UNIVERSITE DES SCIENCES ET TECHNOLOGIES HOUARI BOUMEDIENE

FACULTE D'ELECTRONIQUE ET INFORMATIQUE
DEPARTEMENT INFORMATIQUE

THESE

Présentée pour l'obtention du grade de :

Magister en Informatique

Par

M^{elle} HADDOUCHE Nadia

THEME

Un Schéma de Localisation des Mobiles basé sur les Quorums

Soutenue devant le jury composé de :

Mr. AHMED NACER Mohamed	Maître de conférence, U.S.T.H.B.	Président
Me. AISSANI Aicha	Maître de conférence, U.S.T.H.B.	Examinateur
Mr. LARABI Slimane	Maître de conférence, U.S.T.H.B.	Examinateur
Mr. BELKHIR abdelkader	Chargé de.Cours, U.S.T.H.B.	Examinateur
Mr. BADACHE Nadjib	Maître de conférence, U.S.T.H.B.	Rapporteur

U.S.T.H.B. 2002 / 2003

Je dédie ce travail à :

A mes parents

A ma famille

A mes amis

REMERCIEMENTS

Je tiens à remercier Monsieur M. AHMED NACER qui m’a fait l’honneur de présider le jury de cette thèse.
Je tiens à remercier Monsieur N. BADACHE de m’avoir proposé ce sujet, de m’avoir encadré et d’avoir consacré énormément de temps pour m’aider à la réalisation de ce travail.
Je tiens à remercier Madame AISSANI, Monsieur S. LARABI et Monsieur A. BELKHIR qui ont bien voulu faire partie du jury de cette thèse.

SOMMAIRE

CHAPITRE III :Les Quorums, leurs mesures de qualité et leurs domaines d'application

RESUME

La gestion de la localisation de sites mobiles est un des problèmes les plus importants dans les systèmes de réseaux mobiles. En effet, réduire le coût de recherche augmente celui de la mise à jour et inversement. Aussi, un compromis entre le coût de recherche et de mise à jour doit être défini. Cette thèse propose une architecture de données de localisation basée sur les quorums. Le coût de communication est ainsi réduit car l'information de localisation d'un mobile est sauvegardée de manière efficace dans un sous ensemble de registres de localisation qui change avec les déplacements du mobile. L'algorithme proposé est évalué en terme de coût total de recherche et de mise à jour et est comparé à l'algorithme de Ihn-Han Bae.

Mots clé : Réseaux mobiles, Localisation, Quorum.

La technologie des communications mobiles évolue de plus en plus rapidement et offre à l'utilisateur la capacité d'accéder à l'information et à divers services de n'importe quel endroit et à n'importe quel moment.

Les unités de calcul mobiles sont de plus en plus performants en terme de puissance alors que leur taille ne cesse de diminuer.

Malgré leur taille, la plupart des ordinateurs mobiles sont équipés d'interfaces de connexion sans fil pour communiquer avec la partie fixe du réseau et même avec d'autres ordinateurs mobiles. L'environnement résultant ne contraint plus les utilisateurs à une position fixe ou universellement connue dans le réseau et leur permet une mobilité sans restriction. Aussi, les attentes des utilisateurs évoluent avec l'évolution de la technologie mobile. Idéalement, un utilisateur désire avoir un accès continu à l'information indépendamment de sa position courante. Le fait d'être mobile ne doit pas être un inconvénient et obliger l'utilisateur à se déconnecter temporairement du réseau le temps de mouvement. La mobilité en elle même doit rester transparente à l'utilisateur. Même s'il va d'une cellule à une autre, il continue de maintenir ses connexions au réseau et ne devra pas les briser puis réinitialiser. Un ordinateur reste en contact continu avec les ressources du réseau qu'il utilise par ses applications. En d'autres termes, il doit y avoir une connectivité ininterruptible entre les applications et les ressources **[40]**.

C'est pourquoi, la mobilité pose certaines questions fondamentales telles que : Comment peut-on savoir à n'importe quel moment où se trouve un mobile x ? A qui doit-on s'adresser pour obtenir cette information ? Et une fois cette information obtenue, comment le paquet est délivré au mobile ?

Ces questions résument essentiellement les problèmes de la gestion de la localisation. Ce sont des questions auxquelles il est de plus en plus difficile d'y répondre avec un moindre coût à mesure que le nombre de mobiles augmente et que la taille des réseaux mobiles grandit.

La façon la plus simple pour localiser n'importe quel mobile à tout moment est de maintenir l'adresse de tous les mobiles dans une même base de données. L'adresse d'un mobile y sera mise à jour à chaque fois qu'il se déplace d'une cellule à une autre. Aussi, lorsqu'un mobile X veut localiser un autre mobile Y, il interroge cette base de données qui lui envoie l'adresse de Y qui est ainsi localisé.

Cette solution bien que simple à implémenter, présente un inconvénient majeur. Si ce serveur tombe en panne, la localisation de tous les mobiles devient inaccessible. De plus, cette solution n'exploite pas la distribution géographique des mobiles dans le système et la localité des appels et déplacements pour réduire le coût de recherche et de mise à jour. Enfin, la

charge du système peut être très importante : le nombre de recherches et mises à jours étant très grand pour les réseaux de très grandes tailles, le système risque d'être rapidement saturé. D'où la nécessité d'utiliser une base de données de localisation distribuée.

Dans cette thèse, nous nous intéressons au problème de localisation des objets dans les réseaux mobiles. Nos travaux ont pour but de proposer une architecture distribuée pour la base de données de localisation. Nous définissons pour cette architecture, les protocoles de recherche et de mise à jour exécutés suite à un appel reçu par un mobile ou à son déplacement.

Cette thèse est structurée en cinq chapitres.

Le premier chapitre présentes quatre architectures de bases de données de localisation existantes avec leurs algorithmes de recherche et mise à jour.

Le deuxième chapitre illustre les différentes techniques utilisées pour réduire les coûts de recherche et mises à jour sur les deux architectures les plus utilisées parmi les quatre citées dans le premier chapitre.

Le troisième chapitre présente quelques définitions concernant les quorums : une notion que nous utiliserons ultérieurement pour proposer notre architecture. Ce chapitre comporte également les domaines d'application des quorums.

Le quatrième chapitre porte sur l'utilisation des quorums dans la localisation des mobiles et présente les travaux antérieurs réalisés dans ce sens.

Le dernier chapitre est consacré à notre contribution au problème de la localisation. Il présente l'architecture de la base de données de localisation proposée et qui est basée sur les quorums. Les algorithmes de recherche et de mise à jour y sont évalués en terme de complexité des messages et de coût de communication. Une comparaison avec l'algorithme de Ihn-Han Bae montrera la consistance de notre proposition.

Enfin, nous terminons cette thèse par une conclusion du travail réalisé et les perspectives envisageables.

I. INTRODUCTION

Dans les systèmes distribués actuels, la notion de mobilité apparaît dans différentes formes et applications. De plus en plus de personnes utilisent des ordinateurs mobiles ou les téléphones sans fil. Le software mobile, i.e. du code ou données qui circule dans le réseau se manifeste comme une nouvelle forme de construction d'applications dites distribuées. En présence de mobilité, le coût de la communication est augmenté par celui de la localisation car par exemple lorsqu'un serveur essaye d'envoyer un message à un utilisateur qui est en mouvement, il doit d'abord déterminer sa position courante, i.e. le localiser pour le contacter.
On distingue ainsi deux types de mobilité :
- La mobilité hardware : Les sites mobiles représentent un modèle dans lequel l'utilisateur se déplace avec son téléphone ou ordinateur portable. La caractéristique de ce modèle est que la mobilité de ces sites est directement reliée à celle de leurs utilisateurs. Ceci a deux conséquences importantes. La première est que la vitesse de déplacement d'une personne ne dépasse pas les 1000 miles / h. La seconde est qu'un mobile ne peut se trouver qu'à une seule position à la fois.
- La mobilité software : Les softwares mobiles qui par contre peuvent ne pas être à une seule position à la fois et se déplacent à une vitesse proche de celle de la lumière.
Les techniques de localisation présentées dans ce qui suit sont valables pour ces deux types de mobilités. Aussi, aucune distinction ne sera faite entre les deux et le terme « objet » désignera aussi bien le hardware mobile que le software mobile.

II. ADMINISTRATION DE LA LOCALISATION

II.1. Structure d'un réseau mobile

Dans un réseau mobile, il existe deux ensembles d'entités : les sites fixes et les sites mobiles (SM). Les sites fixes et les liens qui les relient constituent un réseau statique classique. Certains de ces sites fixes sont munis d'une interface de communication sans fil qui couvre une certaine aire géographique appelée *cellule.* Ce type de sites fixes sont appelés *Stations de Base* (SB). Tout site mobile ne peut communiquer qu'avec la SB qui couvre la cellule dans laquelle il se trouve. C'est cette SB qui représentera sa position courante (voir Figure 1.1).

L'échange de messages entre une SB et un SM se fait à travers deux canaux de communication : un canal SB-SM qui transmet les messages de la SB au SM et un autre qui transmet les messages du SM à la SB.

Si un site mobile U1 dans une station de base S1 veut envoyer un message S à un site mobile U2 dans une station de base S2 alors U1 envoie le message S à S1 par une liaison sans fil. S1 envoie S à S2 à travers le réseau statique et enfin S2 envoie S à U2 via son interface de communication sans fil **[8]**.

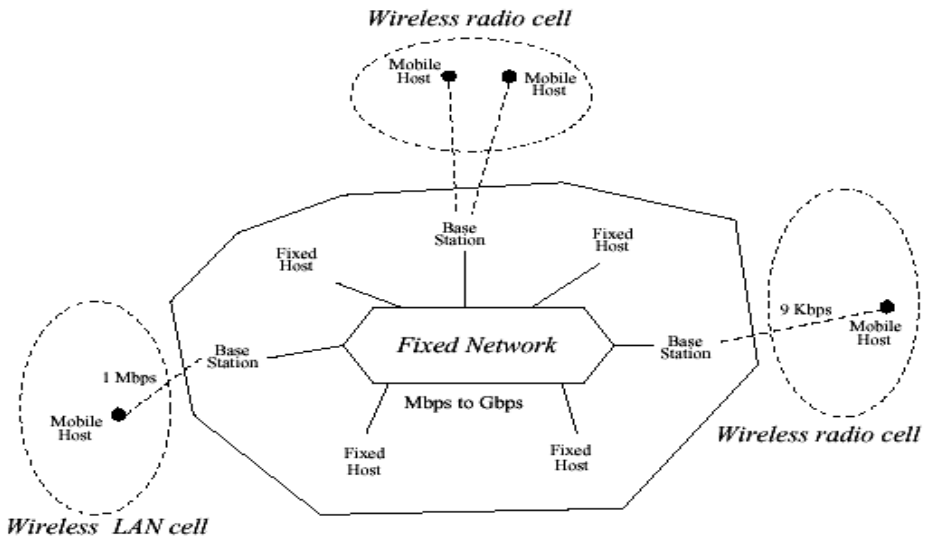
II.2. La gestion de la localisation

Dans les réseaux mobiles (mais aussi pour les réseaux statiques dans le cas du software mobile) les objets peuvent se déplacer d'une position à une autre. Afin de pouvoir localiser ces objets efficacement, les informations de leurs positions courantes doivent être sauvegardées quelque part dans le réseau dans des ***bases de données de localisation***.

La gestion de la localisation implique deux opérations de base :La recherche et la mise à jour .

La recherche est l'opération qui permet de localiser un objet mobile lorsque il y a un besoin de le contacter. L'adresse de ce mobile est recherchée dans la base de données de localisation.

La mise à jour est l'opération qui permet d'informer le réseau (la base de données de localisation) de la position d'un mobile lorsque celui-ci se déplace vers une nouvelle position dans le réseau. Ce mobile met à jour son adresse dans la base de données de localisation.



- Figure 1.1 -

III. ARCHITECTURES DE LA BASE DE DONNEES DE LOCALISATION

Quatre architectures de base pour les bases de données de localisation distribuées sont utilisées pour sauvegarder la localisations des mobiles du réseau **[1]**.

III.1. Schéma à deux niveaux

Dans ce schéma une base de données locale appelée Registre de localisation locale (*Home Location Register* HLR) est associée à chaque mobile. L'emplacement de la HLR est prédéfini pour chaque objet. Elle contient sa position courante dans le réseau. Pour localiser un objet, il suffit d'interroger sa HLR. Lorsqu'un objet se déplace d'une zone à une autre, sa HLR est contactée et est mise à jour **[1]**.

De plus , il existe une base de données autre que la HLR dans chaque zone qui est le registre de localisation des visiteurs (*Visitor location Register* VLR). Elle contient la copie du profil d'un mobile qui se trouve dans sa zone si cette zone ne contient pas sa HLR. Lorsqu'il y a un appel d'une zone i à un mobile x, la VLR de i est d'abord interrogée et si l'objet x n'existe pas alors la HLR de x est interrogée. Pour la mise à jour, lorsqu'un objet x se déplace de la zone i vers la zone j, en plus de la mise à jour de la HLR, l'entrée x est supprimée de la VLR de i et est ajoutée à la VLR de j.

Le problème avec cette approche est que l'affectation d'une HLR à un objet est permanent : malgré une grande mobilité de certains objets, leurs localisations sont fixes ce qui complique la tâche de leur affecter des identificateurs (handles). De plus, ce schéma n'est pas extensible (scalable) et ne s'adapte pas bien aux grands systèmes répartis où les sites sont géographiquement très dispersés. Enfin, la localité des appels et des mouvements n'est pas prise en compte, quelque soit la position d'un appelé par rapport à l'appelant, ce dernier doit passer par la HLR de l'appelé pour le contacter.

III.2. Schéma hiérarchique

Dans ce schéma, les bases de données sont organisées en une structure arborescente. L'arbre est construit en une hiérarchie de domaines qui sont des régions géographiques ou administratives (comme pour le DNS). Chaque feuille de l'arbre représente la base de données d'une région et contient les entrées de tous les objets qui sont dans cette région. Un nœud interne maintient des informations sur tous les objets enregistrés dans les feuilles appartenant au sous arbre qui contient ses fils.

Dans ce schéma, l'information concernant un mobile dans un nœud interne peut être soit un pointeur vers son entrée dans un niveau inférieur soit l'adresse de cet objet. Aussi, le type de localisation choisi affecte considérablement le coût de la recherche ou de la mise à jour.

En effet, soit LCA(i,j) l'ancêtre commun des nœuds i, j le plus proche (*Least Common Ancestor*).

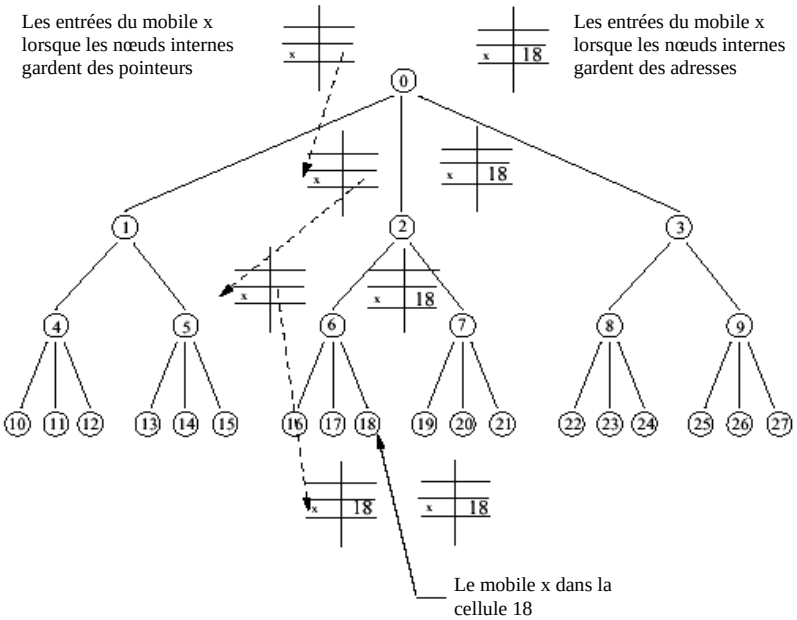
1. Considérons le cas où les nœuds internes de l'arbre contiennent des pointeurs (voir Figure 1.2 à gauche) :

Pour un objet X qui réside dans le nœud 18, X a une entrée dans le nœud 0 qui pointe vers une entrée de X dans le nœud 2 qui à son tour pointe vers l'entrée de X dans le nœud 6.Ce dernier pointe vers l'entrée de X dans le nœud 18.

♦ Lorsque X se déplace de la zone i vers la zone j, les entrées de X dans les bases de données de j vers LCA(i,j) et de LCA(i,j) vers j sont mises à jour. Si X se déplace du nœud 18 vers le 20 les entrées de X sont supprimées des nœuds 18 et 6 et sont ajoutées dans les nœuds 7 et 20 et mises à jour dans le nœud 2.

♦ Lorsque l'objet X dans la zone i appelle un objet y dans la zone j, la procédure de recherche interroge les bases de données à partir de i et monte dans l'arbre jusqu'à la rencontre du premier nœud contenant une entrée de y, en l'occurrence le LCA(i,j).

Ensuite, la recherche continue en suivant les pointeurs vers j (voir Figure 1.2 à gauche). Un appel à partir du nœud 21 pour un objet dans la zone 18 suivra le chemin 21,7,2,6,18 avant de trouver y.



.

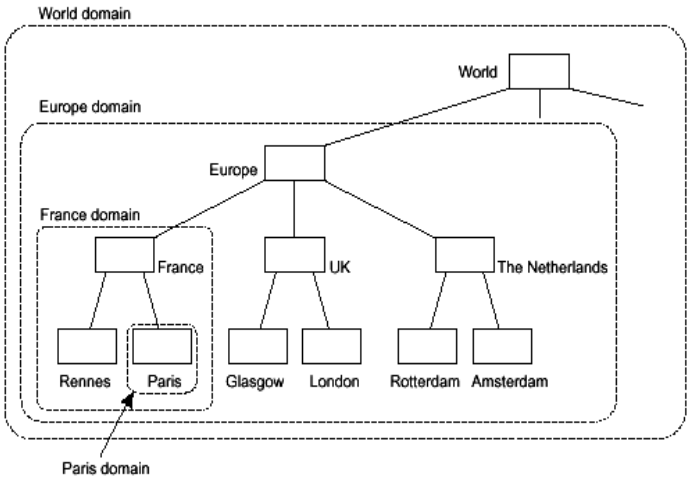
- Figure 1.2 -

2. Considérons maintenant le second cas où tous les nœuds internes contiennent l'adresse actuelle des objets mobiles :

♦ Ici, le déplacement d'un objet X d'une zone i vers une zone j implique des mises à jour de j vers la racine de l'arbre et de la racine vers i (voir Figure 1.2 à droite). Le déplacement de x du nœud 18 vers le nœud 20 implique la suppression des entrées de X dans les zones 18 et 6, la mise à jour des zones 0 et 2 et enfin l'ajout des entrées de X dans les zones 7 et 20.

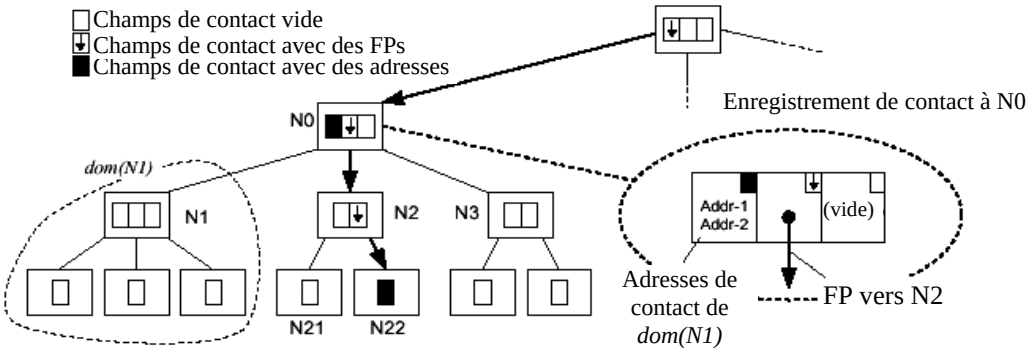
♦ Par contre, si un objet X dans la zone i recherche un mobile y qui se trouve dans la zone j, le coût de la recherche dans ce cas là est réduit car une fois le LCA(i,j) atteint, il suffit d'accéder directement au nœud j (voir Figure 1.2 à droite). Un appel de la zone 21 à un mobile X dans la zone 18 nécessite l'accès aux nœuds 21,7,2 et puis 18.

Un modèle hiérarchique intéressant est le modèle ***GLOBE*** **[21]**. ***Globe*** est un système distribué à l'échelle planétaire (World-Wide-Area) qui offre une structure extensible, qui peut supporter un très grand nombre d'objets, utilisateurs et machines et qui favorise la localité (voir Figure 1.3). Son système de localisation distingue entre deux services, le *naming service* et le *locating service*. Les deux services utilisent trois façons différentes pour désigner un objet : un nom, un handle et une adresse de contact. Le nom de l'objet est un nom facile à retenir et à utiliser, le handle est un identificateur unique de l'objet indépendant de sa position et enfin l'adresse de contact d'un objet est ce qui permet de le retrouver. Le naming service se charge de relier les noms des objets à leurs handles et le locating service de relier les handles aux adresses de contact.



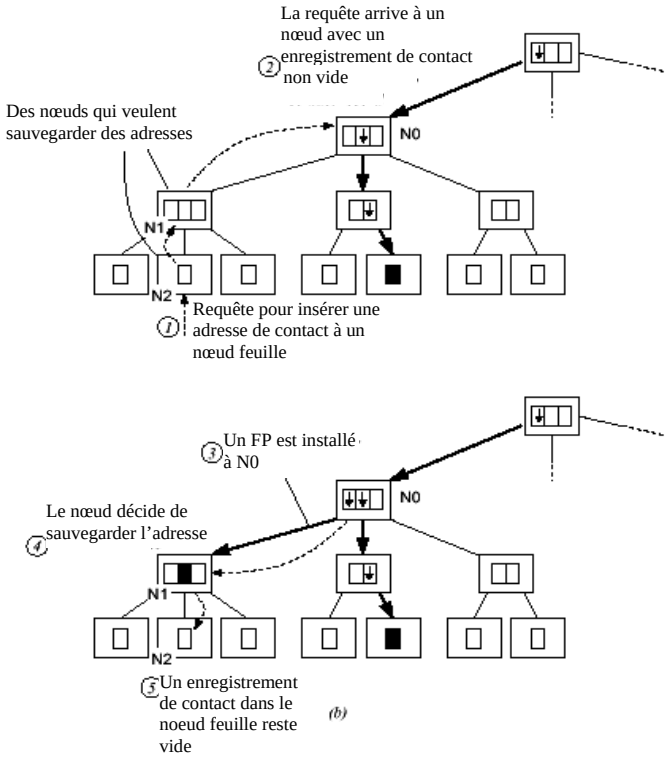
- Figure 1.3 -

Dans ce modèle, chaque nœud se charge de sauvegarder les liens entres les handles et les adresses de contact des objets qui se trouvent dans le domaine couvert par ce noeud. Chaque objet dans un domaine i possède un *enregistrement de contact*. Un enregistrement de contact dans un nœud interne est composé de *champs de contact*, un pour chaque sous domaine de i qui peut contenir soit une adresse, soit un pointeur ou être vide (voir Figure 1.4).

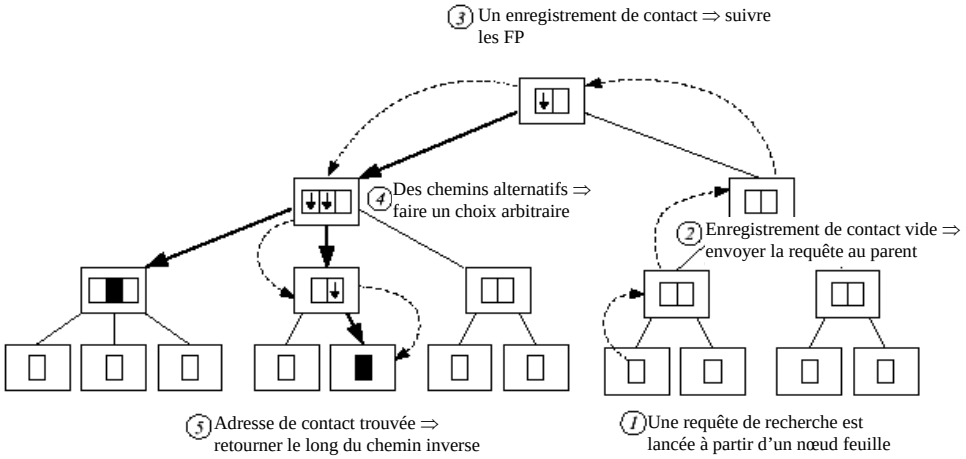


- Figure 1.4 -

Les recherches et mises à jour se font dans le ***Globe*** selon Figures 1.5 et 1.6 **[22,23]**.



- Figure 1.5 -



- Figure 1.6 -

Pour améliorer les performances de ***Globe***, la sauvegarde de l'adresse d'un objet X ne se fait pas forcément dans le nœud feuille qui contient X mais dans un nœud de localisation dit stable qui dépend du comportement de cet objet.
Un nœud est dit stable pour un mobile X si X se déplace fréquemment dans le domaine géré par ce nœud **[24,25]**.
L'inconvénient avec le schéma hiérarchique est que le coût des recherches et mises à jour est assez important en comparaison avec le modèle à deux niveaux ce qui ne convient pas à des systèmes avec de requêtes en temps réel . De plus, plus on monte dans l'arbre plus la taille des bases de données augmente. Ce problème peut être résolu en partitionnant la base de données d'un nœud en un ensemble de sous bases de données physiques, chacune responsable d'un sous ensemble de nœuds de ce nœud telle que l'ensemble de ces sous bases de données formeront une seule base de données logique qui sera visible pour les utilisateurs **[1]**.

Voici un résumé des avantages et inconvénients de l'architecture hiérarchique.

(+) Pas de HLR	(-) Augmente le nombre d'opérations
(+) Supporte la localité	(-) Augmente les accès dans les niveaux hauts.

Un schéma hybride entre le schéma hiérarchique et le schéma à deux niveaux est aussi possible. En supposant que les bases de données sont maintenues dans des nœuds sélectionnés et que les HLRs sont utilisées, un appel d'une zone i déclenche une recherche de l'appelé à partir de i et qui remonte le chemin vers le LCA de i et de la HLR de l'appelé pour descendre ensuite vers l'appelé à moins qu'une entrée pour l'appelé ne soit trouvée en cours de chemin.

III.3. La hiérarchie non arborescente : Appariement régional

L'objectif des *répertoires régionaux* **[9]** est de favoriser les opérations locales. Dans cette approche, une hiérarchie D de δ répertoires régionaux est construite avec δ=log d où d est la distance maximale entre n'importe quel couple de sites du réseau.

Un répertoire régional (*Regional Directory*) RD_i à un niveau i permet la localisation de n'importe quel objet à une distance 2^i de lui. Pour ce faire, Deux ensembles des sites sont associés à chaque site u dans un RD_i : un ensemble de lecture $Read_i(u)$ et un autre d'écriture $Write_i(u)$ avec la propriété que $Read_i(u) \cap Write_i(w) \neq \varnothing$ pour tout couple de sites u, w dont la distance les séparant est inférieure ou égale à 2^i. Chaque site u reporte l'adresse de tous les objets qu'il héberge dans tous les sites qui sont dans son $Write_i(u)$. Pour la recherche d'un mobile, u interroge tous les sites dans son $Read_i(u)$.

Si un objet se déplace à une nouvelle localisation à une distance K, seuls les log K plus bas niveaux de la hiérarchie sont mis à jour pour pointer vers la nouvelle adresse. Les autres répertoires continuent de pointer l'ancienne adresse où un pointeur vers la nouvelle adresse est créé.

Pour limiter la longueur de la chaîne de pointeurs, il est vérifié que pour chaque objet X, la distance C(x) parcourue depuis que son adresse à été mise à jour à RD_i est inférieure ou égale à 2^{i-1}-1 pour chaque niveau i.

Procédure de recherche

```
/* un appel du site w à l'objet x */
i :=0 ; adress := nil ;
Répéter
        i:=i+1;
        /* chercher dans le RDi */
        Pour chaque site u dans Read (w) interroger u ;
Jusqu'à adresse <> nil ;
Répéter
        Suivre les pointeurs
Jusqu'à atteindre x.
```

Procédure de déplacement

/* L'objet X se déplace du site v vers le site w */

Soit RD_j le plus haut répertoire où $C(x) > 2^{j-1}-1$

Pour i :=1 à max (j,δ)

/* Mettre à jour RD_i */

Pour tout site u dans Write (v)

Mettre à jour l'entrée de X

Ajouter un pointeur à RD_{i+1}.

III.4. Système d'administration de base de données centralisé (*Centralized Database Management System CDBMS* ***)***

Dans ce modèle, la localisation de tous les objets mobiles du réseau est maintenue dans une seule base de données centralisée. Dans ce cas, les recherches et mises à jour se font directement sur le CDBMS. Ce système est utile par exemple pour localiser tous les taxis d'une compagnie où une requête typique peut être : retrouver un taxi libre dans un rayon de 1 KM de la 33^{e} ,Avenue colonel Amirouche, Alger. Les applications qui utilisent ce genre de bases de données sont des application avec des bases de données d'objets mobiles (*Moving Objects Databases Applications :* MOD applications) **[6]**.

Les CDBMS spatiaux existantes ne sont pas satisfaisantes car ils ne facilitent pas la tâche de donner des handles à des objets en continuel mouvement. C'est pourquoi, les recherches actuelles **[7]** essaient d'étendre les bases de données spatiales avec de telles capacités.

IV. CONCLUSION

La localisation des objets mobiles est un des problèmes les plus importants mais aussi les plus difficile à résoudre dans les réseaux mobiles. Dans ce chapitre, nous avons présenté les différents schémas de localisation existants pour résoudre ce problème et réaliser la gestion de la localisation des mobiles.

I. INTRODUCTION

Les architectures les plus utilisées pour la localisation des objets mobiles sont le schéma à deux niveaux et le schéma hiérarchique. Aussi on ne s'intéressera dans ce qui suit qu'à ces deux modèles. Tous les deux, utilisés tels qu'ils ont été présentés dans le chapitre précédent, impliquent un coût considérable en recherche et en mise à jour. Pour réduire ce coût, plusieurs techniques ont été proposées pour améliorer ces deux schémas, à savoir le placement des bases de données dans des nœuds sélectifs, le cache, la réplication et les pointeurs en avant **[1]**.

II. PLACEMENT DES BASES DE DONNES

Maintenir la localisation dans tous les nœuds de la hiérarchie implique un coût élevé surtout en mises à jour. Pour réduire ce coût, les entrées des bases de données peuvent être mises seulement dans quelques nœuds. Dans ce cas, seuls les nœuds contenant les bases de données de localisation seront interrogés ou mis à jour.

Un placement possible de ces bases de données est de maintenir l'entrée d'un objet dans le nœud feuille où il réside. Dans ce cas, quand il n'y a pas de HLR associée à chaque mobile, une forme de recherche globale dans la hiérarchie est entamée suite à un appel. Pour ce scénario, trois stratégies de localisation sont proposées :

- La recherche plate (Recherche en largeur) : Elle commence à partir de la racine et parcourt l'arbre en parallèle.
- La recherche étendue : Elle commence par interroger la source de l'appelé (zone où il est enregistré initialement) puis son parent qui à son tour interroge ses autres fils et ainsi de suite.
- La recherche hybride : Elle commence par la recherche étendue puis continue avec une recherche plate si la localisation n'est pas trouvée parmi les enfants du parent de la source de l'appelé.

Trois architectures alternatives sont proposées dans ce qui suit pour le schéma hiérarchique et le schéma à deux niveaux :

II.1. Optimisation

Les informations de localisation sont maintenues dans des nœuds internes sélectionnés de sorte à optimiser certaines performances métriques, à savoir : minimiser ***a)*** le nombre d'accès et mises à jour des bases de données, ***b)*** le coût de la communication, ***c)*** la somme du trafic

dans le réseau. Aussi, le placement de ces bases de données dans une hiérarchie peut être vu comme la résolution d'un problème d'optimisation dont les fonctions incluent les objectifs citée ci-dessus ou une combinaison de ces objectifs.

II.2. Architecture d'une base de données hiérarchique dynamique

Cette architecture permet d'étendre le schéma à deux niveaux en y introduisant un nouveau niveau de bases de données appelé Registre d'annuaires (*Directory Register* DR). Chaque DR couvre un nombre de cellules. Périodiquement, il enregistre la configuration des mobiles se trouvant dans les cellules sous son service. En plus de maintenir les adresses locales de ces mobiles, chaque DR maintient pour certains objets soit l'*adresse distante directe* de leurs localisations actuelles ou l'*adresse distante indirecte* de leurs DRs courants.

Pour chaque objet, la sélection de l'ensemble de DRs qui garderont ses deux adresses dépend de son taux de mobilité et du profil des appels qu'il reçoit. Dans ce cas, la HLR de ce mobile peut garder trace soit de la cellule courante de ce mobile ou de son DR courant suivant aussi son taux de mobilité et le profil de ses appels. Ainsi, contrairement au schéma à deux nivaux et au schéma hiérarchique où la stratégie est la même pour tous les mobiles, ce modèle s'adapte au comportement de chaque objet.

II.3. Partitions

Pour chaque objet, on définit des partitions obtenues en regroupant les cellules parmi lesquelles il se déplace fréquemment et en séparant les cellules entre lesquelles il ne se déplace pas souvent. Pour chaque partition, le LCA de tous ses nœuds est appelé *représentatif*. Ce représentatif sait toujours si un objet se trouve dans sa partition ou pas (information très utile lors de la recherche en largeur). Aussi lorsqu'un objet change de partition, son ancien et nouveau représentatifs doivent être mis à jour.

Une autre approche du modèle des partitions est *les arbres de redirection*. Une seule partition appelée *région locale* contient tous les nœuds que peut visiter un mobile. Le représentatif de la région est appelé *agent de redirection.* Il contient les informations concernant tous les objets de sa région et redirige tous les appels vers un mobile de cette région directement vers cet objet. Les mouvements d'un mobile dans sa région locale sont enregistrés au niveau de l'agent de redirection et non dans le serveur de localisation de cet objet qui peut être hors de cette région.

III.CACHE

Lorsqu'un objet x localise un objet y suite à un appel, x peut garder l'adresse de y pour d'éventuels nouveaux appels vers y : C'est le principe du cache.

❖ Dans le schéma à deux niveaux, chaque fois qu'un objet x est appelé sa position est mise en cache dans la VLR de l'appelant, ainsi tous les appels en provenance de cette zone réutiliseront cette adresse. Aussi pour localiser un objet, la recherche commence par la VLR ensuite le cache de la VLR et finalement la HLR de l'appelé.

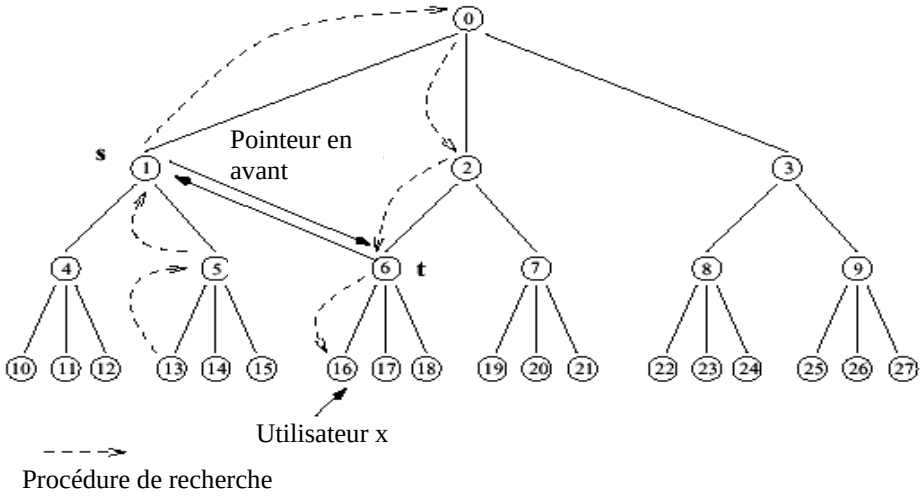
En raison de la mobilité des objets qui rend les adresses mises en cache obsolètes, il existe plusieurs approches pour le cache.

- Le cache actif : A chaque fois qu'un objet change de localisation, tous les caches associés à cet objet sont mis à jour.
- Le cache paresseux : Lors de l'utilisation d'une adresse à partir du cache, si la recherche échoue la HLR associée à l'appelé est contactée pour avoir son adresse et mettre à jour le cache.

Une autre façon de mettre à jour le cache consiste à considérer les caches obsolètes après une certaine période T qui est dynamiquement adaptée aux profils des appels et des mouvements de chaque mobile.

Lorsque la taille du cache est limitée, les techniques de remplacement de cache telle que LRU sont utilisées.

❖ Dans le schéma hiérarchique, lorsqu'un objet x dans un site i cherche un objet y dans le site j, il suit le chemin i $\rightarrow$LCA(i,j) puis de LCA(i,j)$\rightarrow$j. Lors du chemin inverse, deux pointeurs sont créés, l'un appelé *en avant* l'autre *inverse*. Le premier est une entrée dans un ancêtre de i appelé s qui pointe vers un ancêtre de j appelé t. le second pointeur pointe de t vers s. La prochaine fois qu'un objet de i appellera y, la recherche se fera de i vers s puis vers t et enfin vers y (voir Figure 2.1). Placer un pointeur en avant dans un niveau haut s dans l'arbre rend l'entrée de l'objet disponible pour beaucoup de zones mais les appels doivent parcourir un long chemin pour retrouver s. De même, placer un pointeur inverse à un niveau haut t augmente le coût des recherches mais l'entrée cache reste valide tant que l'objet se déplace dans le sous arbre de t.



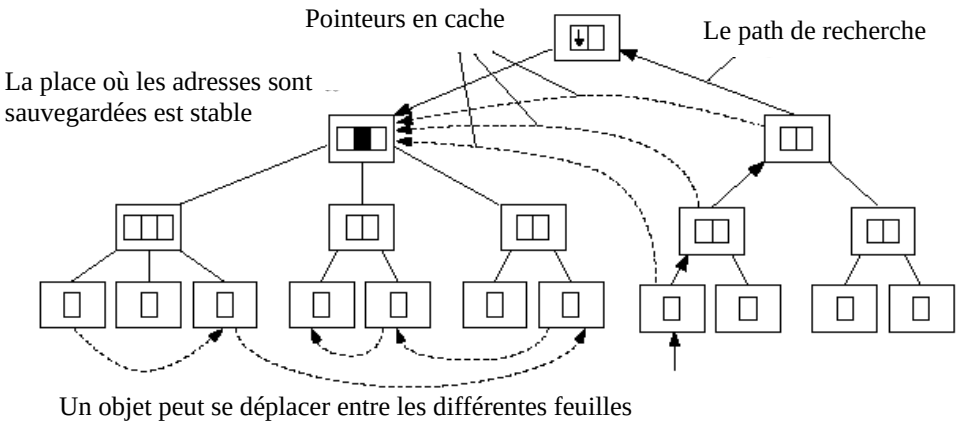
- Figure 2.1 -

Comme dans le schéma à deux niveaux, on parlera aussi ici de cache actif et paresseux pour gérer ses mises à jour.

Remarque

Dans le modèle ***Globe***, le cache ne contient pas l'adresse courante d'un mobile mais un pointeur vers le nœud où est sauvegardée son adresse **[25]**, ce qui réduit considérablement le coût de mise à jour du cache puisque le nœud de sauvegarde est stable (voir Figure 2.2).

Le cache peut être combiné avec la technique des partitions : Au lieu de mettre en cache l'adresse actuelle d'un objet, c'est son représentatif qui est mis en cache. Ceci réduit sérieusement le coût des mises à jour du cache.



- Figure 2.2 -

IV. REPLICATION

La réplication de la position de certains objets sur certains sites permet de réduire le coût de la recherche, mais augmente le coût des mises à jour en cas de mouvement d'un objet. Aussi, l'adresse d'un mobile ne peut être répliquée sur un site que si le coût des recherches et mises à jour est réduit avec l'utilisation de cette réplique.

Soit l'appel au taux de mobilité local LCMR (*Local Call to Mobility Ratio*) tel que $LCMR_{i,j} = C_{i,j} / U_i$ avec $C_{i,j}$: Le nombre d'appels prévus de la zone j à l'objet P_i durant un temps T et U_i : Le nombre de déplacements effectués par P_i durant cette même période T.

Soit α le coût des sauvegardes lors d'une requête locale et β le coût des mises à jour d'une réplique.

Alors une réplication de la localisation de P_i dans la zone j est judicieuse si

$$\alpha * C_{i,j} \geq \beta * U_i \qquad (1)$$

Pour affecter une copie à un site, en plus du coût, il faut prendre en considération la capacité de stockage et de service du site.

L'adresse de toutes les répliques pour un objet peuvent être sauvegardées dans sa HLR.

Au lieu de garder dans une réplique l'adresse exacte d'un mobile, une information grossière de son emplacement peut être sauvegardée. La granularité des répliques dépendra du compromis coût de mise à jour et recherche. Ainsi, choisir les sites qui contiendront les répliques dépend du profil recherche / mise à jour d'un mobile.

Voici quatre schémas de réplication par objet :

IV.1. Réplication par profil d'objet

L'objectif de cette méthode est de minimiser le coût total des mouvements et appels en ayant le maximum de répliques r_i par objet P_i et le maximum de P_i dans la base de données de la zone Z_j. Les répliques de l'adresse de P_i assignées à un ensemble de zones $R(P_i)$ doivent vérifier que le système (2) est minimisé et que les contraintes sur le nombre maximum de répliques par objet et de répliques par zone sont respectées.

$$\sum_{i=1}^{N}\sum_{j=1}^{M} z_j \in R(p_i)\ \beta * U_i - \alpha * C_{i,j} \qquad \text{.....(2)}$$

IV.2. Réplication dans l'ensemble de travail

Chaque objet communique fréquemment avec un petit nombre de sites qui forment son *ensemble de travail*. Il est donc logique d'avoir les copies de son adresse dans les membre de cet ensemble.

Cette approche est similaire à celle du paragraphe précédant excepté qu'ici il n'y a pas de contraintes sur la capacité de stockages des bases de données ou sur le nombre maximum de répliques par objet. L'inégalité (1) est évaluée localement chaque fois qu'au moins un de ses membres est modifié suite à une recherche ou une mise à jour.

IV.3. Réplication dans l'architecture hiérarchique

Comme pour le schéma à deux niveaux, la réplication sur un nœud interne (la sauvegarde proprement dite se fait dans les nœuds feuilles) dépend du LCMR. Sachant que le LCMR d'un nœud est égale à la somme des LCMRs de ses fils, les bases de données des hauts niveaux ont plus de chance d'être choisies pour garder les répliques. Mais les répliques dans les hauts niveaux impliquent un coût excessif en mises à jour. D'où la nécessité de fixer des niveaux maximum pour la réplication.

Voici une technique qui utilise quatre paramètres : Nmax, Smin, Smax et L pour déterminer quel nœud peut être sélectionné pour garder une réplique où Nmax est le nombre maximum de répliques par objet, L le niveau maximum dans la hiérarchie où une réplique peut être placée, Smax et Smin déterminent ensemble quand un nœud peut contenir une réplique.

La position d'un mobile i peut être répliquée sur le site j si $LCMR_{i,j} \geq Smax$. Elle ne peut être répliquée sur j si $LCMR_{i,j} < Smin$. Si $Smin \leq LCMR_{i,j} < Smax$ alors répliquer ou non dépend de la topologie de la base de données (i.e. Nmax et L). L'algorithme qui permet de déterminer les nœuds qui contiendront les répliques de chaque mobile i procède en deux phases :

- Parcourir l'arbre de bas en haut et sélectionner les nœuds j tels que $LCMR_{i,j} \geq Smax$ et le nombre de sites $n < Nmax$.
- Si n est toujours inférieur à Nmax, l'algorithme parcourt les nœuds de niveau inférieur à L de haut en bas et dans l'ordre décroissant de $| LCMR_{i,j} - Smax |$.

Lorsqu'un client ne désire pas avoir l'adresse d'une réplique quelconque mais celle d'une réplique qui vérifie certains critères, une solution est de donner à chaque classe de répliques un identificateur, c'est une perspective indésirable. Le modèle ***Globe*** propose d'étendre le service de localisation pour qu'il supporte des prédicats en terme de valeurs d'attributs et ce

en associant des plans de propriétés (*Property Maps* PM) aux adresses de contact **[24]**. Un PM est une chaîne de 64 bits dont chaque bit représente une propriété. Lorsqu'un utilisateur cherche un mobile, il met à 1 ou à 0 les bits du PM selon les propriétés qu'il désire.

IV.4. Réplication adaptative des données (*Adaptative Data Replication ADR* ***)***

L'algorithme ADR permet de déterminer de manière dynamique l'ensemble optimal (en terme de coût de communication) des sites de réplication pour un objet lorsque son profil lecture / écriture change dans une architecture structurée en arbre.

Soit R l'ensemble courant des sites de réplication d'un mobile x. Une lecture de x est réalisée dans la réplique de R la plus proche alors qu'une écriture de x met à jour toutes les répliques dans R. Voici d'abord deux définitions.

Définition1 : Un site i est un $\overline{R}$-voisin si $i \in R$ mais a un voisin j tel que $j \notin R$ (deux sites sont voisins s'ils ont le même parent).

Définition2 : Si i n'est un ensemble singleton, i est un R-bordure s'il est une feuille dans le sous graphe dessiné par R.

L'algorithme met à jour R pour chaque mobile x après une période T. Des sites spécifiques du réseau effectuent trois tests, l'expansion, la contraction et le switch test. R s'étend si la lecture de x augmente et se contracte lorsque c'est son écriture qui augmente. Le test d'expansion se fait pour chaque $\overline{R}$-voisin i. Il invite chacun de ses voisins j tels que $j \notin R$ de rejoindre R si le nombre de lectures qu'il reçoit de j durant une période T est supérieur au nombre d'écritures qu'il reçoit à partir des mobiles qu'il héberge ou à partir des mobiles d'un de ses voisins différent de j. Le test de contraction est exécuté par chaque R-bordure i. Il demande à ses voisins j de R la permission de quitter R si le nombre d'écritures que reçoit i de j durant une période T est supérieur de lectures qu'il reçoit. Si i est un $\overline{R}$-voisin et un R-bordure alors il exécute d'abord le test d'expansion et si ce test échoue, il exécute le test de contraction. Enfin, le switch test est exécuté, lorsque i est le seul site de R et que le test d'expansion échoue, i demande à un site n d'être le nouveau site singleton de R si le nombre de requêtes reçues par i depuis n durant la période T est supérieur au nombre de requêtes reçues par i de tous les autres sites.

Cet algorithme converge quelque soit la configuration initiale après un nombre de périodes T qui est borné par le diamètre du réseau.

V. POINTEURS EN AVANT (*Forwarding Pointers* FP)

Lorsque les déplacement d'un mobile sont plus fréquents que les appels qu'il reçoit, le coût des mises à jour est très important. Aussi, il est intéressant d'effectuer sélectivement les mises à jour et de rediriger les appels en utilisant les FP.

V.1. Architecture à deux niveaux

L'entrée d'un objet x n'est pas mise à jour à chaque fois qu'il se déplace. A la place, un FP qui va de l'ancienne VLR à la nouvelle est créé. La recherche d'un objet mobile x commence par la HLR de x qui contient l'adresse de la première VLR que x a visité et continue en suivant la chaîne de FPs qui conduit à la position courante de x. La longueur de la chaîne ne doit pas dépasser un maximum K qui est à déterminer. Si c'est le cas, l'adresse de la VLR de x doit être mise à jour au niveau de sa HLR et la chaîne de FP remise à nil.

La stratégie des FPs en opposition à la réplication est plus utile pour les mobiles dont les appels sont beaucoup mois fréquents que ces déplacements.

Une méthode pour déterminer si l'adresse de x dans sa HLR doit être mise à jour ou non de manière dynamique est le schéma de l'ancrage local (*local anchoring scheme* LA). Pour chaque mobile x, une VLR qui est près de lui est choisie pour être son LA qui maintient un pointeur vers la VLR courante de x. La HLR de x contient son LA. Ainsi, pour localiser un mobile x, la HLR de x est d'abord contactée, elle renvoie l'appel vers le LA de x qui contient un FP vers la VLR courante de x. La HLR met à jour ensuite l'adresse de la VLR courante de x.

V.2. Architecture hiérarchique

Dans le chapitre précédent, nous avons vu qu'il y a deux modèles de schéma hiérarchique. La technique des FP est adaptée aux deux modèles :

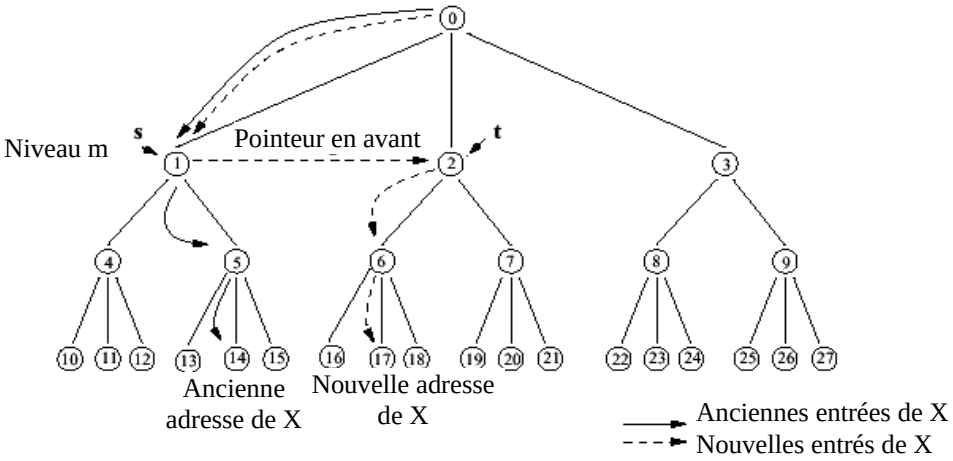
- Cas où les nœuds internes contiennent des pointeurs :

Le déplacement d'un mobile x du site i au site j implique la mise à jour des bases de données de i vers LCA(i,j) et de LCA(i,j) vers j. Afin réduire ce nombre, un FP est défini de S vers T où S est un ancêtre de i au niveau m inférieur au niveau du LCA(i,j) et T un ancêtre de j au même niveau m. La prochaine recherche se fera alors de $i \rightarrow S \rightarrow T \rightarrow j$ (voir Figure 2.3).

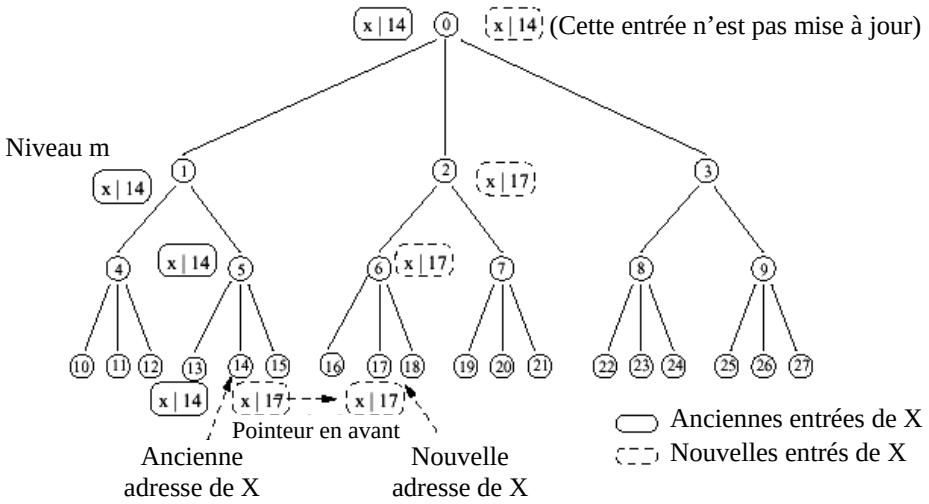
- Cas où les nœuds internes contiennent des adresses :

Les entrées de x sont mises à jour jusqu'au niveau m où un FP pointe directement vers la nouvelle adresse de x (voir Figure 2.4).

Pour réduire la taille des chaînes de pointeurs, une variation du cache est proposée : Après un appel vers x, sa position actuelle est mise en cache dans le premier nœud de la chaîne.
La base de données hiérarchique doit être mise à jour si le nombre de pointeurs de la chaîne ou la distance entre le premier et dernier nœud de la chaîne dépasse un seuil prédéfini.



- Figure 2.3 -



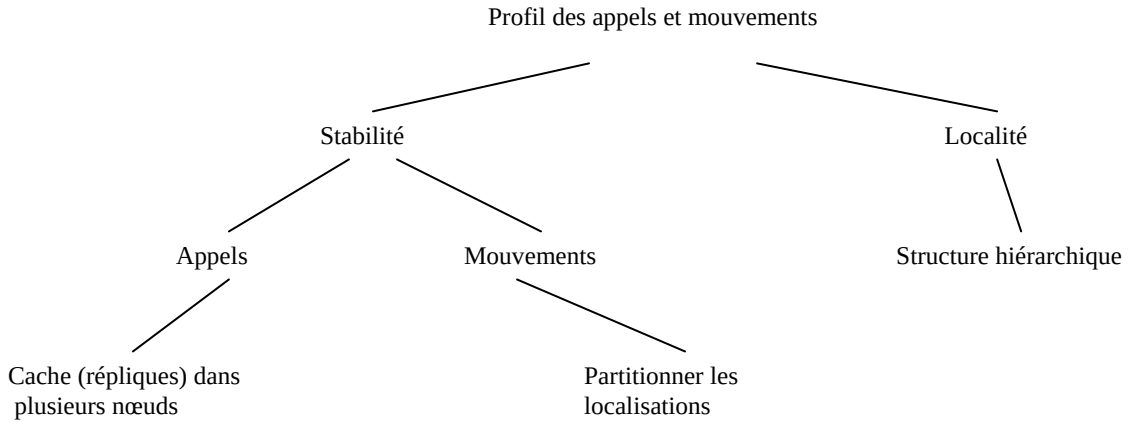
- Figure 2.4 -

VI. TAXONOMIE DES TECHNIQUES DE GESTION DE LA LOCALISATION [1]

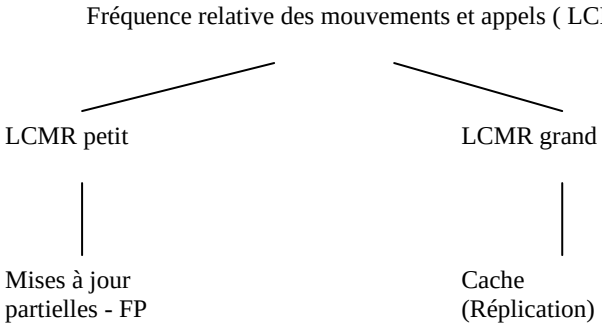
Les techniques proposées dans ce qui a précédé sont basées sur l'exploitation des informations concernant les appels et déplacements d'un mobile. De plus deux caractéristiques sont considérées, stabilité et localité :

- Stabilité des appels reçus : Ils proviennent du même ensemble de sites.
- Stabilité des mouvements : Un mobile a tendance à se déplacer dans un ensemble spécifique de sites.
- Localité des appels : Un mobile reçoit souvent des appels de sites proches.
- Localité des déplacements : Un mobile a tendance à se déplacer dans son voisinage.

Un autre facteur déterminant dans le choix d'une technique de localisation est la fréquence relative des appels et mouvements exprimée en CMR (*Call to Mobility Ratio)* qui détermine l'efficacité de ces méthodes. Les Figures 2.5 et 2.6 définissent suivant la localité, la stabilité des profils de mouvements et appels des mobiles (voir Figure 2.5) et le CMR des mobiles (voir Figure 2.6), la technique à adopter pour la gestion de la localisation. Ces techniques sont orthogonales, elles peuvent être donc combinées.

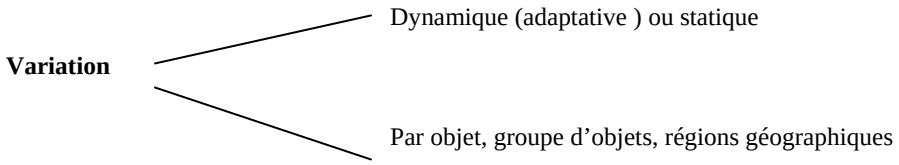


- Figure 2.5 -



- Figure 2.6 -

En plus de développer des techniques pour la sauvegarde de la localisation, les modèles de mouvement peuvent être utilisés pour la recherche de la position d'un objet lorsque l'information sauvegardée n'est pas récente ou précise. Exemple : les recherches peuvent être effectuées suivant les probabilités de trouver un objet dans une zone. Basée sur la vélocité et l'information de direction du client mobile x, cette technique estime la zone où x peut résider.
Enfin, le paramètre important de n'importe quel modèle d'appels ou de mouvements est le temps, les modèles peuvent capter les changements du profil des appels et mouvements dans le temps et leurs fréquences. C'est pourquoi l'adaptation dynamique au taux de mobilité et au profil courants est une caractéristique très recherché dans les techniques de localisation.
Les techniques de localisation peuvent être développées par objet mobile ou pour un groupe suivant leurs emplacements géographiques ou leurs caractéristiques de mobilité et d'appels ou une combinaison des deux. (voir Figure 2.7)



- Figure 2.7 -

Les tableaux 2.1 et 2.2 résument les variations des modèles à deux niveaux et hiérarchique respectivement et leurs propriétés.

Méthode	**Variations**	**Applicable lorsque:**
Cache Lorsque x est appelé par y, cacher la localisation de x dans la zone de y	Cache actif : mise à jour lors des mouvements	LCMR grand Stabilité des appels
	Cache paresseux : mise à jour lors des appels	
Réplication répliquer sélectivement l'adresse de x dans les zones où x reçoit le plus d'appels	Réplication par profil d'utilisateur : Des contraintes sur le nombre de répliques par site et le nombre de répliques par mobile	LCMR grand Stabilité des appels
	Ensemble de travail : Adaptative et distribuée, les sites de réplication sont déterminés de manière dynamique pour chaque mobile localement	
FP lorsque x se déplace, ajouter un FP de son ancienne adresse vers la nouvelle	Restreindre la longueur de la chaîne des pointeurs	LCMR petit

- Tableau 2.1 -

Méthode	**Variations**	**Applicable lorsque:**
Cache Lorsque x dans la zone i est appelé par y dans j, cacher dans un nœud du chemin j à LCA(i,j) un pointeur vers un nœud entre i et LCA(i,j)	A quel niveau de l'arbre sauvegarder le cache Quand mettre à jour le cache	LCMR grand Stabilité des appels
Réplication Répliquer sélectivement l'adresse de x dans un nœud interne et / ou feuille de l'arbre		LCMR grand Stabilité des appels
FP Lorsque x se déplace de i vers j, au lieu mettre à jour les Bds de i vers LCA(i,j) et de LCA(i,j) vers j, mettre à jour les BDs jusqu'au niveau m et ajouter un FP du niveau m ancêtre de i vers le niveau m ancêtre de j	Quand et comment purger' les FP Fixer le niveau m	LCMR petit
Partitions Diviser les zones en ensembles (partition) telles que le mobile se déplace fréquemment dans une partition et rarement entre les partitions Garder trace de l'ensemble où réside le mobile et non son adresse exacte		Stabilité des déplacements

- Tableau 2.2 -

Enfin, un autre paramètre qui affecte le choix de la stratégie de localisation est la topologie du réseau : Comment les sites sont géographiquement connectés et surtout à quel point la technique choisie permet l'extension du réseau en objets et en opérations sur les bases de données.

Les stratégies de localisation sont évaluées sur la base de deux critères :

- ✓ En terme de base de données : Les objectifs sont de réduire le nombre total des recherches et mises à jour des bases de données, leurs tailles et temps d'accès et la latence de ces opérations.
- ✓ En terme de communication : Les objectifs sont de réduire le nombre total des messages, les distances parcourues, le nombre de bits générés et la somme du trafic entre les sites.

VII. PRECISION DE L'INFORMATION DE LOCALISATION

Les techniques présentées ci-dessus font toujours référence à l'adresse courante d'un mobile. C'est à dire que lorsqu'un mobile met à jour sa position courante il donne sa position exacte et c'est celle-ci qui sera portée dans la base de données. Pour des raisons d'efficacité dans la sauvegarde des mises à jour et recherches dans les bases de données de localisation, l'information sauvegardée peut ne pas être précise **[1]**. Ainsi, au lieu de stocker la position courante d'un mobile x, c'est une région de plusieurs sites qui est sauvegardée et pour retrouver l'adresse de x une recherche est lancée. Une autre idée est de ne pas mettre à jour les bases de données après chaque mouvement de x. Ici aussi une recherche de x à partir de l'adresse sauvegardée s'impose.

VII.1. Granularité de l'information de localisation

La granularité de l'information diffère selon le nombre de cellules qu'elle couvre. Dans l'architecture cellulaire, ceci se traduit par combien et quelles cellules sont couvertes par chaque région d'inscription : Si elle couvre un petit nombre de zones, le coût des mises à jour est grand alors que si elle couvre un grand nombre de cellules , c'est le coût des recherches qui augmente. Aussi, la définition de la taille et forme de chaque région est formulée en un problème d'optimisation combinatoire. Son objectif est de minimiser le coût des recherches et mises à jour dans cette région.

VII.2. Fréquence des mises à jour

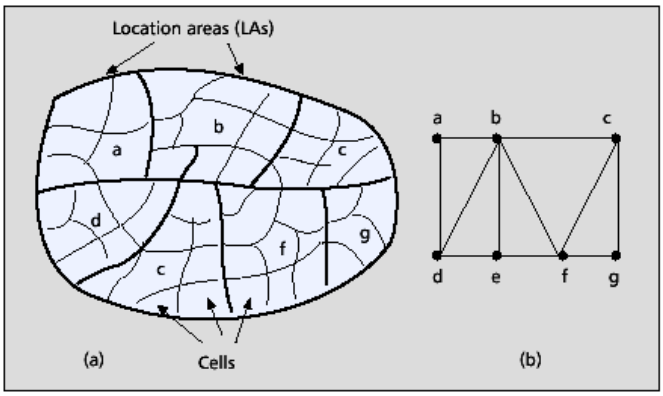
Pour réduire l'incertitude de localisation, chaque mobile doit mettre à jour sa localisation de temps en temps. La procédure de mise à jour commence par un message envoyé par le mobile qui est suivi par des procédures de signalement qui mettent à jour la base de données. Les algorithmes de mise à jour de la localisation peuvent être divisés en deux groupes : statiques et dynamiques. Dans les algorithmes statiques, la mise à jour est déclenchée sur la base de la topologie du réseau. Dans les algorithmes dynamiques, la mise à jour est basée sur le profil d'appels et de mobilité d'un utilisateur. Dans ce qui suit, un résumé des différents schémas de mise à jour de la localisation présentés dans la littérature sera proposé **[39,41]** :

1) *Stratégie basée sur les zones de localisation (Location Areas* LAs) : Dans ce schéma toute la zone que couvre le réseau est partagée en un nombre de LAs, chacune contient un groupe de cellules. Toutes les stations de base de la même LA diffusent périodiquement l'identificateur ID de leur LA. Chaque mobile compare le LA ID qu'il a enregistré avec celui diffusé. La mise à jour est déclenchée lorsque les deux IDs sont différents.

2) *Stratégie basée sur les LAs sélectives* : Un client peut traverser un nombre de LAs par jour. Toutefois, il ne reste que très peu de temps dans certaines LAs. Aussi, la mise à jour ne se fera que dans quelques LAs.

Un modèle analytique dans lequel l'interconnexion des LAs est caractérisée par un modèle graphique (voir Figure2.8) est introduit. Le modèle de mouvement utilisé est Markovien. Un algorithme génétique est utilisé pour obtenir les LAs où il faut effectuer une mise à jour. Le temps de résidence dans une LA suit une distribution géométrique. Pour une probabilité de résidence petite dans certaines LAs et un coût de mise à jour élevé, les résultats montrent que ce schéma donne un coût de localisation management plus réduit que le schéma basé sur les LAs conventionnel.

Pour l'implémentation, l'information concernant la probabilité de transition et le temps de résidence est requise. Pour estimer la probabilité de transition entre les LAs pour un mobile, son mouvement le long d'une journée est observé durant de longues périodes de temps.



- Figure 2.8 -

3) Stratégie basée sur le profil : Le but de ce schéma est de réduire le coût de mise à jour en tirant avantage du profil de mobilité d'un mobile. Le réseau maintient un profil pour chaque utilisateur qui inclut une liste séquentielle des LAs dans lesquelles il est susceptible de se trouver. Cette liste est triée selon la probabilité d'y trouver l'objet. Lorsqu'un appel arrive, les LAs de la liste sont interrogées séquentiellement. La mise à jour est nécessaire lorsque l'objet n'est dans aucune des LAs de la liste.

Pour l'implémentation, chaque mobile doit garder une liste valide qui correspond à un intervalle de temps particulier.

4) Stratégie basée sur le mouvement : Chaque mobile garde trace du nombre de changements de cellules qu'il effectue durant son mouvement. La mise à jour est déclenchée lorsque ce nombre dépasse un seuil de mouvements prédéfini. Ce schéma permet aussi une sélection dynamique du seuil de mouvements par mobile.

Pour l'implémentation, le mobile n'a besoin que d'un compteur qui compte le nombre de changements de cellules. Le compteur est remis à zéro lorsqu'il atteint le seuil de mouvements. Le *CIC* (Cell Identifier Code) peut être utilisé. A chaque cellule est affecté un code, qui n'est pas nécessairement unique. Ce code est utilisé pour identifier l'orientation d'une cellule relativement aux autres cellules d'une même LA. Chaque cellule diffuse périodiquement son code d'identification. Le mobile utilise cette information pour pouvoir décider de la mise à jour.

5) Stratégie basée sur le temps : Chaque mobile met à jour sa localisation toutes les T unités de temps. Un modèle analytique est proposé pour étudier ce schéma : En supposant une distribution gaussienne de la probabilité de localisation d'un mobile et une arrivée des appels qui suit une loi de Poisson, la période de mise à jour qui minimise le coût de recherche et de

mise à jour est déterminée. Les résultats montrent que ce schéma est substantiellement meilleur que le schéma basé sur les LAs classique.

Une variation du schéma basé sur le temps appelée *schéma à seuil adaptatif* est proposée. Ici, le mobile transmet un message de mise à jour toutes les T' unités de temps où la valeur de T' n'est pas constante.

6) Stratégie basée sur la distance : Chaque mobile garde la distance qu'il a parcouru (en nombre de cellules) depuis la dernière mise à jour et émet un signal de mise à jour lorsque cette distance dépasse un certain seuil D.

Cette technique semble donner de meilleurs résultats que les précédentes même si elle reste plus difficile à implémenter. Deux méthodes sont proposées **[27]** pour calculer le seuil optimal D. L'une a pour objectif de minimiser le coût de la gestion de la localisation et l'autre celui de minimiser la latence dans la recherche d'un client mobile et ce de manière dynamique.

7) Stratégie basée sur la distance prédictive : Dans ce schéma, le mobile reporte sa localisation ainsi que sa vitesse durant la procédure de mise à jour. Ainsi, le réseau détermine la fonction de densité de la probabilité de localisation du mobile qui est utilisée pour prédire sa position future. Le mobile compare périodiquement sa position courante et la position prédite. Si la différence dépasse le seuil de distance, la mise à jour est déclenchée. Lorsque un appel arrive, la recherche commence par la position prédite puis une recherche basée sur la plus petite distance d'abord (qui sera présentée ultérieurement) est lancée jusqu'à trouver le mobile.

8) Stratégie basée sur l'état : Dans ce schéma, le mobile décide d'effectuer une mise à jour d'après son *état*. Les informations concernant son état incluent le temps écoulé ou le nombre de cellules parcourues depuis la dernière mise à jour, la distance entre la position courante et la dernière position enregistrée ou d'autres critères. Ainsi, maintenir des informations différentes sur l'état d'un mobile correspond à des schémas de mise à jour différents.

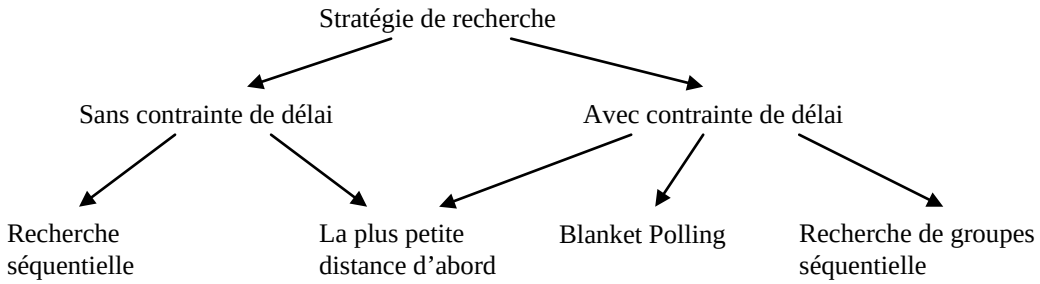
Ce schéma analysé avec comme informations retenues la position courante et le temps écoulé depuis la dernière mise à jour donne des résultats qui montrent que ce schéma améliore de 10% le coût en comparaison avec le schéma basé sur le temps.

9) *La mise à jour LeZi* : Basé sur un algorithme de compression, cet algorithme peut être considéré comme un *schéma basé sur le chemin* dans lequel c'est l'historique du mouvement d'un mobile x qui est envoyé dans un message de mise à jour et non sa position courante. L'historique du mouvement (HM) est une liste d'identificateurs de cellules (ou de LAs) que le

mobile a parcouru depuis sa dernière mise à jour. Aussi, le HM peut être considéré comme une part du profil du mobile qui est maintenu dans le réseau dans une forme compressée.
Une autre approche utilisée pour signaler la mise à jour est de sélectionner un sous ensemble de sites, la mise à jour est effectuée lorsqu'un objet entre dans cet ensemble. Pour une topologie arbitraire du réseau trouver l'ensemble optimal est un problème NP complet. Ceci dit, des solutions sont proposées pour des topologies particulières. Cette stratégie est statique et globale : une fois l'ensemble de sites déterminé, il reste fixe et valable pour tous les objets du réseau.

VII.3. Procédures de recherche

Lorsque l'information n'est pas précise (§ VII.1 et VII.2), une recherche est lancée pour retrouver l'adresse exacte d'un mobile. La procédure identifie d'abord l'ensemble des sites où ce mobile peut être puis les interroge.
Cet ensemble dépend essentiellement de la granularité de l'information et de la politique de mise à jour.
Dans chaque cycle de recherche, des signaux de recherche sont envoyés à toutes les cellules où le mobile peut se trouver. Tous les mobiles écoutent ces messages, seul le mobile recherché envoie un message de réponse. Dans chaque cycle de recherche, il y a une période *timeout*. Si le mobile répond avant ce timeout, la recherche est arrêtée. Sinon, un autre groupe de cellules est choisi pour le prochain cycle de recherche.
Pour éviter de perdre des appels, un mobile doit être localisé avec une contrainte de temps admissible. Le *délai maximum de recherche* correspond au nombre maximum de cycles de recherche. Exemple : Si le délai maximum de recherche est égal à 1, alors un mobile doit être localisé avec une seule itération de recherche.
Comme une bande passante radio est consommée durant la recherche, le coût de recherche est proportionnel au nombre de cycles de recherche et au nombre de cellules interrogées dans chaque cycle. La zone de recherche dépend de l'information donnée par la fonction de mise à jour. Le coût de recherche peut être réduit en essayant de déduire la prochaine position d'un mobile.
Dans cette section, un résumé des différentes stratégies de recherche proposées dans la littérature sera présenté **[39,41]**. Une classification de ces stratégies est présentée dans la Figure 2.9.



- Figure 2.9 -

1) *Blanket Polling :* Dans ce schéma, si le mobile est localisé dans une LA, toutes les cellules de cette LA sont interrogées simultanément. Aussi un seul cycle de recherche est nécessaire pour déterminer la position exacte du mobile. L'inconvénient de cette stratégie est que le coût de recherche est très élevé quand le nombre de cellules dans une LA est très grand.

2) *La plus petite distance d'abord :* La recherche d'un mobile commence par la dernière position à laquelle il a été enregistré et va vers les autres cellules suivant la petite distance.

La distance est mesurée en terme nombre de cellules à partir de la position (adresse) enregistrée à la mise à jour.

Si un seuil de mise à jour (distance ou mouvement) est utilisé alors le nombre de cycles de recherche pour trouver un mobile est limité. Dans ce cas, il faut regrouper des cellules avec des distances différentes dans les cycles de recherche.

Pour illustrer le mécanisme de cette stratégie de recherche, on considère la Figure 2.10-b dans laquelle les cellules ont une topologie hexagonale. Supposons que la cellule libellée par 0 est celle où le mobile x a effectué sa dernière mise à jour. Supposons aussi que c'est la mise jour basée sur la distance qui est utilisée avec un seuil de distance D=3 : la mise à jour est effectuée lorsque x entre dans une cellule libellée par 4.

Sans aucune contrainte de délai de recherche, la séquence de recherche est {{0},{1},{2},{3}}, d'abord la cellule 0 est interrogée. Si aucune réponse de x n'est reçue après le timeout, toutes les cellules libellées par 1 sont interrogées dans le second cycle de recherche. Ceci continue jusqu'à ce que x envoie une réponse à la station de base ou toutes les cellules ont été interrogées.

Dans le cas avec des contraintes de délai, des cellules avec des labels (distances) différents peuvent être interrogées en un cycle de recherche. Par exemple, si le délai de recherche maximum est 3, la séquence de recherche peut être {{0,1},{2},{3}}. Les cellules avec le label

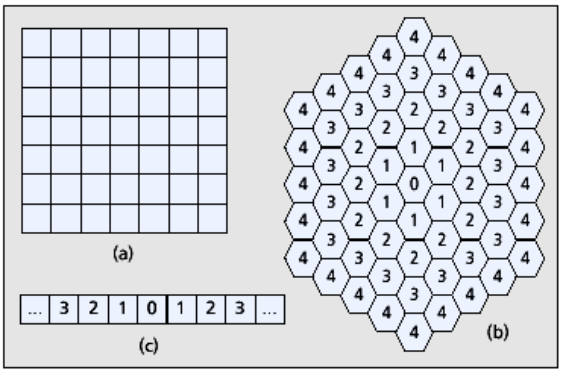
0 et 1 sont interrogées dans le premier cycle. Les cellules de label 2 et 3 sont interrogées dans le second et troisième cycle respectivement.

Cette stratégie est utilisée dans plusieurs schémas de mise à jour tels que le schéma basé sur le mouvement ou la distance.

3) *Recherche séquentielle basée sur la probabilité de localisation d'un mobile* : Dans ce schéma, la position courante d'un mobile est prédite suivant la distribution de probabilité de localisation de ce mobile. Les signaux de recherche ne sont envoyés qu'aux cellules où le mobile a une probabilité de s'y trouver et ce séquentiellement et dans l'ordre décroissant de probabilité.

Lorsqu'il y a un délai de recherche maximum, un groupe de cellules peut être interrogé dans chaque cycle de recherche. Dans ce cas, la séquence de recherche optimale calculée donne un coût de recherche minimum avec une contrainte de délai de recherche moyen.

Cette stratégie est utilisée dans le schéma basé sur le temps ou l'état.



- Figure 2.10 -

4) *Velocity Paging :* Cette stratégie vise à réduire le coût de recherche en réduisant la taille de la zone de recherche. Ceci est réalisé en groupant les mobile en *classes de vitesse* selon leurs vitesses de mouvement à la mise à jour. Lorsqu'un appel arrive, la zone de recherche est générée de manière dynamique d'après le moment de la dernière mise à jour et index de classe de vitesse.

Pour implémenter cette stratégie, les informations telles que la dernière position enregistrée du mobile, sa classe de vitesse et le moment de la dernière mise à jour sont nécessaires dans la base de données profil du mobile.

Lorsque la Velocity Paging est combinée avec l'algorithme de mise à jour basé sur le mouvement, les résultats indiquent que cette combinaison ne donne pas toujours une réduction de coût par rapport au schéma de mise à jour basé sur la LA avec la Blanket Polling. Aussi, un rang de rayons de cellule où cette technique peut être utilisée est déterminé.

5) *Ensemble Polling* : Lorsqu'il y a un grand nombre de requêtes de recherches (d'arrivées d'appels), le délai de recherche peut augmenter et les canaux de recherche se surcharger. L'Ensemble Polling est un modèle qui fait varier les techniques de recherche selon le nombre d'arrivées des appels. Il est formulé pour les mobiles comme une chaîne de Markov. Les requêtes de recherche suivent une loi de poisson. Le taux de service est supposé suivre une loi exponentielle.

Pour une probabilité de localisation donnée, plusieurs stratégies de recherche dont la Blanket Polling, la recherche séquentielle et la recherche de groupes séquentielle sont analysées et le délai moyen entre une requête de recherche et la détermination de la localisation du mobile est calculé.

Les résultats montrent que la Blanket polling donne le plus petit délai pour un petit taux de requêtes alors que la recherche séquentielle peut supporter un plus grand nombre de requêtes.

V. CONCLUSION

La gestion de la localisation est l'un des plus grands problèmes dans les réseaux mobiles car le nombre d'utilisateurs, services et programmes qui circulent dans ces réseaux augmente de manière vertigineuse. Aussi, la gestion de toutes ces ressources devient de plus en plus compliquée. Ce chapitre résume les techniques les plus importantes pour répondre à ce problème à savoir le cache, la réplication , les FP...etc. et ce suivant l'architecture choisie pour représenter la base de données de localisation. Il est vrai que l'architecture la plus utilisée actuellement est le schéma à deux niveaux mais en raison de sa non scalabilité (il ne favorise pas l'extension du réseau), les travaux de recherche se tournent plus vers le schéma hiérarchique pour les grands réseaux à étendue mondiale (les World Wide Networks).

Le choix des techniques de localisation est crucial quand on sait que le coût de la communication est toujours augmenté par le coût de la recherche. Il dépend de plusieurs paramètres dont le profil des appels et mouvements des mobiles.

I. INTRODUCTION

La seule façon d'augmenter la fiabilité d'un service, à part l'utilisation d'un hardware plus robuste, est la réplication. Pour rendre un service robuste, il peut être installé sur plusieurs serveurs identiques, chacun garde trace de l'état de ce service et effectue des opérations de lecture et écriture dessus. Ceci permet au système de fournir des informations et d'effectuer des opérations même si certaines machines tombent en panne. Malheureusement, la réplication implique un coût pour maintenir les serveurs consistants.
Organiser ces serveurs en quorums permet de garder la cohérence des données avec des coûts raisonnables.
Les systèmes de quorum ont été beaucoup étudiés comme une méthode pour maintenir la consistance dans les systèmes distribués. Le principe est d'identifier un certain sous ensemble spécial de nœuds appelé *quorum* tel que l'intersection de deux quorums dans le système est non nulle.
L'accès d'un nœud au système nécessite un accès à tous les nœuds de l'un des quorums de ce système. De même, une mise à jour dans le système nécessite la mise à jour de tous les nœuds de l'un des quorum de ce système. La propriété d'intersection assure que n'importe quel quorum contient au moins un élément qui est à jour. Aussi, la consistance du système est conservée dans le temps ainsi que la disponibilité de l'information.

II. DEFINITIONS

Soit U un ensemble non vide de nœuds. Le nœud fait référence à un site dans un système réparti ou à une copie d'un objet dans une base de données répliquée. Voici quelques définitions concernant les quorums **[5,13,29,34]**

II.1. Distance virtuelle :

La distance virtuelle entre deux nœuds a et b est le temps nécessaire pour envoyer un message d'une certaine taille de a vers b ou vice versa.

II.2. Quorum

Une collection d'ensembles est un ensemble de quorums sous U si :

- $\forall G \in Q,\ G \neq \varnothing$ et $G \subseteq U$.
- Minimalité : $\forall G,H \in Q,\ G \not\subset H$.

Les ensembles $G \in Q$ sont appelés quorums.

Soit U = {a,b,c,d}

Q = {{a,b},{b,c}} est un ensemble de quorums sous U.

Les nœuds $x \in Q$ sont appelés nœuds utilisés.

Soit Q_1 et Q_2 deux ensembles de quorums dans U. Q_1 domine Q_2 si :

- $Q_1 \neq Q_2$
- $\forall H \in Q_2, \exists G \in Q_1 / G \subseteq H$

Un ensemble de quorums Q sous U est dit dominé s'il existe un autre ensemble de quorums sous U qui domine Q. Si un tel ensemble n'existe pas alors Q est dit non-dominé.

Quorum complémentaire : Soit Q un ensemble de quorums dans U. L'ensemble de quorums complémentaire Q^c est un autre ensemble de quorums sous U tel que :

$\forall H \in Q^c, \forall G \in Q, \quad G \cap H \neq \varnothing$.

II.3. Coterie

Un ensemble de quorums est une coterie sous U si la propriété d'intersection est satisfaite :

$\forall G,H \in Q, \quad G \cap H \neq \varnothing$.

Une coterie Q sous U est dite dominée s'il existe une autre coterie sous U qui domine Q. Si une telle coterie n'existe pas alors Q est dite non-dominée.

Les coteries symétriques : Une coterie C sous U est symétrique ssi les propriétés suivantes sont vérifiées :

- Propriété de l'effort égal : $\forall P_i \in U, \ |\{j / i \in Q_j\}| = \beta$. C'est à dire que tous les nœuds appartiennent au même nombre de quorums β.
- Propriété de la taille égale : $\forall Qi, \ |Q_i| = \gamma$. Ce qui veut dire que tous les quorums ont la même taille γ.

Les k-coteries : Un ensemble de quorums C est une k-coterie sous U ssi les propriétés suivantes sont vérifiées

- La propriété de non intersection : $\forall$ h (<k) quorums distincts $Q_1,\ldots, Q_h \in C$ tels que $Q_i \cap Q_j = \varnothing$ (pour $1 \leq i \neq j \leq h$), $\exists Q \in C$ tel que $Q \cap Q_i = \varnothing$ ($1 \leq i \leq h$)
- La propriété d'intersection : $\forall$ k+1 quorums $Q_1,\ldots, Q_{k+1} \in C$, $\exists$ i,j ($1 \leq i \neq j \leq h$) tels que $Q_i \cap Q_j \neq \varnothing$.
- Propriété de minimalité : $\forall Q_i, Q_j \in C, \ Q_i \not\subset Q_j$.

Les k-arbitres : Un ensemble de quorums C est un k-arbitre ssi les conditions suivantes sont satisfaites :

- La propriété d'intersection : $\forall$ k+1 quorums distincts Q_1,...., $Q_{k+1} \in C$, $\bigcap_{i=1}^{k+1} Q_i \neq \varnothing$
- Propriété de minimalité : $\forall\ Q_i, Q_j \in C\ \ Q_i \not\subset Q_j$.

II.4. Bicoterie

La paire $B = (Q, Q^c)$ est appelée bicoterie sous U.

Si Q ou Q^c est une coterie alors B est appelée semicoterie.

Une bicoterie B=(W,R) est une wr-coterie ssi W est une coterie:

Supposons que $B_1 = (Q_1, Q^c_1)$ et $B_2 = (Q_2, Q^c_2)$ sont deux bicoteries sous U. B_1 domine B_2 si :

- $B_1 \neq B_2$ .ie : $Q_1 \neq Q_2$ et $Q^c_1 \neq Q^c_2$.
- $\forall\ H \in Q_2, \exists\ G \in Q_1 / G \subseteq H$
- $\forall\ H \in Q^c_2, \exists\ G \in Q^c_1 / G \subseteq H$

Une bicoterie Q sous U est dite dominée s'il existe une autre bicoterie sous U qui domine Q. Si une telle bicoterie n'existe pas alors Q est dite non-dominée.

III. CRITERES DE MESURE DE LA QUALITE DES QUORUMS

La qualité d'un système de quorums peut être évaluée par plusieurs critères : le nombre et la taille des quorums dans le système, la disponibilité du système et la charge que le système permet **[4,34,35,42]**.

III.1. Disponibilité (Availability)

En supposant que chaque élément peut tomber en panne avec une probabilité p, F_p est la probabilité que les éléments survivants ne forment pas un quorum. Cette probabilité de défaillance mesure à quel point le système est tolérant aux pannes. La disponibilité est la probabilité qu'au moins un quorum survive. $Disp(S) = 1-F_p$.

Il est souhaitable donc pour un bon système que F_p soit aussi petite que possible. Une bonne coterie a une petite probabilité de défaillance et donc une grande disponibilité.

Les coteries non dominées ont la plus grande disponibilité parmi les systèmes de quorums. Elles sont particulièrement résistantes aux défaillances et à la partition des réseaux.

III.2. Charge (Load)

La charge est la probabilité d'accéder au serveur le plus occupé dans le meilleur cas.

Une stratégie d'accès *w* est une règle qui affecte à chaque quorum une probabilité (fréquence) d'accès telle que la somme des fréquences de tous les quorums du système est égale à un.
Soit un site *v*. La charge de *v* pour la stratégie *w* est la somme des fréquences de tous les quorums auxquels appartient *v*. Ceci représente la fraction de temps où un élément est utilisé.
Pour un système de quorums S, La charge de S, L(S) est donnée par la stratégie qui donne la charge minimum de l'élément le plus occupé dans S. La charge mesure la qualité du système de quorums dans le sens suivant : Si la charge est faible, chaque nœud est rarement accédé, il peut donc effectuer d'autres tâches. Dans un bon système de quorums, les élément de U partagent une charge égale.

III.3. Capacité

La capacité est le nombre de requêtes que peut accepter S.
Soit a(S,k) le nombre d'accès aux quorums que S peut supporter durant une période de k unités de temps. La capacité de S $Cap(S) = Lim_{k\to\infty}\ a(S,k)$.
$Cap(S) = 1/L(S)$.

III.4. Complexité d'exploration (Probe complexity)

Comme les nœuds d'un système réparti ont une probabilité de tomber en panne, avant d'accéder à un quorum, une application doit déterminer si les nœuds du quorum sont fonctionnels ou non. Le nombre d'explorations qui doivent être effectués pour ce faire est appelé complexité d'exploration du quorum. Il est nécessaire pour une bonne coterie de réduire cette valeur.

III.5. Cardinalité

La cardinalité d'un quorum dans une coterie est une mesure importante puisqu'elle indique le coût de la communication pour contacter les éléments d'un quorum. Plus la taille du quorum est petite, plus la latence en communication est petite.

III.6. Délai

Soit C une coterie pour un réseau $G = \{V, E\}$ et dist(a,b) la plus petite distance entre deux nœuds a et b dans G. Le délai d'un nœud s est donné par : $delay(s, C) = \min_{Q \in C}\left\{\max_{v \in Q}\{dist(s, v)\}\right\}$.

On considère deux métriques différentes de délai : le maximum des délais entre tous les nœuds *max-delay* et la moyenne arithmétique des délais de tous les nœuds *mean-delay*.
Soit CG l'ensemble de toutes les coteries dans G, la *max-delay optimal coterie* D est une coterie telle que : $\max - delay(D) = \min_{C \in CG}\{\max - delay(C)\}$. De même, La *mean-delay optimal coterie* F est une coterie telle que : $mean - delay(F) = \min_{C \in CG}\{mean - delay(C)\}$.

Les coteries non-dominées sont meilleures par rapport aux autres systèmes de quorums en terme de tolérance aux défaillances et coût de communication. Une coterie non-dominée est un système de quorums qui minimise la taille des quorums et maximise le nombre de quorums tel que supprimer un élément d'un quorum ou ajouter un quorum change ce système en un système qui n'est plus une coterie.

IV. QUELQUES COTERIES

Soit m le nombre de nœuds dans le système.
Il existe plusieurs systèmes de quorums **[35]** dont :

IV.1. La coterie singleton

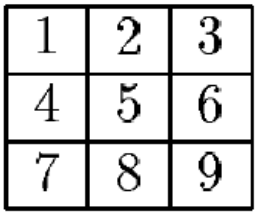
Une coterie Q = {G} qui contient un seul quorum est appelée la coterie singleton. Elle est non-dominée et est symétrique (β = 1 et γ = | G |). Sa charge est 1 et sa disponibilité est (1-p).

IV.2. Schéma basé sur la grille

Soit $m = n^2$ pour un entier n. Une grille n*n est construite. Les points de la grille sont numérotés de 0 à m-1. Le quorum S_i est défini par les points de la ligne et la colonne qui passent par le point i. La taille des quorums ainsi construits est $2*\sqrt{m}-1$ telle que l'intersection des quorums deux à deux est toujours égale à 1.
Si m n'est pas le carré d'un entier, une construction dégénérée de la grille est construite. Dans ce cas, la ligne/colonne partielle est complétée par des points d'une autre ligne/colonne.
Ce système est symétrique. La charge de système est $O(n^{-1/2})$ et sa probabilité de défaillance tend vers 1 lorsque n est très grand.

Exemple

Dans la Figure 3.1, l'ensemble {4,5,6,2,8} est un quorum construit autour du nœud 5.



- Figure 3.1 -

IV.3. Schéma basé sur la grille étendu

Soit m = n^3 pour un entier n. Une grille n*n*n est construite avec des points qui représentent les serveurs de localisation. Un quorum est formé en prenant tous les points de la grille qui appartiennent à trois plans mutuellement perpendiculaires dans la grille à trois dimensions. La taille des quorums est $O(N^{2/3})$.

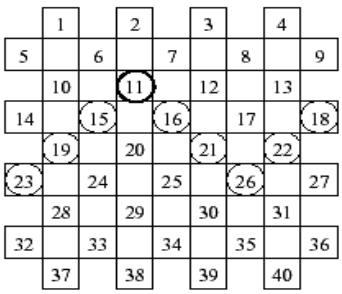
Avec une telle construction, chaque trois quorums ont au moins un serveur de localisation en commun.

IV.4. Schéma en billiard (Billiard Quorums on the Grid)

Dans le schéma basé sur la Grille, pour un nœud de la grille, le quorum est construit par l'union des lignes horizontale et verticale auxquelles appartient ce nœud. Dans ce schéma **[36]**, le choix des segments de lignes associés à un point peut être différent. Au lieu de choisir les lignes horizontales et verticales, ce schéma propose à partir d'un nœud i de dessiner des segments en diagonale avec des déplacement de ±1 case (voir Figure 3.2).Les segments de lignes qui forment un quorum autour du nœud i doivent commencer de l'extrême gauche (ou bas) de la grille à partir de n'importe quel nœud et arriver à l'extrême droite (ou haut) de la grille à n'importe quel nœud telles que lorsque un segment passe par i , il change de direction.

Ce schéma vérifie la propriété de l'effort égal. La taille des quorums est $\sqrt{2}\sqrt{N}$.

Exemple



- Figure 3.2-

IV.5. Le schéma de vote (Voting scheme)

A chaque membre de U est attribué un vote.

Tout système de sous-ensembles de U tel que la somme des votes de chaque sous ensemble est la majorité des votes dans U est un système de quorums.

Exemple :

Soit un univers U tel que | U | = n avec n un nombre impair.

La collection de tous les sous-ensembles de U dont la taille est de $\frac{n+1}{2}$ est une coterie.

Cette coterie est appelée 'majority coterie'. Elle est non-dominée, symétrique (γ = [n/2]+1, $\beta = \binom{n-1}{[n/2]}$). Elle a une très grande tolérance aux pannes. Sa charge est de [n+1/2*n] ≈ ½. Sa probabilité de défaillance est $e^{-\Omega(n)}$.

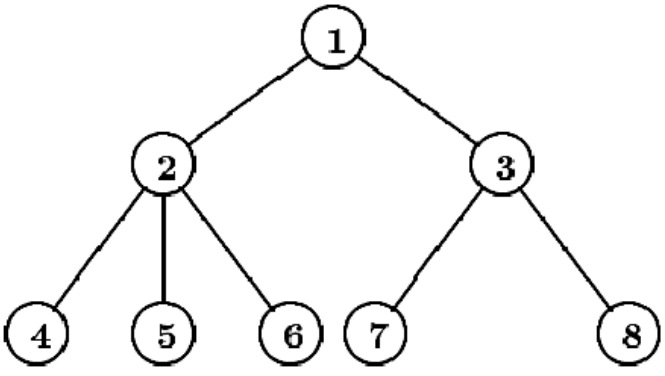
Lorsqu'un poids est associé à chaque vote, la coterie est appelée *majorité pondérée*.

IV.6. Le schéma en arbre

Les nœuds de U sont organisés en une structure logique en forme d'un arbre binaire **[5]**. Un quorum est formé en incluant tous les sites le long du chemin qui va de la racine jusqu'à une feuille de l'arbre. Si un site x est défectueux dans ce chemin le quorum sera formé en substituant x par les sites le long des deux chemins partant des fils de x vers les feuilles de l'arbre. Cette coterie est non-dominée. La taille du quorum est [$\log_2$ (n)].

Exemple :

U = {1,2,3,4,5,6,7,8}



- Figure 3.3 -

C = {{1,2,4},{1,2,5},{1,2,6},{1,3,7},{1,3,8}} est une coterie.

La construction en arbre permet un degré comparable de disponibilité que l'algorithme 'Quorum majority' avec un coût de communication moindre. Elle est meilleure pour les grands réseaux et permet une solution extensible à cause de son coût de communication et de son grand niveau de disponibilité.

IV.7. Schéma de la roue

Il est défini pour des coteries non-dominées. Dans ce schéma, les éléments de U sont arrangés en une roue où un élément est placé au centre et le reste sur la jante. La coterie est définie en formant des ensembles avec l'élément du centre et un élément de la jante avant de regrouper les éléments de la jante dans un autre ensemble.

Exemple :

Soit U = {1,2,3,4,5,6} où le nœud 1 est le centre de la roue. Dans le schéma da la roue, la coterie C est définie comme suit :

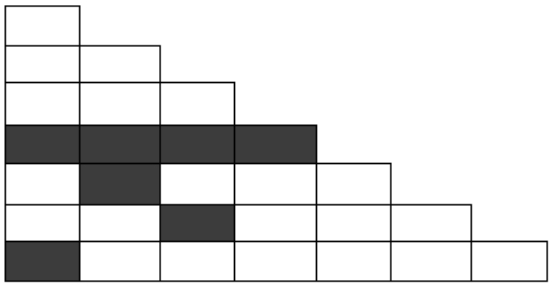
C={{1,2},{1,3},{1,4},{1,5},{1,6},{2,3,4,5,6}}

IV.8. Système triangulaire

Les éléments sont arrangés en d lignes tels que la ligne i contient i éléments. Un quorum est formé d'une ligne complète et d'un représentant de chaque ligne j > i. Cette construction est une coterie non-dominée.

Exemple

Dans la Figure 3.4, les case en couleur forment un quorum qui est l'union de la ligne 4 et un représentant des lignes en dessous, à savoir : la case (5,2) de la ligne 5, la case (6,3) de la ligne 6 et la case (1,7) de la ligne 7.



- Figure 3.4 -

IV.9. Schéma basé sur les plans projectifs finis

Les N nœuds sont divisés en N quorums $L_1,...,L_N$ tels que :

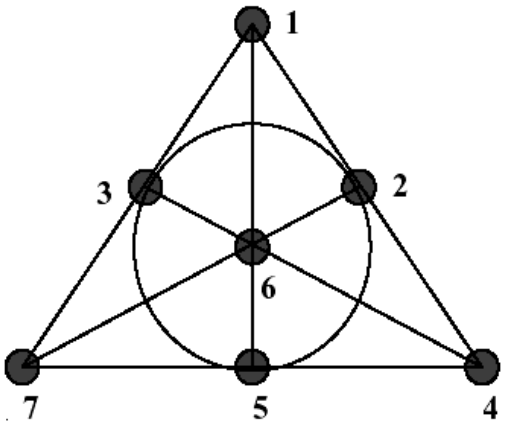
- $i \in L_i \quad 1 \leq i \leq N$
- $L_i \cap L_j \neq \varnothing \quad 1 \leq i,j \leq N$
- $| L_i |$ = m+1 pour $1 \leq i \leq N$ où m est appelé ordre du plan projectif fini avec N = m^2+m+1. D'où m est approximativement $\sqrt{N}$
- Chaque i $1 \leq i \leq N$ est contenu dans m+1 quorums.

Exemple

Dans la Figure 3.5, N=7 et donc m=2.

La taille de chaque quorum est m+1=3.

i	L_i
1	{1, 2, 4}
2	{2, 3, 5}
3	{3, 4, 6}
4	{4, 5, 7}
5	{5, 6, 1}
6	{6, 7, 2}
7	{7, 1, 3}



- Figure 3.5 -

IV.10. Crumbling Walls

Ce schéma est une généralisation de la construction triangulaire et de la roue **[14,15]**. Les éléments sont arrangés en lignes et un quorum est l'union des éléments de toute une ligne et d'un représentant d chacune des lignes au-dessous de cette ligne.

Soit n = $(n_1,...,n_d)$ tel que $\sum_{i=1}^{d} n_i = n$.

Soient $U_1,..,U_d$ des ensembles non vides et disjoints de l'univers U tels que $| U_i | = n_i$.

CW<n> = { $U_i \cup \{u_{i+1},\ldots,u_d\}$:$u_j \in U_j$ pour j = i+1,…,n} est l'ensemble des Crumbling Walls défini par n.

L'ensemble U_i est appelé la $i^{ème}$ ligne et n_i sa taille.

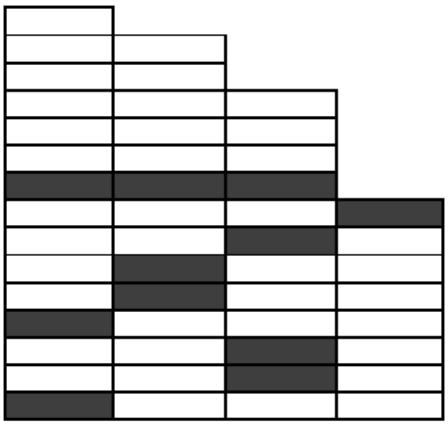
- CW< n_1,...,n_d > est une coterie ssi $n_i \geq 2 \ \forall \ 2 \leq i \leq d$.
- CW< n_1,...,n_d > est une coterie non-dominée ssi $n_1 = 1$ et $n_i \geq 2 \ \forall \ 2 \leq i \leq d$.

Un cas particulier des CW est Le logarithmic-growth crumbling wall. Ce modèle est un CW tel que le nombre de nœuds dans chaque ligne i est [1+Log_2 i]. Sa charge est optimale et est de O(1/Log n). La taille d'un quorum est $\approx \log n - \log(\log n)$.

Le logarithmic-growth crumbling wall est un bon candidat pour une construction de choix en pratique qui permet une grande disponibilité, de petits quorums et une simplicité de structure.

Exemple

Dans Figure 3.6, un quorum est construit avec les éléments de la ligne 7 et un représentant des lignes 8,..,15.



- Figure 3.6 -

V- APPLICATION DES COTERIES

Les systèmes de quorums sont un outil de base qui permet une coordination uniforme et fiable dans les systèmes répartis. Ils ont été utilisés dans plusieurs applications dans les systèmes répartis telles que l'exclusion mutuelle, la duplication de données, la dissémination sélective d'informations, nommer les serveurs, le contrôle de l'accès distribué et signatures, le consensus distribué ou encore l'élection de leader **[5]**.

L'exclusion mutuelle : Le concept de l'intersection des quorums traduit l'essence de l'exclusion mutuelle dans les systèmes répartis. Pour entrer en section critique, un site doit avoir la permission d'un quorum de sites dans le système. Si tous les sites du quorum lui permettent d'accéder à la section critique alors il y accède. Comme tous les quorums pris deux à deux ont au moins un site en commun (propriété d'intersection) l'exclusion mutuelle est garantie. La propriété de minimalité n'est pas nécessaire pour la correction de l'algorithme mais est utile pour son efficacité.

La performance de l'exclusion mutuelle est mesurée par le nombre de messages échangés par une section critique et le délai de synchronisation.

Avec la solution de Lamport la complexité est de 3*(n-1) avec n le nombre de sites dans le système réparti. L'algorithme de Maekawa réduit la complexité à O($\sqrt{N}$) **[38]**. Les algorithmes d'exclusion mutuelle basés sur les coteries et qui sont une généralisation de l'algorithme de Maekawa construisent leur coterie différemment pour réduire la complexité des messages ou augmenter la tolérance du système aux défaillances. **[28]** propose une solution basée sur la construction en arbre logique qui donne des quorums dont la taille est $O(Log_2 N)$. Sa solution est tolérante aux défaillances de sites et de communication mais requiert un plus grand nombre de messages dans ce cas qui peut aller jusqu'à $\frac{N+1}{2}$.

[33] propose un système qui réduit le délai de synchronisation de 2*T à T tout en gardant une complexité de messages de l'ordre de $O(K)$ où K est la taille d'un quorum est qui peut être aussi petite que $O(Log_2 N)$. Ce qui en cas de grande charge du système, divise le temps d'attente pour entrer dans une section critique par deux.

Une des généralisations de l'exclusion mutuelle est l'utilisation des quorums dans l'exclusion mutuelle distribuée d'ordre k+1 **[32]**. Dans l'exclusion mutuelle d'ordre k+1, jusqu'à k processus sont permis d'accéder simultanément à une ressource. Ainsi soit k+1 ensembles de sites qui accordent la permission d'accès à la ressource, alors il doit exister au moins deux parmi les k+1 ensembles avec une intersection non vide. Le problème peut être formulé comme suit :

Propriété d'intersection : $\forall$ (k+1) ensembles $S_1,...,S_{k+1}$, $\exists$ S_i, S_j ($i \neq j$) / $S_i \cap S_j \neq \varnothing$.pour $i,j \in \{1,2,...,k+1\}$.

Propriété de non intersection : $\exists$ k ensembles $S_1,...,S_k$, $\forall$ $i,j \in \{1,2,...,k\}$, $i \neq j$ $S_i \cap S_j = \varnothing$.

Une simple approche pour résoudre l'exclusion mutuelle dans un système réparti est d'utiliser des quorums de taille N/2+1. Donc, pour l'exclusion mutuelle d'ordre (k+1), il suffit de prendre des quorums de taille $\left(\frac{n}{k+1}\right)+1$.

Une autre approche pour la résolution du même problème est de considérer k cas de toute solution d'exclusion mutuelle. Un processus qui souhaite (k+1) accès exclusifs à la ressource demande la permission à chacun des k cas. Ceci assure la propriété d'intersection dès lors que n'importe quel ensemble de (k+1) quorums choisi se compose au moins de deux dans le même cas de solution d'exclusion mutuelle, d'où une intersection non vide. La propriété de non intersection est vérifiée si chaque processus choisi un quorum des différents cas de solutions.

Le concept du k-arbitre **[30]** est un mécanisme général pour l'exclusion mutuelle du type 'h parmi k' lorsqu'il existe k ressources identiques et chaque processus peut en demander à tout instant un nombre quelconque sachant que chaque exemplaire de la ressource ne peut être utilisé que par un processus à la fois. Une solution basée sur le k-arbitre est proposée pour résoudre ce problème. Un k-arbitre généralise la notion de coterie habituellement utilisée pour résoudre l'exclusion mutuelle simple. Pour accéder à une resource dans le cas de k parmi h, un processus doit demander la permission d'accès à un quorum du k-arbitre.

Protocole de réplication de données : Dans un système réparti, plusieurs copies d'un objet sont maintenues dans des sites différents pour améliorer la tolérance aux pannes du système. Chaque copie possède un numéro de version qui est incrémenté après chaque modification de la copie.

A chaque objet est associé un ensemble de quorums de lecture et un autre d'écriture. Une opération de lecture accède à toutes les copies de l'ensemble des quorums de lecture et retourne la copie ayant le numéro de version le plus grand. Une opération d'écriture met à jour toute les copies de l'ensemble de quorums d'écriture et leur affecte un numéro de version égal au maximum des numéros de version de ces copies +1.

Soit Q^c et Q les ensembles s de quorums de lecture et d'écriture respectivement. Pour assurer l'équivalence d'une copie dans le système, Q^c et Q doivent vérifier les propriétés d'intersection suivantes:

- Lecture-écriture : $\forall H \in Q^c, \forall G \in Q \quad G \cap H \neq \emptyset$.
- Ecriture-écriture : $\forall G,H \in Q \quad G \cap H \neq \emptyset$.

Pour le problème de réplication, **[28]** associe à chaque objet à répliquer x un arbre de réplication Tree[x] qui liste l'ensemble des sites qui possèdent une copie de x. Pour effectuer la lecture de x, un quorum de lecture est construit à partir de Tree[x] où c'est la version de qui a le plus grand numéro qui est lue. De même, lors d'une écriture, un quorum d'écriture est construit et toutes les copie de ce quorum avec leur numéro de version sont mises à jour

VI. CONCLUSION

Il existe plusieurs systèmes de quorums. Pour les applications distribuées, le choix d'un système dépend surtout des performances attendues de ce système. En effet, pour réduire le nombre de messages échangés entre les différents sites, il est plus approprié de choisir un système avec des quorums de petites tailles. De même, pour déduire la charge des sites, il faut choisir des quorums qui impliquent de petites charges.

I. INTRODUCTION

Dans le chapitre précédent, nous avons parcouru les domaines d'utilisation des quorums les plus importants dans les systèmes distribués. Nous nous intéressons dans ce chapitre au cas particulier de l'application des quorums dans le problème de la localisation des objets dans les réseaux mobiles.

Trois solution seront exposées ci-dessous. Les deux premières solutions proposent des algorithmes pour réduire la charge du système alors que le souci de la dernière solution est principalement de réduire le coût de communication.

II. APPLICATION DES COTERIES DANS LA LOCALISATION DES MOBILES

Voici les solutions de localisation des mobiles basées sur les quorums que nous avons recensé :

II.1. Solution proposée par Ravi Prakash et Mukesh Singhal

Le système étudié par **[10]** est un réseau mobile composé de stations fixes reliées entre elles par un réseau filaire et de stations mobiles qui communiquent avec les stations fixes via l'interface de communication sans fil.

[10] propose de déterminer l'ensemble A de serveurs de localisation (*Mobile Sevice Station* MSS) qui possèdent l'information de la position d'un mobile (*Mobile Host* MH) donné en fonction de l'identité du mobile et de la MSS qui veut le localiser dans le cas d'une recherche ou de l'identité de du mobile et de la MSS vers laquelle il s'est déplacé dans le cas d'une mise à jour. Ainsi, A est déterminé de façon dynamique.

Ce choix est justifié par le fait que si A est statique, l'adresse du mobile est toujours sauvegardée dans le même ensemble de MSS. Aussi, utiliser juste l'identité du mobile pour le retrouver n'exploite pas la notion de localité : un mobile a tendance à communiquer avec des mobiles qui sont dans son voisinage et qui forment son ensemble de travail. L'ensemble de travail d'un mobile change donc avec ses déplacements alors que les nœuds qui contiennent son adresse restent inchangés.

Soit la fonction $h : MSS * MH \rightarrow S_{MSS}$. h est utilisée pour définir l'ensemble S_{MSS} des MSS que MSS interroge pour localiser un MH. Elle définit aussi l'ensemble S_{MSS} de MSS qui sauvegarde l'adresse d'un MH lorsqu'il va vers une MSS. h détermine ainsi l'ensemble d'écriture lors de la mise à jour (lorsque le MH se déplace) et l'ensemble de lecture lors de la recherche du MH.

L'ensemble défini par h pour un MH correspond à l'ensemble A. Il change avec les déplacements de ce MH tel que les MHs d'une même cellule n'ont pas les mêmes ensembles de localisation.

Définition

Un MH est *hot* s'il est souvent recherché, il est *cold* sinon. Les MH qui sont *hot* sont recherchés plus souvent que les autres MH.

Les nœuds qui demandent après les MH *hot* peuvent être dispersés à travers tout le réseau. C'est pourquoi un plus grand nombre de serveurs de localisation dispersés dans le réseau doivent maintenir les adresses des mobiles *hot*. Moins de serveurs de localisation sont nécessaires pour garder les adresses des mobiles *cold*. Dans ce but, des alias appelés *identités virtuelles* sont affectées aux MH. Un MH *hot* possède plusieurs identités virtuelles alors qu'un MH *cold* n'en possède qu'une seule ; un MH *hot* est équivalent à plusieurs MH *cold*.

Déterminer si un mobile x est *hot* ou *cold* se fait par la MSS y de la cellule dans laquelle il se trouve. Si au temps t, le nombre de recherches par unité de temps reçues par y destinées à x qui ne proviennent pas de y (sont en dehors de y) durant une période [t-T,t] dépasse un seuil prédéfini, x est *hot.* Il est *cold* sinon.

Soit MSS_id l'adresse d'un mobile et VMH_id son identité virtuelle. En utilisant le principe de la fonction de hachage à adressage ouvert, et plus précisément le double hachage, h est définie comme suit :

$h(MSS\_id, VMH\_id) = (h_1(MSS\_id) + VMH\_id * h_2(MSS\_id)) \bmod m$ avec m : le nombre de MSS dans le réseau et h_1 et h_2 des fonctions de hachage auxiliaires. h_1 et h_2 sont uniformément distribuées sur [0,*m*-1]. Soit le couple (MSS_id,VMH_id), h(MSS_id,VMH_id) sera uniformément dispersée sur [0,*m*-1].

Recherche d'un mobile

Lorsqu'une MSS veut localiser un mobile MH, elle cherche dans son cache où sont sauvegardées les adresses des derniers MH recherchés dans cette MSS. S'il y a une entrée pour MH dans le cache, la MSS essaye d'abord de le contacter à cette adresse. Si cette recherche échoue ou que MH ne possède pas d'entrée dans le cache, la MSS détermine toutes les identités virtuelles du MH dont l'identité est MH_id. Elle définit ensuite l'ensemble de recherche :

- Q*uery_Set* $\leftarrow \varnothing$.
- old_MSS sélectionne une VMH_id pour le MH et calcule $j = h(MSS\_id, VMH\_id)$
- Q*uery_Set*$\leftarrow$ Sj.

Elle envoie (diffuse) un message de recherche à toutes les $MSS_i \in$ Q*uery_Set* pour localiser MH. Q*uery_Set* représente l'ensemble de lecture du couple (MSS_id,VMH_id). Si une MSS_i interrogée possède l'information de l'adresse du MH, elle envoie cette information en réponse.

Mise à jour de l'adresse d'un mobile

Lorsqu'un MH se déplace d'une MSS dont l'identité est old_MSS vers une nouvelle MSS dont l'identité est new_MSS. old_MSS détermine toutes les identités virtuelles de MH et calcule $j = h(old\_MSS, VMH\_id)$ pour chaque identité virtuelle. Elle diffuse ensuite un message de suppression aux MSS définies par h pour leur demander de supprimer l'entrée de MH dans leurs bases de données. De même, new _MSS détermine toutes les identités virtuelles de MH et calcule $i = h(new\_MSS, VMH\_id)$ pour chaque identité virtuelle. Elle diffuse ensuite un message d'ajout aux MSS définies par h pour leur demander d'ajouter une entrée de MH dans leurs bases de données qui contient son adresse courante (i.e. new_MSS).

Affectation des identités virtuelles et transition d'état

Soit VMH_id l'identité virtuelle de MH dont l'identité est MH_id.

Soit une constante entière x qui est un paramètre de l'algorithme et qui est déterminée à priori.

Si MH est *cold,* son $VMH\_id = MH\_id + x$. S'il est *hot,* il lui sera affecté plusieurs identités virtuelles. Supposons maintenant qu'un MH *hot* ne possède pas plus de deux identités virtuelles (pour simplifier la solution proposée) $VMH\_id_1 = MH\_id + x$ et $VMH\_id_2 =$ un entier entre 0 et $x-1$ (inclusif) qui n'a pas encore été affecté comme identité virtuelle.

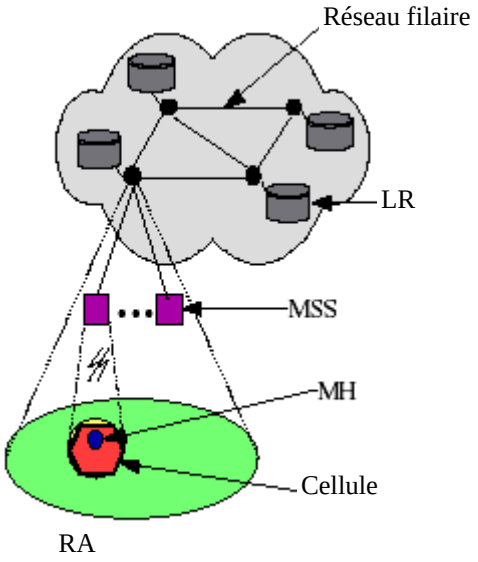
Lorsqu'un MH devient *cold* dans MSS, MSS définie les identités virtuelle $i \in VMH\_id(MH\_id)$ telles que $i \neq MH\_id + x$. Pour chaque identité virtuelle ainsi définie, MSS calcule $j = h(MSS, i)$ et leur envoie un message de suppression. Ces identité virtuelles supprimées sont ajoutées à la liste des identités virtuelles disponibles (dans ce cas une seule identité virtuelle est supprimée). Celles-ci pourront ainsi être affectées à d'autres

MH *hot* par la suite. Dans le cas inverse, lorsqu'un mobile passe de l'état *cold* à l'état *hot* dans MSS, celle ci affecte des identités virtuelles $i \neq MH\_id + x$ à ce mobile en plus de l'identité virtuelle qu'il possédait déjà ($MH\_id + x$). MSS calcule pour toutes ces identités virtuelles i $j = h(MSS, i)$ et leur envoie un message d'ajout pour qu'ils ajoutent une entrée pour MH dans leurs bases de données qui contient l'adresse MSS.
S'il existe plus de x mobiles *hot*, le surplus de MH *hot* se verra attribuer la première identité virtuelle seulement et sera traité comme un MH *cold.*

Construction des ensembles de lecture et d'écriture
Pour minimiser le surcoût en communication induit par la recherche et la mise à jour, il est nécessaire que la taille de chaque quorum soit la plus petite possible. De plus, pour distribuer la responsabilité de sauvegarde de manière équitable, les quorums doivent être symétriques. Pour ce faire, le schéma basé sur la grille semble être indiqué pour construire les quorums.

Pour améliorer sa solution, **[11]** propose de regrouper les MSS en zones d'enregistrement pour réduire le coût de mise à jour. Une zone d'enregistrement (*Registration Area RA*) est le regroupement de plusieurs cellules. Une RA peut être composée d'une ou plusieurs cellules voisines (voir Figure 4.1). Chaque RA possède un registre de localisation (*Location Register LR*) qui contient les informations de tous les mobiles des MSS qui composent cette RA.
L'adresse d'un MH est mise à jour non à chaque fois qu'il change de cellule mais à chaque fois qu'il change de RA. La requête de recherche sur un MH donne comme résultat la RA nommée RA_id dans laquelle il se trouve. Lorsque RA_id est déterminée, une recherche parallèle (diffusion) est lancée dans les cellules de la RA pour trouver l'adresse du MH recherché.
De plus, la fonction de hachage utilisée précédemment assure un équilibre de charge à un instant donné mais ne peut pas le faire à d'autres moments. Par exemple, quelques MH peuvent être *hot* et par conséquent, même si h affecte un nombre égal de couples (MSS,MH) à chaque quorum, quelques quorums reçoivent plus de requêtes de recherche et de mise à jour par rapport à d'autres.
Il y a aussi une possibilité qu'un groupe de nouveaux MH rejoigne le réseau. Suivant la distribution géographique de ces MH et des nœuds qui risquent de les appeler, h peut affecter la responsabilité de leur localisation plus souvent à certains quorums plus qu'à d'autres. Ce qui peut surcharger ces quorums. Une solution à ce problème est d'utiliser le hachage

dynamique. h(MSS,MH) sera une fonction de hachage dont le rang de valeurs peut augmenter ou diminuer selon la charge dans le système.



- Figure 4.1 -

Recherche d'un mobile

Lorsqu'un MH est recherché, si la recherche dans le cache échoue, les opérations suivantes sont effectuées :

- Déterminer toutes les identités virtuelles de MH.
- *Query_Set*←∅.
- Choisir une identité virtuelle et calculer $j = h(MSS\_id, VMH\_id)$: Q*uery_Set*← Sj.
- Diffuser la requête de recherche à Q*uery_Set*

Mise à jour de l'adresse d'un mobile :

Soit old_MSS la cellule de l'ancienne RA où se trouve un MH x lorsqu'il lance la procédure de mise à jour et new_MSS la cellule de la nouvelle RA vers laquelle il s'est déplacé.

Les opérations suivantes sont effectuées pour mettre à jour l'adresse de MH.

- Déterminer toutes les identités virtuelles VMH_id pour MH.
- *purge_set*←∅, i*nform*_set←∅.
- Pour toutes les VMH_id, calculer :

$$\left\{ j = h(old\_MSS, VMH\_id),\ purge\_set \leftarrow purge\_set \cup S_j \right\}$$

- Pour toutes les VMH_id, calculer :

$$\left\{ j = h(new\_MSS, VMH\_id),\ \text{inf}\ orm\_set \leftarrow \text{inf}\ orm\_set \cup S_j \right\}$$

- Diffuser un message de suppression à $purge\_set - \inf orm\_set$
- Diffuser un message d'ajout à $\inf orm\_set - purge\_set$
- Diffuser un message de mise à jour à $purge\_set \cap \inf orm\_set$.

Pour réduire encore plus le coût de recherche, **[12]** rassemble les RA en *zones*. Une zone est constituée d'une ou plusieurs RA physiquement (ou logiquement) regroupées. Elle possède un serveur de localisation qui contient principalement la liste des mobiles qui se trouvent dans les LR qu'elle couvre.

Lorsqu'un nœud x veut localiser un mobile, si celui-ci n'est pas dans la zone où se trouve x alors un quorum de lecture est construit comme expliqué précédemment pour trouver l'adresse du mobile.

Lorsqu'un mobile va d'une RA à une autre, son adresse est mise à jour comme expliqué précédemment. Si de plus, lors de déplacement il change de zone, alors son adresse est supprimée du serveur de localisation de l'ancienne zone et est ajoutée à celui de la nouvelle zone.

[12] propose aussi d'augmenter cette solution en utilisant le cache, les FP ou les stratégies de mises à jour et de remplacer le schéma basé sur la grille qui est symétrique (qui permet l'équilibre de charge) mais qui donne des quorums dont la taille est de $O(\sqrt{m})$ par le schéma basé sur le Crumbling Walls logarithmique qui compromet la charge du système pour réduire la taille des quorums qui varie dans ce modèle entre $Log_2 m - Log_2 Log_2 m$ et $\frac{m}{Log_2 m}$.

Evaluation

Equilibre de charge par les identités virtuelles : Un mobile *hot* a au moins un et souvent deux ensembles d'écriture. Donc, la distance moyenne entre un nœud qui cherche ce mobile et sa position courante est plus petite que dans le cas où l'adresse du mobile se trouve dans un seul ensemble.

De même, les MSS interrogées lors d'une recherche appartiennent à l'un des deux ensembles de lecture ou à un seul :

- Soit un MSS qui appartient aux deux ensembles de lecture : Dans ce cas, le MH *hot* est pour le MSS, équivalent à deux MH *cold* (il a deux identités virtuelles). En raison de la propriété de symétrie de la construction des ensembles de lecture et d'écriture et de

l'uniformité de la distribution de la fonction de hachage, ce MSS sauvegarde moins de MH que les autres qui ne sauvegardent que des MH *cold*.

- Soit un MSS qui n'appartient qu'à un seul ensemble de lecture. Comme les requêtes ne sont envoyées qu'à un seul ensemble de lecture, ce MSS ne recevra en moyenne que la moitié des requêtes de recherche pour la localisation du MH.

Donc, une MSS qui sauvegarde l'adresse de mobiles *hot* sauvegarde les adresses de moins de mobiles et reçoit moins de requêtes de localisation qu'une MSS qui ne sauvegarde que les adresses de mobiles *cold*.

Equilibre de charge avec le hachage dynamique : Lorsque la charge augmente dans les serveurs de localisation, de nouveaux serveurs de localisation sont ajoutés, ce qui crée de nouveaux quorums. Dans le cas inverse, lorsque la fréquence des recherches et mises à jours diminue, quelques serveurs peuvent devenir dormants. Ce qui réduit le nombre de quorums. Lorsque un quorum X est ajouté au système pour réduire la charge d'un autre quorum Y, les responsabilités de localisation assumées par Y seront partagées entre X et Y. De même lorsque la charge diminue, les responsabilités réduites de deux quorums peuvent être assignées à un seul d'entre eux. L'autre sera libéré et deviendra dormant. Dans les deux cas, une reconfiguration des informations de localisation est nécessaire.

L'ajout d'un quorum s'effectue de la manière suivante :

- Ayant un ensemble de MSS, construire un système de quorums à l'avance contenant 2^{limit} quorums distincts.
- Initialement, utiliser un sous ensemble de ces quorums pour la localisation.
- Lorsque la charge augmente (soit à cause de l'ajout de nouveaux MH dans le réseau soit à cause de l'augmentation du nombre de recherches et mises à jours des MH existants), ajouter des quorums au système à partir des quorums en réserve.

Un quorum peut être actif ou en réserve selon la charge des quorums actifs. Aussi, afin de déterminer la charge des quorums, trois actions sont effectuées :

- Chaque fois qu'une mise à jour ou une recherche est envoyée à un serveur de localisation, l'identité du quorum auquel correspond la requête est ajoutée dans le message.
- Un serveur de localisation x peut appartenir à plusieurs quorums actifs, pour chaque quorum actif auquel il appartient, x maintient un compteur initialisé à zéro. Lorsque x reçoit un message de recherche ou de mise à jour, il incrémente le compteur correspondant à l'identité du quorum lue dans le message reçu.

- Périodiquement, les serveurs de localisation sont interrogés. La valeur de leurs compteurs est mesurée pour déterminer le nombre de mises à jour et de recherches effectuées entre l'ancienne et la dernière interrogation, puis les compteurs sont remis à zéro. Si l'interrogation indique que le nombre de requêtes associées à un quorum dépasse un seuil prédéfini, le quorum est déclaré surchargé.

Il est important de choisir un seuil et un intervalle de temps appropriés pour maximiser les bénéfices du hachage dynamique.

On peut dire que le hachage dynamique est employé pour assurer une charge équilibrée gros grains et que les identités virtuelles sont utilisées pour une charge équilibrée grains fins.

Communication

L'algorithme de recherche impose de petits surcoûts de calcul et de communication.

Sachant qu'il y a $O(\sqrt{m})$ éléments dans chaque quorum si la stratégie de construction de quorums utilisée est le schéma basé sur la grille. Dans les procédures de transition d'état, les messages d'ajout ou suppression sont envoyés au plus à un quorum de MSS. Dans la procédure de mise à jour, les messages d'ajout et de suppression sont envoyés au plus à deux quorums, chacun. Lors de la recherche d'un mobile, les messages de recherche sont envoyés à un quorum seulement. D'où, la complexité de message de chaque opération est $O(\sqrt{m})$ si la stratégie de construction de quorums utilisée est le schéma basé sur la grille.

Chaque message a une petite taille, il consiste en au plus de deux identités de MSS, une identité d'un MH et un flag qui indique la nature du message. De plus, ces messages sont transmis dans le réseau statique qui a une plus grande bande passante que les liaisons MH-MSS sans fil.

VI.2. Solution proposée par Akhil Kumar et Tracy Camp

[16] s'est inspiré de la solution de Prakash pour résoudre le problème de la localisation. Lors d'une recherche (resp. une mise à jour), le nœud qui effectue cette opération choisi un quorum de lecture (resp. écriture) pour le faire. Ce quorum est choisi dans l'ensemble de quorums disponibles (tous les nœuds de ces quorums sont non défaillants) tel que la charge du système reste équilibrée. **[16]** propose par contre une nouvelle construction pour les quorums de lecture et écriture et introduit les FP pour améliorer les performances du système.

Mise à jour de l'adresse d'un mobile

Un compteur de déplacement est associé à chaque mobile. Lorsque celui-ci se déplace, son compteur de déplacement est incrémenté et un FP est créé de son ancienne position vers la nouvelle. Lorsque ce compteur atteint le nombre maximum de déplacements sans mise à jour (et qui définit la longueur maximum de la chaîne de FP), un quorum d'écriture est construit. La nouvelle adresse du mobile est enregistrée dans les nœuds de ce quorum d'écriture, le compteur de déplacement associé au mobile est remis à zéro et la chaîne de FP est remise à Nil.

Recherche d'un mobile

Lorsqu'un nœud cherche à localiser un mobile, il construit un quorum de lecture et interroge les nœuds de ce quorum. Ici, deux cas sont possibles : soit le mobile se trouve dans le nœuds indiqué par la recherche, soit il s'est déplacé et dans ce cas, il faut suivre les FP pour trouver son adresse.

Généralisation de la solution

[16] généralise l'architecture proposée en une hiérarchie de grilles car les modèles hiérarchiques tendent à être extensibles tels que le coût de recherche et de mise à jour est proportionnel à la distance entre l'appelé et l'appelant.

Dans une organisation en grille multi-niveaux. Un seul nœud est au niveau 0. Un groupe de n nœuds sont organisés en grille $m * m$. Au niveau 1, un de ces n nœuds sert de représentant pour la grille de sa région : ce représentant participe dans la construction de quorums du niveau supérieur suivant. De même, les régions du niveau 1 sont regroupées en une grille $m * m$ pour former le niveau 2. Là aussi, un nœud de cette grille $m * m$ est pris comme représentant du niveau 2.

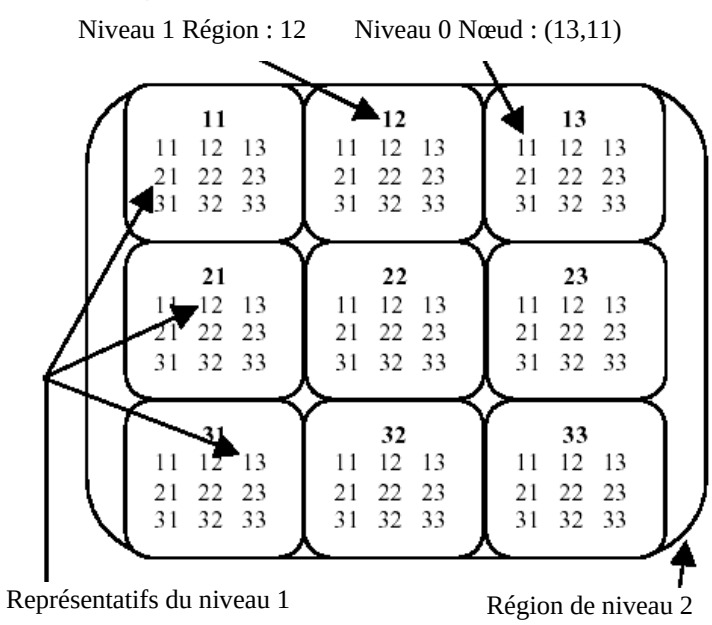
Exemple

Figure 4.2 représente une grille 3*3 au niveau 2 dans laquelle chaque élément est un une grille 3*3 de niveau 1. Au niveau 1, chaque élément dans la grille 3*3 est un nœud du niveau 0.

Le nœud en haut et à gauche peut être décrit par le couple (11,11) où la première coordonnée représente la position de ce nœud dans le niveau 2 et la seconde coordonnée représente sa position dans la grille de niveau 1. Aussi, pour identifier un nœud dans une hiérarchie de k niveaux, il faut utiliser un k-uplet qui détermine sa position à chaque niveau.

Soit le nœud (21,32) qui cherche un mobile qui se trouve dans le nœud (21,23) tel que l'adresse de ce mobile a été sauvegardée dans le quorum {(21,21),(21,22),(21,23)}lorsque ce mobile s'est déplacé vers (21,23).

Pour déterminer l'adresse de ce mobile, (21,32) construit un quorum de lecture. Soit {(21,11),(21,12),(21,22),(21,32)} ce quorum. Le nœud (21,32) connaît alors l'adresse du mobile. En d'autre terme, si l'appelé et l'appelant se trouvent dans la même région du niveau 1, la recherche est effectuée comme expliqué précédemment.



- Figure 4.2 -

Lorsque un nœud x recherche un mobile y qui se trouve dans une autre région, le représentant de la région de x est contacté. Si (21,32) recherche un mobile qui est dans (13,33), (21,32) forme d'abord un quorum de lecture dans sa région 21 ; soit {(21,21),(21,31)}et ne trouve pas l'adresse du mobile. La recherche de ce mobile est étendue au niveau suivant. Dans ce cas, (21,32) envoie une requête de recherche à son représentant (21,12) et le représentant forme un quorum de lecture dans la grille 3*3 du niveau supérieur comme {(21,12),(31,13)}par exemple.

Lorsque le mobile recherché par (21,32) et qui est à (13,33) s'est déplacé vers la région 13 dans le réseau de grille, son adresse à été sauvegardée à plusieurs niveaux de la hiérarchie. Soit {(13,31),(13,32),(13,33),(13,23)} le quorum d'écriture qui a sauvegardé l'adresse de ce mobile au niveau 1 et {(31,13),(32,22),(22,12),(12,32)} le quorum d'écriture qui a sauvegardé

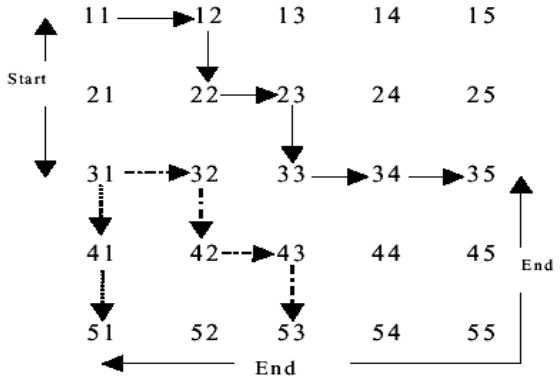
l'adresse de ce mobile au niveau 2, le nœud (21,32) connaît alors l'adresse du mobile à partir de (31,13).

Construction des quorums

[16] propose une solution basée sur la grille. Soit un réseau de n nœuds organisés en une grille de $m * m$ nœuds. La notion de quorums de lecture et d'écriture utilise l'idée de *nœuds adjacents* et de la variable *mid*. Deux nœuds sont adjacents si leurs cordonnées diffèrent dans une seule position. *mid* représente la ligne du milieu de la grille. Si m est impair alors $mid = \frac{m+1}{2}$ sinon $mid = \frac{m}{2} + 1$.

Les quatre bords de la grille sont appelés les bords haut, bas, droit et gauche. Les nœuds entre la ligne 1 et la ligne *mid* (inclusives) le long du bord droit (resp. gauche) sont appelés nœuds du haut du bord droit (resp. gauche). Les nœuds entre la ligne *mid* et la ligne m (inclusives) le long du bord droit (resp. gauche) sont appelés nœuds du bas du bord droit (resp. gauche).

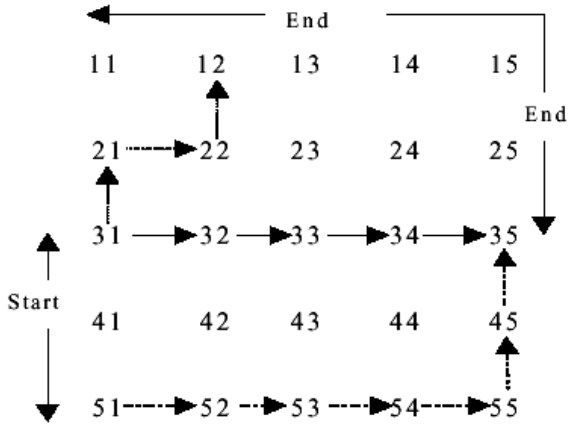
Un quorum de lecture est formé en trouvant un chemin ordonné et minimal de nœuds adjacents qui part d'un nœud entre le nœud (1,1) et le nœud (*mid*,1) le long du haut du bord gauche et qui se termine soit à un nœud entre (m,1) et (m,m) le long du bord bas, soit à un nœud entre (*mid*,m) et (m,m) le long du bas du bord droit. Figure 4.3 montre trois exemples de quorums de lecture possibles dans une grille 5*5.



- Figure 4.3 -

Un quorum d'écriture est formé en trouvant un chemin ordonné et minimal de nœuds adjacents qui part d'un nœud entre le nœud (*mid*,1) et le nœud (m,1) le long du bas du bord gauche et qui se termine soit à un nœud entre (1,1) et (1,m) le long du bord haut, soit à un nœud entre (1,m) et (*mid*,m) le long du haut du bord droit. Figure 4.4 montre trois exemples de quorums d'écriture possibles dans une grille 5*5.

Les quorums de lecture et écriture ainsi construits vérifient la propriété d'intersection des quorums.



- Figure 4.4 -

Evaluation

[16] compare sa solution à celle de **[44]** basée sur la grille CAA et à celle de **[15]** basée sur les Crumbling Walls Logarithmiques en terme de disponibilité, charge et taille da quorums.

La construction basée sur la grille CAA:

Dans cette construction, les nœuds sont disposés en une grille $m * m$. Le quorum de lecture est formé en prenant un nœud de chaque colonne de la grille alors que le quorum d'écriture est formé en prenant une colonne complète.

Exemple

11	12	13	14	15
21	22	23	24	25
31	32	33	34	35
41	42	43	44	45
51	52	53	54	55

- Figure 4.5 -

Dans la Figure 4.5, {21,32,33,54,15} peut être un quorum de lecture et {14,24,34,44,54} un quorum d'écriture.

Le Tableau 4.1 représente la disponibilité de chaque protocole en supposant que chaque nœud du réseau est indépendamment fiable avec une probabilité P=0,9. Avg avail. représente la

disponibilité moyenne entre le quorum de lecture et le quorum d'écriture. La solution basée sur Crumbling Walls Logarithmiques est la meilleure solution, vient ensuite la solution de **[16]** puis enfin la solution basée sur la grille CAA.

Le Tableau 4.2 montre les performances en taille de quorum de chaque protocole pour m entre 3 et 9. Ici par contre, les solutions sont classées dans l'ordre grille CAA, solution de **[16]** puis Crumbling Walls Logarithmiques.

Le Tableau 4.3 compare les trois protocoles en terme de charge. soit f la fraction de lectures, la fraction des écriture est $1-f$. La charge maximum se présente lorsque $f=0$ ou $f=1$. Lorsque $f=0.5$ la charge du système **[16]** est équivalente à celle de la grille CAA. Dans la majorité des cas, CW-Log réalise la plus grande charge.

	New Grid			CAA Grid			CWLog
m	Read Avail	Write Avail	Avg Avail.	Read Avail	Write Avail	Avg Avail.	Avg Avail.
3	0.975	0.975	0.975	0.997	0.980	0.989	0.995
4	0.994	0.975	0.985	1.000	0.986	0.993	0.999
5	0.994	0.994	0.994	1.000	0.988	0.994	1.0
6	0.998	0.994	0.996	1.000	0.989	0.995	1.0
7	0.998	0.998	0.998	1.000	0.989	0.995	1.0
8	0.998	0.999	0.999	1.000	0.989	0.995	1.0
9	0.999	0.999	0.999	1.000	0.988	0.994	1.0

- Tableau 4.1 -

m	New Grid		CAA Grid	CWLog
	Read	Write		
3	3.33	3.33	3	4.00
4	3.75	5.05	4	6.31
5	6.77	6.77	5	7.67
6	8.44	8.64	6	9.58
7	10.51	10.51	7	11.41
8	12.32	12.44	8	13.17
9	14.37	14.37	9	14.89

- Tableau 4.2 -

m	New Grid			CAA Grid	CW-Log
	F =1	F=0.5	F= 0		
3	0.50	0.33	0.50	0.33	0.50
4	0.33	0.25	0.50	0.25	0.43
5	0.33	0.20	0.33	0.20	0.33
6	0.25	0.17	0.33	0.17	0.31
7	0.25	0.14	0.25	0.14	0.30
8	0.20	0.13	0.25	0.13	0.25
9	0.20	0.11	0.20	0.11	0.24

- Tableau 4.3 -

En résumé, la solution de **[16]** est une solution entre la CAA Grid et la CW-Log. En effet, elle est entre la CW-Log et la CAA Grid en terme de disponibilité et l'inverse en terme de taille de quorum et de charge. Elle représente dont une moyenne entre les deux solutions et permet un meilleur compromis entre les trois critères de qualité étudiés.

VI.3. Solution proposée par Ihn-Han-Bae

Dans le système étudié par **[2]**, les MSS sont géographiquement ou logiquement regroupées en RA tel que à chaque RA est associé un LR.

Si les solutions précédentes ont pris l'équilibre de charge et la disponibilité comme critères de qualité, cette solution choisit le coût de communication comme critère de qualité.

[2] reprend la notion de mobile *hot* et *cold* utilisée par Prakash. De plus, il associe à chaque nœud (c'est à dire LR) dans le réseau un quorum ligne et un quorum colonne pour la recherche et la mise à jour. **[2]** a introduit le cache pour réduire le coût de communication.

Recherche d'un mobile

Lorsqu'un nœud A veut localiser un mobile dont l'identité est MH_id, A cherche l'adresse de MH_id dans son cache. S'il ne la trouve pas où celle qu'il trouve est obsolète, A envoie un message recherche à tous les nœuds de son quorum ligne ou de son quorum colonne pour leur demander l'adresse de ce mobile.

Mise à jour de l'adresse d'un mobile

Un mobile met à jour son adresse lorsqu'il va d'un LR à un autre. La procédure de mise à jour dépend de l'état du mobile, à savoir s'il est '*hot*' ou '*cold*'.

Lorsque le mobile est *hot*, deux cas sont possibles :

- Si la nouvelle et l'ancienne MSS sont dans le même LR (i.e. le mobile n'a pas changé de LR), un message de mise à jour est envoyé aux quorums ligne et colonne du LR pour qu'ils mettent à jour l'adresse de ce mobile.
- Si le mobile s'est déplacé d'un LR vers un autre, le nouvel LR envoie un message d'insertion à ses quorums ligne et colonne qui ajouteront une entrée dans leurs bases de données pour ce mobile et qui contiendra l'adresse de la MSS dans laquelle il se trouve. L'ancien LR envoie un message de suppression à ses quorums ligne et colonne pour qu'ils suppriment l'entrée que possède ce mobile dans leurs bases de données.

Lorsque le mobile est *cold* :

- Si la nouvelle et l'ancienne MSS sont dans le même LR, un message de mise à jour est envoyé à tous les nœuds de son quorum ligne.
- Si le mobile s'est déplacé d'un LR vers un autre, le nouvel LR envoie un message d'insertion à son quorum ligne et l'ancien LR envoie un message de suppression à son quorum ligne.

Etat de transition :

Lorsque un mobile passe de l'état '*hot*' à l'état '*cold*' dans un LR, ce LR envoie un message de suppression à son quorum colonne. Lorsque c'est l'inverse qui se produit, le LR lui envoie un message d'insertion.

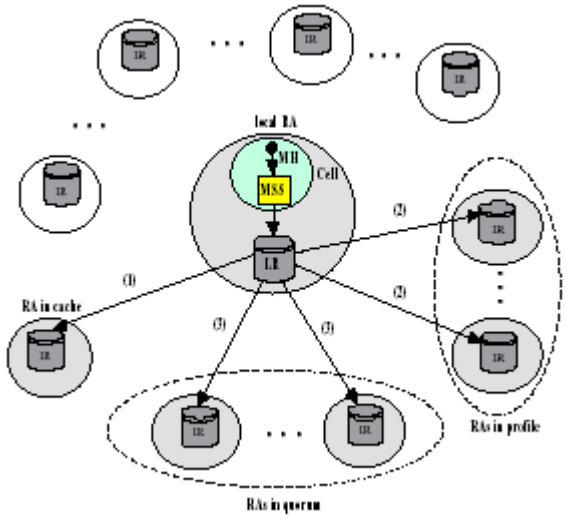
Ihn-Han Bae a amélioré sa solution en y introduisant la notion de profil **[3]**. Chaque LR possède une structure appelée profil qui reproduit le modèle (comportement) de mobilité usuel des mobiles *hot* qui sont plus recherchés que les autres. Il est défini sur la base de l'historique. Chaque entré MH_id dans la structure profil possède trois LR dans l'ordre de leur popularité (la probabilité que MH soit dans la LR à chaque période de temps).

Recherche d'un mobile

Lorsqu'un nœud A veut localiser un mobile dont l'identité est MH_id, A cherche l'adresse de MH_id dans son cache. S'il la trouve, il envoie une requête de recherche à l'adresse trouvée. Si l'adresse trouvée n'est pas obsolète alors le mobile est localisé sinon A supprime l'entrée du mobile dans son cache.

Dans le cas où le mobile n'est pas encore localisé, A envoie un message de recherche aux LR du profil de MH_id dans A. S'il est localisé alors la procédure de recherche est terminée. Dans le cas contraire, si le nombre de A est impair, A envoie un message de recherche à tous les nœuds du quorum colonne pour leur demander l'adresse de ce mobile sinon il l'envoie à son quorum ligne.

La recherche dans le pire cas s'effectue de la manière suivante (voir Figure 4.6). Un LR reçoit une requête de recherche sur MH_id. Il envoie un message au LR qui se trouve dans son cache à l'entrée MH_id qui échoue. Il envoie ensuite des messages de recherches aux LR qui se trouvent dans son profil à l'entrée MH_id et qui échouent tous. Enfin, il diffuse la requête aux LR de son quorum ligne ou colonne avant de trouver l'adresse du MH_id.



- Figure 4.6 -

Mise à jour de l'adresse d'un mobile

Un mobile met à jour son adresse lorsqu'il va d'un LR à un autre. La procédure de mise à jour dépend de l'état du mobile, à savoir s'il est '*hot*' ou '*cold*'.

Lorsque le mobile est *hot*, la procédure de mise à jour ne change pas.

Lorsque le mobile est *cold* :

- Si la nouvelle et l'ancienne MSS sont dans le même LR : Si le nombre de A et impair, A envoie un message de mise à jour à tous les nœuds de son quorum colonne sinon il l'envoie à son quorum ligne.
- Si le mobile s'est déplacé d'un LR vers un autre,
  - Si le nombre du nouvel LR est impair, celui-ci envoie un message d'insertion à son quorum colonne sinon il l'envoie à son quorum ligne.

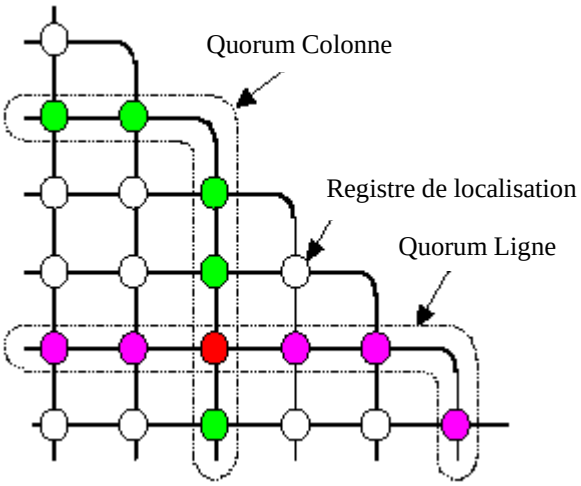
- Si le nombre de l'ancien LR est impair, celui-ci envoie un message de suppression à son quorum colonne sinon il l'envoie à son quorum ligne.

Etat de transition :

Lorsque un mobile passe de l'état '*hot*' à l'état '*cold*' dans un LR, si le nombre de ce LR est impair, il envoie un message de suppression à son quorum ligne sinon il l'envoie à son quorum colonne. Lorsque c'est l'inverse qui se produit, si le nombre de ce LR est impair, il envoie un message de d'ajout à son quorum ligne sinon il l'envoie à son quorum colonne.

Construction des quorums

Dans cette solution, les RA sont organisées en un triangle logique. Les quorums sont définis par la construction *Ligne et Colonne* (Figure 4.7). Chaque deux lignes ont exactement un nœud en intersection. De plus, tous les quorums ont la même taille qui est approximativement $\sqrt{2}\sqrt{n}$ où n est le nombre de RA dans le réseau.



- Figure 4.7 -

Evaluation

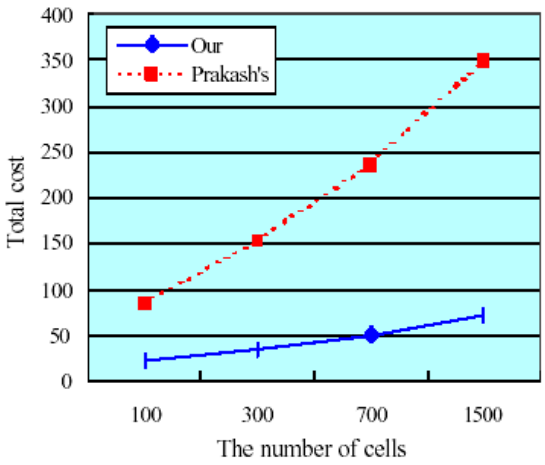
Dans ce modèle, les MH sont supposées uniformément distribuées dans les LR avec la même densité.

Comme les LR pairs et impairs utilisent des quorums différents, l'équilibre de charge dans les processus de recherche et de mise à jour est accompli.

Pour évaluer son travail, **[2,3]** compare sa solution à celle de Prakash (où la construction des quorums est basée sur la grille) en terme de coût de communication total C_{total} (voir Figure 4.8). Le coût total de la communication correspond à la somme du coût de recherche et de

mise à jour lorsqu'un mobile est *hot* et le coût de recherche et de mise à jour lorsqu'un mobile est *cold*.

$C_{total} = P_{hot} * (S_{hot} + U_{hot}) + (1 - P_{hot}) * (S_{cold} + U_{cold})$ avec :



- Figure 4.8 -

Figure 4.8 montre qu'en terme de coût de communication, Ihn-Han Bae a réussi à réduire le coût de communication de façon considérable.

III. CONCLUSION

Dans ce chapitre, nous avons présenté trois solutions pour gérer la localisation des mobiles. Il est clair que le choix du système de quorum influe sur les performances de la solution proposée. Ceci dit, le schéma lui aussi favorise un critère de qualité plutôt qu'un autre. Des techniques peuvent être introduites dans le schéma de localisation et les algorithmes de recherche et mise à jour pour réduire le coût de communication ou la charge du système…

I. INTRODUCTION

Nous nous intéressons dans ce chapitre à notre contribution dans le problème de la localisation des mobiles. Nous proposons un schéma basé sur les quorums. Nous définissons par la suite les algorithmes de recherche et mise à jour. Nous évaluons les performances de cette solution en terme de complexité de messages et de coût de communication total.

II. MODELE DU SYSTEME

Le réseau mobile est constitué de sites fixes qui sont les stations de base et qui sont reliées entre elles par un réseau filaire (statique) classique et de sites mobiles qui communiquent avec ces stations de base via des connections sans fil. Chaque station de base est munie d'une interface de communication sans fil et couvre une aire géographique appelée *cellule.* Un site mobile ne peut communiquer qu'avec la station de base qui couvre la cellule dans laquelle il se trouve. Cette station de base représente alors son adresse.
Les cellules du réseau sont regroupées en zones d'enregistrement (*Registration Area* RA). Chaque RA a un registre de localisation (*Location Register* LR) qui maintient la position de tous les sites mobiles qui sont dans cette RA.

III. PRINCIPE DE LA SOLUTION

Notre solution propose une architecture de la base de données de localisation des mobiles qui est basée sur les quorums. Elle est caractérisée par :

- La diffusion des messages de mise à jour de l'adresse d'un mobile à un ensemble de LR;
- La diffusion des messages de recherche d'un mobile à un ensemble de LR;
- Les LR sont regroupés en ensembles tels que l'intersection entre ces ensembles deux à deux est non nulle.

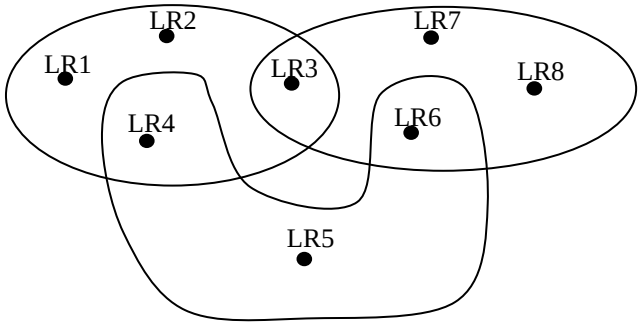
Les LR du réseau sont regroupés en quorums. Aussi, en plus des informations concernant les mobiles qui se trouvent dans les stations de base qu'il contient, un LR possède aussi l'information de tous les mobiles qui se trouvent dans les autres LR appartenant à son quorum.

Remarques :

Soit LR_A le registre de localisation de la zone d'enregistrement RA_A qui se trouve dans le quorum Q_A.

- LR_A possède l'information de tous les mobiles de RA_A en terme de stations de base : Soit x un mobile qui se trouve dans RA_A, l'information sauvegardée au niveau de LR_A concernant l'adresse de x est le numéro de la station de base dans laquelle il se trouve.
- LR_A possède l'information des mobiles des autres LR qui sont dans le même quorum que RA_A en terme de LR : Soit y un mobile qui se trouve dans une zone d'enregistrement RA autre que RA_A et qui appartient à Q_A, l'information sauvegardée au niveau de LR_A concernant l'adresse de y est le numéro de LR.

Un LR qui appartient à l'intersection de plusieurs quorums possède l'information de tous les mobiles des LR qui appartiennent à ces quorums. La taille de la base de données d'une RA A peut être très grande. Cette base de données doit être structurée en sous bases de données où chacune de ces sous bases de données sauvegarde les informations concernant les mobiles de l'un des LR des quorums auxquels appartient A.



- Figure 5.1 -

III.1. Mise à jour de l'adresse d'un mobile

La mise à jour de l'adresse d'un mobile se fait lorsque celui-ci se déplace d'un quorum à un autre.

Lorsque un mobile x se déplace d'une station de base A à une autre station de base B, trois cas sont possibles :

- A et B sont dans le même LR y : Dans ce cas, l'adresse de x est mise à jour au niveau de y
- A et B ne sont pas dans le même LR, mais restent dans le même quorum : Si x se déplace de la RA de A (qui possède LR_A) à la RA de B (qui possède LR_B), alors LR_B et LR_A

mettent à jour l'adresse de x. A savoir, LR_A supprime l'adresse de x dans sa base de donnée associée à LR_A et l'ajoute à celle associée à LR_B. L'inverse se fait au niveau de LR_B.

LR_B diffuse ensuite un message à tous les LR qui sont dans le même quorum que lui (à part LR_A) pour mettre à jour l'adresse de x. Chacun de ces LR va supprimer l'adresse de x dans la sous base de données associée à LR_A et l'ajoute à la sous base de données associée à LR_B.

- A et B sont dans des quorums différents : Si RA_A est dans le quorum Q_A et RA_B est dans le quorum Q_B alors :

- Les LR qui appartiennent à Q_A-Q_B suppriment l'adresse de x de leurs bases de données : LR_A supprime l'adresse de x de sa base de données et diffuse un message de suppression à tous les LR de Q_A-Q_B pour qu'ils suppriment l'adresse de x de leurs bases de données.

- Les LR qui appartiennent à Q_B-Q_A ajoutent l'adresse de x dans leurs bases de données : LR_B ajoute l'adresse de x à sa base de données (ici l'adresse de x est B) et diffuse un message d'ajout aux LR de Q_B-Q_A pour qu'ils ajoutent l'adresse de x dans leurs bases de données (ici l'adresse de x est LR_B)

- Les LR qui appartiennent à $Q_A \cap Q_B$ mettent à jour l'adresse de x : LR_B diffuse aux LR de $Q_A \cap Q_B$ un message de mise à jour pour qu'ils suppriment l'adresse de x de la base de données associées à LR_A et l'ajoutent à la base de données associées à LR_B. Ici aussi l'adresse de x est le LR auquel il appartient (LR_B).

Exemple (voir Figure 5.1)

- Si x se déplace dans LR1 d'une station de base à une autre, LR1 met à jour l'adresse de x ;
- Si x se déplace de LR1 à LR2, LR1 et LR2 mettent à jour l'adresse de x et LR2 envoie un message de mise à jour à LR3 et LR4 pour qu'ils fassent de même ;
- Si x se déplace de LR1 à LR8, LR1 supprime l'adresse de x de sa base de données et diffuse un message de suppression à LR2 et LR4 pour qu'ils fassent de même. LR8 ajoute l'adresse de x à sa base de données et diffuse un message d'ajout à LR6 et LR7 pour qu'ils fassent de même. LR8 envoie aussi un message à LR3 pour qu'il mette à jour l'adresse de x.

Remarques :

- Un LR est affecté à un seul quorum au préalable. Si x se déplace de LR1 à LR6, LR6 diffuse l'information d'ajout et de mise à jour soit à (LR4,LR5) soit à (LR3,LR7,LR8) et non aux deux ensembles.
- Chaque LR doit avoir la configuration des LR du réseau en quorums pour qu'il puisse définir les LR qui appartiennent à l'intersection de son quorum avec les autres quorums du réseau.

Procédure de mise à jour

Soit Mh_id l'identité du mobile, old_SB (resp. new_SB) la station de base de départ (resp. d'arrivée) de x, old_LR(resp. new_LR) le LR de départ (resp. d'arrivée) de x et old_quorum (resp. New_quorum) le quorum de départ (resp. d'arrivée) de x.

Si old_LR=New_LR **alors** mettre à jour l'adresse de x dans old_LR

Sinon Si old_quorum =New_quorum

alors

- mettre à jour l'adresse x dans old_LR et New_LR
- diffuser le message de mise à jour dans old_quorum-{old_LR,New_LR}.

Sinon

- supprimer l'adresse de x de old_LR ;
- diffuser un message de suppression à old_quorum-New_quorum-{old_LR}
- ajouter l'adresse de x dans New_LR ;
- diffuser un message d'ajout dans New_quorum-old_quorum-{New_LR}
- diffuser un message de mise à jour dans New_quorum ∩old_quorum.

III.2. Recherche d'un mobile

Lorsqu'un mobile y qui se trouve dans une station de base A veut localiser un mobile x qui se trouve dans la station de base B. Si A (resp. B) appartient au registre de localisation LR_A (resp. LR_B), A demande à LR_A l'adresse de x. Cinq cas sont possibles :

1. Si x et y se trouvent dans LR_A (i.e. LR_A=LR_B), alors LR_A envoie l'adresse de x (ici B) à A
2. Si LR_A et LR_B sont dans le même quorum, alors LR_A possède l'adresse de x en terme de LR : LR_A sait que x se trouve dans LR_B mais ne sait pas dans quelle station de base de LR_B. LR_A envoie un message de recherche à LR_B qui lui renvoie l'adresse de x (la station de base dans laquelle x se trouve : ici c'est B) et LR_A envoie cette adresse à A.
3. Si LR_A et LR_B ne sont pas dans le même quorum,
   3.1. LR_A consulte son cache pour voir s'il contient une adresse de x.

   Dans un LR, le cache est une structure de taille n qui (comme dans la solution de Ihn-Han Bae) sauvegarde l'adresse -en terme de LR- des n derniers mobiles qui ont été recherché par le LR.

   Si x possède une entrée dans le cache de LR_A (soit LR_x cette entée), LR_A envoie un message de recherche à LR_x. Si x est toujours dans LR_x, LR_x envoie à LR_A l'adresse de x qui la renvoie à A sinon LR_x envoie à LR_A un message d'échec.

   3.2. Si LR_A ne trouve pas l'adresse de x dans son cache, LR_A consulte son profil.

   Le profil d'un mobile est une structure de taille m qui sauvegarde, pour ce mobile, les m LR dans lesquels il y est souvent.

   LR_A diffuse la requête de recherche aux LR du profil de x. Lorsqu'un LR reçoit le message de recherche, si x se trouve dans ce LR, ce dernier envoie l'adresse de x à LR_A qui l'envoie à A sinon il lui envoie un message d'échec.

   3.3. Si LR_A et LR_B ne sont pas dans le même quorum et que LR_A ne trouve pas l'adresse de x dans le cache et que tous les LR associés à x dans le profil répondent négativement à son message de recherche, alors LR_A ne possède pas l'information de l'adresse de x. LR_A diffuse un message de recherche à tous les LR qui appartiennent à l'intersection du quorum de LR_A avec les autres quorums du réseau. Lorsque LR_A reçoit l'adresse de x, il l'envoie à A.

Exemple (voir Figure 5.1)

Si y dans la station de base SB1 qui appartient à LR1 recherche x qui se trouve dans la station de base SB2, SB1 envoie un message de recherche à LR1 :

- Si SB2 est dans LR1 alors LR1 envoie l'adresse x à SB1(ici c'est SB2) ;
- Si SB2 est dans LR2 (resp. LR4 ou LR3) alors LR1 sait que x est dans LR2 (resp. LR4 ou LR3). LR1 envoie un message de recherche à LR2 (resp. LR4 ou LR3) pour lui demander l'adresse exacte de x. Une fois la réponse reçue (c'est à dire SB2), LR1 envoie cette réponse à SB1 ;
- Si SB2 est dans LR8 et que LR8 n'est ni dans le cache, ni dans le profil de LR1, LR1 envoie un message de recherche à LR4 et LR3. LR3 sait que x est dans LR8, il envoie un message à LR8 pour lui demander l'adresse exacte de x. LR3 renvoie cette adresse à LR1 qui l'envoie à son tour à SB1.

Procédure de recherche

Soit SB_id1 (resp. SB_id2) la station de base qui recherche (resp. dans laquelle se trouve) un mobile x dont l'identité est MH_id.

Soit LR_id1 (resp. LR_id2) la LR qui contient SB_id1 (resp. SB_id2).

Soit QR_id1 (resp. QR_id2) le quorum auquel appartient LR_id1 (resp. LR_id2).

Soit $LR_{\cap}$ les LR qui appartiennent à l'intersection de QR_id1 avec les autres quorums du réseau.

Si LR_id1=LR_id2 **alors** LR_id1 envoie l'adresse de MH_id (SB_id2) à SB_id1

Sinon Si QR_id1=QR_id2 **alors**

- LR_id1 envoie un message de recherche à LR_id2 ;
- LR_id2 envoie l'adresse de MH_id à LR_id1 ;
- LR_id1 envoie l'adresse de MH_id à SB_id1.

Sinon

- LR_id1 cherche l'adresse de MH_id dans le cache
- Si elle existe, soit LR_x cette adresse, LR_id1 envoie un message de recherche à LR_x
- Si LR_x=LR_id2 **alors** LR_id2 envoie l'adresse de MH_id à LR_id1 qui l'envoie à SB-id1 sinon LR_x envoie un message d'échec.

Sinon :

- LR_id1 diffuse le message de recherche aux LR du profil de MH_id
- Si LR_id2 est dans le profil de MH_id,

  alors :

  LR_id2 envoie l'adresse de x à LR_id1 qui l'envoie à SB_id1, les autres LR envoient des messages d'échec

  sinon :

- LR_id1 envoie un message de recherche à $LR_{\cap}$;
- Soit LRj ϵ $LR_{\cap}$ / LRj ϵ QR_id2, LRj envoie un message de recherche à LR_id2 ;
- LR_id2 envoie l'adresse de MH_id à LRj ;
- LRj envoie l'adresse de MH_id à LR_id1 ;
- LR-id1 envoie l'adresse de MH_id à SB_id1

IV. COMPLEXITE DE L'ALGORITHME

Lors de la recherche ou de la mise à jours, un nombre de messages est envoyé. Ce nombre peut varier selon les cas de figures.

IV.1. Lors de la mise à jour

Les messages échangés entre les LR sont :

1. Si le mobile ne change pas de LR, aucun message n'est envoyé.
2. Si le mobile ne change pas de quorum, le LR de d'arrivée envoie $q-2$ messages aux LR du quorum pour qu'ils mettent à jour l'adresse du mobile.
3. Si le mobile change de quorum :

   Soit i le nombre de LR qui appartiennent à l'intersection du quorum de départ et d'arrivée et q la taille maximum d'un quorum.

   - $q-(i+1)$ messages de suppression sont envoyés aux LR du quorum de départ par le LR de départ ;
   - $q-(i+1)$ messages d'ajout sont envoyés aux LR du quorum d'arrivée par le LR d'arrivée ;
   - i messages de mise à jour sont envoyés aux LR de l'intersection des deux quorums par le LR d'arrivée.

   Le nombre total des messages envoyés dans ce cas est donc :

   $\{q-(i+1)\}+\{q-(i+1)\}+i=2*(q-1)-i$

   Au maximum ce nombre peut être égal à 2q-3 si $i=1$.

IV.2. Lors de la recherche

Les messages échangés lors d'une recherche sont :

1. Si le mobile appelant et le mobile appelé sont dans le même LR, le nombre de messages échangés est quatre (04) :
   - Un message (site mobile-station de base) : Le mobile appelant envoie un message de recherche du mobile appelé à sa station de base ;
   - Un message (station de base-registre de localisation) : Lorsque la station de base du mobile appelant reçoit la requête de celui-ci, elle envoie un message de recherche à son LR ;

- Un message (registre de localisation-station de base) : Lorsque le LR reçoit le message de recherche, il renvoie à la station de base du mobile appelant l'adresse du mobile appelé ;
- Un message (station de base-site mobile) : Lorsque la station de base reçoit l'adresse du mobile appelé de son LR, elle l'envoie au mobile appelant.

2. Si le mobile appelant et le mobile appelé sont dans le même quorum : le nombre de messages échangés est six (06) :

- Un message (site mobile-station de base);
- Un message (station de base-registre de localisation);
- Un message (registre de localisation-registre de localisation) : Le LR de la station de base du mobile appelant interroge le LR de la station de base du mobile appelé sur l'adresse de ce dernier ;
- Un message (registre de localisation-registre de localisation) : Le LR de la station de base du mobile appelé envoie l'adresse de celui-ci au LR de la station de base du mobile appelant ;
- Un message (registre de localisation-station de base) ;
- Un message (station de base-site mobile).

3. Si le mobile appelant et le mobile appelé sont dans deux quorums différents :

Si le LR appelant trouve l'adresse du mobile recherché (c'est à dire l'adresse de son LR, soit LR_MH) dans son cache, il envoie la requête de recherche à LR_MH. Si le mobile est localisé alors le nombre de messages échangés est six (06) :

- Un message (site mobile-station de base) ;
- Un message (station de base-registre de localisation);
- Un message (registre de localisation-registre de localisation) : Le LR de la station de base du mobile appelant interroge le LR du cache sur l'adresse du mobile appelé ;
- Un message (registre de localisation-registre de localisation) : Le LR du cache envoie l'adresse du mobile appelé au LR de la station de base du mobile appelant ;
- Un message (registre de localisation-station de base) ;
- Un message (station de base-site mobile).

Si la recherche dans le cache échoue et que le mobile appelé est dans l'un des LR du profil du LR appelant, le LR appelant interroge les LR du profil du mobile appelé. Si P est la taille du profil, le nombre de messages échangés est $2*(p+3)$:

- Un message (site mobile-station de base) ;
- Un message (station de base-registre de localisation) ;
- Un message (registre de localisation-registre de localisation) : Le LR de la station de base du mobile appelant interroge le LR du cache sur l'adresse du mobile appelé ;
- Un message (registre de localisation-registre de localisation) : Le LR du cache qui envoie un message d'échec au LR de la station de base du mobile appelant ;
- P messages (registre de localisation-registre de localisation) : Le LR de la station de base du mobile appelant interroge les P LR du profil sur l'adresse du mobile appelé ;
- P messages (registre de localisation-registre de localisation) : Les LR du profil renvoient au LR appelant $p-1$ messages d'échecs et un message qui contient l'adresse du mobile appelé;
- Un message (registre de localisation-station de base) ;
- Un message (station de base-site mobile).

Si le mobile n'est toujours pas localisé, le LR appelant diffuse la requête de recherche à tous les LR qui appartiennent à l'intersection de son quorum avec les autres quorums. Le nombre de ces LR est au maximum $(q-1)$ LR. Donc le nombre de messages échangés dans ce cas est $2*\{P+(2*q)+1\}$:

- Un message (site mobile-station de base) ;
- Un message (station de base-registre de localisation) ;
- Un message (registre de localisation-registre de localisation) : Le LR de la station de base du mobile appelant interroge le LR du cache sur l'adresse du mobile appelé ;
- Un message (registre de localisation-registre de localisation) : Le LR du cache qui envoie un message d'échec au LR de la station de base du mobile appelant ;
- P messages (registre de localisation-registre de localisation) : Le LR de la station de base du mobile appelant interroge les P LR du profil sur l'adresse du mobile appelé ;

- P messages (registre de localisation-registre de localisation) : Les LR du profil renvoient au LR appelant des messages d'échecs;
- $(q-1)$ messages (registre de localisation-registre de localisation) : Le LR appelant diffuse le message de recherche aux LR qui appartiennent à l'intersection de son quorum avec les autres quorums.
- $(q-1)$ messages (registre de localisation-registre de localisation) : au maximum, les $(q-1)$ LR connaissent le LR de la station de base du mobile appelé et lui envoient un message de recherche.
- $(q-1)$ messages (registre de localisation-registre de localisation) : Le LR de la station de base du mobile appelé renvoie l'adresse du mobile appelé aux $(q-1)$ LR qui l'ont interrogé ;
- $(q-1)$ messages (registre de localisation-registre de localisation) : Les $(q-1)$ LR renvoient l'adresse du mobile appelé au LR de la station de base du mobile appelant ;
- Un message (registre de localisation-station de base) ;
- Un message (station de base-site mobile).

V. COUT EN COMMUNICATION

Le coût de communication C_{Total} est la somme du coût de mise à jour de l'adresse d'un mobile C_U et le coût de recherche C_S.

V.1. Calcul du coût de recherche

Le coût de la communication dépend de la position du mobile appelé par rapport au mobile appelant.

1. La probabilité que le mobile appelant et le mobile appelé soient dans le même LR est $\frac{1}{R}$ avec R : le nombre de LR dans le réseau. Le coût de la communication dans ce cas est donc : $C1 = (1/R) * (2 * C_{wireless} + 2 * C_{fixed})$ avec $C_{wireless}$: le coût de communication sans fil entre un mobile et une station de base et C_{fixed} : Le coût de la communication filaire.

2. La probabilité que le mobile appelant et le mobile appelé soient dans le même quorum est $(1-1/R)*(q/R)$ avec q : le nombre maximum de LR dans un quorum. Le coût de la communication dans ce cas est : $C2 = (1-1/R)*(q/R)*(2*C_{wireless} + 4*C_{fixed})$.
3. La probabilité que le mobile appelant et le mobile appelé soient dans deux quorums différents est : $(1-1/R)*(1-q/R)$

   3.1. La probabilité que le mobile appelant et le mobile appelé soient dans deux quorums différents et que l'adresse du mobile appelé soit dans le cache est :
   $(1-1/R)*(1-q/R)*P_{cache}$ avec P_{cache} la probabilité que l'adresse du mobile appelant soit dans le cache.
   Le coût de la communication dans ce cas est :
   $C3 = (1-1/R)*(1-q/R)*P_{cache}*(2*C_{wireless} + 4*C_{fixed})$.

   3.2. La probabilité que le mobile appelant et le mobile appelé soient dans deux quorums différents et que le mobile appelé soit dans l'un des LR du profil est :
   $(1-1/R)*(1-q/R)*(1-P_{cache})*P_{profile}$ avec P_{profil} la probabilité que l'adresse du mobile appelé soit dans le profil.
   Le coût de la communication dans ce cas est :
   $C4 = (1-1/R)*(1-q/R)*(1-P_{cache})*P_{profile}*(2*C_{wireless} + 2*(P+2)*C_{fixed})$.

   3.3. La probabilité que le mobile appelant et le mobile appelé soient dans deux quorums différents et que l'adresse du mobile appelé ne soit ni dans le cache ni dans l'un des LR du profil est $(1-1/R)*(1-q/R)*(1-P_{cache})*(1-P_{profile})$.
   Le coût de la communication dans ce cas est :
   $C5 = (1-1/R)*(1-q/R)*(1-P_{cache})*(1-P_{profile})*(2*C_{wireless} + 2*(2*q+P)*C_{fixed})$.

Le coût total de la recherche C_S est :
$C_S = C1 + C2 + C3 + C4 + C5$ et donc :

$C_S = (1/R)*(2*C_{wireless}+2*C_{fixed}) + (1-1/R)*((q/R)*(2*C_{wireless}+4*C_{fixed}) + (1-q/R)*(P_{cache}*(2*C_{wireless}+4*C_{fixed}) + (1-P_{cache})*(P_{profil}*(2*C_{wireless}+2*(P+2)*C_{fixed}) + (1-P_{profil})*(2*C_{wireless}+2*(2q+P)*C_{fixed}))))$

V.2. Calcul du coût de la mise à jour

Soit G le nombre moyen de handoffs durant un « RA crossing » avec $G = \frac{1}{C} * \sum_{i=1}^{C} i = \frac{C+1}{2}$

Soit H le nombre moyen de handoffs durant un « quorum crossing » avec $H = \frac{1}{T} * \sum_{i=1}^{T} i = \frac{T+1}{2}$

C représente le nombre de stations de base (ou de cellules) dans un LR.

T représente le nombre de stations de base dans un quorum. D'où $T = C * q$

1. La probabilité que le mobile traverse un quorum lors de son déplacement est $(1/H)$. Le coût de mise à jour au maximum est donc $C1 = (1/H) * (2 * q - 3) * C_{fixed}$.
2. La probabilité que le mobile ne change pas de quorum lors de son déplacement mais change de LR est $(1 - 1/H) * 1/G$. Le coût de mise à jour est donc $C2 = (1 - 1/H) * (1/G) * (q - 2) * C_{fixed}$.
3. La probabilité que le mobile ne change ni de quorum ni de LR lors de son déplacement (il change de station de base seulement) est $(1 - 1/H) * (1 - 1/G)$. Dans ce cas aucun message n'est envoyé.

Le coût de mise à jour par handoff est :

$C_U = C1 + C2$ et donc :

$$C_U = [((1/H) * (2 * q - 3)) + ((1 - 1/H) * (1/G) * (q - 2))] * C_{fixed}$$

VI. CONSTRUCTION DES QUORUMS

La performance du schéma de localisation dépend de la construction des quorums.

Pour minimiser le coût de communication et le nombre de messages envoyés lors d'une recherche ou d'une mise à jour, il faut réduire la taille des quorums.

Le schéma choisi pour la construction des quorums est la construction en arbre. Lorsque l'arbre est binaire, la taille d'un quorum est approximativement $Log_2 N$ où N est le nombre

de LR dans le réseau. De même, si l'arbre est d[aire], la taille d'un quorum est approximativement $Log_d N$. De plus, le schéma en arbre est une coterie non dominée.

Pour construire cet arbre logique, il faut organiser les nœuds de telle sorte que la coterie soit une *mean-delay optimal coterie,* sachant que pour une organisation physique en arbre le calcul d'une telle coterie est $O(N)$.

VII. COMPARAISON AVEC L'ALGORITHME DE IHN-HAN BAE

La première différence entre l'algorithme de Ihn-Han Bae et notre solution est que Ihn-Han Bae associe à chaque LR deux quorums un quorum ligne et un quorum colonne. Lors d'un déplacement, ces quorums sont mis à jour suivant l'état du mobile concerné. Si ce mobile est *hot*, les deux quorums ligne et colonne sont mis à jour. S'il est '*cold*', c'est l'un des deux quorums qui mis à jour. Dans notre solution, la notion de quorum de lecture et d'écriture ou de quorum ligne et colonne n'existe pas. L'ensemble des LR du réseau est organisé en quorums où chaque LR peut appartenir à plusieurs quorums et est affecté à un seul quorum. Lors d'une recherche, un LR interroge les autres LR appartenant à son quorum.

De plus, la mise à jour dans la solution de Ihn-Han Bae se fait lorsqu'un mobile change de LR. Dans notre solution, l'adresse d'un mobile est mise à jour lorsqu'il se déplace d'un quorum à un autre.

Enfin, la construction des quorums utilisée est différente. Ihn-Han Bae a utilisé la construction basée sur le triangle (construction en ligne et en colonne) où les quorums ont la même taille qui est approximativement $\sqrt{2}\sqrt{N}$. Dans notre solution, la construction utilisée est le schéma en arbre où la taille des quorums est approximativement $Log_d N$ avec N le nombre de LR dans le réseau et d le degré de l'arbre (le nombre de fils que possède un nœud interne de l'arbre).

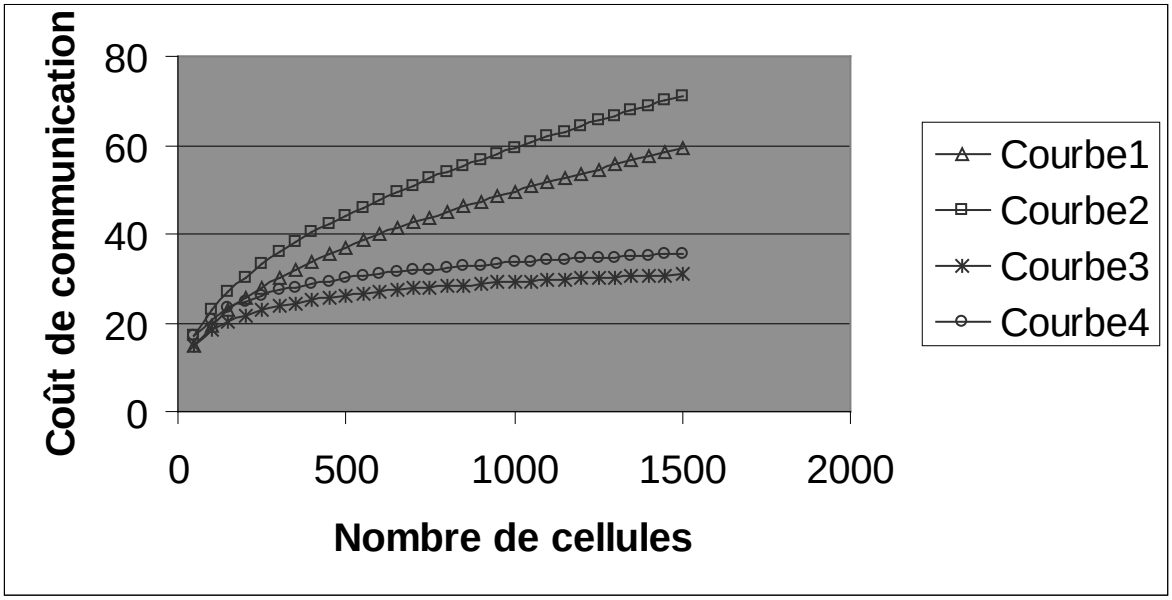
Dans ce qui suit, nous présentons une comparaison entre les performances de l'algorithme de Ihn-Han Bae et notre solution dans les conditions qu'a utilisé Ihn-Han Bae pour comparer son algorithme à celui de Prakash. A savoir : C_{fixed}=1, $C_{wireless}$=3, C=7, P=3, P_{cache}=0.2, P_{profil}=0.3 et P_{Hot}=0.5.

Pour évaluer notre solution, nous la comparons en terme de coût de communication avec la solution de Ihn-Han Bae dans le cas où le schéma utilisé pour la construction des quorums est le schéma en triangle (Ici la taille d'un quorum est $q = \sqrt{2}\sqrt{n/C}$) et dans le cas où c'est le

schéma en arbre (Ici la taille d'un quorum est $q = Log_2(n/C)$) avec n : le nombre de stations de base dans le réseau.

La Figure 5.2 montre la différence de performances entre les deux solutions.

- Courbe1 :représente la performance de notre solution avec $q = \sqrt{2}\sqrt{n/C}$
- Courbe2 :représente la performance de la solution de Ihn-Han Bae avec $q = \sqrt{2}\sqrt{n/C}$
- Courbe3 :représente la performance de notre solution avec $q = Log_2(n/C)$
- Courbe4 :représente la performance de la solution de Ihn-Han Bae avec $q = Log_2(n/C)$



- Figure 5.2 -

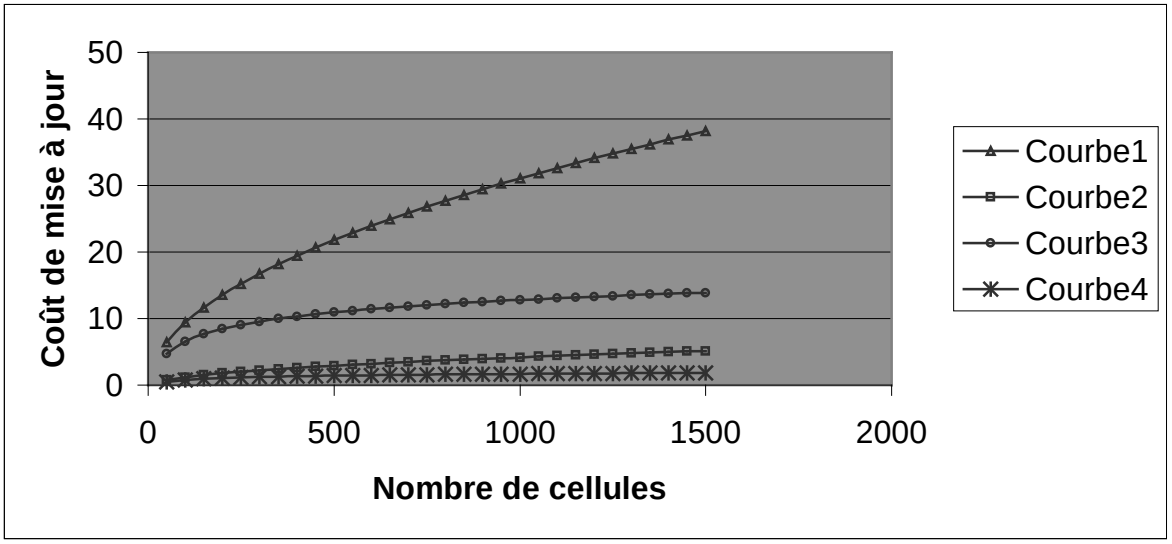
D'abord, il est clair que le choix de la construction de quorum est très important. Dans les deux solutions, l'écart du coût de communication est considérable entre la construction en arbre et la construction en triangle.

Ensuite, que ce soit dans le cas de la construction en triangle ou la construction en arbre, notre solution donne de meilleurs résultats que celle de Ihn-Han Bae même si l'écart entre les deux solutions n'est pas très important dans le cas de la construction en arbre.

Cette figure nous montre que dans l'ensemble notre solution est meilleure que celle de Ihn-Han Bae. Figure 5.3 et Figure 5.4 nous montre le détail de cette courbe en comparant le coût de recherche et de mise à jour indépendamment.

Figure 5.3 compare le coût de mise à jour entre les deux solution telle que :

- Courbe1 :représente la performance de la solution de Ihn-Han Bae avec $q = \sqrt{2}\sqrt{n/C}$
- Courbe2 :représente la performance de notre solution avec $q = \sqrt{2}\sqrt{n/C}$
- Courbe3 :représente la performance de la solution de Ihn-Han Bae avec $q = Log_2(n/C)$
- Courbe4 :représente la performance de notre solution avec $q = Log_2(n/C)$

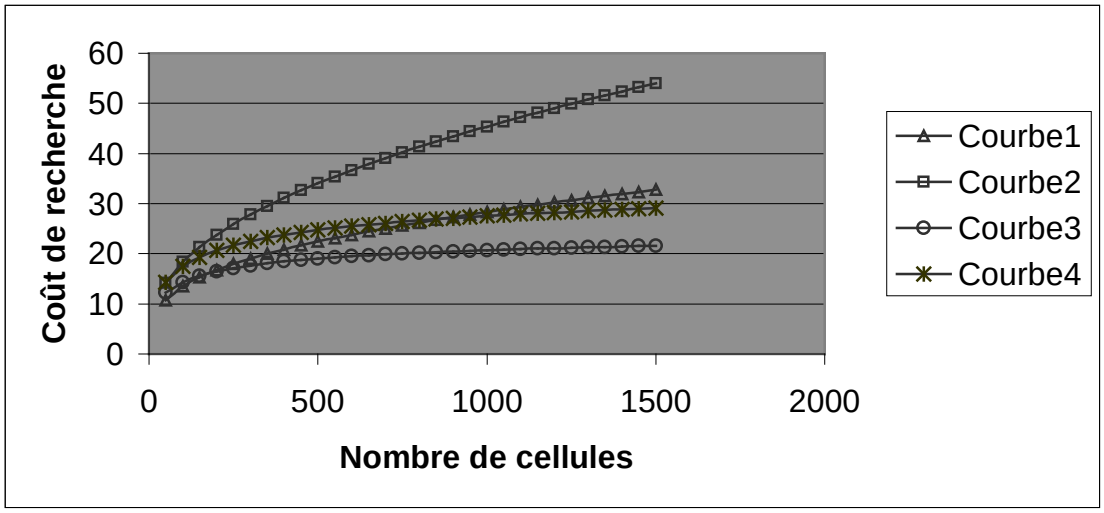


- Figure 5.3 -

Notre solution réduit énormément le coût de mise à jour par rapport à la solution de Ihn-Han Bae. En fait, le coût de la mise à jour de notre solution avec une construction en triangle est plus faible par rapport au coût de mise à jour de la solution de Ihn-Han Bae avec une construction en arbre et ce sachant que la taille d'un quorum dans la construction en triangle est $q = \sqrt{2}\sqrt{n/C}$ et que celle d'un quorum dans la construction en arbre est $q = Log_2(n/C)$. Ceci est dû au fait que dans le cas de la solution de Ihn-Han Bae, lorsqu'un mobile est hot, quatre quorums sont mis à jour et deux seulement s'il est cold alors que dans notre solution deux quorums seulement sont mis à jours dans tous les cas. De plus, un mobile a plus de chance de passer d'un LR à un autre que de passer d'un quorum à un autre. Aussi, le nombre de mises à jour dans notre solution est inférieur au nombre de mises à jour dans la solution de Ihn-Han Bae.

Figure 5.4 compare les deux solutions en coût de recherche telle que :

- Courbe1 :représente la performance de la solution de Ihn-Han Bae avec $q = \sqrt{2}\sqrt{n/C}$
- Courbe2 :représente la performance de notre solution avec $q = \sqrt{2}\sqrt{n/C}$
- Courbe3 :représente la performance de la solution de Ihn-Han Bae avec $q = Log_2(n/C)$
- Courbe4 :représente la performance de notre solution avec $q = Log_2(n/C)$



- Figure 5.4 -

Dans le cas de la construction en triangle comme dans le cas de la construction en arbre, l'écart entre notre solution et celle de Ihn-Han Bae est considérable en sa faveur. Dans l'ensemble, la solution de Ihn-Han Bae (pour une construction en triangle) est meilleure que la notre (pour une construction en arbre) pour de petits réseaux avec un nombre de cellules qui ne dépasse pas 800. Par contre pour les grands réseaux qui dépassent les 800 cellules notre solution donne un coût de recherche meilleur.

VIII. SOLUTION AMELIOREE

Il est clair d'après la Figure 5.4 que le coût de recherche est assez important même si le coût de communication total de notre solution est meilleur que celui de celle de Ihn-Han Bae. Pour réduire le coût de recherche, il faut revoir la politique de recherche.

VIII.1. Principe

Lorsqu'un nœud x recherche un mobile y et trouve le LR dans lequelle il se trouve, la recherche s'arrête ici, pas besoin de chercher la station de base dans laquelle il se trouve. Si x veut envoyer un message à y, x envoie ce message non à la station de base qui le transmet à y mais au LR dans lequel se trouve y. Celui-ci se charge d'acheminer ce message vers la station de base dans laquelle se trouve y. Aussi la procédure de recherche devient :

VIII.2. Recherche d'un mobile

Lorsqu'un mobile y qui se trouve dans une station de base A veut localiser un mobile x qui se trouve dans la station de base B. Si A (resp. B) appartient à la zone d'enregistrement RA_A (resp. RA_B) qui possède le registre de localisation LR_A (resp. LR_B), A demande à LR_A l'adresse de x. Cinq cas sont possibles :

1. Si x et y se trouvent dans LR_A (i.e. LR_A=LR_B), alors LR_A envoie l'adresse de x (ici B) à A
2. Si LR_A et LR_B sont dans le même quorum, alors LR_A possède l'adresse de x en terme de LR, LR_A envoie cette adresse à A.
3. Si LR_A et LR_B ne sont pas dans le même quorum :
   3.1. LR_A consulte son cache pour voir s'il contient une adresse de x. Si x possède une entrée dans le cache de LR_A (soit LR_x cette entrée), LR_A envoie un message de recherche à LR_x. Si x est toujours dans LR_x, LR_x envoie à LR_A l'adresse de x qui le renvoie à A sinon il lui envoie un message d'échec.
   3.2. Si LR_A ne trouve pas l'adresse de x dans son cache, LR_A diffuse la requête de recherche aux LR du profil de x. Lorsqu'un LR reçoit le message de recherche, si x se trouve dans ce LR, ce dernier envoie l'adresse de x à LR_A qui l'envoie à A sinon il lui envoie un message d'échec.
   3.3. Si LR_A ne trouve pas l'adresse de x dans le cache et que tous les LR associés à x dans le profil répondent négativement à son message de recherche, alors LR_A ne possède pas l'information de l'adresse de x. LR_A diffuse un message de recherche à l'ensemble $LR_\cap$ qui contient tous les LR qui appartiennent à l'intersection du quorum de LR_A avec les autres quorums du réseau. Chaque LR de $LR_\cap$ envoie l'adresse de x (c'est à dire le LR dans lequel se trouve x) à LR_A s'il la possède, il lui envoie un message d'échec sinon. Lorsque LR_A reçoit l'adresse de x, il l'envoie à A.

Procédure de recherche

Soit SB_id1 (resp. SB_id2) la station de base qui recherche (resp. dans laquelle se trouve) un mobile x dont l'identité est MH_id.

Soit LR_id1 (resp. LR_id2) la LR qui contient SB_id1 (resp. SB_id2).

Soit QR_id1 (resp. QR_id2) le quorum auquel appartient LR_id1 (resp. LR_id2).

Soit $LR_{\cap}$ les LR qui appartiennent à l'intersection de QR_id1 avec les autres quorums du réseau.

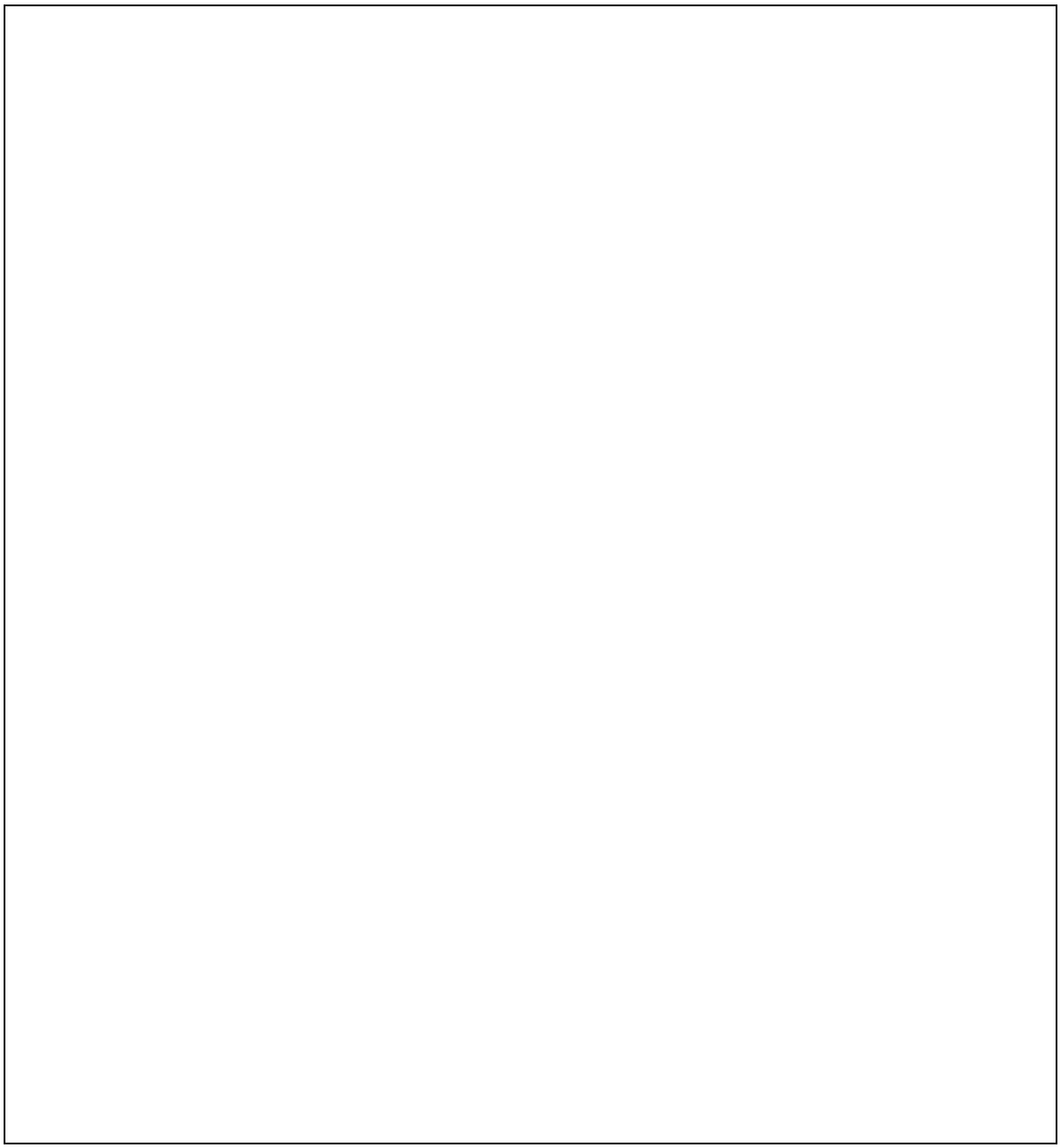
Si LR_id1=LR_id2 **alors** LR_id1 envoie l'adresse de MH_id (ici SB_id2) à SB_id1

Sinon Si QR_id1=QR_id2 **alors**

- LR_id1 envoie l'adresse de MH_id (ici LR_id2) à SB_id1.

Sinon

- LR_id1 cherche l'adresse de MH_id dans le cache
- Si elle existe, soit LR_x cette adresse, LR_id1 envoie un message de recherche à LR_x
- Si LR_x=LR_id2
  alors LR_id2 envoie l'adresse de MH_id à LR_id1 qui l'envoie à SB-id1
  sinon LR_x envoie un message d'échec.
- LR_id1 diffuse le message de recherche aux LR du profil de MH_id
- Si LR_id2 est dans le profil de MH_id,
  alors LR_id2 envoie l'adresse de x à LR_id1 qui l'envoie à SB_id1, les autres LR envoient des messages d'échec.
- LR_id1 envoie un message de recherche à $LR_{\cap}$;
- Soit LRj ϵ $LR_{\cap}$ / LRj ϵ QR_id2,
- LRj envoie l'adresse de MH_id (LR_id2)à LR_id1 ;
- LR-id1 envoie l'adresse de MH_id à SB_id1



VIII.3. Complexité de la procédure de recherche

1. Si le mobile appelant et le mobile appelé sont dans le même LR, le nombre de messages échangés est toujours quatre (04).
2. Si le mobile appelant et le mobile appelé sont dans le même quorum : le nombre de messages échangés passe de six (06) à quatre (04) :
   - Un message (site mobile-station de base);
   - Un message (station de base-registre de localisation);
   - Un message (registre de localisation-station de base) : Comme le LR de la station de base appelant connaît le LR dans lequel se trouve le mobile appelé, il envoie cette adresse à la station de base .

- ❖ Un message (station de base-site mobile).

3. Si le mobile appelant et le mobile appelé sont dans deux quorums différents :
   3.1. Si le LR appelant trouve l'adresse du mobile recherché (l'adresse de son LR) dans son cache, et soit LR_MH cette adresse, il envoie la requête de recherche à LR_MH. Si le mobile est localisé alors le nombre de messages échangés reste six (06) ;
   3.2. Si la recherche dans le cache échoue et que le mobile appelé est dans l'un des LR du profil du LR appelant, le LR appelant interroge les LR du profil du mobile appelé. Si P est la taille du profil, le nombre de messages échangés reste $2*(p+3)$;
   3.3. Si le mobile n'est toujours pas localisé, le LR appelant diffuse la requête de recherche à tous les LR qui appartiennent à l'intersection de son quorum avec les autres quorums. Le nombre de ces LR est au maximum $(q-1)$ LR. Donc le nombre de messages échangés dans ce cas passe de $2*(P+(2*q)+1)$ à $2*(P+q+2)$:
      - ❖ Un message (site mobile-station de base) ;
      - ❖ Un message (station de base-registre de localisation) ;
      - ❖ Un message (registre de localisation-registre de localisation) : Le LR de la station de base du mobile appelant interroge le LR du cache sur l'adresse du mobile appelé ;
      - ❖ Un message (registre de localisation-registre de localisation) : Le LR du cache qui envoie un message d'échec au LR de la station de base du mobile appelant ;
      - ❖ P messages (registre de localisation-registre de localisation) : Le LR de la station de base du mobile appelant interroge les P LR du profil sur l'adresse du mobile appelé ;
      - ❖ P messages (registre de localisation-registre de localisation) : Les LR du profil renvoient au LR appelant des messages d'échecs;
      - ❖ $(q-1)$ messages (registre de localisation-registre de localisation) : Le LR appelant diffuse le message de recherche aux LR qui appartiennent à l'intersection de son quorum avec les autres quorums.
      - ❖ $(q-1)$ messages (registre de localisation-registre de localisation) : Les $(q-1)$ LR renvoient soit l'adresse du mobile appelé (le LR dans lequel il se trouve) au

LR de la station de base du mobile appelant soit un message d'échec, sachant qu'au moins un LR n'envoie pas de message d'échec ;

- ❖ Un message (registre de localisation-station de base) ;
- ❖ Un message (station de base-site mobile).

VIII.4. Calcul du coût de recherche

Le coût de la recherche dans le cas où le mobile appelant et le mobile appelé sont dans :

1. le même LR : il ne change pas.
2. le même quorum est : $C2 = (1-1/R)*(q/R)*(2*C_{wireless} + 2*C_{fixed})$.
3. deux quorums différents et que :
   3.1. l'adresse du mobile appelé soit dans le cache : il ne change .pas
   3.2. le mobile appelé soit dans l'un des LR du profil : il ne change pas.
   3.3. l'adresse du mobile appelé ne soit ni dans le cache ni dans l'un des LR du profil est:

$$C5 = (1-1/R)*(1-q/R)*(1-P_{cache})*(1-P_{profile})*(2*C_{wireless} + 2*(q+P+1)*C_{fixed}).$$

Le coût total de la recherche C_S est :

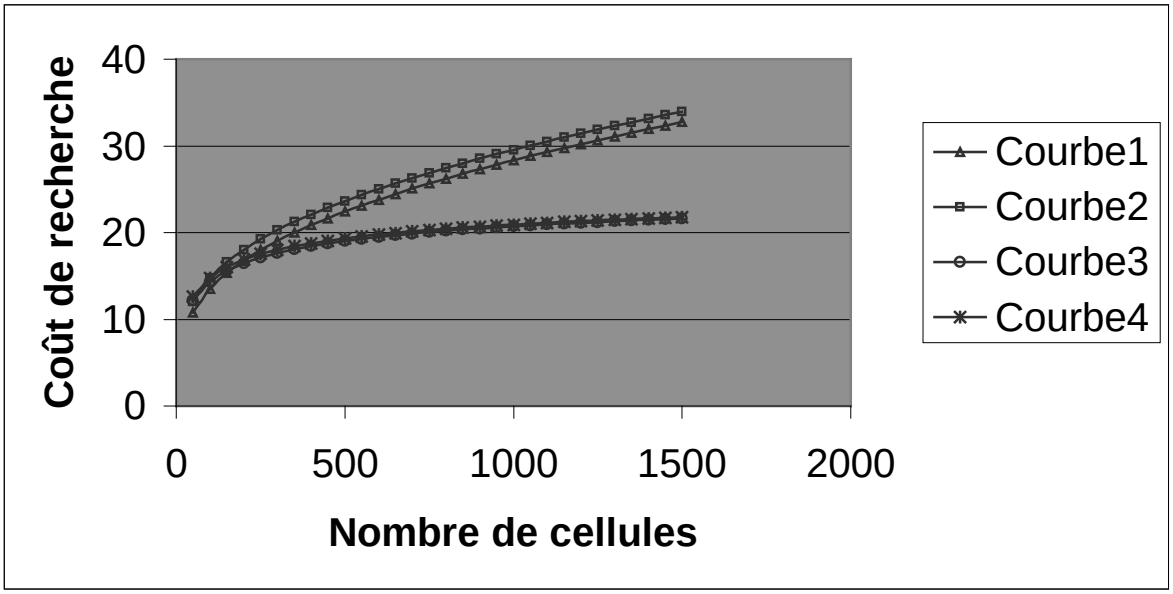
$C_S = C1 + C2 + C3 + C4 + C5$ et donc :

$$C_S = (1/R)*(2*C_{wireless}+2*C_{fixed})+ (1-1/R) *((q/R)*(2*C_{wireless}+2*C_{fixed})+ (1-q/R)* (P_{cache} *(2*C_{wireless}+4*C_{fixed})+ (1-P_{cache})*(P_{profil} *(2*C_{wireless}+2*(P+2)*C_{fixed})+ (1-P_{profil})*(2*C_{wireless}+2*(q+P+1)*C_{fixed}))))$$

VIII.5. Comparaison avec la solution de Ihn-Han Bae

Figure 5.5 compare les deux solutions en coût de recherche telle que :

- Courbe1 :représente la performance de la solution de Ihn-Han Bae avec $q = \sqrt{2}\sqrt{n/C}$
- Courbe2 :représente la performance de notre solution avec $q = \sqrt{2}\sqrt{n/C}$
- Courbe3 :représente la performance de la solution de Ihn-Han Bae avec $q = Log_2(n/C)$
- Courbe4 :représente la performance de notre solution avec $q = Log_2(n/C)$

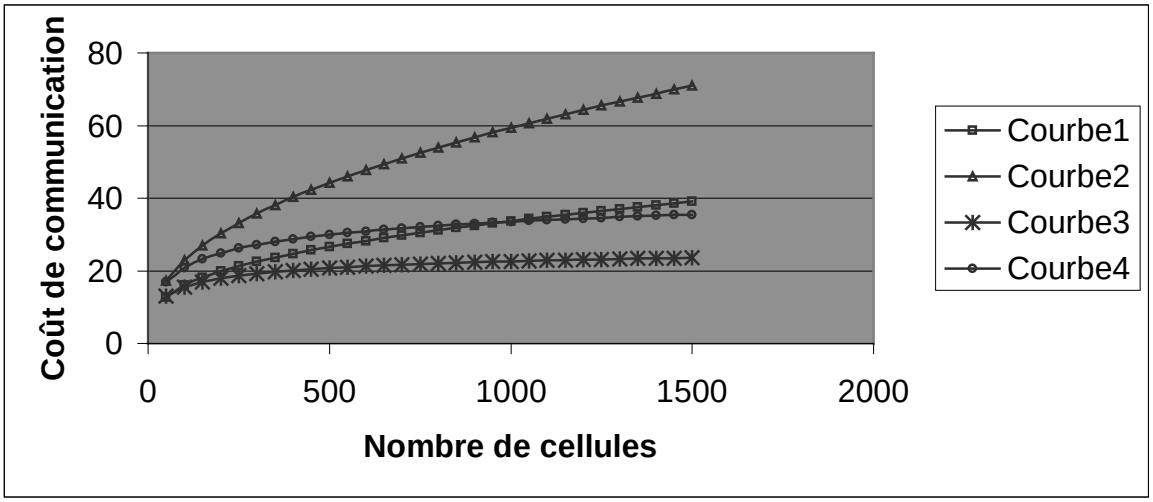


- Figure 5.5 -

Dans le cas de la construction en triangle, la solution de Ihn-Han Bae est sensiblement meilleure que la notre. Mais dans la construction en arbre, les deux courbes sont presque identiques avec un léger avantage pour la courbe de Ihn-Han Bae.

En réduisant le coût de recherche, nous avons automatiquement réduit le coût de communication. Figure 5.6 compare les deux solutions en coût de communication telle que :

- Courbe1 :représente la performance de notre solution avec $q = \sqrt{2}\sqrt{n/C}$
- Courbe2 :représente la performance de la solution de Ihn-Han Bae avec $q = \sqrt{2}\sqrt{n/C}$
- Courbe3 :représente la performance de notre solution avec $q = Log_2(n/C)$
- Courbe4 :représente la performance de la solution de Ihn-Han Bae avec $q = Log_2(n/C)$



- Figure 5.6 -

Le coût de communication total entre notre solution et celle de Ihn-Han Bae est plus important.

IX. CONCLUSION

Dans ce chapitre, nous avons proposé un schéma de gestion de localisation basé sur les quorums où la construction choisie pour les quorums est la construction en arbre binaire. L'information de localisation d'un mobile est sauvegardée dans un ensemble de taille $O(\log_2 n)$: un quorum. Cette information est aussi mobile. Elle se déplace avec les déplacements du mobile, l'adresse d'un mobile est mise à jour lorsqu'il va d'un quorum à un autre. Le coût de communication total induit par notre solution est plus petit que celui induit par celle de Ihn-Han Bae.

La gestion de la localisation des objets est l'un des problèmes les plus importants dans les réseaux mobiles. Plusieurs solutions ont été proposées pour le résoudre. Les deux solutions les plus utilisées sont le schéma à deux niveaux et le schéma hiérarchique.

Il existe plusieurs critères pour évaluer les solutions proposées. L'un des critères les plus significatifs est le coût de communication.

La recherche et la mise à jour dans le schéma à deux niveaux et le schéma hiérarchique impliquent un coût de communication élevé. Aussi, plusieurs améliorations ont été proposées pour le réduire telles que l'utilisation du cache, des Forwarding Pointers, de la réplication...

Le choix de ces techniques dépend de plusieurs paramètres dont le profil des appels et mouvements des mobiles qui permet d'exploiter la distribution géographique des mobiles dans le système et la localité des appels et déplacements pour réduire le coût de recherche et de mise à jour.

Dans cette thèse, nous avons proposé une architecture basée sur les quorums pour la gestion de la localisation. Nous avons évalué les algorithmes de recherche et de mise à jour en terme de complexité de message et de coût de communication total. Nous avons montré que notre solution réduit le coût de communication total par rapport à celle de Ihn-Han Bae. Mais l'étude du coût de recherche et du coût de mise à jour chacun à part a montré que notre solution réduit le coût de mise à jour considérablement alors que le coût recherche reste sensiblement le même. Aussi, pour les mobiles dont le nombre de déplacements est plus important que le nombre de recherches reçues, le coût de communication induit par notre solution est très faible par rapport celui induit par la solution de Ihn-Han Bae alors que dans le cas inverse (mais pour des rapports très importants), la différence entre les deux coûts n'est plus aussi grande.

En conclusion, le coût de communication dépend de l'architecture choisie, des algorithmes de recherche et mise à jour définis sur cette architecture et des techniques utilisées pour améliorer les procédures de recherche et mise à jour.

Il dépend aussi du système de quorums choisi. En effet, le schéma en arbre où la taille d'un quorum est $O(Log_2\ n)$ réduit le coût de communication par rapport au schéma triangulaire où la taille d'un quorum est $O(\sqrt{n})$ alors que la charge induite par le schéma en arbre est plus importante. Aussi, et comme perspectives nous proposons de creuser d'avantage pour définir un schéma de quorums qui permette un meilleur compromis charge et taille de quorum.

De plus, pour évaluer la performance de notre solution, nous avons supposé que les mobiles sont uniformément distribués dans les LR. Il serait intéressent de simuler le comportement de nos algorithmes pour des cas réels qui ne vérifient pas forcément cette hypothèse.

BIBLIOGRAPHIE

[1] E.Pitoura and Samaras.*Locating Objects in Mobile Computing*. IEEE Transactions on Knowledga and Data Engineering, 2000.

[2] Ihb-Han Bae. *A quorum-Based Dynamic Location Management Method for Mobile computings.* Dept. Of Computer Enineering, Catholic University of Taegu-Hyosung. Kyungsan, Kyungpook 712-702, Korea. 1999.

[3] Sun-Jin Oh, Jae-Gyeong Lee, Ihb-Han Bae. *A distributed dynamic Location Management Scheme for Mobile computing Systems*. 2000

[4] Moni Noar and Avishai Wool. *The load, Capacity and Availability of Quorum Systems.* Proceedings of the 35th IEEE Symposium of the Foundations of Computer Science (FOCS), pp. 214-225,1994.

[5] Mitchell L. Neilson. *Quorum Structures in Distributed Systems*. PhD thesis, Dept. Computing and Information Sciences, Kansas State University. 1992

[6] O. Wolfson, B. Xu and L. Jiang. *Moving ObjectsDatabases: Issues and Solutions.* Proceeding of the 10th International Conference on Scientific and Statistical Database Management (SSDBM98), Capri, Italy, July 1-3, 1998, pp. 111-122.

[7] O. Wolfson, P. Sistla, B. Xu, J. Zhou, S. Chamberlin, Y. Yesha and N. Rishe. *Tracking Moving Objects Using Database Technology in DOMINO.* 2000.

[8] T. Gherbi. *Structuration d'applications réparties dans un environnement mobile*. Thèse de Magistère, U.S.T.H.B., 2000.

[9] B. Awerbuch and D. Peleg. *Online Tracking of Mobile Users*. Journal of the ACM, 42(5), 1995.

[10] Ravi Prakash, Mukesh Singhal. *A dynamic Approch to Location Management in Mobile computing systems*. Dept. Of Computer and Information Science. The Ohio State University. Columbus, Ohio 43210. 1996.

[11**]** Ravi Prakash, Mukesh Singhal. *Dynamic Hashing + Quorum = Efficient Location Management for Mobile Computing Systems.* 1997.

[12**]** Ravi Prakash, Mukesh Singhal, Zygmunt Haas. *Load-Balanced Location Management for Cellular Mobile systems using Quorums and Dynamic Hashing*. Wireless Networks 0, 2001.

[13**]** Takashi Harada and Masashita. *Nondominated Coteries on Graphs*.

[14**]** David Peleg Avishai Wool. *Crumbling Walls: A Class of High Availability Quorum Systems*. March 7, 1994.

[15**]** David Peleg Avishai Wool. *Crumbling Walls: A Class of Practical and Efficient Quorum Systems*. August 14, 1996.

[16**]** Akhil Kumar, Tracy Camp. *A new Matching Algorithm for Managing Location Information in Mobile Computing*. 2000.

[17**]** P.Krishna, N. H. Vaidra and D. K. Pradhan. *Static and Dynamic Location Management in Mobile Wireless Networks*. Journal of Computer Communications (spatial issue on mobile Computing), 19(4), Mach 1996.

[18**]** S. Rajagopalan and R. Badrinath. *An Adaptative Location Management Strategy for Mobile IP*. Proceeding of First ACM Mobicom 95, November 1995.

[19**]** EIA/TIA IS-41.3 (Revision B), *Cellular Radio Telecommunications Intersystem Operations*, July 1991.

[20**]** M. Mouly and M. B. Pautet. *The GSM System for mobile Communications*. Book published by authors, 49, rue Louise Bruneau, F-91120 Palaiseau, France, 1992. ISBN 2-9507190-0-7.

[21**]** M.Van Steen, F.J.Hauck and A.S.Tanenbaum. A model for Worldwide Tracking of Distributed Objects.1996.

[22**]** M. Van Steen, F.J.Hauck, G.Ballintijin and A.S.Tanenbaum. *Algorithmic Design of Globe Wide-Area Location Service*. The Computer Journal, Vol. 41, No. 5, 1998.

[23**]** G.Ballintijin, M.Van Steen and A.S.Tanenbaum. *Exploiting Location Awareness for Scalable Location-Independent Object Ids.* In Proc. 5th ASCI Annual Conference, pp. 321-328, Heijen, The Netherlands, June 1999. ASCI.

[24**]** A.Baggio, G.Ballintijin, A.S.Tanenbaum, M.Van Steen. *Efficient Tracking of Mobile Objects in Globe*. Internal report IR-481, November 2000.

[25**]** A.Baggio, G.Ballintijin and M.Van Steen. *Mechanisms for effective caching in the Globe location Service.* 2000.

[26**]** P.Krishna. *Mobility management and Tolerance in Wireless Networks.* September, 1995.

[27**]** GuoHui Li, Kam-Yiu Lam and Tei-Wei Kuo. *Location Update Generation in Cellular Mobile Computing Systems.* 2000.

[28**]** Divyakant Agrawal and Amr el Abbadi. *An efficient and fault-tolerant Solution for Distributed Mutual Exclusion*. ACM Transactions on Computer Sysytems, pages 1-20, February . 1991.

[29**]** Mitchell L. Nelson and Masaaki Misuno. *Nondominated K-Coteries for Multiple Mutual Exclusion*. Information Processing Letters, Vol. 30, pp 247-252, 1994.

[30**]** Roberto Baldoni, Yoshifumi Manabe Michel Raynal and Shigemi Aoyagi. *K-arbiter: A safe and general Scheme for h-out of-k mutual exclusion problems*. N° 2523. 1995.

[31**]** Divyakant Agrawal and Amr el Abbadi.

[32**]** Divyakant Agrawal and Amr el Abbadi. *Analyse des quorums et protocoles basés sur l'exclusion distribuée d'ordre (k+1*).

[*33***]** Guohong Cao, Mukesh Singhal. *An efficient Coterie-Based Mutual Exclusion Scheme With Fault-Tolerance Capability.*1995.

[*34***]** Ada Waichee Fu. *Delay-Optimal Quorum Consensus for Distributed Systems*. IEEE Transactions on parallel and distributed systems, Vol. 8, N° 1, January 1997.

[*35***]** Dahlia Malkhi. *Quorum systems*. AT & T Labs-research. March 23 1999.

[36**]** Divyakant Agrawal, Omer Egecioglu and Amr el Abbadi. *Billiard Quorums on the Grid*. Information Processing Letter 64, page 9-16, 1997.

[*37***]** G. Krishnamurthi, M. Azizogluand A. K. Somani. *Optimal location management algorithms for mobile networks*. Proceedings of the fourth Annual ACM International Conference on Mobile Computing and Networking (MOBICOM'98), pp. 223-232, 1998.

[*38***]** M. Maekawa. *A$\sqrt{N}$ Algorithm for Mutual Exclusion in Decentralized systems.*. ACM Transactions on Computer Systems, 3(2):145-159, May 1985.

[*39***]** W. S. Vincent, Wong and victor C. M. Leung. *Location management for Next-Generaion Personal Commnications Networks.* IEEE Network, September/October 2000.

[*40***]** R. Shankaran, V.Varadharajan, M. Hitchens. *A distributed Location Management Scheme for Mobile Hosts.* Departement of Computing Macquarie University, Australia. 0-7695-1153-8/01, IEEE, 2001.

[*41***]** R. Subrata, A. Y. Zomaya. *Location Management in mobile Computing: A Survey.* Parallel computing Reseach Laboratory, Departement of Electrical and Electronic engineering, The university of Western australia. Western australia 6907. 2000.

[42**]** Moni Noar and Avishai Wool. The load, Capacity and Availability of Quorum Systems.1998 Society for Industrial and Applid Mathemtics 006. SIAM J. COMPUT. Vol. 27, N° 2, pp. 423-447, April 1998.

[43**]** I. Demirkol[1], C. Ersoy[1], M. U. Caglayan[1], H. Delic[2]. *Location area Planning in Cellular Networks Using Simulated Annealing.* [1]NETLAB, Departement of Computer Engineering. [2]BUSIM Lab, deprtement of electrical an Electronic engineering. Bogazici University, Bebek 80815 Istanbul, Turkey. 2000.

[44**]** S. Cheung, M.H Ammar, M. Ahamad. *The Grid Protocol: A high Performance Scheme for Maintaining Replicated Data.* IEEE Transactions on Knowledge and Data engineering, 4(6):582-592,1992.