre aux questions de mise à échelle (simulation de grandes topologies) et aux problèmes d'hétérogénéité qui se traduit par l'interaction entre protocoles.

Le plus qu'apporte le projet VINT c'est des outils pour la visualisation et l'interprétation car il se fonde sur NS pour le simulateur et NAM (Network Animator, outil d'animation pour visualiser les résultats de simulations et les traces de paquets) pour la visualisation

Le simulateur NS est très bien adapté pour simuler la circulation de paquets dans les réseaux distribués. On fait appel à NS pour tester des algorithmes de file d'attente, des contrôles de congestion, des protocoles de transport, et la mobilité.

II. Description de l'outil Network simulator :

NS est un simulateur à évènements discrets orientés objets , développé en C++ et en OTcl (Objet Tools Command Language) qui est une extension objet du langage interprété Tcl ; NS inclut deux hiérarchies de classes, l'une compilées d'objets écrits en C++ et l'autre interprétées d'objets écrits en Otcl.

Ces deux hiérarchies sont étroitement liées : quand l'utilisateur crée un nouvel objet par interpréteur OTcl, un objet correspondant appelé objet reflet est aussi crée dans la hiérarchie compilée. On dit que ces objets sont des « objets fendus ». Bien entendu, les objets peuvent être accédé aussi bien en OTcl qu'en C++ grâce à la mise en place de procédures d'appel entre OTcl et C++
Au plus bas niveau, il y a six classes qui définissent l'ensemble de la structure du programme et fournissent les méthodes élémentaires. Il s'agit des classes *Tcl*, *TclObject*, *TclClass*, *TclCommand*, *EmbeddedTcl*, *InstVar*. Elles définissent entre autres les méthodes utilisées par C++ pour accéder à l'interpréteur, la hiérarchie, les principales commandes de haut niveau et les méthodes pour accéder aux variables C++ et OTcl.

La simulation est configurée, contrôlée et gérée à l'aide des interfaces fournies par la classe OTcl *Simulator*. Cette classe fournit des procédures pour créer et gérer la topologie, initialiser le format des paquets et choisir le planificateur d'évènements. Elle stocke intérieurement des références à chaque élément de la

topologie. Un script devra donc toujours commencer par l'instanciation d'une variable de cette classe.

L'utilisateur crée ensuite la topologie à travers OTcl en utilisant les classes *node* et *link*, composants essentiels de la topologie. Ces éléments sont décrits dans la sous section suivante.

II.1. Architecture :

La topologie NS-2 est essentiellement composée de nœuds et de liens. La définition des nœuds se fait dans un premier temps à travers l'instance de *Simulator* puis à travers l'instance de la classe *Node*. La fonction d'un nœud est de recevoir des paquets, les examiner et les mapper à ses interfaces sortantes appropriées (voir **Figure III-1**). Cette classe est composée d'un classificateur et de méthodes pour configurer un nœud. Les méthodes proposées sont des fonctions de contrôle, de gestion d'adresse et de port, de gestion d'agents et de repérage des voisins. Le classificateur est la partie du nœud qui traite chaque segment des paquets reçus. Il en existe donc plusieurs, chacun étant spécifique au champ examiné (ex : le classificateur d'adresse est utilisé pour supporter la propagation des paquets).

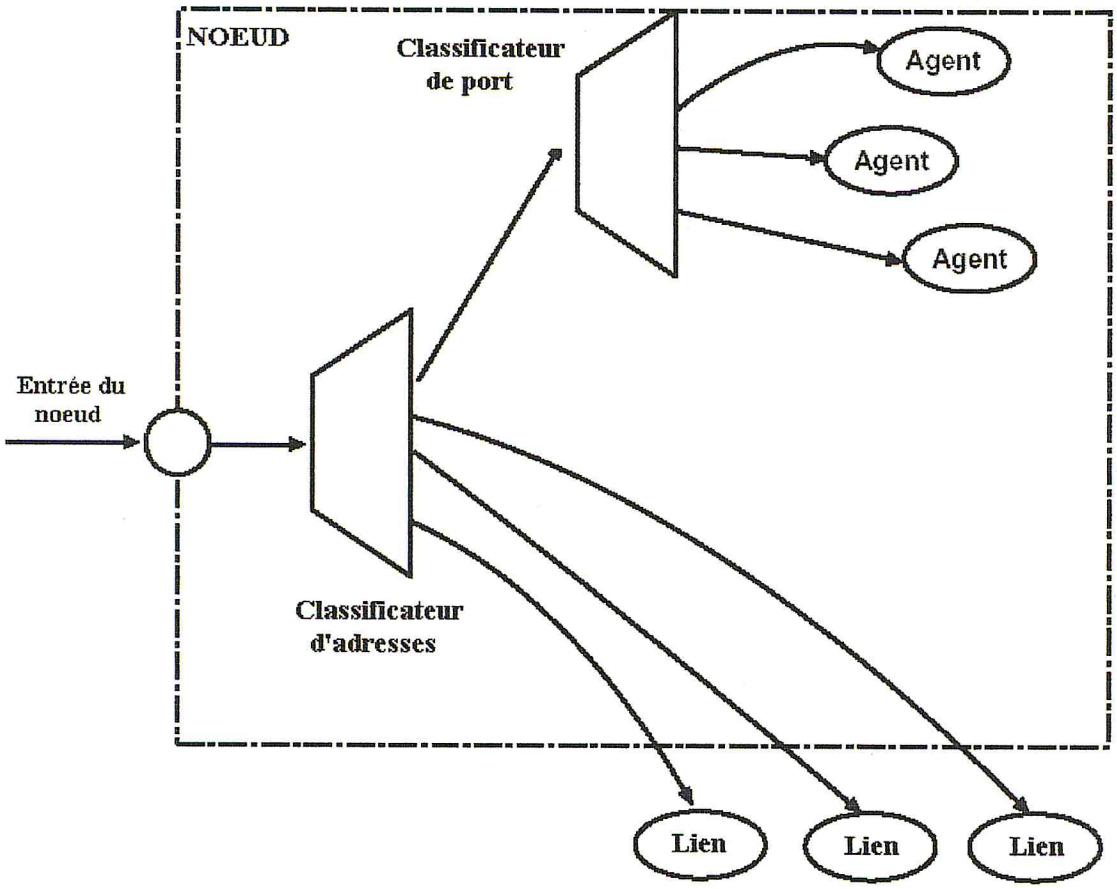
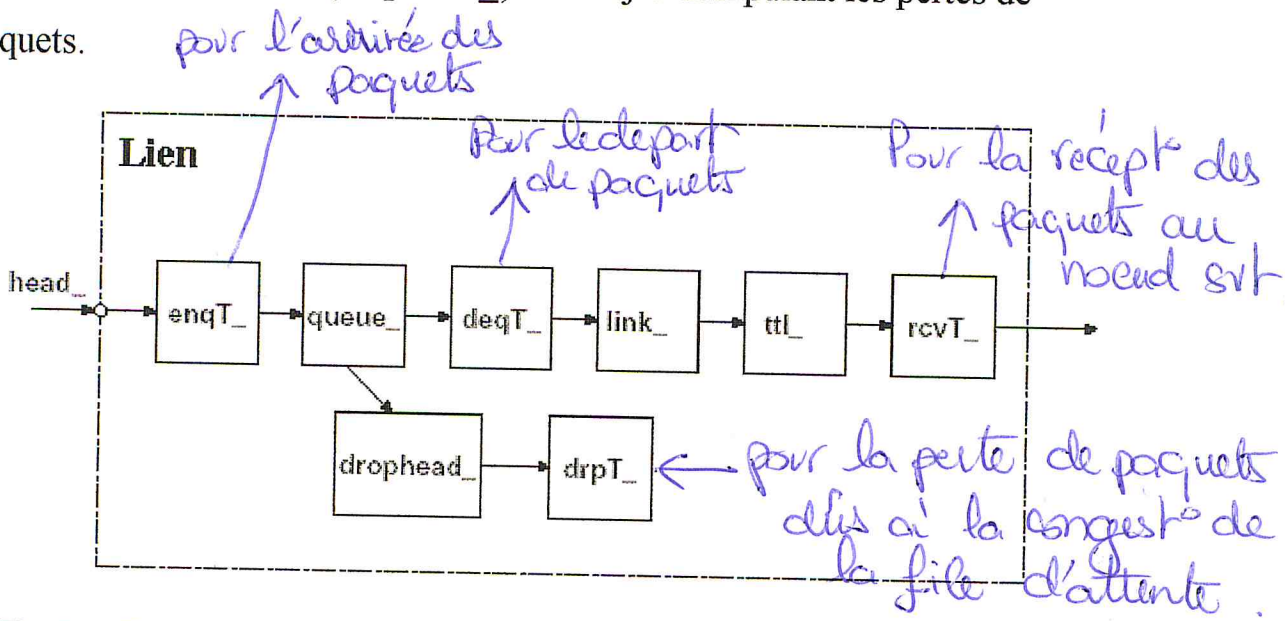


Figure III-1 : Structure d'un nœud unicast

Les liens constituent la deuxième partie de la topologie. Les liens entre les nœuds sont définis dans la classe *Link* et *SimpleLink*, plus précisément lorsqu'il s'agit de relier deux nœuds. Plusieurs types de liaisons sont supportés, comme le point à point, le broadcast ou les liaisons sans fil pour la mobilité. Cette classe définie cinq variables clés (représentées dans la **Figure III-2**)

- La file d'attente de connecteurs (***queue_***).
- La tête de file(***head_***).
- Le lien(***link_***)

- La durée de vie (***ttl_***)
- Elimination de la tête(***drophead_***) : un objet manipulant les pertes de Paquets.



EnqT, DeqT, DrpT et rcvT sont des procédures qui manipulent la file d'attente (enfilement, défilement, destruction et réception).

Figure III-2 : Structure d'un lien [NS 02]

II .1.1. La gestion des files d'attente :

La gestion des files d'attente et la simulation des délais sur les liens sont implémentés dans les classes *Queue* et *LinkDelay* respectivement. Les files d'attente actuellement disponible dans NS sont :

- FIFO
- RED buffer management – Random Early-Detection gateways –
- CBQ (priorité et circulaire) – Class-Based Queuing –
- Plusieurs variantes de file d'attente juste (Fair Queue) – FQ et Stochastic FQ –
- DRR (Deficit Round Robin scheduling).

Pour simuler un quelconque délai dans la réception ou l'émission d'un paquet, la file d'attente correspondante est simplement bloquée.

II .1.2. Les agents :

A un niveau supérieur, on retrouve les agents (*classe Agent*) qui jouent un rôle important dans les simulations. Les utilisateurs créent de nouvelles sources ou récepteurs à partir de la classe *Agent*. Ils font partie intégrante d'un nœud et sont les points terminaux vis à vis des paquets couche réseau ; leur rôle est de générer et réceptionner des paquets suivant les protocoles de transport (TCP, UDP, SRM « Sacalable Reliable Multicast »,...). La **Figure III-1** montre les interactions entre ces différents composants dans NS.

II .1.3. Les applications :

La génération de trafic dans NS peut se faire de deux manières différentes qui sont décrites dans la classe *Application*. Il est possible de générer des paquets par un générateur de trafic (classe *TrafficGenerator*) ou par la simulation d'applications existantes (classe prenant le nom de l'application), toutes ces classes étant dérivées dans la classe *Application*. Les générateurs de trafic peuvent être de quatre types :

- Classe *Exponential* : génère un trafic ON/OFF à intervalle de temps régulier.
- Classe *Pareto* : génère un trafic ON/OFF à intervalle de temps aléatoire.
- Classe *CBQ* : débit de bit constant.
- Classe *Trace* : permet de lire la génération de trafic dans un fichier.

Remarque : un nœud qui génère du trafic ON/OFF est un nœud qui alternativement émet des paquets et stoppe son émission.

Tous ces générateurs de trafic peuvent être associés au protocole de transport UDP.

Actuellement, seules les applications existantes FTP, Telnet et récemment HTTP sont disponibles dans NS pour simuler des applications TCP. Le schéma entre ces deux méthodes de génération de trafic est présenté dans la **Figure III-3**

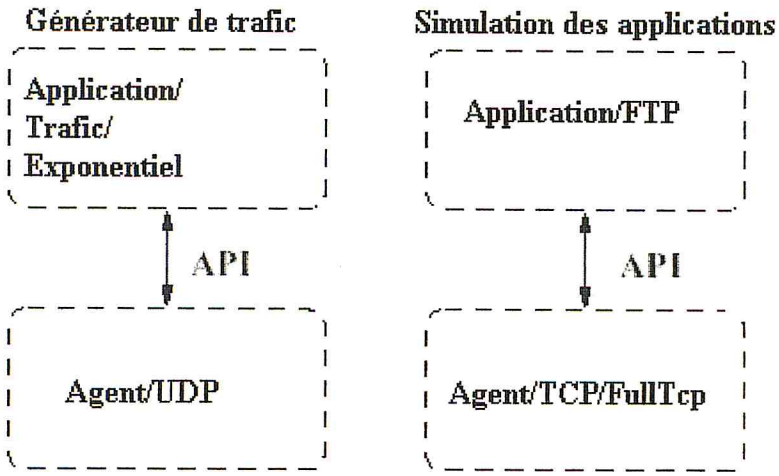


Figure III-3 : Exemple de composition d'une application

II .2. Support pour Environnements mobiles :

La simulation de réseaux mobiles dans Ns 2 est basée sur le modèle du *CMU monarch's wireless* [NS 02]. L'apport de la mobilité dans ce modèle, passe par l'ajout d'un nouveau type de nœuds, mobiles et stations de base définis dans la classe *MobileNode*. Les caractéristiques de la mobilité telles que le mouvement des nœuds, les mises à jour de localisation ou les limites de la topologie sont implémentées en C++. Par contre, les composants réseaux comme le nœud mobile lui-même (classificateur, couche liaison…) sont implémentés en OTcl.
Ce modèle est caractérisé par :

- Un réseau divisé en domaines.

- La division en domaines est assurée par l'adressage hiérarchique à trois niveaux.
- Les sites mobiles sont au départ affiliés à une station de base (*homeAgent)*.
- La station de base joue le rôle de passerelle entre les nœuds mobiles et les nœuds fixes.
- Le transfert des données entre les mobiles et le réseau fixe est :
    - Direct du mobile aux sites fixes.
    - Indirect (les paquets passent par la station de base) du site fixe au mobile.
- La mobilité dans ce modèle est gérée par le protocole MobileIP.

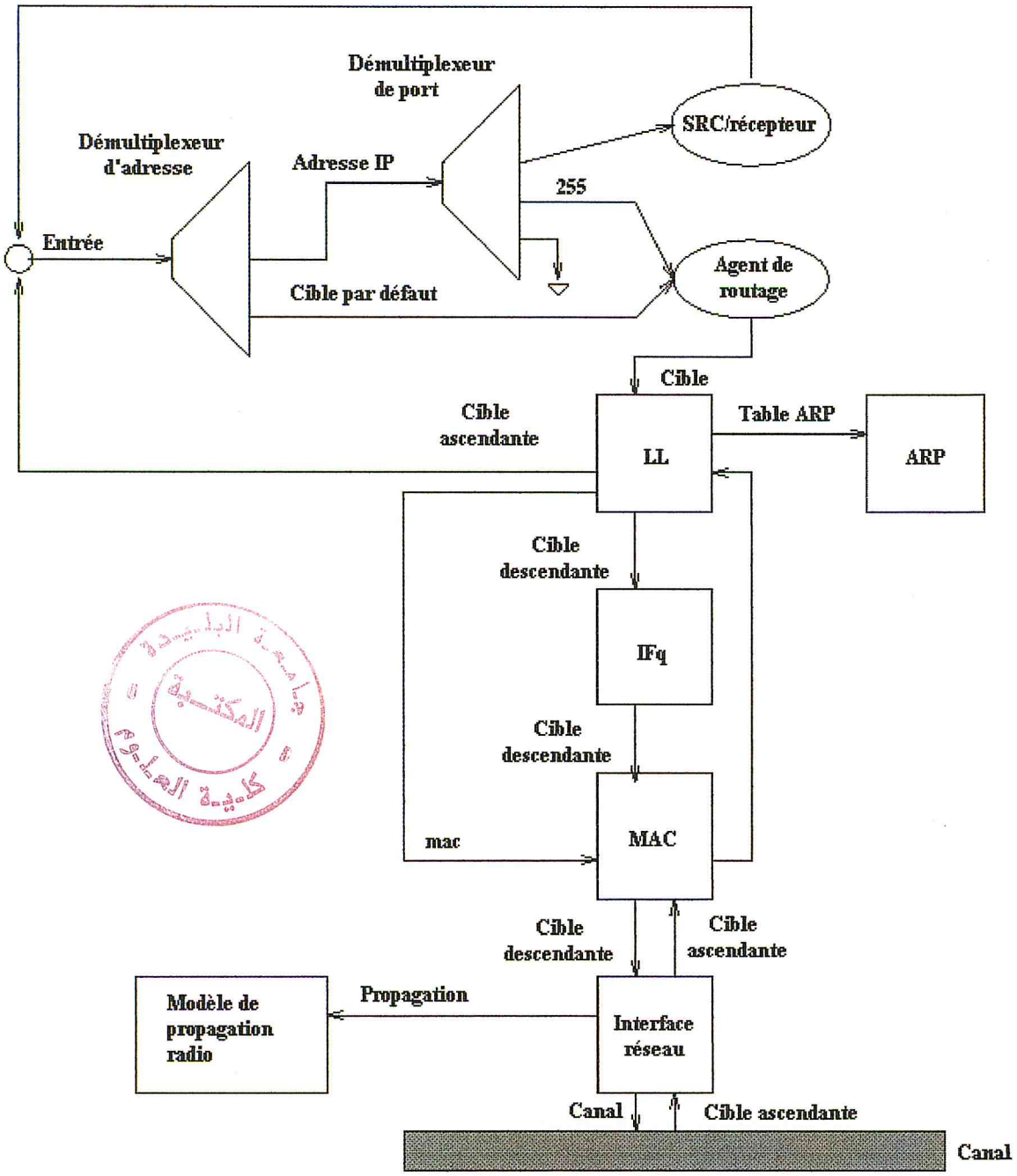


Figure III-4 : Schéma d'un nœud mobile sous les extensions du *CMU monarch's wireless* de NS-2 [NS 02]

Lorsqu'un nœud mobile est crée dans une simulation, le simulateur crée un objet *MobileNode*, un agent de routage et une pile réseau. Ensuite ces composants sont interconnectés et la pile est connectée au canal. Ces composants sont illustrés dans la **Figure III-4**.

La simulation des environnements mobiles avec infrastructure passe la définition d'un autre nœud qui gère la mobilité intra-domaine. Ce nœud est décrit dans la **Figure III-5**.

La gestion de la mobilité dans Ns 2 est assurée par le protocole Mobile IP. Le scénario de Mobile IP (explicité dans l'annexe) consiste en des agents mère, des agents visités et des hôtes mobiles qui se déplacent de l'un à l'autre. Les agents mères et les agents visités sont grossièrement des stations de base. Ils sont définis dans *MobileNode/MIPBS*. Ils contiennent un agent d'enregistrement qui envoie les beacons (c'est des messages de synchronisation envoyés périodiquement par les points d'accès, contenant des informations pour l'identification) et effectue l'encapsulation et la décapsulation des paquets. Leur structure est décrite dans la **Figure III-5**. L'hôte mobile tel que défini dans *MobileNode/MIPMH* a aussi un agent d'enregistrement qui réceptionne et émet des beacons. Leur structure est la même que celle décrite dans la **Figure III-5**, sans l'encapsulateur et le décapsulateur.

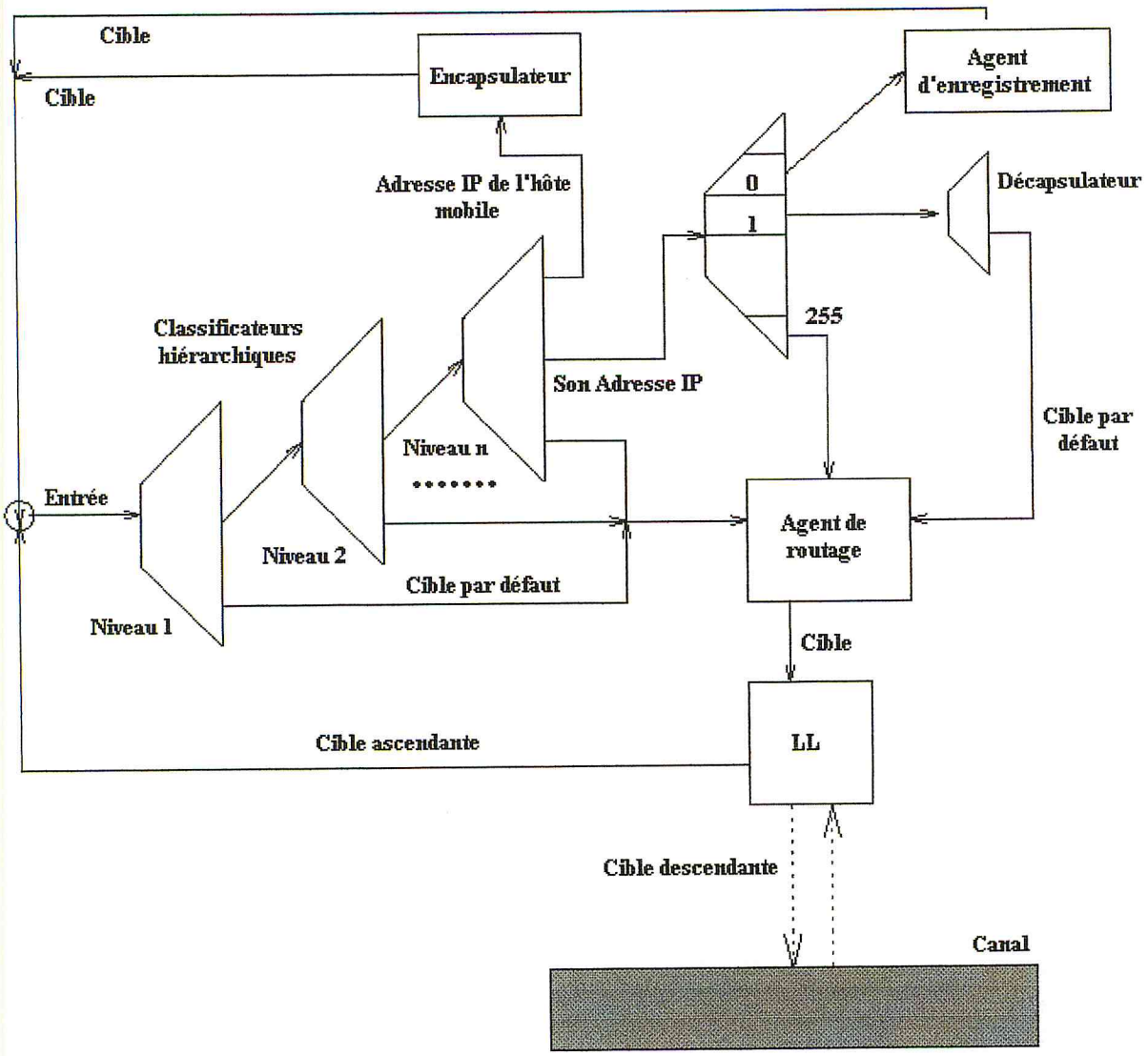


Figure III-5 : Structure d'un nœud *Station de base* pour MIP dans le modèle du CMU Monarch

III . Supports d'analyse de NS-2 :

NS-2 fournit plusieurs types de support pour analyser les résultats d'une simulation. D'une part, NS-2 inclus des classes pour suivre à la trace les fluctuations des paquets, pour calculer et enregistrer diverses statistiques sur

l'ensemble des paquets ou uniquement pour un certain flux. D'autre part, NS-2 travaille de paire avec l'outil de visualisation NAM qui permet de visualiser l'ensemble de la topologie dans une fenêtre graphique.

NS-2 propose deux types de monitoring :

- Traceur (*Trace*) : enregistre chaque paquet (arrivée, départ ou suppression) sur un lien ou dans une file d'attente. Ces objets sont configurés dans la simulation comme des nœuds dans la topologie de réseau. Ils sont définis dans plusieurs sous-classes pour des évènements précis.
- Moniteur (*Monitor*) : enregistre le décompte de différentes quantités comme le nombre d'arrivée de paquets, nombre de bit… Il traque ainsi la dynamique des paquets dans une file d'attente en faisant des moyennes. Quand un paquet arrive ou quitte un lien, il est passé à un objet spécial qui contient une référence sur l'objet *QueueMonitor*. Le contrôle basé sur un certain flux de paquets et non sur l'ensemble, fonctionne avec des classes spécialisées.

L'inspection et le mapping du flux est réalisé par un objet classificateur. Le paquet passe ensuite par le moniteur de flux qui enregistre les états pour chaque flux.

III . 1. Network Animator (NAM):

NAM [Est 99] est un outil d'animation basé sur Tcl/TK (Tool Command Language/Tool Kit) pour l'observation des traces de paquet. Les données utilisées par NAM peuvent provenir d'un simulateur ou de tests sur des réseaux réels. Il supporte l'affichage de la topologie, l'animation des échanges de paquets et des outils d'inspection de données divers.

NAM interprète un fichier de trace contenant des évènements réseau, indexés par le temps de différentes manières comme présenté dans la **Figure III-6**. Ces évènements sont principalement les arrivées, départs et suppression de paquets, rupture de lien. Pour les simulations de réseau sans fil, la localisation et les mouvements des nœuds s'ajoutent aux évènements interprétés

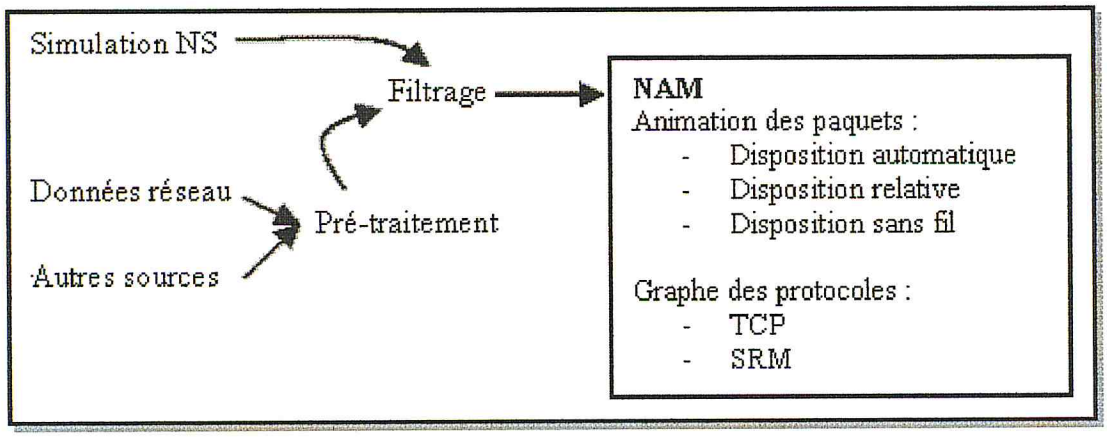


Figure III-6 : Diagramme de NAM

L'animation de paquets est représentée **Figure III-7 :**

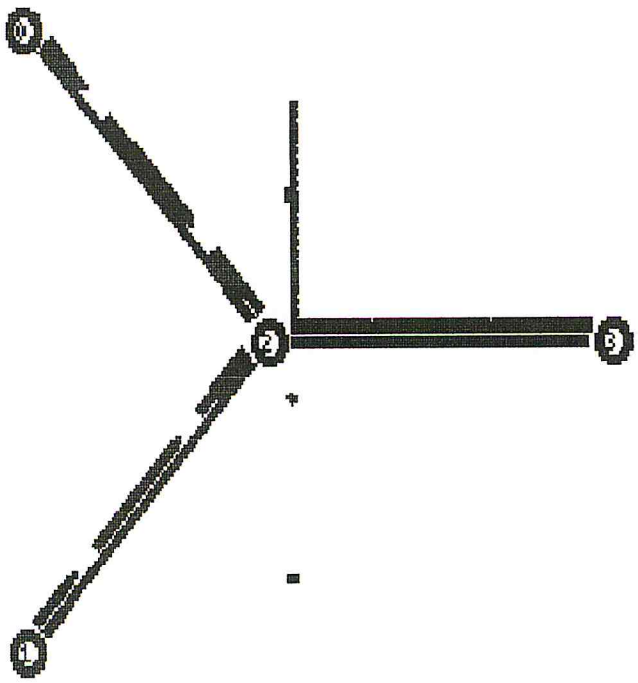


Figure III-7 : animation de paquets dans NAM

Remarque : Notons seulement que la visualisation avec Nam pourra ne pas refléter la simulation de NS (le positionnement des nœuds est exemples).

III .1.1. NAM et la mobilité:

Les nœuds mobiles apparaissent comme les autres nœuds à la différence prés qu'ils ne sont pas reliés entre eux. A l'heure actuelle, les paquets échangés entre nœuds mobiles sont représentés par des petits points lorsque les mobiles sont à portée l'un de l'autre. L'émission des beacons sur l'interface radio est simulée par l'apparition périodique de cercle autour du point d'accès.
Dans la figure suivante, on veut simuler un déplacement d'un mobile d'une station de base à une autre. La topologie est composée de deux stations de base, d'un nœud fixe et d'un mobile.

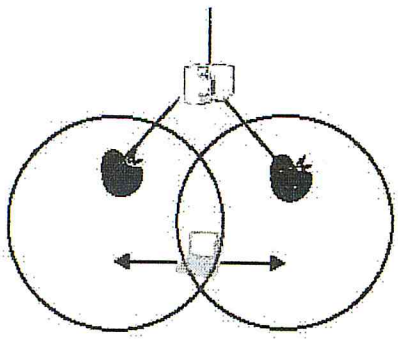


Figure III-8-a : Simulation souhaitée

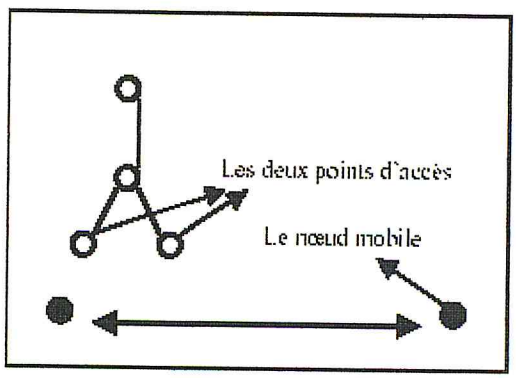


Figure III-8-b : Simulation observée

IV . Conclusion :

Cette partie ne constitue qu'une brève présentation de l'outil de simulation de réseau NS-2. d'autres fonctionnalités offertes comme les liaisons satellites, des techniques d'émulation pour utiliser un réseau réel, n'ont pas été exposées. NS-2 est un produit universitaire en cours de développement et non un produit commercial ; c'est pourquoi il comporte quelques erreurs aussi bien à la

compilation (les versions de Linux pourront en être une cause) que lors des tests spécifiques. Il existe une mailing liste ou on trouve les problèmes des utilisateurs et souvent avec leurs solutions.

La rapide évolution des versions s'avère être à la fois un inconvénient et une qualité. Un inconvénient car toutes ces extensions ajoutées par des groupes de chercheurs ne tournent pas forcément sur les mêmes versions. L'utilisation de ces extensions demande donc beaucoup de patience puisque chaque version a ses problèmes d'installation. Une qualité, car NS-2 devient rapidement de plus en plus robuste, complet et son domaine d'application augmente.

chapitre IV

Mise en oeuvre

I . Introduction :

Après étude et compréhension de l'algorithme proposé d'exclusion mutuelle adapté à l'environnement mobile ainsi que l'outil de simulation NS-2 ; ce qui va suivre expose le travail accomplis du point de vue conception.

II . Topologie de la simulation :

La figure IV-1 résume la topologie de la simulation a accomplir.

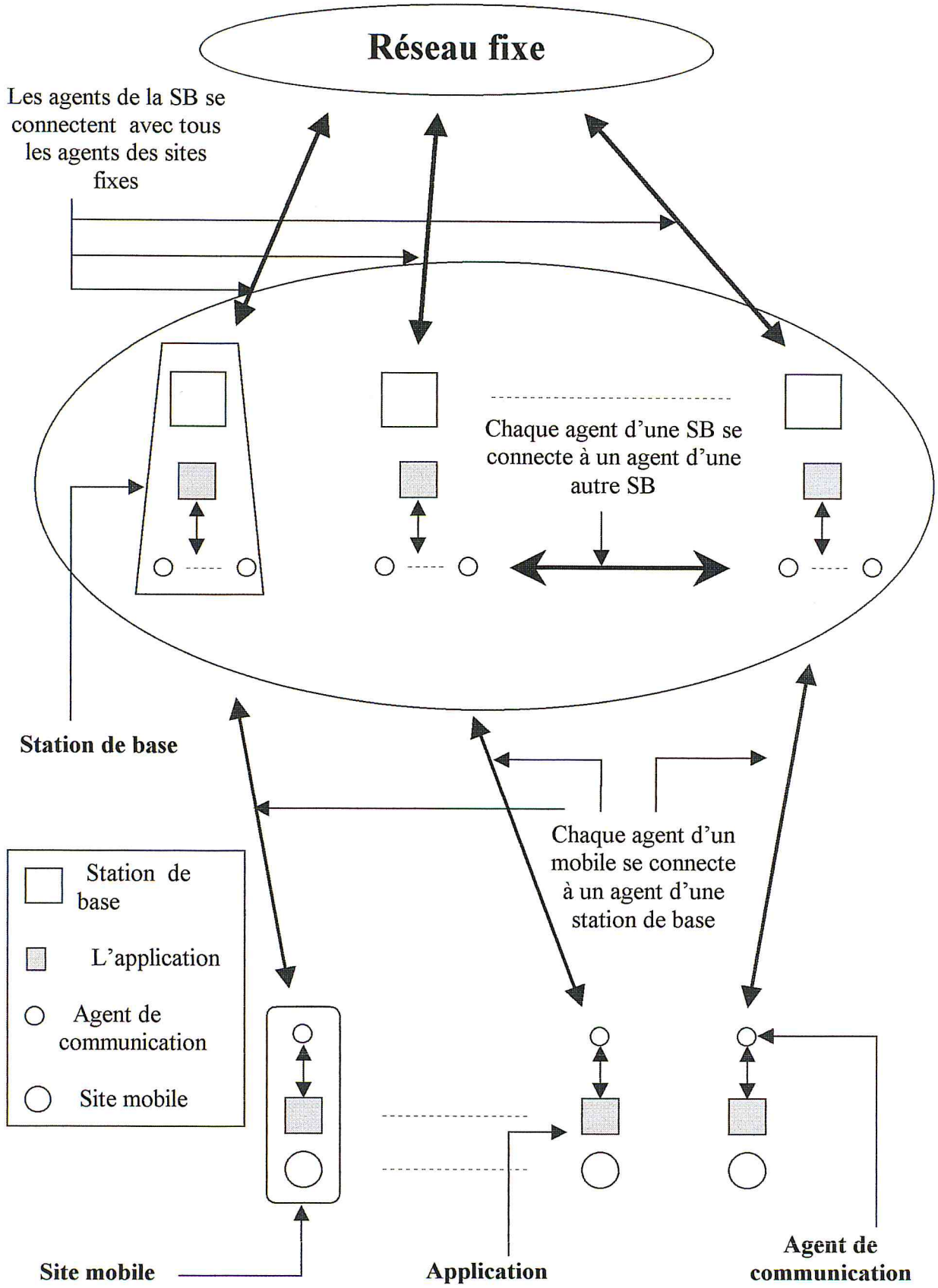


Figure IV-1 : Topologie de la simulation

III . Mise en œuvre :

III .1. Fonctionnement du programme :

Dans le programme proposé on s'inspire de l'algorithme proposé et l'algorithme de Lamport.

Le programme fonctionne principalement suivant les techniques suivantes :

- L'utilisation d'une file FIFO distribuée (principe de Lamport) contenant les stations mobiles.
- L'utilisation du jeton pour pouvoir accéder à la section critique.
- L'utilisation d'une pile contenant les stations mobiles par ordre croissant de leurs horloges.

III .1.1. Principe :

Une station mobile voulant accéder à la SC commence par envoyer une demande à la MSS qui la gère, celle-ci (la station de base) lance une recherche au niveau des autres MSS pour chercher la station mobile ayant l'horloge maximum quand elle l'aura trouvée h= contiendra la valeur de l'horloge de la station trouvée, on va empiler cette station dans une pile, donc la pile contiendra les stations mobiles voulant accéder à la SC et ils seront classés par ordre croissant de leurs horloges.

Si une station mobile arrive au sommet de la pile et aura l'horloge maximum donc il lui faut attendre l'envoi du jeton à sa MSS pour qu'elle puisse accéder à la section critique.

Le programme proposé garantit l'accés à la section critique ; il décharge le mobile de l'envoi et de la réception de (6 (N_{MH} –1) – 3) messages via la liaison sans fil ce qui ce traduit par une diminutions de l'énergie consommée sur le mobile et libération du medium sans fil.

III .1.2. Calcul du coût induit par le programme proposé :

On peut estimer le coût induit par le programme proposé comme suit :

$$C_{wireless} + 2(C_{wireless} + C_{fixe}) + 4(N_{MSS}-1)C_{fixe} + C_{fixe} + C_{search}.$$

- $C_{wireless}$: Coût d'envoi de la demande du mobile à la MSS
- $2(C_{wireless} + C_{fixe})$: Coût d'envoi des messages d'autorisation et de libération
- $4(N_{MSS}-1)C_{fixe}$: Coût de diffusion de la demande
- C_{fixe} : Coût d'envoi du jeton d'une MSS à une autre
- C_{search} : Coût de localisation du mobile

$$C = 3C_{wireless} + (4N_{MSS} - 1)C_{fixe} + C_{search}.$$

III .2. Définition des classes :

Pour notre application nous avons définit 3 classes :

III .2.1. La classe file d'attente :

La classe file d'attente est une classe qui implémente une file d'attente pour la gestion des stations mobiles.

L'entête de cette classe est décrit comme suit :

```
Class file_enreg
{ public :
enreg*e;
int I;

file_enreg();
~file_enreg();
void push (MHI*pmhi,double phorloge,int pservi);
void pop (MHI* mhi);
enreg* get (MHI* mhi);
};
```

- **Description de la classe :**

1. enreg*e : la tête de la liste.
2. i : l'itérateur.
3. file_enreg() : constructeur dans la file.
4. ~file_enreg() : destructeur dans la file.
5. push() : enfiler.
6. pop() : défiler.
7. get() : recherche d'une station mobile dans la file et elle retourne « NULL» si inexistant.

III .2.2. La classe station mobile MHI :

La classe station mobile gère les stations mobiles et son entête est définit comme suit :

```
Class MHI : public mobile node
{ public :
 inline MHI() : mobile node(){} ;
inline~~MHI(){} ;
virtual int command (int argc,const char*const argv)
void send_msg (int message,MSS*mss_dest);
void rec_msg (int message,MSS*mss_sender);
protected
MSS*current_mss;
};
```

- **Description de la classe :**

1. class MHI : public mobile node : déclaration de la classe MHI comme « Mobile Node » qui est une classe propre à NS-2 et cela induit que la classe MHI hérite des fonctionnalités de la classe mobile node.
2. inline MHI() : mobile node(){} : constructeur de nœud mobiles.
3. inline ~MHI(){} : destructeur de hôtes mobiles.
4. virtual int command() : lien avec NS-2 pour interprétation des commandes .
5. send_msg() : fonction d'émission des messages.
6. rec_msg() : fonction de notification à la réception des messages.
7. MSS* current_mss : donne la station de base courante.

III .2.3. La classe station de base MSS :

La classe station de base gère les stations de bases et son entête est décrit comme suit :

```
Class MSS : public node
{ public:
file_enreg*file_attente;
MHI*file_demande [MAX_MHI_MSS] ;
MHI*file_service [MAX_MHI_MSS] ;
Int c_demande, c_service ;
Int hor ;
Int jeton_présent,dedant
Int requet [MAX_MHI_MSS],jeton [MAX_MHI_MSS]
Inline MSS() : node()
{ file_attente= new file_enreg;
c_demande= 0;
c_service= 0;
hor= 0 ;
int i ;
for (i= 0 ;i<MAX_MHI_MSS ;i++) requete [i]= 0;
for (i= 0 ;i<MAX_MHI_MSS ;i++) jeton [i]= 0;
}
inline~MSS(){ delete file_attente;}
virtual int command( int argc,const char*const*argv);
void send_msg_mss( int message,MSS*mss_dest,MHI*mhi_dest);
void send_msg_mhi ( int message,MHI*mhi _dest);
void rec_msg_mss( int message,MSS*mss_sender,MHI*mhi_sender);
void rec_msg_mhi( int message,MHI*mhi_sender);
double get max();
MHI*PPNEP (MHI*mhi);
MHI*PSNEP (MHI*mhi);
```

- **Description de la classe :**

1. class MSS : public node : déclaration de la classe MSS comme « node » qui est une classe propre à NS-2 et cela induit que la classe MSS hérite des fonctionnalités de la classe node.
2. file_enreg*file_attente : file d'attente pour la gestion des stations mobiles.
3. MHI*file_demande[] : tableau des demandes.
4. MHI*file_service [] : tableau des services.
5. c_demande : demande actuelle.

6. c_service : service actuel.
7. hor : l'horloge.
8. jeton_présent, dedant : variables booléennes propre au fonctionnement de l'algorithme.
9. inline MSS() :node() : constructeur d'initialisation des variables.
10. inline~MSS(){} : destructeur et netoyage de la mémoire utilisée.
11. virtual int command() : lien avec NS pour implémentation des commandes.
12. send_msg_mss() : emteur de messages vers une station de base.
13. send_msg_mhi() : emteur de messages vers une station mobile.
14. rec_msg_mss() : notificateur de messages d'une station de base.
15. rec_msg_mhi() : notificateur de messages d'une station mobile.
16. get max() : fonction donnant le temps le plus grand.
17. MHI*PPNEP() : fonction qui donne la MH précedente.
18. MHI*PSNEP() : fonction qui donne la MH suivante.

III .3. Autres définitions :

Au sein du fichier « Mhi.h» on dispose plusieurs définitions utiles à la compilation du programme :

1. **# include « node.h »** : parent de la classe station de base.
2. **# include « mobilenode.h »**: parent de la classe station mobile.
3. **# include « sheduler.h »** : fichier à inclure pour le lien avec le noyau de simulation network simulator.
4. **#define MAX_MHI_MSS.50**: définit le maximum de stations mobiles gérées à la fois par une station de base : Arbitraire.

5. **#define DJ**_DEMANDE
   #define DJ_ LIBRE
   #define DJ_ACCESS
   #define DJ_JETON
   #define DJ_AUTHORISATION
   #define DJ_REQ
   #define DJ_REP
   #define DJ_SERVIR
   #define DJ_PANNE
   #define DJ_ELECTION
   #define DJ_ RELEASE

Définit les messages éventuels entre les stations mobiles et les stations de base

6. **sheduler ::instance(). Clock** : fournit le temps actuel.

7. ***Typedef struct tag_enreg***
   { public :
   MHI*mhi;
   Double horloge;
   Int servi ;
   } enreg ;

Structure de données pour la pile stockant les informations concernant les stations mobiles gérées.

III .4. Intégration des modules dans NS-2 :

L'ajout des modules définit auparavant est régi par une procédure spécifique de NS-2, en effet chaque module intégré doit être dérivé des modules existants dans NS.

CONCLUSION GENERALE :

Pour rendre des applications accessibles n'importe où dans le monde avec des interfaces similaires, il sera nécessaire de procéder à une normalisation de l'informatique mobile. Des travaux de recherche sont actuellement en cours, mais il n'existe pas encore de plate-forme unifiée qui permettrait de gérer les différents types de mobilités et autoriserait des accès aux divers services fournis.

La conception des algorithmes pour les systèmes distribués classique était basé sur la supposition que la localité de l'ensemble des hôtes du réseau est fixe et universellement connue, et que la connexion entre eux est fiable en l'absence de pannes.

L'avènement des hôtes mobiles invalide cette supposition, introduit de nouveaux facteurs spécifiques aux mobiles et à la communication sans fil, et rend nécessaire la révision des algorithmes des systèmes répartis classiques pour les adapter à l'informatique mobile ; parmi ces algorithmes, nous nous sommes intéressés aux problèmes d'exclusion mutuelle pour permettre aux mobiles de bénéficier de tous les services classiques des systèmes distribués.

Tout d'abord, nous avons présenté l'environnement mobile avec ses caractéristiques et ses problèmes, en constatant que les modes de fonctionnement d'une station mobile diffèrent de ceux d'une station fixe, et que la communication sans fil est sujette à une déconnexion. dans notre travail nous avons opté pour l'adaptation d'un algorithme basé sur la communication par

messages en utilisant la technique du jeton [SUZ,82] et lamport [LAM,78] auquel nous avons appliqué le principe des deux- tiers [ARC,94] afin de décharger les mobiles des opérations et traitements gaspillant de l'énergie en faisant circuler le jeton entre les stations de base minimisant ainsi l'utilisation du médium sans fil pour sa bande passante limitée, ensuite, ont été traités les problèmes liés à la mobilité tel que l'autorisation d'entrer en section critique ou liés à l'algorithme lui même tel que la perte du jeton et sa régénération.

Annexes

I. Mobile IP :

Les déplacements des mobiles nécessitent des changements de configuration périodiques effectués par des stations fixes. Ces nouvelles configurations doivent être gérées par i*un protocole spécifique, l'un des protocoles les plus connu est le Mobile IP présentée par Charles Perkins [PER 96], que nous décrirons dans la suite :

Mobile IP, quatre nouveaux acteurs sont définis :

Nœud mobile : équipement IP qui est capable de se déplacer sur Internet et implémentant Mobile IP.

Agent mère (Home Agent) : routeur d'accès avec une interface sur le même lien que le nœud mobile(dans le réseau mère du mobile)

Agent visité (Foreign Agent) : routeur d'accès avec une interface sur le lien courant du mobile (dans un réseau visité). Cet agent n' existe que dans MIPv4.

Points d'accès : équipement intermédiaire entre le réseau filaire et le nœud mobile qui offre la connexion aux nœuds mobiles qui lui sont rattachés

Mobile IP est conçu pour fonctionner sur des architectures spécifiques, l'architecture de base est décrite dans la figure suivante.

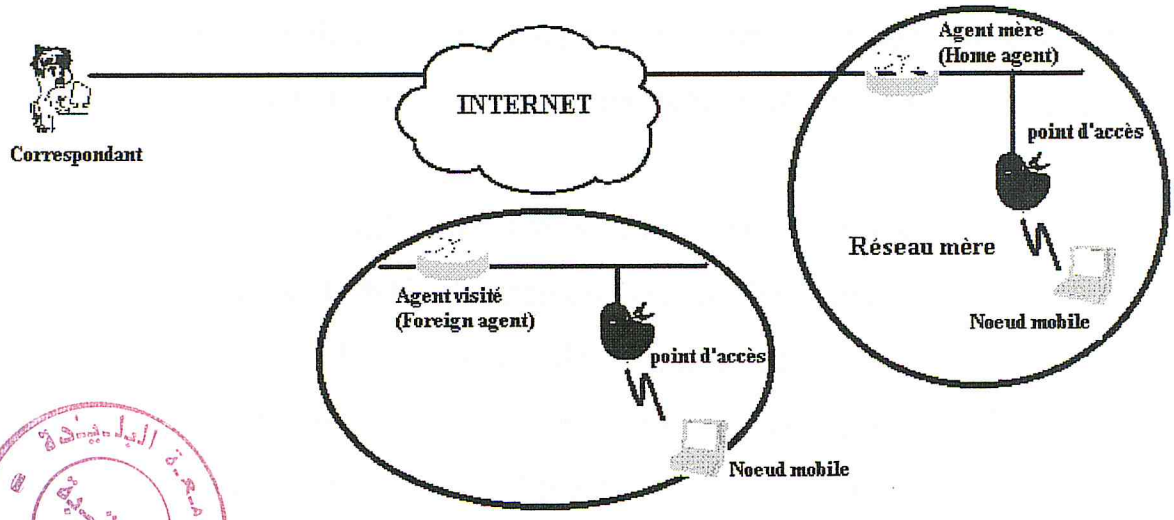


Figure A : Architecture de base pour Mobile IP

- Xgraph xgraph version 12 (Outil de visualisation des statistiques de simulation ; optionnel)
- GT-ITM Georgia Tech Internetwork Topology Modeler (Composant optionnel)
- SGB Stanford GraphBase package (Composant optionnel)
- CWEB CWeb version 1.0 (?) (Composant optionnel)
- ZLib zlib version 1.1.3 (Composant optionnel)

II.1.1. Installation du paquetage :

Pour installer NS tapez la commande « ./install » depuis le répertoire de Ns, ceci configurera et compilera puis installera tous les paquetages présents dans Ns. Un message de fin demandera l'installation des variables de PATH nécessaires pour l'exécution de ns et nam.
Pour plus d'informations voir le site Internet <http://www.isi.edu/nsnam/ns/ns-build.html>

II.2. Point de vue de l'utilisateur :

Dans cette partie ne sera exposée que les commandes principales pour mieux comprendre le fonctionnement du simulateur. Pour une plus ample prise en main de NS-2 se référer au tutorial de Marc Greis http://www.isi.edu/ nsnam/ns/tutorial/.

Comme le simulateur est un interpréteur interactif de scripts, deux possibilités s'offrent à l'utilisateur : soit lancer le simulateur et taper les scripts dans l'interface du simulateur, soit écrire le script dans un fichier « .tcl ».La première solution peut être appréciée au départ pour se familiariser avec les commandes. La seconde vaut pour des simulations de plus grande ampleur et doivent ensuite être exécutées par la commande :

```
ns <fichier>
```

Avant toute action il faut instancier un objet *simulator* qui permettra de crée et gérer tout autre objet. Sa création est faite par :

```
Set ns_  [new simulator]
```

Ensuite, il faut définir la topologie :

1. nœuds :

```
Set n0 [$ns node]
Set n1 [$ns node]
```

On peut créer des nœuds avec une boucle itérative comme suit :

```
For { set i 0 } {Si < 7 } {incr i }  {
   Set n(Si) [$ns node]
}
```

2. liens :

```
$ns duplex-link $n0 $n1 1Mb 10ms DropTail
```

Le lien crée est un lien duplex (communication en double sens), avec une bande passante de 1Mb, un délai de 10ms et une file d'attente « DropTail »(élimination de la queue).

Ensuite il faut définir les transferts possibles entre chaque nœud. Pour cela il faut attacher des agents aux différents nœuds et les connecter ensemble. La création se fait comme suit :

```
Set cbr0  [new agent/CBR]
$ns attach-agent $n0 $cbr0
```

On peut également régler les paramètres des paquets envoyés (taille et période) :

```
$cbr0 set packetsize_ 500
$cbr0 set interval_ 0.005
```

De l'autre coté, il faut créer un agent récepteur sur l'autre nœud et enfin les connecter :

```
Set null0 [new agent/null]
$ns_  attach-agent $n1 $null0
$ns_ connect $cbr0 $null0
```

Pour voir les effets des tests sur les protocoles de transport, on peut attacher un agent UDP ou TCP à un nœud générateur de trafic. Voici la mise en place d'un trafic CBR par un agent UDP :

```
Set udp0 [new Agent/UDP]
$ns_ attach-agent $n0 $udp0
set cbr0 [new Agent/Traffic/CBR]
$cbr0 attach-agent $udp0
$udp0 set packetSize_ 536
```

En admettant qu'un agent « Null » null0 a déjà été attaché au nœud 1, on peut connecter les agents par :

```
ns_ connect $udp0 $null0
```

On définit les évènements du scénario, c'est-à-dire quand le trafic commence, se termine :

```
$ns_ at 0.5 $cbr0 start
$ns_ at 4.5 $cbr0 stop
```

Le lancement de la simulation se fait par la commande :

$ns_ run

II.3. NS pour les réseaux mobiles :

On va l'illustrer par un exemple : avant toute simulation on doit définir les composantes réseau tel : interface queue (IfQ), MAC Layer, canal sans fil …

Définition des options :

```
#
# ======================================================================
#  Définition des options
#
# ======================================================================
set val(chan)           Channel/WirelessChannel  ;#  type du canal
set val(prop)           Propagation/TwoRayGround ;# modele de radio-
propagation
set val(ant)            Antenna/OmniAntenna      ;# type d'antenne
set val(ll)             LL                       ;# Link layer type
set val(ifq)            Queue/DropTail/PriQueue  ;# type d'Interface
queue
set val(ifqlen)         50                       ;# max packet dans ifq
set val(netif)          Phy/WirelessPhy          ;# type interface
réseau
set val(mac)            Mac/802_11               ;# type MAC
set val(rp)             DSDV                     ;# protocole routage
ad -hoc
set val(nn)             2                      ;# number de mobilenodes
```

II.3.1 Configuration des nœuds :

```
# Configuration des nodes
        $ns_ node-config -adhocRouting $val(rp) \
                         -llType $val(ll) \
                         -macType $val(mac) \
                         -ifqType $val(ifq) \
                         -ifqLen $val(ifqlen) \
                         -antType $val(ant) \
                         -propType $val(prop) \
                         -phyType $val(netif) \
                         -topoInstance $topo \
                         -channelType $val(chan) \
                         -agentTrace ON \
                         -routerTrace ON \
                         -macTrace OFF \
                         -movementTrace OFF
```

II.3.2. Positionnement des noeuds :

```
$node_(0) set X_ 5.0
$node_(0) set Y_ 2.0
$node_(0) set Z_ 0.0
```

Le nœud peut bouger :

```
$ns_ at 10.0 "$node_(0) setdest 20.0 18.0 1.0"
```

II.3.3. Wired-cum-wireless:

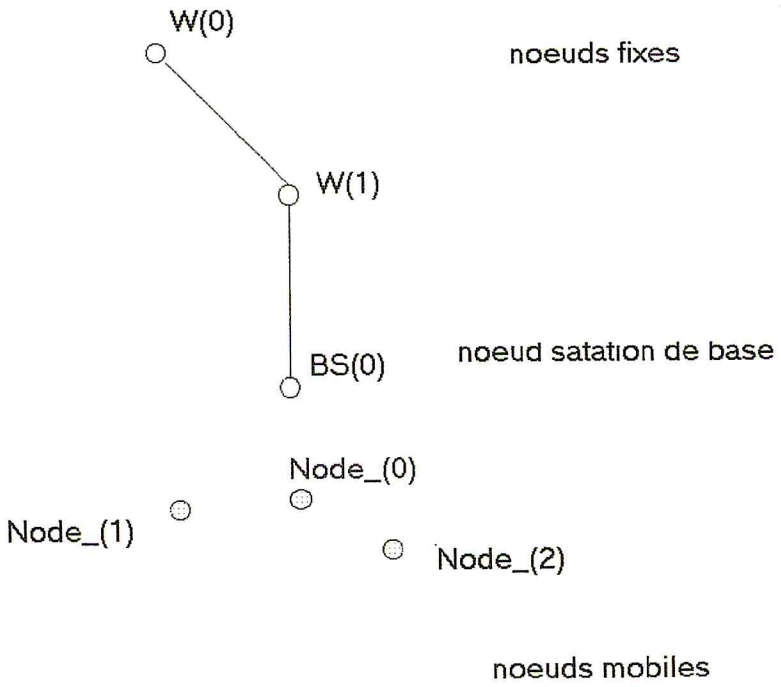
Dans cette section on essaie de montrer comment créer une topologie mixte de domaine fixe et sans fil.

Pour mixer la simulation on doit utiliser l'adressage hiérarchique pour router les paquets entre le domaine fixe et mobile.

```
$ns_ node-config -addressType hierarchical
AddrParams set domain_num_ 2            ;# nombre de domaine
lappend cluster_num 2 1                 ;# nombre de clusters dans
chaque
                                        ;#domaine
AddrParams set cluster_num_ $cluster_num
lappend eilastlevel 1 1 4               ;# nombre de noeuds dans
chaque cluster
AddrParams set nodes_num_ $eilastlevel ;# pour chaque
                                        ;# domaine
```

Dans les lignes de codes ci-dessus : premièrement on configure les nœuds pour avoir le type d'adressage hiérarchique. Le nombre de domaine est 2 (un pour le nœud fixe et l'autre pour la station de base), le nombre de clusters pour chaque domaine et « 2 1 » i.e le premier domaine (fixe) à 2 clusters, le 2eme domaine (sans fil) a 1 cluster. La ligne suivante définit le nombre de nœuds pour chaque cluster qui est « 1 1 4 » i.e un nœud pour chaque cluster (pour le

domaine fixe) et 4 nœuds pour le cluster du domaine sans fil. La topologie obtenue est de trois niveaux (voir **laFigure B**)



Exemple de la topologie wired-cum wireless

Ensuite on doit créer les nœuds fixes, mobiles et les stations de bases.

- **Création des nœuds fixes :**

```
# création des nœuds fixes
set temp {0.0.0 0.1.0} ;# on utilise l'adressage hiérarchique
for {set i 0} {$i < $num_wired_nodes} {incr i} {
 set W($i) [$ns_ node [lindex $temp $i]]  ;# la commande de
                                          ;# création
}
```

- **Création des stations de bases et des nœuds mobiles :**

Cette partie se résume dans ce bloc de création suivant :

```
$ns_ node-config -adhocRouting  valeur
                 -llType        valeur
                 -macType       valeur
                 -ifqType       valeur
                 -ifqLen        valeur
                 -antType       valeur
                 -propType      valeur
                 -phyType       valeur
                 -topoInstance  valeur
                 -channelType   valeur
                 -wiredRouting   ON /OFF
                 -agentTrace    ON /OFF
                 -routerTrace   ON /OFF
                 -macTrace      ON /OFF
                 -movementTrace ON /OFF
```

Ces paramètres sont respectivement :

- Le protocole de routage, qui peut prendre DSDV, TORA, AODV,DSR, et pour notre simulation on prend DSDV le seul qui supporte le Modèle du CMU monarch.
- La couche physique, sa valeur est toujours LL.
- La couche MAC : la couche d'allocation des canaux, peut prendre les valeurs Mac/802_11, Mac/Tdma ou Mac/802_3.
- La queue où les paquets sont reçus ou perdus, elle prend les valeurs : Drop tail, FQ (fair queuing), SFQ (Stochastic FQ), DRR (Deficit Round Robin), RED (Random Early Detection), CBQ (class-based queuing).
- La taille de la queue utilisée.
- L'antenne : une seule valeur est définie, Antenna/OmniAntenna.
- Le modèle de propagation radio : Propagation/TwoRayGround, Propagation/Shadowing, Propagation/FreeSpace.
- La couche physique : Phy/WirelessPhy.

- Topographie : la surface de la simulation (exemple 670 × 670 m²)
- Le type du canal : Channel/WirelessChannel
- L'activation du routage fixe ou non (pour les nœuds mobiles : OFF).
- Les quatres derniers sont utilisés pour activer la trace.

- **Création des nœuds mobiles :**

```
$ns node-config -wiredRouting OFF
for {set j 0} {$j < $opt(nn)} {incr j} {
    set node_($j) [ $ns_ node [lindex
$temp[expr$j+1]]]
    $node_($j) base-station [AddrParams
addr2id \
            [$BS(0) node-addr]]
}
```

- **Mhi.cc :**

```
# include "Mhi.i"

static class MSS class: public Tclclass {
public:
          MSS class() : TclClass ("MSS") {}
          Tclobject*create (int,const char*const*) {
           Return ( new MSS};
}
} class_MSS;

static class MHI class: public Tclclass {
public:
          MHI class() : TclClass ("MHI") {}
          Tclobject*create (int,const char*const*) {
           Return ( new MHI};
}
} class_MHI;
```

```
File_enreg :: file_enreg()
{ e= new enreg [MAX_MHI_MSS] ;
 i=0 ;
}

file_enreg ::~file_enreg()
{ delete e ;
}

int MSS:: command (int argc, const char*const*argv)
{
Tcl & tcl= Tcl:: instance();
Return Node:: command (argc,argv)
}

void MSS:: send_msg_mss(int message,MSS*mss_dest,MHI*mhi_dest)
{ mss_dest→rec_msg_mss(message,this,mhi_dest);
}

void MSS:: send_msg_mhi(int message,MHI*mhi_dest)
{ mhi_dest→rec_msg (message,this);
}

void MSS:: rec_msg_mss(int message,MSS*mss_sender,MHI*mhi_sender)
{ if (message==DJ_servir)
{ if (enrg*mhj=file_attente→get(mhi_sender))
   {mhj→horloge=(mss_sender→file_attente→get(mhi_sender))→horloge
    mhj→servi=1;
    }
 }
if(message==DJ_RELEASE)
{file_attente→pop(mhi_sender);
}
}

void MSS::rec_msg_mhi(int message,MHI*mhi_sender)
{ if(message==DJ_DEMANDE)
 {
 Node*tnode=nodehead_Un_first;
 Double h=0,h1 ;
 For( ;tnode ;tnode=tnode→nextnode())
   {
    if(classid(tnode)==MSS)
     h1=((MSS*)tnode)→get_MAX()
    if(h1>h)h=h1;
    }
file_attente→push(mhi_sender,h+1,0);
```

```
 For( ;tnode ;tnode=tnode→nextnode())
{send_msg_mss(DJ_servir,(MSS*)tnode,mhi_sender) ;
}
if(((file_attente→get(Mhi_sender))→horloge>=get_Max())
  &&((file_attente→get(mhi_sender))→servi))
 send_msg_mhi(DJ_ACCESS,mhi_sender);
}
{file_demande[c_demande++]mhi_sender ;
if!(jeton_demande)
{jeton_demande=0 ;
if!(jeton_present)
{hor++ ;
for( ;tnode ;tnode=tnode→nextnode())
{if(classid(tnode)==MSS)send_msg_mss(DJ_REQ,(MSS*)tnode,mhi_sender);
}
}else
{dedans=0;
jeton_present=0;
for(int i=0;i<c_demande,i++)file_service=file_demande
c_demande=0;
c_service--;
send_msg_mhi(DJ_AUTORISATION,mhi_sender) ;
   }
 }
}
if(message==DJ_jeton)
{dedans=0;
 jeton_present=0;
for(int i=0;i<c_demande;i++)file_service=file_demande;
c_demande=0;
send_msg_mhi(DJ_AUTHORISATION,mhi_sender);
}
if(message==DJ_PANNE)
 {
panne [get_id(mhi_sender)]=0;
if(this==PPNEP(mhi_sender))
if!(jeton_present)
{
send_msg_mhi(DJ_ELECTION,PSNEP(mhi_sender));
}
}*/
}
MHI*MSS::PSNEP(MHI*mhi)
{
 return(tnode→nextnode());
}
MHI*MSS::PPNEP(MHI*mhi)
{
return(tnode→prevnode());
}
```

```
Double MSS ::get_MAX()
{double h=0;
int i;
for(i=0;i<file_attente→i;i++)
{if(h<(file_attente→e[i]).horloge)h=(file_attente→e[i]).horloge;
}
return(h);
}

void file_enreg::push(MHI*pmhi,doublephorloge,int pservi)
{i++
e[i].mhi=pmhi;
e[i].horloge=phorloge;
e[i].servi=pservi;
}
void file_enreg::pop(MHI*mhi
{for(intj=0;j<i;j++)
{if(mhi==e[j].mhi)e[j]=e[j+1]
}
}
enreg*file_enreg::get(MHI*mhi)
{enreg*t=0;
for(intj=0;j<i;j++)
{if(e[j].mhi==mhi)t=&e[j];
}
return(t);
}
//1:$mhi send_demande$mss
//2:$mhi set_mss$mss
int MHI::command(intargc,constchar*const*argv)
{
Tcl&tcl=Tcl::instance();
If(argc==3){
      If(strcmp(argv[1],"send_demande")==0){
          Node*node=(Node*)Tclobject::lookup(argv[2]);
           If(node==0){
              Tcl.resultf("invalidnode%s",argv[2]);
               Return(Tcl_ERROR);
            }
            send_msg(DJ_DEMANDE;current_mss);
            return TCL_OK;
        }
        if(strcmp(argv[1],"set_mss")==0){
            Node*node=(Node*)Tclobject::lookup(argv[2]);
             If(node==0){
             Tcl.resultf("invalidnode%s",argv[2]);
            Return(TCL_ERROR);
           }
           current_mss=(MSS*)node;
```

```
      ReturnTCL_OK ;
      }
    }
return Node ::command(argc,argv) ;
}

void MHI::send_msg(intmessage,MSS*mss_dest)
{mss_dest→rec_msg_mhi(message,this);
}

void::rec_msg(intmessage,MSS*mss_sender)
{if(message==DJ_ACCESS)
{//section critique
//do something,we have"jeton"
send_msg(DJ_LIBERER,mss_sender);
}
}
```

Bibliographie

[ACH, 94] A. Acharya, B.R.Badrinath & T. Imielinski. Designing distributed algorithms for mobile computing Networks, Technical report, Rugters, 1994.

[BAD, 94] B.R.Badrinath & A.Acharya. A Framework for delivering multicast messages in network with mobil hosts. Technical report DCS- TR-310 Rugters, 1994.

[BAG, 95] Baggio, A. Environnements mobiles: Etude et synthèse bibliographiques. TR Masi 95/29, Université Pierre et Marie Curie, Laboratoire Masi, Sept 1995.

[BAG, 96] Baggio, A. Environnements mobiles : Caractéristiques et problèmes. TR Masi 95/29, Université Pierre et Marie Curie, Laboratoire Masi, Sept 1995.

[DAH, 98] F. Dahamni & A. Maddi. Exclusion Mutuelle dans un Environnement Mobile. CARI, 98. 4éme Colloque Africain sur la recherche en informatique (Dakar), 1998

[DIJ, 74] E.W. Dijkstra. Self-Stabilizing Systems in Spite of Distributed Control, Comm. ACM, Vol.17,11, 1974.

[DIJ, 80] E.W. Dijkstra & C. S. Scholten. Termination Detection for Diffsing Computations, Inf. Proc. Letters Vol. 11,1,1980.

[EST, 99] D. Estrin, M. Handley, J. Heidemann, S. McCanne, Y. Xu, H.Yu. Network visualization with the VINT Network Animator NAM, Mars 1999.

[FOR, 94] G. Forman & J. Zahorjan. The challenges of Mobile Computing, IEEE Computer, Vol.27, N°4, 1994.

[GRA,00] Gwendal Le Grand, Jalel Ben-Othman, Eric Horlait. Réservation de ressources dans un environnement mobile avec MIR : Mobile IP Protocol. Université de Pierre et Marie Curie,2000.

[GRA,01] Gwendal Le Grand, Jalel Ben-Othman, Eric Horlait. Gestion de trafic intra et inter-cellulaire dans les réseaux sans fil. Université de Pierre et Marie Curie,2001.

[HEH,81] E. C. R. Hehner & R. K. Shymasundar. An Implementation of P et V, Inf. Proc. Letters. Vol 12,4, 1981.

[IMI, 94] T. Imielinski & B. R. Badrinath. Mobile Wireless Computing: Challenges in Data Management, Comm. ACM, 37 (10), 1994.

[ISI, 96] Université de Californie du sud ISI. Virtuel InterNetwork Testbed (VINT): methods and systems, 1996.

[LAM,74] L. Lamport. A new Solution of Dijkstra's Conccurrent Programing Problem, Comm. ACM, Vol 17,8 , 1974.

[LAM,78] L. Lamport. Time, clocks and ordering of events in a distributed system, Comm. ACM, 27 (7), 1978.

[LEL, 77] G. Le Lann. Distributed Systems, Towards a Formal Approch, IFIP Congress, TORONTO, 1977.

[MAD,87] A. Maddi. Contrôle des transferts d'informations dans les systèmes distribués Application à la détection de l'interblocage et à l'élection dans un réseau quelconque de processus, Thèse de Docteur de l'Université de Rennes 1, N° d'ordre 118, 1987.

[NS, 02] The ns Manual (formely ns Notes and Documentation). The VINT Project A Collaboration between researches at UC Berkeley, LBL, USC/ISI, and Xerox Parc. Kevin Fall, Editor. Kannan Varadhan, Editor. January12, 2002.

[PAU,99] Paul Couderc, Anne Marie Kermarrec, Michel Banatre. Approches adaptatives en mobilité : Une Synthése. IRISA, project Solidor, 1999.

[PER,96] J.M.Perrichon. Les réseax sans fil, Masson. 1996

[RAY,84] M. Raynal. Algorithmique de parallélisme: Le problème d'exclusion mutuelle, Dunod. 1984.

[RAY,85] M. Raynal. Algorithmes distributes et protocols, EYROLLES. 1985.

[RIC , 81] G. Ricart & A.K. Agrawala. An Optimal Algorithm for Mutual Exclusion in Computer Networks, Comm. ACM. Vol.24,1. 1981.

[SIN , 95] M. Singhal, N.G. Shivaratri & R. Prakash. Distributed dynamic channel allocation for mobile computing. Proc. Of the 14th ACM Symp. On principles of Distributed Computing. 1995.

[SUZ, 82] I. Suzuki, I. Kasami. A distributed mutual exclusion algorithm. ACM Transactions On Comuters Systems, Vol 3, N°4, 1982.