N° d'ordre : ……/2007-E/IN

REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE
MINISTERE DE L'ENSEIGNEMENT SUPERIEUR ET DE LA RECHERCHESCIENTIFIQUE

Université des Sciences et de la Technologie
Houari Boumediene (U.S.T.H.B.) Alger
Faculté d'Electronique et d'Informatique
Département Informatique

THÈSE

Présentée pour l'obtention du diplôme de : **DOCTORAT D'ÉTAT**

EN: **INFORMATIQUE**

Par

Samira MOUSSAOUI - BOUALLAG

SUJET

Contribution à l'Amélioration de l'Accessibilité des Données sur Réseaux Mobiles Ad Hoc

Soutenue publiquement le 17 Décembre 2007 devant le Jury:

Mme	A.	MOKHTARI AÏSSANI	Professeur (USTHB)	Présidente
Mr	N.	BADACHE	Professeur (USTHB)	Directeur de thèse
Mr	M.	AHMED NACER	Professeur (USTHB)	Examinateur
Mme	M.	BOUKALA	Professeur (USTHB)	Examinateur
Mr	A.	BOUABDALLAH	Professeur (Univ. Compiègne, France)	Examinateur
Mr	M.	BOUFAIDA	Professeur (Univ. Constantine)	Examinateur

Abstract

A Mobile Ad Hoc Network (MANET) is a collection of mobile nodes that communicate through wireless links with a restricted bandwidth. Nodes move freely and link/node failures are common. This leads to numerous network partitions. The network topology may change rapidly and unpredictably. When the network partition occurs, mobile nodes in one partition are not able to access data hosted by nodes in other partitions. This may significantly degrade the performance of data access.

Replication is the key approach to increase data availability. It's a process of creation and relocation of data replicas. The replication techniques for wired environments or wireless environment with infrastructures are not well-suited to the dynamic environment defined by MANETs. In MANETs, replication achieves data availability by enabling access to the data through transient connections.

This work focuses on the issue of data replication on MANETs. We propose a new replication approach based on two phases. In the first phase, *preventive* replicas for new shared data are created at ***k*** hops distances. This is a preventive mean in case of unpredictable partitioning. In the second phase, replicas are dynamically relocated. To preserve data availability, it considers the user needs, the node neighbourhood and the access time.

We propose data replication methods based on this two phases approach. The first method creates the preventives replicas for each new shared data. To limit traffic overload of replication, the second method creates the preventives replicas only for the very important data. In these two methods the user needs are evaluated through the access frequencies rates. The third method considers the access time to optimize access performances. The last method is based on a logical topology of nodes groups. The simulation results indicate that these methods can increase the data accessibility with a moderate overload.

Key Words

Mobile Ad hoc NETwork (MANET), Data replication, Data accessibility, Data update.

Résumé

Un réseau mobile ad hoc (MANET) est une collection de nœuds mobiles qui communiquent à travers des liaisons sans fil à bande passante restreinte. Les nœuds se déplacent librement et les pannes de liaisons/nœuds sont fréquentes. Ceci produit de nombreuses partitions du réseau. La topologie du réseau peut changer rapidement et de manière imprévisible. Lorsqu'un partitionnement se produit, les nœuds mobiles sur une partition ne peuvent accéder aux données localisées sur les nœuds des autres partitions. Ceci peut dégrader les performances d'accès de manière significative.

La réplication est l'approche clé pour améliorer l'accessibilité aux données. C'est le processus de création et de relocalisation des copies de données. Les techniques de réplication pour les environnements câblés ou, sans fil avec infrastructures ne sont pas bien adaptées aux environnements dynamiques définis par les MANETs (Mobile Ad hoc NETworks). Sur les MANETs, la réplication offre une disponibilité des données en permettant l'accès à travers des connexions transitoires.

Ces travaux traitent du problème de réplication sur les MANETs. Nous proposons une nouvelle approche de réplication basée sur deux phases. Dans la première phase, des copies préventives de la nouvelle donnée partagée sont crées à des distances de k sauts. C'est un outil préventif dans le cas de partitionnements imprévisibles. Dans la seconde phase, les copies sont dynamiquement re-localisées. Pour préserver la disponibilité des données, elle considère les besoins de l'utilisateur, le voisinage d'un nœud et le temps d'accès.

Nous proposons des méthodes de réplication basées sur cette approche à deux phases. La première méthode crée les copies préventives pour chaque nouvelle donnée. Afin de limiter le surcoût en trafic de la réplication, la deuxième méthode crée les copies préventives uniquement pour les données jugées très importantes. Pour ces deux méthodes, les besoins d'un utilisateur sont évalués à travers les taux de fréquence d'accès. La troisième méthode considère le temps d'accès pour optimiser les performances d'accès. La dernière méthode est basée sur une topologie logique des groupes de nœuds. Les résultats de simulation indiquent que les méthodes peuvent apporter un gain en accessibilité à un coût acceptable.

Mots clés

Réseau mobile ad hoc (MANET: Mobile Ad hoc NETwork), Réplication de données, Accessibilité aux données, Mise à jour de données.

Remerciements

Je remercie Madame Aïcha MOKHTARI AÏSSANI, Professeur au département d'Informatique de l'Université des Sciences et de la Technologie Houari Boumédiene (USTHB), qui m'a fait l'honneur de présider ce jury.

Je remercie Madame Malika BOUKALA et Monsieur Mohamed AHMED NACER Professeurs au département d'Informatique de l'Université des Sciences et de la Technologie Houari Boumédiene (USTHB), de l'intérêt qu'ils ont manifesté pour cette thèse en acceptant de faire partie de ce jury.

Mes remerciements vont également à Monsieur Mahmoud BOUFAIDA, Professeur à l'Université de Constantine et à Monsieur Abdelmadjid BOUABDALLAH Professeur à l'Université de Compiègne (France), pour avoir accepté d'examiner ce travail.

Je voudrais exprimer ma profonde gratitude au Professeur Nadjib BADACHE, Directeur du Laboratoire LSI au département d'Informatique de l'Université des Sciences et de la Technologie Houari Boumédiene (USTHB), qui a dirigé mes travaux de recherche. Je lui serais toujours reconnaissante de m'avoir permis d'évoluer dans le domaine des systèmes distribués dès mes premiers pas dans la recherche. La justesse de ses conseils et sa patience, en période de doute, ont permis l'accomplissement de cette thèse.

Table des matières

Table des figures

Liste des tableaux

Introduction

L'évolution technologique des communications sans fil et des ordinateurs mobiles offre l'accès à l'information "any where and any time". Les réseaux mobiles ad hoc sont le support idéal de cette liberté vis-à-vis de la position et des équipements. Beaucoup de travaux de recherche se sont penchés sur la mise à disposition des utilisateurs des services nécessaires à un environnement informatique omniprésent, que ce soit pour le travail, les loisirs ou les tâches quotidiennes.

Un réseau mobile ad hoc est un ensemble de nœuds mobiles connectés par des liens radio sans fil. Ces environnements présentent l'avantage d'un déploiement rapide et peu coûteux puisqu'ils ne nécessitent l'installation d'aucune infrastructure. Les nœuds se déplacent librement et se connectent dynamiquement les uns aux autres. Les réseaux mobiles ad hoc sont utilisés dans de nombreux domaines (opération de secours, travail collaboratif, campagne scientifique, …). Ils doivent leur existence à une complète et incontournable auto organisation. En effet, leurs caractéristiques, dont principalement le manque d'infrastructure, exigent des nœuds de participer activement aux mécanismes de fonctionnement du réseau.

Si le service de routage de l'information a suscité le plus grand intérêt de la part de la communauté des chercheurs, la gestion des données mobiles a été pour un certain temps le parent pauvre de la recherche sur les réseaux mobiles ad hoc. Pour les systèmes "pervasive", le challenge était d'abord de faire communiquer les différents dispositifs. Mais, le besoin s'est vite fait ressentir avec le développement d'applications avancées sur ces environnements. Ainsi, de plus en plus de chercheurs tentent d'apporter leur contribution pour répondre aux attentes des utilisateurs mobiles en qualité de service, notamment pour l'accès à l'information.

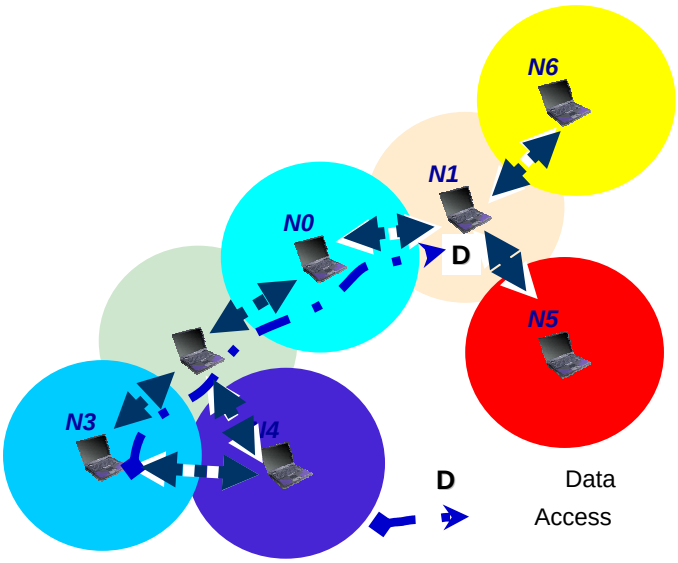

Figure I.1: Accès multi-sauts

Un nœud peut jouer le rôle de client et de fournisseur pour la majorité des services dont l'accès aux données partagées en mode "peer to peer" (figure I.1). Cependant, le mouvement

libre des nœuds d'un réseau mobile ad hoc fait que la topologie change dynamiquement et les déconnexions sont fréquentes. Le réseau peut se trouver subdivisé en partitions disjointes. Ce partitionnement limite les possibilités d'accès aux données (figure I.2) car les données d'une partition deviennent inatteignables par l'autre partition.

La réplication dans les réseaux mobiles ad hoc occupe une place importante à cause des besoins des utilisateurs en partage de données et en disponibilité alors que les ressources en mémoire sont limitées. Elle consiste à faire des copies de la donnée partagée et à les placer sur des nœuds dans le but de pallier aux déconnexions imprévisibles et de permettre l'accès aux données pour les nœuds malgré leur mobilité. La réplication est aussi un outil pour répartir la charge du ou des nœuds serveurs à capacités restreintes. D'autre part, l'accès aux copies les plus proches peut améliorer les temps d'accès et réduire le coût en trafic. En effet, en ajustant l'emplacement des copies en fonction de l'utilisation de la donnée, l'une d'entre elles a des chances de se trouver proche de son utilisateur. Une autre motivation est d'assurer la continuité des services à travers le pré chargement des données nécessaires à certaines tâches. Les informations de fonctionnement d'une application sont alors disponibles malgré les perturbations que peut subir la configuration du système.

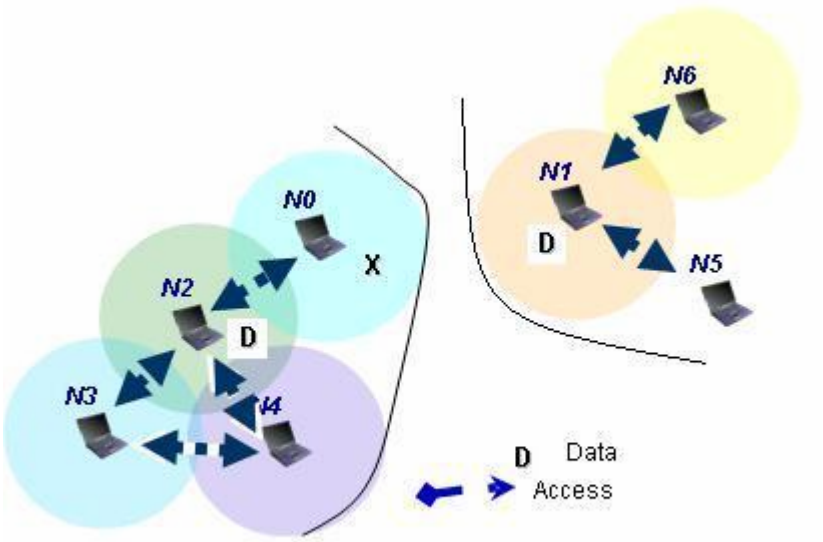

Figure I.2: Partitionnement du réseau et réplication

Le placement optimal de copies est un problème NP complet [Jamin 01] [Qiu 01] dans le cas d'une topologie statique. Pour les réseaux mobiles ad hoc, ce problème est d'autant plus complexe que la topologie est dynamique. La réplication pose donc de nouveaux problèmes très spécifiques aux réseaux mobiles ad hoc. Les restrictions en capacités mémoires des mobiles font que toutes les données ne peuvent être répliquées. Une stratégie judicieuse de sélection des données à répliquer doit être établie. Cette stratégie dépend souvent des particularités de l'environnement mobile ad hoc (type de mobilité, capacité de stockage, ..), du type d'application (temps réel, collaborative, ..), du type de donnée (structure, taille, degré d'importance, donnée urgente …) et des modes d'accès possibles (en lecture ou en écriture). Dans le cas d'application permettant la modification des données, une autre question se pose:

"Comment assurer le maintient de la consistance des copies?". L'existence de copies et la volonté de laisser la plus grande liberté aux utilisateurs dans la manipulation des données font que les copies initialement identiques finissent par diverger. La cohérence des copies est un autre problème crucial sur les réseaux mobiles.

Les protocoles de réplication traditionnels sont souvent basés sur des hypothèses non appropriées aux réseaux mobiles ad hoc. Ils supposent les copies toujours atteignables et les communications symétriques. Ceci n'est pas toujours le cas sur les environnements mobiles. Beaucoup de systèmes de réplication visant la tolérance aux pannes et l'amélioration des temps d'accès ont été élaborés pour les environnements de gestion de bases de données distribuées. Un certain nombre d'entre eux ont été étendus pour prendre en considération la mobilité des nœuds [Satyarayanan 02][Petersen 97]. Les spécificités des réseaux mobiles ad hoc exigent d'apporter des solutions nouvelles et rénovatrices.

Une stratégie de réplication doit définir:

- Les conditions de réplication de données: "Quelles données répliquer?" et "A quelle condition?"
- Le type et la granularité de la donnée à répliquer: "Quoi répliquer?"
- Les critères de placement des copies: "Comment choisir la localisation d'une nouvelle copie?"
- La gestion de l'accès et de la localisation des données : "Comment accéder à une donnée?"
- La gestion des opérations de mise à jour des données pour les données accédées en écriture : "Comment maintenir la cohérence des données malgré les mises à jours concurrentes sur même donnée?"

Cette thèse traite de la dissémination de données, du placement de copies et de la réplication d'une manière générale sur les réseaux mobiles ad hoc. Une étude des principaux travaux effectués ces dernières années sur la gestion des données nous a permis d'apporter une contribution par la proposition d'une nouvelle approche de réplication sur les réseaux mobiles ad hoc. Cette approche adopte une réplication préventive des données partagées jugées importantes. Une donnée partageable est répliquée en deux phases. Une première phase vise à disséminer uniformément des copies de la donnée. Ces premières copies sont réalisées dans un but de prévention des déconnexions, des partitionnements et des pannes du nœud source. La deuxième phase de réplication d'une donnée est une réplication adaptative qui réagit aux changements des besoins de l'utilisateur. Les besoins d'un utilisateur sont principalement représentés par le taux d'accès à chaque donnée.

Un certain nombre de méthodes ont été développées sur cette approche. Chaque nouvelle méthode tente de résoudre une contrainte de la méthode précédente à la recherche d'un meilleur compromis. Ces méthodes apportent des réponses aux quatre premières questions précédentes (i…iv) en considérant des données accédées uniquement en lecture. La dernière

question trouve une première réponse à travers la proposition d'un protocole de mise à jour de données qui ne traite que des opérations de mises à jour.

Divers évaluations de cette approche ont été menées sur un environnement de simulation "GloMoSim" [Bagrodia 99][UCLA]. Les résultats montrent l'intérêt de la proposition qui peut améliorer les performances d'accès aux données sur les réseaux mobiles ad hoc.

Le document est organisé en cinq principaux chapitres. Le premier chapitre présente les environnements mobiles et en particulier les environnements mobiles ad hoc. Il met en évidence les caractéristiques des communications sans fil et leurs conséquences sur le fonctionnement de ces réseaux. Les problèmes que posent ces environnements pour l'implantation de nouveaux services, sont alors clairement définis. Le deuxième chapitre nous donne quelques définitions et concepts de la réplication. Le troisième chapitre présente un état de l'art sur la gestion de l'accès aux données partagées et le placement des copies. Le quatrième chapitre propose une nouvelle approche pour l'amélioration de l'accessibilité, et un protocole de mise à jour de données sur les réseaux mobiles ad hoc. Le cinquième chapitre résume les principales mesures d'évaluation par simulation. Nous concluons ce document en rappelant notre contribution et en énumérant les perspectives envisageables.

Chapitre 1

Environnements Mobiles Ad Hoc

Introduction

Les avancées récentes en terme de performances des ordinateurs portables et des technologies de transmission sans fil, nous incitent à envisager une nouvelle approche de développement d'applications. Ces applications ne doivent plus considérer le matériel et les données comme étant liés à une localisation fixe mais comme des éléments attachés à un utilisateur qui lui-même peut être mobile.

Les connexions sans fil ont révolutionné l'utilisation de l'informatique. Le besoin de plus en plus grand des utilisateurs en indépendance vis-à-vis du matériel et de la position géographique, tout en bénéficiant d'une connexion permanente et d'une qualité de service comparable à celle obtenue sur les réseaux statiques, nécessite la mise en œuvre d'applications avancées des traitements mobiles. Ainsi, les environnements mobiles doivent être soutenus par des opérations robustes et efficaces qui tentent de remédier aux nouveaux problèmes posés [Badache 98].

Actuellement, la principale utilisation des réseaux sans fil se fait dans le cadre d'une architecture centralisée. En effet, les usagers se connectent à un point d'accès central qui leur fournit l'accès au réseau. La recherche actuelle tend néanmoins à proposer des solutions n'utilisant pas de point d'accès et se basant sur la collaboration des entités formant le réseau. Chaque nœud mobile doit alors assurer les fonctions de base du routage pour permettre la liaison entre nœuds qui ne sont pas à portée de communication. Ce type de réseau est appelé réseau mobile ad hoc et est supporté par le groupe MANET (Mobile Ad hoc NETwork) de l'IETF (Internet Engineering Task Force) [MANET]. De nombreux défis s'offrent à la recherche quand il s'agit de fournir sur les réseaux mobiles ad hoc, des fonctionnalités du niveau de celles disponibles sur les réseaux filaires.

Les réseaux mobiles ad hoc conviennent parfaitement aux applications caractérisées par une absence totale d'infrastructures, telles que les applications militaires, les applications stratégiques de secours (tremblement de terre, incendies, …) et les missions d'exploration [Hauspie 05]. Les applications collaboratives sont un autre type d'application de plus en plus demandées par les utilisateurs mobiles qui doivent coopérer à la réalisation d'une tâche commune sans contraintes de localisation. Cependant, cette coopération nécessite le partage d'un certain nombre d'objets dont principalement les données (fichiers, base de données, ..).

Ce chapitre a pour objectif de décrire les concepts relatifs aux environnements à communication sans fil et en particulier aux réseaux mobiles ad hoc.

1.1 *Caractéristiques des mobiles*

Un mobile est généralement une unité de calcul compacte et de petit poids tels que les ordinateurs portables, les assistants personnels PDAs (Personal Digital Assistant), les

ordinateurs de bord, …. Ces unités mobiles sont souvent caractérisées par des capacités de calcul et de stockage limitées. En effet, la réduction de leur taille et de leur poids n'est pas sans conséquence sur leurs capacités bien qu'elle soit la clé de leur mobilité. Leur autonomie énergétique est restreinte puisqu'ils sont dépendants d'une batterie qui doit être régulièrement rechargée. Plus les capacités d'un mobile sont meilleures plus sa taille et son poids doivent augmenter. Dans ce cas, la consommation en énergie augmente aussi.

Ces dernières années, on trouve des ordinateurs portables qui peuvent souvent être étendus avec un certain nombre de périphériques comme par exemple un disque dur ou une batterie supplémentaire. Ils sont munis d'une interface radio leur permettant de se connecter à un réseau filaire ou à un autre mobile à travers une architecture ad hoc. Mais, ces mobiles restent vulnérables devant les risques de vol, de perte ou de destruction d'où la nécessité de vigilance quand à la protection de leur contenu.

	Ressources				Encombrement		Autonomie
	Taille de l'écran (pouces) Résolution maximale (pixels) Nombre de couleurs	Processeur (MHz)	Mémoire (extensible à) (Mo)	Disque dur (Go)	Dimensions (L x l x e cm)	Poids (kg)	
Assistants personnels (*Personal Digital Assistants*)	3.6 - 3.8			-	15 x 10 x (1 - 2.5)	0.1 - 0.25	
↪ Organiseurs (*Organizers, Palmtops*)	160 x 160 16 (derniers modèles à 65536)	16 - 33	8 - 16 (64)				2 - 8 semaines
↪ *Pocket PC, Palm-size PC*	340 x 320 65536	131 - 400	16 - 64 (128)				6 - 15 h
Ordinateurs de poche (*Handheld Computers*)	6.5 - 10 640 x 480 65536	90 - 350	16 - 32 (96)	-	(18 - 25) x (10 - 20) x (2 - 5)	0.25 - 1	6 - 10 h
Ordinateurs portables (*Notebooks, Laptops*)	10 - 15.1 1400 x 1050 16 millions	600 - 2200	128 - 512 (1024)	10 - 60	(26 - 33) x (22 - 28) x (2 - 5)	1.5 - 4	1.5 - 3.5 h
Stations fixes	17 - 22 2048 x 1536 16 millions	800 - 3000	256 - 512 (2048)	20 - 80	-	-	-

Tableau 1.1: Caractéristiques physiques des terminaux [Le Mouël 03]

L'évolution de la technologie ne cesse de perfectionner ces mobiles qui sont devenus des composants à part entière de la vie actuelle. Par exemple, sur les ordinateurs portables de type NoteBooks et Laptops de DELL et IBM, les ressources en puissance de calcul (jusqu'à 2,2 Ghz) et en mémoire (jusqu'à 512Mo) sont comparables à celles d'un ordinateur fixe (tableau 1.1). Cependant, ces capacités avantageuses limitent l'autonomie (entre1h30mn et 3h30mn).

1.2 *Communications sans fil*

Les communications sans fil offrent aux utilisateurs le bénéfice d'une indépendance totale vis-à-vis du lieu où ils peuvent se trouver que ce soit pour leurs activités de loisirs ou

professionnelles. Ces communications sont basées sur des ondes hertziennes qui ont étés jusqu'au début des années 80 principalement utilisées pour les transmissions radio. L'avènement des systèmes GSM (Global System for Mobile communication) [Hild 95] [Scourias 96] pour les téléphones mobiles a permis aux utilisateurs de rentrer en communication avec n'importe qui à partir de n'importe quelle position. L'inconvénient des GSM est qu'ils sont dépendants d'installations lourdes pour assurer le lien entre les téléphones portables et le réseau téléphonique fixe. De nos jours des satellites sont aussi mis à contribution pour la couverture de régions très vastes afin de fournir certains services à une échelle internationale tel que le GPS (Global Position System).

Relativement aux communications filaires, les communications sans fil sont caractérisées par un débit faible (quelques Mbps selon les générations de réseaux), par un taux d'erreurs de transmission plus élevée et, par des déconnexions accidentelles (obstacles). Les communications sans fil possibles sont déterminées par l'entrelacement des portées de communication des mobiles entre eux ou avec des stations fixes. La figure 1.1 résume les caractéristiques principales des machines mobiles et des communications sans fil [Forman 94].

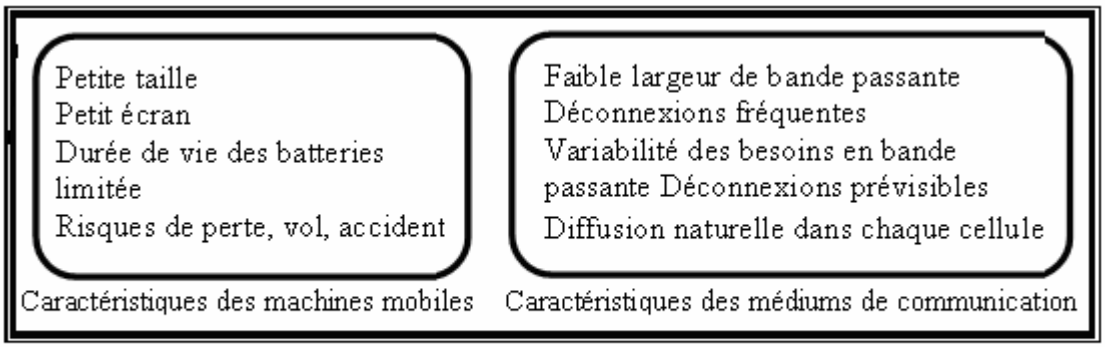

Figure 1.1 : Caractéristiques des environnements mobiles

1.3 *Modes de fonctionnement d'un mobile*

Si sur un réseau statique, une déconnexion représente une panne de communication, sur un réseau mobile à communication sans fil, une déconnexion est un phénomène fréquent qui ne correspond plus à une situation exceptionnelle mais qui représente un nouveau mode de fonctionnement. Une machine mobile est susceptible d'avoir l'un des modes de fonctionnement suivant (figure 1.2):

- **Mode veille**: Ce mode permet de restreindre la consommation en énergie d'un mobile. Les exécutions sont suspendues et la vitesse de l'horloge est réduite. Cependant, la liaison avec le réseau est maintenue afin de permettre la réception des messages.
- **Mode totalement connecté**: La machine mobile est reliée au réseau par un lien de communication normal sans fil ou filaire.

- **Mode déconnecté**: Dans ce mode un mobile est totalement déconnecté du réseau. Ce mode peut être involontaire à cause de l'état des liaisons ou même volontaire.
- **Mode partiellement déconnecté**: En mode partiellement déconnecté, la machine ne dispose que d'un lien de connexion faible à cause d'une mauvaise liaison hertzienne ou d'un faible niveau d'énergie.

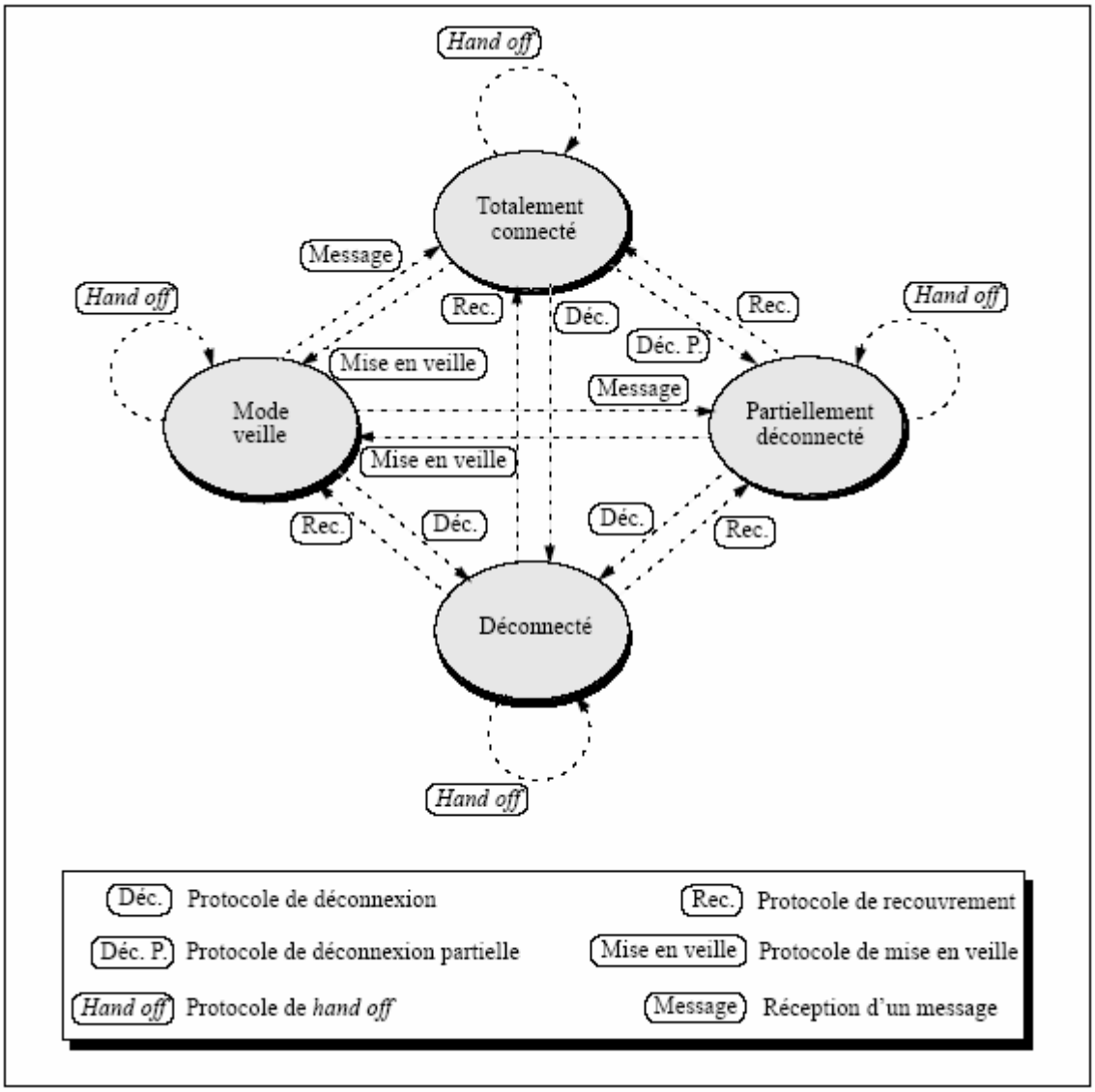

Figure 1.2: Les différents modes d'opérations sur les mobiles [Pitoura94][Baggio 95]

Le challenge des environnements mobiles est de permettre à un utilisateur mobile de poursuivre l'exécution de ses tâches indépendamment de l'état des liaisons. Les variations des états de ces liaisons deviendraient alors transparentes aux applications. Et, le concept d'opération déconnectée permet d'assurer une continuité de service en situation de déconnexion. Un avantage des réseaux mobiles est que la plupart des déconnexions peuvent être détectées au préalable pour entreprendre des actions spécifiques qui permettraient à la

machine de continuer ses exécutions après la déconnexion. Ces actions sont principalement des opérations de pré-chargement des données nécessaires à la poursuite des tâches de travail de l'utilisateur mobile. Un protocole de reconnexion doit être prévu pour assurer le recouvrement d'un état cohérent du système lors du rétablissement des liaisons. Le fonctionnement en mode déconnecté peut mener à des actions locales à un mobile qui sont incohérentes avec des actions concurrentes sur le reste du réseau [Pitoura 93] [Pitoura 94].

1.4 *Réseaux sans fil*

Les dispositifs mobiles peuvent bénéficier d'une connexion sans fil vers un autre dispositif mobile ou un site fixe quelconque. Cette connexion est temporaire puisqu'elle dépend de la position géographique d'un nœud mobile. Toutefois, ces connexions temporaires et imprévisibles constituent des réseaux dits sans fil. Sur ce type de réseaux, les utilisateurs peuvent bénéficier des services habituels indépendamment de leur position et de leur mobilité. Il suffit pour cela d'avoir une connexion. Parfois même sans connexion grâce à la mise en œuvre d'opérations en mode déconnecté. Ces réseaux permettent aussi la mise en réseau de nouveaux dispositifs à moindre coût. Les réseaux sans fil se caractérisent par l'architecture qu'ils adoptent et par la technologie réseau utilisée [Baggio 95].

1.4.1 Technologies des Réseaux sans fil

Les différentes technologies des réseaux sans fil sont:

- ✓ Les réseaux WPAN (*Wireless Personal Area Network*): Ces réseaux sont d'une taille de quelques dizaines de mètres. Ce type de réseau relie des périphériques ou des PDA à un ordinateur sans liaison filaire. Ils permettent aussi une liaison sans fil entre deux machines peu distantes. Deux technologies permettent de rendre cette communication sans fil possible :
  - **IrDA** (*Infrared Data Association*): Il s'agit d'une communication à infrarouge. Elle a une faible portée et nécessite un contact visuel et une liaison point à point (entre deux entités bien définies).
  - **Bluetooth :** Connu aussi sous le nom norme IEEE 802.15.1. C'est la principale technologie WPAN. Elle permet un débit théorique de 1Mb/s pour une portée de 30 mètres.
- ✓ Les réseaux WLAN (*Wireless Local Area Network*): Ces réseaux ont une portée de quelques centaines de mètres. Il existe plusieurs technologies pour ce type de réseaux, parmi lesquelles:

- **HyperLAN2** (High Performance Radio LAN 2.0): permet d'obtenir un débit théorique de 54Mb/s sur une zone d'une centaine de mètres.

- **WiFi**: Connu aussi sous le nom de IEEE 802.11. Cette technologie offre un débit de 11Mb/s et 54Mb/s selon la norme.

✓ Les réseaux WMAN (*Wireless Metropolitan Area Network*): sont basés sur la norme IEEE 802.16. Ils offrent un débit de 1 à 10Mb/s pour une portée de 4 à 10 kilomètres, ce qui les destine à la télécommunication.

✓ Les réseaux WWAN (*Wireless Wide Area Network*) : Connus aussi sous le nom de réseaux cellulaires mobiles, ce sont les réseaux sans fil les plus répandus. Les principales technologies sont :

- **GSM** (*Global System For Mobile Communication*): C'est une norme numérique européenne utilisant plusieurs bandes de fréquences notamment à 900 et 1800 MHz. Elle est idéale pour les communications vocales.

- **GPRS** (*General Pack Radio Service*): C'est une norme dérivée du GSM offrant un débit de données plus élevé.

- **UMTS** (*Universal Mobile Telecommunication System*): Elle offre un débit cinq à dix fois plus élevé que GPRS.

Les réseaux sans fil peuvent être classés en deux catégories : les réseaux avec infrastructure et les réseaux sans infrastructure.

1.4.2 *Architecture des réseaux sans fil*

On distingue deux types d'architectures: les réseaux avec infrastructure et les réseaux mobiles ad hoc sans aucune infrastructure.

Les réseaux mobiles avec infrastructure ou réseaux cellulaires sont constitués de deux ensembles d'entités distinctes : les sites fixes et les sites mobiles Les sites fixes du réseau filaire sont appelés "stations de base". Ils communiquent avec les sites mobiles situés dans une zone géographique limitée (cellule) et ce via une interface de communication sans fil. Les sites fixes sont reliés entre eux par une connexion filaire (voir figure 1.3). A un moment donné, une unité mobile n'est connectée qu'à une seule station de base, la communication avec les autres sites se fait à travers celle-ci [Badache 98].

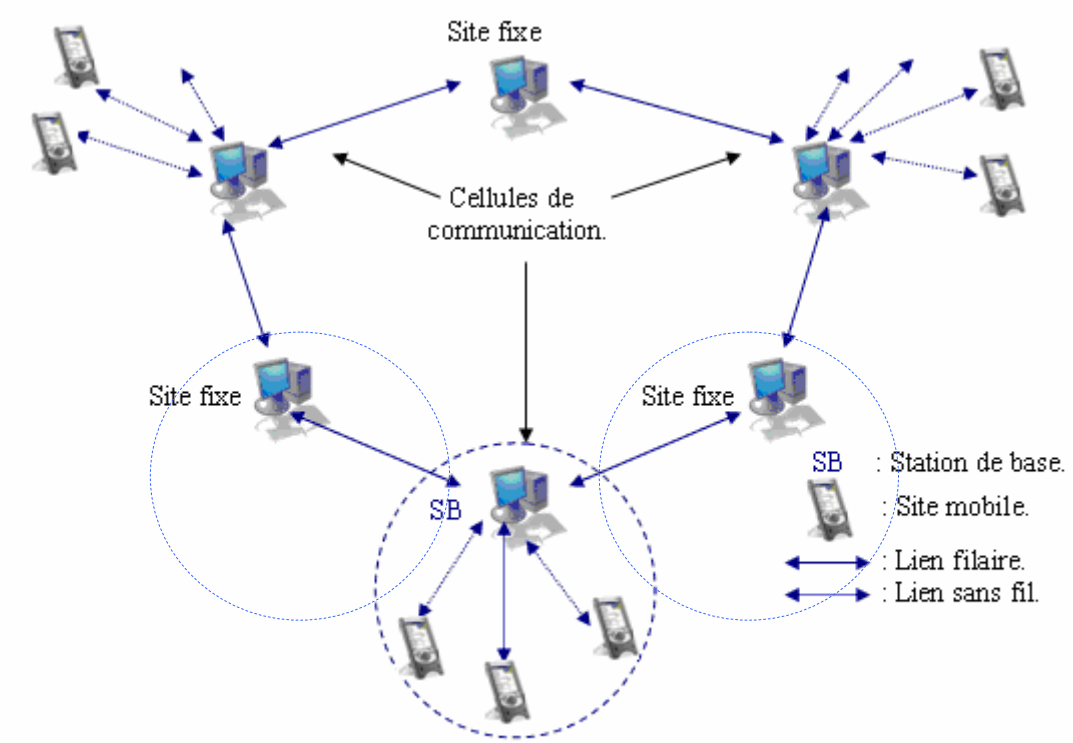

Figure 1.3: Réseau mobile cellulaire

En se déplaçant, une unité mobile risque de s'éloigner de la station de base de sa cellule. Afin de poursuivre ses communications, l'unité doit faire appel à une nouvelle station de base. L'unité mobile scrute alors les signaux reçus des stations fixes. Le signal le plus fort correspond alors à la station de base la plus appropriée pour elle. Lorsqu'en se déplaçant, une unité mobile quitte une cellule pour passer dans une autre, les deux stations de base des deux cellules doivent assurer la transition. Cette transition est appelée *transfert intercellulaire* ou *Handoff* [Badache 98][Baggio 95].

Les réseaux mobiles ad hoc sont des réseaux multi sauts composés d'hôtes autonomes sans fil. Chaque hôte peut servir de routeur pour assister le trafic provenant des autres nœuds (figure 1.4). Il y a absence totale d'infrastructure. Ils se forment au hasard des positions et des portées de communication de chaque élément. Ils sont très utilisés pour les secours en cas de catastrophe, conférences, hôpitaux, campus, champs de bataille …

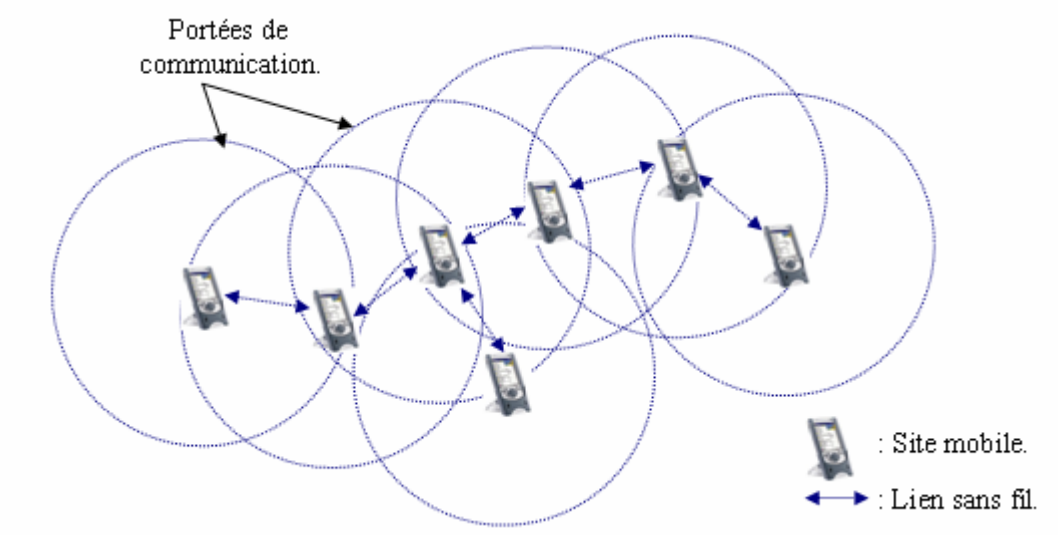

Figure 1.4 : Échanges en mode mobile ad hoc.

1.5 *Les réseaux mobiles ad hoc (MANETs)*

1.5.1 Modélisation

Un réseau Ad Hoc peut être modélisé par un graphe $G_t = (V_t, E_t)$ où :

- V_t représente l'ensemble des nœuds (i.e. les unités ou les hôtes mobiles) du réseau.
- E_t modélise l'ensemble des connexions qui existent entre ces nœuds.

Une connexion e entre deux nœuds u et v est représentée par $(u, v) = e$ (figure 1.5).

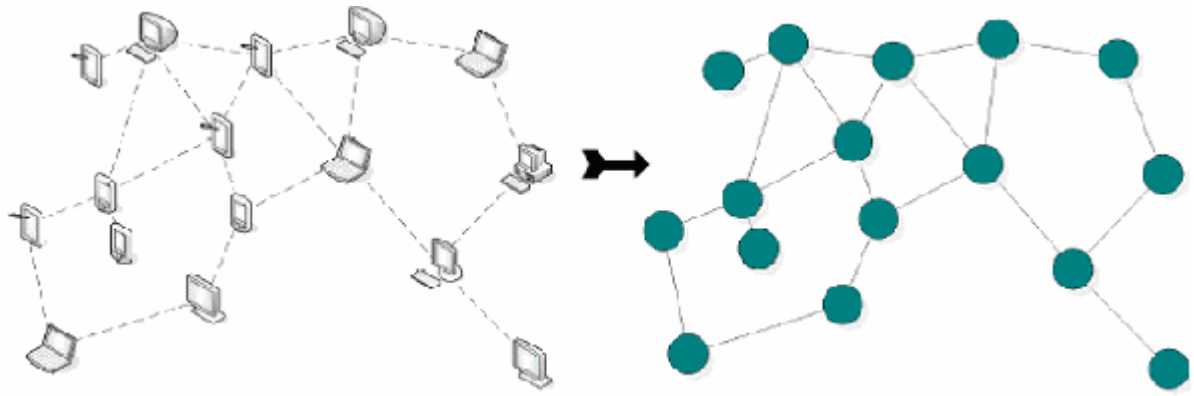

Figure 1.5 : Modélisation d'un réseau mobile Ad Hoc

1.5.2 Caractéristiques des réseaux mobiles Ad Hoc

Les nœuds des réseaux mobiles ad hoc sont équipés d'émetteurs et de récepteurs sans fil utilisant des antennes qui peuvent être omnidirectionnelles (broadcast), fortement directionnelles (point à point), probablement orientables, ou une combinaison de tout ça. A un

instant donné, en fonction de la position des nœuds, de la configuration de leur émetteur-récepteur, des niveaux de puissance de transmission et d'interférence entre les canaux, il y a une connectivité sans fil qui existe entre les nœuds, sous forme d'un réseau ad hoc.

Dans les réseaux à infrastructures, un nœud mobile joue simplement le rôle d'émetteur récepteur. Les infrastructures fixes assurent les fonctionnalités de gestion du réseau. Un réseau mobile ad hoc ne bénéficie d'aucune d'infrastructure. Ces fonctionnalités ne peuvent donc être assurées que par les noeuds eux-mêmes. Ces réseaux ont les caractéristiques spécifiques suivantes [Carson 99]:

1) Auto gestion du réseau:

Un noeud établit des liens radio dynamiques avec ses voisins. Il doit assurer la gestion de ses propres ressources radioélectriques. Il doit aussi se charger de l'acheminement des paquets de messages en provenance des noeuds voisins (le routage). Nous pouvons retrouver toutes les fonctionnalités présentes dans un réseau mobile grâce à la forte implication des noeuds dans les fonctions d'administration et de gestion du réseau. Cependant, un noeud possède des ressources limitées et sa connaissance du réseau n'est que partielle et temporaire.
Un nœud participe aussi à la création, la modification et même à la destruction du réseau par sa présence, ses déplacements et son absence (panne de batterie par exemple).

Le fait que le contrôle des réseaux mobiles ad hoc soit décentralisé est un avantage qui améliore sa robustesse, contrairement aux problèmes pouvant survenir sur les points centraux dans des environnements plus centralisés. En plus, ces réseaux se prêtent bien à une extension en réseaux à grande échelle. Les mécanismes de bases nécessaires à ce passage à l'échelle sont offerts par ces réseaux (déploiement rapide, autocontrôle, services en mode "peer to peer", ..)

2) Topologies dynamiques:

Les nœuds sont libres de se déplacer arbitrairement, ce qui fait que la topologie du réseau typiquement multi-sauts, peut changer aléatoirement et rapidement n'importe quand (figure 1.6). Elle peut être constituée à la fois de liaisons unidirectionnelles et bidirectionnelles.

3) Topologie évolutive:

Pour ajouter un noeud à un réseau ad hoc préexistant, il suffit de rapprocher le nouveau composant d'au moins l'un des membres du réseau. De même, il suffit de l'éloigner pour le détacher du réseau.

4) Liaisons à débits variables et à bande passante limitée:

Les liaisons sans fil auront toujours une capacité inférieure à leurs homologues câblés. En plus, le débit réel des communications sans fil - après avoir déduit les effets des accès multiples, du bruit, des interférences, etc. - est souvent inférieur aux taux de transfert maximum de la radio.

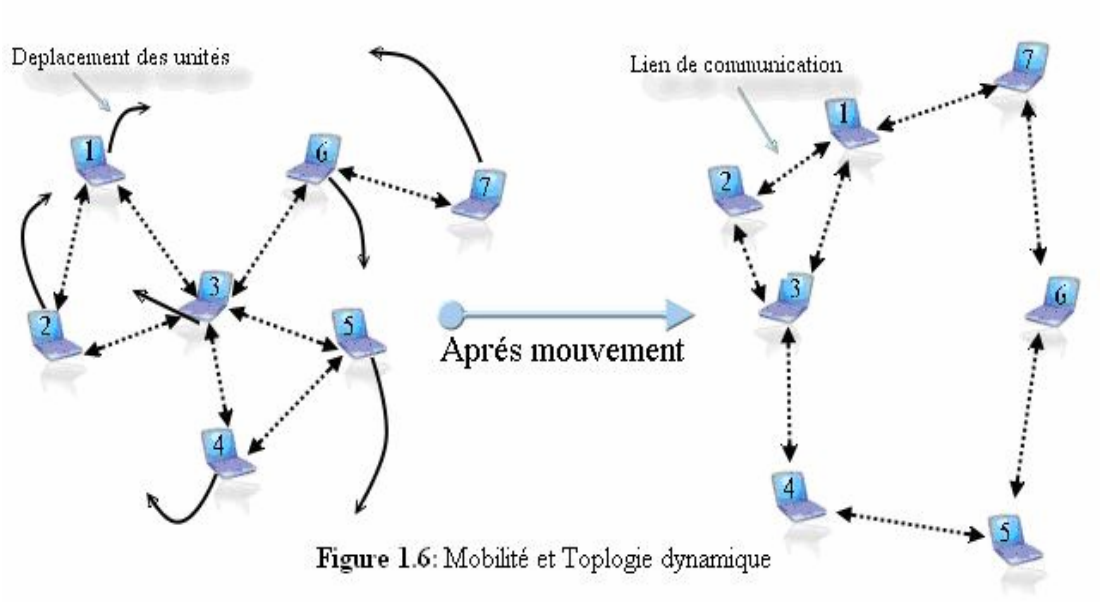

Figure 1.6: Mobilité et Toplogie dynamique

Un des effets de ces débits de liaison relativement faibles est la congestion fréquente. La demande de service de routage des applications distribuées approchera ou dépassera souvent la capacité du réseau. Les utilisateurs mobiles ont tendance à solliciter les mêmes services que sur les réseaux statiques Cette demande ne cesse de croître avec l'augmentation des traitements multi médias et des applications basées sur les réseaux.

5) Utilisation limitée de l'énergie:

Une partie des nœuds d'un MANET, voire l'ensemble des nœuds, peut reposer sur des batteries ou un autre moyen limité pour puiser leur énergie. Pour ces nœuds, le plus important est sans doute de mettre en place des critères d'optimisation pour la conservation de l'énergie.

6) Sécurité physique limitée:

Les réseaux sans fil mobiles sont généralement plus sensibles aux menaces physiques que ne le sont les réseaux câblés fixes. Les possibilités accrues d'attaques par écoute passive, par usurpation d'identité et par déni de service doivent être étudiées avec attention. Les techniques existantes pour la sécurité des liaisons sont souvent appliquées au sein des réseaux sans fil pour réduire les risques d'attaques [Michiardi 03].

1.5.3 Les applications des réseaux mobiles Ad Hoc

Les réseaux mobiles ad hoc sont idéaux pour les applications caractérisées par une absence (ou la non fiabilité) d'une infrastructure préexistante, soit parce que son installation s'avère trop chère ou que son déploiement soit impossible dans une région donnée [Frodigh 00] [Djenouri 05]. Parmi ces situations nous trouvons:

- ✓ Applications militaires: Un réseau mobile ad hoc est la solution idéale pour maintenir la liaison entre les différents constituants d'une armée (avions de chasse, chars d'assauts, militaires, …).

- ✓ Missions d'exploration : Des capteurs (sensors) placés dans une région inaccessible peuvent être reliés par un réseau mobile ad hoc pour faire une collecte et éventuellement un échange d'information.

- ✓ Opérations de secours : Il est impossible d'installer des infrastructures fixes dans une région touchée par une catastrophe naturelle (cyclone, tremblement de terre, …) d'où la nécessité de déployer un réseau mobile ad hoc.

- ✓ Conférences : Un réseau mobile ad hoc peut être utilisé par un groupe d'étudiants pour partager des informations dans une salle de lecture par exemple.

- ✓ Utilisation privée : Ce type de réseaux peut facilement être utilisé à l'intérieur d'une maison (réseau appelé : "home network"), où un ensemble de robots ou bien d'équipements (audio/vidéo, alarme…) peuvent s'échanger des informations.

1.5.4 Le routage dans les réseaux mobiles Ad Hoc

Les techniques de routage déjà utilisées se basent sur l'utilisation de routeurs avec des tables de routage souvent statiques. Et même dans le cas où le routage est dynamique, le routeur quant à lui reste statique d'où la nécessité d'utiliser de nouvelles techniques dans le cas des réseaux mobiles ad hoc.
Parmi les techniques de routage existantes, on distingue trois grandes classes (figure 1.7) [Badache 02] [Lemlouma 00]:

- **Les protocoles pro-actifs:** très sensibles aux changements de topologie; ils remettent constamment les tables de routage à jour. Les chemins sont déterminés préalablement aux demandes. Bien que le chemin soit vite trouvé, les ressources sont surexploitées à cause du nombre important des messages de contrôle. OLSR

(Optimized Link State Routing) est un exemple de ce type de protocoles de routage.

- **Les protocoles réactifs**: L'établissement d'un chemin ne se fait qu'à la suite d'une demande. Les ressources ne sont pas sur utilisées mais les délais de communication sont rallongés. Ces retards dans l'établissement de chemins peuvent être la cause de congestion. AODV (Ad hoc On Demand Distance Vector) est un exemple de ce type de protocoles de routage.

- **Les protocoles hybrides**: Ces protocoles sont basés sur les deux protocoles précédents selon certaines conditions. Ils peuvent adopter un fonctionnement proactif local et un fonctionnement réactif pour les nœuds éloignés. Par exemple, ils peuvent utiliser le premier type de protocole pour les noeuds distants de *n* sauts au plus et le second type de protocole pour les autres (Avec n>0). Le protocole ZRP (Zone Routing Protocol) est un protocole hybride.

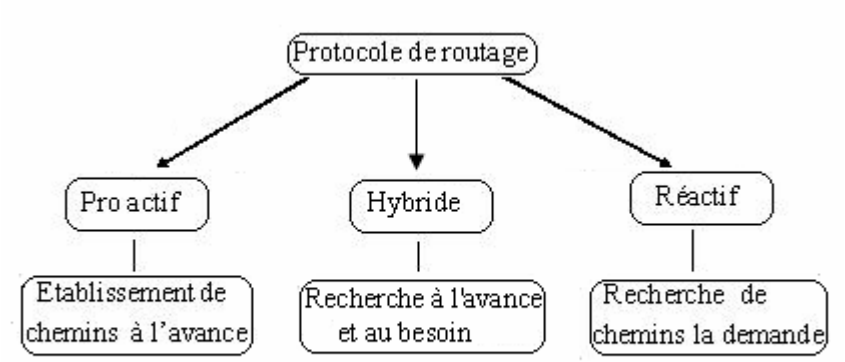

Figure 1.7 : Classification des protocoles de routage

1.5.5 Gestion des ressources locales

Les ordinateurs mobiles dépendent pour leur fonctionnement, de leur batterie, souvent de faible autonomie et dont le rechargement n'est pas toujours possible dans les périodes de mobilité. De même, leur espace de stockage devient vite insuffisant du fait du nombre sans cesse croissant de données exploitées par les utilisateurs. Les fonctions de communication et d'accès aux données sont celles qui contribuent majoritairement à l'épuisement de ces ressources critiques.

Les communications sans fil nécessitent une plus grande consommation d'énergie comparée aux communications câblées. Or, les utilisateurs mobiles communiquent essentiellement via le réseau sans fil. Aussi, la consommation d'énergie générée par la communication dépend du type du réseau sans fil. Dans les réseaux à base d'infrastructure, seuls les nœuds mobiles qui émettent ou reçoivent les messages sont concernés par les dépenses d'énergie. En revanche, dans les réseaux en mode mobile ad hoc, tous les mobiles

sont affectés, à des degrés divers, par les dépenses d'énergie. Ceci est dû au fait que les interfaces réseaux de tous les mobiles sont à l'écoute du support de transmission. Dès qu'un message est émis, tous les terminaux de la portée de communication directe le reçoivent, mais seul le noeud destinataire du message le valide, tandis que les autres noeuds l'ignorent. Ils existent divers travaux sur l'optimisation de la consommation en énergie sur les réseaux mobiles ad hoc.

En ce qui concerne la gestion de l'accès aux données, les ressources affectées sont l'espace de stockage ainsi que l'énergie. Il est donc nécessaire d'adapter la gestion de l'accès aux données en fonction de la disponibilité de ces ressources, tout en réduisant la consommation des accès en ressources mémoire et énergétiques. Un grand effort de recherche est déployé ces dernières années pour la gestion de l'accès aux données en fonction de la disponibilité des ressources tout en minimisant leur consommation. La majorité des mécanismes sont proposés au niveau middleware avec parfois une coopération avec des niveaux plus bas du routage.

CONCLUSION

Ce chapitre nous a permis de mettre en évidence les composants de base des réseaux sans fil et de comparer les architectures disponibles. En particulier, les réseaux mobiles en mode ad hoc sont d'un grand intérêt. Ce sont des réseaux totalement autonomes, à topologie fortement dynamique, ayant pour avantages un déploiement rapide à un coût réduit. Ils offrent une robustesse de par leur conception qui se veut évolutive et adaptée intrinsèquement à la mobilité. Dans ce chapitre, nous avons présenté les réseaux mobiles ad hoc, leurs caractéristiques et leurs domaines d'applications.

Les réseaux mobiles ad hoc sont la solution à un certain nombre de problèmes. Cependant, leur mise en oeuvre soulève de nombreuses questions, notamment sur le maintien d'une qualité de service acceptable avec peu de ressources et des liens imprévisibles.

Chapitre 2

La réplication de données

Introduction

Grâce à leur mobilité et leur déploiement rapide, les réseaux mobiles ad hoc sont très intéressants pour un certain nombre d'activités, comme, les opérations militaires, les interventions de secours, le travail collaboratif, …. Cependant, le mouvement imprévisible des nœuds et les fréquentes déconnexions peuvent diviser le réseau en partitions complètement disjointes sans aucun lien de communication.

La gestion des données sur les réseaux mobiles ad hoc est un axe de recherche en plein essor vu la demande croissante en accessibilité et en disponibilité de l'information. Le défi consiste à fournir aux noeuds mobiles une continuité de service et d'accès malgré les contraintes de ces environnements. La réplication est alors l'approche clé pour apporter des solutions dans ce cas. Qu'il s'agisse de données (fichiers, bases de données, ..) ou de programmes, la réplication est l'unique outil possible face aux partitionnements.

Sur les réseaux mobiles ad hoc la préoccupation majeure des utilisateurs est l'accessibilité aux données. Bien que l'absence d'infrastructure statique fait qu'elle soit moins évidente à assurer. La réplication d'une donnée est la création et le placement en mémoire d'une ou de plusieurs copies sur des nœuds du réseau. Le placement doit se faire de manière judicieuse afin de satisfaire au mieux les objectifs d'accessibilité (disponibilité, temps de réponse, …). Sur les réseaux statiques, la réplication est principalement utilisée dans la tolérance aux pannes. Elle représente aussi un moyen de prévention de la surcharge d'un serveur unique et de la congestion des liaisons de communication [Helal 86].

L'accessibilité est le fait de permettre à un utilisateur d'atteindre une donnée quelque soit sa position et ses déplacements. Une deuxième préoccupation du placement et de la distribution des copies sur un réseau est d'offrir de bonnes performances en temps d'accès et en trafic. Un utilisateur n'est pas toujours disposé à attendre des délais importants même s'il est assuré d'accéder à la donnée. La largeur de la bande passante étant limitée, un surcoût important en trafic peut dégrader les capacités énergétiques des nœuds mobiles et provoquer un écroulement du système. Les données sont répliquées sur ou près des nœuds qui en ont le plus besoin. Ce qui leur confère plus d'autonomie et réduit les connexions parcourues entre le nœud demandeur d'accès et la donnée.

Dans les réseaux à infrastructures ou filaires, les techniques de réplication proposées utilisent généralement une approche centralisée. Elles se basent sur un certain nombre de serveurs fixes de données [Barbara94] [Wolfson97] [Jing97] [Hauspie05] [Saito05] [Kouici06]. Les clients sont informés de la localisation des serveurs, qu'ils peuvent interroger pour la recherche d'informations ou de services. De telles approches sont inadéquates pour les réseaux mobiles ad hoc, à cause de l'absence totale de toute infrastructure centralisée. Certains travaux ont adapté ces solutions en désignant des noeuds mobiles comme nœuds assurant un rôle de contrôle central. Ces adaptations sont souvent non satisfaisantes à cause du manque de sûreté de fonctionnement d'un nœud mobile.

Ces dernières années un certain nombre de travaux ont été menés sur la réplication pour les réseaux mobiles ad hoc. Des méthodes utilisant des approches différentes ont été proposées [Padmanabhan 07][Padmanabhan 06]. Ces solutions se distinguent principalement par le type de données pris en compte et les opérations de manipulation considérée (données accédées en lecture seule, ou en écriture, …). Les opérations d'écriture concurrentes sur des copies d'une même donnée font, que les copies divergent de plus en plus à chaque nouvelle opération. Dans ce cas, des systèmes de maintien de la cohérence et de la consistance doivent être mis en œuvre [Saito 05][Barreto 06].

Ce chapitre présente quelques définitions, notions et concepts généraux de la réplication en environnement mobile ad hoc. Une étude des principales solutions pour la réplication est exposée au chapitre suivant.

2.1 *Définitions et Concepts de la réplication de données*

Copie de donnée:

Une copie est un double ou une reproduction d'une donnée source. La reproduction nous donne deux exemplaires identiques d'une donnée. Le tout premier exemplaire d'une donnée sur le réseau est souvent désigné par copie *originale*. Les autres copies sont dites *secondaires*. Généralement, une copie *originale* ne doit être supprimée en aucun cas; afin d'assurer toujours l'existence d'au moins un exemplaire.

Une copie *de travail* est une copie de donnée stockée localement en fonction des accès effectifs du nœud hôte. Il s'agit donc d'une donnée utilisée par le nœud et pour laquelle il manifeste un besoin dans l'accomplissement de ses tâches.

Une copie *préventive* est une copie qui a été stockée localement à un nœud par anticipation de l'indisponibilité de la donnée qui serait provoquée par les déconnexions ou la disparition du nœud source. Une copie préventive doit être créée à des moments opportuns sur des nœuds mobiles judicieusement choisis.

Serveur de donnée:

Un serveur d'une donnée est un nœud hôte d'une copie *secondaire* ou *originale* de la donnée. Il reçoit des requêtes d'accès à la donnée et accepte de les traiter. Le serveur hôte de la copie *originale* de la donnée est souvent désigné comme serveur *primaire*. Les autres serveurs de la donnée sont des serveurs *secondaires*. Le serveur *primaire* est souvent désigné pour assurer des tâches de contrôle centralisé notamment dans le cas de synchronisation de copies divergentes.

La réplication:

La réplication consiste en la création et le placement en mémoire de copies d'une donnée (ou objet en général) sur les noeuds. L'ensemble des copies représente un seul objet logique qui est la donnée initiale. La stratégie de placement et le choix des nœuds hôtes doivent viser une amélioration des performances des accès ultérieurs en accessibilité et en temps de réponse aux requêtes d'accès. Des mécanismes de localisation accompagnent souvent un système de réplication. Avec l'instauration d'un système de réplication des données, les procédures de recherche devraient être optimisées en temps et en trafic. La réplication de données sur les réseaux mobiles ad hoc peut avoir la vocation d'économiser l'énergie et la consommation de la bande passante.

Gestion de cache:

Deux techniques très proches visent l'amélioration du traitement des requêtes d'accès aux données sur les réseaux mobiles ad hoc: la réplication et la gestion de cache [Perrich 05]. Ces techniques sont si proches que souvent elles ont tendance à être confondues. Un cache permet de stocker temporairement une donnée accédée pour éviter le coût des requêtes distantes. La présence d'un système de cache assure une meilleure efficacité. La principale différence entre un système de cache et la réplication est que l'utilisateur peut contrôler le nombre de copies des données à répliquer, alors qu'il ignore tout d cache. Un cache joue le rôle d'intermédiaire entre les requêtes de l'utilisateur et les données. Le but d'un système de cache est d'augmenter tant que possible, le rapport: nombre de requêtes qui accèdent au cache sur le nombre de requêtes hors cache.

La continuité de service:

La réplication permet à un nœud qui se retrouve déconnecté, de détenir localement les ressources auxquelles il accède le plus. Il a donc plus la possibilité de continuer à travailler normalement une fois isolé. Les services qu'il utilise sont alimentés en données répliquées préalablement à la déconnexion. La prédiction de déconnexions permet à un système de sauvegarder les données nécessaires à la continuité des services fournis à l'utilisateur. Une fois déconnecté, un nœud peut continuer à avoir certains services grâce au pré chargement des données distantes.

La redondance:

La redondance peut être définie comme étant le nombre de copies d'une même donnée dans un voisinage proche. On peut aussi la définir par le fait qui si *m* copies sont suffisantes à assurer une certaine qualité de service requise, la présence de *n* copies avec *n* supérieur à *m*, implique que *n*-m copies ont été réalisées inutilement. Il y a donc redondance et l'espace occupé par les copies supplémentaires aurait pu servir au stockage d'autres données à

performances moindres. Bien que la redondance ait souvent une intonation négative nous définissons deux types de redondance.

Nous qualifions la redondance d'une copie qui accapare sans un intérêt global un espace mémoire qui serait utile à d'autres données, par redondance *inutile*. Nous définissons aussi la notion de redondance *utile* comme étant une redondance dont l'apport est nécessaire au maintien des performances d'accès sollicitées. Par exemple, si deux nœuds très proches utilisent chacun une même donnée de manière très importante par rapport aux autres données, il est plus utile de laisser une copie sur chaque nœud. Car, la suppression d'une copie obligera le nœud correspondant à demander l'accès au nœud qui détient encore la donnée. Ce dernier nœud devra fournir un effort supplémentaire en calcul et en énergie. Et, la déconnexion du nœud client impliquera un taux important d'échec d'accès puisque la donnée est très utilisée. Dans ce cas, la redondance est acceptable et nous la qualifions de redondance *utile*.

Le degré de réplication:

Le degré de réplication exprime le niveau de présence de copies répliquées sur le réseau. Ce degré peut être défini par un taux ou par le nombre de copies existantes pour une donnée. Certains systèmes de réplication définissent le degré de réplication requis et, ils veillent alors à maintenir ce degré sur le réseau, sur une partition, sur une région du réseau, ou sur un sous ensemble des nœuds du réseau.

La cohérence:

La cohérence est une relation qui définit le degré de similitude entre les copies d'une donnée répliquée. Dans le cas idéal, cette relation caractérise des copies qui ont des contenus identiques. Dans les cas réels, où les copies évoluent de manière parallèle, la cohérence définit les limites de divergence autorisées entre copies.

La relation de cohérence est assurée par des mécanismes de synchronisation de copies.

2.2 *Intérêts et objectifs de la réplication de données*

Sur les réseaux mobiles ad hoc, la réplication conserve non seulement tous les avantages acquis sur les réseaux statiques et mobiles mais aussi, elle a de nouveaux intérêts irréfutables pour ces réseaux. Le fonctionnement global de ces environnements et l'intégration de services avancés passent obligatoirement par l'incorporation de procédures de réplication. On peut résumer les intérêts et les objectifs de la réplication en les points suivants (figure 2.1):

- **Accessibilité**:

  Dans le cas où une donnée risque d'être non disponible ou inatteignable par une partie des nœuds du réseau, la présence de plusieurs copies de la donnée est le meilleur

moyen de permettre aux nœuds qui la recherche de la découvrir. La disponibilité d'une donnée pour un nœud est exprimée par le fait qu'un nœud puisse atteindre au moins une quelconque des multiples copies. Le perpétuel mouvement des nœuds, les déconnexions et le partitionnement font de la réplication un allié essentiel des environnements mobiles ad hoc.

- **Performances d'accès**:

Deux principaux paramètres représentent les performances d'accès aux données: le temps de réponse et le trafic conséquent. Avec l'instauration de plusieurs copies d'une même donnée, lors de l'accès a une donnée, un nœud recherchera une copie locale ou une copie sur le proche voisinage. Les messages générés par la recherche de la donnée et par les requêtes d'accès engendrent alors un trafic réduit si ce n'est optimal. D'autre part, l'accès à une copie proche sur des liaisons non encombrées permet d'obtenir des temps de réponse satisfaisants.

- **Fiabilité:**

La réplication permet aux applications de se prémunir contre les problèmes de pannes subites volontaires ou involontaires et, contre le partitionnement. L'allocation de plusieurs copies d'une donnée offre une plus grande fiabilité aux applications.

- **Répartition de charge**:

La présence d'un serveur unique, celui de la copie *originale*, peut mener vers une surcharge du serveur et de ses voies de communications. Cette congestion peut être une cause de temps de réponse de plus en plus long jusqu'à écroulement du système. En se partageant la réception et le traitement des requêtes, des serveurs multiples distribués sur le réseau de manière appropriée évitent ce problème de surcharge.

2.3 *Inconvénients de la réplication*

De part sa définition, un réseau mobile ad hoc requiert une approche de réplication complètement différente de celle déjà développée pour des réseaux plus conventionnels. En effet, sur ces systèmes, certains inconvénients des méthodes connues sont retrouvés avec une ampleur souvent plus importante. Quoique la réplication soit avant tout un outil pour résoudre des problèmes divers sur les réseaux mobiles ad hoc, elle présente les complications suivantes (figure 2.1):

- **Divergence des copies:**

La réplication d'une donnée sur plusieurs machines peut faire évoluer les copies d'une manière parallèle et indépendante. Ces copies finissent par avoir des contenus qui sont

dans des états différents et incohérents. L'intégration d'un algorithme de convergence est alors indispensable [Défago 98][Qi Lu 96][Pitoura 95][Wiederhold 90].

- Surcoût en communication:

Les opérations de re-localisation de copies de données et l'évaluation de certains paramètres (connectivité, degré de réplication, …) de l'algorithme de réplication produisent un surcoût en trafic, en traitement, et en consommation énergétique. Sur un réseau mobile ad hoc, cet "overload" peut vite devenir important car l'algorithme de réplication doit s'adapter en permanence aux changements dynamiques de la topologie. Cet effort d'adaptation peut entraîner un surcoût considérable. La conception de toute solution de réplication doit veiller à modérer ce surcoût.

- Complexité du système:

L'intégration de la réplication nécessite souvent l'observation de l'état global ou local de l'environnement de déploiement de sa stratégie. L'incorporation de ces contrôles fait que le système obtenu est plus complexe. Par exemple, lors d'un accès, il ne suffit pas de consulter la mémoire locale mais il faut maintenir des structures de localisation et de description des données susceptibles d'être utilisées et qui se trouvent sur le réseau. Ces structures doivent continuellement êtres mises à jours selon les changements de comportements des utilisateurs et de la configuration. Car, les informations peuvent devenir très vite non valides.

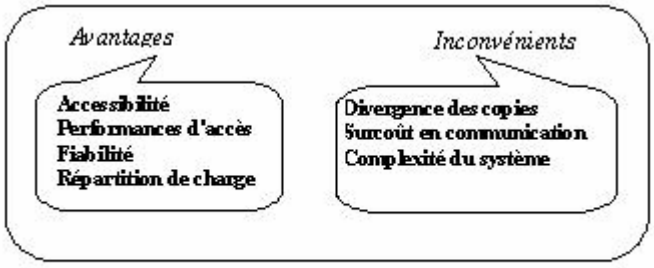

Figure 2.1: Avantages et inconvénients de la réplication

2.4 *Critères de base de la réplication*

Il n'existe pas beaucoup de travaux sur le placement optimal de copies sur un réseau mobile ad hoc. Sans doute parce que le problème est complexe. Cette complexité est due à la variété des critères de réplication (besoin de créer une copie) et de placement. La majorité des travaux sur la réplication sont basés sur un critère ou un sous ensemble de critères tels que [Bellavista 05c][Gossa 05][Cao 04b][Hara 00]:

- la fréquence d'accès à la donnée,
- la taille des données,
- le nombre de serveurs à maintenir,

- la distance entre serveurs et clients ou entre serveurs eux même,
- la sémantique des données,
- l'état des connexions du nœud client ou de celles du serveur de données.

Les critères d'une technique de réplication sont les conditions qui nous permettent de répondre aux questions suivantes:

✓ A quel instant doit t-on lancer une réplication de données (période, seuil, délais, état de la topologie, …)?

Certains systèmes de réplication définissent une période d'allocation de copies. Cette période est souvent fonction de la mobilité du réseau. D'autres observent l'état de la topologie pour juger de l'opportunité de lancer une réplication.
La réponse à cette question est loin d'être évidente, car une re-localisation des copies doit satisfaire les besoins d'utilisateurs mobiles accédant à des données elle mêmes mobiles. La détermination d'un instant de réplication approprié semble laborieuse à cause de ce dynamisme à facettes multiples.

✓ Comment déterminer les données à répliquer (connectivité, fréquence d'accès, nombre de serveurs, taille, temps d'accès, sémantique, .. etc.)?

Les données candidates à une réplication sont sélectionnées selon divers critères. La fréquence d'accès est l'un des critères les plus communément utilisés car elle permet de prédire les besoins futurs d'un utilisateur. Ces besoins peuvent aussi être définis à travers des profils d'utilisateurs définis au préalable. Il existe des systèmes où il est possible de déterminer des relations sémantiques entre les données. Ainsi, si un utilisateur référence une donnée D, le système déduit qu'il aura prochainement besoin des données sémantiquement liées à cette donnée.

✓ Comment déterminer le ou les nœuds qui doivent devenir des hôtes de nouvelles copies de données?

La détermination d'un nœud hôte pour une copie a pour objectif de rendre la copie la plus accessible possible par les nœuds. Il peut être intéressant de répliquer une donnée importante sur un nœud ayant une forte connectivité afin d'augmenter le nombre de chemins possibles pour l'atteindre. Mais si, par exemple, la charge en énergie du nœud choisi est faible voire très faible, ce choix s'avèrera vite sans intérêt. La fiabilité des liens de communication est un autre critère déterminant de la fiabilité d'un serveur.

✓ Quelle est la granularité de réplication des données?

Il n'est pas utile de répliquer toute une base de données si, par exemple, les utilisateurs qui en partagent l'accès ne peuvent travailler que sur une petite partie (par exemple, bloc d'enregistrement) à une période de temps donnée.

Il peut par ailleurs, être intéressant d'avoir plusieurs copies d'un petit fichier, tandis qu'avec un gros fichier, nous finissons par obtenir certainement une saturation des mémoires de réplication disponibles.

✓ Enfin, dans le cas de l'utilisation de plusieurs métriques, lesquelles privilégier?

Si un nœud ne dispose que de l'espace nécessaire à une seule donnée, doit-il héberger la donnée dont il a le plus besoin et qui est toujours atteignable ou une donnée qui n'est pas d'un grand intérêt pour lui mais qui représente un intérêt certain pour les nœuds de son voisinage.

✓ Quel est le type des données traitées?

Beaucoup de systèmes de réplication sont définis de manière générale pour des données de type quelconque. Cependant, certaines approches ont été développées pour des données d'un type bien défini et, elles ne peuvent être appliquée à un autre type de données. Par exemple, [Neyem 06] considère des documents XML et permet le partage de branche du document. Ce mode de partage ne peut être appliquée à un type de données non hiérarchiques. La technique de réplication est fortement dépendante du type de la donnée. Elle ne peut être appliquée que pour le type de données considérées. On trouve aussi, l'approche des Transformées Opérationnelles (TO) [Oster 06]. Dans cette dernière approche, le type de données considéré est le type caractères chaînés pour un éditeur collaboratif.

✓ Quels paramètres devront être déterminés automatiquement et comment?

Une technique de réplication doit souvent allouer des copies sur les nœuds qui en ont le plus besoin ou dans le voisinage proche de ces nœuds. Ce besoin peut être défini manuellement par l'utilisateur lui-même. Ce qui n'est pas toujours intéressant car, il peut oublier de faire certaines actions. Des paramètres observés automatiquement sont souvent calculés pour prédire le besoin d'un utilisateur. Les fréquences d'accès à une donnée sont, sans aucun doute, un de ces paramètres, si ce n'est le plus significatif. Un algorithme de réplication doit prendre le soin d'évaluer au mieux les besoins futurs des utilisateurs pour une plus grande efficacité. D'autres paramètres peuvent être pris en compte tel que la sémantique des données qui peut permettre de déduire les besoins ultérieurs d'un utilisateur. La sémantique des données permet d'établir des liens entre les données et de déduire que si un nœud vient d'utiliser une donnée *Di*, il devra utiliser la donnée *Dj* dans un futur proche. Dans ce cas aussi, la sémantique des données est définie à travers des métas données (structure descriptives de données) qui peuvent être initialisées par l'utilisateur lui-même. Ces structures donnent une description des relations sémantiques entre les données. Mais encore une fois la détermination automatique de cette sémantique est plus intéressante bien qu'elle ne soit pas toujours possible.

2.5 *Challenges de la réplication*

Le contexte des réseaux mobiles ad hoc doit considérer le partitionnement comme un des états ordinaires du réseau. Pour assurer la disponibilité d'une donnée, il faut alors en créer des copies. Et, pour une meilleure disponibilité, il faut tenter de prévoir au moins une copie sur chaque partition. L'allocation de copies doit être conçue comme un processus dynamique et réactif aux changements de la topologie et de la demande des utilisateurs. En plus du partitionnement, une technique de réplication doit faire face aux contraintes induites par les limitations des mobiles (laptop, PDA, ..) eux mêmes et par les communications sans fil (voir chapitre précédent). Un nombre de défis doivent être surmontés par les MANETs dont principalement [Yin 04a][Padmanabhan 07]:

2.5.1 **Adaptation à une capacité mémoire limitée**:

L'un des problèmes majeurs soulevé par la réplication est: où placer les copies? Ce problème ne se poserait pas si un nœud mobile disposait d'un espace mémoire toujours suffisant pour héberger les copies de toutes les données partagées. Ce qui n'est évidemment pas le cas, l'espace de réplication disponible sur un nœud varie suivant ses caractéristiques propres et n'est pas nécessairement le même pour tous les nœuds du réseau. Sur un nœud mobile, l'espace mémoire local n'est pas entièrement mis à la disposition du partage. Généralement, un certain espace est consacré au partage dans le cas où le nœud est coopératif.

Les méthodes de réplication doivent nécessairement tenir compte des contraintes en mémoire pour rester réalistes. La sélection des données à répliquer et des nœuds qui doivent les héberger devient alors une opération critique et déterminante pour l'efficacité de la réplication. Cependant, de même que pour le routage, l'auto gestion et la coopération des nœuds d'un réseau mobile ad hoc est aussi un atout de l'accès aux données. Chaque nœud qui collabore partagera son espace de stockage avec ses nœuds voisins. On peut trouver des nœuds égoïstes qui profitent des services des autres nœuds mais qui utilisent divers artifices pour les priver de leurs potentialités en ressources afin de les préserver uniquement pour leur intérêt propre.

Le partage de l'espace de stockage impose la réalisation d'un compromis entre la satisfaction des besoins propres au nœud et la satisfaction de son voisinage. Le partage ne doit pas se faire au détriment d'un noeud. Ce compromis n'est pas évident pour un processus de réplication. En effet, jusqu'à quel point un nœud doit pouvoir satisfaire ses besoins avant d'attribuer une partie de son espace pour le partage?

2.5.2 Adaptation à une capacité énergétique limitée:

L'alimentation en énergie étant assurée par des batteries à capacité finie, leurs décharges éventuelles peuvent occasionner de nombreux problèmes. Si le nœud concerné constitue le seul lien entre deux parties du réseau, un partitionnement se produit. S'il hébergeait des données importantes non disponibles sur d'autres nœuds, ces données deviennent inaccessibles. L'énergie est un critère capital dont doit s'en soucier toute solution qui tient à maintenir l'accès à des données partagées [Yin 04a].

D'autre part, les opérations d'accès aux données engendrent une consommation d'énergie de la part du nœud qui sollicite l'accès, de la part des nœuds qui participent au routage de la requête et aussi, de la part du nœud serveur. Un serveur peut ainsi subir une chute de son niveau d'énergie si plusieurs nœuds le sollicitent.

Une technique de réplication doit réduire la consommation en énergie des serveurs par une répartition équitable de la charge sur l'ensemble des serveurs. Les serveurs à niveau énergétique élevé sont plus appropriés à l'hébergement des données à fort taux d'accès. Les nœuds clients ont aussi une source d'énergie limitée. La gestion de l'accès aux données doit réduire les temps de réponse et éviter les attentes très longues qui consomment inutilement de l'énergie. Les délais d'attente doivent être contrôlés et bornés.

Bien qu'elle ne soit pas prise en compte par la plupart des solutions existantes, l'énergie occupe une importance capitale dans la mise en œuvre d'une méthode efficace de réplication. Contrairement à beaucoup de pannes, les décharges de batterie peuvent être facilement prévues et prises en compte lors du placement des copies. Enfin, comme la mémoire, son partage et son utilisation nécessitent un compromis, de façon à maintenir tant que possible tous les nœuds en fonctionnement.

2.5.3 Adaptation aux restrictions des liens sans fil:

L'échange de messages de contrôle nécessaires à la collaboration entre les nœuds en vue de parvenir à une réplication efficace, peut induire une consommation excessive de la bande passante disponible. D'où des retards dans la livraison des données et des messages. Un retard dans la livraison des données peut conduire à des dommages importants pour les applications critiques. Les retards de livraison des messages de contrôle affectent négativement l'ensemble des applications y compris le processus réplication elle même. Dans ce cas, l'évaluation de l'état de l'environnement global ou partiel pour la prise de décision devient dépassée et invalide.

2.5.4 Adhésion des déconnexions omniprésentes:

Un serveur qui se déconnecte va provoquer l'inaccessibilité des données qu'il contient pour les nœuds dont il se sépare. La déconnexion d'un client fait que le temps de traitement des requêtes du client par un serveur, est inutile. La déconnexion du serveur entraîne des délais d'attente importants et superflus au niveau du client. Cette situation peut être évitée par une réplication préventive des déconnexions. C'est-à-dire la réplication de ces données avant la déconnexion partielle ou complète d'un serveur [Padmanabhan 07].

Cependant, si on peut estimer la charge d'une batterie pour déterminer quand est ce qu'elle se déchargera complètement et en prévoir les conséquences, il est difficile de prévoir toutes les fréquentes causes de déconnexions et déterminer avec précision quand est ce qu'elles surviendront. Il est très difficile voire des fois impossible de répondre à certaines des questions soulevées par le mode de fonctionnement des réseaux mobiles ad hoc, parmi lesquelles nous avons :

- Quand est ce que qu'un utilisateur éteindra son poste pour économiser sa batterie?
- Quand est ce qu'un poste tombera en panne matérielle ou logicielle?
- Quand est ce qu'un utilisateur sortira de la zone de couverture de tous les nœuds du réseau?

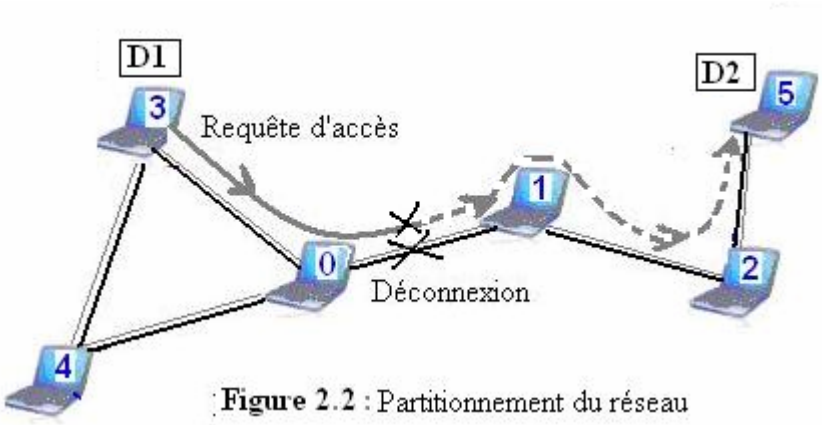

Figure 2.2 : Partitionnement du réseau

2.5.5 Prise en charge du partitionnement:

Le partitionnement est un problème des plus graves sur les réseaux mobiles ad hoc. La principale cause du partitionnement est les déconnexions (voir figure 2.2) mais aussi le défaut en énergie, panne subite,…Une technique de réplication doit prendre en considération le partitionnement. Pour cela, elle doit considérer des protocoles de prédiction du partitionnement. Ces protocoles peuvent rapidement s'avérer très complexes par l'adjonction des divers contrôles dynamiques et continus des causes du partitionnement [Hauspie 03b].

2.5.6 Adaptation à la mobilité:

La stratégie de réplication choisie sur un environnement doit être adaptée au degré de mobilité. La mobilité d'un groupe d'explorateurs en montagne est différente de la mobilité d'un autre groupe d'explorateurs munis de véhicules au désert. Certaines solutions peuvent convenir à une mobilité importante alors que d'autres ne sont efficaces que pour une catégorie d'applications où la mobilité est restreinte. Une méthode de réplication qui ne convient pas à la mobilité de l'environnement peut, par exemple, se trouver dans une situation où un nœud devient vite très éloigné d'une copie de donnée créée pour satisfaire au mieux ses demandes d'accès.

D'autres part; les nœuds mobiles étant souvent en déplacement, il n'est pas possible de déterminer la trajectoire d'un nœud. Par conséquent, le maintien des informations sur la localisation des copies et des clients peut produire un surcoût non négligeable. Ce surcoût est aussi fonction du degré de mobilité.

2.5.7 Type d'application:

Une technique de réplication doit s'accorder aux exigences de l'application qui sollicite les données. Par exemple, dans le cas d'une application temps réel tel qu'un sauvetage, les délais de réponses sont critiques. Dans ce cas, il faut, par exemple, prévoir, des délais d'expiration pour les requêtes d'accès. Les conditions d'exécution de l'application déterminent clairement les conditions de réplication et d'accès.

2.6 *Gestion de la cohérence*

2.6.1 Type de cohérence:

Il existe des types de cohérence différents selon la nature des données et l'application considérée [Dedieu 00] [Judge 98]:

a) Cohérence forte:

La cohérence forte ou stricte impose que toutes les copies physiques apparaissent comme une seule entité logique. Ce qui signifie que toutes les copies doivent avoir la même valeur à tout instant.

Pour assurer la cohérence stricte, les mises à jour ne sont permises que sur une copie unique à un instant donné. Les applications nécessitant une cohérence forte sont dites pessimistes.

Ce type de cohérence ne convient pas :

- Lorsque l'efficacité est un critère prépondérant.
- Lorsque la disponibilité des objets doit être très grande.
- Lorsque les temps de communication sont importants
- Lorsque la sémantique des objets n'est pas forcément très stricte et que l'on accepte de laisser des copies diverger.

b) Cohérence faible:

En cas de cohérence faible, la lecture d'une copie locale ne reflète pas forcément toutes les écritures antérieures. Cependant, elle garantit que ces dernières soient toutes répercutées sur la copie au bout d'un temps fini.

Cette cohérence est adaptée à certains types d'applications dites *optimistes* qui tolèrent le décalage des contenus des copies tels que les agendas partagés et la messagerie électronique.

La cohérence faible, consiste à laisser évoluer chaque copie de façon autonome, puis de faire des contrôles à posteriori. Chaque serveur d'une donnée informe les autres serveurs de la même donnée des modifications qu'il a effectué.

2.6.2 Réconciliation :

En cas de cohérence faible, le contrôle a posteriori permet la convergence ou la re-synchronisation des copies après une phase dite de *réconciliation*. Une réconciliation est un protocole exécuté pour faire converger les copies vers un même contenu. Souvent ce protocole est périodique.

2.6.3 Conflits de mise à jour:

Lors de cette phase, des écritures conflictuelles peuvent être découvertes sur des copies différentes. Par exemple, deux enseignants ont réservé la même séance pour deux modules différents. Le premier enseignant effectue la mise à jour m1 sur la séance S1 et le deuxième effectue la mise à jour m2 sur la séance S1. Ces mises à jour génèrent ce qu'on appelle un *conflit* qui sera détecté dans la phase de synchronisation.

Dans ce cas, il faut défaire les modifications ("Roll back") et en référer à l'arbitrage d'un coordinateur administrateur ou aux utilisateurs. Il existe des systèmes complexes qui

proposent des procédures automatiques pour la résolution des conflits. La détection automatique est basée sur des définitions de pré conditions sémantiques.

2.6.4 Réplication optimiste:

Cette approche offre une cohérence faible. Elle permet un plus haut degré de parallélisme et une meilleure disponibilité des données que ne le permet l'approche pessimiste offrant une cohérence forte. Elle est d'autant plus efficace qu'il y a peu d'accès conflictuels.

Les copies évoluent indépendamment. Par la suite, un protocole est exécuté pour reproduire toutes les modifications sur toutes les copies et les faire converger vers un état commun. Les stratégies les plus couramment utilisées sont la propagation immédiate et la propagation à la demande. Dans la première stratégie, après toute modification locale d'une copie, une procédure de synchronisation est initiée. Dans la deuxième stratégie, le système offre un support pour répertorier les opérations qui ne sont pas encore reportées aux autres copies. Ce support est un journal des opérations qui doivent être effectuées sur toutes les copies. Ces opérations sont notifiées aux autres copies soit à l'initiative du serveur local, soit à l'initiative de chaque copie.

La stratégie de synchronisation appropriée dépend de la fréquence des modifications, du nombre de serveurs, de la localisation des utilisateurs, et des contraintes sur la cohérence des copies perçues. Quelle que soit l'approche de contrôle de convergence mise en œuvre et quelle que soit la méthode et la stratégie de synchronisation adoptée, les copies n'ont pas, à tout instant, le même état (par définition de la cohérence faible). Ceci est dû essentiellement au fait que les communications ne sont pas instantanées et que des fautes diverses peuvent survenir. La réplication optimiste est appropriée à certaines applications très courantes comme dans l'informatique mobile (nomade) où un utilisateur peut travailler en mode déconnecté.

2.6.5 Protocole de Gestion de Cohérence:

Un protocole de gestion de cohérence entre les différentes copies d'un même objet a pour objectif d'améliorer la fiabilité des données et/ou les performances (tant en écriture qu'en lecture) du système. Malheureusement, ces deux objectifs sont antagonistes. Pour obtenir une bonne fiabilité, il est nécessaire d'avoir une cohérence forte ce qui pénalise les performances.

A l'inverse pour obtenir de bonnes performances il est nécessaire de relâcher la cohérence, ce qui pénalise la fiabilité. Cela est d'autant plus vrai que l'on augmente le nombre de copies. En effet, dans le cas de la cohérence forte, un site accède toujours au dernier élément de la séquence globale des écritures. Ceci peut être réalisé par un protocole basé, par exemple, sur une diffusion atomique des valeurs, ou par le verrouillage global de la donnée

avant l'ajout d'une opération. Ces méthodes se basent sur un ordre total des écritures. Par contre dans le cas de la cohérence faible, la valeur d'une copie sur un site n'est plus nécessairement la dernière de la séquence globale. Cela est généralement réalisé par la construction d'un ordre partiel sur les écritures. Il est donc nécessaire de faire des compromis.

Conclusion

Ce chapitre nous a permit de définir les principaux concepts de la réplication en environnements mobiles ad hoc. La réplication est plus qu'avantageuse pour ces environnements; elle est indispensable et incontournable au fonctionnement des MANETs. Un rappel assez détaillé des intérêts apportés par l'adjonction d'un système de réplication a été présenté. Cependant, elle peut être cause d'un certain nombre de nouveaux problèmes tels que le surcoût induit et la divergence des copies manipulées en écritures concurrentes.

Chapitre 3

Systèmes et méthodes de réplication de données

Introduction

L'intérêt croissant accordé à la mobilité des utilisateurs a donner naissance à un nouveau type d'application offrant au moins deux modes de fonctionnement: le mode connecté et le mode déconnecté [Kistler 91]. Nous pouvons aussi avoir des modes intermédiaires comme le mode partiellement connecté. Afin d'assurer à l'utilisateur la transparence vis-à-vis du changement de modes, les systèmes actuels doivent prévoir des services de détection de déconnexions, de chargement préventif en cache des informations et, un service d'intégration qui prend en charge les reconnexions et les mène vers des états cohérents [Kouici 06]. Le problème de maintien de cette cohérence est la conséquence de cette mobilité. Des outils parfois complexes de sauvegarde régulière d'état, de journalisation des opérations et de synchronisation sont mis en œuvre.

La réplication est l'un des services alliés de cette indépendance de l'utilisateur vis-à-vis de sa position et de ses déplacements. Dans le domaine de l'informatique mobile, plusieurs stratégies de réplication ont été proposées [Barbara 94][Wolfson 97][Jing 97][Pitoura 95]. Elles supposent que les nœuds mobiles accèdent aux données sur des sites fixes. Elles abordent la question de la cohérence entre les données originales et leurs répliques à moindre coût de communication.

Plusieurs études ont été menées afin d'améliorer l'accessibilité des données sur les réseaux mobiles ad hoc via la réplication [Yin 04][Nuggehalli 03][Huang 06]. Dans [Chen 02] puis dans [Sailhan 03], des mécanismes de détection de partitionnement permettent de prévoir des copies avant le partitionnement. Cependant, ces méthodes supposent que les nœuds ont une capacité mémoire illimitée. D'autres travaux se sont penchés sur le problème d'optimisation de la consommation de l'énergie pour l'accès aux données dont [Cao 04]. Cependant ce dernier considère un serveur central pour assurer la mise à jour et la diffusion des données modifiées.

La réplication nécessite une maintenance régulière nécessaire à cause du dynamisme de la topologie du réseau. Un grand nombre d'algorithmes de réplication assurent une maintenance périodique de la localisation des copies [Hayashi 05][Hara 01][Yin 04][Jing 04].

Les techniques de réplication peuvent être classées selon leur architecture:

(i) Utilisation d'une topologie plate
(ii) Utilisation d'une logique en groupes de nœuds pour la réplication
(iii) Utilisation d'un nœud central assurant des fonctionnalités de contrôle particulières pour la réplication.

En plus de leur architecture, les différentes techniques se distinguent par la prise en compte d'un ou de plusieurs paramètres de création et de localisation de copies que nous pouvons résumer par: (1) état de la topologie, (2) taille des données, (3) fréquence des accès, (4) niveau de batterie, (5) profil de l'utilisateur, (6) sémantique de la donnée.

[Padmanabhan 07] classifie les techniques de réplication en quatre catégories: "Non partition-aware", "Non power aware", "Non real-time-aware", et une catégorie qui englobe le reste des techniques qui ne correspondent à aucune de ces classes. Cette classification semble

insuffisante car plusieurs solutions ne se retrouvent dans aucune des classes définies. La classification que nous voyons ressortir en premier plan est:

- Ensemble des techniques basées sur une topologie plate
- Ensemble des techniques basées sur des groupes de réplication
- Ensembles des techniques basées sur la détection des partitionnements
- Ensemble des techniques de réplication dédiée à un type particulier de données (par exemple, texte ou chaîne de caractères pour un éditeur collaboratif, ou données structurées tel que XML)

Le partage de données sur des environnements de type MANETs passe nécessairement par la création et la distribution de copies. Le placement de ces copies peut nécessiter un algorithme de remplacement de données. Parmi les techniques de remplacement utilisées on peut citer LRU (Least Recently Used), qui consiste à remplacer la copie la moins récemment accédée; NRU (Not Recently Used), qui consiste à remplacer aléatoirement l'une des copies les moins récemment accédées; et NFU (Not Frequently Used), qui consiste à remplacer la copie la moins fréquemment accédée. Ainsi, d'anciennes copies ou des copies peu utilisées peuvent être remplacées à condition que la copie à supprimer ne soit pas une copie *originale*.

Ce chapitre présente quelques uns des travaux les plus importants sur la gestion de l'accès aux données sur les réseaux mobiles ad hoc. Cette présentation précise le domaine d'application de chaque méthode, ses hypothèses, ses avantages et ses inconvénients. Ce chapitre englobe les techniques de mise en cache et de réplication de données non mises à jour dans une première partie. Le but est d'étudier les différentes approches: de choix de la donnée à copier, de détermination de son emplacement, et de l'instant d'intervention du processus de réplication. Une deuxième partie est consacrée à la prise en charge des mises à jour par les systèmes de réplication.

3.3 *Méthodes de réplication sans mise à jour de données*

1 - *Méthodes SAF, DAFN et Améliorations*

3.1.1.1 Méthodes *SAF* et *DAFN* [Hara 01]

Les données à répliquer sont supposées accédées en lecture seule. Ces techniques sont basées sur les fréquences d'accès aux données. Ces fréquences d'accès (le taux ou le nombre de fois qu'un nœud accède à une donnée durant un intervalle de temps) sont supposées connues et statiques. Elles ne changent donc pas. Cette hypothèse ne semble pas réaliste car elle suppose que les besoins des utilisateurs sont connus et figés. C'est sur cette fréquence d'accès qu'est basée la décision de répliquer ou non une donnée et c'est aussi sur la base de ce paramètre que se fait la sélection du nœud sur lequel la copie sera allouée. Ces méthodes prennent en considération le fait que chaque nœud a une capacité de stockage limitée. La notion de copie *originale* est adoptée dans les travaux de T. Hara afin d'éviter l'élimination complète d'une donnée sur un réseau mobile ad hoc.

SAF (Static Access Frequency) est une méthode qui fait que chaque nœud réplique les données dont il a le plus besoin sans prendre en considération son voisinage (voir exemple en table 3.1 et figure 3.1). Il utilise pour cela une liste décroissante des fréquences d'accès aux données. Ce type d'algorithmes est dit *glouton*. A travers l'exemple de la figure 3.1, il apparaît clairement un problème de redondance caractéristique de ce type d'algorithme. La majorité des données se retrouvent sur presque tous les nœuds. Nous avons, sur cet exemple, six nœuds ayant chacun la capacité d'héberger quatre données différentes. Pourtant, le nombre total de données différentes présentes sur le réseau est six (D_1, .., D_6).

Données	Noeuds Mobiles					
	*M*1	*M*2	*M*3	*M*4	*M*5	*M*6
*D*1	0.65	0.25	0.17	0.22	0.31	0.24
*D*2	0.44	0.62	0.41	0.40	0.42	0.46
*D*3	0.35	0.44	0.50	0.25	0.45	0.37
*D*4	0.31	0.15	0.10	0.60	0.09	0.10
*D*5	0.51	0.41	0.43	0.38	0.71	0.20
*D*6	0.08	0.07	0.05	0.15	0.20	0.62
*D*7	0.38	0.32	0.37	0.33	0.40	0.32
*D*8	0.22	0.33	0.21	0.23	0.24	0.17
*D*9	0.18	0.16	0.19	0.17	0.24	0.21
*D*10	0.09	0.08	0.06	0.11	0.12	0.09

Table 3.1: Exemple 1 de fréquences d'accès

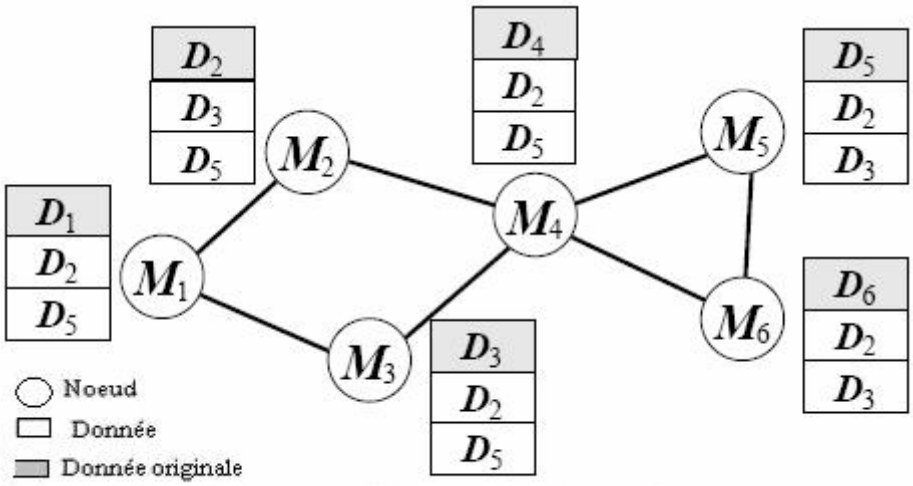

Figure 3.1: Exemple d'exécution de SAF

La méthode ***DAFN*** (Dynamic Access Frequency and Neighborhood) considère des fréquences d'accès calculées dynamiquement. Cette solution tente de remédier au problème précédent de redondance des données. Elle procède comme suit: la méthode ***SAF*** est appliquée normalement, ensuite une procédure d'élimination des copies redondantes sur les voisins immédiats est exécutée. Périodiquement, les nœuds diffusent à un saut les identités des données disponibles localement. Si deux noeuds ont chacun une copie d'une même donnée, le nœud ayant la plus petite fréquence d'accès à cette donnée la remplace par la prochaine donnée sur sa liste des données ordonnées par fréquence d'utilisation (figure 3.2). Dans le cas où l'une des copies est la copie originale, c'est nécessairement l'autre copie qui est remplacée quelque soit sa fréquence d'accès. Le remplacement des copies redondantes

permet d'avoir un plus grand nombre de données différentes dans un même voisinage. L'accessibilité se trouve alors améliorée.

Cependant, cette redondance n'est pas complètement éliminée par le fait que le processus de recherche de copies redondantes est réalisé en une seule passe. En effets, les remplacements peuvent à leur tour être sources de redondances. La méthode ***DCG***, présentée plus loin, est une amélioration de ***DAFN*** qui tente de réduire ce problème de redondance.

Nous observons pour ***DAFN***, l'inconvénient du pré chargement des données selon les fréquences locales d'un nœud. Le coût de ce pré chargement est en partie inutile. Car, ces données risquent d'être remplacées l'instant qui suit pour éviter les duplications entre voisins. Lorsqu'une copie redondante est détectée, la copie avec la fréquence d'accès la plus basse est supprimée à condition qu'elle ne soit pas la copie originale. L'espace est alors attribué à une autre donnée selon la liste décroissante des fréquences d'accès.

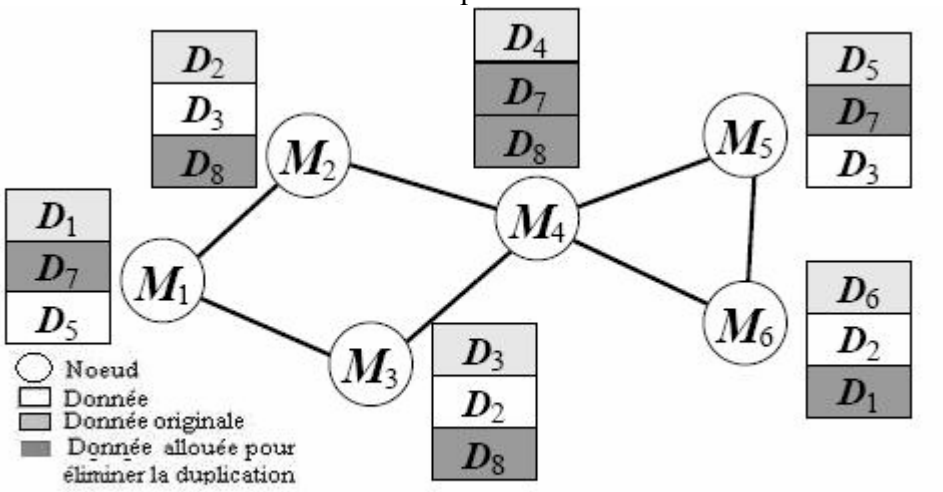

Figure 3.2: Exemple d'exécution de DAFN

3.1.1.2 Méthodes *DAFN-S1* [Hara 03a]

Dans cette amélioration de la méthode ***DAFN***, la stabilité des liens radio est prise en considération. La stabilité d'un lien exprime une forte probabilité qu'un lien soit maintenu pour une période de temps. Deux nœuds mobiles connectés par un lien radio, peuvent provoquer une rupture de ce lien lorsqu'ils s'éloignent l'un de l'autre d'une distance plus grande que la portée de communication.

L'élimination d'une copie redondante peut s'avérer cause d'une inaccessibilité à la donnée si l'instant d'après le lien reliant le nœud à la deuxième copie se trouve rompu. La comparaison des mouvements de deux nœuds mobiles voisins, permet d'estimer l'instant de leur déconnexion. ***DAFN-S1*** utilise cette estimation pour déterminer les liaisons stables d'un nœud. Les données sont là aussi supposées accédées en lecture seule. Mais des hypothèses supplémentaires ont été nécessaires:

- Les nœuds mobiles se déplacent selon le modèle RWP ("Random Way Point") [Camp 02] [Naoumov 03]. Chaque nœud mobile connaît sa destination, sa vitesse et son temps de repos et peut connaître sa position courante grâce à la présence d'un GPS.
- Toutes les données sont de la même taille, et l'originale de chaque donnée est tenue par un nœud mobile particulier dans le système.

En utilisant le modèle RWP, le délai t_{ij} de déconnexion de deux nœuds mobiles M_i et M_j peut être estimé dans l'un des cas suivants:

1. Les deux noeuds mobiles se reposent. Le minimum du temps de repos restant représente t_{ij}.
2. Chaque nœud se déplace d'une position courante vers une position de destination. L'instant où la distance, qui les sépare devient plus grande que la portée de communication, représente t_{ij} en supposant que l'instant de départ est 0. t_{ij} peut être estimé par une valeur plus petite que les temps d'arrivée des deux noeuds mobiles à leur destination.
3. Un noeud mobile se repose et l'autre se déplace. Ce cas est similaire au cas précédent. Le délai de parcours nécessaire pour que la distance qui les sépare devienne plus grande que la portée de communication représente t_{ij}. Il est choisi plus petit que le temps de repos restant du premier nœud et le temps d'arrivée du second.

La stabilité **B** des liens est exprimée par une valeur réelle qui varie entre 1 et 0. Grâce aux informations sur la vitesse et sur la destination des deux nœuds. Il est possible de calculer le temps $t_{ij,}$ pour que la distance séparant deux nœuds devienne plus grande que la portée de leur liaisons radio. Un lien radio entre deux nœuds M_i et M_j est stable pendant une période de temps **T**, si le temps t_{ij} est supérieur a **T** (pour ***DAFN-S1***, **T** est la période d'exécution de l'algorithme d'allocation). Dans ce cas, la valeur de $\mathbf{B_{ij}}$ de stabilité du lien reliant M_i à M_j est égale à 1, sinon si $t_{ij} \leq \mathbf{T}$ (le lien va se couper avant la fin de la période **T**) la valeur de B_{ij} est estimée par $t_{ij}/\mathbf{T}$ [Har03a] :

$$B_{ij} = \begin{cases} 1 & \text{si } t_{ij} > T \\ \dfrac{t_{ij}}{T} & \text{sinon} \end{cases}$$

La valeur de $\mathbf{B_{ij}}$ devra être supérieure à une certaine valeur x ($0<x \leq 1$) pour estimer que le lien entre les nœuds M_i et M_j restera stable durant la prochaine période **T** de réallocation.

DAFN-S1 adopte fondamentalement l'algorithme de ***DAFN***. A part que, la stabilité des liens entre les nœuds mobiles est prise en considération lors de l'élimination les copies doubles. L'élimination de la redondance se fait alors comme suit: Pour chaque paire de nœuds M_i et M_j voisins, la valeur $\mathbf{B_{ij}}$ sera calculée. Si **Bij**>x, le lien entre ces deux nœuds est jugé stable. S'il existe des duplications de données entre ces deux nœuds, celle avec la fréquence d'accès la plus faible est sélectionnée pour un remplacement à condition qu'elle ne soit pas *originale*.

Le résultat de l'exécution de ***DAFN-S1*** sur un ensemble de nœuds voisins est illustré par un exemple représenté par la table 3.2 et la figure 3.3. ***DAFN-S1*** commence par l'exécution de la méthode ***SAF*** comme première étape. Une deuxième étape sera exécutée pour éliminer les duplications entre les nœuds voisins à lien stable. Pour le couple de nœuds (M1, M2) la donnée D2 a été remplacée par D8. Par contre pour le couple (M2, M3) connecté par un lien

radio instable, la duplication de la donnée D3 a été ignoré (le lien ne tiendra pas jusqu'à la prochaine exécution de la méthode de réplication).

Donnée	Mobile				
	M1	M2	M3	M4	M5
D1	0.65	0.25	0.17	0.22	0.31
D2	0.44	0.62	0.23	0.40	0.11
D3	0.20	0.47	0.50	0.18	0.23
D4	0.10	0.53	0.67	0.60	0.09
D5	0.51	0.28	0.72	0.58	0.71
D6	0.08	0.38	0.43	0.33	0.26
D7	0.17	0.32	0.11	0.49	0.62
D8	0.22	0.24	0.21	0.23	0.57

Table 3.2: Exemple 2 de fréquences d'accès

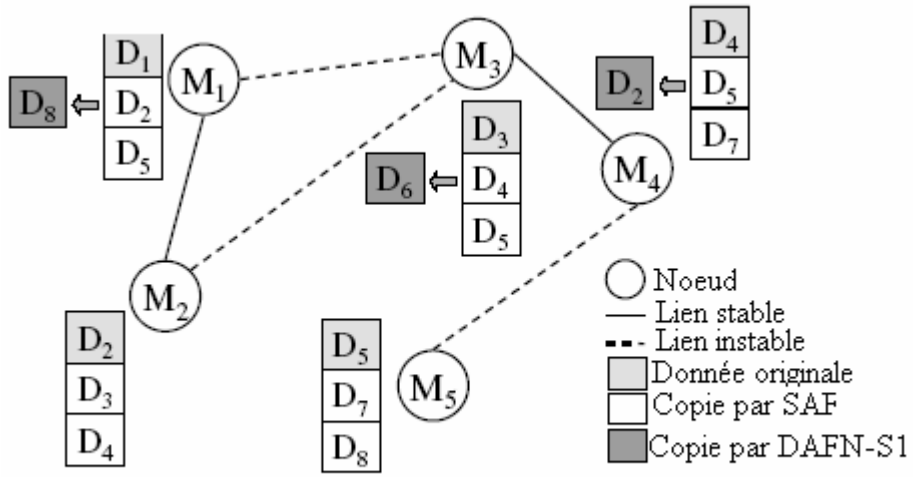

Figure 3.3: Exemple d'exécution de DAFN-S1

3.1.1.3 Méthodes *E-SAF* et *E-DAFN* [Hara 03b]

Ces méthodes sont en fait les solutions précédentes étendues pour considérer les données mises à jour périodiquement. Cette hypothèse de mise à jour périodique ne semble pas très réaliste. Ces méthodes se basent sur le comportement des accès et le modèle de lecture/écriture.

Pour une adaptation des solutions précédentes à un environnement où les mises à jours sont périodiques, un facteur *PT* est défini. Il représente le nombre moyen de requêtes d'accès émises pour une donnée Dj jusqu'à sa prochaine mise à jour. Sa valeur est définie comme suit:

$$p_{ij} \cdot \tau_j = p_{ij} \cdot (T_j - t_j)$$

Tels que :

- p_{ij} est la probabilité qu'une requête d'accès à une donnée D_j soit émise par un noeud M_i à une période de temps, c'est-à-dire la fréquence d'accès.

- τ_j représente le temps restant pour la génération de la prochaine mise à jour de D_j.
- T_j est la période de mise à jour de D_j ;
- t_j est le temps écoulé depuis la dernière mise à jour de D_j à la période la plus récente.

PT prend sa valeur maximale ($p_{ij} \cdot T_j$) à l'instant de mise à jour de Dj. Ainsi, l'allocation de copies de données ayant une valeur élevée du facteur *PT*, permettra une meilleure accessibilité. L'allocation des copies ne considère plus simplement les fréquences d'accès en lecture comme c'est le cas pour les méthodes proposées dans [Hara 01] mais plutôt ce facteur *PT*. Les données seront ordonnées selon leur valeur *PT* et celles avec les plus fortes valeurs de *PT* seront répliquées.

L'algorithme de ***E-SAF*** (Extended Static Access Frequency) est similaire à celui de ***SAF*** en considérant la valeur de *PT*. Si un nœud qui doit répliquer localement une donnée n'arrive pas à atteindre le nœud détenant la copie originale, l'espace qui lui est réservé est temporairement utilisé pour contenir une des données déjà allouées lors des périodes précédentes mais non sélectionnées pour la période courante. Parmi les données possibles, celle avec le plus grand *PT* est choisie. Si aucune donnée ne peut y être sauvegardée alors l'espace reste libre. Lors d'une connexion au serveur *primaire* l'espace est occupée par une copie valide de la donnée initialement sélectionnée. ***E-SAF*** donne un même résultat d'exécution que *SAF* pour l'exemple de la figure 3.1 en considérant que les valeurs du tableau 3.1 sont les valeurs de *PT*.

La méthode *E-DAFN* (Extended Dynamic Access Frequency and Neighborhoods) commence par exécuter ***E-SAF*** en première phase au début de chaque période de réallocation. En deuxième phase, elle procède à l'élimination des redondances de copies. L'algorithme est alors similaire à celui de ***DAFN***:

1 - A chaque période de réallocation (période de réplication), chaque nœud diffuse ses fréquences d'accès aux données qu'il utilise. Cette diffusion permet aux nœuds de connaître les voisins à un saut.
2 - Chaque noeud exécute ***E-SAF*** comme une première réplication.
3 - Dans chaque ensemble de noeuds connectés et en commençant par le nœud à plus petit identifiant M_i. M_i va procéder à l'élimination des copies redondantes avec chacun de ses voisins. De la même manière que ***E-SAF***, si aucune donnée ne peut être sauvegardée à la place d'une copie supprimée, l'espace reste libre. A la reconnexion au serveur primaire l'espace sera occupé par une copie valide de la donnée sélectionnée à la période courante.

Cette méthode produit évidement un coût en trafic plus élevé que la méthode précédente. Cependant, l'accessibilité aux données est améliorée. L'inconvénient pour ces méthodes est qu'elles se basent sur un placement des copies en considérant le paramètre *PT*. Or, il est calculé avec l'hypothèse que les mises à jour sont régulières et périodiques. Ce qui semble loin de la réalité.

3.1.1.4 Méthode *C-SAF* et *C-DAFN*: [Hara 04]

[Hara 04] propose, encore une fois, des améliorations des méthodes de base présentées dans [Hara 01]: ***C-SAF*** (Correlated - SAF) et ***C-DAFN*** (Correlated - DAFN). La création de copies est fondée sur la "data priority" d'une donnée. Cette "data priority" est définie à partir de la relation entre les données accédées par l'utilisateur. La corrélation entre les données est la probabilité qu'un utilisateur accède à un ensemble de données. Cette corrélation est d'autant plus forte que la probabilité que les données soient accédées en même temps est grande. Ces travaux supposent que les accès aux données multiples sont émis en même temps et que les corrélations significatives sont connues et ne changent pas.

Le principe de la réplication est alors le suivant:

a) Pour toute donnée D_j, la fréquence d'accès est calculée comme suit:

$$F_{ij} = \sum_{k=1}^{n} p_i - j_k$$

Où, p_i-j_k représente la fréquence des accès d'un nœud N_i aux données D_j et D_k en même temps. Deux données parmi celles ayant la corrélation la plus forte sont choisies.

b) Parmi les données non sélectionnées, une nouvelle donnée avec la corrélation la plus forte aux données déjà sélectionnées est choisie.

c) Les données hébergées par le nœud se voient alors attribuer une priorité maximale. Les autres reçoivent une priorité qui dépend de l'ordre de leur sélection.

d) L'algorithme de réplication de chaque méthode est alors similaire à celui de la méthode correspondante présentée dans [Hara 01]. A part que, dans ce cas, l'algorithme se base sur la priorité définie par les corrélations et non plus uniquement sur les fréquences.

Ces solutions devraient donner une meilleure accessibilité dans les environnements où souvent les données sont accédées en même temps et non de manière isolée.

3.1.1.5 Méthode *E-SAF*$^+$ et *E-DAFN*$^+$ [HARA 06]:

Ces deux méthodes représentent une finalité aux améliorations précédentes [Hara 01][Hara 03] en considérant les mises à jour apériodiques des données. Les mises à jour sont appliquées aux données primaires puis propagées aux données secondaires. Les fréquences d'accès aux données de chaque nœud ainsi que les fréquences des mises à jour sont calculées périodiquement. Cependant, le réseau mobile ad hoc est supposé de taille assez petite c'est-à-dire d'une douzaine de nœuds.

[Hara 06] définit un facteur RWR (rapport *Read/Write)*: $RWR = R_{ij}/W_j$ où :

- R_{ij} représente la probabilité, dans une unité de temps, que le nœud M_i effectue une opération de lecture sur la donnée D_j.
- W_j représente la probabilité, dans une unité de temps, d'une mise à jour de la donnée D_j par le nœud possédant la copie originale.

Ce paramètre est défini comme critère de réplication : Si le RWR d'une donnée est supérieur à 1, la donnée est plus utilisée en lecture qu'en écriture et donc le facteur accessibilité doit être privilégié. La donnée est alors répliquée. Dans le cas contraire, la réplication devrait être évitée car la donnée est souvent mise à jour, et donc un protocole de cohérence coûteux devrait être exécuté.

Ces techniques utilisent aussi une notion de profil d'un utilisateur qui représente un ensemble de données ***"open objects"*** utilisées ou qui doivent être utilisées dans un futur proche par l'utilisateur. Ce profil peut être défini implicitement ou explicitement. Les données constituants ce profil sont triées par ordre décroissant de leur facteur RWR.

De même, le besoin urgent d'une donnée est considéré. Les données urgentes sont répliquées en priorité sans condition. Leurs fréquences d'accès sont calculées de la même façon que pour les autres données. S'il n'y a pas assez d'espace pour toutes les données urgentes, les données non urgentes avec les plus petits RWR seront remplacées. Le RWR d'une donnée urgente qui sera demandée dans un délai inférieur à la période de réallocation**,** sera défini par:

$$RWR_{ij} = MaxValue / HappenTime.$$

La valeur MaxValue assure que les données urgentes aient les valeurs les plus élevées de RWR. Ces données sont une catégorie des ***"open objects"***.

La méthode ***E-SAF***$^+$ exécute un algorithme périodiquement pour créer des copies de données. Chaque nœud possédant des données originales, envoie les informations concernant les fréquences de mises à jour de celles-ci. Un nœud M_i détermine alors le RWR de chaque donnée et alloue alors C_i (capacité mémoire) données dans l'ordre décroissant des RWR.

La méthode ***E-DAFN***$^+$ exécute un algorithme de réallocation similaire à celui de ***DAFN*** en substituant les fréquences d'accès par les valeurs du RWR. Les copies sont alors allouées dans l'ordre décroissant des RWR en favorisant les données "*open object*".

2 - *Méthode HybridCache* [Yin 04a] [Yin 04b]

Cette approche "*HybridCache*" met à contribution deux méthodes de mise en cache: la méthode "*CachePath*" et la méthode "*CacheData*". Elle vise l'amélioration des performances

en temps d'accès et en trafic par la détermination et le maintien à jour de chemins d'accès par "*CachePath*". L'accessibilité est aussi améliorée à travers la création et la mise en cache de copies locales par "*CacheData*". Un nœud détecte l'utilité de créer une copie par scrutation des requêtes qui l'atteignent. Une donnée avec un grand nombre d'accès est jugée importante et sa réplication utile. L'existence d'une copie locale permettra aux nœuds de satisfaire les prochaines requêtes à la donnée. Le nœud source de la donnée est un serveur relié par une infrastructure ou simplement un nœud hébergeant la copie *originale*.

Pour éviter que tous les nœuds, constituant un chemin d'accès entre un client et un serveur ne mettent en cache la même donnée, les conditions suivantes sont respectées:

→ Un noeud ne crée pas une copie pour une donnée si toutes les requêtes à cette donnée sont émises par un même nœud.

→ Un client est supposé créer une copie locale d'une donnée dont il a besoin.

3.1.2.1 Méthode "*CacheData*"

La figure 3.4 présente un exemple de réseau mobile ad hoc. On suppose que le nœud N11 détient une donnée D. Le nœud N11 peut être un nœud qui dispose d'une connexion à un réseau filaire. Sur un réseau mobile ad hoc les requêtes d'accès aux données sont acheminées saut par saut jusqu'à atteindre le nœud source et alors ce dernier répond et retransmet la donnée sollicitée. La méthode *CacheData* met en cache une donnée faisant l'objet de beaucoup de requêtes d'émetteurs différents ou lorsque le nœud local dispose d'espace libre. Par exemple, si N6 et N1 requiert l'accès à D à travers N5, N5 met en cache local la donnée D. Les futures requêtes de N3, N4 ou N5 pourront être servies directement par N5.

Cependant l'utilisation de la méthode *CacheData* doit être pratiquée de manière réfléchie car elle engendre une consommation de l'espace mémoire disponible. Pour cela, une règle est établie: *un nœud ne doit pas répliquer une donnée requise par un seul nœud*. Avec cette nouvelle règle, si le nœud N3 reçoit des requêtes pour D et que ces requêtes sont toutes émises par un même nœud N1, par exemple, alors N3 ne mettra pas en cache D mais N1 va créer une copie de la donnée D dans son propre cache. On remarque qu'une donnée est mise en cache par au moins le nœud qui y accède.

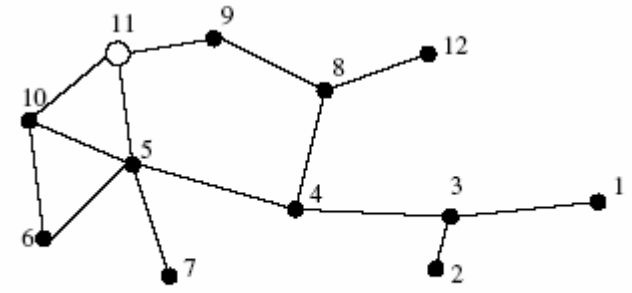

Figure 3.4: *Exemple de mise en cache de donnée*

3.1.2.2 Méthode "*CachePath*":

Sur l'exemple de la figure 3.4, supposant que, le nœud N11 reçoit une requête d'accès à la donnée D à partir de N1. Lorsque N3 participe à l'acheminement de la donnée de N1, il enregistre que N1 possède une copie de la donnée D. Si N2 requiert l'accès à D, N3 sait que N11 hébergeant la donnée source est à trois sauts et que N1 est à un seul saut. Ainsi, N3 achemine la requête vers N1 au lieu de N4. La méthode "*CachePath*" permet à un nœud de sauvegarder des chemins vers les copies de données.

Avec cette méthode un nœud n'a pas à sauvegarder tous les chemins des données qui le traversent. Lorsque la donnée D est transmise par N11 vers N1, elle parcourt les nœuds N5, N4 et N3 dans cet ordre. N4 et N5 n'ont pas besoin de sauvegarder une copie du chemin d'accès vers D car ils sont plus proches du nœud N11 source de la donnée que du nœud N1. Donc, un nœud ne sauvegarde un chemin d'accès que si ce chemin est plus court que le chemin menant vers le nœud source de la donnée (copie *originale*). La sauvegarde d'un chemin consiste à enregistrer uniquement les informations sur le nœud destination.

Dans le but d'éviter les redondances inutiles, lorsqu'un noeud reçoit une donnée à router de la source de donnée vers un noeud Ni, il sauvegarde dans son cache l'information selon laquelle la donnée est aussi disponible au niveau du noeud Ni. De même, lorsque ce noeud doit acheminer une requête concernant la donnée précédemment routée, il la redirige vers le noeud Ni si celui-ci est plus proche de lui que la source de la donnée. Ce qui a pour avantage d'économiser un certain nombre de sauts.

Avec cette méthode, on est confronté aux problèmes posés par la mobilité et la limitation de l'espace de stockage des chemins. Lorsque le noeud qui cache une copie de la donnée se déplace, le noeud au niveau duquel le chemin a été brisé le constate par l'absence de liens radio au moment du routage. Il redirige donc la requête vers la source initiale de la donnée.

De même, lorsqu'une copie est remplacée dans le cache d'un nœud, le chemin menant à cette copie devient invalide, bien que certains autres noeuds continuent de conserver cette correspondance dans leur cache. L'exploitation de ces correspondances invalides conduit à diriger des requêtes futures vers des noeuds qui ont déjà remplacée la donnée requise. A l'arrivée d'une requête de ce type, il devient également nécessaire de l'orienter vers la source initiale de la donnée. Le temps perdu par la requête avant d'être redirigée, les opérations supplémentaires engendrées ainsi que leur coût, peuvent sérieusement affecter la qualité de service. Pour cela, un noeud ne sauvegarde un chemin que si celui ci est plus proche de lui que la source de la donnée. Cette proximité est déterminée par une fonction *H*. Cette fonction est évaluée selon les informations fournies par le protocole de routage, telles que la stabilité du chemin, la distance en sauts et le temps de validité de la donnée.

Soit Ni un noeud chargé de retransmettre une requête (pour l'obtention de la donnée D), envoyée par le client Nj vers la source Ns. Soit Nk un noeud qui détient une copie. Si Ni sait que la donnée D est déjà disponible au niveau de Nk, l'arrivée de la requête au niveau de Ni sera traiter comme suit:

```
Si H (i, k) < H (i, s) alors
    Rediriger la requête vers Nk.
    Hsave = H (i, s)-H (i, k).
Sinon
    Rechercher la donnée depuis la source de donnée.
Fsi
```

3.1.2.3 Méthode "*HybridCache*":

La technique "*HybridCache*" tente de tirer profit des avantages des deux méthodes *"CachePath*" et *"CacheData"*. Lorsqu'un noeud achemine une donnée, il peut choisir de la mettre en cache ("*CacheData*") ou de sauvegarder uniquement le chemin permettant de l'atteindre sur le site destinataire ("*CachePath*").

Soit Di une donnée à acheminer, le choix de la méthode à utiliser se fait selon les critères suivants: la taille *Si* de Di, le temps de validité *TTLi* (Time To Live) de Di et le nombre de sauts conservés en chargeant Di noté Hsave (voir algorithme précédent). On constate assez aisément que:

- Si la taille *Si* est petite (*Si* < Seuil S), la méthode "*CacheData*" est mieux indiquée, car la donnée n'occupe pas beaucoup d'espace mémoire. Sinon mieux vaut utiliser *"CachePath*" pour économiser de l'espace mémoire.

- Si *TTLi* est petit (*TTLi*<SeuilTTL). CachePath ne serait pas intéressante vu que la donnée deviendra très rapidement invalide et que l'opération de redirection est coûteuse. Dans le cas contraire (TTLi élevé), il n'y aurait pas d'inconvénient à utiliser CachePath.

- Si Hsave est élevé (Hsave<SeuilHsave), *CachePath* pourra être utilisée. Sinon mieux vaut cacher entièrement la donnée pour améliorer les performances, s'il existe de l'espace mémoire disponible c'est à dire utiliser (*CacheData*).

Algorithme de *HybridCache* au niveau d'un noeud Ni

```
A) A l'arrivée de la donnée Di
    Faire
        Si (la requête provient du noeud courant) alors
            CacheData (Di) ;
            Aller à fin ;
        Fsi
        Si (il y a une version ancienne de Di  en cache) alors
            Mettre à jour Di
        Sinon
            Si (Si < Seuil S  ou il y a une copie invalide en cache
                                ou un chemin menant à  Di en cache)
            alors
              CacheData (Di)
            Sinon
              Si (Hsave>SeuilHsave et TTLi > SeuilTTL) alors
                    CachePath (Di )
              Fsi
            Fsi
        Fsi
    Fait
B) Si un remplacement est nécessaire à faire
    alors
        Tant que (il n'y a pas suffisamment de place)
        Faire
            Supprimer une donnée invalide ;
        Fait ;
                                /*s'il n'y a toujours pas suffisamment de place et
                                                il n'y a plus de données invalides*/
        Tant que (il n'y a pas suffisamment de place)
        Faire
            Supprimer une donnée valide ;
        Fait ;
C) Arrivée d'une requête pour la donnée  Di
     Si (il y a une copie valide de Di en cache) alors
        Envoyer Di au noeud demandeur
     Sinon
        Si (il y a un chemin valide menant à Di en cache)
        alors
            Acheminer la requête vers le noeud hôte de la copie
        Sinon
            Acheminer la requête vers la source initiale de Di
        Fsi
     Fsi
```

Dans cette méthode, lorsque le résultat de l'algorithme est l'exécution de *CacheData*, le chemin d'accès est sauvegardé en même temps. Car, il se peut que par la suite, la donnée soit remplacée. Le chemin permettra alors de retrouver la donnée. Après un remplacement de

donnée, on dispose toujours d'un chemin pour l'atteindre. C'est comme si *CacheData* se dégradait en *CachePath*.

3 - ***Méthode REDMAN***: [Bellavista 05a] [Bellavista 05b]

REDMAN (REplication in Dense MANets) est un système de réplication qui est une amélioration du système ***K_DID*** *("K hops Distance IRPs Dissémination"*). Ces systèmes sont fondés sur une approche basée sur la notion de *degré* de réplication sur un réseau dense. Le *degré* de réplication est défini comme le nombre nécessaire de copies pour assurer l'accessibilité. Cette approche considère un environnement relativement stable avec des données dont la consistance ne cause pas un problème majeur (pages web, fichier audio ou vidéo etc.). Le principe est de détecter les variations dynamiques de topologie dans le but d'assurer l'accessibilité par le maintien du degré de réplication et le placement optimal des copies sur le réseau. L'architecture du middleware se compose principalement de trois modules (figure 3.5): *DMC, RD* et *RR*.

3.1.3.1 *DMC* ("Dense MANET Configuration")

Ce module s'occupe de l'identification de nouveaux noeuds dans le réseau et de l'élection décentralisée des nœuds : un noeud est élu *Délégué* ou *Manager*. Un *Délégué* héberge des copies et met en oeuvre un processus de découverte de données par les autres noeuds. Il sert les requêtes d'accès aux données qu'il héberge par l'envoi d'une copie de la donnée. Un *Manager* a pour rôle de contrôler et de maintenir le degré de réplication dans une région dense du réseau.

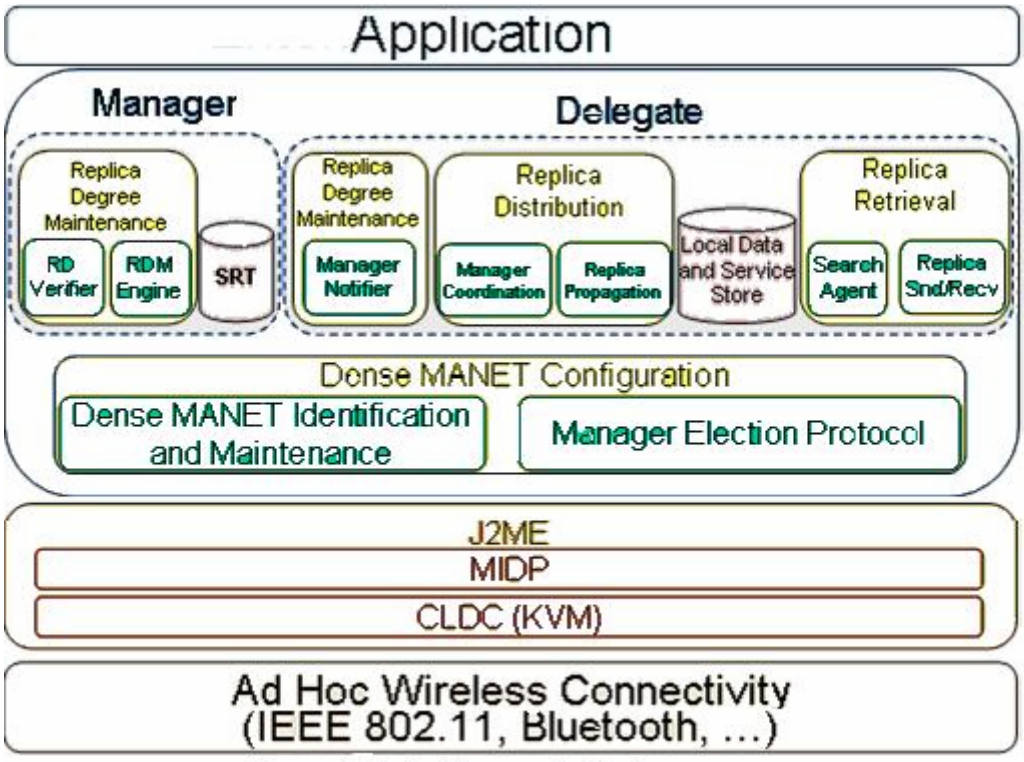

Figure 3.5: Architecture de Redman

3.1.3.2 *RD* ("Replica Distribution")

Ce module est responsable de la dissémination des copies dans le réseau. Quand un *Délégué*, hébergeant une donnée partagée, entre dans une région dense, il transmet le descripteur RDF ("Resource Descriptor Framework") de la donnée ou ressource qu'il souhaite partager au *Manager* en charge de la maintenance du degré de réplication dans cette région.

Sur la base des informations contenues dans le RDF et d'autres paramètres comme le nombre de noeuds de la zone, le *Manager* détermine le degré de réplication à maintenir pour la nouvelle ressource partagée. Il charge ensuite le *Délégué* du processus de dissémination.

L'idée principale de ce processus est de disséminer la ressource suivant une direction constante sur des noeuds situés à *k* sauts les uns des autres. Concrètement, cela se fait par l'envoi d'un paquet de réplication qui contient:

- Une copie de la source à disséminer
- La distance *k* entre deux copies consécutives de la direction constante.
- Le degré de réplication courant qui est la différence entre le degré de réplication initiale et le nombre de copies déjà disséminées.

La direction constante n'est pas maintenue par l'utilisation d'un GPS, mais par une heuristique où, chaque noeud choisit son successeur sur la direction constante en désignant, parmi ses voisins à un saut, le noeud qui a le moins de voisins communs avec lui. Plus la densité est uniforme sur la surface couverte par la région dense, plus la méthode s'avère efficace.

Afin de maintenir le degré de réplication, le mouvement des *Délégués* est contrôlé. Lorsqu'un délégué doit quitter une région dense, il transmet sa copie à l'un des voisins. Si par contre, un *Délégué* détecte qu'il a déjà quitter une région dense, il en informe le *Manager* qui procède à l'élection d'un nouveau *Délégué*.

En cas de panne subite où le *Délégué* est déconnecté brusquement, une incohérence du degré de réplication est tolérée pour un intervalle de temps relativement limité. Une telle déconnexion est rapidement perçue par le *Manager*, car, les délégués lui envoient périodiquement des messages de contrôle pour signaler leur présence dans la région dense.

3.1.3.3 RR ("Replica Retrieval")

Pour permettre une localisation rapide des copies, la notion de dissémination d'IRPs (Information about Ressource Placement) est utilisée. La mobilité des *Délégués* peut invalider les IRPs. Deux stratégies ont alors été proposées: IRP Flooding et Query Flooding.

A – *IRPF* ("IRPs Flooding")

Cette méthode consiste, pour chaque délégué de ressource, à inonder son voisinage d'IRPs. L'objectif étant de faire en sorte que chaque noeud puisse disposer des IRPs de toutes les ressources partagées. Ce qui n'est évidemment pas pratique pour les trois raisons qui suivent :

- Dans un environnement très dense, la sauvegarde de tous les IRPs provoque une utilisation abusive des mémoires de réplication
- L'inondation engendre la surcharge du réseau
- La mise à jour des IRPs devient très complexe avec la fréquence de la mobilité.

B – QF ("Query Flooding")

Dans cette méthode, les IRPs ne sont pas disséminés mais l'inondation est utilisée pour la recherche des copies. Ce qui peut devenir fastidieux dans un environnement où les requêtes sont assez fréquentes (surcharge du réseau, temps de réponse élevé etc.).

Les IRPs relatives à une même ressource doivent être situées à au moins k sauts des autres IRPs de la même copie. Chaque noeud doit connaître, parmi ses voisins situés à au moins k sauts, ceux qui sont *Délégués* ou détenteurs d'IRPs. Un *Délégué* informe son voisinage qu'il détient une copie par inondation. Il détermine par ce message les distributeurs d'IRPs à k sauts. Chaque distributeur va procéder de la même manière.

Bien que réduisant la recherche à une moyenne de k/2 sauts, cette méthode s'adapte difficilement à une mobilité relativement forte. De plus l'algorithme de dissémination est relativement complexe. Cette complexité est surtout due au nombre de messages diffusés. Aussi, le temps nécessaire pour la dissémination totale des IRPs n'est pas très raisonnable.

La technique de réplication *REDMAN* est en fait une amélioration de la solution précédente proposée pour le système *K_DID*. L'objectif de *REDMAN* est de mettre en oeuvre une dissémination moins complexe. La dissémination des copies se fait selon une direction relativement constante. Un *Délégué* transmet la copie qu'il détient vers des voisins choisis aléatoirement jusqu'à ce que le degré de réplication soit atteint. Les chemins d'accès IRPs et les copies sont disséminés en même temps et sur une même trajectoire. Les IRPs sont placées le long de la direction constante entre deux *Délégués* successifs. Tout se passe comme si les copies et les IRPs formaient une ligne traversant le réseau d'une extrémité à l'autre. Un client qui désire accéder à une information exploite cette disposition et utilise le même principe. Il dirige sa requête suivant une direction constante jusqu'à ce qu'elle croise la ligne portant les IRPs de la copie recherchée. Au bout d'un certain délai, si la requête n'aboutit pas, une autre direction est explorée (figure 3.6).

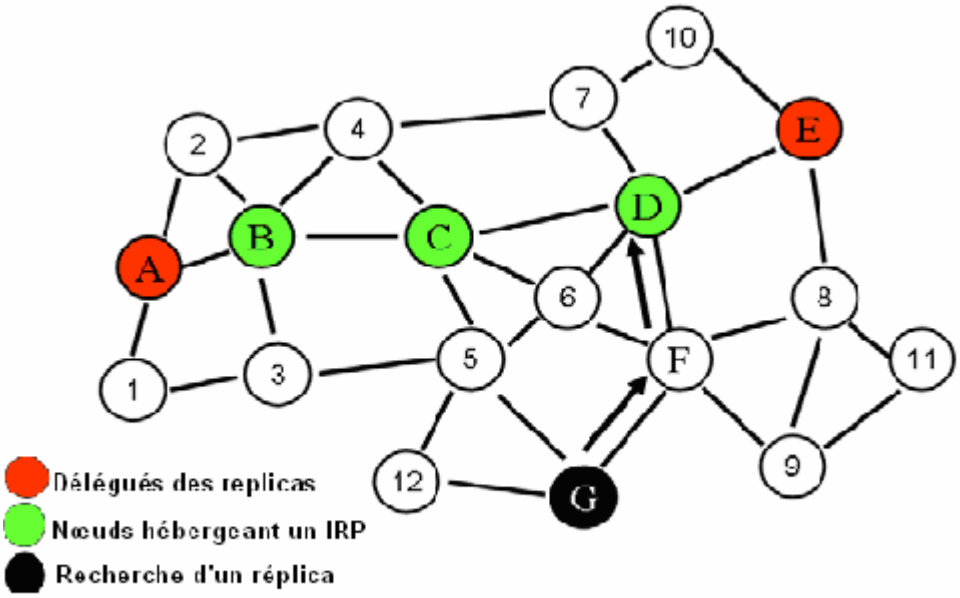

Figure 3.6: Exploitation directionnelle approximative d'une requête pour l'accès

Cette technique à l'avantage d'être simple et moins gourmande en mémoire de réplication. Cependant les temps de recherche sont rallongés. Un autre problème auquel elle doit faire face est la stabilité de la ligne portant les copies et les IRPs. Ce problème est géré par le DMC (Dense MANET Configuration). Lorsqu'un nœud de la ligne se rend compte du déplacement de son successeur ou de son prédécesseur, il diffuse localement un message de reconstruction. Tous les noeuds qui reçoivent les deux messages de reconstruction, celui du prédécesseur et celui du successeur, à un instant donné réagissent. En envoyant un message au prédécesseur seulement, ce dernier choisit l'un d'entre eux pour rétablir le tronçon remis en cause. Dans le cas où il n'existe pas de remplaçant potentiel, le prédécesseur ne reçoit pas de réponse. Il en déduit que plusieurs éléments de la ligne se sont déplacés en même temps. Il lance alors une nouvelle dissémination d'IRPs.

Cette technique ne vise pas une consistance stricte de la ligne d'IRPs, mais plutôt une reconstruction simple de la ligne à chaque fois qu'elle risque de se briser. Les méthodes proposées pour la dissémination des copies et des IRPs par [Bellavista 05a] [Bellavista 05b] sont assez intéressantes, car elles essayent de limiter le nombre de copies d'une même donnée, dans une région dense du réseau. D'où, une utilisation raisonnable des ressources. Cependant, aucune stratégie n'est définie pour déterminer le *degré* de réplication à maintenir. C'est pourtant l'un des paramètres les plus importants portant sur l'efficacité de la méthode. D'autre part, la densité d'un réseau n'est pas toujours assurée. Enfin, la définition d'une trajectoire pour offrir l'accès à une donnée est une idée intéressante mais elle ne peut être efficace qu'en cas de mobilité limitée. Une mobilité importante remettrait constamment toute détermination d'un chemin. D'où une complexité importante inutile.

4 - Méthodes *DCG* :[Hara 01]

DCG (Dynamic Connectivity based Grouping) est une solution étendue des méthodes ***SAF*** et ***DAFN*** [Hara 01] [Hara 06]. Elle est basée sur la construction de groupes stables de

noeuds bi-connectés [Aho 74]. Tout nœud d'un même groupe est connecté à au moins deux nœuds. Elle prend en considération une connaissance de la configuration complète de la topologie du réseau.

Contrairement à la méthode ***DAFN*** qui considère le voisinage d'un noeud, celle-ci organise le réseau en groupes de nœuds stables bi-connectés. Les fréquences d'accès sont évaluées pour les nœuds individuellement et aussi pour chaque groupe. La fréquence d'accès à une donnée par un groupe est la somme des fréquences d'accès à cette donnée par les nœuds du groupe. Les données sont allouées dans un groupe dans l'ordre décroissant de leur fréquence d'accès dans le groupe. C'est l'intérêt du groupe qui est privilégié sur l'intérêt du nœud seul. Une donnée qui doit être allouée dans un groupe est placée sur le nœud du groupe ayant la plus grande fréquence d'accès à cette donnée. Cette opération est renouvelée jusqu'à épuisement de l'espace mémoire du groupe. Un exemple est donné par la table 3.3 et la figure 3.7. Cependant, si un nœud n'est pas connecté à un nœud disposant d'une copie, l'allocation de la copie devient au moins temporairement impossible. L'espace réservé à la donnée demandée sera provisoirement attribué à une autre donnée jusqu'à ce que la connexion soit établie.

Donnée	Mobiles						Groupes	
	M_1	M_2	M_3	M_4	M_5	M_6	G_1	G_2
D_1	0.65	0.25	0.17	0.22	0.31	0.24	1.29	0.55
D_2	0.44	0.62	0.41	0.40	0.42	0.46	1.87	0.88
D_3	0.35	0.44	0.50	0.25	0.45	0.37	1.54	0.82
D_4	0.31	0.15	0.10	0.60	0.09	0.10	1.16	0.19
D_5	0.51	0.41	0.43	0.38	0.71	0.20	1.73	0.91
D_6	0.08	0.07	0.05	0.15	0.20	0.62	0.35	0.82
D_7	0.38	0.32	0.37	0.33	0.40	0.32	1.40	0.72
D_8	0.22	0.33	0.21	0.23	0.24	0.17	0.99	0.41
D_9	0.18	0.16	0.19	0.17	0.24	0.21	0.70	0.45
D_{10}	0.09	0.08	0.06	0.11	0.12	0.09	0.34	0.21

Table 3.3: Exemple de fréquences d'accès de groupes

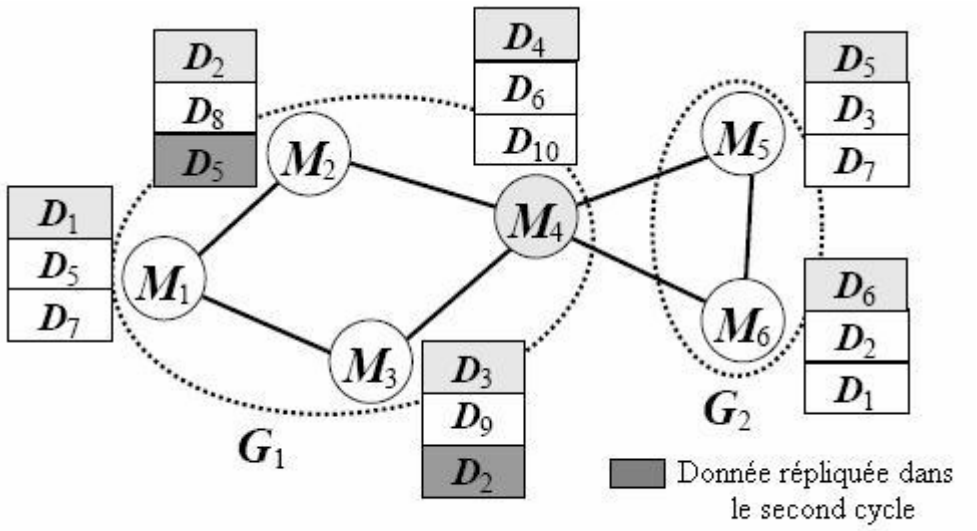

Figure 3.7: Exemple d'exécution de DCG

Comparée aux deux autres méthodes ***SAF*** et ***DAFN***, ***DCG*** est exécutée en plusieurs étapes:

1. reconnaissance de la topologie actuelle,
2. détermination des noeuds pouvant former des groupes,
3. détermination des copies à allouer,
4. transmission des ordres de réplication aux nœuds.

Durant le temps d'exécution de l'algorithme, la topologie du réseau peut avoir changer. Cette remarque est la cause de la proposition suivante.

3.1.4.1 La méthode *DCG-S1*: [Hara 03a]

DCG-S1 se base fondamentalement sur l'algorithme ***DCG*** en considérant uniquement les liens stables dans la construction des groupes bi-connectés. ***DCG-S1*** fait les mêmes hypothèses que ***DAFN-S1*** (modèle RWP, GPS,…). La stabilité d'un lien est définie comme suit:

- A chaque période de réallocation de données, les nœuds effectuent une diffusion de leur identificateur. Grâce à ces informations le temps t_{ij} restant avant la déconnexion de deux nœuds peut être calculé en comparant leurs vitesses et leur directions. Ce temps t_{ij} permet d'évaluer la stabilité B_{ij} du lien pendant la période **T** de réallocation courante (voir ***DAFN-S1***).
- Pour chaque ensemble de noeuds mobiles et en commençant par le noeud ayant le plus petit identificateur une recherche des bi -connexions est effectuée. Une bi-connexion est considérée comme stable si B_{ij} est supérieure à un certain seuil *x*.
- La procédure de réplication des données dans le groupe est alors celle de ***DCG***.

DCG-S1 n'est pas un algorithme *glouton* ; il assure une réplication selon les besoins d'un ensemble de nœuds formant un groupe. Par conséquent, dans un voisinage de nœuds constituant un groupe, le nombre de données différentes accessibles est plus grand. Aussi, la condition de stabilité des liens augmente la sûreté et la fiabilité du partage. Cependant la détection des liens stables reste dépendante d'hypothèses pas très réalistes (connaissance de la destination d'un nœud, connaissance de la topologie complète pour la détermination des groupes, …). Cette solution offre une meilleure accessibilité aux données mais, la construction des groupes et la détection des liens stables engendrent une complexité relativement importante en messages. La figure 3.8 nous donne un exemple d'application de cette méthode en considérant les fréquences des noeuds de la table 3.3.

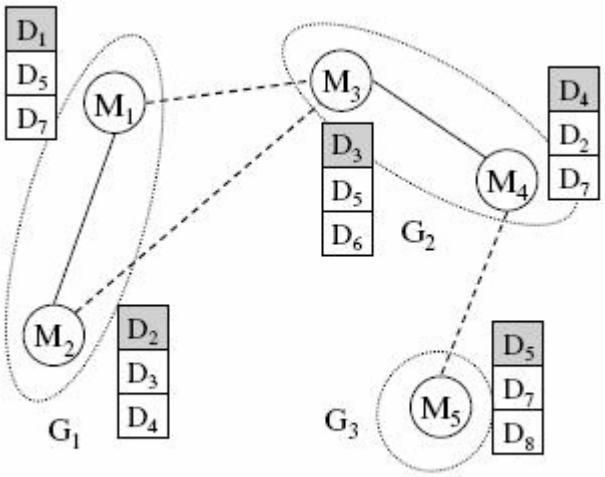

Figure 3.8: Exemple d'exécution de ***DCG-S1***

3.1.4.2 La méthode *E-DCG* [Hara 03b]:

E-DCG (Extended-Dynamic Connectivity based Grouping) est définie sur la base de composition de groupes bi-connectés. Cette bi-connexion assure le groupe contre le partitionnement dans le cas où un seul des nœuds disparaîtrait ou si une des liaisons se romprait.

Tout comme ***E-SAF*** et ***E-DAFN***, cette méthode considère le paramètre *PT* défini dans les paragraphes précédents. L'algorithme de base appliqué est le même que celui de ***DCG*** mis à part que, dans le choix des allocations de copies, la fréquence d'accès d'une donnée est remplacée par *PT*. Il s'agit toujours d'un algorithme périodique. En début de chaque période, les nœuds diffusent les informations sur leurs données. Ainsi, chaque nœud peut connaître les nœuds auxquels il est connecté.

Chaque ensemble de nœuds formant un circuit dans le graphe modélisant le réseau représente un groupe bi connecté. Un nœud peut appartenir à plusieurs circuits simultanément. Pour régler ce conflit, un nœud se rattache au premier circuit qu'il identifie. La fréquence d'accès du groupe à chaque donnée est calculée en additionnant les fréquences d'accès de tous les membres du groupe. A partir de ces fréquences, le facteur *PT* du groupe est évalué pour chaque donnée. Les *PTs* sont triés dans un ordre décroissant par le nœud du groupe qui a le plus petit identifiant. Il détermine pour chaque nœud de son groupe, la liste des données qu'il doit répliquer dans sa mémoire et il la lui transmet. Une copie est allouée sur le nœud où son facteur *PT* est plus élevé des nœuds du groupe. Une donnée originale présente sur l'un des nœuds du groupe n'est pas dupliquée. L'ensemble des données du groupe et leur localisation sont ensuite transmis aux membres du groupe afin d'optimiser les accès ultérieurs. Les performances obtenues par ***E-DCG*** en accessibilité et en trafic sont meilleures que celles obtenues par ***DCG*** [Hara 03b].

3.1.4.3 La méthode *E-DCG*$^+$: [Hara 06]

Cette méthode est une autre amélioration de ***DCG*** en considérant les mêmes hypothèses que ***E-DCG***. L'algorithme est aussi similaire à celui de ***E-DCG***, à part que le critère de réplication est le rapport *RWR* des fréquences de lecture et d'écriture. Toute fois les données importantes "*open objects*" sont prioritaires pour le processus de réplication.

E-DCG$^+$ offre une bonne disponibilité des données. Elle présente les même inconvénients que les méthodes sur lesquelles elle est fondée (*DCG*, *E-DCG*, ..) dont les hypothèses contraignantes sur le déplacement et les vitesses des nœuds. Ces deux informations sont imprévisibles et très variables. Aussi, la condition de regroupement des nœuds est très forte. Elle a pour conséquence la formation de beaucoup de groupes *singletons*. La construction de ces groupes perd tout son intérêt si une majorité de groupes se retrouvent composés d'un nœud mobile unique.

5 - *Système AdhocFS (Ad hoc File System)*: [Boulekenafed 03]

AdhocFS est un système de gestion de fichiers pour les réseaux ad hoc avec présence d'un serveur primaire fixe. Le mécanisme de partage de fichiers et de réplication construit des groupes de nœuds connexes (complètement connectés). Ces groupes sont reconstruits périodiquement pour considérer les changements dynamiques de la topologie. Tous les fichiers partagés dans un groupe sont mentionnés sur un répertoire virtuel. Cette structure mentionne chaque fichier local et distant ainsi que sa localisation (figure 3.9).

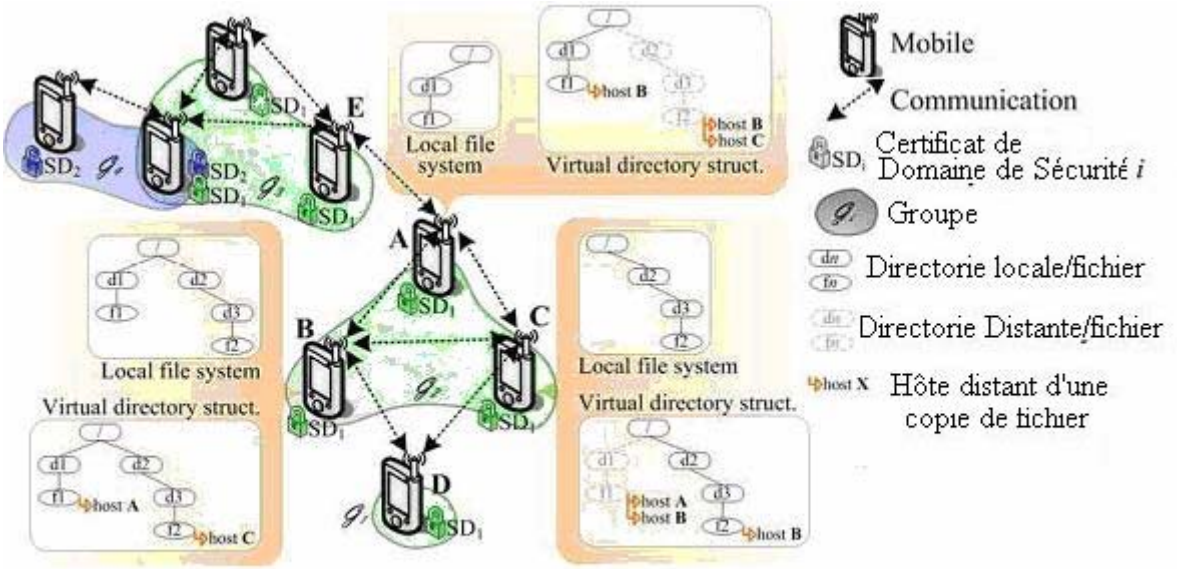

Figure 3.9: Groupes AdhocFS et structures de directories virtuelles [Boulkenafed 03]

AdhocFS considère l'existence d'un serveur fixe appelé "*Home Server*" et un ensemble de groupes de mobiles nommés "*Ad hoc groups*". Toutes les demandes des noeuds mobiles sont destinées au serveur s'ils sont connectés. Une fois déconnectés de ce dernier, les noeuds d'un groupe mobile ad hoc se partagent les copies dont ils disposent. Les groupes définis par AdhocFS sont constitués de nœuds à un saut. Le système vérifie dynamiquement les modifications des groupes ad hoc ; c'est-à-dire, les déconnexions de nœuds du groupe et les nouvelles connexions.

Tous les nœuds d'un groupe mobile AdhocFS ont connaissance des données stockées sur les nœuds du groupe et de l'état de leurs ressources. Ils peuvent donc identifier les noeuds mobiles qui peuvent participer à l'accroissement de la disponibilité des données au niveau du groupe. A chaque donnée est associée une méta-donnée. Une méta-donnée est constituée d'un ensemble d'information de description d'une donnée. L'algorithme d'allocation de copies utilise deux types de profils.

3.1.5.1 Profil d'une entité mobile:

Un nœud communique son profil au leader du groupe pendant l'étape d'échange des méta-données. Ce profil indique:

(i) l'espace de stockage disponible pour le partage;
(ii) la possibilité de stocker des copies de données, autre que les copies nécessaires pour l'accès local, en fonction de l'énergie disponible;
(iii) le temps prévu avant de quitter le groupe, déterminé grâce aux indications de l'utilisateur.

Trois paramètres peuvent être identifiés comme des paramètres qui contribuent à réduire la disponibilité des données, soit en favorisant les déconnexions, soit en limitant la réplication :

(i) La durée prévue dans le groupe: Cette durée est estimée en fonction des indications de l'utilisateur, par exemple son emploi du temps. Son évolution est représentée par la fonction linéaire décroissante suivante :

$$Durée\ (t) = cst - t$$

Où, *cst* est la durée estimée, pendant laquelle l'utilisateur est supposé être connecté au groupe de travail. *Durée*(t) permet de générer des copies, si la déconnexion volontaire de l'utilisateur est proche.

(ii) Le manque d'énergie: l'énergie est calculée à partir des fonctions système à travers des interfaces de type ACPI (Advanced Configuration and Power Interface) qui permettent la configuration et la gestion de l'énergie des entités mobiles à partir du système d'exploitation. Elle est définie par la fonction linéaire décroissante suivante:

$$Energie\ (t) = init - t \times (init / batterie)$$

Où, *t* est la variable temps, *init* est l'autonomie initiale de la batterie au moment de l'initialisation du groupe, et *batterie* est une estimation de la durée de vie de la batterie définie en fonction de la charge de l'entité mobile.

(iii) L'insuffisance de l'espace de stockage: Un espace de stockage est représenté par une variable dont la valeur dépend du taux de réplication. Cette variable doit être recalculée périodiquement. Elle permet de déterminer si l'entité mobile peut être sollicitée pour stocker une nouvelle copie préventive.

AdhocFS évalue ces paramètres périodiquement afin d'anticiper les déconnexions et d'adapter la réplication des données qui pourraient devenir inaccessibles. La notion de profil d'une entité mobile est définie par le regroupement de ces paramètres. Selon son profil, un nœud peut être dans l'un des trois états:

- *Optimal*: Le noeud mobile est apte à collaborer activement et à partager ses ressources avec les autres entités mobiles du groupe.
- *Acceptable*: Le nœud mobile est capable de partager ses ressources avec les autres noeuds du groupe, mais ses aptitudes au partage sont moins intéressantes que ceux des nœuds mobiles dont le profil est *Optimal*.
- *Faible*: Le nœud mobile dispose de peu de ressources. Par conséquent, il ne peut pas les partager avec les autres noeuds du groupe.

3.1.5.2 Profil d'un groupe mobile AdhocFS:

Le profil d'un groupe mobile AdhocFS est défini par l'ensemble des profils de ses noeuds. Il est mis à jour périodiquement, par le leader du groupe en même temps que l'échange des méta-données. Le profil d'un groupe, à l'instar des autres méta-données, est propagé vers tous les nœuds du groupe. Chaque noeud a une vision globale des ressources disponibles au niveau de son groupe.

La technique de réplication différencie les copies de travail *RT* ("Replicas" de Travail) des copies préventives *RP* ("Replicas" Préventifs). Au niveau de chaque groupe, et pour chaque donnée utilisée, il existe au moins une copie maintenue à jour. Les copies préventives sont créées lorsque les copies *RT*s existantes n'assurent plus la disponibilité de la donnée. Elles permettent ainsi d'anticiper de futures déconnexions qui peuvent compromettre la disponibilité des données.

6 - Méthode pour la continuité de service: [Chen 02]

Cette approche vise à optimiser la qualité du service d'accès aux données partagées par un groupe de nœuds. Les deux couches routage et middleware sont mises à contribution dans le

but d'améliorer la qualité du service d'accès aux données. L'accessibilité est définie comme étant la possibilité pour un nœud d'accéder à une donnée lorsqu'il la requiert et sa capacité à obtenir des informations sur les données à partir d'autres nœuds. Un groupe d'accès aux données est décrit comme étant tout ensemble de nœuds communicants qui se partage un espace mémoire commun. Pour éviter la redondance, chaque groupe tente de détenir un exemplaire unique de chaque donnée.

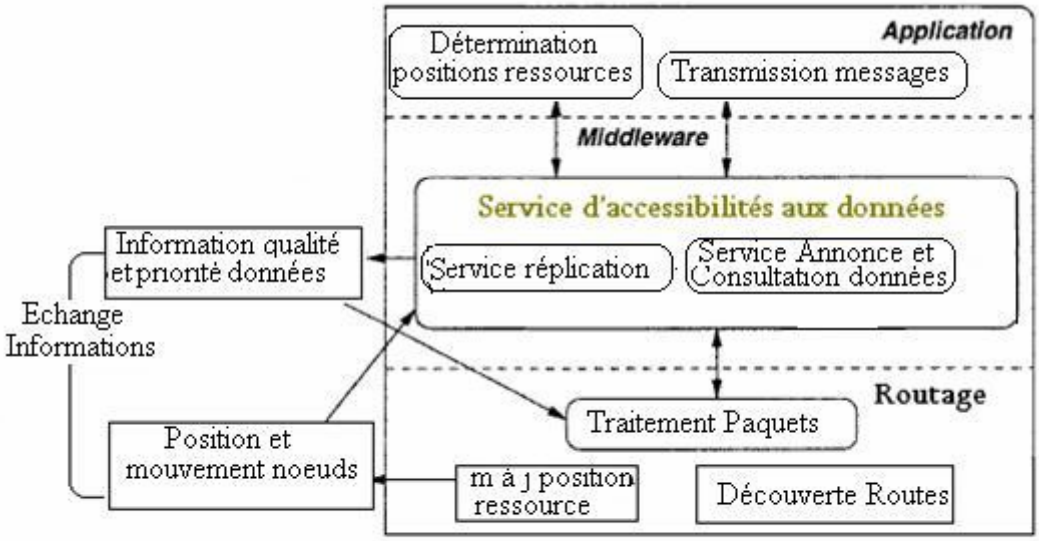

Figure 3.10: Composantes du système de continuité d'accès

Cette proposition suppose que les nœuds ont tous accès à un GPS qui définit avec précision leur position. Les différentes horloges des nœuds sont synchronisées par ce même GPS. Le service d'accès aux données se compose de deux sous services (figure 3.10) : le service d'annonce et de consultation de données, et le service de réplication.

3.1.6.1 Service d'annonce et de consultation:

Ce service entretient, pour chaque nœud, une table de description des données disponibles dans son voisinage grâce aux diffusions périodiques du profil du nœud et des annonces:

i. Le profil est composé des informations suivantes: identité du noeud, une estampille, la position (deux coordonnées), la direction, la vitesse, la bande radio, l'énergie disponible, et l'utilisation CPU. Si l'estampille d'un noeud n'est pas mise à jour au bout d'un certain délai, le nœud est considéré déconnecté. La capacité d'un nœud est calculée par la somme des trois valeurs suivantes: Espace disponible, énergie restante, et utilisation CPU.

ii. Une annonce ou un message ***Ad*** ("Advertising message") est une liste de description des données disponibles sur le nœud. Ainsi, chaque nœud peut construire une table des données disponibles dans son voisinage (Table 3.4).

Adresse du nœud expéditeur		
Num-seq	Espace libre	
Id-donnée	Time-stamp	description
Id-donnée	Time-stamp	description
Etc…	Etc…	Etc…

Table 3.4 : Contenu d'un message ***Ad***

Un message ***Ad*** ("Advertising") comporte un numéro séquentiel Num-seq, l'espace disponible sur le nœud, et la liste des données disponibles localement. Chaque donnée est représentée dans le message par son identité Id-donnée, l'estampille de sa création ou de sa dernière mise à jour et une description de la donnée. Les données sont supposées se prêter à une description avec un langage de description de données quelconque tel que XML.

Le service d'annonce et de consultation est responsable du traitement des messages ***Ad***. Il partage aussi avec la couche routage un certain nombre d'informations sur les données comme: Le niveau d'importance et la qualité des données (figure 3.11).

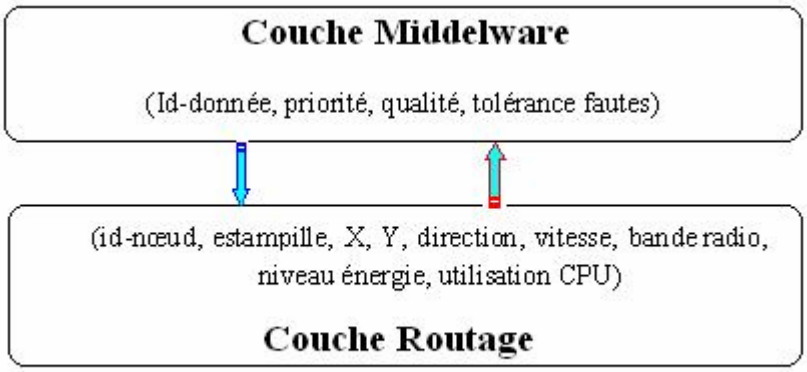

Figure 3.11: Echange d'information

Le niveau middleware fournit à la couche routage des informations sur la priorité des données sous forme d'un profil partagé de données (identité de la donnée, priorité de la donnée, niveau de qualité de service requis pour la donnée et, le degré de tolérance aux fautes de cette donnée).

La priorité des données est transmise au niveau des paquets de routage correspondants. Le niveau de tolérance aux fautes est considéré par le routage pour évaluer le degré de tolérance aux erreurs nécessaire à la transmission des paquets correspondants.

Pour réduire le trafic induit par les messages *Ad*:

- Entre l'envoi de deux messages *Ad* complets, des messages plus courts, ne contenant que les différences avec le dernier message *Ad* complet, sont envoyés. Ces messages ne comportent que l'adresse de l'expéditeur et les identificateurs des données modifiées (ajout, suppression,..) (Table 3.5).

- Au court d'une période d'observation, les messages en circulation ne doivent pas dépasser un seuil maximal défini en nombre de messages. Chaque nœud mémorise le nombre ***m*** de messages reçus durant le temps d'observation courant. A l'émission d'un message *Ad* par le nœud, il doit en premier comparer ***m*** avec un seuil ***n***. Si ce dernier est encore supérieur a ***m*** alors ce nœud va envoyer son message avec une probabilité qui est égale à 1, sinon si ***m*** est plus grand que $k \times n$ ($k > 1$) le message ne sera pas envoyé, sinon il sera envoyé avec une probabilité comprise entre 1 et 0.

Adresse du nœud expéditeur		
Num-seq	Espace libre	
Id-donnée(supprimée)	Id-donnée(supprimée)	
Etc.	Etc.	
Id-donnée	Time-stamp	description
Etc.	Etc.	Etc.

Table 3.5: Message *"Ad Differential*"

Un message *Ad* est traité comme suit par un nœud récepteur:

- Le récepteur compare sa capacité à celle de l'émetteur. Si ce dernier a une meilleure capacité, alors le récepteur détruit la copie locale de la donnée.
- Il met à jour la table des données selon les informations reçues (suppression, ajout, etc.). Une donnée non mise à jour depuis un temps t supérieure à un seuil, est supposée inatteignable.

3.1.6.2 Service de réplication:

L'objectif principal de ce service est de détecter un éventuel partitionnement afin de réaliser une réplication prédictive. Il utilise les informations fournies par le service d'annonce et, les informations de la couche routage sur la position des nœuds et leurs mouvements. En cas de détection d'un partitionnement, le protocole de réplication va dupliquer les données d'une partition sur l'autre selon l'ordre décroissant des fréquences d'accès. Les données d'un groupe (par exemple, G1 sur la figure 3.12) sont répliquées sur l'autre groupe et réciproquement. En cas de manque d'espace, les données les moins importantes (en fréquence d'accès) sont remplacées.

La détection des déconnexions fait appel à une hypothèse supplémentaire sur le mouvement et la localisation des nœuds en les considérant prévisibles. Le niveau routage se charge de la prédiction des positions et de la qualité de service du routage.

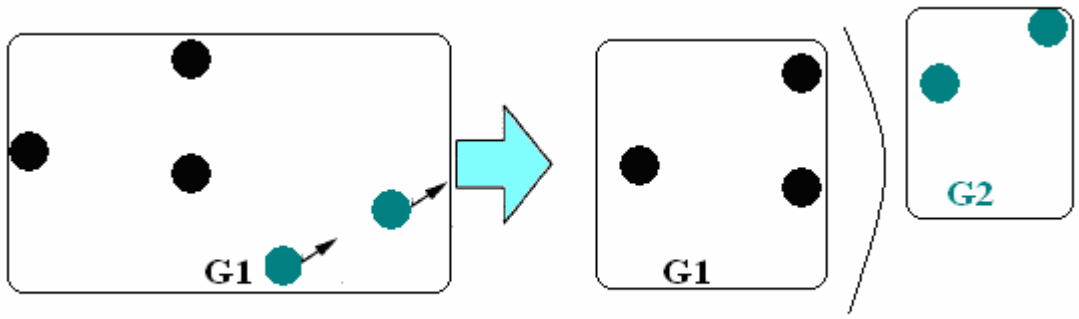

Figure 3.12: Partitionnement d'un groupe

Pour assurer cette prédiction, les données sur les positions et les mouvements d'un nœud sont transmises périodiquement. Elles peuvent aussi être transmises instantanément, en cas de nécessitée; si la position actuelle est déviée de plus d'une certaine valeur δ par rapport à la position prédite. Pour cela un nœud doit périodiquement déterminer sa position. Les informations sur la position d'un nœud sont diffusées sur tout le réseau afin que chaque nœud puisse construire une vue globale et complète de toute la topologie. Ce service de prédiction est utilisé pour assurer une qualité du service du routage. Chaque nœud détient une table de description de tous les autres nœuds du réseau.

Les évaluations des auteurs de [Chen 02] montrent une amélioration de l'accessibilité. Toutefois le taux d'accessibilité ne dépasse jamais les 50%. Le coût d'entretient des tables de données et de la topologie semble important vu que chaque nœud doit avoir une connaissance complète du réseau. Le passage à l'échelle de cette approche semble non évident. Ce coût ne peut être justifié que dans le cas où l'accès aux données est crucial pour assurer la continuité de service. L'accessibilité peut être remise en cause en cas d'une déconnexion subite ou d'une mauvaise prédiction. Donc, cette solution ne considère pas les pannes et les déconnexions volontaires. Aussi, les hypothèses sur la connaissance des positions et surtout sur l'orientation des déplacements ne sont pas réalistes. D'autre part, est–il justifié d'éliminer une copie s'il n'y a pas de manque d'espace?

7 - Synthèse et critiques

Nous pouvons classer, d'après cette étude, les méthodes de réplication pour les réseaux mobiles ad hoc en trois groupes:

- La réplication individuelle (méthodes gloutonnes), où chaque noeud réplique ses propres besoins en données.
- La réplication par groupe, où les nœuds géographiquement proches coopèrent à répliquer un maximum de données partagées.
- La réplication prédictive à base de prédiction des partitionnements avec recours à l'utilisation d'un GPS.

Les trois méthodes de base proposées par Hara ***SAF***, ***DAFN*** et ***DCG*** adoptent un processus de réplication périodique basée sur les fréquences d'accès. Afin de réduire la redondance des copies identiques, les deux dernières examinent la topologie courante du réseau pour décider des réplications à effectuer. En cas de changement de localisation d'un nœud, sa nouvelle position géographique est considérée à la prochaine période. Nous constatons que: si deux nœuds formant un groupe bi-connecté tel que défini par ***DCG***, et si les deux nœuds accèdent à une même donnée avec des fréquences d'accès importantes, l'algorithme d'élimination des copies redondantes va procéder à la suppression de la donnée à fréquence d'accès inférieure, même si elle est très sollicité par son nœud hôte.
En plus du coût en communication entre les deux nœuds, de la consommation énergétique et du temps de calcul du serveur qui reste, une potentielle déconnexion est possible. Une première solution aurait été de définir un seuil minimal permettant le remplacement d'une donnée. Au dessus de ce seuil, la redondance est jugée utile vu l'importance de la donnée pour les nœuds hôtes des copies. Un nœud qui utilise beaucoup une donnée aura beaucoup moins besoin d'autres données. Il n'est pas toujours intéressant de la remplacer par une autre donnée à laquelle il accède moins souvent et le contraindre à transmettre des requêtes à distance.

L'approche de [Chen 01] est un protocole dont l'objectif est d'assurer une continuité de service par prédiction du partitionnement. Ce protocole n'est pas périodique. Il est déclenché pour une seule et unique raison, la détection d'un futur partitionnement du groupe. Chaque nœud qui détecte de futurs partitionnements réplique les données qu'il possède dans son cache sur les noeuds de l'autre partition. Ces données sont traitées par ordre décroissant des besoins des nœuds.

Cao et Yin [Yin 04] proposent de stocker des copies des données acheminées pour servir les requêtes futures. Cette vision améliore sensiblement la disponibilité, mais ne prévoit pas les partitionnements possibles qui peuvent isoler un serveur source d'une partie des noeuds. D'autant plus, qu'un partitionnement peut précéder les requêtes d'accès. Il présente aussi un mécanisme de mise en cache de chemins menant aux copies. La sauvegarde des chemins permet de diriger les futures requêtes et, optimise l'utilisation des espaces de réplication. Mais l'utilité de cette sauvegarde est limitée en cas mobilité forte où les chemins s'avèrent rapidement invalides.

L'avantage des méthodes proposées par [Yin 04] réside surtout dans leur simplicité. Elles ne nécessitent pas de calculs complexes. La mise en cache se fait dynamiquement suivant les besoins et l'utilisation d'informations fournies par les protocoles de routage facilite la mise en oeuvre. Il faut, cependant, noter que le choix des différents seuils est très déterminant pour les performances escomptées lors de la mise en oeuvre de cette dernière.

Le principal inconvénient de l'approche [Yin 04] est qu'elle ne tient pas compte du partitionnement. Aussi, elle ne se préoccupe pas de l'équilibrage de la charge d'accès sur le réseau.

Le système *REDMAN* utilise souvent des diffusions sources de goulot d'étranglement surtout qu'il considère des environnements denses. Sur un environnement dense la connectivité est importante et donc il d'agit certes de maintenir l'accessibilité mais surtout de définir les chemins optimaux afin de répondre dans les meilleurs délais aux requêtes. Le système *REDMAN* permet aussi d'éviter la surcharge des serveurs par un équilibrage de la charge. Il prévient le partitionnement du réseau en observant les déconnexions dans le voisinage de chaque serveur.

3.2 Méthodes de réplication avec mise à jour de données

Dans le domaine de l'informatique mobile, plusieurs stratégies de réplication on été proposées pour la gestion des mises à jour de données [Barbara 94], [Huang 94], [Jing 97], [Pitoura 95], [Wu 96]. La plupart d'entre elles supposent que les nœuds mobiles accèdent aux données sur des sites fixes.

Dans cette dernière partie du chapitre, nous allons étudier différentes stratégies de gestion de l'accès en mise à jour. Afin de mieux appréhender les problèmes de gestion de cohérence de copies multiples, cette étude considère les solutions conçues pour la prise en compte de la mobilité en général (avec ou sans infrastructures statiques). Ces systèmes diffèrent par leur modèle de communication qui varie du client/serveur strict au client/serveur étendu et peer-to-peer. Les solutions adoptées pour les réseaux mobiles ad hoc sont souvent des variantes de celle proposées pour les réseaux mobiles cellulaires.

3.2.1 Le système de fichiers CODA

CODA est un système de gestion de fichiers pour un environnement distribué à grande échelle qui a été adapté aux utilisateurs mobiles [Kistler 91] [Satyarayanan 02]. Il est basé sur le modèle de communication Client/Serveur strict. CODA apparaît à l'utilisateur comme un système de fichiers Unix partagé traditionnel. Il fait une distinction entre les sites serveurs, qui sont des machines physiques sûres, fiables et gérées par une équipe technique qualifiée, et les sites clients qui sont hébergés sur des machines dispersées et qui peuvent être mobiles et déconnectées pour de longues périodes.

Pour réaliser une meilleure disponibilité en présence des clients mobiles, CODA permet à des clients d'accéder aux fichiers stockés dans le cache local tout en étant déconnectés des serveurs. Le but d'un tel système est d'offrir aux clients la continuité d'accès aux données en cas d'éventuelles déconnexion ou de pannes. Il permet aux utilisateurs de fonctionner en mode déconnecté.

3.2.1.1 Stratégie de réplication:

La stratégie de réplication utilisée dans CODA est la suivante : un client obtient des données d'un membre de son ***AVSG*** (**A**vailable **V**olume **S**torage **G**roup) appelé *Preferred Server.* Le *Preferred Server* peut être choisi au hasard ou sur la base de critères d'exécution tels que la proximité, la charge du serveur, ou la puissance de calcul. Lorsque les données sont transférées au client les autres serveurs entrent en contact avec ce dernier pour vérifier que le *Preferred Server* a bien la dernière copie des données. Si ce n'est pas le cas, un membre de l'***AVSG*** ayant la dernière copie actualise ses données. Ensuite, il annonce à l'***AVSG*** que certains de ses membres ont des copies non mises à jour.

Cette approche est simple. Elle vise à augmenter la probabilité de présence de données à jour sur les divers hôtes possibles. La charge de propagation de données est attribuée au client. En effet, l'unité centrale d'un serveur est le point de surcharge critique dans beaucoup de systèmes de fichiers répartis. En les déchargeant, le système favorise le passage à l'échelle de la solution.

3.2.1.2 Gestion du cache du client:

Venus est un processus de CODA qui gère le cache local de chaque client. Lors d'un accès à un fichier du système. ***Venus*** se charge de charger le fichier. Il opère dans un des trois états : *Accumulation*, *émulation*, et *réintégration* (figure 3.13). Le processus Venus est généralement dans l'état d'*Accumulation*. Il se base sur le serveur pour les opérations sur les fichiers. Mais, il reste toujours en alerte en cas d'éventuelles déconnexions. Lors d'une déconnexion, il passe à l'état d'*émulation*. Dans cet état, toutes les opérations sur fichier (lecture/écriture) peuvent être servies localement. La stratégie de réplication et la gestion des mises à jour sont réalisées par une technique de réplication optimiste.

Lorsque le client est connecté au serveur, CODA se base sur le système de fichier ***AFS*** (*Andrew File System*) qui emploie un protocole pessimiste pour la gestion de la cohérence des copies. Si le client se déconnecte, un protocole optimiste est utilisé afin de permettre à ce dernier de lire et de mettre à jour la donnée locale. Les mises à jour seront effectives et ne seront envoyées aux autres clients que lors de la reconnexion du client au serveur.
Lors de la reconnexion, ***Venus*** passe à l'état de *réintégration* pour intégrer les modifications effectuées durant la déconnexion sur les autres copies.

Les mises à jour concurrentes sur les copies peuvent se produire après déconnexion provoquant des incohérences. Afin de résoudre ce problème, une approche basée sur les vecteurs de version est adoptée [Barreto 03]. Chaque fois qu'un client modifie un fichier, un message de mise à jour est envoyé à tous les serveurs le possédant. Ce message contient le nouveau contenu du fichier ainsi que le vecteur de version détenu par le client. Si le vecteur contenu dans le message est supérieur à celui qui du serveur, ce dernier met à jour le fichier

ainsi que le vecteur du serveur. Si le client est déconnecté, cette opération est retardée jusqu'à la phase de réintégration.

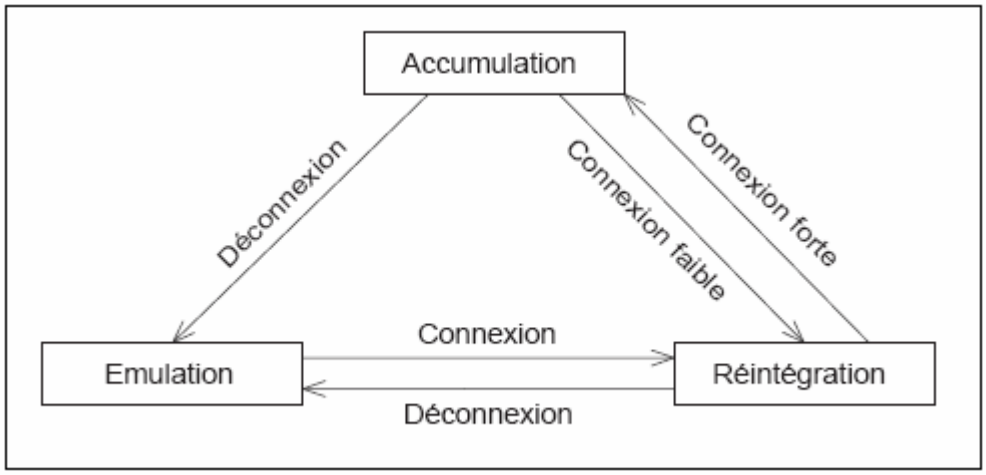

Figure 3.13 : Etats du processus ***Venus*** dans Coda

L'opération de déconnexion ainsi que la phase de réintégration sont acceptables sur des configurations où les déconnexions sont rares. Cependant, dans un réseau mobile ad hoc, les déconnexions sont fréquentes. La technique de réconciliation devient donc inadéquate vu que les mises à jour sont d'abord envoyées au serveur. Un nœud d'un réseau mobile ad hoc qui n'a pas accès au serveur agira comme un client isolé.

3.2.1.3 Détection et résolution des conflits:

Dans le système CODA, les clients déconnectés manipulent librement leurs fichiers locaux. Les mises à jour sont journalisées et transmises dès la reconnexion à un ensemble de serveurs répliqués (**AVGS**). La propagation des mises à jour entre les copies des sites clients est interdite. Ces manipulations peuvent produire des conflits lors de la réintégration des opérations concurrentes (par exemple, deux clients qui ont loué la même voiture pour une même journée). CODA propose des mécanismes de résolution automatique de certains conflits inhérents aux systèmes de fichiers, tels que les conflits suppression/modification de fichier. Mais, lorsqu'un autre type de conflit apparaît, sa résolution est à la charge de l'utilisateur.

3.2.2 Système de réplication de fichiers ROAM

ROAM est un système de réplication de fichiers pour un environnement mobile. ROAM utilise le modèle *Ward* (Wide Area Réplication Domains) [Ratner 04]. Un *Ward* est une collection dynamique de machines géographiquement proches. Une de ces machines joue le rôle *Ward Master*. Un *Ward Master* peut être comparé à un serveur dans le modèle client/serveur. Toutefois, quelques différences sont à noter:

- ✓ Chaque membre dans un *Ward* est un noeud selon le modèle *peer-to-peer*. Il peut donc se réconcilier avec n'importe quel noeud (le modèle client/server n'autorise pas de communication client/client).
- ✓ Tous les membres d'un *Ward* peuvent tenir le rôle de *Ward Master.* Des mécanismes d'élection et de reconfiguration sont mis en place si le *Ward Master* est défaillant. Dans les modèles *peer-to-peer* classiques, chaque noeud connaît l'existence de tous les autres. Dans ce cas, le *Ward Master* est le seul lien entre les *Wards* (figure 3.14). Il est le point d'accès unique permettant de localiser toutes les copies. Il peut donc appliquer un algorithme de cohérence à un niveau supérieur, entre les *Wards* uniquement. Si nous définissons toutes les copies d'un Ward par l'abstraction copie du Ward, une stratégie de cohérence adaptée assure la cohérence à un niveau supérieure entre les copies du *Ward*.

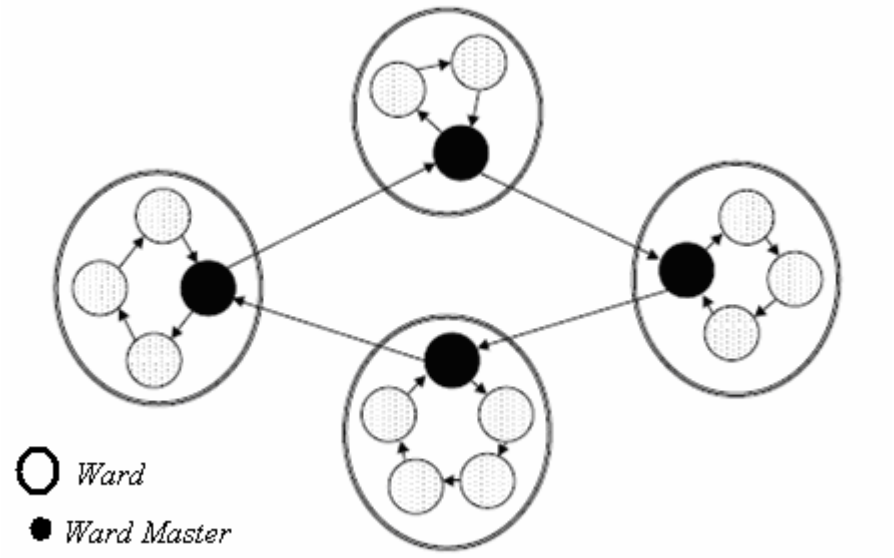

Figure 3.14 : configuration des ***Wards*** dans ROAM.

Dans ROAM, les copies sont crées lors des accès effectifs. La granularité de la réplication dans ROAM est le volume de données, qui peut être partitionné pour une granularité plus fine dans un modèle de ***Ward*** étendu. Chaque ***Ward*** dispose d'un processus chargé des mises à jour et de la synchronisation avec les autres ***Wards***. L'allocation d'une nouvelle copie d'une donnée dans un ***Ward*** est à la charge de l'utilisateur.

Dans le cas de mobilité d'une des machines appartenant à un ***Ward***, ROAM ne prévoit pas de prise en compte automatique. La machine continue de se synchroniser avec le gestionnaire de son ***Ward*** initial. Même si la liaison devient coûteuse. L'utilisateur peut demander explicitement une déconnexion de son ***Ward*** initial et une affection à un nouveau Ward plus proche géographiquement.

ROAM est une solution de réplication pour le calcul mobile. Par une réplication optimiste inspirée du modèle peer to peer, il permet à n'importe quelle copie de communiquer et de se synchroniser directement avec n'importe quelle autre copie.

3.2.3 Système de gestion de bases de données BAYOU

BAYOU est un système de réplication optimiste à cohérence faible. Il a été conçu pour supporter des applications de base de données collaboratives dans un environnement mobile. Son objectif est de fournir des mécanismes permettant aux clients nomades de lire ou d'écrire des données partagées.

BAYOU est basé sur un modèle client/serveur étendu. Pour permettre la détection de conflits, les applications doivent indiquer des conditions qui déterminent les accès contradictoires. L'application doit aussi indiquer le processus de résolution des conflits en fournissant des procédures de traitement pour chaque type de conflits.

Dans **BAYOU**, l'allocation des copies se fait à la demande de l'utilisateur sans aucune notion d'anticipation ou d'adaptation au contexte d'exécution. La figure 3.15 illustre le modèle adopté par le système **BAYOU**. Les clients peuvent accéder aux données stockées sur n'importe quel serveur à condition que la communication soit établie. Réciproquement, chaque machine détenant une copie de la base de données peut répondre aux requêtes de lecture et d'écriture d'autres entités avoisinantes (figure 3.15).

BAYOU met en place une stratégie de réplication optimiste de type *read-any/write-any*. Cette stratégie autorise les clients à lire et à écrire sur n'importe quel serveur. Les serveurs propagent les écritures entre eux grâce à un protocole anti-entropique exécutant une procédure dite de réconciliation. Cette procédure assure une propagation épidémique des opérations de mises à jour. Progressivement, toutes les copies convergent vers le même état et parviennent finalement à un état identique si aucune nouvelle modification n'est faite. Cela nécessite que les serveurs reçoivent toutes les requêtes d'écriture et qu'ils les réordonnent de manière unique. Pour cela, une variante de vecteur dynamique de version est utilisée.

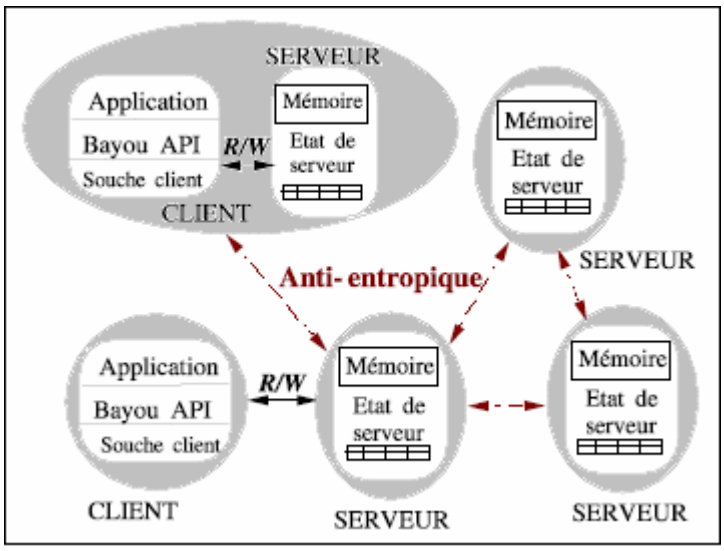

Figure 3.15 : Architecture du système Bayou

3.2.3.1 Détection et résolution de conflits :

BAYOU est un système orienté applications. Il offre des mécanismes grâce auxquels l'application peut spécifier la résolution de conflits et paramétrer le protocole de réconciliation entre les serveurs. On distingue deux types de conflits :

- ✓ les confits *write/write* lorsque deux clients mettent à jour la même donnée de manières différentes
- ✓ les confits *read/write* lorsqu'un client met à jour une donnée en se basant sur la lecture d'une donnée en cours de mise à jour.

La détection de confits s'effectue lors des opérations d'écriture. À chaque opération d'écriture est associée une pré-condition (*dependency check*) définie par l'application. Si celle-ci n'est pas satisfaite, il y a confit et le serveur invoque alors une procédure de résolution de confit (*merge procedure*) également définie par l'application.

L'application peut agir sur plusieurs paramètres du protocole de réconciliation :

- ✓ Le niveau de cohérence (“session guarantees: Read Your Writes, Monotonic Reads, Write Follow Reads, and Monotonic Writes”).
- ✓ Le niveau de validation d'une écriture (stable, tentative). Une mise à jour tentative est une mise à jour qui peut être défaite par un mécanisme de "Rollback". Une mise à jour stable est une mise à jour confirmée qui ne risque plus d'être défaite.
- ✓ Le choix du moment de la réconciliation (de manière périodique, à l'initiative de l'utilisateur, lors de satisfaction de conditions système).
- ✓ Le choix des copies à réconcilier (copies primaires, copies les plus à jour, etc.).
- ✓ Le niveau de vitalité de la réconciliation (troncation plus ou moins importante du journal des écritures pour une stabilisation plus ou moins rapide).
- ✓ Le choix des serveurs pour la création de nouvelles copies.

Ce système présente les inconvénients suivants:

- ✓ La communication entre les serveurs dans le protocole d’anti-entropie nécessite l’envoi d'un vecteur de version, si la taille de ce dernier augmente, le coût d’échange des vecteurs de version dominera l’opération d’anti-entropie. Le passage à l'échelle de ce système n'est pas assuré.
- ✓ Si les opérations de mise à jour (écriture) augmentent plus vite que les opérations de stabilisation, la taille du journal augmente ce qui influe sur les ressources de stockage du système.
- ✓ L’application de la politique du choix arbitraire du serveur à réconcilier dans un réseau peut entraîner la surcharge du réseau dans le cas de serveur éloigné. Dans le cas d’un réseau ad hoc, l’éloignement des serveurs dans la phase de réconciliation entraînera la surcharge du réseau.

3.2.4 Système de gestion de fichiers AdhocFS

AdHocFs a déjà été présenté du point de vue architecture et allocation de copie de donnée. Nous le reprenons ici avec un intérêt particulier à l'approche qu'il adopte pour la mise à jour des données. AdhocFS est constitué de trois couches au dessus du système d'exploitation (figure 3.16). La première couche comprend les fonctionnalités de découverte et d'initialisation pour la gestion des groupes. La deuxième couche correspond aux fonctionnalités de gestion de la cohérence et de la disponibilité des données. Et, la troisième est l'interface AdhocFS [Boulkenafed 03].

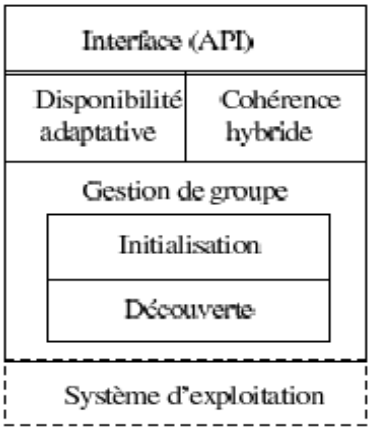

Figure 3.16 : Architecture d'AdHocFS

Le modèle de groupe de AdhocFS partage quelques similarités avec le modèle Ward. AdhocFS exploite la connectivité élevée qui existe à l'intérieur des groupes (groupes connexes à un saut) pour proposer une gestion de cohérence à deux niveaux.

Le ***Home Server*** (serveur statique) affecte des estampilles aux copies d'un fichier. Une copie de fichier assignée à un utilisateur mobile reflète, au début, une valeur stable du contenu. Cette valeur deviendra instable ou tentative car le fichier sera modifié pendant la période de déconnexion. AdhocFS applique une stratégie optimiste de réplication qui permet à n'importe quel utilisateur de servir les demandes d'une manière indépendante.

La gestion de la cohérence au sein d'un groupe est une gestion pessimiste qui offre une cohérence forte au sein d'un groupe. Cette cohérence forte permet d'éviter les conflits des mises à jour au sein d'un groupe. Par contre, la cohérence entre les groupes est assurée par un protocole optimiste ou les conflits sont possibles. Leur résolution passe par un ordonnancement total au niveau du *Home Server*.

On peut distinguer dans AdhocFS trois inconvénients :

- ✓ Seuls les *Home Servers* peuvent stabiliser les opérations de mise à jour. Les utilisateurs mobiles génèrent uniquement des opération tentatives ou

expérimentales. En cas de déconnexion, les utilisateurs ne peuvent accéder qu'à des versions expérimentales non stables des fichiers.

- ✓ les *Home Servers* assurent finalement le rôle de serveurs primaires. En cas de panne du *Home Server* ou de partitionnement, les mises à jour échouent.

3.2.5 Méthodes de mises à jour périodiques et apériodiques

Parmi les premiers travaux sur la réplication pour les réseaux mobiles ad hoc sans infrastructure, nous trouvons ceux réalisés par T. Hara et ses collaborateurs [Hara 01] [Hara 03b]… [Hara 06].

L'auteur de [Hara 02a] propose un protocole de mise à jour périodique. Une opération d'écriture sur une donnée est effectuée uniquement par le serveur *primaire* détenant la copie originale. Par la suite, la modification est propagée aux autres serveurs. En ce qui est des serveurs déconnectés du primaire, leurs copies deviennent *invalides*. Et, ils ne peuvent recevoir les mises à jour. Un type d'application où cette approche serait envisageable est le cas où des nœuds mobiles doivent effectuer régulièrement des prélèvements ou des mesures importantes. Ces mesures sont alors transmises aux autres nœuds périodiquement.

L'hypothèse de réplication périodique reste peu réaliste. L'auteur a considéré ces travaux comme une étape transitoire à cause de la complexité du problème de mise à jour de données sur les réseaux mobiles ad hoc. Il n'a pas tardé à présenter une extension à travers de nouveaux travaux pour la prise en compte de données à mises à jour apériodiques [Hara 02a].

Il suppose que la probabilité pour qu'une nouvelle mise à jour se produise est représentée par une fonction de densité et que chaque nœud possédant une donnée *originale* connaît cette fonction. Cette supposition est aussi irréaliste car il est difficile de connaître les caractéristiques de production d'une mise à jour. Il a ensuite proposé, avec d'autres auteurs, un certain nombre de solutions améliorées jusqu'aux plus récentes (***E-SAF***, ***E-DAFN*** et ***E-DCG***) [Hara 06]. Pour ces dernières, il propose une approche qui utilise le rapport entre les lectures et les écritures, en notant chacune de leurs occurrences. Ce rapport est utilisé comme paramètre pour la réplication (approche présentée dans la première partie de ce chapitre).

Dans [Hara 06], chaque donnée est mise à jour aléatoirement par le nœud possédant la copie originale. Après la mise à jour d'une donnée, si un nœud possédant une copie n'est plus connecté au nœud original, sa donnée devient invalide. Ceci signifie que les répliques sont valides si et seulement si elles sont identiques à la donnée originale.

Les données originales sont considérées *primaires*, et elles jouent un rôle particulier lors des opérations de mise à jour. Les autres copies sont dites *secondaires* par rapport au processus de mise à jour. Dans le cas de plusieurs copies, les nœuds peuvent lire la dernière

version sur leur propre site ou sur l'un des nœuds auxquels ils sont connectés. Les mises à jour sont appliquées aux données *primaires* puis propagées aux données *secondaires*.

Les techniques proposées permettent une grande accessibilité aux données mises à jour aléatoirement. Cependant, un nœud peut accéder à une donnée invalide dont l'originale a été modifiée. Ces accès invalides consomment de l'énergie des nœuds, et posent des problèmes aux nœuds ayant de faibles ressources.

Pour pallier ce problème, deux approches peuvent être adoptées :

- Un temps de vie est associé à chaque copie sur un nœud. Lorsque ce temps est écoulé, le nœud considère la donnée non valide.
- Un rapport d'invalidation est envoyé lorsqu'un *primaire* met à jour sa donnée originale. Ce rapport indique aux nœuds possédant cette donnée qu'elle n'est plus valide.

Dans la première approche, aucun envoi de message n'est nécessaire. Cependant, l'efficacité de cette méthode dépend des temps de vie associés à chaque copie. En effet, il est difficile de connaître le temps de vie d'une copie car les mises à jour sont aléatoires. La deuxième solution est plus efficace car les caractéristiques des mises à jour ne sont pas supposées connues.

Dans le paragraphe suivant, l'impact de l'invalidation des copies sur les accès sera examiné. Cette étude est faite à travers la présentation de deux méthodes [Hara 06]:

- *Update Broadcast* (UB).
- C*onnetion Rebroadcast* (CR).

3.2.5.1 Méthodes d'invalidation des copies

Chaque nœud possède une table d'estampilles dans laquelle les informations concernant les estampilles de toutes les données sont enregistrées. Une estampille d'une donnée au niveau d'un nœud est la dernière date de mise à jour de la donnée connue par le nœud. Elle peut donc être différente de l'estampille actuelle (estampille de la donnée sur le *primaire*).

a) La méthode UB

Dans cette méthode, un rapport d'invalidation est envoyé par le serveur *primaire* d'une donnée à chaque mise à jour. Le rapport d'invalidation contient les informations suivantes :

- L'identifiant de la donnée.
- L'estampille mise à jour.

Lorsqu'un nœud reçoit le rapport d'invalidation, il met à jour sa table d'estampilles si celle ci est dépassée. Dans ce dernier cas, il fait aussi passé sa copie à

l'état *invalide*. Mais, il conserve l'espace mémoire qui lui est réservé. La copie valide sera à nouveau allouée dans cet espace lorsque le nœud accède à la donnée.

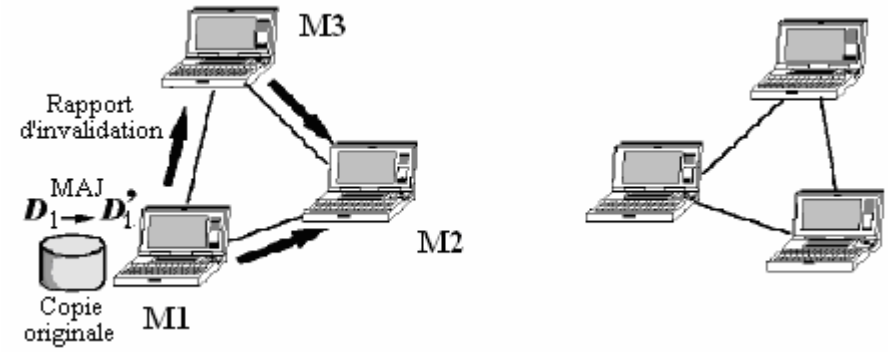

Figure 3.17 : Exemple de la méthode UB

La figure 3.17 est un exemple de cette méthode. Le nœud M_1 est le nœud primaire de la donnée D_1. Il la met à jour et envoie le rapport d'invalidation aux nœuds M_2 et M_3.

Dans cette méthode, le trafic est faible vu que le rapport d'invalidation n'est envoyé que lorsqu'une donnée est mise à jour. Ce rapport est un message de petite taille (il sera composé de très peu de paquets de transmission). La transmission de la donnée complète à chaque mise à jour pourrait produire un trafic considérable à cause de la taille importante d'une donnée (plusieurs paquets de routage). Les nœuds non connectés au primaire ne recevront ni le rapport d'invalidation ni la mise à jour de la donnée (figure 3.17). Ainsi, différentes versions d'une même donnée peuvent être présentes à un même instant (cas de partitionnement).

b) La méthode CR

Comme dans la méthode précédente, un nœud primaire diffuse un rapport d'invalidation à chaque mise à jour. De plus, chaque fois que deux nœuds se connectent l'un à l'autre, ils s'échangent les estampilles recueillies et diffusent les rapports d'invalidation non reçus précédemment (voir figure 3.18). Cette méthode est donc une extension de la méthode précédente par la prise en compte des reconnexions. Les nœuds recevant le rapport d'invalidation effectuent le même traitement sur les données que dans le cas de la méthode UB.

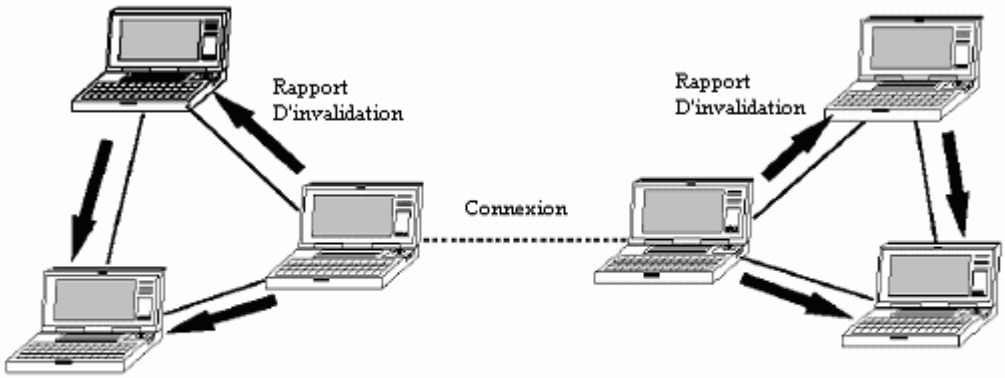

Figure 3.18 : Exemple de la méthode CR

Dans cette méthode, les nœuds reconnectés finissent par avoir la même table d'estampille vu qu'un rapport d'invalidation est envoyé chaque fois que deux nœuds se connectent. Tous les nœuds sont informés le plutôt possible de l'invalidité de leurs données. Cette méthode réduit donc le nombre d'accès aux données invalides par rapport à la méthode précédente. Cependant, les changements fréquents de la topologie font que le trafic causé par l'envoi des rapports d'invalidation est plus élevé que pour UB.

3.2.5.2 Accès aux données

Lorsqu'un nœud veut accéder à une copie dont il ne possède pas l'originale, il diffuse sa requête d'accès sur le réseau. Puis, si l'un des nœuds auxquels il est connecté possède l'originale, il accède à la donnée. Sinon, si ses voisins possèdent des copies, il va accéder à la copie la plus récente. Plus tard, lorsqu'il se connectera au nœud primaire, il vérifiera si l'accès à la donnée a réussi ou non. Pour se faire, il enregistre l'estampille de la copie (sa version) au moment de l'accès.

La technique de maintien des tables d'estampilles diffère de la méthode UB à la méthode CR. Chacune d'entre elles possède une technique d'accès spécifique.

a) Accès aux données avec UB

Pour savoir si les nœuds voisins possèdent l'originale de la donnée ou une copie, le nœud demandeur envoie un message "*data query packet*" de recherche de la donnée. Ce message contient :

- L'identité du nœud qui a envoyé la demande.
- L'identité de la donnée.

Si le nœud qui reçoit la demande de recherche possède la donnée originale ou une copie, il envoie une réponse "*data query reply packet*" au nœud qui a émis la demande. Dans la méthode UB, des nœuds connectés n'ont pas obligatoirement la même table d'estampille, et donc ils peuvent posséder plusieurs versions de la copie. Ainsi la réponse contient les informations suivantes :

- L'identifiant du nœud qui envoie la réponse.
- L'identifiant de la donnée.
- L'indicateur qui informe si ce nœud détient la donnée originale ou non.
- L'estampille de cette donnée sur ce nœud.

Si le nœud demandeur de la donnée reçoit la réponse du nœud primaire, il envoie un paquet de demande de la donnée "*data request packet*" à ce nœud original et accède à la donnée. Dans le cas contraire, il attend un certain temps. S'il reçoit des réponses à partir de nœuds détenant des copies, il envoie un paquet de demande de la donnée au nœud ayant la dernière version parmi ceux qui lui ont répondu. Enfin, il accède à la donnée.

b) Accès aux données avec CR

La procédure de recherche est similaire à la précédente. Un nœud envoie un paquet de recherche de donnée de la même manière qu'avec UB. Un nœud qui reçoit la demande et qui détient la donnée originale transmet une réponse au nœud demandeur. La réponse contient les informations suivantes :

- L'identifiant du nœud qui a envoyé la réponse.
- L'identifiant de la donnée.
- L'indicateur qui informe si le nœud détient l'originale de la donnée ou une copie.

Dans cette méthode, les nœuds connectés possèdent les mêmes tables d'estampilles. Par conséquent, les nœuds qui envoient une réponse détiennent les mêmes versions de la donnée. Il n'est donc pas nécessaire d'ajouter l'estampille de la donnée dans le paquet de réponse. Ainsi, sans attendre un certain délai, si le nœud reçoit une réponse du nœud primaire ou d'un nœud possédant une copie, il envoie immédiatement une demande de la donnée au nœud qui lui a envoyé la réponse.

Conclusion

Dans ce chapitre, nous avons présenté les principales techniques et les protocoles de réplication les plus reconnus présents à ce jour dans la littérature. La réplication permet aux environnements mobiles ad hoc d'offrir aux utilisateurs une indépendance encore plus grande vis-à-vis du lieu et du moment d'accès. Cette réplication a, cependant, un coût. Mais, l'évaluation judicieuse d'un certain nombre de compromis permet une meilleure qualité des services offerts. A travers l'exposé des différentes solutions, nous avons pu constater les problèmes et les avantages engendrés par les divers compromis.

Nous avons présenté quelques exemples de systèmes de réplication. Une première partie a été consacrée aux méthodes visant l'amélioration de l'accessibilité aux données non modifiables. La mise à jour des données en environnements mobiles a été traitée dans une deuxième partie du chapitre. En particulier, l'approche protocole à invalidation pour la mise à jour de données fera l'objet d'une amélioration de notre part. Dans le chapitre suivant, nous présentons une nouvelle approche de réplication qui tente de maintenir un certain nombre des avantages observés.

Chapitre 4

Méthodes de réplication préventive

Introduction

Le développement fulgurant des communications sans fil combiné à la prolifération des calculateurs ultra légers connectables en réseaux mobiles ad hoc amènent une profonde mutation dans la conception, le déploiement et l'utilisation des systèmes d'information. Cette évolution technologique doit permettre à un utilisateur d'accéder à des données et d'exécuter des traitements n'importe où, n'importe quand et à partir de n'importe quel terminal.

Les applications de ce nouveau type de traitement sont multiples. Il peut s'agir d'applications personnelles dans lesquelles un utilisateur voudrait avoir la possibilité d'accéder à tout moment à des données publiques (météo, trafic routier, cours de la bourse …) ou privées (données bancaires, agenda, dossier médical, bookmarks …). Il peut également s'agir d'applications professionnelles dans lesquelles des utilisateurs doivent accéder et partager à tout moment, et où qu'ils se trouvent, des données communes à leur travail (travail collaboratif). Ces besoins sont en contraste avec les contraintes de l'environnement matériel et logiciel: faible débit des réseaux hertziens, déconnexions fréquentes (volontaires ou non volontaires), faibles capacités des terminaux mobiles en terme d'affichage, d'autonomie électrique, de puissance de traitement et de stockage, inadéquation des outils de médiation (middleware) conçus jusqu'à présent pour interconnecter des clients et des serveurs, …etc.

La gestion des données en environnement mobile pose des problèmes spécifiques par rapport à la mobilité en général. Elle doit accomplir de nouveaux challenges en terme d'économie de ressources telles que: énergie électrique, RAM, stockage persistant, … Ce problème est d'autant plus complexe que les calculateurs ultralégers sont souvent basés sur des architectures matérielles très spécialisées. Les déconnexions imprévisibles sont une autre contrainte (figure 4.1). Pour résoudre ces problèmes, les systèmes de gestion de données mobiles font appel à la réplication. Celle ci permet d'offrir aux utilisateurs mobiles des services de haut niveau exauçant leurs exigences.

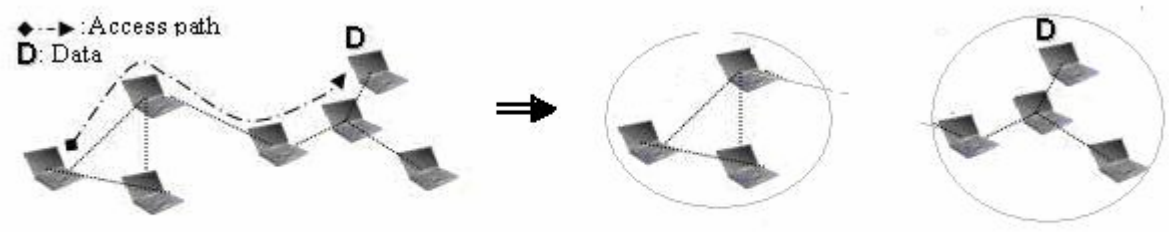

Figure 4.1: Accès aux données distantes

Sur les réseaux mobiles ad hoc, l'absence d'infrastructure fixe apporte un niveau supplémentaire de complication (figure 4.2). La première partie de ce chapitre présente une nouvelle approche de réplication de données pour les réseaux mobiles ad hoc. Cette approche propose une réplication en deux phases pour le partage des données importantes : une phase de réplication préventive, et une autre de réplication adaptative.

La première phase de réplication est une réplication *préventive* et primaire. Elle est exécutée à l'arrivée d'une nouvelle donnée sur le réseau. C'est-à-dire, à la création ou au chargement de la donnée à partir d'une station fixe (en cas d'une éventuelle connexion avec la station fixe). L'algorithme de réplication dissémine la donnée sur tout le réseau par une distribution de ses copies. Celles ci sont réparties uniformément à une distance de ***k*** sauts les unes des autres [Moussaoui 06].

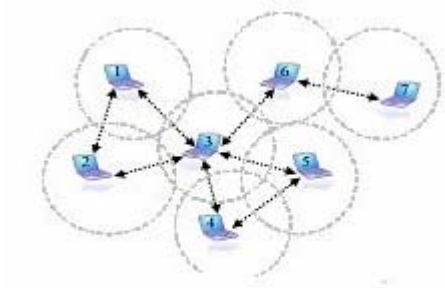

Figure 4.2: Réseau mobile ad hoc

La réplication *préventive* a pour objectifs:

(i) La dissémination de l'information à travers le réseau.

(ii) La distribution uniforme des copies dans le but de servir au mieux les demandes des utilisateurs.

(iii) La prévention d'un partitionnement prochain inattendu.

La deuxième phase de l'approche de réplication proposée est une réplication *adaptative*. Elle tente d'ajuster la localisation des copies aux changements dynamiques de la topologie et des besoins des utilisateurs [Moussaoui 08b]. L'algorithme de réplication *adaptative* considère:

(i) La fréquence des accès aux données. Nous distinguons deux types de fréquences d'accès: les fréquences d'accès internes et les fréquences d'accès externes.

(ii) La distance en nombre de sauts entre les copies.

(iii) Les temps de réponse aux requêtes générées.

Pour améliorer la disponibilité et les performances d'accès aux données, quatre méthodes de réplication sont proposées en début de ce chapitre. Ces méthodes considèrent les données accédées en lecture seule. Un protocole à invalidation pour la prise en compte des opérations de mise à jour est présenté en fin de chapitre.

La première méthode ***HBR*** proposée est une application des deux phases de réplication pour les données importantes. Deux autres méthodes ***IBR*** et ***TBR*** sont des améliorations de la première. Dans ***IBR***, la phase de réplication *préventive* est adaptée au degré d'importance de la donnée [Moussaoui 07c]. Pour les données pas très importantes, la dissémination seule d'une description de données est réalisée. Cette discrimination du niveau d'importance permet de réduire le coût de la réplication selon les contraintes de l'environnement. ***TBR*** est une méthode qui prend un paramètre supplémentaire pour améliorer l'accessibilité: le temps

d'accès aux données [Moussaoui 07a][Moussaoui 08a]. Aucun des travaux dont nous avons eu connaissance ne considère ce paramètre directement. Dans la majorité des cas, les temps d'accès sont améliorés à travers la sélection de chemins d'accès courts. Cependant, un paquet de données peut traverser un chemin assez long, en un temps beaucoup plus rapide que le parcours d'un chemin plus court. L'encombrement du chemin et la surcharge en calcul des nœuds empruntés influencent les délais de propagation. Un flux d'information important peut retarder considérablement les communications.

La quatrième méthode ***GBR*** adopte la même approche. Mais, elle se distingue par une réplication sur une architecture logique en groupes de nœuds. Ce concept est très utilisé sur les réseaux mobiles ad hoc [Huang 06]. ***GBR*** utilise un algorithme simple de construction de groupes, non centralisé et complètement distribué. L'approche de réplication préventive et adaptative est alors adaptée à cette structure en groupes, afin d'en acquérir tous les avantages possibles. Aucune restriction en nombre de nœuds du réseau n'est obligatoire à priori. Le passage à l'échelle peut donc être envisagé et étudié.

A la fin de ce chapitre, nous proposons une méthode de réplication qui considère des données accédées en lecture/écriture. Divers solutions de mise à jour de données distribuées existent. Celles-ci se classifient en deux grandes catégories. Les stratégies de réplication pessimistes à cohérence forte et les stratégies de réplication optimistes à cohérence faible. Les réseaux mobiles ad hoc se prêtent difficilement aux stratégies pessimistes de part leurs caractéristiques. En effet, la réplication pessimiste exige souvent un consensus global ou partiel alors qu'il est difficile d'assurer la présence de tous les éléments d'un ensemble de nœuds à un instant donné.

Cependant, il existe des applications qui nécessitent un degré de cohérence important. Dans ce cadre, nous apportons une contribution à ce problème complexe par la proposition d'une méthode de réplication pour la cohérence forte. Cette méthode est de type protocole à rapports d'invalidation.

4.1 *Environnement de travail*

L'environnement considéré est un ensemble de terminaux mobiles (nœuds) qui coopèrent à la construction d'un espace mémoire partageable. Chaque nœud peut émettre et recevoir des ondes hertziennes sur une portée de communication de rayon ***R***. Un lien de communication entre deux nœuds est maintenu tant que leurs portées de communication se recouvrent. Les liens de communication sont supposés bidirectionnels.

Un nœud mobile peut générer une nouvelle donnée partageable. Ce premier exemplaire est la copie *originale* de la donnée. Il peut aussi sauvegarder localement dans son cache des descriptions de chemins d'accès aux données afin d'activer la localisation des données.

Dans cet environnement un nœud peut importer une donnée à partir d'un nœud statique en cas d'une éventuelle connexion. La consistance des données n'est pas prise en compte dans

une première étape de ce travail. Les données concernées sont alors accessibles uniquement en lecture. Nous pouvons trouver une multitude d'applications où la modification des données ne présente pas un intérêt majeur. Par exemple, les informations relatives au plan géographique d'une région, les cartes routières et les plans de disponibilités de services utiles (hôpital, aéroport, …).

Nous supposons que:

- chaque nœud est désigné par une identité unique N_i ; tel que, $0 \leq i \leq n$ où n est le nombre maximal de nœuds possibles.
- chaque donnée est identifiée par une identité unique D_{ij}, où i est l'identité du nœud primaire détenteur de la copie *originale* et j est un numéro de séquence attribué à la donnée à sa création sur le serveur primaire.
- les nœuds mobiles ont une même taille d'espace mémoire réservé à la sauvegarde des copies et des chemins d'accès aux données.
- périodiquement un nœud mobile informe ces voisins des caractéristiques des données locales qu'il détient.
- deux types de fréquences d'accès sont définis sur chaque nœud et pour chaque donnée:
    - une fréquence *externe* qui représente le pourcentage des accès d'un nœud à la donnée si elle est non locale.
    - Une fréquence *interne* qui représente le pourcentage des accès des nœuds globalement à la donnée si elle est locale.

4.2 *Partage d'accès aux données sans mise à jour*

Nous proposons une approche décentralisée et complètement distribuée pour la réplication de données sur un réseau mobile ad hoc. Quatre méthodes basées sur cette approche sont détaillées. Le principe de choix et de localisation des données ainsi que les procédures d'accès sont présentés pour chaque solution.

4.2.1 *Méthode HBR ("Hop-Based Replication")*

Cette méthode traite le cas où la donnée vient juste d'être introduite ou créée sur le réseau. Et aussi, le cas où la donnée est déjà utilisée mais nécessite de répondre au dynamisme de la topologie et à la mobilité des nœuds. Elle est basée sur une approche à deux phases que nous détaillerons avec l'exposé de cette première méthode. Les deux phases de réplication : la réplication préventive et la réplication adaptative, correspondent à deux comportements différents selon que:

- La donnée est nouvellement arrivée sur le réseau. La dissémination est alors nécessaire afin d'en informer les nœuds. Cette dissémination est accompagnée d'une réplication préventive. L'objectif est de prévenir les partitions hâtives et imprédictibles.

- La donnée existait déjà sur le réseau mais des changements au niveau des accès et de la topologie se sont opérés.

La réplication peut utiliser des techniques de prédiction du partitionnement pour déduire une nécessité de réplication. Cependant, ces dernières produisent un "overload" en consommation de bande passante, en énergie et en calcul de la part des nœuds [Chen 02]. Ce coût devra être rajouté au coût de la réplication. D'autre part, ces prédictions ne sont pas sûres. De nombreuses approches évitent ces prédictions par une réplication préventive [Hara 01][Shinohara 06] [Pandamanan 07][Hara 06]. Mais pour la plupart, la condition de réplication est définie sur des paramètres qui expriment le besoin de l'utilisateur. Ce besoin est exprimé à travers des métriques telles que les fréquences d'accès, les relations sémantiques entre les données, ou encore les profils des utilisateurs.

A la création d'une donnée, les besoins en accès à celle ci par un utilisateur ne sont pas toujours connus. Par exemple, pour qu'une fréquence d'accès induise une réplication, il faut : (1) que l'utilisateur ait pris connaissance de son existence, (2) qu'il y ait accédée, (3) et que la valeur de cette fréquence ait atteint le niveau requis par la condition de réplication. Durant tout ce temps, des déconnexions peuvent se produire avec l'apparition de partitions. La donnée peut alors devenir inatteignable pour un ensemble des noeuds. Les données ne peuvent être répliquées sur chaque nœud à cause des espaces mémoires réduits. Il faut donc sélectionner les nœuds de placement des copies. Nous avons choisi le critère de distance pour la détermination de ces nœuds. Cette distance est maintenue à une valeur de ***k*** sauts entre les copies. Ce paramètre est intéressant car:

(i) il assure pour chaque nœud l'accès à la donnée à une distance inférieure à ***k*** sauts.
(ii) et, il permet d'exprimer le degré de partage et d'importance de la donnée. Pour une donnée importante, ***k*** prendra une petite valeur.

Dans le cas où la donnée n'est pas nouvellement créée, il s'agit d'offrir une réplication qui assure la relocalisation des copies de manière continue puisque toute condition antérieure de localisation d'une donnée peut ne plus être satisfaite. En effet, les nœuds hôtes sont mobiles et les utilisateurs eux-mêmes peuvent avoir changer de position. Cette deuxième phase de réplication est dite adaptative car elle doit déterminer les conditions de réplication correspondantes à la configuration actuelle et aux besoins courants des utilisateurs. Pour enfin, adapter la localisation des copies.

4.2.1.1 Réplication préventive

Cet algorithme duplique les données originales à la création et distribue les copies uniformément. Le principe de cette distribution est de garantir les accès aux données à tous

les nœuds. Elle procède par un compromis entre l'espace mémoire utilisé et l'accessibilité. Effectivement, les données sont répliquées sur les nœuds séparés d'une distance de ***k*** sauts. La valeur de ***k*** peut être définie comme un paramètre de génération de la donnée. La valeur de ***k*** indique l'importance de la donnée. Une valeur élevée ne peut être affectée qu'à une donnée de faible importance ou dans le cas d'un réseau à mobilité limitée.

La figure 4.3 représente un exemple de réplication préventive à ***k*** sauts (***k*** égale à trois). Nous pouvons voir que le nœud N_0 a trois voisins N_1, N_3, et N_4. L'algorithme de réplication *préventive* placera une copie de la donnée *originale* D_0 de N_0 sur le nœud N_4. On remarque que N_0 peut accéder à une variété de données (D_0, .., D_8) à un ou deux sauts (***k*** = 3). Pour accéder à une donnée telle que D_1, N_0 sollicitera le nœud serveur de D_1 le plus proche, c'est-à-dire N_3. Si N_0 ne peut atteindre N_1, il accède à la donnée à partir de N_2. Cette distribution améliore l'accessibilité, optimise l'utilisation de l'espace mémoire et peut même améliorer les temps d'accès (accès aux données les plus proches). L'objectif est de garantir à chaque nœud l'accès aux données les plus importantes à une distance inférieure à ***k*** sauts.

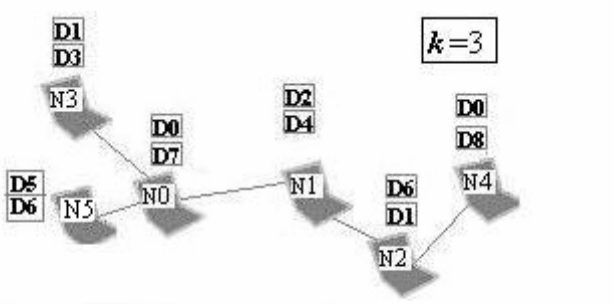

Figure 4.3: Exemple de réplication préventive à ***k*** sauts

• **Principe de l'algorithme:**

A la création d'une donnée originale $\mathbf{D_k}$, le nœud hôte **NodeAdd** initialise une diffusion du message de création préventive **PCreat (NodeAdd, $\mathbf{D_k}$, HopCpt)** avec le compteur de sauts **HopCpt** initialisé à 0. Le rôle du message **PCreat** est:

(1) d'informer les noeuds de la disponibilité d'une nouvelle donnée sur le réseau,
(2) d'estimer les distances entre les noeuds, et
(3) de placer les copies sur les nœuds en respectant la condition de séparation de deux copies successives de ***k*** sauts.

Lorsqu'un message est reçu par un nœud, il le traite comme suit:

- Le compteur des sauts est incrémenté de un si la donnée existe déjà localement. Sinon, il est réinitialisé à 0.
- Si le compteur est égal à ***k*** sauts et s'il n'existe pas de copie locale au nœud ou à un des voisins, la donnée est répliquée localement. Par exemple, sur la figure 4.4 (exemple avec ***k***=3), le nœud N_9 reçoit un compteur égal à ***k***-1. Il l'incrément à ***k*** et

crée une copie à la première réception du message. Il assigne 0 au compteur et diffuse le message avec la nouvelle valeur du compteur.

- Si le compteur est égal à ***k*** et si la donnée existe sur l'un des voisins les plus proches à *h* sauts (*h*<***k***-1), le compteur des sauts est initialisé à *h*. Il est par la suite diffusé aux nœuds *suivants* qui ne disposent pas de la donnée. Les nœuds *suivants* sont les nœuds à partir desquels le nœud n'a pas reçu de messages ***PCreat***. Dans l'exemple de la figure 4.4, N_{10} reçoit un compteur égal à deux à partir de N_6. Il incrémente le compteur à trois, car il n'a pas encore reçu de message de N_9. Mais avant de créer une copie de la donnée localement, il diffuse à ses voisins immédiats (N_9, N_{11} et N_{12}) un message d'intention de création de la donnée. Dans ce cas N_9 doit répondre à ce message par un *ACK* négatif car il en détient une copie. Suite à cette réponse, N_{10} déduit qu'il existe une copie de la donnée à un saut sur le nœud N_9. Il ne crée pas de copie locale pour éviter une redondance non utile et rediffuse le message ***PCreat*** avec un compteur de sauts égal à un. De la même manière N_{12} obtient en premier un compteur de valeur trois (après incrémentation). Mais il ne réplique pas la donnée car N_{10} lui répond par un *ACK* négatif. Ce dernier a connaissance de l'existence d'une copie à un saut sur N_9. Donc, N_{12} peut accéder à deux sauts (2<***k***) à la donnée sur N_9.

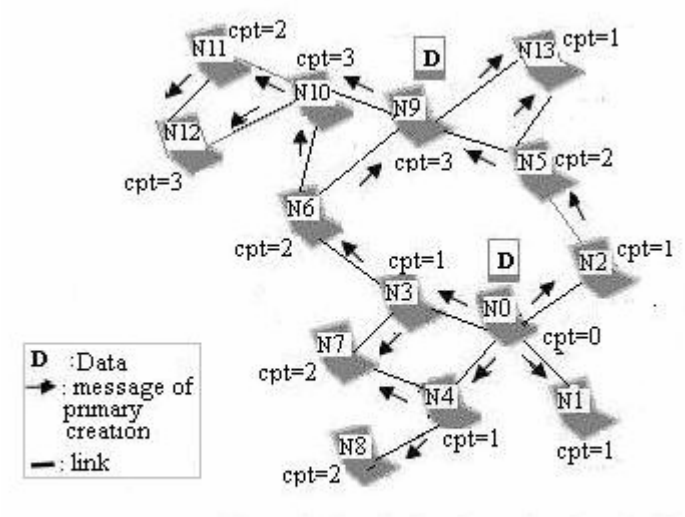

Figure 4.4:Exemple de réplication préventive (***k***=3)

Seule la première réception du message de création préventive est traitée. Toutes les réceptions suivantes du même message sont ignorées pour la création des copies. Lors du placement d'une copie, s'il y a un manque d'espace, l'algorithme de remplacement sélectionnera la donnée la moins utilisée (ayant une des fréquences internes f_i les plus faibles) comme candidate au remplacement. Toute fois cette réplication préventive ne doit pas détruire une donnée en cours d'utilisation. Une donnée ne peut être remplacée par une copie préventive que si et seulement si cette donnée n'est pas utilisée depuis un temps d'utilisation T_u qui soit inférieur à une durée seuil T_s. L'algorithme de remplacement sélectionne alors

parmi ces données celle qui détient le facteur f_t le plus faible. Cette donnée correspond à la donnée la moins utilisée par le nœud et par son voisinage. Ce facteur est calculé comme suit:

$$f_t = \frac{f_i}{T_u}$$

f_t exprime l'intérêt de: (i) la fréquence d'utilisation et, (ii) des délais récents d'utilisation. En ayant deux données Di et Dj anciennement utilisées, l'algorithme sélectionne celle dont le facteur f_t a la plus petite valeur. Si les deux données ont le même temps d'ancienneté dans l'utilisation, l'algorithme préférera remplacer la donnée la moins utilisée. On considère qu'un utilisateur qui a beaucoup utilisé dans le proche passé une donnée aura plus de chance de réutiliser cette même donnée. Par contre devant deux données de même fréquence d'accès, l'algorithme sacrifie la donnée la plus anciennement utilisée. Une copie *originale* ne peut jamais être remplacée.

L'accessibilité à une donnée à une distance maximale inférieure à ***k*** sauts peut contribuer à l'amélioration des temps de réponse aux requêtes. La charge de service est distribuée de manière équilibrée entre les serveurs. L'utilisation de la mémoire est optimisée à travers l'élimination des redondances injustifiées. La redondance est calculée par le nombre de copie d'une même donnée sur un voisinage. La gestion dynamique de ces copies est assurée par la réplication adaptative présentée dans le paragraphe suivant (paragraphe 4.2.1.2).

- **Algorithme:**

a) Les fonctions et les algorithmes utilisés sont principalement: ***PReplica*** ("Preventive Replica") et ***PCreat*** ("Preventive Creation").

- ***PReplica***(**D_k**): Cette fonction
  1) crée une copie de la donnée **D_k**,
  2) affecte la valeur zéro **à** la variable **HopCpt**, et
  3) diffuse **PCreat**(**NodeAdd,D_k,HopCpt**) aux nœuds *suivants*.

- ***PCreat*****(NodeAdd, D_k, HopCpt):** dans ce message, **NodeAdd** est l'identité du dernier noeud atteint par le message et qui héberge une copie de la donnée. **HopCpt** est le nombre de nœuds parcourus depuis ce nœud hôte de la donnée.

b) Algorithme de réplication préventive de la donnée D_k (k=3) sur N_j :

```
Procedure of new data creation ()
Begin                                   /* node N_J creates a new data D_k */
  HopCpt = 0;
  NodeAdd= N_J;
  Send PCreat(NodeAdd, D_k, HopCpt) to following nodes
End procedure;
```

```
Procedure PCreat (NodeAdd, Dk , HopCpt) Reception
Begin
 If first reception  then
   HopCpt = HopCpt +1;
   If (HopCpt  =3)  then
    If ( Dk replica not exist on neighbours at p hops (p <k-1)
     then PReplica (Dk );                    /*ACK s are used to know that*/
     else
       HopCpt =p;  NodeAdd= Ni ;                    /* Ni local node */
       Send PCreat (NodeAdd, Dk, HopCpt) to following nodes
     end if
   end if
  else
   If (Hop Cpt =2) and (Ni has unique neighbour) then
   else PReplica (Dk );                    /* to prevent its disconnection* /
     Send P Creat (NodeAdd, Dk,  HopCpt)  to the following nodes
   end if
 end if
End procedure;
```

4.2.1.2 Réplication *adaptative*

La mobilité des noeuds et les fréquentes connexions/déconnexions modifient la topologie. La réplication adaptative ajuste la localisation des copies afin d'améliorer l'accessibilité des noeuds. Pour obtenir une meilleure accessibilité et de meilleurs temps de réponse, il est intéressant de répliquer les données à proximité des nœuds qui les utilisent le plus souvent. Les données les plus utilisées par un nœud sont répliquées localement ou à proximité. Cette relocalisation des copies traite aussi du problème de redondance d'une donnée sur des nœuds voisins. Les paramètres de la réplication dans ***HBR***, sont: la fréquence d'accès et la distance entre les copies.

Les accès fréquents à une donnée distante engendrent un surcoût en trafic, en consommation en bande passante et en énergie des nœuds parcourus, en particulier, si le chemin d'accès est long. L'évolution dynamique des accès des utilisateurs et des positions des copies est considérée pour limiter ces inconvénients. Nous rappelons que sur un noeud N_i, deux fréquences d'accès à une donnée D_k sont définies:

- Fréquence d'accès externe fe_{ik} : Taux des fréquences d'accès du noeud N_i à la donnée D_k si elle est non locale.

$$fe_{ik} = \frac{n}{U}$$

où, n est le nombre des requêtes d'accès à D_k le long d'une période de temps U .

- Fréquence d'accès interne fi_{ik} : Taux des accès à une donnée D_k si elle est locale.

$$fi_{ik} = \frac{n}{U}$$

où, ***n*** est le nombre des demandes d'accès à la donnée locale D_k par l'ensemble des noeuds.

Les taux des fréquences d'accès sont calculés dynamiquement et à chaque période ***U***. Un seuil ***S*** de fréquence externe est défini. Lorsque ce seuil est atteint, la donnée est localement répliquée. En cas de manque d'espace, l'algorithme de remplacement présenté dans le paragraphe précédent est exécuté. La réplication adaptative est implantée à travers deux procédures: La réplication sur accès et la réplication au besoin.

- **Réplication sur accès:**

A la génération d'une requête d'accès par un nœud,

- la donnée est, en premier, recherchée localement,
- si elle existe localement, la requête est traitée immédiatement par le nœud,
- autrement, la requête devrait être diffusée sur le réseau. Ceci produirait une surcharge en trafic sur un réseau étendu. Par conséquent, pour chaque donnée distante, un nœud N_i sauvegarde le chemin d'accès le plus court qu'il détecte. (***IdNode***, ***IdData***, ***NbHop***) est la description d'un chemin d'accès où, ***IdNode*** est l'identité du serveur le plus proche, ***IdData*** est l'identificateur de la donnée et, ***NbHop*** est le nombre de sauts nécessaires pour atteindre ***IdNode.*** La sauvegarde de chemins d'accès limite le coût en message et en consommation énergétique de la procédure de recherche d'une donnée. Ainsi, la requête est diffusée uniquement en cas d'inexistence de chemins d'accès. Cette requête comporte l'identité du noeud *appelant* source de la requête (N_j), l'identité de la donnée et le nombre de sauts parcourus par la requête. La progression de cette requête véhicule des informations utiles qui peuvent être exploitées par la réplication adaptative.

A l'accès à une donnée par un noeud ***Nj,*** l'algorithme suivant est exécuté:

```
Procedure of data access by a node Nj ()
Begin
 If  Nj has  a replica  of the data
  then       The access is locally done
  else
   If Nj has a valid access path to a holder node of the data
       then
      Send request to specified node in the path
     else
      Broadcast request to the neighbours.
   End if
 End if
End procedure;
```

A la réception d'une requête, un serveur ou un nœud qui a un chemin à la donnée envoie le chemin d'accès au nœud *appelant* source de la requête. Si le plus court chemin recueillis par le nœud *appelant* est d'une longueur supérieure à ***k*** sauts, il réplique la donnée localement ou sur l'un de ses voisins à moins de ***k*** sauts. La création de cette copie optimisera les accès ultérieurs du nœud *appelant*.

Nous désignons cette réplication par réplication sur accès car la stratégie exploite simplement les requêtes d'accès pour décider de l'utilité du placement d'une nouvelle copie. La notion d'utilité ici est définie par rapport aux conditions établies sur la localisation des copies par la réplication préventive. La requête d'un nœud N_i peut être reçue par:

- ✓ un nœud non serveur de la donnée qui décide alors
  - de répondre au nœud appelant par la transmission d'un chemin d'accès valide s'il en dispose,
  - ou de diffuser la requête aux nœuds *suivants* dans le cas contraire.
- ✓ un serveur N_j hôte d'une copie de la donnée qui répond au nœud *appelant* par un message réponse **Rep (*N_j, N_i, D_k*, *HopCpt*)** où, D_k est l'identité de la donnée, et ***HopCpt*** est le nombre de sauts nécessaires pour que N_i puisse atteindre N_j. A la réception de ce chemin par le nœud appelant, il l'enregistre pour pouvoir le comparer aux autres chemins reçus.

Périodiquement, les messages de découverte de voisinage sont utilisés pour identifier les voisins directs. Ils sont aussi exploités pour la construction des chemins d'accès au niveau des nœuds. Quand un nœud découvre un nouveau voisin, il lui transmet la liste des données locales et des chemins valides. Le nœud récepteur effectue les mises à jours nécessaires sur ses structures de localisation des données. Lorsqu'un nœud crée, par exemple, une nouvelle donnée, il diffuse cette information aux voisins immédiats. Ces derniers procèdent alors à la mise à jours des chemins d'accès.

La mobilité fait que les chemins peuvent devenir invalides. Ils nécessitent donc souvent une révision. Nous associons un délai *TTL* (Time To Live) à chaque chemin. C'est la durée moyenne de maintient d'un chemin sur le réseau. A l'écoulement de ce délai, le chemin devient invalide. Ce délai est fixé à une valeur assez grande pour une mobilité limitée. Il dépend intrinsèquement de la mobilité de l'environnement. Lors de la sauvegarde d'un nouveau chemin, s'il y a un manque en espace mémoire, le chemin le plus ancien est remplacé car il devra probablement expirer dans peu de temps.

Sur un nœud N_i l'algorithme de mise à jour des chemins, est le suivant:

```
Procedure of Paths update
Begin
 If reception of data list from a node then
     Save the shortest path to non local data;
 End if;
 If reception of PCreat (Nj ,Dk, HopCpt) and Ni not holder of Dk replica then
  If HopCpt < k and not exist a shorter path
   then
   Save the access path (Nj, Dk, HopCpt)
   End if;
 End if;
 If reception Rep (Nj, Nk, Dk, HopCpt) to an access request to Dk carried out by Nk
 then                          /* Nj node holder of replica*/
   HopCpt= HopCpt+1;       /*nb of hops from Nj to Ni */
    If Ni <> Nk then
      If   not exist a shorter path then
      Save the access path: (Nj, Dk, HopCpt) ;
      Send Rep(Nj, Nk, Dk, HopCpt) to Nk
      Else       /*exist shorter path (Np, Dk, HopCpt1) */
           Send Rep(Np, Nk, Dk, HopCpt1) to Nk
      End if
    Else                                        /* Ni = Nk */
     If not exist a shorter path then
       In order to execute the request, save the shorter access path: (Nj, Dk, HopCpt);
     End if;
   End if;
End if;
End procedure;
```

La figure 4.5, nous montre un exemple de réplication suite à une demande d'accès. Cette réplication permet de servir la requête avec de meilleures performances. Dans cet exemple, nous supposons que N_{11} et N_{12} viennent de se connecter au réseau et qu'ils ne disposent pas d'une copie de la donnée ***D***. ***k*** est supposé égal à trois.

Pour accéder à ***D***, N_{12} diffuse une requête d'accès à N_{11} et N_7. N_{11} n'est pas un hôte de la donnée et il ne détient pas de chemin vers cette donnée. Alors, il diffuse encore la requête aux noeuds *suivants*. Par contre, en recevant la requête, N_6 transmet à N_{11}, un chemin d'accès à la donnée. N_{12} recevra donc un chemin de N_{11} (***D*** sur N_9) et, un autre de N_7 (***D*** sur N_0). La donnée est accessible pour N_{12} à partir de N_9 et N_0 qui sont respectivement à ***k***+1 et à ***k*** sauts. Dans ce cas, afin de mieux résister aux déconnexions et d'avoir de meilleurs temps de réponses, N_{12} crée une copie de la donnée localement s'il a de l'espace ou sur un nœud de son voisinage à k-1 sauts (ce nœuds pourrait être , par exemple, N_{11}).

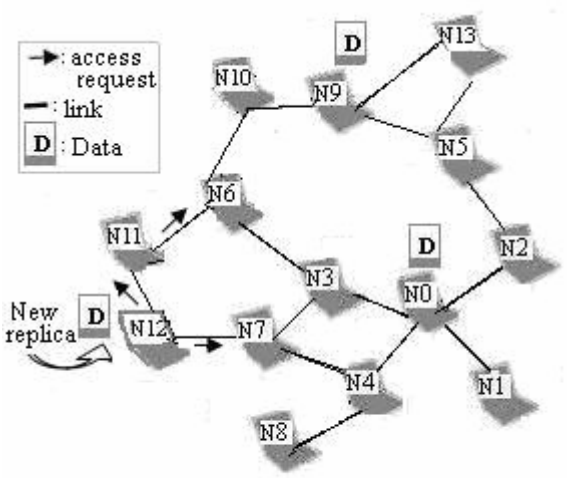

Figure 4.5: Exemple de demande d'accès

• **<u>Réplication au besoin</u>:**

Dans le but de répliquer localement, les données les plus utilisées, à chaque période de temps U, un nœud N_i:

- calcule, pour chaque donnée D_k, les taux de fréquences d'accès fe_{ik} et fi_{ik}.
- Si fe_{ik} dépasse le seuil de réplication S, une requête de réplication est envoyée à un serveur s'il y a localement un chemin valide à la donnée. Ou bien, la requête est diffusée à la recherche d'un nœud source de réplication (si aucun chemin n'est disponible localement). Le traitement de cette requête de réplication est similaire au traitement d'une requête d'accès. Globalement, les copies sont localisées à k sauts l'une de l'autre. Mais au cas où une donnée est très utilisée par un nœud, il la réplique localement pour maintenir de bonnes performances d'accès. Dans ce cas, même en cas de redondance, celle-ci est tolérée car elle est jugée utile. En cas de manque d'espace mémoire, l'algorithme de remplacement décrit précédemment est exécuté pour choisir la donnée à remplacer.

  Evidemment, les données sont traitées dans l'ordre décroissant des fréquences d'accès externes.

4.2.2 *Méthode IBR ("Importance-Based Replication")*

Sur un réseau mobile ad hoc avec un grand nombre de nœuds, chaque nœud peut générer différentes données *originales*. La création des copies préventives lors de la première phase de réplication peut avoir un coût non négligeable en consommation de ressources. Il est vrai que ces créations sont relativement rares par rapport aux accès. Toutefois, dans le cas où un certain nombre d'entre elles coïncident dans un laps de temps court, la surcharge peut être significative. Pour répondre à ces préoccupations, cette méthode considère toujours les

données importantes. Mais, elle traite différemment les données les plus importantes de celles qui le sont moins. Cette méthode est une variante de la méthode précédente.

Les données originales jugées très importantes sont disséminées et subissent une réplication préventive lors de leur création. Par contre, pour les données jugées moins importantes, une description de la donnée (Méta-donnée) est générée et diffusée sur le réseau. Par exemple, sur un champ de bataille, si un combattant prend connaissance d'une information importante, il ne serait pas raisonnable d'attendre jusqu'à ce que cette information soit demandée par les autres combattants. Par contre, l'information sur les permissions accordées est moins urgente et il suffit d'informer les combattants de sa disponibilité. Ils pourront par la suite la consulter. Ce même scénario peut être retrouvé au niveau d'un groupe de chercheurs qui coopèrent à des travaux communs. Un chercheur qui obtient un résultat significatif pour le projet commun, n'attend pas que les autres serveurs fassent la demande de consultation du résultat. L'algorithme de réalisation de la première phase préventive est alors adapté comme suit:

```
Procedure of new data creation ()
Begin                                   /* node  N_J creates a new data D_k  */
    HopCpt = 0;
    NodeAdd = N_J ;
    If D_k will be very requested    then
          send PCreat(NodeAdd, D_k, HopCpt) to neighbours
    else
          send Crtl_Msg (NodeAdd, D_k,  HopCpt );
    End if;
End procedure;
          /*Crtl_Msg: disseminate information (D_k exist) */
```

L'algorithme de réplication adaptative reste le même que celui de la méthode précédente.

4.2.3 *Méthode TBR ("Time-Based Replication")*

Les précédentes méthodes visent l'amélioration des temps de réponse des accès aux données mais aucune ne considère clairement ce paramètre. Son amélioration est déduite à travers l'amélioration des temps de recherche et de localisation d'une donnée. En effet, la limitation des distances entre un client et un serveur apporte une contribution certaine dans l'amélioration du temps de traitement des requêtes d'accès. Mais, est-ce toujours vrai?

Un des principaux paramètres à considérer aussi par une procédure de réplication est le paramètre qui représente les besoins en données de l'utilisateur. Dans les solutions les plus

souvent proposées, ce paramètre est: la fréquence d'accès aux données par l'utilisateur. Est-il vraiment le paramètre le plus significatif ou le seul à considérer?

Dans cette méthode variante des deux précédentes, le temps de réponse effective des données est considéré. Nous définissons l'accessibilité comme étant la capacité d'un nœud à atteindre une donnée avec des performances d'accès satisfaisantes (temps d'accès et trafic engendré). Il est certain qu'un utilisateur aimerait accéder à sa donnée avec les meilleurs temps d'accès. Une technique de placement de copies peut améliorer l'accessibilité, tout en optimisant les temps de réponse des requêtes si elle considère ce dernier paramètre.

Une technique de réplication basée uniquement sur les fréquences d'accès et les distances n'est pas toujours suffisante pour apporter une amélioration sûre au niveau des temps d'accès. Les temps de réponse aux requêtes peuvent être rallongés par divers facteurs; à savoir la distance parcourue évidemment mais aussi l'état des communications sur le chemin emprunté et la surcharge en calcul des nœuds traversés. Par exemple, si on suppose deux données D_i et D_j non locales au nœud N_k qui y accède respectivement avec des fréquences externes de 55% et de 45%. Si leurs temps d'accès respectifs sont de 1ms et 2ms, est-il plus intéressant pour N_k de détenir une copie de D_i ou une copie D_j.

On observe que la création d'une copie de D_j localement à N_k améliorera son temps d'accès global de 90ms (gain 62% du temps global) alors que la réplication de D_i apportera une amélioration de 55ms (gain de 38% du temps global).

TBR est une amélioration des méthodes précédente. La différence se situe dans la deuxième phase de réplication adaptative. La condition de réplication ne porte plus uniquement sur les fréquences d'accès mais aussi sur les temps de réponse des requêtes accomplies récemment. Pour chaque donnée D_k, et sur chaque nœud N_i, une fonction ft_{ik} de la moyenne des temps d'accès est définie comme suit:

$$ft_{ik} = \frac{T_{ik}}{U * fe_{ik}}$$

ft_{ik} représente le temps d'accès moyen à D_k par N_i le long de la période courante. T_{ik} représente la somme des temps d'accès de N_i à D_k durant la période U précédente. fe_{ik} est le fréquence externe d'accès à D_k par N_i.

La condition de réplication adaptative est alors la suivante: les données prioritaires pour une duplication sont celles jugées très utilisées et dont la réplication améliore le temps d'accès global du nœud. Ainsi, au début de chaque période de réplication et sur chaque nœud, les données ayant une fréquence d'accès dépassant le seuil S défini et, dont les fonctions ft_{ik} sont les plus importantes subiront en premier une réplication locale à N_i.

4.2.4 *Méthode GBR ("Group-Based Replication")*

La création d'un nombre important de copies d'une donnée sur des noeuds géographiquement proches, peut restreindre les capacités de l'espace mémoire commun partageable. Les méthodes précédentes tentent de limiter les redondances inutiles. Cependant le traitement parallèles et la mobilité peuvent produire des redondances non contrôlées. Cette redondance bien qu'elle améliore l'accès à la donnée en question, elle restreint l'accessibilité globale. Des données importantes pourraient ne pas être répliquées faute d'espace. Aussi, ces méthodes ne considèrent pas la qualité des liaisons des nœuds choisis pour héberger des copies.

Dans cette méthode, l'ensemble des nœuds du réseau est subdivisé en sous ensembles de nœuds ou groupes. Les nœuds associés en groupe sont connectés et géographiquement proches. Pour chaque groupe, un processus est exécuté pour la réplication des données dont le groupe a besoin. Dans ce cas, les besoins en données ne sont plus évalués individuellement pour chaque nœud. Les membres d'un groupe coopèrent dans le but de maintenir l'accessibilité aux données pour l'ensemble du groupe. La notion de groupes de réplication a été utilisée dans plusieurs travaux récents [Padmanabhan 07]Examinateur [Huang 06] [Hara 04][Hara 06][Boulkenafed 03]. Cette notion résiste mieux aux problèmes de partitions imprévisibles et de déconnexions. L'approche groupe de réplication est aussi très favorable à un passage à l'échelle du réseau.

La méthode de réplication ***DCG*** ("Disconnected Communication Groups") [Hara 03a][Hara 06] est basée sur un algorithme de construction de groupes. Ces travaux définissent une notion de groupes de réplication stables bi–connectés. Chaque nœud dans un groupe est connecté à au moins deux nœuds du même groupe par des liens dits stables. Par conséquent, les nœuds appartenant à un même groupe doivent former un circuit. Sur de tels groupes la connectivité est assez forte, mais les hypothèses requises ne sont pas toujours évidentes (voir chapitre 3). Ces solutions supposent un réseau de taille limitée et la construction de l'état global de la topologie au niveau de chaque nœud pour déterminer les groupes.

D'autres parts, la condition de bi-connexion semble trop forte et plutôt contraignante. Elle engendre des groupes de dimension très petite. D'où la gestion d'un espace mémoire commun réduit aussi. La bi-connexion produit souvent des groupes *singleton* d'un seul membre. Quel est l'intérêt d'une construction aussi laborieuse pour obtenir une majorité de groupes singletons ou à petit effectif en membres? Nous rappelons qu'un groupe est un ensemble de nœuds partageant un espace mémoire pour l'accès aux données communes.

La méthode ***GBR*** propose toujours une réplication adaptative périodique (période ***U***) [Moussaoui 07e]. A chaque période trois étapes s'exécutent:

(i) Construction des groupes de réplication. Ces groupes sont basés sur la notion de stabilité des liens de communication.
(ii) Allocation adaptative de copies de données sur les groupes du réseau.
(iii) Accès aux données.

4.2.4.1 Stabilité des liens de communication

Un lien de connexion est dit stable si la probabilité pour qu'il soit maintenu durant une période de temps ***t*** prochaine est jugée importante. Nous considérons deux nœuds N_i et N_j à une distance d_1. Le lien de communication entre ces nœuds est stable si à l'écoulement de la période de temps ***t*** considérée, la distance d_2 séparant les nœuds est inférieure à 2x***R*** (voir figure 4.6).

Nous supposons qu'en moyenne les nœuds se trouvent à une distance ***R*** (dans ce cas d_1 est en moyenne égale à ***R***). La distance parcourue $d=(d_2-d_1)$ devrait donc être inférieure ou égale à ***R***. Le temps moyen de déplacement pour le parcours de la distance ***d*** est estimé par : d/v_{moy}, où v_{moy} est la vitesse moyenne de déplacement. Nous définissons la durée maximale t_s de stabilité d'un lien (délai maximal de maintient d'un lien) par: $t_s = R / v_{moy}$.

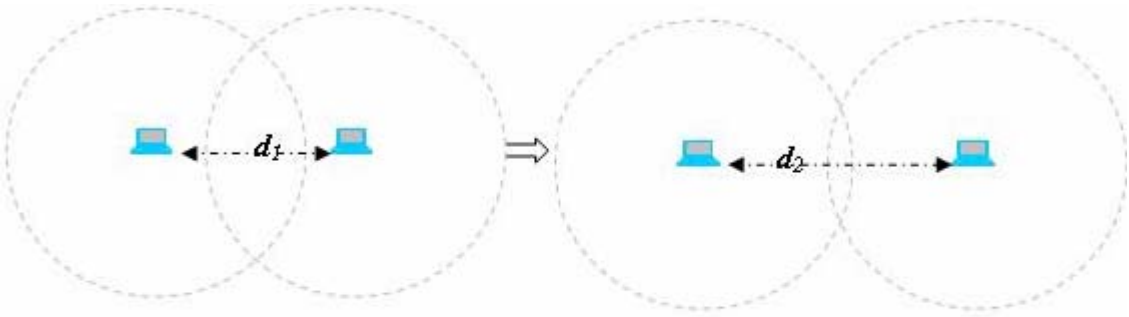

Figure 4.6: Durée moyenne de stabilité d'un lien

En considérant que la distance moyenne parcourue par un nœud avant la rupture d'un lien soit de ***R***, les liaisons stables sont déduites. Ces liaisons seront maintenues lors du délai t_s prochain. Leur détermination nous permet de construire de groupes pour la réplication. Pour une adéquation entre les groupes définis et le processus de réplication, la valeur de la période ***U*** de l'algorithme de réplication ne doit pas remettre en cause la stabilité des nœuds. ***U*** doit donc être approximativement égale à t_s ($U \approx t_s$).

Nous supposons que le délai moyen de propagation d'un message à un saut (sur une distance ***R***) a été évalué sur l'environnement par une valeur t_{moy} (caractéristiques de transmission). Pour déterminer les liens stables, un nœud diffuse un message à un saut. Au bout d'un délai d'attente de $2xt_{moy}$, il déduit l'ensemble de ses connexions stables pour la période courante. Cet ensemble est constitué des nœuds ayant répondu par un accusé de réception au message émis au bout d'un temps ***t*** inférieur à $2xt_{moy}$ (figure 4.7).

La méthode de réplication adoptée est un processus périodique qui exécute en début de chaque période, deux phases:

- une phase de construction des groupes stables sur le réseau,
- une phase de création et de placement de copies selon les besoins des utilisateurs.

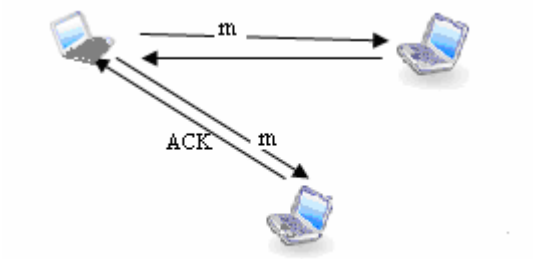

Figure 4.7: Identification de liens stables

4.2.4.2 Construction des groupes

La construction consiste à déterminer en premier, les nœuds stables à un saut et à les rassembler dans des groupes dits *secondaires*. Dans chaque groupe un nœud sera désigné comme chef ou leader (***LeaderSecondaire***). Pour ne pas restreindre les groupes à des groupes à un saut, tel que dans les travaux [Boulkenafed 03] par exemple, nous définissons une structure en groupes hiérarchiques. Les groupes à un saut sont élargis pour constituer des groupes à ***k*** sauts. Ces derniers comporte chacun un leader (***LeaderPrincipal)***. Les leaders secondaires sont éloignés du leader principal d'une distance ***p***=[***k***/2]. Le but est d'implanter des groupes dont les membres sont à une distance inférieure ou égale à ***k***. La construction de groupes se fait en plusieurs étapes (figure 4.8).

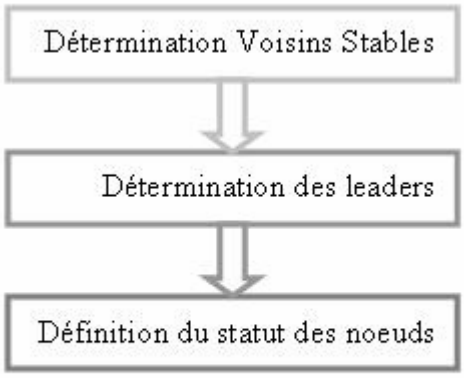

Figure 4.8: Étapes de construction des groupes

▸ Détermination des voisins stables

Chaque nœud ***Ni*** identifie ses liens stables pour constituer la liste des voisins stables VS_i:

- Si $VS_i = \emptyset$, N_i est un groupe *singleton* dont il en est ***LeaderPrincipal***.
- Si $VS_i \neq \emptyset$, N_i envoie un message ***Inf****(C, E, I)* à tous ses voisins qui se trouvent sur des liaisons stables. *C* (Connections) est le nombre de liens stables de N_i. *E*

(Energy) est le niveau d'énergie actuelle de N_i. *I* (Identity) est l'identificateur N_i du noeud.

▸ Détermination des leaders

L'état d'un nœud est défini par le triplet *(C, E, I)*. Chaque nœud N_i effectue un traitement local pour désigner le leader et les membres de son groupe. Pour se faire, il doit :

→ Trier la liste VS_i dans un ordre décroissant selon l'état de chaque membre. L'état est défini par une fonction *F* des variables *C*, *E* et *I*.
F= C/ C_{max} + E/ E_{max}, si C et E dépassent chacune une valeur seuil tolérée respective de connectivité et d'énergie. Sinon, F prend la valeur zéro. E_{max} est le niveau d'énergie maximale chargeable, et C_{max} est un nombre important de connexions possibles pour un nœud.
Les nœuds sont classés selon les valeurs décroissantes de cette fonction. Le nœud N_i est inclus dans la liste VS_i.

→ Si le nœud N_j en tête de la liste triée est différent du nœud N_i, N_i devrait être membre du groupe secondaire G_j dont N_j est le ***LeaderSecondaire*** (N_i passe à l'état ***Member*** du groupe G_j). Il doit effectuer les tâches suivantes:

1) Envoyer à son leader un message ***Freq***() comportant les informations sur ses accès aux données.
2) Envoyer un message ***NotLeader***(id) à tous les nœuds qui sont classés avant lui dans la liste VS_i triée. Il les informe qu'ils ne seront pas choisis comme leaders de sa part.
3) Il vérifier si, à son tour, il a reçu des messages ***Freq***(). Dans ce cas, il est ***LeaderSecondaire*** d'un groupe G_i. Il insère l'identité des nœuds émetteurs dans la liste des membres du groupe G_i dont il est leader.

→ Dans le cas où N_i est en tête de liste VS_i. N_i est leader d'un groupe *principal* GP_i. Il doit déterminer les membres de son groupe principal. Pour cela, il diffuse au membres de son groupe secondaire G_i un message ***LeaderPrincipal***(N_i , ***HopCpt***). Ce message les invite à faire partie de son groupe principal. ***HopCpt*** est un compteur des sauts parcourus par le message (ce compteur est initialisé à zéro). A la première réception de ce message par un membre N_j:

- Il incrémente le compteur ***HopCpt***.
  1. S'il est inférieur à p=[k], il transmet un *ACK* positif à son ***LeaderPrincipal*** N_i.
  2. Si N_j est ***LeaderSecondaire*** d'un groupe G_j à un saut. Il propage le message aux membres de son groupe secondaire, si le compteur est inférieur à p.
  3. Un membre qui reçoit un compteur égal à p va initialiser la construction d'un groupe principal dont il est ***LeaderPrincipal***.

Après cette construction, chacun des nœuds se retrouve dans l'un des états de la table 4.1.

LeaderPrincipal	Nœud leader d'un groupe principal à ***k*** sauts
LeaderSecondaire	Nœud leader d'un groupe secondaire à un saut et membre d'un groupe principal à ***k*** sauts.
Member	Nœud membre d'un groupe secondaire

Table 4.1: Etats d'un nœud

4.2.4.3 Réplication

▸ **Réplication préventive:**

Dans ce cas, la phase de réplication *préventive* devient plus efficace. En effet, le réseau est déjà réparti en groupes à liaisons stables. A la génération d'une donnée *originale*, une copie de celle-ci est juste transmise à chaque ***LeaderPrincipal*** pour la placer sur un des nœuds de son groupe. Il sélectionne le nœud le plus apte à détenir la donnée (espace mémoire, énergie, …). La prise en compte de ces derniers critères constitue un intérêt de supplémentaire par rapport aux méthodes de réplication précédentes.

▸ **Réplication adaptative:**

Chaque ***LeaderPrincipal*** détermine pour son groupe principal, les fréquences d'accès aux données dans un ordre décroissant. Il construit pour chaque nœud une liste des données qu'il doit répliquer localement. Une donnée sera répliquée sur le nœud qui a la plus grande fréquence d'accès à cette donnée s'il y a de l'espace libre. Sinon, le nœud de fréquence suivante est choisi.

4.2.4.4 Accès aux données

Pour l'accès aux données, la procédure adoptée tente de tirer le meilleur profit de l'architecture hiérarchique définie. L'algorithme de localisation et de recherche d'une donnée met à contribution: la procédure de définition des liaisons entre les groupes, la procédure de recherche locale, et celle de la recherche distante.

▸ **Définition des liaisons:**

A la fin de la construction des groupes, les nœuds informent leurs voisins directs de l'identité de leur leader. Un nœud qui reçoit un tel message compare l'identité reçue. Si elle est différente de celle de son propre leader, il déduit qu'il existe une liaison entre les deux leaders. Ainsi, un nœud enregistre les différentes liaisons avec les groupes voisins.

► **Recherche locale de données:**

Une donnée à laquelle veut accéder un nœud peut se trouver dans l'une des situations:

- ✓ La donnée est disponible
  - localement,
  - sur un nœud voisin direct (qu'il soit dans le même groupe ou non),
  - sur un membre du groupe secondaire,
  - ou sur un membre du groupe principal.
- ✓ La donnée est disponible sur un autre groupe principal.
- ✓ La donnée n'est pas disponible.

Un nœud qui génère une requête d'accès, la diffuse à un saut à ses voisins directs. Ces derniers consultent leurs tables locales de chemins d'accès aux données pour répondre à la requête. Si la requête n'est pas satisfaite, le nœud demandeur transmet la requête au ***LeaderPrincipal*** de son groupe principal. Ce dernier recherche la donnée au niveau de son groupe principal. En cas d'échec, le nœud émetteur de la requête lance une requête distante d'accès.

► **Recherche distante de données:**

Le nœud transmet sa requête au ***LeaderSecondaire*** de son groupe. Ce dernier utilise la table des liaisons pour l'acheminer aux leaders des groupes voisins. Un leader qui reçoit la requête:

a. S'il, est un ***LeaderSecondaire***, il la transmet à son ***LeaderPrincipal***.

b. S'il est ***LeaderPrincipal***, il recherche la donnée dans son groupe principal. En cas d'échec, ces leaders vont à leur tour utiliser leur table de liaisons pour diffuser la requête. Un ***LeaderPrincipal*** qui retrouve la donnée, stoppe à son niveau la propagation de la requête. Et, il envoie une réponse au nœud demandeur (chemin d'accès). Si ce chemin est jugé long (>***k***), une réplication est sollicitée par le noeud émetteur. La copie sera placée localement, sur son groupe secondaire, ou sur son groupe principale selon les contraintes en espace et en énergie.

Pour éviter les boucles, la requête contient la liste des groupes parcourus. De plus, une durée de vie représentée par le nombre maximal de sauts possibles est attribuée à chaque requête. Sur le nœud demandeur d'accès, si la requête n'est pas satisfaite au bout d'un certain délai, la donnée est jugée inaccessible.

4.3 *Partage d'accès aux données avec mise à jour*

L'étude de quelques techniques et protocoles de mise à jour de données sur les réseaux mobiles ad hoc met en évidence deux classes principales de protocoles :

- Les protocoles optimistes où, à un instant donné, différentes versions d'une même donnée peuvent exister. Ces protocoles favorisent la disponibilité à la cohérence. Ils supposent que les partitionnements du réseau sont rares afin d'assurer la condition de convergence des copies [Moussaoui 07d][Moussaoui 07b].
- Les protocoles pessimistes qui tentent d'assurer une cohérence stricte aux dépend de la disponibilité. Ces protocoles sont très contraignants. Mais, ils représentent l'unique issue pour assurer une cohérence stricte. Les protocoles à invalidation en sont une catégorie.

Les premiers protocoles pessimistes qui ont existé effectuaient les mises à jour au niveau du serveur *primaire* et propageaient par la suite la donnée modifiée. Dans un protocole à invalidation, la mise à jour d'une donnée rend toutes les autres copies invalides par la diffusion d'un simple rapport d'invalidation. La dernière version de la donnée est chargée au besoin par les serveurs *secondaires,* pour éviter une surcharge inutile en trafic.

Sur les réseaux mobiles ad hoc, cette approche présente quelques problèmes liés surtout à l'utilisation d'un serveur *primaire* qui assure un contrôle centralisé de la cohérence des copies. En effet, une panne, une déconnexion ou un partitionnement peut survenir et priver les nœuds en totalité ou en partie des services du primaire.

Notre contribution consiste en une adaptation du protocole d'invalidation proposé dans [Harar 06] pour la prise en compte des déconnexions du serveur *primaire*. Il faut noter que ce problème constitue la principale et la première critique de ces travaux. Sur un réseau mobile ad hoc la prise en compte des déconnexions doit être à la base de toute solution.

• **Principe:**

Les différentes versions d'une donnée sont distinguées grâce à leur valeur d'estampille de la donnée. Chaque nœud N_i maintient une table $Tstamp_i$ d'estampilles des données. Pour chaque donnée D_k et à chacune de ses mises à jour, son estampille $stamp_k$ est incrémentée. Cette estampille est une date logique qui indique la dernière mise à jour considérée par la copie.

4.3.1 Traitement d'une requête de mise à jour:

Un protocole d'invalidation se base sur le fait que la mise à jour de la copie *primaire* fait passer les copies *secondaires* dans l'état *invalide*. Le principe appliqué pour assurer la cohérence des données sur différents nœuds du réseau est:

- La mise à jour d'une donnée D_k ne peut être faite que par son serveur *primaire*.
- Lorsqu'un nœud veut modifier une donnée, il envoie sa requête au serveur *primaire*. Cette requête contient l'identité du nœud, l'identité de la donnée ainsi que le numéro de séquence de la mise à jour demandée par ce nœud (pour assurer l'ordonnancement des opérations émises par le même nœud).
- Le serveur primaire met à jour la donnée après vérification du numéro de séquence,
- puis diffuse un rapport d'invalidation contenant l'identité de la donnée modifiée, l'identité du nœud ayant demandé la mise à jour ainsi que la date de mise à jour (l'estampille $stamp_k$ après incrémentation).
- Chaque nœud N_i qui reçoit ce rapport d'invalidation, vérifie s'il ne l'a pas déjà reçu en comparant l'estampille du rapport avec celle de sa table $Tstamp_i$.
  - → Si l'estampille dans sa table est supérieure ou égale à celle du rapport, cela signifie qu'il l'a déjà reçu, et il n'en tient pas compte.
  - → Sinon, le nœud met à jour sa table d'estampille $Tstamp_i$, puis vérifie si la donnée est disponible. Si c'est le cas, il invalide sa copie et l'espace qui lui était réservé restera libre jusqu'à ce qu'il récupère la nouvelle version de cette donnée.

Traitement requête de mise de jour():

A la génération d'une requête de mise à jour:

Begin /* Le nœud N_i veut mettre à jour une donnée D_{jk} */
  Si N_i serveur *primaire* de D_{jk} **alors**
    mise à jour D_{jk}
    $stamp_k = stamp_k + 1$;
    mise à jour de $Tstamp_i$[k];
    Diffuser *InvalidReport* (D_{jk}, $stamp_k$) aux nœuds *suivants*;
  Sinon émission requête au serveur *primaire* N_j de la donnée D_{jk}
  Fsi
Fin.

A la réception d'une mise à jour par un *primaire* N_j pour D_{ik}:

Début
  mise à jour de la donnée ;
  $stamp_k = stamp_k + 1$;
  mise à jour de $Tstamp_i$[k];
  Diffuser *InvalidReport* (D_{jk}, $stamp_k$) aux nœuds *suivants*;
Fin.

A la réception d'un rapport d'invalidation de D_{jk} par N_i:

Début
  Si $stamp_k > Tstamp_i$[k] **alors**
    $Tstamp_i$[k] = $stamp_k$ /* Mettre à jour l'estampille de D_{jk} */
    Si une copie de la donnée est disponible **Alors**
      Etat (D_{jk})= invalide
    Fsi
    Diffuser *InvalidReport* (D_{jk}, $stamp_k$) aux nœuds *suivants*.
  Fsi
Fin.

4.3.2 Traitement d'une nouvelle connexion:

La ou les déconnexions d'un nœud peuvent isoler le nœud seul ou avec un certain nombre d'autres nœuds (partition). Le serveur primaire ne peut alors faire parvenir les messages d'invalidation à tous les noeuds. Lors de la détection d'une nouvelle connexion entre un nœud M_i et un nœud M_j, il s'agit de propager par épidémie ces rapports d'invalidation (figure 4.9). La propagation épidémique assure une transmission plus rapide et efficace des rapports antérieurs [Hara 06]. La procédure de mise à jour de la table des estampilles est alors la suivante:

1- Le nœud avec l'identité la plus élevée (j) envoie sa propre table d'estampilles à l'autre nœud.

2- Le nœud M_i compare chaque entrée de la table d'estampille avec celle qu'il a reçu du nœud M_j et la met à jour. Puis les étapes suivantes sont exécutées :

- Le nœud M_i envoie un rapport d'invalidation pour toutes les données dont l'estampille détenue par M_i est inférieure à celle détenue par M_j. Ce rapport est diffusé à tous les nœuds suivants de M_i.
- M_i envoie à M_j les informations concernant les données dont les estampilles sont supérieures à celle de M_j. Le nœud M_j effectue les mêmes opérations que celles décrites précédemment pour M_i.

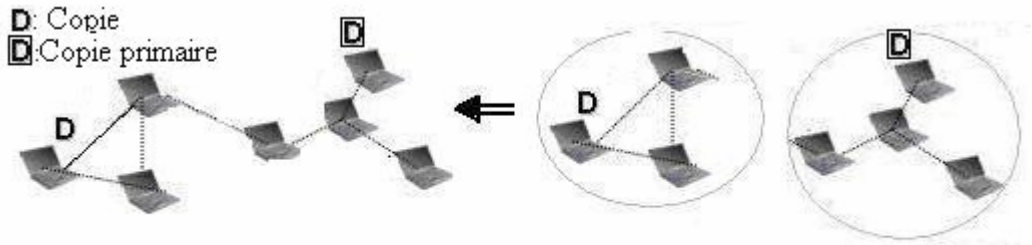

Figure 4.9: Nouvelle connexion de deux noeuds

4.3.3 Traitement d'une requête d'accès:

Plusieurs approches différentes peuvent être considérées pour l'accès aux données. Parmi lesquelles nous citons :

- La première approche consiste à répliquer la donnée dès qu'elle est mise à jour. Elle considère que si un nœud possède une copie, les fréquences d'accès à cette donnée sont élevées. Même si cette méthode améliore les temps d'accès, un nœud hébergeant une donnée peut ne plus en avoir besoin. Un coût important en trafic et en énergie peut s'avérer inutile.

- La seconde approche consiste à évaluer la fréquence d'accès à la donnée (fréquence d'accès internes). Si celle-ci dépasse un certain seuil (***S***), le nœud va demander la dernière mise à jour au *primaire*. Sinon, le chargement de la donnée se fera au besoin.
- La troisième approche qui a été adopté dans [Hara 06] réalise une copie de la donnée mise à jour à la demande. Dans ce cas, la cohérence de la copie obtenue est privilégiée sur sa disponibilité. Les temps d'accès sont alors plus longs puisqu'à chaque accès, il faut faire une copie de la dernière version de la donnée. Cette méthode permet d'éviter un trafic et une consommation d'énergie inutiles. Mais, elle a un impact négatif sur les temps de réponse et sur l'accessibilité car la solution est alors plus sensible aux défaillances du *primaire*.

La seconde approche offre un compromis intéressant et paramétrable par le seuil ***S***. Avec ***S*** très petit, nous nous rapprochons de la première solution. Alors qu'avec de grandes valeurs, la solution tend vers la troisième approche).

4.3.4 Traitement de la déconnexion du serveur primaire:

Avoir recours à un serveur *primaire* permet d'assurer la cohérence des données. En effet, une demande de mise à jour passe par le *primaire* qui contrôle toute modification de la donnée. A tout moment, un nœud N_i sait si sa donnée est la dernière version ou non grâce au rapport d'invalidation et à la table d'estampille $Tstamp_i$. Cependant, il peut arriver qu'un serveur *primaire* tombe en panne ou se déconnecte. Cela empêchera toute nouvelle modification des données du serveur et immobilisera ainsi plusieurs nœuds. Pour pallier à ce problème, nous proposons une méthode qui fait en sorte qu'il y ait toujours un serveur *primaire* dans le réseau même en cas de déconnexion.

Sur chaque nœud mobile, une procédure périodique détecte les paramètres de risque de déconnexion du primaire (niveau d'énergie et nombre de connexions stables). Nous supposons deux seuils pour ces paramètres. Un premier seuil de déconnexion SD_1 où le serveur *primaire* détecte un risque de déconnexion, et un second SD_2 où la probabilité de ce risque devient importante.

Lorsqu'un serveur primaire atteint le premier seuil, il diffuse cette information. Chaque serveur qui reçoit le message, vérifie s'il remplit les conditions (sa fonction sur l'énergie disponible et le nombre de connexions stables possède une valeur supérieure à SD_1) pour remplacer le serveur *primaire*.

Si c'est le cas, il envoie son état au *primaire* (son identité, ses propriétés ainsi que la liste des données du *primaire* dont il est aussi serveur). Le *primaire* reçoit les informations des différents serveurs. Lorsqu'un serveur *primaire* N_i atteint le seuil SD_2:

- Il diffuse un message ***Deconnect***(N_i) sur le réseau pour avertir les nœuds du réseau qu'il risque de se déconnecter. Ce message est nécessaire dans le cas où un nœud

prévoit d'envoyer une demande de mise à jour ou un accès à distance à la donnée du *primaire* au moment où il va se déconnecter.

- Pour chaque donnée D_{ik} dont il est serveur *primaire*, N_i envoie un message ***Désigner***(N_i,D_{ik}) au serveur qui a l'état le plus sûr. Ce message contient la donnée afin que le nouveau *primaire* ait la dernière version.
- Á la réception du message, ce serveur devient *primaire* de la donnée reçue. Puis il diffuse l'information sur le réseau. L'identité de cette donnée restera la même, car l'information sur la déconnexion d'un *primaire* peut ne pas parvenir à un nœud. Si ce nœud désire plus tard accéder à la donnée, il ne trouvera pas le *primaire* d'une part, et d'autre part cette donnée n'existerait plus (son identité a été modifiée).

  Il faut informer les autres nœuds que leurs requêtes de mise à jour doivent s'adresser à un nouveau *primaire*. Pour cela, une variable supplémentaire associée à la description de chaque donnée indiquera le *primaire* actuel de la donnée.
- S'il reçoit une requête de mise à jour une fois le seuil SD_2 atteint, il redirige la requête vers le nouveau primaire qui accusera réception au nœud émetteur initial.

Procédure de déconnexion du primaire :

1 - Lorsque le primaire atteint le premier seuil :

Lorsque qu'un serveur N_i primaire atteint le seuil SD_1:

Début Diffuser message ***Deconnect***(N_i, 1) /* probable déconnexion */

Fin.

Lorsqu'un serveur reçoit ***Deconnect***(N_i, 1):

Début

Si première réception **alors**

Si capacité énergétique et liaisons satisfaisantes **alors**

Envoyer ***Etat***(identité, énergie, connexions stables) au *primaire*

Fsi

Fsi

Fin.

Lorsqu'un primaire N_i reçoit l'état d'un serveur N_j :

Début

Pour chaque donnée de N_i dont N_j est serveur

Faire **Si** N_i a déjà reçu une réponse pour cette donnée

Alors comparer les états des serveurs

Stocker le meilleur état.

Sinon stocker état de N_j

Fsi

Fait

Fin.

2 - Lorsque le primaire atteint le deuxième seuil :

Lorsqu'un serveur primaire N_i détecte sa future déconnexion:

Début

Diffuser ***Deconnect***(N_i,2) aux nœuds du réseau /* Déconnexion */

Pour chaque donnée dont il est primaire

Faire Envoyer ***Désigner***(N_i,D_{ik}) à N_j /* N_j Nouveau primaire */

Fait

Fin.

Lorsqu'un nœud reçoit ***Deconnect*** (N_i, 2):

Début
Si la donnée D_i est locale au nœud **Alors**
Si le nœud veut effectuer une opération sur D_i **Alors** attente()
Fsi
Fin.
Lorsque le nœud N_i reçoit ***Désigner***(N_i, D_{ik}) du primaire N_j :
Début /* N_i Nouveau serveur Primaire */
Diffuser ***Nou_Prim***(N_i, D_{jk}) à tous les nœuds du réseau .
Fin.

4.3.5 Traitement du partitionnement

Lors d'un partitionnement du réseau, le serveur *primaire* peut avoir changer selon les conditions précédentes. Une des partitions ignore alors l'existence du nouveau *primaire*. Pour pallier à ce problème, à la reconnexion de deux nœuds, en plus de vérifier les tables d'estampilles, l'identité du serveur primaire de chaque donnée est vérifiée. Pour se faire, un numéro $NumP_k$ est associé au serveur *primaire* de chaque donnée D_k. A la désignation d'un nouveau primaire pour D_k, $NumP_k$ lui est affectée après incrémentation. A la connexion de deux nœuds, le numéro de *primaire* le plus grand représente l'identité du *primaire* actuel.

4.3.6 Prédiction de déconnexion d'un primaire

Nous avons abordé, plus haut, les notions de seuil SD_1 et SD_2. Ces seuils correspondent en fait aux potentialités d'un nœud en disponibilité d'énergie et qualité des connexions. La première caractéristique ne dépend que du nœud. Par contre, la deuxième dépend des voisins du *primaire* et de leur mobilité. La qualité de la connectivité du *primaire* est définie par un nombre nécessaire minimal de connexions stables.

4.3.7 Problème des mises à jour en cas de partitionnement:

Ce protocole d'invalidation permet la mise à jour de données sur un réseau mobile ad hoc en assurant un degré élevé de cohérence. Le maintien de cette cohérence passe nécessairement par une certaine centralisation du contrôle représenté par le nœud primaire.

La solution CR [Hara 06] est une solution pour les réseaux mobiles ad hoc mais qui ne considère ni la déconnexion du primaire ni le partitionnement du réseau. Ces deux problèmes peuvent être abordés comme suit:

- ✓ La déconnexion du primaire implique que le réseau se retrouve sans nœud primaire. Dans ce cas, non seulement les requêtes de mises à jours restent sans réponse mais aussi, les requêtes d'accès en lecture à une version récente ne sont

plus satisfaites. On peut dire que l'accessibilité à la donnée se trouve alors sérieusement pénalisée. D'où l'apport de la contribution présentée à travers la prédiction et le traitement de la déconnexion du primaire.

- ✓ Le partitionnement du réseau est un autre problème qui se résume en la présence d'au moins deux partitions du réseaux. C'est-à-dire, deux ensembles de nœuds séparés par l'absence de connexions les reliant. Dans ce cas, il n'y a pas déconnexion du *primaire*. Ce dernier continue à assurer correctement sa fonction pour les nœuds de sa partition. Mais, les nœuds de la deuxième partition sont privés de ses services. Le problème est que ces nœuds ne reçoivent plus les rapports d'invalidation:

  a. S'ils détiennent une copie avec la dernière estampille enregistrée, ils accéderont à la donnée en tant que version actuelle alors qu'elle peut avoir été modifiée. D'où la mise en cause de la cohérence.

  b. Si la copie d'un nœud est déclarée invalide parce qu'elle ne correspond pas à la dernière estampille reçue, un noeud peut vouloir accéder à la donnée. Dans ce cas, sa requête pour récupérer la dernière version n'atteindra pas le *primaire*.

La proposition d'une solution à ce problème constitue une perspective intéressante d'extension du protocole d'invalidation. Nous pouvons proposer les premières idées suivantes selon deux situations possibles envisagées:

1. Maintenir une cohérence forte: Une solution serait qu'à chaque accès à la donnée, le serveur secondaire s'assure de l'existence du primaire dans sa partition. Dans le cas négatif, l'accès est refusé. On observe à quel point la cohérence forte peut être contraignante sur les environnements mobiles. Toutefois, il y a des applications critiques qui peuvent exiger ce degré de cohérence.

2. Tolérer un relâchement de la cohérence: Il existe beaucoup d'applications dites optimistes qui tolèrent une cohérence non stricte. Le degré de cohérence accepté peut varier d'une application à une autre. Il peut être défini à travers la latence permise (délai maximal admis pour la convergence de copies différentes).

Notre perspective dans ce cas est de tenter d'apporter une réponse à la précédente question (2.) par le développement d'un protocole à deux niveaux de cohérence. Il adopte alors deux comportements:

(i) Un comportement de protocole d'invalidation à cohérence forte au niveau d'une partition.

(ii) Un comportement optimiste à cohérence plus faible entre partitions différentes. Dans ce cas, à la détection de l'absence du serveur primaire dans une partition, un nouveau primaire est désigné (par une procédure

d'élection qui considère la fonction d'aptitude définie plus haut). Chaque partition dispose alors de son serveur *primaire*. Cette définition ne correspond plus tout à fait à la définition d'un serveur *primaire*. Nous pourrions, par exemple, le désigner par *PrimaireP* (serveur Primaire de Partition).

L'intérêt est que l'accessibilité au niveau d'une partition est améliorée par la présence d'un *primaire* (ou substitut de *primaire*: *PrimaireP*) qui assure les fonctions nécessaires aux accès. Il est alors clair qu'il ne s'agit plus de cohérence stricte ni de protocole pessimiste. Le nombre d'applications optimistes est beaucoup plus important que le nombre des applications pessimistes (application critiques qui exigent une cohérence forte). Cette nouvelle approche serait donc avantageuse. Il faut noter, que sans ce nouveau primaire la cohérence n'était pas respectée non plus.

Les nœuds des différentes partitions peuvent alors continuer de fonctionner en dépit des partitionnements. Cependant, l'existence de plusieurs primaires pour une même donnée, suppose des manipulations concurrentes de la donnée. Les copies des primaires divergent. A la reconnexion de partitions différentes, un protocole de réconciliation est indispensable. Un protocole de réconciliation épidémique est alors envisageable et intéressant pour considérer le passage à l'échelle [Terry 95].

Les serveurs primaires devront enregistrer dans un journal les opérations de mises à jours. Une mise à jour sera estampillée par une horloge physique unique si un GPS est disponible. Dans le cas contraire une horloge logique vectorielle ne poserait pas de problème de passage à l'échelle car le nombre maximal de partitions pourra être borné par hypothèse.

Conclusion

La connectivite dans les environnements mobiles est de nature intermittente (discontinue), ce qui rend inaccessible des données nécessaires aux utilisateurs mobiles. De ce fait, la réplication des données au niveau des nœuds mobiles devient inévitable. Elle renforce la disponibilité des données. Mais, elle doit prendre en compte, en plus des besoins des utilisateurs, la disponibilité des moyens au niveau des terminaux mobiles. En effet, une réplication massive ou systématique des données compromet les ressources de stockage réduites au niveau de certains types de machines mobiles.

Les communications sans fil, multi-sauts, mises à contribution pour l'accès aux données augmentent la consommation d'énergie des mobiles émetteurs, des mobiles destinataires et aussi des nœuds intermédiaires impliqués dans le routage. Le placement des données doit

également prendre en compte la nature discontinue des connexions. Les stratégies de réplication doivent pouvoir s'adapter continuellement aux variations dynamiques de la configuration.

L'accès aux données sur un réseau mobile ad hoc est une tâche non évidente. En particulier, lorsque le but est d'assurer aux nœuds des performances satisfaisantes. Pour répondre à ces préoccupations, nous avons considéré, dans une première partie l'accès aux données sans mises à jour. La mise à jour des données étant un problème complexe mais aussi très important, nous avons proposé un protocole d'invalidation adapté aux réseaux mobiles ad hoc en deuxième partie du chapitre.

Un nouveau schéma de réplication à la fois préventive et adaptative est proposé pour la dissémination et l'accessibilité aux données. La phase préventive tente de maintenir des chemins d'accès à une donnée malgré les déconnexions. Elle procède par une dissémination uniforme des copies de données à la création. Cette répartition a certes un coût mais les algorithmes de détection du partitionnement qui pourraient constituer une alternative, ont aussi un coût non négligeable. D'autre part, la prédiction des déconnexions n'est pas toujours assurée et ne résiste pas aux pannes imprévisibles.

La phase de réplication adaptative est un algorithme périodique qui s'exécute régulièrement afin de mieux répondre aux exigences des utilisateurs malgré les changements dynamiques de la topologie et le mouvement des nœuds. L'algorithme doit en permanence réajuster et adapter la localisation des données selon la dernière configuration du réseau, et selon la demande courante en données des utilisateurs.

Quatre méthodes basées sur cette approche ont été présentées: ***HBR***, ***IBR***, ***TBR*** et ***GBR***. La première méthode considère des données importantes avec une localisation des copies à une distance de ***k*** sauts. Cette méthode a pour avantage de prévenir les déconnexions et de maintenir des chemins d'accès malgré d'éventuels partitions précoces. L'inconvénient est que ***HBR*** peut produire une charge supplémentaire inutile si la donnée n'est pas d'une importance sûre pour les mobiles ou si le réseau soufre déjà de congestion.

Une première réponse se trouve dans la valeur fixée pour ***k*** à la génération d'une donnée. La méthode ***IBR*** vient apporter une amélioration, par des traitements différents d'une donnée selon son niveau d'importance. Pour les données jugées pas très importantes, des descripteurs seront disséminés sur le réseau pour informer les nœuds de l'existence de la donnée et de sa localisation. Cette dissémination contribue aussi à la construction des *PathData* ou chemins d'accès aux données.

Les deux premières méthodes définissent les besoins en données d'un utilisateur à travers les fréquences d'accès. Cette représentation des besoins d'un utilisateur est communément utilisée dans les stratégies de remplacement des systèmes de gestion d'espaces de stockage. Nous avons distingué les fréquences d'accès internes des fréquences d'accès externes. Aussi,

notre contribution à travers ***TBR*** a été de tenir compte des temps de réponse pour un choix plus optimal des nœuds hôtes de copies. Un temps de réponse long véhicule indirectement des informations sur la position de la donnée et sur la qualité des voies de communication empruntées pour l'atteindre. Cette méthode accorde un privilège aux données utilisées et à temps de réponse long.

Une quatrième méthode ***GBR*** propose une architecture hiérarchique en groupes de nœuds de communication. L'approche topologie logique en groupes est très étudiée actuellement par les travaux de recherche sur le passage à l'échelle des réseaux mobiles ad hoc. Un certain nombre de travaux ont prouvé son efficacité.

Le problème de mise à jour de données partagées a été abordé. Nous avons proposé un protocole d'accès aux données mises à jour. Ce protocole est de type protocole à invalidation adapté aux réseaux mobiles ad hoc. Il traite, en particulier, la déconnexion du serveur primaire. La solution a été de prédire la déconnexion du primaire et de permettre son remplacement. Cette prédiction n'est pas étendue au cas de panne imprévisible du primaire. L'exploration de cette situation constitue une des perspectives. Le problème de cohérence de données en présence de partitionnement a été étudié. Les grandes lignes d'un protocole adapté à cette situation ont été exposées. Nous espérons le finaliser dans un futur proche.

Dans le chapitre suivant, nous verrons quelques résultats obtenus lors de la simulation de ces méthodes.

Chapitre 5

Simulation et Evaluation des Performances

Introduction

Il y a deux approches pour évaluer un système sur réseau mobile ad hoc. La première utilise les mesures d'expérimentation et la seconde se base sur la représentation du comportement du système via un modèle [Kurose 88]. Les techniques d'expérimentation sont appliquées sur des environnements réels ou des prototypes. Actuellement, peu de tests et de mesures d'évaluation sont effectués sur des environnements mobiles ad hoc réels [Broch 00]. En effet, il serait très coûteux voir impossible de mettre en place un réseau à des fins de tests. L'expérimentation reste limitée dans les possibilités de variation de scénarios, comme par exemple, la variation du modèle de mobilité. D'autre part, les mesures expérimentales ne peuvent pas être répétées en conservant exactement les mêmes conditions. L'étude du passage à l'échelle, de la variation de la vitesse et du modèle de mobilité des utilisateurs est difficile à réaliser. Pour tester un système pour ces environnements nous avons souvent recours à la simulation.

L'utilisation d'un simulateur ou d'un modèle analytique permet la variation des paramètres de l'environnement en considérant un large éventail de situations possibles. A ce jour, l'analyse à l'aide de la modélisation mathématique a apporté un aperçu sur le fonctionnement d'un certain nombre de systèmes informatiques. Cependant, plusieurs systèmes sont si complexes qu'ils sont pratiquement impossibles à résoudre par un modèle mathématique. Parmi ces systèmes s'inscrivent les systèmes mobiles caractérisés par la complexité de leur environnement physique [Obaidat 03]. Les méthodes analytiques sont souvent immatures et non suffisamment détaillées en terme de mobilité.

La simulation offre un outil standard et flexible pour l'évaluation. Elle donne une vue assez précise sur les caractéristiques fondamentales et les comportements d'un système mobile. A l'aide du développement d'un modèle de simulation, il est possible de prédire les comportements d'un protocole dans des situations difficiles à réaliser par une expérience compte tenu de la complexité des environnements mobiles.

L'évaluation des performances d'un système via une simulation consiste en: (i) le choix d'un modèle, (ii) l'évaluation par une technique de simulation et, (iii) l'interprétation des mesures recueillies. Un grand nombre de modèles de simulation ont été développés pour l'étude d'architectures et de protocole sous divers scénarios réseaux (nombre de nœuds, mobilité, …). Ils ont été largement utilisés pour l'évaluation des protocoles de routage.

Les réseaux mobiles peuvent être simulés à l'aide de langages de programmation à utilisation générale tels que C, C++ et JAVA ou de langages de simulation tels que MODSIM III, SIMSCRIPT II.5 et SLAM II. Dans la seconde moitié des années 90, avec l'élaboration de plusieurs normes pour les réseaux sans fil à portée limitée, un certain nombre de simulateurs ont été développés conjointement. On cite, par exemple, Network Simulator 2 [NS-2], OPNET [OPNET] et GloMoSim [GloMoSim]. NS-2 est certainement le simulateur de

réseaux le plus utilisé par la communauté de chercheurs travaillant sur les environnements mobiles sans fil. Cependant, notre choix s'est porté sur GlomoSim ("Global Mobile system Simulator"). Ce simulateur a été conçu spécifiquement pour les réseaux mobiles ad hoc ce qui doit certainement lui procurer une meilleure adéquation à ces environnements. GloMoSim est un simulateur totalement gratuit disponible sur Internet.

Dans ce chapitre nous allons évaluer les méthodes de réplication de données proposées. Les métriques les plus importantes sont évaluées en faisant varier plusieurs paramètres tels que : la surface, le nombre de nœuds, la vitesse, etc. Pour les méthodes de réplication sans mise à jour, nous avons pu les comparer avec la méthode ***DCG-S1*** [Hara 01] [Hara 03]. Ce choix provient du fait que nos solutions sont inspirées de ces travaux. Et, ***DCG-S1*** est une des méthodes de T. Hara les plus satisfaisantes en performances.

5.1 Le simulateur GloMoSim

GloMoSim a été développé à l'UCLA ("University of California at Los Angles") pour le projet de communication mobile globale (GloMo) de DRAPA (Defences Advanced Research Projects Agency) [GloMoSim]. Ce simulateur est extensible. Il est conçu sous forme d'un ensemble de modules [Nk 99][UCLA]. Comme la majorité des simulateurs réseaux, GloMoSim adopte une architecture en couches du réseau semblable à celle du modèle OSI.

GloMoSim utilise le langage PARSEC ("Parallel Simulation Environment for Complex Systems"). C'est est une bibliothèque de simulation d'évènements parallèles discrets basée sur le langage C. Le noyau de la simulation PARSEC permet à GloMoSim d'avoir une grande vitesse d'exécution et un bon passage à l'échelle. GloMoSim supporte différents modèles de mobilités des nœuds. Plusieurs protocoles de différentes couches (réseau, application, liaison de données, transport) y sont déjà implémentés. La figure 5.1 représente la pile des couches de protocoles implantés sur le simulateur. GloMoSim est conçu comme un ensemble de modules. Chaque module simule un protocole spécifique de communication sans fil [**Erreur ! Liaison incorrecte.** 00]. Il a été développé de façon à pouvoir facilement étendre des protocoles existants et, à en ajouter de nouveaux.

Figure 5.1:Protocoles implantés sur GloMoSim

5.1.1 Modèles de mobilité et Paramètres du simulateur

Au début de la simulation, des paramètres de configurations sont injectés au modèle de simulation à partir d'un fichier d'entrée nommé config.in. Parmi ces paramètres:

- SIMULATION-TIME **:** représente le temps maximum d'exécution de la simulation.
- TERRAIN-DIMENSIONS(X, Y) : représente les dimensions du terrain de simulation.
- NUMBER-OF-NODES : permet de préciser le nombre de noeuds dans le réseau à simuler.
- NODE-PLACEMENT et NODE-PLACEMENT-FILE: décrivent les emplacements des nœuds.
- MOBILITY **:** indique le modèle de mobilité adopté pour les déplacements des noeuds :
  - TRACE: Dans ce cas, le mouvement des nœuds est lu à partir d'un fichier.
  - RANDOM-DRUNKEN: A intervalle régulier, le noeud peut se déplacer vers l'une des positions voisines. Si la position actuelle du noeud est (x,y) le mouvement peut se faire vers l'une des destinations suivantes: (x+1,y), (x-1,y), (x,y+1) ou (x,y-1).
  - MOBILITY RANDOM-WAYPOINT: Dans ce modèle, le nœud choisit aléatoirement une destination vers laquelle il se déplacera avec une vitesse calculée entre la vitesse minimum et une vitesse maximum. Quand le noeud atteint cette position, il s'y installe pendant un temps de pause avant de migrer vers une nouvelle destination.
  - MOBILITY NONE: Aucune mobilité, les noeuds sont immobiles et gardent leurs positions initiales.

5.2 Environnement de simulation

Parmi les modèles qu'offre GloMoSim pour la simulation de la mobilité des nœuds, nous avons choisi le modèle RWP ("Random Way Point"). C'est le modèle le plus communément utilisé dans la simulation des réseaux mobiles ad hoc étant donné l'imprévisibilité du mouvement des nœuds dans un tel environnement [Bettstetter 01] [Camp 02] [Naoumov 03].

La table 5.1 résume les principaux paramètres de configuration de l'environnement de simulation. Nous étudions l'impact des variations de ces paramètres sur les méthodes proposées. La simulation adoptée modélise un réseau de 50 nœuds. Initialement les nœuds sont placés aléatoirement sur une surface de 1000m x 1000m. Le mouvement des nœuds est distribué selon le modèle RWP. La vitesse de déplacement d'un nœud varie entre une valeur minimale de 1 m/s et une valeur maximale de 20m/s. La vitesse maximale de déplacement et le temps de pause définissent le niveau de mobilité du modèle. Pour créer un réseau mobile ad hoc modérément mobile, le temps de pause est fixé à 5 secondes. Nous supposons que toutes les données sont de même taille et, que tous les nœuds ont une même taille mémoire. L'espace d'un nœud est fixé pour contenir 50 données.

La capacité du canal est de 2 Mbit/s. Nous considérons des nœuds dont la portée de communication est la même pour chacun. Cette portée recouvre une région dans un cercle de rayon ***R***. Nous simulons différentes densités du réseau. La densité en nœuds est variée en augmentant le nombre de noeuds entre 5 et 250 sur un espace fixe de 1000m x 1000m. La variation de la densité du réseau et la variation du rayon ***R*** de transmission (50m à 250m) influence le degré de connectivité. Le passage à l'échelle est étudié en augmentant le nombre de nœud jusqu'à 250 nœuds en même temps que la surface qui atteint 5000m x 5000m. Chaque simulation s'exécute pendant 15x60 secondes.

Paramètres	**Valeur**	**Intervalle**
Surface (m^2)	1000^2	1000^2 - 5000^2
Nombre de noeuds	50	50 - 250
Nombre de données	50	
Taille de données (KB)	1	1 – 4
Taille mémoire (KB)	20	5 – 50
Vitesse maximale (m/s)	10	1 – 20
Temps de pause (s)	5	
Seuil de fréquence S (%)	0.5	0.01– 0,9
Bande passante (Mbps)	2	
Intervalle de création de données (s)	60	1 - 60
Intervalle de génération de requêtes (s)	60	1 - 60
Portée Radio R (m)	50	50 - 250
Période de réplication U (s)	200	50 - 200
Temps de simulation (s)	15*60	

Table 5.1: Paramètres de simulation

Les performances du système dépendent des proportions de ***U***, ***R*** et de la mobilité d'un nœud [Hara 02b]. La génération des requêtes de création et d'accès effectue un choix aléatoire du nœud source de la requête. Ce processus suit un modèle de POISSON pour la modélisation d'évènements aléatoires.

La fréquence d'accès est calculée en utilisant la relation exponentielle de moyenne mobile ("moving average") [**Erreur ! Liaison incorrecte.** 87] [**Erreur ! Liaison incorrecte.** 96] qui est communément utilisée pour les analyses techniques. Elle est utilisée pour calculer une valeur moyenne sur une période donnée. La fréquence d'accès est alors calculée par la formule:

$$MAfij = \beta.MAfij^* + (1-\beta).fij,$$

Tel que:

MAfij **:** moyenne mobile de la fréquence d'accès à la donnée D_j par le nœud N_i pour la nouvelle période.

MAfij* **:** moyenne mobile de la fréquence d'accès à la donnée D_j par le nœud N_i pour l'ancienne période.

fij : fréquence d'accès du noeud N_i à la donnée D_j pour la nouvelle période.

β : est une constante de lissage ("smoothing constant"); dans notre simulation ***β***=0.5**.** Avec cette valeur, l'historique des accès de chaque nœud a autant d'impact sur les accès futurs que les nouvelles fréquences d'accès obtenues.

MAfij est calculée à chaque unité de temps ***U***. Si la valeur de ***MAfij*** dépasse le seuil ***S***, La donnée correspondante doit être répliquée. Le processus d'évaluation des temps moyens de réponse aux requêtes est similaire à l'évaluation de la valeur moyenne des fréquences d'accès.

5.3 Paramètres d'évaluation

Les méthodes proposées sont évaluées selon les principales mesures: taux d'accessibilité, charge supplémentaire en trafic, taux d'accès invalides, … etc.

5.3.1 Taux d'accessibilité aux données

Ce paramètre représente le taux de succès des requêtes d'accès aux données. Il est calculé par la formule suivante :

$$Acc = \frac{SuccR}{AccR},$$

où, ***Acc*** est le taux d'accès réussis, ***SuccR*** est le nombre de requêtes satisfaites ou réussies durant la simulation, et ***AccR*** est le nombre de requêtes formulées durant la simulation.

L'objectif de toute stratégie de réplication est d'offrir un taux d'accessibilité aux données le plus satisfaisant possible. Une requête d'accès réussit si le nœud source arrive à atteindre la donnée à laquelle il veut accéder. Une requête échoue si aucune réponse positive n'est reçue par le nœud émetteur au bout d'un délai d'attente fixé par le système.

Pour le protocole d'accès et de mise à jour de données, nous étudierons toujours ce paramètre d'accessibilité. Mais dans ce cas, la définition de l'accessibilité est reprécisée comme sut: c'est le nombre d'accès réussis sur le nombre totale de requêtes d'accès émises. Un accès réussie est une requête d'accès où le demandeur reçoit la donnée (les accès réussis englobent les accès à des versions valides ou invalides).

5.3.2 Trafic généré

A travers ce paramètre la charge supplémentaire générée par la réplication est estimée en nombre de messages (à un saut) circulant dans le réseau durant toute la simulation.

La formule de calcul du trafic est la suivante :

$$T = \sum_{i=0}^{n} T_i,$$

T_i est le nombre de messages transmis par un nœud i durant toute la durée simulation. *n* est le nombre de nœuds ayant participé à la constitution du réseau.

5.3.3 Mises à jour réussies

Ce paramètre représente le nombre de requêtes de mises à jour effectuées par rapport au nombre total de requêtes de mise à jour générées. Une mise à jour effectuée est une requête reçue et réalisée par le *primaire*.

$$MajR = \frac{NbMajE}{NbMajT} * 100$$

Où, *MajR* est le pourcentage de requêtes de mises à jour réalisées par le *primaire*, *NbMajE* est le nombre de mises à jour effectuées, et *NbMajT* est le nombre total de mises à jour générées.

5.3.4 Accès invalides

Ce paramètre représente le nombre d'accès invalides par rapport au nombre d'accès réussis (il s'agit là d'un pourcentage). Un accès invalide est une requête qui réussit à accéder à la donnée mais la copie utilisée est une version antérieure à celle du primaire. La formule pour le calcul du pourcentage d'accès invalides est la suivante :

$$IA = \frac{NbIA}{SuccR} * 100$$

Où, *IA* est le pourcentage d'accès invalides (Invalid Access), *NbIA* est le nombre d'accès invalides, et *SuccR* est le nombre de requêtes satisfaites ou réussies durant la simulation (Succeeded Requests).

5.4 *Résultats de Simulation*

5.4.1 *Méthodes de partage d'accès sans mise à jour*

Dans ce paragraphe nous présentons les résultats de simulation observés pour les méthodes d'améliorations de l'accessibilité sans prise en compte des mises à jour. Les méthodes ***HBR***, ***IBR***, ***TBR*** et ***GBR*** sont évaluées à travers deux paramètres: Accessibilité (taux de requête d'accès réussies) et trafic généré.

5.4.1.1 Effet de variation du paramètre *k*

Sur la figure 5.2, nous observons l'effet de la variation du paramètre de réplication ***k*** sur l'accessibilité et sur le trafic. Les résultats montrent que l'accessibilité décroît avec l'accroissement des valeurs de ***k*** au-delà d'une certaine valeur. Ce résultat est valable pour les trois méthodes ***HBP***, ***IBR*** et ***TBR*** à des taux de décroisement légèrement différents.

L'accessibilité aux données est relativement bien maintenue dans la majorité des cas même pour la méthode ***IBR*** dont la réplication préventive est sélective. L'explication se trouve certainement dans l'action complémentaire des deux phases: la phase préventive et la phase adaptative. La phase de réplication adaptative recouvre le manque en copies préventives. La courbe d'accessibilité prend une allure décroissante pour les valeurs croissantes de ***k***. Cependant, on remarque que l'accessibilité reste satisfaisante. Dans ce cas, il est vrai que la réplication préventive distribue moins de copies sur le réseau. Mais la réplication adaptative observe les accès des utilisateurs pour déduire leurs besoins en données. Une donnée jugée utile sera rapprochée de son utilisateur. La deuxième phase est basée sur d'autres paramètres que ***k*** telle que la fréquence des accès.

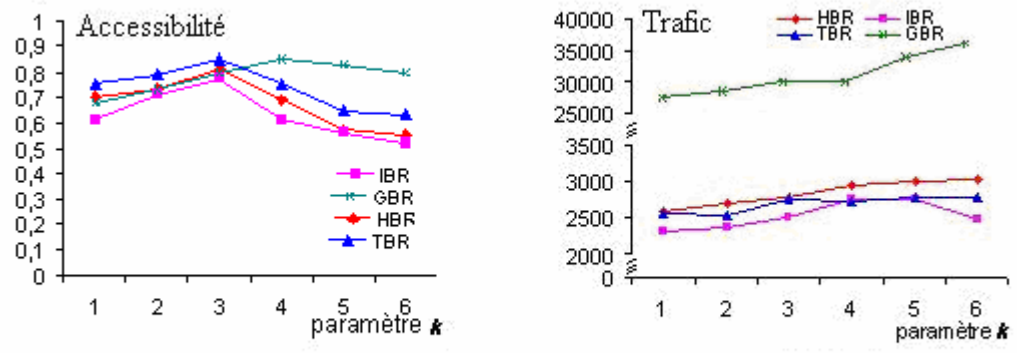

Figure 5.2: Effet de variation du paramètre ***k***

Lorsque la valeur de ***k*** est très petite, l'accessibilité n'est pas optimale alors que la condition de réplication préventive place des copies très proches les une des autres (à un ou deux sauts). Bien qu'une accessibilité très forte soit prévue, la courbe indique le contraire. Cette observation peut être expliquée facilement, par le fait que la surcharge du réseau avec les mêmes données, produit un manque en espace pour les autres données. Dans le voisinage d'un nœud le nombre total de données accessibles devient limité. L'accessibilité globale est donc réduite.

Plus la valeur de ***k*** croît, plus les requêtes d'accès subissent des échecs. Car le nombre de copies sur le réseau est faible et ces dernières sont éloignées. Les partitions du réseau arrivent plus facilement à isoler une partie des données.

Nous remarquons que l'accessibilité offerte par ***IBR*** est moins importante que celle des autres méthodes. Et, ***TBR*** présente un résultat en accessibilité légèrement meilleur que ***HBR***. Ceci peut est dû au fait que ***TBR*** considère aussi les temps de réponse. Cette méthode place près de l'utilisateur, les copies des données qui présentent des temps de réponse longs. Ces données se situent souvent à un nombre de sauts important. La distance est un facteur de rallongement des délais de réponse. Elle favorise aussi la rupture des chemins d'accès.

GBR offre une accessibilité moins élevée pour les petites valeurs de ***k.*** Elle devient plus intéressante pour de grandes valeurs de ***k***. Nous pouvons interpréter cette observation par le fait que pour de petites valeurs de ***k***, la taille des groupes est à son tour réduite. Dans un voisinage de même dimension, ***TBR***, par exemple, arrive à placer plus de copies. ***GBR*** doit en plus assurer la stabilité des liens. Les données sont répliquées sur des nœuds à liens stables.

Le surcoût en trafic n'est pas sensible aux variations des valeurs de ***k*** pour les trois premières méthodes. En effet, ceci pourrait être le fait que la réplication importante pour les petites valeurs de ***k*** produit un "overload" en messages qui est compensé par la suite par plus d'accès locaux ou proches. ***GBR*** produit un trafic plus important à cause du coût de construction et d'entretient des groupes. Le surplus en trafic est l'effort à fournir pour bénéficier des meilleurs taux d'accessibilité pour les grandes valeurs de ***k***.

5.4.1.2 Effet de variation du seuil *S*

Un noeud mobile réplique une donnée si sa fréquence d'accès dépasse la valeur définie pour le seuil ***S*** (figure 5.3). Le but de ces mesures est d'évaluer l'effet du choix de la valeur de ***S***. Cette valeur dépend du type de l'application et de l'environnement. Une étude préalable à toute implantation est nécessaire pour définir la valeur appropriée de ***S.***

Les résultats montrent que lorsque la valeur de ce seuil augmente, le taux d'accessibilité décroît. Il finit par se stabiliser au-delà d'une certaine valeur de ***S***. Lorsque ce seuil est bas, la

majorité des fréquences d'accès le dépassent. Les données respectives présentent alors une accessibilité importante. Dans le cas contraire, les données sont plus rarement répliquées et plus de requêtes seront transmises à distance. Ces requêtes ont plus de chances d'échouer.

Nous observons que le trafic augmente légèrement avec l'accroissement des valeurs de ***S*** (figure 5.3). Pour de petites valeurs de ***S***, un maximum des données utilisées sont répliquées. La présence des données les plus utilisées localement ou sur son voisinage réduit le coût en messages des requêtes d'accès. Une telle évaluation permet de fixer une valeur pour le seuil ***S*** selon l'accessibilité requise et le coût toléré.

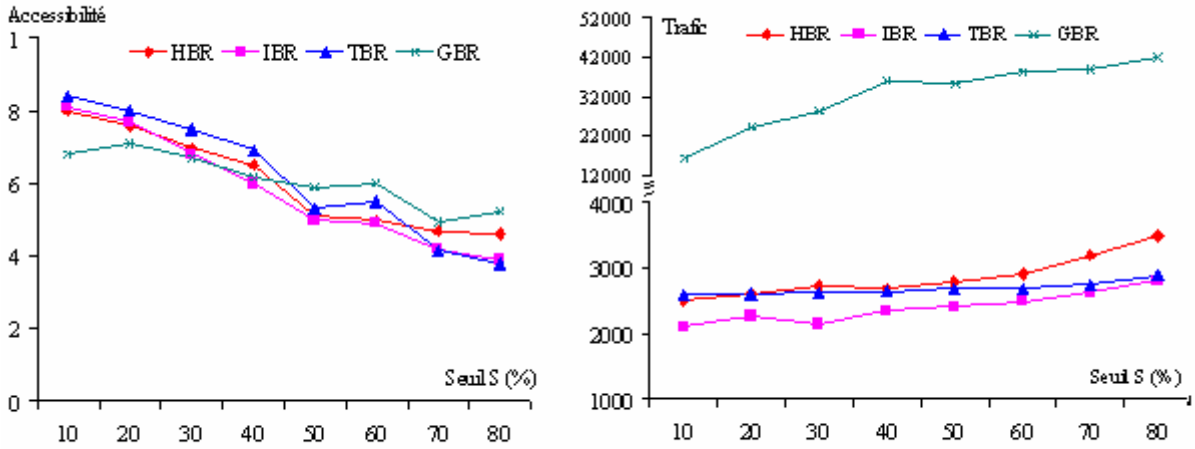

Figure 5.3: Effet de variation du seuil de fréquence

GBR produit toujours un trafic très important mais ces performances en accessibilité sont maintenues même dans le cas d'une valeur importante du seuil. Un seuil important donne un niveau faible de réplication. Mais, pour ***GBR***, le peu de copies placées sont sur des nœuds à liaisons stables. Les accès aux données les plus utilisées ont plus de chances de réussir. C'est une réplication plus efficace.

5.4.1.3 Effet de variation de la taille mémoire

L'effet des variations de la taille mémoire sur l'accessibilité et le trafic est représenté sur la figure 5.4. Ces courbes montrent que les taux d'accessibilité sont globalement satisfaisants. Ils augmentent lorsque la taille de la mémoire augmente.

Nous avons rajouté les résultats de simulation de la méthode ***DCG-S1*** [Hara 03][Hara 06]. Lorsque la taille mémoire augmente, un nœud mobile peut stocker un plus grand nombre de données diverses. Ses potentialités d'accès et celles de son entourage sont améliorées.

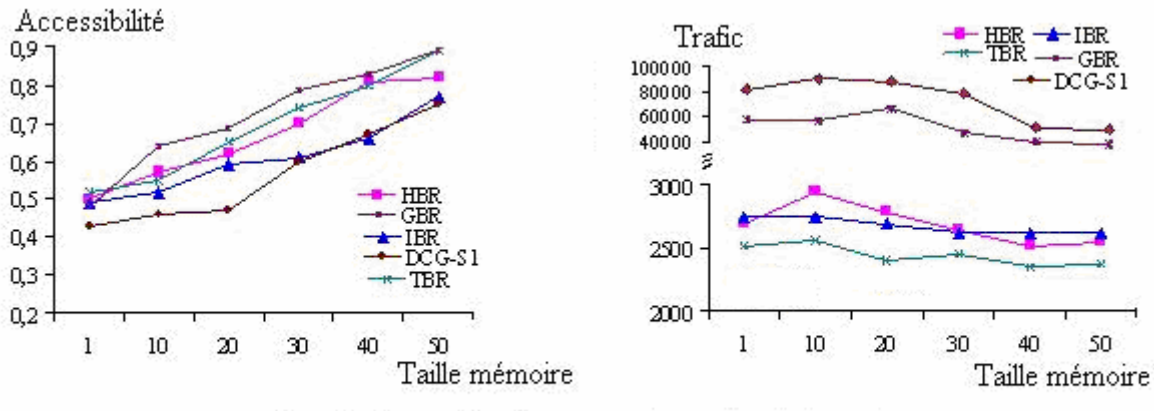

Figure 5.4: Effet de la variation de la taille mémoire

Nous observons que lorsque l'espace mémoire est grand, les méthodes donnent des taux d'accessibilité meilleurs. Ceci est dû à la création de copies préventives qui s'avèrent plus efficace lorsque l'espace mémoire est moins restreint. Avec un espace de stockage réduit, peu de copies d'une donnée seront sur le réseau. Lors d'une réplication adaptative de la donnée, la probabilité d'échec, à atteindre un nœud source, devient plus importante.

Le trafic induit par les trois méthodes ***HBR***, ***IBR*** et ***TBR*** n'est pas très élevé. ***GBR*** et ***DCG-S1*** donne un trafic bien supérieur (figure 5.4) aux autres trafics. Cependant, ce coût pourrait justifier l'avantage de ***GBR*** sur les autres méthodes en taux d'accès réussis. Pour ***DCG-S1***, le trafic important revient à la complexité de construction de groupes stables bi connectés. L'accès aux données dans ce dernier cas reste non favorisé à cause de la taille des groupes. Un groupe de petit effectif offre un espace mémoire commun restreint. L'impact de la variation de l'espace mémoire est alors plus important. Les groupes voisins peuvent avoir des caractéristiques d'accès similaires qui donnent une redondance de données entre les groupes. La diversité des données dans un voisinage est alors limitée.

5.4.1.4 Effet de variation de la vitesse de déplacement

Sur la figure 5.5, nous observons que l'accroissement de la vitesse influence négativement le taux des accès réussis et le trafic généré pour l'ensemble des solutions. La vitesse n'est pas le seul paramètre définissant une mobilité. La mobilité est définie par la vitesse et aussi par le sens de déplacement des nœuds les uns vis-à-vis des autres. Le sens de déplacement étant un paramètre difficile à contrôler, nous étudions la variation de la vitesse qui reste un facteur prépondérant du niveau de mobilité d'un système.

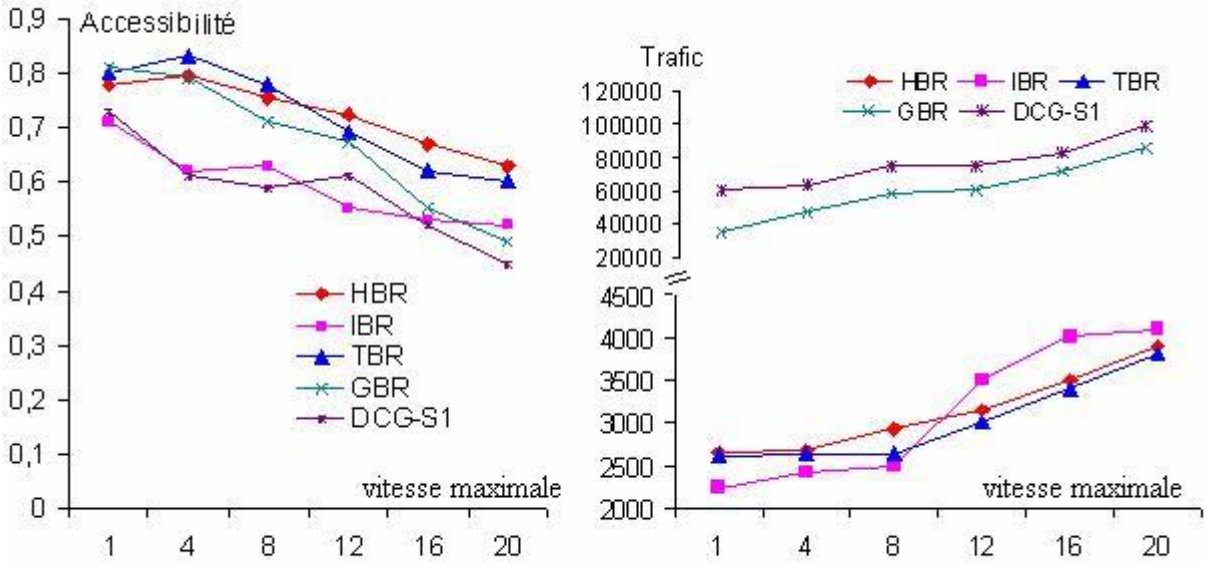

Figure 5.5: Effet de variation de la vitesse de déplacement

Une plus grande mobilité des nœuds entraîne des changements dynamiques de plus en plus fréquents de la topologie du réseau. Dans ce cas, une construction de groupes ***DCG-S1*** devient rapidement invalide; c'est-à-dire non exploitable. Cette construction doit alors être refaite plus souvent en fixant des périodes de réplication plus courtes. Aussi, la taille des groupes devient encore plus restreinte vu des liaisons stables plus rares. L'augmentation de la vitesse a aussi un impact négatif sur le trafic induit. ***GBR*** réagit aussi moins bien à l'accroissement de la vitesse. Il est clair qu'une mobilité plus importante remet plus rapidement en cause la construction des groupes.

HBR semble être la méthode la moins influencée par les variations de vitesse. ***TBR*** donne des résultats proches de ***HBR*** quoique légèrement moins bons. L'évaluation des temps de réponses est remise en causes plus fréquemment à cause des délocalisations des nœuds.

5.4.1.5 Effet de variation de la densité

En maintenant une même surface de simulation et, en variant le nombre de nœuds mobiles, nous obtenons une variation de la densité. Sur la figure 5.6, nous observons que l'accessibilité augmente avec l'augmentation de la densité. Un réseau plus dense favorise l'apparition de connexions multiples.

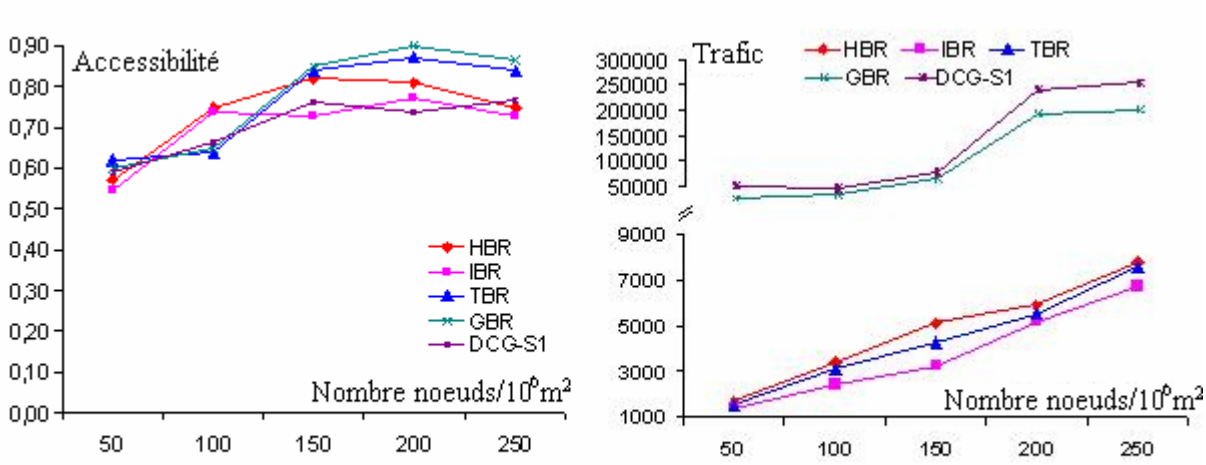

Figure 5.6: Effet de variation de la densité

Globalement, l'accessibilité aux données croît avec l'accroissement de la densité. La densité est un facteur qui diminue les risques de partitionnement. Un nœud a alors plus de chances de trouver un chemin pour atteindre une donnée. Si la densité favorise l'accessibilité, elle favorise aussi la surcharge en messages car toute diffusion entraîne un nombre considérable de messages.

Les méthodes ***GBR*** et ***TBR*** atteignent un haut degré d'accessibilité qui décroît lorsque la densité devient très grande. Car, chaque nœud qui crée une nouvelle donnée, doit exécuter le processus de placement de copies préventives. Lorsque le nombre de nœuds augmente le nombre de nouvelles données augmente aussi. Les copies de ces données provoquent une consommation en espace mémoire et un overload en messages. Pour ces raisons, l'accessibilité diminue un peu lorsque le nombre de nœuds augmente.

Le trafic augmente avec l'accroissement du nombre de nœuds sur le réseau. Car, une grande densité produit plus de connexions. ***IBR*** produit un trafic moins important que les autres méthodes bien que le trafic augmente pour toutes les solutions lorsque la densité augmente. ***TBR*** donne un trafic relativement faible par rapport aux autres méthodes.

La taille moyenne des groupes de réplication est plus importante. ***GBR*** et ***DCG-S1*** obtiennent des groupes qui se partagent un espace mémoire commun assez grand. Les accès se font alors sur le voisinage proche et à travers des liaisons stables plus fiables. Pour ***GBR***, les données se retrouvent souvent à une moyenne de ***k*** sauts. La construction des groupes bien qu'elle soit avantageuse en accessibilité (pour ***GBR***), produit un overload important en trafic.

5.4.1.6 Effet du passage à l'échelle

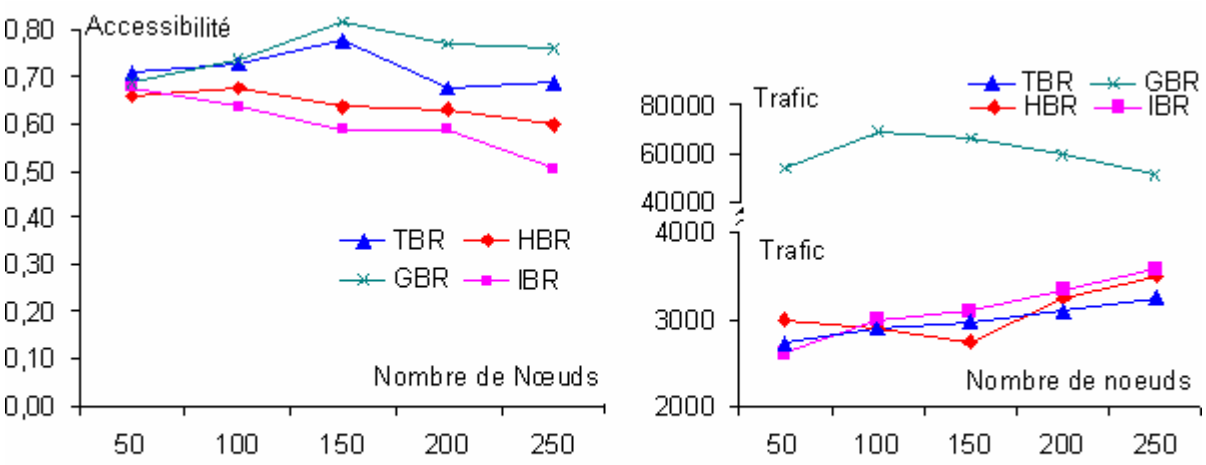

Figure 5.7: Effet du passage à l'échelle

Pour étudier le passage à l'échelle, le nombre de nœuds est augmenté en même temps que l'augmentation de la surface. Pour les solution ***GBR*** et ***TBR***, on observe que l'accessibilité croît avec l'augmentation du nombre de nœuds pour ensuite décroître progressivement pour un nombre de nœuds important (au-delà de 150). L'amélioration de départ de l'accessibilité pourrait s'expliquer par le fait qu'un nombre plus grand de nœuds peut favoriser le degré de connectivité et donc indirectement l'accessibilité. Par contre, la diminution de l'accessibilité s'explique par la fréquence des partitionnements qui devient plus importante avec le passage à l'échelle.

HBR présente une accessibilité stable initialement mais qui se dégrade avec le nombre de nœuds qui devient grand. Le passage à l'échelle en maintenant la même densité des nœuds peut produire des partitionnements plus fréquents d'une part. D'autre part, ces partitions ont plus de chances de regrouper des effectifs plus importants en nombre de nœuds. Chacun de ces nœuds pourrait détenir une donnée non disponible sur une autre partition. ***IBR*** offre une accessibilité satisfaisante mais moins intéressante que les autres méthodes. Ce qui montre l'avantage de la réplication préventive. Le trafic induit par ***IBR*** n'est pas le moins important comme il aurait été prévisible. Le gain en trafic pour la réplication préventive qui est plus rare avec ***IBR***, est compensé par une consommation plus grande en messages lors de la phase recherche et de localisation de données.

GBR produit clairement une surcharge en messages bien plus importante relativement aux autres méthodes à cause de la gestion des groupes. Mais, elle présente une différence positive en accessibilité aux données. Ce résultat était prévisible puisque l'architecture en groupes est un des outils de passage à l'échelle des réseaux mobiles ad hoc.

5.4.2 *Méthode de Partage d'accès aux données avec mise à jour*

Vu les objectifs du protocole de mise à jour, les mesures retenues pour l'évaluation sont principalement le pourcentage de mises à jour réussies et le pourcentage des accès invalides. Les résultats des mesures sur le trafic sont analogues à ceux obtenus pour les méthodes précédentes. Nous ne les présentons pas par la suite, car elle apporte une seule information: le trafic est croisant pour toutes les mesures. Mais, les valeurs maximales restent acceptables.

5.4.2.1 Mises à jours réussies et variation de la densité:

La figure 5.8 représente le pourcentage du nombre de requêtes de mises à jour réussies en fonction de la densité et du passage à l'échelle. Pour chaque densité, nous avons étudié les résultats pour quatre tailles différentes du réseau (50, 100, 150 et 200 nœuds). Cette mesure nous permet de voir l'effet de variation de la densité et du passage à l'échelle sur le protocole d'invalidation proposé pour l'accès aux données avec mise à jour.

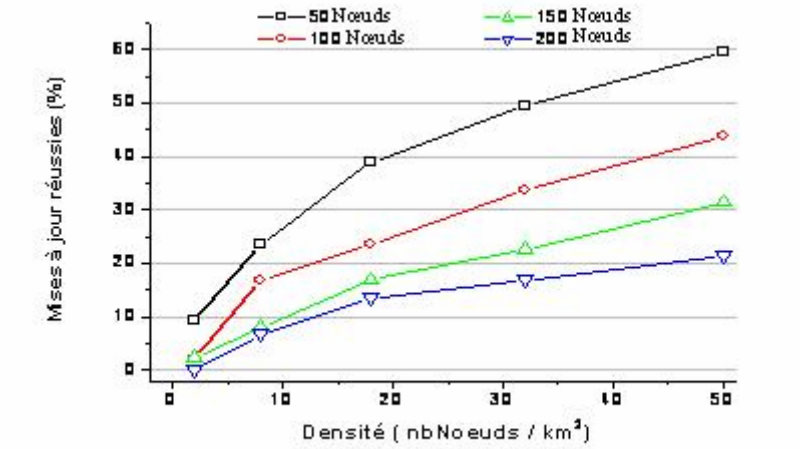

Figure 5.8 : Mises à jour réussies / Densité (avec passage à l'échelle)

Plus la densité augmente, plus le nombre de demandes de mises à jour satisfaites augmente jusqu'à atteindre 60%. Cela est dû au fait qu'avec une grande densité, les nœuds sont moins isolés. De ce fait, ils ont plus de chances d'atteindre le *primaire*. Cependant, une forte densité peut aussi affecter les messages. En effet, pour atteindre le *primaire*, un message peut effectuer plusieurs sauts, donc il peut être atténué ou même perdu. Ceci explique en partie le fait que les mises à jour réussies ne dépassent pas 60%.

Nous constatons que les courbes ont une même allure. Pour une même densité, plus le nombre de nœuds augmente (passage à l'échelle), plus le nombre de demandes de mise à jour satisfaites diminue. Pour une densité faible, les pourcentages de mises à jour réussies sont assez proches. Alors que pour une forte densité, leur variation est assez significative. Ceci est dû au fait que lorsqu'on a un faible nombre de nœuds avec une forte densité, il y a plus de chances pour un nœud d'avoir le serveur primaire comme voisin.

5.4.2.2 Mises à jour réussies et variation de la vitesse:

Le modèle RWP (Random Way Point) que nous avons choisi comme modèle de mobilité des nœuds, fait varier la vitesse des nœuds aléatoirement entre une vitesse minimum et une vitesse maximale. Nous avons choisi de tester le protocole de mise à jour par rapport à la mobilité des nœuds et cela en variant les intervalles de vitesse comme suit:

- Vitesse faible : dans ce cas la vitesse de déplacement des nœuds varie entre 0 et 5 m/s.
- Vitesse moyenne : la vitesse de déplacement varie entre 5 et 10 m/s.
- Grande vitesse : la vitesse de déplacement varie entre 10 et 15 m/s.
- Très grande vitesse : la vitesse de déplacement varie entre 15 et 20 m/s.

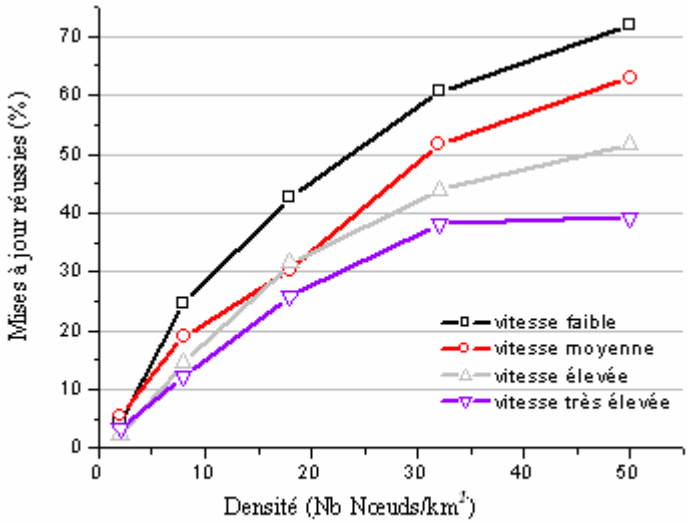

Figure 5.9: Mises à jour réussies / Densité (avec changement de vitesse).

La figure 5.9 représente le nombre de demandes de mises à jour réussies par rapport à la densité. La vitesse varie d'une valeur faible à une valeur très élevée.

Cette mesure nous permet de voir le comportement du système selon la vitesse des nœuds. Nous observons que plus la vitesse augmente, plus le nombre des demandes de mise à jour effectuées diminue. Ceci est dû au fait qu'une mobilité plus importante influence la topologie par des changements plus fréquents.

5.4.2.3 Accessibilité et densité

La figure 5.10 représente le pourcentage de requêtes d'accès réussis en fonction de la densité et de la variation du nombre de nœuds (passage à l'échelle pour chaque densité). Les courbes obtenues ont approximativement une même allure. Elles représentent un taux d'accessibilité satisfaisant. Globalement, nous observons que pour une même densité, un

nombre de nœuds plus grand favorise l'accessibilité à la donnée. En effet, la présence de plus de nœuds augmente la probabilité de trouver un serveur de la donnée et réduit le risque d'isolement d'un noeud. La donnée accédée ici peut être valide ou invalide. Car, si nous considérons que l'accès aux données valides les résultats seraient certainement moins favorables à cause du partitionnement. Ainsi, ce protocole permet un passage à l'échelle avec un taux satisfaisant d'accès satisfaits.

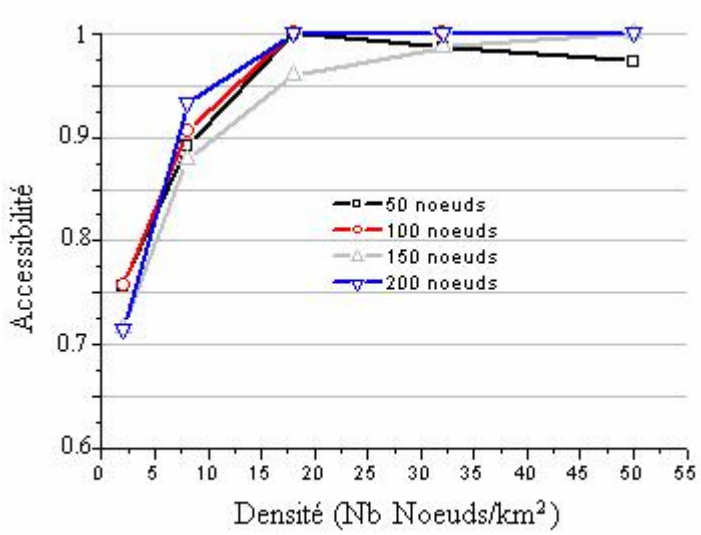

Figure 5.10: Accessibilité / Densité (avec passage à l'échelle).

5.4.2.4 Accessibilité et vitesse

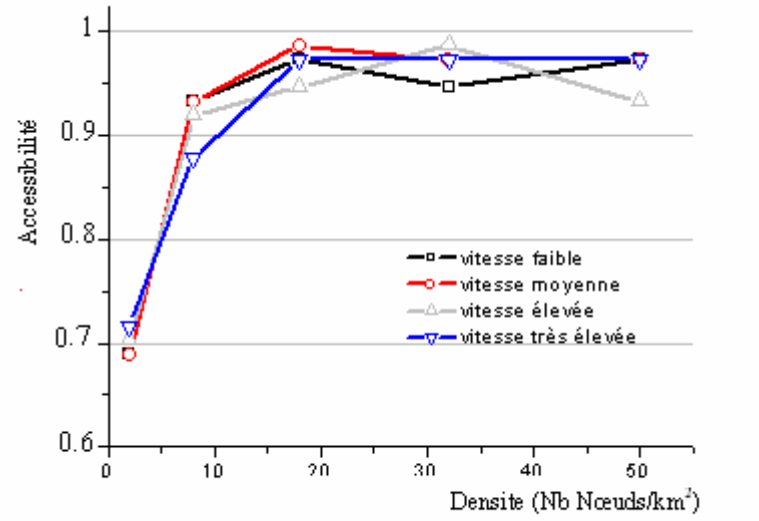

Figure 5.11: Accessibilité / Densité (avec changement de vitesse).

La figure 5.10 nous permet d'observer la variation de l'accessibilité en fonction de la mobilité du système. Nous constatons que la vitesse n'a pas un impact négatif important sur le taux d'accès aux données. Pour les densités moyennes, les vitesses élevées donnent une accessibilité légèrement plus faible mais toujours satisfaisante. Ceci prouve l'importance de la

densité sur les réseaux mobiles ad hoc. Avec une plus grande densité, une requête d'accès peut atteindre plus vite un des serveurs.

Vu que cette accessibilité considère les accès valides et invalides, ces résultats prouvent l'efficacité de la réplication. Quant au protocole de mise à jour, les paramètres les plus significatifs sont le taux de mises à jour réussies et le taux des accès invalides.

5.4.2.5 Accès invalides et charge

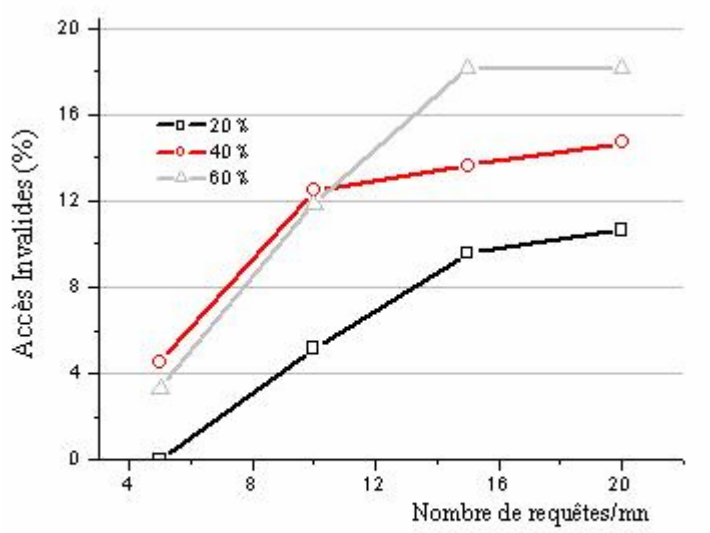

Figure 5.12: Accès invalides / Charge

Les accès invalides ont été étudiés en fonction de la charge du réseau (nombre de requêtes par minute). Il y a deux types de requêtes : les demandes de mise à jour et les demandes d'accès. L'étude du nombre d'accès invalides est faite selon divers taux de mises à jour par rapport au nombre total des accès (20%, 40%, et 60 % de requêtes de mises à jours par rapport au nombre total des requêtes). La figure 5.12, montre que le taux d'accès invalides augmente avec la charge. Pour une même charge globale, lorsque le nombre de mises à jour augmente, le taux d'accès invalides augmente.

5.4.2.6 Mises à jour réussies et densité

Nous avons voulu constater l'impact de la prise en compte de la déconnexion du *primaire*. Pour cela, nous avons étudié les mêmes mesures que celles décrites précédemment (mises à jour réussies, accès réussies et accès invalides) par rapport à la densité. Cependant, les résultats pour les accès réussis et invalides ne sont pas très différents. Ceci est dû au fait que lors d'une demande d'accès, la requête n'est pas forcément destinée au primaire, mais à un nœuds quelconque qui en possède une copie. D'autre part, il y a toujours un serveur primaire unique sur le réseau. Par contre, les requêtes de mise à jour doivent obligatoirement atteindre le serveur primaire. Le remplacement du nœud primaire, en cas de déconnexion, a-t-il eu un apport?

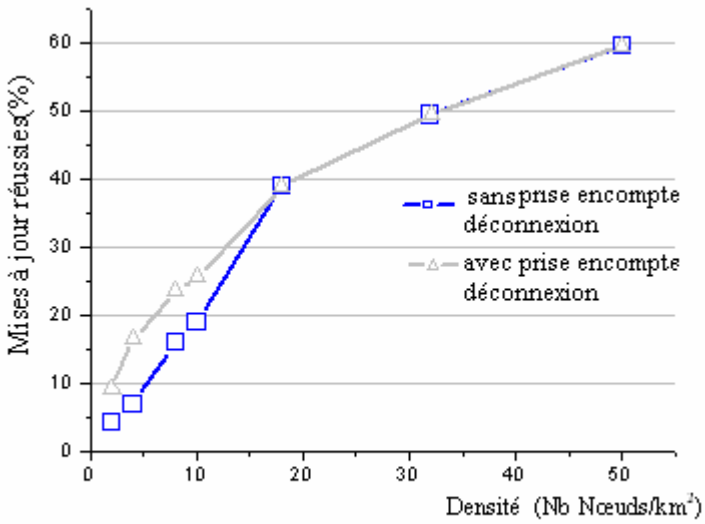

Figure 5.13: Mises à jour réussies / Densité

La figure 5.13 montre que pour de faibles densités, il y a plus de demande de mises à jour réussies avec la prise en charge de la déconnexion du primaire que sans. Plus la densité d'un réseau est faible, plus il y a de déconnexions et donc gérer ces déconnexions a un impact positif sur les résultats. Car, ce problème se pose justement pour les réseaux non denses. Lorsque la densité augmente, les deux courbes se superposent. Ces résultats montrent l'importance de la densité d'un réseau mobile ad hoc.

Conclusion

Pour les solutions qui considèrent des accès sans mise à jour, les résultats de simulation indiquent un gain en accessibilité aux données pour toutes les méthodes. De même, le trafic induit reste acceptable comparé à la solution ***DCG-S1*** très référencée par les chercheurs. Ces évaluations montrent que la notion de groupes utilisée pour ***GBR*** permet d'avoir de bons résultats en cas de passage à l'échelle. Cependant, elle engendre un overload conséquent en trafic. Il faudrait rechercher des mécanismes plus efficaces pour la gestion des groupes.

La solution ***TBR*** est très intéressante en performances d'accès lorsque le taux de création de nouvelles données reste bas. La prise en compte des temps de réponses obtenus dans la politique de réplication s'avère donc intéressante. Mais, le passage à l'échelle de ***TBR*** est moins évident. ***IBR*** présente un intérêt certain pour un environnement avec des composants mobiles à capacité mémoire limitée. ***HBR*** offre un meilleur taux d'accessibilité mais cet avantage se manifeste par un coût supplémentaire. Lors du passage à l'échelle, les trois premières solutions offrent une accessibilité qui décroît avec l'augmentation du nombre de nœuds. Cette accessibilité reste satisfaisante mais le trafic croît de façon exponentielle.

Nous avons évalué le protocole d'invalidation proposé pour l'accès aux données avec mise à jour. Les mises à jour et les accès réussis ainsi que les accès invalides ont été mesurés selon divers paramètres. Une comparaison du protocole avec prise en charge de la déconnexion du *primaire* a montré l'avantage que le protocole apporte. Cependant, nous avons remarqué que la solution proposée ne passe pas suffisamment bien à l'échelle en ce qui est du taux des mises à jour réussies. Ceci est dû encore une fois aux partitionnements plus fréquents dans le cas du passage à l'échelle.

Conclusion

Nous organisons cette conclusion en trois parties : Dans une première partie, nous résumons le positionnement du problème abordé et les motivations de cette thèse. Nous rappelons brièvement les services de gestion des données que doit offrir un système en environnement mobile ad hoc, et ses contraintes. Dans une deuxième partie, nous résumons notre contribution qui porte sur l'amélioration de l'accès aux données partagées. Enfin, dans une troisième partie, nous présentons quelques perspectives de recherche.

- Bref positionnement du problème :

Dans les environnements mobiles cellulaires, l'accès aux données distantes est, en général, conditionné par la possibilité d'établir une connexion sans fil avec le serveur fixe. Les noeuds mobiles déconnectés ne peuvent avoir accès qu'aux données stockées localement. Dans le cas des réseaux mobiles en mode ad hoc, les noeuds mobiles peuvent, dans le cas d'un partage de données, avoir accès aux données stockées sur d'autres noeuds mobiles se trouvant dans leur portée de communication. Le problème de la connectivité intermittente des liaisons de communication se pose alors. La disponibilité des données peut être abordée de deux points de vue :

- du point de vue d'un noeud isolé qui doit avoir recours à son cache local lorsque le serveur de données est inaccessible; et
- du point de vue d'un ensemble de noeuds mobiles qui coopèrent à travers un espace mémoire commun partageable et distribué sur les divers noeuds. Dans ce cas, les nœuds peuvent avoir recours aux données stockées sur des noeuds reliés par des chemins à un, deux ou plusieurs sauts.

La déconnexion volontaire ou involontaire de l'un des noeuds peut générer la perte de données communes. La solution est alors la réplication des données. Cette technique consiste en la duplication des données et le placement des copies sur le réseau.

Au niveau d'une entité isolée, cette solution se traduit par des techniques de pré chargement qui consistent à anticiper les déconnexions des entités mobiles en répliquant localement les données nécessaires pour la période de déconnexion. Au niveau d'un réseau mobile ad hoc, cette solution consiste à placer les copies de sorte qu'elles puissent être utilisées par les nœuds qui en ont besoin.

Une réplication excessive ou systématique des données peut compromettre les ressources de certaines entités mobiles et notamment augmenter leur consommation d'énergie et diminuer l'espace de stockage. Ainsi, pour maintenir l'accès aux données, tout en évitant ces inconvénients, nous ne devons considérer que les données essentielles pour l'application visée. Elles doivent être répliquées de manière rationnelle et à des moments opportuns en prévention des déconnexions.

Dans les réseaux mobiles ad hoc, les requêtes sont acheminées saut par saut jusqu'à atteindre la destination souhaitée. De nombreuses techniques de routage ont été proposées à cet effet. Ces techniques visent à réduire le coût des requêtes en proposant aux paquets, le meilleur chemin possible. Le chemin proposé peut parfois être long ou impraticable, ralentissant de ce fait l'accès aux données. Il apparaît donc clairement que les protocoles de routage, à eux seuls, sont insuffisants dans l'amélioration de la qualité de service. D'où, encore une fois, l'intérêt de la réplication pour assurer une meilleure qualité de service.

Cependant, les protocoles de routage véhiculent avec les données à acheminer, de nombreuses informations qui peuvent être très utiles pour une technique de réplication. Cette dernière, en plus d'adapter la proximité des copies de données aux exigences des utilisateurs qui y accèdent, doit aussi offrir des moyens d'optimisation des temps de recherche. Un ensemble de structures et de procédures doivent être prévues pour enregistrer la trace des données.

- Contributions

Les constatations précédentes ont été le point de départ de nos travaux. L'objectif de cette thèse a été d'offrir des solutions pour améliorer le partage de données en environnement mobile ad hoc. En premier, nous aborderons les solutions proposées pour le maintien de l'accessibilité des données sans considérer les mises à jour. En second, nous exposerons notre contribution dans la prise en compte des mises à jour de données partagées. Nous voyons ces contributions comme une boite à outil pour la définition d'un système de réplication, et son adaptation à une application spécifique. Toutefois, les applications très particulières telles que les applications temps réel nécessitent d'autres mécanismes spécialisés.

Une des contributions a été de proposer une nouvelle approche de réplication à deux phases. Cette approche a été exposée à travers une méthode de base: la méthode ***HBR*** pour la réplication de données. Cette dernière a été critiquée pour être progressivement la source de trois améliorations successives (***IBR***, ***TBR*** et ***GBR***).

La première phase de cette nouvelle approche est une réplication préventive. L'objectif de cette réplication préventive est d'offrir un outil pour contrecarrer les difficultés résultantes des partitions du réseau. Même avec un système de réplication, un partitionnement précoce qui survient après la création d'une donnée, peut mener à une non disponibilité de la donnée pour une partie des noeuds. La réplication préventive proposée est une solution à ce problème. Elle consiste à répliquer les données partagées importantes et à distribuer uniformément des copies préventives. Le niveau de réplication est contrôlé par un paramètre ***k*** définissant les distances qui séparent les copies. La valeur de ***k*** est définie selon le degré d'importance du partage et selon les contraintes de l'environnement (charge, espace mémoire, etc.…). Ce paramètre permet d'adapter la solution à l'application et à l'environnement visé.

La deuxième phase est une réplication adaptative qui a pour rôle d'ajuster l'allocation des copies de données aux changements dynamiques de la topologie et des demandes de l'utilisateur. Cette phase est exécutée continuellement à travers le contrôle des requêtes d'accès et à travers une procédure périodique d'évaluation des besoins des nœuds.

Notre approche se distingue de toutes celles dont nous avons pu prendre connaissance par le fait que le remplacement et le placement d'une copie ne se font pas selon le besoin propre au nœud hôte mais selon les besoins exprimés par tous les noeuds qui accèdent à la copie locale. En effet, dans une région d'un réseau, il n'est pas toujours intéressant pour les performances d'accès global, de déplacer ou de remplacer une donnée parce qu'elle n'est pas la plus importante pour le nœud hôte alors que le voisinage de ce nœud y accède souvent. Pour cela, nous avons défini, pour chaque donnée sur un noeud, deux types de fréquences d'accès: la fréquence interne et la fréquence externe. La fréquence d'accès interne exprime l'intérêt que représente une copie pour son voisinage.

Notre proposition est une solution peu complexe bien qu'elle soit préventive. Elle offre aussi d'importantes possibilités d'adaptation au type de partage envisagé. Elle met à contribution un ensemble de critères des plus importants (temps d'accès, fréquence d'accès, distance, importance sémantique). Ces paramètres ne sont pas statiques et sont évalués dynamiquement.

L'approche de réplication ne suppose pas une technologie de positionnement géographique (GPS, ..) qui peut s'avérer indisponible ou très coûteuse. Elle ne fait aucune hypothèse de prédiction sur le mouvement ou la position futur des nœuds. Nous avons deux schémas possibles pour la prise en compte du partitionnement:

- La prédiction d'une partition dans un futur proche afin de pouvoir effectuer des duplications des données qui apparaissent dans l'une des partitions et non dans l'autre [Chen 02] [Wang 02]. Cette proposition quoique très intéressante nécessite des hypothèses souvent peu réalistes sur la mobilité des nœuds. Elle fait aussi généralement appel à un matériel de positionnement. Ceci n'empêche pas qu'un partitionnement soudain puisse apparaître et mettre à défaut la prédiction.

- Une autre solution est la gestion de la qualité des liaisons à travers la construction de groupes de communication basés sur des liens stables [Hara 02][Hara 03][Hara 06]. Les données sont répliquées selon les besoins des groupes. Seules les copies des données les plus utilisées sont placées sur les noeuds d'un groupe. Les données dont l'utilisation effective n'est pas encore connue ne seront pas prises en charge par la réplication.

La méthode ***GBR*** proposée est basée sur cette dernière conception d'une architecture de réseau en groupes de nœuds à liaisons stables. La notion de groupe est un des moyens les plus reconnus pour lutter contre le partitionnement et pour assurer le passage à l'échelle d'un système. Pour ces raisons, et pour limiter le problème de redondance de données des méthodes précédentes (***HBR***, ***IBR***, ***TBR***), la méthode ***GBR*** a été développée et évaluée. Elle

est basée sur l'approche proposée et inspirée des groupes à liens stables de [Hara 02b]. Ces derniers ont pour inconvénient la production de groupes de petites tailles. ***GBR*** a remédié à ce problème par une construction de groupes hiérarchiques qui élargie l'effectif d'un groupe. Les résultats obtenus par ***GBR*** s'avèrent prometteurs.

La première méthode ***HBR*** se limite à appliquer l'approche proposée pour les données importantes. Cependant, le coût d'une réplication préventive peut être important. La méthode ***IBR*** est venue apporter une première réponse, en proposant une réplication préventive discriminante. Les données très importantes se voient appliquer une procédure de réplication préventive à leur génération sur le réseau. Alors que les données moins importantes sont soumises à une procédure de dissémination d'un descripteur correspondant. Ces deux propositions définissent les besoins d'un nœud uniquement à travers ses fréquences d'accès. Or les temps de réponse expriment aussi les besoins en performances des accès d'un utilisateur. Dans ce cadre, la méthode ***TBR*** a amélioré la méthode ***HBR*** pour incorporer les temps de réponse comme critères de réplication.

Enfin, nous avons aussi apporté une contribution au problème de la mise à jour de données partagées. Cette contribution est donnée à travers un protocole à invalidation. Les protocoles à invalidation sont très utilisés sur les réseaux mobiles ad hoc lorsqu'il s'agit d'offrir un niveau important de cohérence. Dans ce type de protocole, le serveur primaire assure la fonction de contrôle et de maintien de la cohérence d'une donnée. L'état du serveur *primaire* constitue alors un point critique. Sa déconnexion est un problème des plus importants qui peut mener à une inaccessibilité de la donnée. Notre protocole apporte une solution par une prédiction des déconnexions d'un serveur *primaire* et par son remplacement dans ces cas. Le partitionnement est un autre problème complexe qui exige une révision complète des concepts de contrôle de cohérence impliqués. Toutefois, les principales idées d'une solution en cours d'évaluation ont été présentées.

- Perspectives :

L'intérêt du problème abordé par cette thèse en fait un thème d'actualité pour la recherche. Les diverses questions qui peuvent faire l'objet d'une extension à ce travail sont sans doute nombreuses. Néanmoins, nous optons pour les premières perspectives suivantes:

- Une étude plus approfondie du passage à l'échelle. L'approche topologie logique à base de groupes est certainement intéressante mais un effort considérable reste à fournir en ce qui concerne le coût de gestion de l'architecture hiérarchique. Une étude élargie et comparative permettrait certainement de définir les grandes lignes d'une construction optimale des groupes.

- L'adjonction d'un mécanisme de détection de partitionnement et l'évaluation de son apport et de son coût est aussi une des perspectives prochaines.

- Prendre en considération un type de données particulières, par exemple, les données structurées (XML, etc.…). La structure de certaines données offre l'opportunité d'appliquer des solutions de réplication adaptées telles que l'utilisation des TO (Transformés Opérationnelles). Dans ce cadre, nous avons une proposition en cours d'un protocole d'intégration pour une réplication à base de TO sur un réseau mobile ad hoc. Nous envisageons de la soumettre à une conférence dans les mois prochains.

- Une autre perspective est sans doute la réalisation d'un protocole optimiste de cohérence qui soit indépendant d'une entité centrale quelconque.

Références

[Allard 06] G. Allard, P.Minet, D.Nguyen and N.Shrestha, "Evaluation of the Energy Consumption in MANET", Rapport de Recherche INRIA, Février 2006.

[Aho 74] A.V. Aho, J.E. Hopcroft, and J.D. Ullman, "The Design and Analysis of Computer Algorithms", Addison-Wesley, 1974.

[Badache 98] Nadjib Badache, "La mobilité dans les sytèmes répartis", Thèse de Doctorat d'Etat en Informatique obtenue à l'Universite des Sciences et Technologies Houari Boumedienne, Institut d'Informatique, Janvier 1998.

[Badache 02] Nadjib Badache, Djamel Djenouri, Abdelouahid Derhab and Tayeb lemlouma, "Les ptotocoles de routage dans les réseaux mobiles Ad Hoc", revue RIST, Volume 12, N° 2, 2002, Page 77-112.

[Nk 99] R. Bagrodia, X. Zeng, and M. Gerla, “GloMoSim - A Library for Parallel Simulation of Large-scale Wireless Networks”, Comp. Sc. Dept. U. California-Los Angles, 1999.

[Bajaj 00] L. Bajaj, M.Takai, R.Ahjua, KTang, R. Bagrodia and M. Gerla, "GloMoSim: A Scalable Network Simulation Evironment", Computer Science Department University of California et Los Anglos, 2000.

[Barbara 94] D. Barbara and T. Imielinski, "Sleepers and Workholics: Caching Strategies in Mobile Environments", Proceeding ACM SIGMOD ’94, pp. 1-12, 1994.

[Barreto 03] J.P.Barreto, "Information sharing in mobile networks: a survey on replication strategies", Technical Report RT/015/03, Instituto Superior T´ecnico/Distributed Systems Group, Inesc-ID Lisboa, Septembre 2003.

[Bellavista 05a] Paolo Bellavista, Antonio Corradi and Eugenio Magistretti, " Lightweight replication middleware for data and service components in dense MANETs". Sixth IEEE International Symposium on a World of Wireless Mobile and Multimedia Networks, WoWMoM 2005. 13-16 June 2005, pp: 142- 152, ISBN: 0-7695-2342-0

[Bellavista 05b] Paolo Bellavista, Antonio Corradi, Eugenio Magistretti, "REDMAN: A Decentralized Middleware Solution for Cooperative Replication in Dense MANETs", REplication in Dense MANET. International Conference on Pervasive Computing and Communication Workshops, 2005, pp.158-162.

[Bellavista 05c] P.Bellavista, A. Corradia and E. Magistretti, "Comparing and Evaluating Lightweight Solutions for Replica Dissemination and Retrieval in Dense

MANETs", 10th IEEE International Symposium on Computers and Communications (ISCC), July 2005.

[Bettstetter 01] C. Bettstetter, "A random mobility model for simulation of wireless networks", in: Proceedings of the ACM Workshop on Modeling, Analysis and Simulation of Wireless and Mobile Systems, 2001, pp. 19–27.

[Boulkenafed 03] M. Boulkenafed and V. Issarny, "A Middleware Service for Mobile Ad Hoc Data Sharing, Enhancing Data Availability", *4th ACM/IFIP/USENIX International Middleware Conf.*, June 2003.

[Broch 00] J. Broch, D.A. Maltz and D.B. Johnson, "Quantitative lessons from a full-scale multi-hop wireless ad hoc network testbed", Proceedings of the IEEE Wireless Communications and Network Conference 2000 (WCNC 2000).

[Camp 02] T. Camp, J. Boleng, and V. Davies, "Mobility models for ad hoc network simulations", Wireless Communication and Mobile Computing (WCMC), Special issue on Mobile Ad Hoc Networking, Research, Trends and Applications, 2002.

[Cao 04a] Cao Guohong, "Power-Aware Cache Management in Mobile Environments", Department of computer Science and Engineering. Pennsylvania University, 2004.

[Cao 04b] G. Cao, L. Yin, C.R. Das, "Cooperative Cache-based Data Access in Ad Hoc Networks", IEEE Computer, vol. 37, No. 2, Feb. 2004.

[Chen 02] Kai Chen, H. Shah Samarth and Klara Nahrsdted, "Cross layer Design for Data Accessibility in Mobile Ad hoc Networks", Journal of Wireless Personal Communications, vol. 21, pp: 49-76, 2002

[Carson 99] S.Carson and J.Macker, "Mobile Ad hoc Networking (MANET) : Routing Protocol Performance Issues and Evaluation Considerations", RFC 2501, The Internet Society, 1999. http://abcdrfc.free.fr/rfc-vf/rfc2501.html

[Crowder 87] S. V.Crowder, "Average Run Lengths of Exponentially Weighted Moving Average Charts", Journal of Quality Technology, vol.19, 1987 .

[Dedieu 00] Olivier Dedieu, "Réplication optimiste pour les applications collaboratives asynchrones", Thèse de Doctorat de l'Université de Marne*-la-Vallée. 14 septembre 2000.*

[Défago 98] Xavier Défago, André Schiper and Nicole Sergent, "Semi passive réplication", Symposium on Reliable Distributed systems, 1998, pp: 43-50, Computer department of federal school of Lausanne Suisse, {DBLP,http://dplp.uni_trier.de}.

[Djenouri 05] D. Djenouri, L. Khelladi and N. Badache, "A Survey on Security Issues in Mobile Ad hoc and Sensor Network", IEEE Communications Surveys, vol. 7, n° 4, pp: 2-28, December 2005**.**

[Fife 03] L. Fife and L. Gruenwald: "Research issues for data communication in mobile ad hoc network data base systems". ACM SIGMOD Record, 2003.

[Forman 94] Georges Formanand John Zahorjan : *The Challenges of Mobile Computing*, Computer Science & Engineering, University of Washington, US, IEEE Computer, avril 1994, pages 39 à 47.

[Frodigh 00] M. Frodigh, P. Johansson and P. Larsson, "Wireless ad hoc networking – The art of networking without a network ", Ericsson Review n° 4, pp: 248-263, 2000.

[GloMoSim] GloMoSim, Global Mobile Information Systems Simulation Library. Availabe from <http://pcl.cs.ucla.edu/projects/glomosim/>.

[Gossa 05] J. Gossa, J-M Piersonand L. Brunie, "Dynamic Placement of Content Replicas in Distributed Multimedia Systems", Research Report LIRIS, INSA de LYON– 01-2005

[Guy 98]R. G. Guy, P. L. Reiher, D. Ratner, M. Gunter,W. Ma, and G. J. Popek, "Rumor: Mobile data access through optimistic peer-to-peer replication". In ER Workshops, pp: 254–265, 1998

[Hara 00] T. Hara, K.Harumoto, M.Tsukamoto and S.Nishio: "Dynamic replica allocation using database migration in broadband networks", International Conference on Distributed Communication Systems ICDCS 2000, pp.376-384, 10-13 April, 2000.

[Hara 01] Hara T., "Effective replica allocation in ad hoc networks for improving data Accessibility". International Proceeding of IEEE INFOCOM , vol. 3, pp: 1568-1576, April 2001.

[Hara 02a] T. Hara "Replica Allocation in Ad Hoc Networks with Periodic Data Update". Proceeding of MDM 2002 IEEE, 2000.

[Hara 02b] T. Hara "Replicating Data with Aperiodic Update in Ad Hoc Networks". Proceeding of International Conference Communication IASTED, Internet and Information technology , St. Thomas, US Virgin Islands, November 18-20, 2002.

[Hara 03a] Takahiro Hara, Yin-Huei Loh and Shojiro Nishio "Data Replication Methods Based on the Stability of Radio Links in Ad Hoc Networks", Department of

Multimedia Engineering, Osaka University, {hara, nishio}@ist.osaka-u.ac.jp, September 2003. and in Proceeding DEXA IEEE 2003

[Hara 03b] T. Hara. "Replica allocation methods in ad hoc networks with data update", ACM-Kluwer", Journal on Mobile Networks and Applications, 8(4), 2003.

[Hara 04] T. Hara, N;Murakami and S. Nishio, "Replica Allocation for Correlated Data Items in Ad Hoc Sensors networks", ACM SIGMOD RECORD, pp. 38-43, 2004,.

[Hara 04] T. Hara and S. K. Madria: "Dynamic Data Replication Using Aperiodic Updates in Mobile Ad hoc Networks", 9th int. conf. DASFAA 2004

[Hara 06] T. Hara and S.K. Madria, " Data Replication for Improving data accessibility in ad hoc Networks", IEEE Transaction on mobile computing, vol.5, n°.11, 2006.

[Hauspie 02] M. Hauspiey, D. Simplot and J. Carle, "Replication decision algorithm based on link evaluation for services in MANET", International Report, University of Lille, May 2002**.**

[Hauspie 03a] M. Hauspie, D. Simplot-Ryl, and J. Carle,"Partition detection in ad-hoc networks", In Proceedings of the 2nd IFIP Mediterranean Workshop on Ad-Hoc Networks (MED-HOC-NET 2003), Madhia, Tunisia, June 2003.

[Hauspie 03b] M. Hauspie, D. Simplot-Ryl, and J. Carle. Partition detection in ad-hoc networks using multiple disjoint paths set", In Proceedings of the 1st International Workshop on Objects models and Multimedia technologies, Geneva, Switzerland, September 2003.

[Hauspie 04] M. Hauspie, A. Panier, and D. Simplot-Ryl, "Localized probabilistic and dominating set based algorithm for efficient information dissemination in ad hoc networks", In Proceedings of the 1st IEEE International Conference on Mobile Ad-hoc and Sensor Systems (MASS'04), Fort Lauderdale, Florida, USA, 2004.

[Hauspie 05] M. Hauspie, "Contributions à l'étude des gestionnaires de services distribués dans les réseaux ad hoc", Thèse présentée pour l'obtention du grade de Docteur en Informatique. Université de Lille, n° 3506, 2005.

[Hayashi 05] Hayashi Hideki, Hara Takahiro and Nishio Shojiro, " A Replica Allocation Method Adapting to Topology Changes in Ad Hoc Networks", Department of Multimedia Engineering, Osaka University, 2005.

[Helal 86] A. Helal, A. Heddaya and B. Bhargava, "Replication techniques in Distributed Systems", Kluwer Academic Publisher, 1986.

[Hild 95] Stefan Hild, "A Brief History of Mobile Telephony", Cambridge University, Mars 1995.

[Huang 94] Y. Huang, P. Sistla and O. Wolfson, "Data Replication for Mobile Computer ", Proceeding of ACM SIGMOD '94, pp. 13-24, 1994.

[Huang 06] Huang Jiun-Long and Chen Ming-Syan, "On the Effect of Group Mobility to Data Replication in Ad Hoc Networks", IEEE Transactions on Mobile Computing, vol. 1, n°5, May 2006.

[Jamin 01] Jamin S., Jin C., Raz D. and Shavitt Y, "Constrained Mirror Placement on the Internet", Proceeding IEEE INFOCOM Conference, April, 2001.

[Jing 97] J. Jing, A. Elmagarmid, A. Helal and R. Alonso, "Bit-Sequences: An Adaptive Cache Invalidation Method in Mobile Client/Server Environments ", ACM/Baltzer Mobile Networks and Applications, vol. 2, no. 2, pp. 115-127, 1997.

[Jing 04] Jing Zheng, Yijie Wang, Xicheng Lu and Kan Yang, " A Dynamic Adaptative Replica Allocation Algorithm in Mobile Ad hoc networks", Scholl of Computer, National University of Defense Technology, Changshan, China, 2004.

[Judge 98] A. Judge., P. Nixon and V. Cahill, "Overview of Distributed Shared memory", TCD - CS - 1998 - 24, Trinty College Dublin, 1998, Disponible sur : http://WWW.cs.tcd.ie/Publications/tech-reports/tr-index.98.html

[Kouici 06] Nabil Kouici, Denis Conan and Guy Bernard, "Survey of software cache management for tolerating disconnections in mobile environments", GET/INT, CNRS Samovar, Evry, France.

[Kistler 91] J. J. Kistler and M. Satyanarayanan, "Disconnected Operation in the Coda File System", Proceeding 13th ACM Symposium on Operating Systems principles. Pacific Grove, USA, P. 213-225, Octobre 1991.

[Kurose 88] J.F. Kurose and H. Mouftah, "Computer-aided modeling of computer communication networks", IEEE Journal on Selected Areas in Communications 6 (1) (1988), pp: 130–145, 1988.

[Le Mouël 03] Frédéric Le Mouël, "Environnement adaptatif d'exécution distribuée d'applications dans un contexte mobile", Thèse de Doctorat, Université de Rennes 1, Mention Informatique, IRISA, Décembre 2003.

[Lemlouma 00] T. Lemlouma and N. Badache, " Le routage dans les réseaux ad hoc ", Rapport Interne, USTHB, 2000.

[MANET] MANET (Mobile Ad-hoc NETwork) group of IETF (Internet Enginering Task Force). URL: http://tonnant.itd.nrl.navy.mil/manet/manet_home.html.

[Michiardi 03] Pietro Michiardi and Rekik Molva, "Ad hoc networks security", ST Journal of system Research, vol. 4, n° 1, Mars 2003.

[Montgomery 96] D. C. Montgomery, "Introduction to Statistical Quality Control", John Wiley & Sons, Inc. 1996.

[Moussaoui 06] Samira Moussaoui, Mohamed Guerroumi and Nadjib Badache, "Data Replication in Mobile Ad Hoc Networks", Proceeding of Int. Conf. Mobile Ad Hoc and Sensors Networks MSN'06, Springer LNCS 4325, pp. 685-698, HongKong, December 2006.

[Moussaoui 07a] Samira Moussaoui, Mohamed Guerroumi and Nadjib Badache, " Replication Approach for MANETs ", Int. Journal System and Iformation Sciences Notes, SIWN (Systemics and Informatics World Network), ISSN 1753-2310, Vol.1, N° 3, pp. 255-262, July 2007.

[Moussaoui 07b] Samira Moussaoui, Mohamed Guerroumi and Nadjib Badache, "sharing Data Access on Mobile Ad hoc Networks", Proceeding of International conference on Wireless Communication and Mobile Computing M-WCMC 2007, Amman Jordan, September 6-8,2007.

[Moussaoui 07c] Samira Moussaoui, Mohamed Guerroumi and Nadjib Badache, "Improving Data Accessibility in Mobile Ad hoc Networks", International Journal AJIT, Special Issue: Cross Layer Design of Multihop Wireless Networks, ISSN: 1449-2679, 2007 (queued for editing).

[Moussaoui 07d] Samira Moussaoui and Nadjib Badache, "Replicas updates on Mobile Ad hoc Networks", Proceeding of conference ICMTD'07, 27-30 December, 2007. (To appear)

[Moussaoui 07e] Samira Moussaoui and Nadjib Badache, "Hierarchical Groups for Data Replication in Mobile Ad hoc Networks", Proc. of ICMTD'07, 27-30 December 2007. (To appear)

[Moussaoui 08a] Samira Moussaoui, Mohamed Guerroumi and Nadjib Badache, "Two phases replication approach on Mobile Ad hoc Networks", The International Journal for Ad Hoc and Ubiquitous Computing (IJAHUC). (Acceptable for publishing – under final revision).

[Moussaoui 08b] Samira Moussaoui, Mohamed Guerroumi and Nadjib Badache, "Data Access in Mobile Ad hoc Networks", International Arab Journal of Information Technology. (Accepted for publication).

[Nandan 05] A. Nandan, S. Das, G. Pau, M. Gerla and M. Y. Sanadidi, "Co-operative Downloading in Vehicular Ad-hoc Wireless Networks", *2nd IEEE Conference on Wireless On demand Network Systems and Services (WONS)*, Jan. 2005.

[Naoumov 03] V. Naoumov and T. Gross," Simulation of Large Ad Hoc Networks". ACM MSWiM'03, San Diego, California, USA. September 19, 2003.

[Neyem 06] André Neyem, Sergio F. Ochoa and José A. Pino, "A Strategy to Share Documents in MANETs using Mobiles Devices", ICACT2006, February 2006, ISBN 89-5519-129-4.

[NS-2] The Network Simulator–ns-2, http://www.isi.edu/nsnam/ns/index.html.

[Dk 03] Mohammad S. Obaidat and Georgios I. Papadimitriou, "*Applied System Simulation- Methodologies and Applications*", ISBN 1 – 4020 – 7603 - 7, 2003.

[Oster 06] G. Oster, P. Urso, P. Molli, A. Imine, "Data Consistency for P2R Collaborative Editing". ACM CSCW'06, Banf, Alberta, Canada, November 2006.

[Padmanabhan 06] Prasanna Padmanabhan and Le Gruenwald, "*Managing Data Replication in Mobile Ad-Hoc Network Databases* ", International Conference on Collaborative Computing: Networking, Applications and Worksharing, November 2006.

[Padmanabhan 07]Examinateur Prasanna Padnabhan Le Gruenwald, Anita Vallur and Mohamed Atiquzzaman, "*A Survey of Data Replication Techniques for Mobile Ad-Hoc Network Databases*", Accepted for publication in Journal of Very Large Data Bases, 2007

[Perrich 05] Filip Perich, Anupam Joshi, and Rada Chirkova, "Data Management for Mobile Ad-Hoc Networks", Book: Enabling Technologies for Wireless e-Business Applications, Editors: W Kou, and Yelena Yesha, Springer July , 2005.

[Petersen 97] K. Petersen, M. J. Spreitzer, D. B. Terry, M. M. Theimer and A. J. Demers, "Flexible update propagation for weakly consistent replication", In 16th Symposium on Operating Systems Principles (SOSP), St. Malo, France, 288–301, 1997.

[Pitoura 95] E. Pitoura and B. Bhargava, " Maintaining Consistency of Data in Mobile Distributed Environments ", Proceeding of IEEE International first Conference on Distributed Computing Systems (ICDCS'95), pp. 404-413, 1995.

[Pitoura 94] E. Pitoura and B. Bhargava, "Building Information Systems for Mobile Environments", Proceeding 3rd International Conference On Information and Knowledge Management Gaithesburg, MD, pp: 371-378, November, 1994.

[Qi Lu 96] Qi Lu, "Improving Data Consistency for Mobile File Access Using Isolation-Only Transactions", PhD Thesis, CMU-CS-96-131, School of Computer, 1996.

[Qiu 01] L.Qiu, V.N.Padmanabhan, and G.M.Voelker, "On the placement of Web Server Replicas", Proceeding of IEEE INFOCOM Conference, April 2001.

[OPNET] OPNET Modeler, http://www.opnet.com/products/ modeler/home.html.

[Ratner 04] D. Ratner, P. Reiher and J. Popek, "Roam: A scalable Replication System for Mobility", Mobile Networks and Applications, 9(5), pp. 537-544, 2004.

[Sailhan 03] Sailhan F. and Issarny V., "Cooperative Caching in Ad Hoc Networks", Proc. Int'l Conf. Mobile Data Management (MDM '03), pp. 13-28, 2003.

[Saito 05] Yasushi Saito and Marc Shapiro, "Optimistic replication". ACM Comput. Survey, 37(1), pp: 42–81, 2005.

[Satyarayanan 02] M. Satyanarayanan, "The evolution of Coda", ACM Transactions on Computer Systems (TOCS), 20(2):85–124, 2002.

[Scourias 96] J. Scourias," Overview of GSM: The Global System for Mobile Communications, 1996. URL: citeseer.nj.nec.com/scourias96overview.html.

[Shinohara 06] Shinohara Masako, Hayashi Hideki, Hara Takahiro and Nishio Shojiro," Replica Allocation Considering Power Consumption in Mobile Ad Hoc Networks", Department of Multimedia Engineering, Osaka University, 2006.

[Tait 95] C. Tait, H. Lei, S. Acharya, and H. Chang. "Intelligent File Hoarding for Mobile Computers". In Proceeding of the First ACM International Conference on Mobile Computing and Networking -MobiCom'95,1995.

[Terry 95] D.B. Terry, M.M. Theimer, K. Petersen, A.J. Demers, M.J. Spre-itzer et C.H.Hauser, " Managing Update Conflicts in Bayou, A Weakly Connected Replicated Storage System", In Proceedings of the 15th Symposium on Operating Systems Principles, pp. 172-183, Copper Mountain Resort, Colorado, ACM, December 1995.

[Yin 04] L. Yin and G. Cao, "Supporting Cooperative Caching in Ad Hoc Networks", IEEE INFOCOM, 2004.

[Yin 04b] Yin Liangzhong and Cao Guohong, "Balancing the Tradeoffs between Data Accessibility and Query Delay in Ad Hoc Networks", Proceedings of the 23rd IEEE International Symposium on Reliable Distributed Systems, pp: 289 – 298, 18-20 Oct. 2004

[UCLA] UCLA Comp. Sc. Dept. Parallel comp. Lab. and wireless Adaptative Mobility Lab., "GloMoSim: A scalable simulation Environment for Wireless and Wired Network Systems". http://pcl.cs.ucla.edu/projects/domains/glomosim.html

[Wang 02] K. Wang and B. Li, "Efficient and guaranteed service Coverage in partitionable mobile ad hoc networks". International Proceeding of IEEE INFOCOM, 2002.

[Wiederhold 90] G. Wiederhold and X. Qian, "Consistency control of replicated data in federated databases", In Proceedings of the 1st Workshop on the Management of Replicated Data, pp: 130-132, Houston, November 1990.

[Wolfson 97] Wolfson O., "Data Management in Mobile Computing", ACM/Baltzer Journal of MONET, 2, 2, Oct. 1997.

[WU 96] K.L. Wu, P.S. Yu, and M.S. Chen, "Energy-Efficient Caching for Wireless Mobile Computing", Proceeding IEEE International first Conference on Data Engineering (ICDE '9), pp. 336-343, 1996.