RÉPUBLIQUE ALGÉRIENNE DÉMOCRATIQUE ET POPULAIRE
MINISTÈRE DE L'ENSEIGNEMENT SUPÉRIEUR ET DE LA RECHERCHE SCIENTIFIQUE

Université des Sciences et de la Technologie Houari Boumediene (U.S.T.H.B) Alger

Faculté d'électronique et d'informatique
Département d'Informatique

THESE

présentée pour l'obtention du diplôme de MAGISTER
EN **INFORMATIQUE**
Option : intelligence artificiel et bases de données avancées

par

Ameur BENDRISS

SUJET

La réplication dynamique des données dans les environnements mobiles

Soutenue publiquement le 15 juillet 2010, Devant le jury :

Mr	M.	AHMED NACER	Professeur (USTHB)	Président du jury.
Mme	S	MOUSSAOUI	Docteur (USTHB)	Directrice de thèse
Mr	M	BENCHEIBA	Docteur (USTHB)	Examinateur
Mme	F	SOUAMI	Docteur (USTHB)	Examinateur

Résumé

Les réseaux ad hoc sont des réseaux distribués, auto organisés qui ne nécessitent pas d'infrastructure. Les entités formant un tel réseau doivent collaborer afin d'assurer le bon fonctionnement de leurs services, tel que le routage et la gestion de données. Dans un tel environnement, de nombreux algorithmes développés pour le monde filaire ne peuvent être adaptés de façon naïve sans entraîner une congestion importante du réseau qui va réduire son efficacité

Un des problèmes posés est l'accessibilité des données communes. Une solution à ce problème sur ce type d'environnement est la réplication de l'information au niveau des unités mobiles. Aujourd'hui, Il existe plusieurs approches de solution de réplication qui sont sujet de test et d'évaluation.

Dans notre thèse on propose un algorithme de réplication dynamique basé sur la notion de groupe, notre solution permet d'avoir un accès aux données répliquées en lecture et en écriture. Cette solution tente de tirer le meilleur profit des caractéristiques de ces environnements. Les résultats de simulation indiquent que notre solution peut apporter un gain en accessibilité à un coût acceptable.

Notre travail consistait à proposer solution pour la dissémination des données afin offrir une meilleure disponibilité et cohérence. Nous avons proposé et simuler un protocole de réplication dans les réseaux mobiles ad hoc. En premier lieu, nous avons étudiés différentes approches de réplication. Cette étude nous a permis de différencier deux classes de protocoles, techniques de réplication à base de placement de copie et techniques de réplication avec mise a jour. L'approche hybride de réplication par groupe semble la plus intéressante en terme de collaboration efficace des nœuds. Elle offre un plus grand espace virtuelle commun pour la réplication des données accédées par plusieurs membres d'un groupe.

Notre approche de réplication par groupe est basée sur la notion des liaisons stables. Car sans un certain degré de confiance accordé aux liaisons, tout service offert serait inefficace et il ne servirait à rien. Cette stabilité des liaisons permet de rentabiliser au mieux la réplication par la limitation des copies de données redondantes. Cette limitation va augmenter le nombre de données accessibles par les nœuds d'un groupe

L'idée du protocole se mesure par le faite de mettre les données qui ont la fréquence d'accès les plus importantes dans l'espace du groupe

Nous avons aussi apporté une contribution au problème de la mise à jour de données partagées. Cette contribution est donnée à travers un protocole de mise à jour à base de messages d'invalidation. Ce protocole a été intégré à une gestion en groupe du réseau afin de tirer le meilleur profit de la coopération entre les nœuds. Le protocole s'exécute à deux niveaux ; interne aux groupes et intra-groupes. Le degré de cohérence de données est stricte localement à un groupe et plus relâché entre les groupes.

.

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

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 Monsieur Mohamed AHMED NACER, 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 Monsieur Mohamed BENCHAIBA et Madame Feriél SOUAMI, de l'intérêt qu'ils ont manifesté pour cette thèse en acceptant de faire partie de ce jury.

Je voudrais exprimer ma profonde gratitude au Madame Samira MOUSSAOUI qui a dirigée mes travaux de recherche durant cette thèse. Je lui serais toujours reconnaissant pour sa présence, sa bienveillance et l'excellence de ses conseils, son aide précieuse et constante, ainsi que pour la confiance dont elle ma honorés en me proposant ce sujet, et l'immense disponibilité qu'elle m'a prodiguée

Dédicace

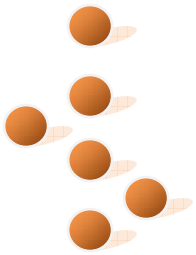
A maman et papa.
A ma chérie immy, qui ma donner le courage, la force et qui a veillez avec insistance à ce que je termine ce travail.
A tous mes amis.

SOMMAIRE:

CHAPITRE 4

SIMULATION & EVALUATION DES PERFORMANCES

Table des figures



Introduction

Les réseaux sans fil ad hoc, sont des réseaux ne disposant d'aucune infrastructure préexistante et formés de nœuds mobiles interconnectés par des liaisons sans fils. Leurs architectures changent au gré de l'apparition et du mouvement des nœuds. L'absence d'infrastructure se traduit par la nécessitée de mettre en place des solutions adaptées reposant sur la participation de l'ensemble des nœuds formant le réseau ad hoc.

L'accès aux données dans les environnements mobiles est un problème majeur dû à la nature de la communication sans fil, la réplication améliore l'accessibilité des données en terme de temps de réponse de disponibilité et de fiabilité, ceci ne peut être garanti que par une distribution efficace des données sur l'ensemble des nœuds du réseau et aussi par un algorithme de gestion d'accès en lecture et en écriture qui prend en considération la manière dont les données sont distribuées sur l'ensemble des nœuds. Un tel algorithme à pour but de donner un temps de réponse réduit quant à l'accès aux données et de diminuer la charge du réseau.

De nombreuses études ont été menées pour résoudre cette problématique, en particulier celles de l'accès aux données, et qui sont toutes convergées vers la réplication des données. La réplication n'est pas une nouvelle technique qui n'existait pas auparavant. En effet, dans le réseau filaire Internet, on trouve le concept de réplication des bases de données. Ce principe consiste à mettre la base de données sur des serveurs distribues dans différents continents dans le monde afin d'améliorer le temps d'accès et éviter la congestion des liaisons de communication …etc.

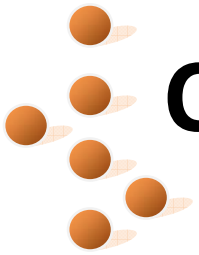
La réplication en environnement *ad hoc* consiste à mettre des copies des documents partagés dans plusieurs hôtes mobiles pour permettre aux entités mobiles une meilleure accessibilité à ces derniers, même après un éventuel partitionnement du réseau dû à leurs particularités

Un algorithme de réplication doit fournir une méthode de distribution des données, un mécanisme d'accès en lecture et en écriture et garantir la cohérence de ces données en cas de modification. Le degré de cohérence fournie par l'algorithme de réplication dépendra de la nature des données à répliquer : des applications critiques telles que les transactions bancaires exigent une cohérence forte d'autres applications telles que le partage des fichiers multimédias tolèrent une cohérence faible.

Dans notre thèse nous proposant un algorithme de réplication dynamique basé sur la notion de groupe, notre solution permettra d'avoir un accès aux données répliquées en lecture et en écriture.

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 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.

Notre thèse est constituée d'une étude théorique consacrée aux principales méthodes de réplications existantes. Ces méthodes constituent une base de travail à partir de laquelle nous proposerons par la suite la conception d'un protocole de réplication basé sur des groupes de nœuds possédant des liens stables. Divers évaluations de cette approche ont été menées sur un environnement de simulation "GloMoSim".



Chapitre 1

Les environnements Mobiles

1. Introduction :

La technologie de la communication sans fil a connu ces dernières années une évolution exponentielle marquée non seulement par l'augmentation des dispositifs mobiles (handheld digital devices, personal digital assistants, or wearable computers) de plus en plus performants, mais aussi par la diversité des applications dans ce domaine. Cette Evolution offre aux utilisateurs la possibilité d'utiliser des ordinateurs portables ou nomades. Ils ne sont plus forcés de travailler depuis leurs ordinateurs du bureau mais au contraire ils peuvent se déplacer librement avec leurs portables et accéder de manière transparente à leurs données sans changement de profil ou d'environnement de travail.

Les réseaux mobiles ad hoc permettent aux utilisateurs de former d'une manière aléatoire et décentralisée un réseau de communication sans fil et sans infrastructures préexistantes. Leur but est de garder l'information accessible n'importe où et n'importe quand. Les réseaux mobiles ad hoc partagent certaines caractéristiques principales avec les systèmes peer to peer, comme l'auto organisation et la décentralisation. Ces systèmes trouvent leurs applications dans le domaine de communication Internet et le partage de fichiers

2. L'environnement mobile

Un environnement mobile est un système composé de sites mobiles qui permettent à leurs utilisateurs d'accéder à l'information indépendamment de leurs positions géographiques. Les réseaux mobiles sans fil peuvent être classés en deux catégories:

- Les réseaux avec infrastructure qui utilisent généralement le modèle de la communication cellulaire utilisant les stations de base fixes
- les réseaux sans infrastructure ou les réseaux ad hoc.

Plusieurs systèmes utilisent déjà le modèle cellulaire connaissent une très forte extension ces dernières années (notamment les réseaux GSM) mais ont besoin d'une lourde infrastructure matérielle fixe.

Le modèle de réseau sans infrastructure préexistante n'utilise pas les stations fixes, tous les sites du réseau sont mobiles et communiquent d'une manière directe en utilisant leurs interfaces de communication sans fil. L'absence de l'infrastructure oblige les unités mobiles à se comporter comme des routeurs qui participent au cheminement de données aux autres unités.

3. la communication 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 ludiques 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) [Hild95] [Scourias96] 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 mis à contribution pour la couverture de régions très vastes afin de fournir certains services à une échelle internationale telle que le GPS (Global Position System).

4. 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, ils relient 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 [Baggio95] :
  - **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 elles:
  - **HyperLAN2** (High Performance Radio LAN 2.0): qui permet d'obtenir un débit théorique de 54Mb/s sur une zone d'une centaine de mètres.
  - **WiFi**: Connu aussi sous le nom d'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): 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): 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.

5. Réseaux mobiles ad hoc

5.1 Définition

Un réseau mobile ad hoc est un système autonome formé dynamiquement par des nœuds mobiles qui sont reliés par l'intermédiaire des liens sans fil, sans utiliser une infrastructure préexistante ou une gestion centralisée. Les nœuds se déplacent d'une manière libre et aléatoire et s'organisent arbitrairement. Ainsi, la topologie du réseau ad hoc peut changer rapidement et de manière imprévisible.

Un tel réseau appelé généralement MANET (Mobile Ad hoc NETwork), peut fonctionner dans un mode autonome ou peut être relié à l'Internet. En général, les itinéraires entre les nœuds dans un réseau ad-hoc peuvent inclure des sauts multiples. Chaque nœud peut communiquer directement avec n'importe quel autre nœud qui réside dans sa marge de transmission. Pour communiquer avec les nœuds qui résident au-delà de cet intervalle, un nœud mobile a besoin d'utiliser les nœuds intermédiaires.

5. 2 Les caractéristiques des réseaux ad hoc

Les réseaux mobiles ad hoc sont caractérisés par ce qui suit :

Un système autonome sans infrastructure préexistante: Un réseau ad hoc n'a besoin d'aucune infrastructure préexistante ou d'une gestion centralisée. Chaque nœud fonctionne en mode distribué de peer to peer, il agit comme un routeur indépendant et produit des données

indépendantes. La gestion de réseau doit être distribuée à travers les différents nœuds, ce qui apporte une difficulté dans la détection et la gestion des défauts.

Routage de multi-sauts : Car tous les nœuds dans un réseau ad hoc jouent le rôle d'un routeur et aucun routeur donné par défaut, les messages transmis par l'émetteur doivent être envoyés nœud par nœud afin d'arriver au destinataire.

Topologie dynamique : Les unités mobiles du réseau se déplacent d'une façon libre et arbitraire. Par conséquent la topologie du réseau peut changer à des instants imprévisibles, d'une manière rapide et aléatoire. Les liens de la topologie peuvent être unis ou bidirectionnels.

Bande passante limitée : Une des caractéristiques primordiales des réseaux basés sur la communication sans fil est l'utilisation d'un médium de communication partagé. Ce partage fait que la bande passante réservée à un hôte est modeste.

Energie autonome : Les nœuds mobiles sont alimentés par des sources d'énergie autonomes comme les batteries. Une partie importante de cette énergie est consommée quand les unités transmettent ou reçoivent les messages à travers une connexion sans fil. Ce qui implique que les protocoles conçus pour les réseaux ad hoc doivent réduire au maximum le trafic, pour minimiser la consommation d'énergie.

Sécurité physique limitée : Les réseaux mobiles ad hoc sont plus touchés par le paramètre de sécurité par rapport aux réseaux filaires classiques. Cela se justifie par les contraintes et les limitations physiques qui font que le contrôle de données transférées doit être minimisé.

6. Le problème d'accessibilité de données dans les réseaux mobiles

Dans les réseaux mobiles ad hoc, chaque nœud joue le rôle d'un routeur et communique avec les autres. Même si la source et destination ne sont pas en contact direct, les paquets sont envoyés au nœud mobile destination en utilisant des nœuds intermédiaires. La nature de déplacement aléatoire des nœuds mobiles, produit la division fréquente du réseau. La figure1.1 présente un exemple de deux partitions. Dans cette figure, les nœuds de la partition P1 ne peuvent pas accéder aux données de la partition P2 et vice versa. Cette nature inhérente

aux réseaux ad hoc fait que l'accessibilité de données soit toujours inférieure à celle fournie dans les réseaux fixes.

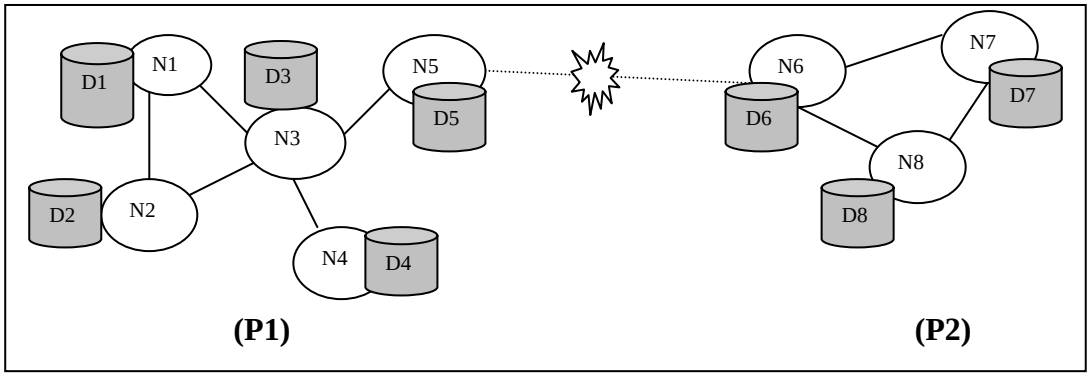


Figure 1.1 : Exemple de partitionnement.

Les nœuds dans un réseau mobile ad hoc se caractérisent par une mobilité aléatoire. Ce genre de déplacement peut partitionner le réseau en plusieurs parties. Un nœud mobile ne pourra jamais accéder aux données se trouvant dans les autres partitions, ce qui diminue le taux d'accessibilité de données.

Une solution possible, pour garder les données accessibles, est de créer pour chaque donnée, une copie identique dans les nœuds de l'autre partition, avant que le partitionnement soit produit. Dans la figure 1.2, si les copies de données sont créées avant l'éloignement des deux partitions, chaque nœud mobile peut accéder aux données après le partitionnement du réseau.

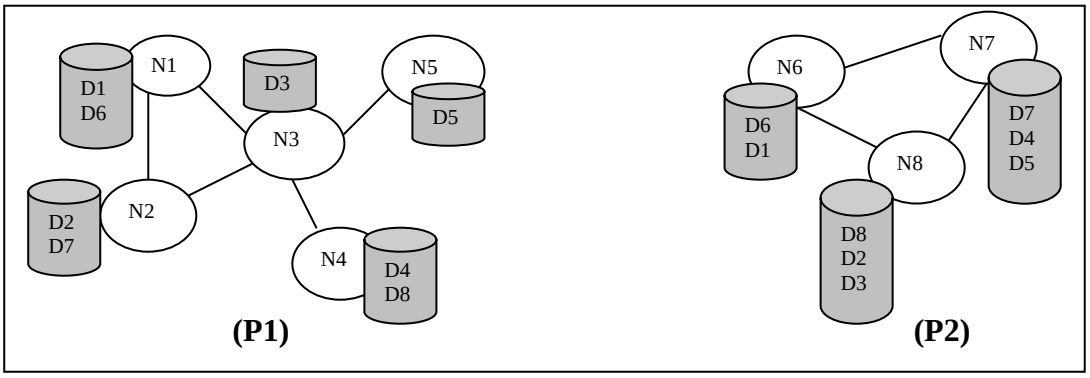


Figure 1.2 : Exemple de création des copies de données.

Cette technique est connue sous le nom de réplication de données. Elle est très efficace pour améliorer l'accessibilité des données. Les nœuds mobiles ont généralement des ressources

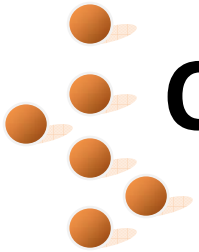
faibles, la réplication de données doit prendre en considération la puissance d'énergie modeste et la bande passante limitée aussi bien que la mobilité de toutes les unités du réseau.

7. Conclusion

Un réseau mobile ad hoc est par conception très peu structuré. Il s'agit de nœuds, tous équivalents du point de vue du réseau, qui doivent se charger eux-mêmes d'établir toute l'infrastructure permettant les communications.

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.

L'utilisation de la réplication peut rendre les données accessibles et réduire le temps de réponse. Cette technique doit aussi prêter attention aux ressources limitées caractérisant les nœuds mobiles. Dans le prochain chapitre nous allons étudier les principes de cette technique dans l'environnement mobile à travers les systèmes existants.



Chapitre 2

Réplication de données &Mobilité

1. Introduction :

Depuis quelques années, les nombreuses évolutions réalisées dans les domaines des terminaux portables et des réseaux sans-fil suscitent un intérêt croissant pour l'informatique mobile. L'utilisation de ces nouveaux environnements introduit de nouvelles problématiques et crée de nouveaux besoins.

La gestion des données dans 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. Elle consiste à créer des copies de la donnée partagée et les placer sur des noeuds dans le but de pallier aux déconnexions imprévisibles et de permettre l'accès aux données. La réplication est aussi un outil pour répartir la charge sur des noeuds serveurs à capacités restreintes. 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.

La réplication dans les environnements mobiles exige des solutions fondamentalement différentes que celles des systèmes stationnaires parce que la mobilité présente un paradigme de calcul fondamentalement nouveau et différent.

Dans ce qui suit nous allons présenter quelques aspects de la mobilité et ces exigences sur la réplication, ainsi nous présentant quelques modèles de réplication pour les environnements mobiles.

2. Définitions :

2.1 La réplication

La réplication est le processus de création et de placement de copies d'entités logicielles. Les entités dupliquées peuvent être des données du code ou les deux à la fois ; Des exemples sont respectivement les fichiers, les tâches de calcul scientifique et les objets.

La création de copies consiste à reproduire la structure et l'état des entités dupliquées. La copie d'un fichier est un autre fichier de même contenu. La copie d'un programme est un autre programme qui exécute le même code et dont l'état d'exécution est celui du programme initial. [Van03]

Le placement des copies consiste à choisir, en fonction des objectifs de duplication, un environnement d'exécution pour les copies. Par exemple, le placement dans un environnement accessible localement assure le travail en mode déconnecté alors que le placement sur plusieurs machines permet la distribution de la charge.

Le processus de duplication peut être décomposé en plusieurs opérations. La mise en œuvre de ces opérations est définie dans un *protocole de réplication.*

2.2 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.

2.3. 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, ils veillent alors à maintenir ce degré sur le réseau, sur une partition, sur une région du réseau, ou sur un sous ensemble des noeuds du réseau.

2.4. Schémas de réplication :

Un schéma de réplication donne le nombre et l'endroit (placement) des copies dans un système distribué. Badrinath [Hara04] définit six scénarios pour le placement des copies des données. Dans les trois premiers, il n y a aucune technique de cache utilisée. Une copie est donc sur chaque serveur de location ou sur le client mobile. Dans le reste, une copie de donnée est cachée sur plusieurs places :

1. le serveur réplique une copie de donnée sur le client mobile. Sur chaque écriture, le serveur est obligé de maître à jour la copie sur le client mobile. L'écriture exige une localisation du client. Une lecture se fait sur la copie locale du client mobile.
2. La copie réside sur le serveur de location du client. Ainsi, le client lit à partir de son serveur de location. Ici, les lectures et les écritures sont sur des copies statiques.
3. Le serveur S a une copie de donnée sur son serveur de location Ls. Le client mobile effectue les lectures et les écritures sur le Ls.
4. Le serveur S a une copie cachée sur son serveur de location.
5. Un cache de la copie de serveur est sur le serveur de location du client. La copie sur le serveur de location est invalidée sur la première écriture depuis la derrière requêtes de lecture du client. La lecture d'un cache invalide exige la localisation le serveur mobile. Mais, si le cache est valide alors la lecture est depuis la copie du serveur de location Lc.
6. Le cache est maintenu sur le client mobile. S'il y a une lecture et le cache est invalide, alors le serveur mobile est contacté pour compléter la lecture.

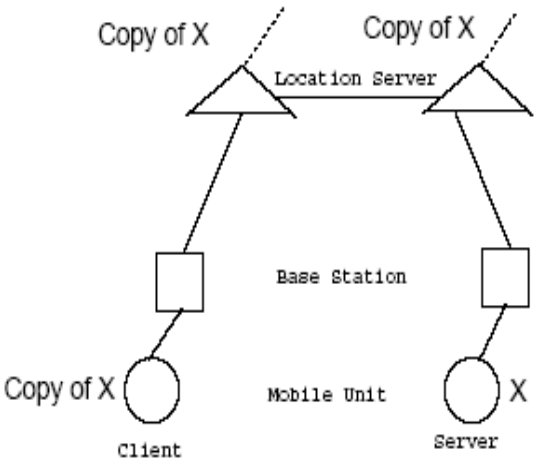


Figure 2.1 ***Possible schémas de réplication***

3. Utilisation de la réplication :

La réplication et la cohérence sont largement utilisées pour répondre aux problèmes issus de la distribution : elles améliorent les performances d'accès, assurent la tolérance aux pannes et facilitent le passage à l'échelle.

- ***Amélioration des performances d'accès* :** L'implication des réseaux dans les interactions distribuées introduit une différence entre les accès locaux et les accès distribués pouvant atteindre plusieurs ordres de grandeur. Les performances d'accès peuvent être améliorées par duplication des services sur des machines ayant des connexions réseau de bonne qualité. Cette technique est typiquement utilisée dans les systèmes de cache et dans les environnements à usagers mobiles. Dans le cas de ces derniers, la duplication est guidée par les déplacements des usagers et leur fournit un accès continu à leurs environnements habituels de travail.
- ***Tolérance aux pannes* :** La tolérance aux pannes est une des utilisations principales de la duplication et de la cohérence ; Les pannes de machines sont tolérées en copiant un service sur plusieurs machines, alors que les pannes de connexion sont gérées en copiant les services sur des machines accessibles par des chemins différents. Les pannes de connexions sont également gérées dans les environnements à usagers mobiles où la duplication et la cohérence permettent le travail en mode déconnecté.
- ***Passage à l'échelle* :** La mise à l'échelle aggrave tous les problèmes issus de la distribution ; Les environnements à grande échelle sont plus vulnérables aux pannes, subissent un trafic qui sature les canaux de communication et doivent faire face à un nombre de requêtes qui dépasse la capacité de traitement des services. La duplication de services permet de placer des copies partout dans les grandes infrastructures, de diminuer le trafic et de répartir la charge en aiguillonnant les clients vers les copies disposant de plus de ressources.

La tolérance aux pannes, l'amélioration des performances d'accès et la mise à l'échelle sont nécessaires dans de nombreux domaines. Les protocoles de duplication et de cohérence trouvent ainsi des applications dans les systèmes de fichiers, les bases de données, les mémoires partagées réparties, les processus distribués, etc.

4. Problèmes liés à la réplication :

Avant d'étudier un système de réplication il faut se poser quelques questions :

D'abord sur le modèle de réplication lui même :

- ***(Quoi ?) Unité de duplication :*** Quelles sont les entités dupliquées? Quel est leur type?

Un protocole de duplication est caractérisé par le type des copies qu'il gère, ainsi que par les critères de sélection des entités effectivement copiées. Le type des copies est généralement

choisi en fonction des domaines d'application des protocoles, alors que les critères de sélection sont définis en fonction des objectifs de performances des protocoles.

- Type de copie

Le type caractérise la structure et la sémantique d'une entité. La structure définit l'organisation interne de l'entité et joue un rôle important lors du processus de création de copie. Ce processus, consiste à reproduire la structure et le contenu de l'entité.

La sémantique définit les opérations autorisées sur une entité. Elle définit des critères de correction qui sont à la base de la gestion de la cohérence entre copies. Par exemple, la sémantique des fichiers stipule que deux opérations d'écriture sur un même fichier sont conflictuelles mais que les opérations de création et de destruction de fichiers différents dans un répertoire ne le sont pas.

• ***(Comment ?) Technique de copie :*** Comment une copie est-elle créée? Quelles sont les informations dupliquées?

Le processus de création de copie dépend de la structure et de l'état de l'entité à dupliquer. La structure de l'entité peut être indivisible ou composée, alors que l'état peut être constitué de données, de code et d'un éventuel état d'exécution.

• ***(Quand ?) Moment de copie :*** Quand une copie est-elle créée? Est-elle créée statiquement, avant l'exécution, ou dynamiquement, en cours d'exécution?

La création de copies peut être faite de manière statique ou dynamique. La création statique correspond à la définition de copies d'une entité avant l'exécution de celle-ci, alors que la création dynamique permet la création de copies en cours d'exécution.

La création dynamique de copies permet l'adaptation de l'ensemble de copies aux variations de l'environnement d'exécution. Les facteurs qui déclenchent de telles créations dynamiques sont typiquement les problèmes de connexion, les problèmes de surcharge ou encore les nombreux accès consécutifs aux copies. Les nouvelles copies répondent à ces problèmes en assurant respectivement la disponibilité lors de partitions, la répartition de charge et la localité d'accès.

• ***(Où ?) Placement de copie :*** Où une copie s'exécute-t-elle? Comment le placement répond-il aux objectifs de mise à l'échelle, de performance et de tolérance aux pannes?

Le placement des copies a un impact direct sur le fonctionnement d'un système dupliqué. Les schémas de placement sont définis en fonction des objectifs de performance d'accès, de tolérance aux fautes ou encore de mise à l'échelle.

Pour une amélioration des performances d'accès, le placement doit prendre en compte les caractéristiques des environnements d'exécution des utilisateurs.
Pour la mise à l'échelle, le placement doit assurer que les copies recouvrent la totalité de l'infrastructure considérée.

Ensuite sur la gestion de la cohérence entre copies :

• ***Critères de correction*** Pourquoi les copies doivent-elles se synchroniser? Ce besoin est-il défini par rapport à un état de référence de l'entité dupliquée ou par rapport au contexte d'utilisation de celle-ci?
La correction d'exécution des copies d'une entité peut être spécifiée de deux manières différentes. La première approche consiste à définir une exécution de référence qui est imposée aux copies. Les copies sont alors synchronisées selon une logique de cohérence dite ***forte*** puisqu'elles ne sont pas autorisées à diverger. La deuxième approche n'impose pas d'exécution de référence aux copies mais considère les exigences des entités qui interagissent avec celles-ci. Les copies sont synchronisées dans ce ca là selon une logique de cohérence dite ***relâchée*** puisqu'elles sont autorisées à diverger tant qu'elles fournissent l'illusion d'une exécution qui satisfait ces entités.

• ***(Qui ?).Copies de synchronisation*** Quand une opération doit être effectuée sur une copie, quelles sont les copies qui décident de la manière de traiter cette opération?
Parmi les copies d'une entité dupliquée, les copies de synchronisation sont celles qui décident de la manière dont une opération sur une copie doit être traitée. Appelées ***maîtres***, ces copies détiennent l'état de référence de l'entité dupliquée et sont les seules à pouvoir le faire évoluer. Les autres copies, dites ***esclaves***, se réfèrent aux décisions des maîtres.
Les protocoles de cohérence font usage de deux schémas de gestion des maîtres : le maître unique et le multi-maître. Dans le cas de maître unique, une seule copie détient et met à jour l'état de référence de l'entité dupliquée. Dans le cas du multi-maître, les opérations de modification sont traitées par plusieurs copies.
Le schéma multi-maître utilise, dans la plupart des cas, des protocoles à cohérence relâchée ; il répartit les opérations de modification entre plusieurs copies. Ainsi, il permet de mieux répondre aux besoins de mise à l'échelle et d'amélioration des performances.

• ***(Comment ?).Techniques de synchronisation*** Comment les copies se mettent-elles d'accord sur la manière de traiter une opération? Quels sont leurs échanges? Qui déclenche une synchronisation?

Pour se synchroniser, les copies d'une entité dupliquée doivent se mettre d'accord sur l'ordre des opérations à traiter, sur le type des informations à échanger, ainsi que sur la manière dont ces informations sont propagées. Dans la mesure où il est considéré que les copies se partagent l'état de l'entité dupliquée, l'ordre définit les contraintes sur les accès concurrents à cet état partagé. Les informations échangées peuvent porter sur l'état actuel des copies ou sur les opérations qu'elles ont effectuées. Elles peuvent être propagées à l'initiative de copies maîtresses ou alors à la demande des esclaves.

• ***(Quand ?) Moment de synchronisation.*** Quand le processus de synchronisation est-il déclenché? La synchronisation est-elle faite avant une modification de copie ou après plusieurs modifications?

Une copie peut se synchroniser avec les autres copies avant ou après l'exécution d'une opération. Les deux approches correspondent à deux stratégies de gestion de la cohérence : la gestion **pessimiste** et la gestion **optimiste**.

Dans l'approche pessimiste, tout traitement est précédé par un accord global entre copies sur la manière dont ce traitement doit être effectué. Les divergences entre copies ne sont donc pas permises et les copies vérifient les conditions de cohérence forte. Les opérations sont exécutées de manière définitive et fournissent des résultats fiables. Toutefois, le déterminisme d'exécution nécessite un processus de synchronisation qui est trop coûteux pour les environnements à grande échelle et non réalisable dans les environnements à partitions.

La gestion optimiste adopte l'approche contraire : elle n'impose pas d'accord global avant l'exécution des opérations et synchronise les copies ultérieurement. Elle repose sur l'hypothèse que les accès conflictuels sur les copies sont rares et traite les opérations sans délai. Toutefois, les résultats de traitement sont provisoires et peuvent être invalidés lors de l'étape de synchronisation. Cette dernière est responsable non seulement de la propagation des opérations entre copies, mais également de la détection et de la résolution d'éventuels conflits.

4.1. Le conflit disponibilité / efficacité / cohérence [Phi94]

Dans tout système réparti, on est généralement en butte au conflit entre trois caractéristiques : disponibilité des données, performance d'accès suffisante vis-à-vis des besoins (tant en écriture qu'en lecture) et non cohérence entre les copies. Malheureusement, ces trois caractéristiques tendent à être antagonistes, et il faut généralement faire des compromis.

Pour obtenir une bonne disponibilité, on souhaite augmenter le nombre de copies de façon à améliorer la tolérance aux fautes et la proximité des données. Cependant, le problème de la cohérence devient alors plus délicat à résoudre.

Pour améliorer l'efficacité, on peut augmenter le nombre de copies (pour rapprocher les données du consommateur), mais cela complique le maintien en cohérence et il existe un risque d'indisponibilité en cas de fautes complexes (partition de réseau notamment).

L'autre solution consiste à diminuer le nombre de copies pour que les algorithmes de maintien en cohérence soient plus simples et plus efficaces, mais le risque de perte de toutes les copies augmentera d'autant.

Enfin, maintenir des copies en cohérence forte permanente est généralement très inefficace, et l'on peut souhaiter utiliser de la cohérence faible, qui augmente la disponibilité immédiate mais réduit la disponibilité des valeurs à jour.

4.1.1 Privilégier efficacité et cohérence

La première solution est de favoriser l'efficacité et la cohérence. Le prototype d'un tel système est le système centralisé, ou système à copie unique : l'efficacité est correcte, le maintien en cohérence n'est pas un problème, mais la disponibilité est médiocre.

Certaines solutions essaient de favoriser l'efficacité en utilisant des aches (ce qui soulève le problème de la cohérence de ces caches), ou la disponibilité en augmentant le nombre de copies. Les algorithmes d'élection rentrent dans cette catégorie, mais pour conserver une bonne efficacité, il faut avoir recours à des formes spéciales d'élections (par exemple, l'élection avec témoins). De plus, le comportement des algorithmes d'élection n'est pas très bon en cas de partition de réseau (perte de disponibilité), et cela nécessite encore une autre forme plus complexe de mise en oeuvre (par exemple, l'élection avec fantômes).

Enfin, les algorithmes d'élection sont très mal adaptés aux réseaux à grande distance, où la vitesse est relativement faible et la fiabilité des communications médiocre : ils nécessitent trop de messages pour parvenir à un accord.

4.1.2 Privilégier disponibilité et cohérence

La deuxième solution est de favoriser la disponibilité et la cohérence, et de ne plus privilégier d'efficacité. Tous les mécanismes qui cherchent à maintenir de la cohérence forte au travers des réseaux grande distance rentrent dans cette catégorie (algorithme d'élection de base,

transaction, jeton. . .), mais il existe quelques mécanismes qui parviennent à obtenir une bonne efficacité sur un réseau local avec une très bonne disponibilité et cohérence forte.

4.1.3 Privilégier disponibilité et efficacité

Enfin, on peut aussi penser à pénaliser la cohérence et ne pas craindre des incohérences temporaires partielles. Le problème qui se pose est alors de détecter ces incohérences pour ensuite les corriger. La détection est un problème délicat en effet on doit détecter toutes les incohérences et savoir reconnaitre les fausses incohérences. La correction est généralement impossible sans des connaissances sémantiques sur la donnée et reste dans tous les cas très complexe.

Cette méthode doit être retenue uniquement quand l'on souhaite obtenir une disponibilité maximale en présence de réseaux à grande distance, même en cas de partition de réseau. Cette catégorie comprend peu de méthodes actuellement, mais il s'agit de sujets d'avenir.

4.2. Les exigences de la réplication dans l'environnement mobile :

Les utilisateurs mobiles ont des conditions spéciales au-dessus et au delà de celles sollicitées par tous les utilisateurs souhaitant partager et répliquer des données. Dans cette section nous discuterons les conditions qui sont particulières à l'utilisation mobile:

a. La communication tout à tout :

Par définition, les utilisateurs mobiles changent leur endroit géographique. En conséquence, on ne peut pas prévoir où établir a priori qu'elle machine sera géographiquement placée à n'importe quel temps donné. Il est en général plus pratique, plus rapide et plus efficace pour communiquer avec un associé local plutôt qu'un autre à distance, les utilisateurs mobiles veulent être capable de communiquer directement et de se synchroniser avec qui que ce soit de plus proche.

L'uniformité peut être correctement maintenue même si deux machines ne peuvent pas se synchroniser directement l'un avec l'autre, mais la synchronisation locale augmente la rentabilité et le niveau de la fonctionnalité tout en diminuant le coût.

Puisque les utilisateurs comptent que les machines voisines peuvent synchroniser l'une avec l'autre rapidement et efficacement, et ils ne peuvent pas prévoir l'emplacement d'une machine à l'avenir, nous avons besoin d'un modèle de réplication capable de soutenir la communication tout à tout. C'est-à-dire, le modèle doit permettre à n'importe quelle machine de

communiquer avec n'importe quelle autre; il ne peut y avoir aucun client de deuxième classe dans le système.

Fournir une communication tout à tout est équivalent à utiliser un modèle de réplication pair à pair; si chacun peut directement synchroniser et échanger des mises à jour avec n'importe qui d'autre, alors chacun doit être par définition égal, au moins en ce qui concerne des capacités de la génération de mise à jour et de réconciliation.

b. Des facteurs de réplication plus larges :

La plupart des systèmes de réplication prévoient seulement un petit nombre de reproductions de n'importe quel objet donné. En plus, les algorithmes pairs à pair n'ont jamais donné un grand nombre de reproductions. Tandis que nous ne réclamons pas un besoin de milliers de copies, il semble probablement que le terrain d'environnements aujourd'hui est envisagé pour le futur proche exigera de plus grands facteurs de réplication.

En premier lieu, les utilisateurs mobiles ont besoin des copies locales sur leurs terminaux. Une fois que chaque utilisateur crée une copie additionnelle, le nombre de copie double au moins et peut se développer.

En second lieu, considérons le cas de la mobilité d'appareils. On suppose que chaque utilisateur a une machine stationnaire et une machine mobile; cependant, le futur à court terme verra l'utilisation de beaucoup plus de dispositifs capables de stocker des données répliquées. Les ordinateurs de PalmTop deviennent plus communs. il existe même une montre bracelet qui peut télécharger des données de calendrier à partir d'un autre ordinateur. Ce n'est pas difficile d'imaginer d'autres dispositifs dans un proche avenir ayant des possibilités de stocker et de mettre à jour des données répliquées; de tels dispositifs augmenteraient les facteurs de copie considérablement.

En conclusion, certains ont discuté le besoin de plus grands facteurs de réplication indépendants du scénario mobile, comme dans le cas de la commande de trafic d'air. D'autres exemples des environnements exigeant de plus grands facteurs de réplication incluent les bourses des valeurs, les tables et les mécanismes de cheminement de réseau, les grands systèmes de base de données comme des systèmes de réservation de ligne aérienne, et la commande et le contrôle militaire.

c. Un contrôle détaillé sue les copies :

Par définition un service de réplication fournit aux utilisateurs un certain degré de contrôle sur les copies. Beaucoup de systèmes fournissent la réplication sur une grande base de granularité, signifiant que les utilisateurs ayant besoin d'une petite partie de données doivent localement copier le conteneur entier. De tels systèmes sont peut être proportionnés dans les environnements stationnaires lorsque les utilisateurs ont accès dans les grands espace de donnée et utilisent les ressources de réseau quand des données importantes ne peuvent pas être stockées localement. Cependant, le contrôle de réplication devient énormément plus important pour les utilisateurs mobiles. Les utilisateurs nomades n'ont pas en général accès aux ressources des machines déconnectés, et donc les objets qui ne sont pas localement stockés sont inaccessibles. Tout ce que l'utilisateur exige doit être répliqué localement, cela devient problématique quand le volume de réplication est grand.

d. Des actions de pré mouvement :

Un modèle possible de conception est que les utilisateurs se registrent eux même comme nomades pour un temps spécifique avant de devenir mobiles. En faisant cela, les structures de contrôles et les algorithmes du système de réplication peuvent être considérablement simplifiés; les utilisateurs sont généralement stationnaires, et enregistrent leur mouvement comme cas inhabituel. Supposez qu'un utilisateur prenait un voyage de trois jours de Los Angeles à New York. Avant le déplacement, les machines à Los Angeles et New York peuvent échanger l'état pour reconfigurer le mobile de l'utilisateur pour agir l'un sur l'autre correctement avec les machines à New York. Le processus de reconfiguration représenterait le cas inhabituel; le cas normal n'assumerait aucun mouvement.

Cependant, une telle politique de conception limite rigoureusement la manière dont la mobilité peut se produire, et ne s'assortit pas avec la réalité d'une utilité mobile. La mobilité ne peut pas toujours être prévue ou programmée.

Pour ces raisons, les solutions qui exigent des actions de pré mouvement ne sont pas fiables dans le scénario mobile. Les actions de Pré mouvement force l'utilisateur à s'adapter au système plutôt que d'avoir l'appui sur le comportement désiré d'utilisateur.

4.3 Placement des copies :

Un problème important, c'est de décider sur le nombre de reproductions et sur leurs emplacements dans le réseau.

Le modèle statique impose un choix fixe des emplacements des copies par des administrateurs en se basant sur des caractéristiques d'accès observées. Dans le modèle dynamique, le système surveille l'accès sur les différentes ressources, et ajuste continuellement l'ensemble des copies.

Le placement statique impose des conditions moins rigoureuses pour l'algorithme de distribution des requêtes, parce qu'il ne doit pas laisser prévoir des effets de réplication ou de migration individuelle d'objet. Ainsi, un algorithme plus efficace de distribution peut être employé, qui pour la même attribution des reproductions aux serveurs, fournirait une meilleure réduction du trafic et de latence. D'autre part, le placement statique peut devenir infaisable pour un système à grande échelle. D'ailleurs, il ne s'ajuste pas sur des changements des modèles d'accès ou de l'environnement (par exemple, topologie d'Internet).

Dans l'approche statique, l'issue principale est le choix du bon algorithme de distribution des requêtes (choix de serveur). La sélection du serveur peut se baser sur beaucoup d'heuristiques (métriques). Le problème ici et de choisir les bonnes métriques et les combiner pour arriver à une stratégie de compromis pour le choix de serveur.

Pour l'approche dynamique, la distribution des requêtes doit être combinée avec l'algorithme de placement des copies. La combinaison doit considérer des facteurs de charge et de proximité et fournir la réponse acceptable dans l'ajustement avec les changements des modèles d'accès.

Un autre problème dû au placement dynamique des copies est la granularité de la réplication. En premier lieux, le placement de chaque objet peut être considéré séparément. L'autre extrémité doit considérer le placement des serveurs entiers. D'autres approches intermédiaires de groupement d'objets sont aussi possibles. Une granularité plus fine impose une surcharge pour collecter et maintenir des statistiques d'accès et pour des décisions de placement. Une granularité plus brute peut augmenter la surcharge pour transférer les objets entre les sites.

Dans certains cas, un groupe de fichiers, d'exécutables, et tous autres états d'environnement doivent être émigrés ensemble. Le coût de transfert de tels objets empaquetés peut être élevé et devrait être pris en considération dans un algorithme de placement dynamique.

Le problème de consistance se pose aussi entre l'ensemble des copies et la base de données qui garde trace des identificateurs logiques d'objets et de leurs emplacements physiques. Les changements dans l'ensemble des copies doivent être reflétés sur cette base. Si cette base n'est pas conforme aux ensembles réels des reproductions, des requêtes peuvent être assignées aux

reproductions inexistantes d'objets. D'autre part, la migration d'un objet et la mise à jour de la base dans une transaction distribuée interféreraient sur des demandes de traitement des clients. Une meilleure approche est de s'assurer que l'ensemble de reproduction indiqué dans la base de données soit toujours un sous-ensemble de l'ensemble réel des copies.

En d'autres termes, quand une reproduction doit être supprimée, elle est d'abord supprimée de la base, et physiquement ensuite seulement enlevé du serveur principal. Quand une reproduction doit être crée, on l'ajoute à la base seulement après être créé avec succès par le serveur principal. La migration peut être traitée comme création de reproduction suivie de suppression de reproduction.

4.4 Choix entre copies :

Les algorithmes de distribution des requêtes se fondent sur des métriques pour faire un choix parmi les copies. Plusieurs métriques on été proposées. Parmi elles, il y a la charge de serveur web, la latence d'un message ping, le temps de réponse, la distance de réseau, la distance géographique (en miles). Une partie de ces métriques intervient quand la redirection des requêtes aura lieu.

5. Stratégies de réplication :

5.1 La réplication active

Dans cette technique il n'y a aucune commande centralisée. Le client envoie la demande à toutes les copies et celles ci traitent la requête de la même manière et dans le même ordre; et renvoient alors une réponse. Le client attend la première réponse ou une majorité des réponses identiques [Lam01].

La réplication active est caractérisée par la symétrie des comportements des copies d'un élément dupliqué. Cette stratégie se déroule en 3 étapes à la suite d'une opération de mise à jour d'une copie:

- Réception des requêtes : toutes les copies reçoivent la même séquence de requêtes dans le même ordre.
- Traitement des requêtes : toutes les copies traitent les requêtes de manière déterministe.
- Émission des réponses : toutes les copies émettent la même séquence de réponses afin d'avoir une convergence de tous les copies.

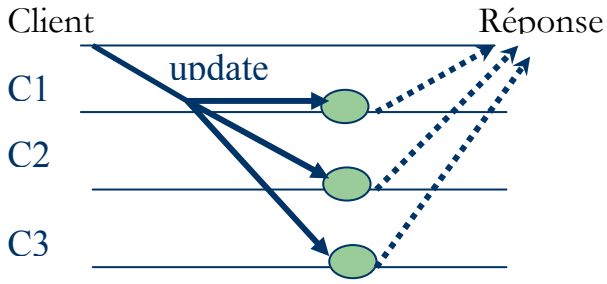


Figure 2.2 : La Réplication Active

5.2 La réplication passive

Pour ce type de réplication, on distingue deux types de copies: une copie ***primaire*** et des copies ***secondaires*** qui sont les backups des données. Une série d'étapes est suivie afin de transmettre cette mise à jour à tous les autres nœuds.

- Réception des requêtes : la copie primaire est la seule à recevoir et traiter les requêtes du nœud qui veut faire la mise à jour.
- Émission des réponses : la copie primaire est la seule à émettre les réponses au nœud qui a généré la mise à jour après les avoir reçues des copies secondaires.

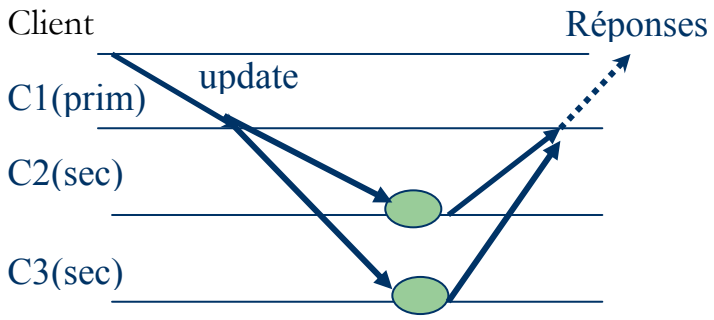


Figure 2.3 : La Réplication Passive

5.3 La réplication semi-active

La réplication semi-active se situe à mi-chemin entre la réplication active et la réplication passive. La copie primaire est appelée leader et les copies secondaires sont appelées suiveurs.

Principe

La réplication semi-active est définie ainsi :

- Réception des requêtes : toutes les copies reçoivent le même ensemble de requêtes.

- traitement des requêtes : toutes les copies traitent toutes les requêtes. La copie primaire traite une requête dès qu'elle la reçoit. Par contre, une copies secondaire doit attendre une notification de la copie primaire pour pourvoir traiter une requête ;
- émission des réponses : la copie primaire est la seule à émettre les réponses.

6. Techniques de réplications pour les environnements mobiles :

Les schémas de réplication traditionnels sont statiques dans le sens où le nombre et le placement des copies sont prédéterminés et fixés. Le calcul manuel des coûts d'accès et la redistribution des copies sont nécessaires pour refléter les nouveaux changements. Cela est acceptable dans les environnements distribués traditionnels puisque les hosts sont installés dans des endroits fixes et le modèle d'accès est relativement statique. Cependant, Dans les environnements mobiles, la réplication statique est insuffisante. En contre partie; les schémas de réplication dynamiques, essaient de surmonter le problème en maintenant continuellement des statistiques sur le modèle d'accès et la charge du travail dans le but de recalculer le coût d'accès et reconfigurer la structure de réplication.

Diverses solutions ont étés proposées pour la réplication dans les réseaux mobiles ad hoc. Les solutions portent sur deux grandes questions:

- Identification des nœuds porteurs des copies des données et de leurs distributions.
- Propagation des mises à jour des copies et le maintient de la cohérence entre les copies.

6.1. Technique de réplication à base de placement de copies :

Le problème de placement des copies a été abordé dans plusieurs travaux [Bel 05], [Hara03], [Hara04], [Hara, Y03], [Hung06], [Tamori02] et [Yu05]. Les solutions proposées envisagent différents scénarios et mécanismes pour décider quels nœuds auront les données répliquées. Le principal objectif est de réduire le regroupement des copies et d'accroître l'accessibilité des données.

6.1.1. Réplication basé fréquences d'accès

Une solution faisant usage aux fréquences d'accès des données et en supposant des mises à jour périodiques a été proposée par Hara dans [Hara03]. Les copies sont diffusées aux nœuds où les données sont plus susceptibles d'être consultées. La décision est prise au moyen des statistiques d'accès aux données recueillies. Lors de l'examen de l'accès des fréquences, si une

donnée D est fréquemment consultée dans le nœud N, alors D est reproduite en N toujours. La mise à jour est périodique signifie que les mises à jour des données se produisent seulement à intervalles réguliers.

Dans un autre travail, Hara et al (dans [Hara04]) on proposé une solution d'examiner non seulement les fréquences d'accès mais aussi la corrélation des données. Corrélation entre les données se produisent lorsque certaines données sont toujours accessibles avec d'autres données particulières. Par exemple, dans le cas d'un système de gestion des désastres, demander des informations sur le nombre de survivants dans un site ou un bâtiment est toujours suivi d'une demande du nombre de secouristes qui travaillent sur ce site et sur la feuille de route vers ce site.

Dans [Hung 06] le schéma de réplication utilise les profils des utilisateurs pour sauvegarder les comportements d'accès et l'historique de lecture/écriture et reconfigurer les copies pour s'ajuster aux changements du comportement d'utilisateur et l'état du système.

Ce schéma est caractérisé par :

- un maintient détaillé sur les statistiques d'accès pour l'utilisateur individuel dans son profil afin que les décisions sur la réplication puissent être adaptées aux mouvements et aux besoins de données de chaque utilisateur.
- offrir à l'utilisateur la possibilité de donner son programme quotidien qui est utilisé pour prédire la mobilité de l'utilisateur et le besoin de données. Cela permet aux systèmes de données un certain degré de prédiction qui alloue les copies en avance.

Les profils d'utilisateurs sont les structures dans lesquelles on définit les tâches des utilisateurs pour maintenir les statistiques d'accès. Chaque tâche est définie par le temps, la localisation et l'activité que l'utilisateur a planifie de faire avec les données requises.

Une mise en correspondance est faite pour déterminer le besoin de données d'une tâche en prenant en considération, les caractéristiques de l'utilisateur, la localisation et l'activité. Les historiques lecture/écriture sont les statistiques d'accès d'utilisateur sur les objets de données dans le système.

6.1.2 Réplication basée sur la stabilité de la liaison radio entre deux noeuds

Une solution compte tenu de la stabilité de la liaison radio entre deux noeuds lors de la diffusion des copies a été proposée par Hara et al. Dans [Hara, Y03]. Si la liaison radio entre

deux noeuds est faible, les nœuds ne sont pas considérés comme des voisins et sont autorisés à tenir des copies de la même donnée.

6.1.3. Les copies ont une distance d'au moins n-hops

Une autre solution a été proposée par Tamori et al [Tamori02] celle de diffuser des données entre les noeuds pour accroître l'accessibilité. La solution propose de réduire le regroupement des copies parmi n hop voisins, en veillant à ce que les copies aient une distance d'au moins n sauts entre eux. La figure 2.3 montre comment la diffusion est réalisée dans cette solution.

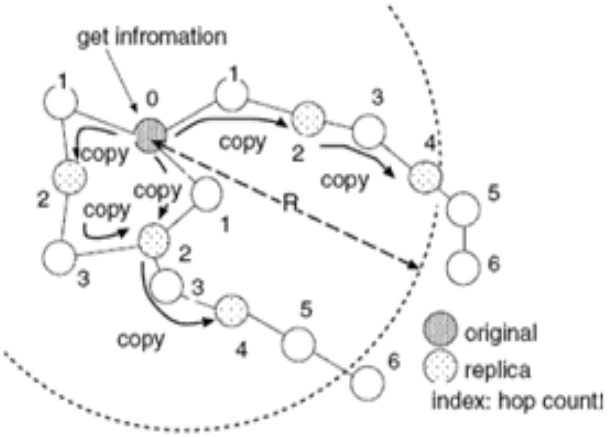


Figure 2.4: Diffusion des copies

6.1.4. Réplication basé cluster hiérarchique :

Il existe des scénarios où les nœuds se déplacent en groupes appelés clusters. Dans de tels scénarios, il serait peut-être efficace de stocker une seule copie de la donnée dans chaque cluster. Yu et coll, Dans [Yu05], et Huang Chen et dans [Hung 06], proposent des solutions de réplication dans de tels scénarios. La solution de Yu et al, Dans [Yu05] est une structure hiérarchique, où les noeuds sont disposés dans un arbre. La gestion de la réplication est effectuée par le chef du groupe. Dans [Hung06], les nœuds sont supposés se déplacer en groupe. La solution propose d'identifier le comportement du groupe du modèle de mouvement des noeuds. Les données sont répliquées, un exemplaire pour chaque groupe. De cette façon, la quantité de mémoire utilisée est conservée Au minimum.

6.1.5. Les copies et leurs emplacements ont une distance d'au moins n sauts

Une solution similaire à celle de [Tamori02] a été proposée par Bellavista et al dans [Bel05]. Les données sont à nouveau réparties entre les nœuds à n-hop de distance les unes des autres.

Cependant, dans cette solution, les informations sur les noeuds qui détiennent les copies sont également réparties entre les noeuds. La figure 2.4 montre un modèle de la solution

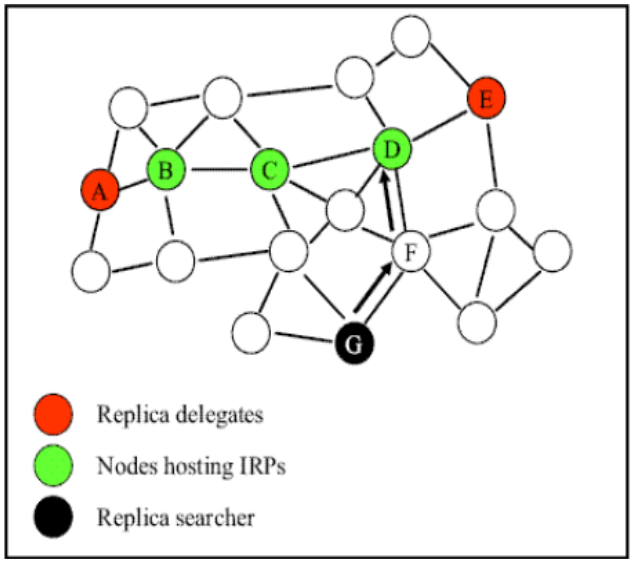


Figure 2.5: Diffusion des copies.

Les principales questions avec les solutions fournies dans [Yu05] et [Hung06] sont celles qui assument un comportement de regroupement dans les nœuds. Dans un contexte purement réseau ad hoc, ceci pourrait être une contrainte sévère, et les solutions pourraient avoir à recalculer les groupes très souvent

6.1.6 Réplication basée sur la notion de groupe :

6.1. Réplication basée sur la densité :

Density-Based Data Réplication est un protocole basé sur une métrique appelée densité, celle-ci à pour but de calculer l'importance d'un nœud dans le réseau en terme d'excentricité. Le protocole est basé sur la notion de groupe construit auteur d'un nœud central calculé à base de la densité et le K voisinage.

Le protocole de réplication peut être vu comme trois parties : construction des groupes, placement des copies et accès aux données.

a) Construction des groupes :

La solution proposée est basée principalement sur la notion de groupe à K saut, la construction du groupe se fait à la base d'une métrique appelée densité, qui permet de déterminer l'importance d'un noeud donnée dans le réseau (le nœud central du groupe), le centre d'un groupe est le nœud ayant la plus grande densité, les nombres sont le K voisinage du centre.

- Calcule de la s-densité grâce aux messages hello périodiquement diffusés.
- diffusion du message hello à K saut contenant la densité du nœud.
- Chaque nœud du réseau construira une table contenant l'identifiant et la densité de son K voisinage. A partir de cette table chaque noeud décidera d'être chef du group ou non selon s'il existe dans sa table un voisin ayant une densité plus élevé ou pas.
- le nœud ayant la plus grande densité envoie un message de regroupement autour de lui à tous son K voisinage.

b) Placement des copies :

Copie primaire :

Le placement des copies dépend directement de la notion de groupe

Le principe est de garder une copie de chaque donnée dans un groupe, et de la placer dans le nœud le plus performant en termes de densité d'énergie et de charge.

Lors de la création d'une nouvelle donnée :

- une copie marquée comme originale et copie primaire sera sauvegardée localement dans le cache
- insertion dans la liste des données disponibles sur le nœud (cette liste sera diffusé périodiquement au K voisinage pour prendre connaissance de la donnée);
- le maître du groupe sera informé par un message de création, celui-ci diffusera ce message vers les autres maîtres des groupes
- chaque maître de groupe choisira parmi les nœuds le plus performant pour héberger la copie.
- ajout de la donnée dans la liste des données disponibles comme copie primaire dans le noeud élu.

Copie secondaire :

Afin d'améliorer les performances d'accès au niveau d'un groupe des copies secondaires de la même donnée seront crées dans les nœuds, ces copies ne seront utilisées que pour les accès locaux.

La création d'une copie secondaire dans un nœud est limitée par les deux conditions :

1. Nombre de sauts : Une requête non satisfaite à un nombre de sauts inférieur à k+1 peut entraîner la création locale d'une copie secondaire de la donnée concernée.
2. Fréquence d'accès pondérée par le temps d'accès : Lorsque le produit du temps d'accès par la fréquence d'accès à une donnée dépasse un seuil, on crée une copie secondaire de la donnée concernée.

C. Accès aux données

L'accès se fait comme suit :

Accès local si donné existe, Sinon Consultation de la table de voisinage, si information existe : choix du meilleur chemin possible selon deux critères nombre de saut, largeur de bande, Sinon la requête sera envoyée vers le nœud central, en parcourant à chaque fois le nœud actuellement père et recherchant dans sa table de voisinage cette donnée, si on atteint le nœud centrale sans à atteindre la donnée, la requête sera orientée vers les nœuds centraux des autres groupes.

6.2. Technique de réplication avec mise à jour :

6.2.1 Coda :

Coda est un système de fichiers distribués, descendant du système de fichiers Andrew (AFS), développé à l'Université Carnegie−Mellon par l'équipe de Mahadev Satyanarayanan. Il offre un accès continu aux données en cas de panne de serveur ou du réseau. Un des principaux trait du système est l'attention portée au support du travail en mode déconnecté (qualifié aussi de travail nomade). [Oliv2000]

Coda utilise un algorithme optimiste qui est une variante de l'algorithme des Copies Disponibles et autorise les lectures et les écritures dans toutes les partitions. Ce choix a été fait pour trois raisons :

- cette approche permet d'offrir une grande disponibilité,

- les utilisateurs doivent pouvoir travailler à partir de portables connectés au système ou isolés, une approche pessimiste interdit le travail en mode isolé.
- les concepteurs estiment que le partage en écriture entre des utilisateurs est relativement peu fréquent dans les environnements universitaires qui sont visés.

Coda augmente la disponibilité des données par la duplication des fichiers sur un ensemble de serveurs et par l'utilisation d'un mécanisme de cache des fichiers sur les machines des clients. Ce mécanisme permet aux clients de travailler entièrement en local sans avoir à contacter les serveurs pour peu que l'ensemble des fichiers auxquels ils accèdent soit dans le cache. L'accès aux fichiers est réalisé de façon complètement transparente à l'utilisateur. L'utilisateur accède en effet à ses fichiers à travers une interface du type *open, read, write, close*. Un processus système, appelé Venus, est chargé de transmettre la requête utilisateur à l'ensemble des serveurs.
L'unité de duplication choisie est le volume, c'est à dire un ensemble de fichiers. L'ensemble des serveurs contenant une copie d'un volume constitue le groupe de stockage du volume (VSG). L'ensemble des serveurs d'un VSG accessible à un instant donné par Venus est appelé le VSG accessible (AVSG). Différents clients (processus Venus) peuvent avoir différents AVSG pour le même volume. Venus teste de façon périodique les membres des VSG dont il a des données dans son cache. Ce test est relativement peu fréquent dans la mesure où il a lieu toutes les dix minutes.

Fonctionnement :

Pour lire une donnée qui ne se trouve pas dans le cache, Venus s'adresse à un serveur privilégié (PS) de l'AVSG afin qu'il lui retourne cette donnée ainsi qu'une information sur l'état de cette donnée. Ce PS peut être choisi en fonction du temps de réponse qu'il fournit. Néanmoins, Venus contacte aussi en parallèle les autres serveurs afin de déterminer si le PS lui a retourné une donnée à jour ou non. La figure suivante schématise le déroulement de ce protocole dans le cas d'une faute de cache.

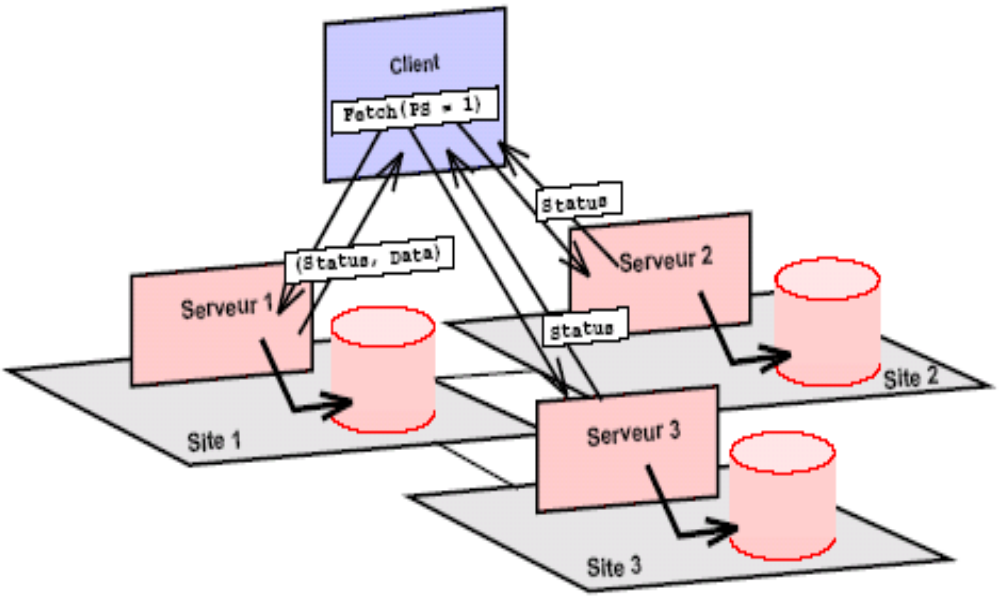


Figure 2.6 : Système de fichier coda

Venus compare les états reçus pour tester si les copies sont toutes équivalentes. Si un conflit est détecté, l'appel système qui a provoqué la faute de cache est abandonné et la procédure de correction est appelée. En revanche, si aucun conflit n'est décelé, mais qu'une des copies n'est pas à jour, Venus informe de manière asynchrone l'AVSG qu'une remise à jour de certaines copies est nécessaire. Le contrôle des conflits écriture−écriture est réalisé grâce à des vecteurs de versions (CVV) mis à jour et testés lors des fermetures de fichiers ou régulièrement par Venus.

Lorsqu'un fichier est fermé après modification, Venus le transfère en parallèle à tous les membres de l'AVSG. Cette approche a été préférée à l'approche qui consiste à transférer le fichier au PS, puis à faire réaliser la propagation de la nouvelle copie par le PS en tâche de fond. Elle permet de ne pas utiliser, pour la propagation, le temps CPU des serveurs qui constituent le goulot d'étranglement du système. Cette technique doit permettre la construction d'un système de grande taille (c'est à dire, plusieurs centaines de sites).

De plus, un serveur n'exécute aucune opération de remise à jour après une panne, ce sont les clients qui l'informeront que ses données ne sont plus à jour ou qu'elles sont en conflit avec d'autres. Bien que cette stratégie ne viole pas les règles de cohérence garanties par le système, elle augmente les chances d'un futur conflit. C'est pourquoi, les concepteurs du système envisagent de remettre automatiquement à jour un serveur par collaboration avec les autres.

Les performances observées dans Coda montrent que le surcoût de la duplication des fichiers est acceptable. Cela est principalement dû à l'utilisation d'un protocole optimiste qui s'adapte très bien à la sémantique des objets manipulés, les fichiers, ainsi qu'à un degré de partage relativement faible. Un tel système n'est absolument pas apte à supporter une base de données dans laquelle les objets sont petits, partagés et souvent modifiés.

6.2.2 Le Modèle Ward :

Le modèle en cluster combine les éléments classiques pairs à pair traditionnel et le modèle client serveur rapportant une solution qui fournit plus de flexibilité des copies permettant une configuration dynamique de la topologie de synchronisation. Le mécanisme principal du modèle est le cluster (the Ward) [Rat02]. Le cluster est une collection de machines proches, géographiquement situées dans un certain secteur. Tandis que tous les membres du cluster sont des pairs égaux, le cluster à un maître, similaire au modèle client serveur avec plusieurs différences importantes :

- Puisque tous les membres du cluster sont des pairs deux membres quelconques peuvent directement synchroniser entre eux. Les solutions typiques client serveur ne permettent pas la synchronisation client-client.
- Puisque tous les membres du cluster sont des pairs, n'importe quel membre peut servir comme maître du cluster. La re-élection et la reconfiguration automatique du maître peuvent se produire quant le maître disparaît.
- Le maître du cluster n'a pas besoin de stocker toutes les données pour tous les intra clusters bien qu'il doive pouvoir identifier l'ensemble complet.

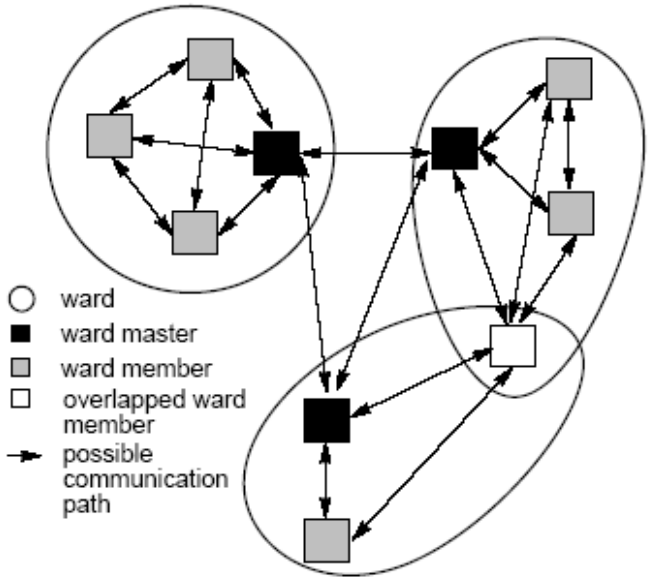


Figure 2.7 : Le modèle Ward.

Le maître du cluster est le seul lien vers d'autres clusters ; c'est pourquoi, seul le maître du cluster se rend compte d'autres copies en dehors du cluster. Le modèle de pair traditionnel force chaque copie pour se renseigner sur l'existence des autres copies. Dans le modèle en cluster, les copies sont informées seulement au sujet des autres copies dans leur propre cluster.

6.2.3. Roam

Roam [D, Rat98] est construit en utilisant le Modèle en cluster (Ward model) ; une nouvelle architecture de réplication qui a été conçue particulièrement avec à l'esprit la mobilité, bien que naturellement elle s'applique également bien aux environnements stationnaires. Le modèle en cluster fournit un nouveau paradigme de réplication qui n'est pas ni un modèle pair à pair ni un modèle client serveur; plutôt, un modèle hybride des deux.
En plus, Roam fournit un certain nombre de nouveaux algorithmes distribués. Par exemple, Roam contient des algorithmes distribués nouveaux et améliorés pour la désaffectation de l'espace disque occupé par les dossiers inutilisables.

De façon générale Roam, fournit:

- Un système de réplication optimiste capable de soutenir les utilisateurs stationnaires et mobiles.

- les capacités pour la communication paire à pair et synchronisation entre deux reproductions quelconques dans le système entier.
- la capacité de devenir mobile à tout moment sans avertissement anticipé ou opérations requises de "pré-mouvement"

Le système utilise la réplication sélective, signifiant que le cache de chaque client contient un sous-ensemble du système de fichier. Le système de fichier est divisé en clusters, où chaque cluster représente une collection de répertoires. Roam n'exige pas que le client mobile cache le cluster entier; il permet plutôt à des dossiers spécifiques dans le cluster d'être cachés.

La communication pair à pair est contrôlée dans Roam par le modèle en cluster. Un cluster se compose d'un groupe de clients mobiles qui sont géographiquement proches entre eux. Tous les clients dans un cluster peuvent communiquer entre eux sur une base pair à pair, bien qu'une telle communication puisse être intermittente ou de mauvaise qualité. Un client peut être potentiellement dans plusieurs clusters. Les clusters forment une hiérarchie à deux niveaux; chaque cluster a un maître qui le relie à un autre de niveau plus élevé. Les clusters et les maîtres sont créés, contrôlés, et détruits dynamiquement durant toute la vie du système.
Pour contrôler la consistance de données entre les clients dans un cluster (où entre les maîtres de cluster au niveau supérieur). Les clients communiquent essentiellement avec leurs voisins sur l'anneau pour recevoir des mises à jour des données partagées. La communication est à sens unique: une copie cible tire toutes les mises à jour d'une copie source. La cible se renseigne sur toutes les mises à jour que la source avait faites ou reçues, mais la source n'apprend rien de la cible, puisque les clients sont rangés dans un anneau, les mises à jour sont passées transitivement entre les clients. L'adaptabilité de l'anneau lui permet d'éviter les clients qui changent de cluster ou qui ne répondent pas temporairement.
Le principal avantage d'employer une topologie d'anneau pour échanger des mises à jour est de limiter le nombre de messages envoyés entre les clients dans un cluster.
Le procédé de réconciliation entre une source et une cible implique des étapes suivantes : D'abord, pour chaque dossier dans le cache locale, la cible demande la source si elle a des mises à jour plus récentes pour ce dossier.
Pour déterminer si un dossier est plus récent ou pas, Roam maintient *des vecteurs de version* pour chaque dossier dans la copie, dans cette approche, Roam maintient un vecteur v, où le $|v|$ est le nombre de copie dans le système. v_i contient le nombre de mises à jour faites. Si la source a les mêmes versions ou plus récentes de toutes les reproductions que la cible, alors le

fichier source domine la cible, quand un dossier domine l'autre, il n'y a aucun conflit. Si ni l'un ni l'autre ne domine, alors les deux reproductions rentrent en conflit sur ce dossier, ainsi le dossier est marqué en tant que invalide.

6.2.4. Bayou :

Bayou [Ter06] est un système de gestion de données répliquées sur un groupe de machines pour soutenir des applications collaboratives. Comme Roam, Bayou utilise un modèle pair à pair pour la communication. À la différence de ce dernier, qui emploie une hiérarchie à deux niveaux et une communication basée anneau, Bayou n'impose aucune restriction sur la communication, n'importe quelle machine peut communiquer avec n'importe quelle autre. Bayou laisse le modèle d'intercommunication des machines pour la couche application. Ceci a l'avantage de permettre à l'application d'optimiser la communication de donnés pour ses besoins spécifiques. Cependant, tandis que Roam peut rendre quelques garanties au sujet du temps nécessaire pour propager une mise à jour par le système, Bayou ne le peut pas.

Chaque machine garde trace de toutes les écritures faites sur localement ou reçu d'une autre, avec une base de données qui est crée par l'exécution de ces écritures dans l'ordre. Une écriture consiste principalement en une série de mises à jour de la base de données. Roam exige que ces fichiers soient envoyés à travers le réseau, Bayou transfère seulement les opérations d'écritures.

Le partage des mises à jour entre deux machines est semblable à celui de Roam. La communication est une opération par pair à sens unique, avec une cible et une source. Quand la cible se connecte à la source, elle reçoit toutes les nouvelles écritures faites par la source par rapport a celles faites par la cible. Afin de détecter les écritures les plus récentes sur une machine, chaque machine maintient un *vecteur de version* contenant le temps des écritures reçues récemment.

Puisque la synchronisation est une opération par pair, le temps nécessaire pour faire ceci est relatif seulement au nombre des écritures dans le système. En outre, les vecteurs de version s'assurent que chaque écriture est envoyée à une machine et seulement une fois durant toute la vie du système. L'ajout des copies affects seulement la taille du vecteur de version, et pas le temps nécessaire pour partager les écritures.

le principal avantage est la flexibilité de Bayou. Bayou permet des procédures de fusion et des contrôles arbitraires, et il permet à la couche application de définir la topologie de

communication entre les copies. À la différence de Roam, le système ne souffre pas quand la communication entre les machines est perdue, puisque la seule condition est que les machines puissent communiquer de temps en temps.

6.2.5. Rover : [Tamori 02]

Rover est un système qui supporte le partage des données dans des applications mobiles. À la différence des deux systèmes décrits ci-dessus, Rover emploie une architecture client serveur. Une application Rover utilise un ensemble de serveurs qui stockent les données et des clients stockent un cache de ces données. Les clients ne sont pas autorisés à communiquer entre eux. Les données globales dans Rover sont modélisées comme un ensemble d'objets dynamiques réadressables (RDOS). Le RDOS contient les données et le code de l'application. Le code du RDO peut être exécuté sur des clients et sur les serveurs.

Le serveur maître du RDO stocke la copie primaire du RDO, et les clients téléchargent des parties du RDOS à leurs caches locales. Un client peut alors faire des mises à jour aux données de RDO dans son cache indépendamment de sa connectivité extérieure.
Pour transférer les mises à jour du RDOS dans les deux sens, Rover utilise des appels à des procédures distantes (QRPCs). Quand un client veut envoyer un RDO au serveur, il ajoute cette demande à la liste des QRPC, et l'envoie quand le serveur devient disponible. La même chose se produit du côté de serveur : si le serveur doit envoyer une réponse à une demande de client, le serveur essayera périodiquement de contacter le client s'il est temporairement indisponible.

Les serveurs maintiennent l'uniformité des données quand des mises à jour contradictoires sont reçues des clients. Rover permet à la couche application de détecter et résoudre ces conflits. Pour aider l'application dans ce processus, Rover garde des informations sur chaque RDO, y compris les vecteurs de versions, d'appel de méthode et de modification de donnée. Une fois que les mises à jour sont traitées par le serveur, elles sont commises, et le nouveau RDO sera envoyé en réponse à toutes les futures demandes des clients.

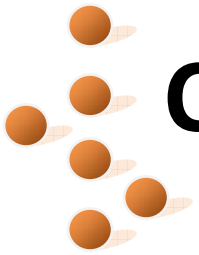
Une caractéristique principale du système Rover est la vaste flexibilité donnée aux applications pour déterminer des politiques de système. D'abord, les applications peuvent décider d'exécuter le code de RDO sur le client ou sur le serveur. Ceci permet aux clients mobiles de pauvres ressources d'accomplir des tâches efficacement.

10. Conclusion :

Dans ce chapitre nous avons étudiés les mécanismes de réplication dans les systèmes répartis. En premier lieu, on a présenté les motivations et la problématique pour les quelles la réplication est une solution ; ainsi le but de cette dernière est d'offrir un mécanisme pour la disponibilité des données, la tolérance aux pannes ou les performances d'accès, quelques problèmes sont présentés du point de vue de la réplication et du point de vue de gestion de cohérence entre copies dupliquées, une classification de ces technique et une étude de cas du système de réplication coda. Quelques axes de recherches sont présentés.

L'étude de quelques techniques et protocoles de réplications 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.
- Les protocoles pessimistes qui tentent d'assurer une cohérence stricte. Ces protocoles sont très contraignants. Mais, ils représentent l'unique issue pour assurer une cohérence stricte.



Chapitre 3

Réplication de données et cohérence

1. Introduction :

L'accès aux données dans les environnements mobiles est un problème majeur dû à la nature de la communication sans fil, la réplication améliore l'accessibilité des données en terme de temps de réponse, de disponibilité et de fiabilité, ceci ne peut être garanti que par une distribution efficace des données sur l'ensemble des nœuds du réseau et aussi par un algorithme de gestion d'accès en lecture et en écriture qui prend en considération la manière dont les données sont distribuées sur l'ensemble des nœuds, un tel algorithme a pour but de donner un temps de réponse réduit quant à l'accès aux données et aussi de diminuer la charge du réseau.

Un algorithme de réplication doit fournir une méthode de distribution, un mécanisme d'accès en lecture et en écriture et doit garantir la cohérence de ces données en cas de modification, le degré de cohérence fourni par l'algorithme de réplication dépendra de la nature des données à répliquer : des applications critiques telles que les transactions bancaires exigent une cohérence forte des données ; d'autres applications telles que le partage des fichiers multimédias tolèrent une cohérence faible.

Dans ce qui suit nous allons présenter notre algorithme de réplication basé sur les groupes, proposer une solution pour la gestion des accès en mise à jour basé sur la technique à rapport d'invalidation adapté pour les groupes, notre protocole assure une gestion forte de la cohérence au sein du même groupe, cependant cela devient une tache plus lourde pour le reste des groupes pour cela nous tolérons une cohérence faible inter groupes.

2. Hypothèses:

- On considère un réseau réparti en groupes de nœuds stables, c'est à dire que la liaison entre les nœuds d'un même groupe est assurée. Ceci est aussi une prévention contre le partitionnement.
- On suppose que tous les nœuds appartiennent à un même groupe d'intérêt; c à dire qu'ils présentent un intérêts pour un même sous ensemble de n données (D_1, ..,D_n). Ceci peut être le cas d'un ensemble de noeuds qui participent à un travail collaboratif commun et qui se partagent l'accès à certaines données.
- On suppose que les données du système n'imposent pas de contrainte forte pour la cohérence des copies.
- Tous les nœuds possèdent la même portée de transmission,

- Chaque donnée a le même identificateur au niveau de tous les nœuds, cela veut dire que si une donnée est nommée D1 dans un nœud elle aura le même identificateur dans tous les autres nœuds.

Affaiblissement tolérable

Les ondes hertziennes subissent un affaiblissement dû à la réflexion, réfraction, diffraction et à l'absorption. L'affaiblissement tolérable est le rapport entre la puissance du signal émis et reçu pour dire que le nœud émetteur est stable par rapport au nœud destinataire et inversement (signal stable).

Affaiblissement (dB) = 10*Log (p2/p1)

P1 est la puissance du signal à l'émission.

P2 est la puissance du signal à la réception.

L'atténuation (affaiblissement) s'accroit avec l'augmentation de la fréquence ou de la distance. De plus lors de la collision avec un obstacle, la valeur de l'atténuation dépend fortement du matériau composant l'obstacle. Généralement les obstacles métalliques provoquent une forte réflexion, tandis que l'eau absorbe le signal.

Atténuation en espace libre

Formule de Friis [1].

Att = -20*Log (c*d/ 4*Pi*F)

Att : Atténuation en dB.

C : célérité de la lumière.

d : distance en mettre.

F : fréquence en Hz.

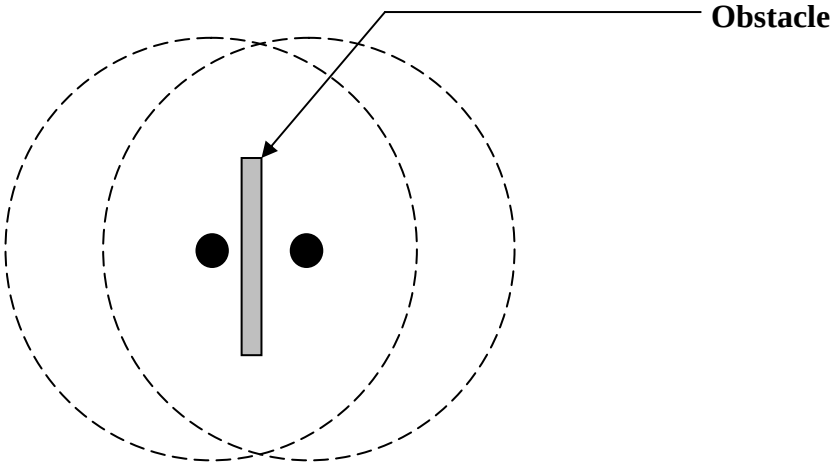


Figure 3.1 : Exemple signal de communication faible

La valeur de l'affaiblissement tolérable dépend des quatre paramètres suivants :

- Vitesse de déplacement des nœuds,
- Période de réallocation T,
- Type de l'environnement, urbain ou libre (exemple : chantier, bâtiment, champs de bataille, désert…etc.),
- Densité de l'environnement i.e. nombre d'unîtes mobiles par unité de surface (y compris la portée de transmission).

L'affaiblissement augmente proportionnellement avec la vitesse, la période T et le type d'environnement, et inversement par rapport a la densité

3. Principe:

Notre protocole de réplication est composé de deux phases : construction de groupe à base de liaisons stables et réplication.

Dans la première phase les nœuds sont regroupés selon le critère de la stabilité des liaisons entre eux, ceci assurera le fait de garder un voisinage stable durant une période de temps T, le groupe construit sera formé de trois niveaux hiérarchiques : le nœud chef (leader déclencheur), un ou plusieurs leader membres et les membres simples qui seront les racines finales.

Le but principale de la construction de groupes et d'avoir une indépendance presque totale des restes des nœuds du réseau en terme de disponibilité de donnée (c'est à ce stade qu'intervient la phase réplication) et aussi d'avoir un mécanisme intelligent d'accessibilité aux données en lecture et en écriture, cela nous permettra d'augmenter la disponibilité et d'éliminer la diffusion des requêtes dans tout le réseau et d'avoir un domaine de broadcast locale au sein d'un même groupe (minimiser la charge du réseau), ce principe est comparable à la notion de VLAN dans les réseaux locaux ;

Lors de la phase de réplication et de placement des replicas, le leader chargé de cette tâche attribue pour chaque donnée disponible dans le groupe un serveur primaire dans son groupe. Ce serveur est le premier serveur désigné pour détenir une copie de la donnée sur la liste de réplication et diffuse la liste de réplication sur tous les membres de son groupe, cette liste de réplication contiendra pour chaque nœud Ni les données à répliquer.

Le leader connaît donc l'identité du serveur primaire de chaque donnée dans le groupe. De même l'ensemble des nœuds d'un groupe a cette information puisqu'ils reçoivent tous la liste de réplication (*"listereplications*").

Un serveur primaire dans un groupe assure un algorithme de mise à jour de type "protocole à invalidation". Ce type d'algorithme permet à l'ensemble des membres d'un groupe de se partager l'accès à un même contenu d'une donnée D_i. A l'intérieur d'un groupe, nous avons donc une cohérence forte permise par le type de connexions disponibles.

Entre les différents groupes, l'assurance d'une cohérence forte peut s'avérer difficile voir même impossible sans imposer des contraintes très lourdes; à cause des déconnexions fréquentes et des partitionnements imprévisibles. Sur ce type de réseau, une cohérence forte engendrerait un taux d'indisponibilité important de la donnée D_i. C'est pour cela, nous optons pour une cohérence plus relâchée entre les différents groupes, qui peuvent éventuellement, être complètement déconnectés et appartenir à deux partitions différentes du réseau.

3.1. Phase de construction des groupes :

Cette phase comporte deux étapes principales: la détermination des voisins stables et le regroupement des nœuds.

3.1.1. La première étape

Cette étape consiste à construire la liste des voisins à liens stables. On définit la stabilité d'un lien entre deux nœuds comme étant, l'assurance que ces deux nœuds resteront connectés jusqu'à la prochaine phase de construction des groupes.

Principe

Initialement, chaque nœud est dans un état libre, il identifie ses liens stables et instables par l'émission d'un message de type «**IDF**» (ou Hello) transportant son identificateur.
Après un temps fini "t" tous les nœuds ont reçu ce message. Ainsi à chaque réception de ce message, le nœud se met en état d'Attente, vérifie la puissance du signal reçu pour chaque IDF et calcule le niveau d'atténuation.

Si cette valeur est supérieure ou égale à un affaiblissement tolérable (seuil), il le considère comme stable et il l'ajoute dans sa liste des voisins stables sinon il l'ignore.
Cependant, tous les nœuds connaissent leurs voisins stables dans leur "**list_voisin**".
Chaque nœud est dans un état Attente, il calcule localement la cardinalité de "**list_voisin**" et récupère son niveau de batterie. Il définit une structure Profil {card, énergie, IDF} puis il envoie un message de type « **PROFILE** » transportant son Profil à ces voisins.

Chaque nœud qui reçoit ce message durant un temps fini t2, vérifie localement si profil.IDF appartient à sa "**list_voisin**", alors MAJ de la liste voisin (compléter les champs énergie et card).
A la fin de t2 tous les nœuds disposent d'une liste de voisins stables et leurs profils.

Algorithme

```
Réception (nœud, message)
   Début
  Type_de_message = retourner_type (message)
  IDF   = retourne_IDF (nœud)

      Switch (type_de_message):

            Case (Hello)
                  Début:
                        Si (! Fin1) && (IDF  !appartient pas a  ListeVoisin)
                              Alors  P= puissance_signal_recu ();
                                    Aff=calculer_aff(P) ;
                                     Si (Aff >= seuil)
                                        Alors
                                              ListeVoisin=ListeVoisin+IDF
                                      fsi
                              fsi
            fincase

      Case (profil)
            Début:
                  Si ( !fin2) alors S:= recherche_ds_listevoisin (IDF);
                     Si S!= Nul alors S.card:= profil.card
                                      S.energie := profil.energie


                     Fsi
                Fsi

      Fincase
```

3.1.2. La deuxième étape

Cette étape est responsable de la désignation du leader de chaque groupe et de l'expression des «**BESOIN**» en données de chaque groupe.

Principe

Dans cette étape, les nœuds procèdent à la détermination de leurs états puis au regroupement. Chaque nœud effectue un traitement local pour désigner son leader et en même temps, il détermine les nœuds qui peuvent être ses membres. Pour ce faire, il doit:

1. Trier la liste de ses voisins stables qu'il a conçus dans la première phase selon un ordre décroissant. Cet ordre est effectué comme suit :
   Dans la comparaison, le vecteur sera classé selon le cardinal, si ce dernier est égal au cardinal du deuxième vecteur; ils seront comparés selon le niveau d'énergie. Au cas où ils soient égaux l'identité est utilisée pour décider du classement.
2. Mit à jour l'identité de son leader. Leader (Ni) est le nœud tête de la liste des voisins stables (précédemment triée).
3. Comparer le vecteur tête de liste avec son vecteur, trois cas possibles peuvent se présenter :
   a) Le nœud deviendra un leader déclencheur quand il sort de cette comparaison avec son vecteur, ce nœud attend toutes les listes des besoins (message « BESOIN ») de ses voisins stables (ou bien un refus qui est un message « BESOIN » mais il n'est pas destiné à lui).
   b) Le nœud deviendra un membre simple lorsqu'il se trouve en queue de liste. Dans ce cas, il envoie sa liste des besoins (message « BESOIN ») à son leader qu'il soit un leader déclencheur ou bien un leader membre.
   c) Le nœud deviendra probablement un leader membre lorsque il se trouve au milieu de la liste (ni tête ni queue) dans ce cas, ce leader attend les listes des besoins des nœuds qui se sont classés derrière lui.

4. Comme nous l'avons déjà dit, les membres simples envoient leurs listes des besoins à leurs leaders. Lors de la réception de ces listes, le traitement sera comme suit:

Je suis destinataire et mon état est leader déclencheur

Dans ce cas, le nœud récepteur ajoute le nœud émetteur dans sa liste groupe et sauvegarde sa liste besoins. Après la réception de tous les messages le leader déclencheur construit la liste des besoins du groupe et passe à la phase REPLICATION.

Je suis destinataire et mon état est leader membre

Ce cas est similaire au premier, le nœud récepteur ajoute le nœud émetteur dans sa liste groupe et sauvegarde sa liste de besoins. Après la réception de tous les messages attendus, le leader membre construit la liste des besoins du groupe et l'envoi a son leader.

Je ne suis pas destinataire et mon état est leader déclencheur

A la réception de ce message, le nœud comprend que c'est un message « refus » alors ce nœud ne fait pas parti de mon groupe.

Je ne suis pas destinataire et mon état est leader membre

A la réception de ce message, le nœud comprend que c'est un message «refus » alors ce nœud ne fait pas parti de mon groupe.

Si après la réception de tous les messages attendus, si la liste_groupe = {vide} alors si :

Je suis leader déclencheur alors mon état devient leader singleton,

Si je suis leader membre alors mon état devient membre simple.

Ce procédé d'envoi de messages besoins évite l'émission de beaucoup d'autres messages entre autre Acceptation, Refus, Invitation. En effet, la liste besoin informe le leader de son acceptation à soumettre dans son groupe, l'informe de ces besoins en données, et refuse tous autres nœuds voisins à être membre de son groupe.

Algorithme

A la réception d'un message « BESOIN » par un nœud Ni, il vérifie s'il est destinataire alors Ajouter l'identificateur de l'émetteur à son groupe.

Ajouter la liste des besoins reçus à la liste des besoins du groupe

Si Ni a reçu tous les messages attendus des nœuds qui sont derrières lui dans la liste des voisins triée alors

Si l'état du nœud est leader membre alors

IDFB = Leader(Ni)

Envoyer liste besoin du groupe (message « BESOIN »).

Suite de la fonction Réception :

```
Case (besoin)
|
| //cpt est le nombre de messages attendue
|       Cpt--;
|       Si (Cpt !=  0)
|       |  Alors si (profil.IDF != IDFB)//je suis pas destinataire
|       |       |  Alors efface_voisin (listeVoisin)// refus
|       |       |        Si      (IDFB !=mon_leader)      alors
|       |       |        List_voisin_refus+=                 +=
|       |       |        listeBesoin.IDF
|       |       |
|       |       |  Sinon //je suis destinataire
|       |       |        listeGroup+= listeBesoin.IDF
|       |       |        Listebesoin=Concaténer ();// cette fct
|       |       |        consiste à    concaténer  les listes de
|       |       |        besoin reçu avec sa liste
|       |       fsi
|       |  Sinon  //j'ai reçu tous les messages attendus
|       |       Si (état=="leader membre")
|       |       |  Alors
|       |       |      IDFB:= tête.listeVoisin;
|       |       |      diffuse (message, Besoin, listeBesoin, IDFB);
|       |       fsi
|       fsi
Fincase
```

3.1.3. Vue du système obtenu après la première phase

A la fin de la première phase, le système sera composé de :

Leader déclencheur est leader dont il n'a pas de leader, i.e. il n'appartient qu'au groupe dont il est chef,

Leader membre est un nœud qui appartient à deux groupes de classe différente, un comme étant leader et autre comme étant membre,

Membre simple est un nœud membre d'un et un seul groupe et dont il n'est pas leader,

Leader singleton est un nœud qui n'appartient a aucun groupe (i.e. le nœud n'a pas de liaison stable avec ces voisins),

Un groupe global est composé de :

Un seul leader déclencheur,

Un ou plusieurs leaders membres,

Un ou plusieurs membres simples,

Deux groupes globaux sont disjoints, c'est-à-dire qu'ils n'ont pas de membres communs.

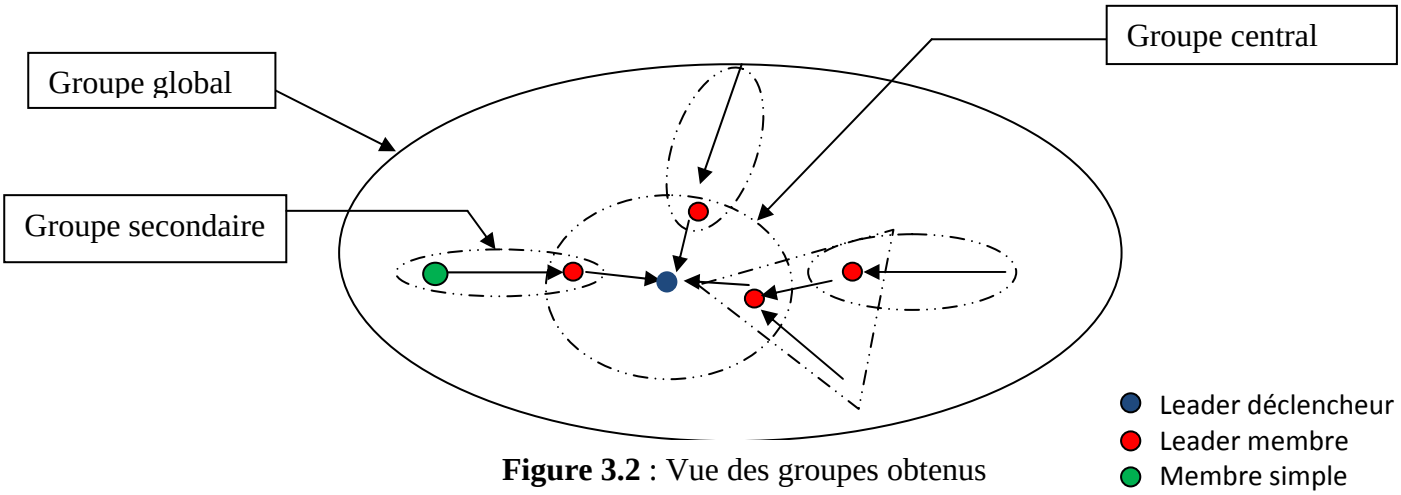
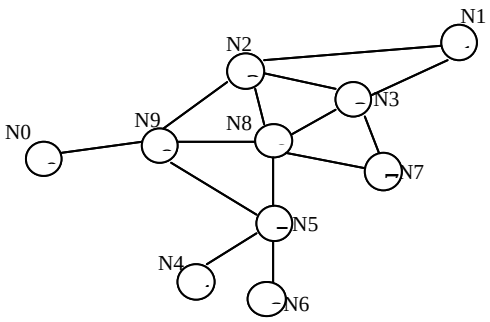


Figure 3.2 : Vue des groupes obtenus

Exemple

Dans cet exemple, nous considérons que la phase de construction des liens stables est déjà faite.

Identité	Cardinalité	Energie%
N0	1	70
N1	2	50
N2	4	30
N3	4	40
N4	1	50
N5	4	40
N6	1	60
N7	2	80
N8	5	40
N9	4	40



Après l'échange des vecteurs (Cardinale, Energie, Identité) entre les voisins qui s'est effectué dans la première étape, chaque nœud dispose d'une liste de vecteurs, qu'il va trier selon un ordre décroissant.

Les queues de listes qui sont des membres simples commencent à diffuser leurs listes des besoins contenant le IDFBesoin qui est la tête de ses listes (IDFBesoin = mon leader).

Dans notre cas les listes des voisins stables triées sont :

Nœud N0 :

N9	N0
4	1
40%	70%

Nœud N1 :

N3	N2	N1
4	4	2
40%	30%	50%

Nœud N2 :

N8	N9	N3	N2	N1
5	4	4	4	2
40%	40%	40%	30%	50%

Nœud N3 :

N8	N3	N2	N7	N1
5	4	4	2	2
40%	40%	30%	80%	50%

Nœud N4 :

N5	N4
4	1
40%	50%

Nœud N5 :

N8	N9	N5	N6	N4
5	4	4	1	1
40%	40%	40%	60%	50%

Nœud N6 :

N5	N6
4	1
40%	60%

Nœud N7 :

N8	N3	N7
5	4	2
40%	40%	80%

Nœud N8 :

N8	N9	N3	N2	N7
5	4	4	4	2
40%	40%	40%	30%	80%

Nœud N9 :

N8	N9	N5	N2	N0
5	4	4	4	1
40%	40%	40%	40%	70%

Les nœuds N0, N1, N4, N6, N7 passent directement à l'état membre simple et envoient leurs liste besoins a leurs leaders respectivement N9, N3, N5, N5, N8.

Le nœud N2 attends un seul message de N1, et comme N1 a choisit N3 alors N2 met a jour son état a membre simple et envoie sa liste besoin à son leader N8

N9 attend trois messages :

N9 accepte le nœud N0 dans son groupe,

Et reçoit un refus de N2 (car N2 a envoyé sa liste besoin au nœud N8),

Et attend un message de type besoin de N5

N3 attend trois messages :

N3 accepte le nœud N1 dans son groupe,

Reçoit un refus de N7 (car N7 a envoyé sa liste besoin au nœud N8),

Reçoit un refus de N2 (car N2 a choisi comme leader N8),

N3 a reçu tous les messages attendus, alors met a jour son état à leader membre puis envoi la liste des besoins de son groupe (N3 et N1) au nœud N8.

N5 accepte les nœuds N4 et N6 dans son groupe, passe à l'état leader membre et envoi la liste des besoins de son groupe (N4, N6 et N5) a son leader N8.

N9 reçoit un refus de N5 (car N5 a choisit N8),

N9 a reçu tous les messages attendus, met à jour son état à leader membre et envoi la liste besoin de son groupe (N9 et N0) au nœud N8.

N8 accepte les nœuds N2, N3, N5, N7, N9 dans son groupe et comme il n'a pas de leader alors N8 passe à l'état leader déclencheur et enfin lance la phase de réplication.

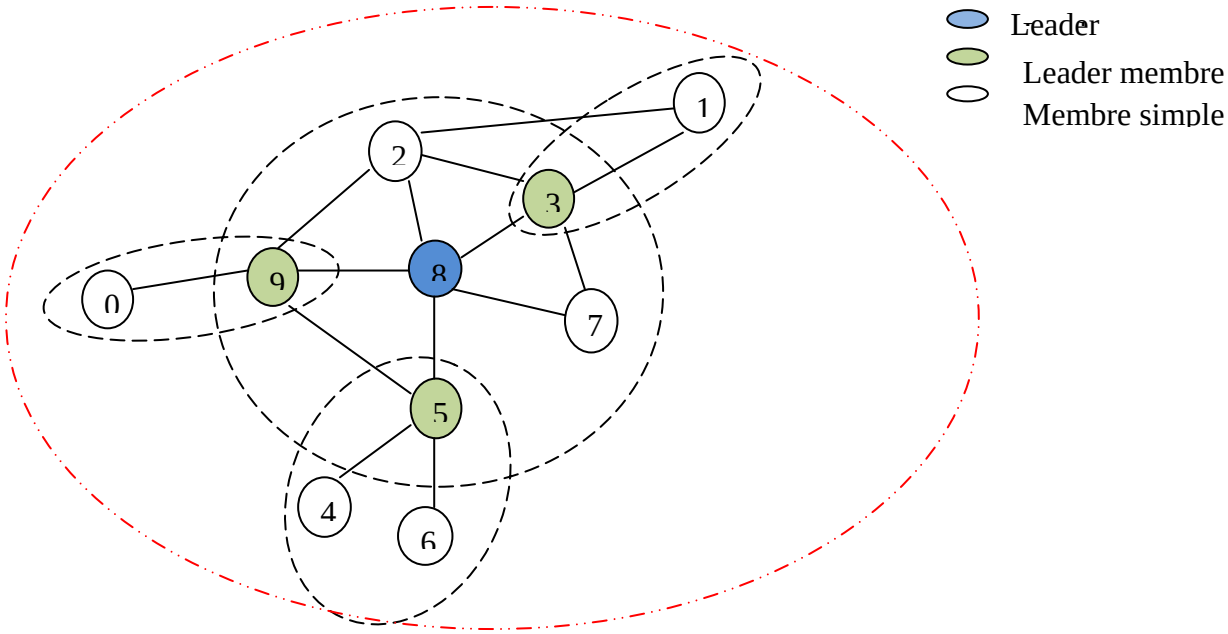


Figure 3.3 : Résultat de la phase de construction des groupes

3.2. Phase de réplication

Dans cette phase nous utilisons un seul type de message.

Message	Explication
Ordre	Ce message est utilisé pour transmettre l'ordre de réplication Tab_Ordre du *Leader déclencheur* à tous les membres du groupe global et contient aussi un entier Hop Et une Adresse AddL

Le champ Hop est un entier qui désigne le nombre de sauts séparant le nœud récepteur de son leader déclencheur contenu dans le deuxième champ AddL.

Structure de données

Chaque nœud du réseau possède les champs suivants :

Champ	Explication
IDF	Identificateur du nœud
List_voisin	La liste des nœuds voisins
List_groupe	La liste des nœuds de mon groupe (secondaire)
Etat	Etat du nœud (libre, Attente, Leader déclencheur, leader membre, membre simple, leader singleton)
Cardinalité	le nombre de voisins stables
Energie	valeur du niveau de batterie
Donnees_dispo	tableau des données disponible dans le groupe global, Donnees_dispo[i]=j veut dire que le nœud Ni contient la donnée Dj.
Mon_leader_declencheur	l'identificateur de mon leader déclencheur.
Nb_Hop	le nombre de saut me séparant de Mon_leader_declencheur.
Mon_leader	l'identificateur de mon leader direct.

Le but de cette phase est de placer les données les plus demandées par le groupe global auprès des nœuds qui les demandent le plus afin de rapprocher les données aux nœuds.

3.2.1. Principe

A la fin de la phase précédente, le nœud leader déclencheur aura reçu la liste des besoins de tous les nœuds aux quels il est connecté (lien à un saut ou plus). Ainsi c'est à lui qu'incombe la responsabilité de répliquer les données pour analyser et synthétiser les besoins réels du groupe, et, minimiser ainsi le trafic au maximum en offrant une meilleure accessibilité interne au groupe.

Le leader déclencheur commence par calculer la fréquence d'accès du groupe pour chaque donnée en additionnant les fréquences d'accès de tous les nœuds. Ensuite, il procédera à la construction de l'ordre de réplication qui contient les données qui doivent être répliqué

par chaque nœud. Pour ce faire, il commence par la donnée qui a la fréquence d'accès la plus grande (plus demandé par le groupe) puis il cherchera le nœud le plus approprié à répliquer cette donnée (celui qui a la plus grande fréquence d'accès à celle ci) à condition qu'il lui reste de l'espace pour répliquer la donnée. Sinon, c'est au nœud qui le suit en importance sur la fréquence d'accès, ainsi la donnée sera ajoutée à l'ordre de réplication (Tab_Ordre). Cette opération sera reprise jusqu'à épuisement de l'espace mémoire.

Enfin, Le leader déclencheur envoie l'ordre de réplication à ses membres directs.

Les leaders membres exécutent l'ordre de réplication puis envoient à leur tour l'ordre de réplication à leurs membres.

Enfin, les membres simples exécutent l'ordre réplication seulement.

Algorithme

Structure de données

Ordre [nb_node] tableau d'ordre de réplication, Ordre [Ni]=Dj veut dire que le nœud Ni doit répliquer la donnée Dj.

```
Programme_principal
Début
        Si((phase_replication) ET (Node->état = 'leader
        déclencheur'))
                Alors    Lancer_ réplication ( Node ) ;
    fsi
Fin


Lancer réplication (Node)
Début
 String Ordre [nb_node]
     While (Espace>=0)
     Début
          Data =Recherche_donnee_plus_importante();
          Node =Rechercher_noeud_approprie(Data);
          Ordre [Node] =Data à répliquer ;
          Espace = Espace - taille (Data à répliquer)

 Fin_while

Fin
```

3.2.2. Vues du système après la deuxième phase

A l'issue de la deuxième phase, chaque nœud connaît : Son leader déclencheur, Le nombre de saut qui le sépare de son leader déclencheur, Tous les nœuds du groupe global dont il est membre.

Pour chaque nœud du groupe global son niveau hiérarchique par rapport à lui (supérieur ou inférieur).

Toutes les données qui doivent coexister dans le groupe globale et le nœud qui les détient, et ceci grâce à l'ordre de réplication qui contient a chaque Ni un ensemble (D1, D2,…D espace).

Chaque leader membre connait ses membres.

Chaque leader déclencheur connait tous les membres de son groupe global.

Chaque groupe global est identifié par un identifiant unique (l'identifiant du leader déclencheur).

La seule information qui n'est pas détenus par un nœud est la liste des nœuds de même groupe (secondaire) sauf pour les leaders.

Toutes ces informations rendent le système distribué, par conséquent toute décision sera prise immédiatement et sans solliciter les nœuds voisins ou distant (a un saut ou plus).

Exemple

(Suite de l'exemple précédent)

Afin de simplifier l'exemple en suppose que la capacité de stockage d'un nœud est d'une (01) case.

Au début de cette phase le leader déclencheur calcul le besoin de son groupe en additionnant les besoins de tous les membres.

Soit la table des besoins suivants (10 données) :

	D0	D1	D2	D3	D4	D5	D6	D7	D8	D9
N0	2	5	4	1	1	1	4	3	1	1
N1	0	10	0	1	2	2	2	3	1	2
N2	0	4	1	1	1	0	5	1	3	1
N3	1	1	2	8	2	1	2	1	3	0
N4	7	1	2	2	3	4	1	2	3	2
N5	4	0	2	0	0	0	0	2	4	3
N6	1	1	0	9	2	3	0	1	5	2
N7	7	1	7	3	0	3	3	2	0	0
N8	0	2	5	1	3	5	0	1	5	1
N9	3	5	1	0	3	3	4	4	6	2

On peut interpréter ce tableau comme suit : le nœud Ni a fait TAB [I, J] requête a la donnée Dj.

Comme le nœud N8 est leader déclencheur de tous les nœuds, alors la liste des besoins globaux est :

données	D0	D1	D2	D3	D4	D5	D6	D7	D8	D9
Besoins globaux	25	30	24	26	17	20	21	19	31	14

Le leader déclencheur N8 construit alors l'ordre de réplication comme suit :

Tant qu'il y a de l'espace faire

1-Chercher la donnée la plus importante dans le groupe : Dj

2-Chercher le nœud le plus approprie à répliquer Dj : Ni

3-décrémenter la valeur de l'espace de stockage.

A la fin on obtient le tableau suivant :

Données	D0	D1	D2	D3	D4	D5	D6	D7	D8	D9
Nœuds	N4	N1	N7	N6	N3	N8	N2	N5	N9	N0

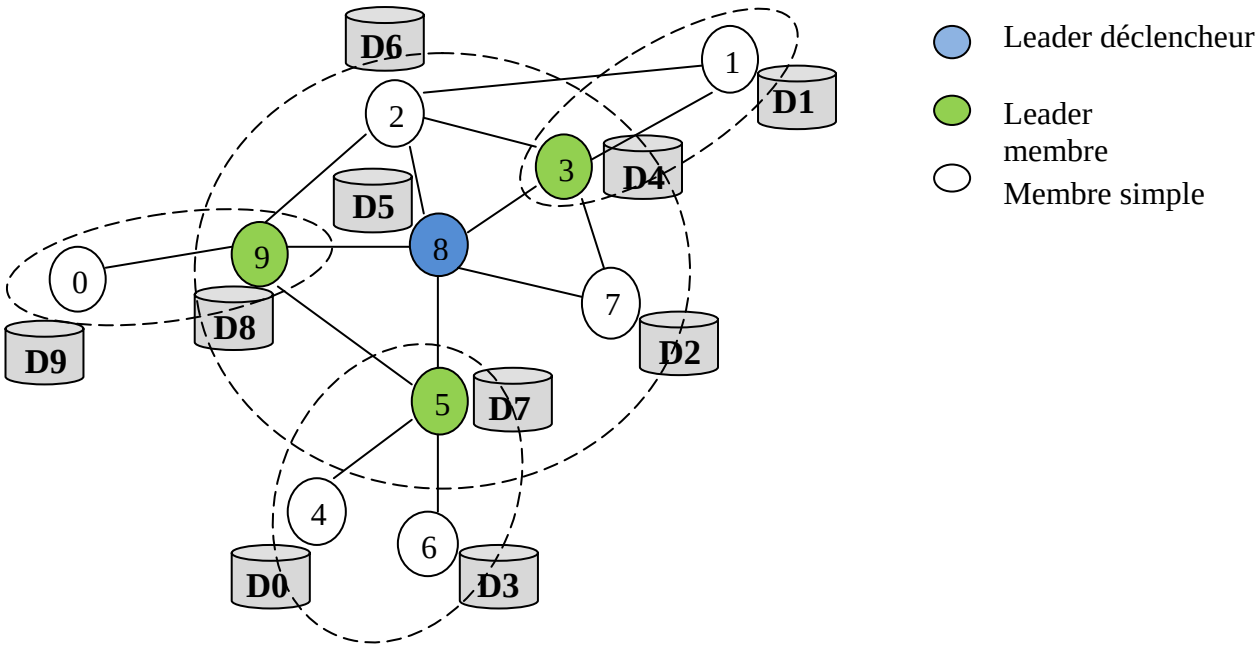


Figure 3.4 : Résultat de la phase de réplication

3.3. Gestions de l’accès aux données :

3.3.1 Accès en lecture:

Lors de l'accès à une donnée par un nœud, nous avons deux comportements possibles:

- Si le nœud ne dispose pas de la donnée, il transmet une requête au serveur primaire du groupe pour la donnée afin que ce dernier lui envoie un replicas de la donnée (ce mécanisme nous permet d’avoir un algorithme décentralisé du moment que chaque nœud connait le serveur primaire de chaque donnée alors il lui transmit directement sa demande sans avoir à passer par le leader déclencheur ceci permet d’alléger le leader et de décentraliser les requêtes de lectures). la procédure de recherche du serveur primaire de la donnée se fait par une diffusion de la requête sur les voisins stables à leurs tours les nœuds qui reçoivent cette requête la diffusent de la même façon jusqu'à localisation du serveur primaire. Un nœud qui reçoit ce message une deuxième fois alors qu’il ne fait pas partie du même groupe rejette le message, ce processus nous permet de réduire la diffusion au sein de même groupe (d’où l’avantage des groupe). Si le nœud ne connaît pas de serveur primaire pour la donnée, il transmet une requête au Leader qui exécutera une procédure de recherche localement au groupe et inter groupe afin de récupérer la donnée et de lui désigner un serveur primaire.

- Si le nœud dispose déjà d'une copie, il vérifie le numéro de version de la donnée locale avec le dernier rapport d'invalidation qu'il a reçu du serveur primaire de la donnée. Si le numéro est différent, c'est que le primaire dispose d'une version plus récente et donc il transmet au primaire une requête afin d'obtenir la dernière copie de la donnée.

Algorithme :

Demande_lecture (Dj)

```
Begin
    Si (Dj est dans node->data)     Le nœud dispose d'une copie de Dj.
        Si (Version_node->data (Dj) != invalidation_node->data(Dj))
        (la copie locale est invalide)
            - recupére_copie (Dj) ;
        Sinon (copie locale valide)
          Lecture locale.
    Sinon (le nœud ne dispose pas de copie de Dj)
        Si (node->primaire (Dj)!=NULL) (le groupe à une copie primaire de
Dj)
            -recupére_copie (Dj) ;
        Sinon (le groupe n'a pas d'une copie primaire de Dj)
            Envoie demande lecture aux leader () ;
End.
```

Suite de la Fonction réception :

```
Case (demande_lecture)  // le leader reçois une demande de lecture
    Si (node->état=leader)
        //  Recherche d'un serveur primaire de Dj dans le réseau
        diffusse_message ('recherche') ;
Fin ;
```

3.3.2. Accès en écriture:

La mise à jour s'effectue sur deux niveaux : mise à jour interne au groupe et la mise à jour globale inter groupe :

3.3.2.1. Mise à jour interne au groupe :

Un nœud envoie les mises à jour au primaire de la donnée dans son groupe. Le nœud primaire reçoit la requête comportant l'opération de mise à jour à effectuer, il effectue l'opération et il incrémente le numéro de version de la donnée. Il diffuse alors un rapport d'invalidation à l'ensemble des nœuds du groupe. Ce rapport est un

message court qui comporte l'identité de la donnée, du primaire, du groupe, et le numéro de version. En ce qui est de la diffusion à l'intérieur du groupe, il suffit que le primaire envoie les requêtes sur ces liaisons stables. Un nœud qui reçoit une première fois ce rapport et qui appartient au même groupe enregistre le message et diffuse à son tour le message sur ses liaisons stables. Un nœud qui reçoit ce message alors qu'il n'appartient pas au même groupe va ignorer le message et stopper sa progression à son niveau pour limiter la diffusion, Evidemment, toute nouvelle réception du même message est ignorée.

Algorithme :

Envoi mise a jour(Dj)

Begin

Si (node->état !='serveur primaire de Dj')

Je ne suis pas un serveur primaire et je veut faire une mise à jour de Dj, alors j'envoie une demande de mise à jour au serveur primaire de Dj.

Deux cas se présente :

Si je connais le serveur primaire de la donnée alors :

- Envoie message ('type=Mise à jour', 'destinataire=serveur primaire','donnée=Dj') ;

Sinon

Je ne connais le serveur primaire de la donnée Dj:

- Envoie message ('type=Mise à jour', 'destinataire=leader', 'donnée=Dj') ;

Sinon (je suis serveur primaire et je veux faire une mise à jour de Dj)

- mise_a_jour(Dj);
- Envoie le rapport d'invalidation pour la donnée Dj sur tout le réseau.

End.

3.3.2.2. Mise à jour inter groupe :

Pour permettre la propagation de la mise à jour vers les autres groupes, un nœud ne possède pas la connaissance sur les autres groupes, alors il confie cette mission à son leader en lui envoyant une demande de mise à jour globale, ce dernier détient la liste des nœuds à travers lesquels il peut accéder aux autres groupe.

Afin de construire cette liste, le leader de chaque groupe exécute une procédure de découverte après la construction de groupe. Cette procédure consiste à la diffusion d'un message découverte vers tous les nœuds, à la réception de ce message par un nœud, ce dernier vérifie s'il appartient au même groupe que l'émetteur, si oui il diffuse ce message à son tour, si non il indique à son leader que le nœud émetteur est un nœud de passage vers un autre groupe.

Le leader envoie la demande de mise à jour aux autres groupes via cette liste, à la réception de ce message par les nœuds concernés, ces derniers exécuteront la procédure de mise à jour interne et inter groupe.

Algorithme :

```
Fonction_ Découverte () ;
begin
        Si (node->état = leader_groupe)
                Diffusse_message ('découverte') ;
End ;
Suite Fonction réception :
Case (découverte)
                Si (node->leader=sender->leader)
                        Diffusse_message ('découverte') ;

                Sinon                   // j'ai recu un message d'un autre groupe
                        Liste_groupe->id=sender->leader;
                        Liste_groupe->passage=sender->id ;

Fin ;
```

L'algorithme est composé de quatre procédures :

- l'envoie des demandes de mise à jour
- le traitement des demandes de mises à jour
- l'envoie de rapport d'invalidation
- le traitement d'un rapport d'invalidation

Reception_mise_a_jour(Dj)

Begin

Si (node->état =' serveur primaire de Dj')

Je ne suis pas serveur primaire de Dj et j'ai reçu une demande de mise à jour, alors Deux cas se présente :

- J'ai déjà reçue ce message alors ne rien dropper le message.
- c'est la première fois que je reçois ce message :

  - mise_a_jour(Dj);
  - version Dj++
  - Envoie le rapport d'invalidation pour la donnée Dj sur tout le réseau.

Sinon

Je ne suis pas serveur primaire et j'ai reçu une demande de mise à jour, alors Deux cas se présente :
-J'ai déjà reçue ce message alors ne rien faire et dropper le message.
- c'est le première fois que je reçois ce message : alors diffusion vers les nœud voisin stables.

End.

Reception_rapport_d'invalidation (Dj)

Begin

Si Dj n'est pas une copie primaire

Deux cas se pressentent
- J'ai déjà reçue ce message alors ne rien dropper le message.
- c'est la première fois que je reçois ce message :

  -Enregistrer le numéro de la version reçu dans le rapport d'invalidation
  - diffusion vers les nœuds voisins stables

Sinon

- ne rien faire.

End.

Discussion :

Un nœud Ni appartenant à un groupe X veut faire une mise à jour d'une donnée DJ :

- Ni vérifier s'il est serveur primaire de Dj en consultant sa table de réplication reçu lors de la phase réplication :
  - Si Ni est SP alors il exécute la mise à jour localement, incrémente la version de DJ et envoie le rapport d'invalidation de Dj ; ce rapport contiendra la référence de Dj ainsi que le numéro de la version de Dj.
  - Sinon Ni n'est pas SP de Dj, Ni recherche le SP de Dj dans sa table de réplication s'il la trouve alors il lui envoie une demande de mise à jour de Dj, sinon il envoie la demande au leader principale, ce dernier à la réception de ce message vérifie dans sa table de réplication s'il y a un SP de Dj dans le groupe, si oui il le transmet la demande de mise à jour, sinon (il n y a pas un SP pour DJ dans le groupe) il transmet la demande vers les leaders des autres groupes.
- A la réception d'une demande de mise à jour, si le récepteur est destinataire du message, il vérifie s'il est SP de Dj si oui il exécute la mise à jour localement, incrémente la version de Dj et envoie le rapport d'invalidation de Dj, si non (je suis destinataire du message et je ne suis pas le SP de Dj, donc je suis le leader du groupe) le groupe ne dispose pas de SP pour Dj alors la demande va être envoyé vers les leader des autres groupes dans le réseau.
- A la réception d'un rapport d'invalidation, le nœud vérifie la version du rapport avec la dernière version reçue, si la version du rapport est plus grande alors il sauvegarde le nouveau rapport et diffuse le message à son tour, dans le cas contraire le message sera rejeté.

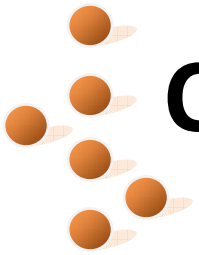
4. Conclusion

Nous avons proposé un protocole de réplication basé sur une répartition en groupe d'un réseau mobile ad hoc. La notion de groupe est très intéressante pour les réseaux mobiles ad hoc car par définition un réseau mobile ad hoc peut se retrouver répartir-en sous réseaux.

Notre algorithme de construction de groupe est un algorithme complètement distribué non centralisé. Aucun nœud ne joue un rôle de contrôle continu. La réplication des données

est faite de manière à maintenir une copie de la donnée au niveau de chaque groupe et de son voisinage direct. L'architecture en groupes proposée est exploitée pour optimiser les accès.

Un protocole d'accès aux données avec mise à jour a été proposé, cet algorithme est de type algorithme à invalidation que nous avons adapté avec la notion de groupe.



Chapitre 4

Simulation & Evaluation des performances

1. Introduction :

Afin d'évaluer les performances de notre algorithme de réplication et d'accès aux données, nous avons utilisé le simulateur GloMoSim, ce dernier nous permet d'avoir une vue assez précise sur les caractéristiques fondamentales et les comportements d'un système mobile, à 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.

2. GloMoSim

GloMoSim est un simulateur réseau modulaire pour la simulation et l'évaluation des performances des réseaux sans fil (locaux ou étendus) [Bzg98]. Il utilise la simulation parallèle à événements discrets fournie par le langage PARSEC (Parallel Simulation Environment for Complex System) développé par PCL (*Parallel Computing Laboratory*) de l'université UCLA (*University of California at Los Angles*). Parsec permet de bien séparer la description du modèle de la simulation de sa nature d'exécution (parallèle ou séquentielle). Pour réaliser une simulation, GloMoSim utilise une approche simple. En effet, si nous supposons que chaque nœud dans la simulation est une entité à part, alors nous allons remarquer non seulement une dégradation mais aussi une limitation des performances de la simulation car l'initialisation d'un millier d'entités peut influer sur le temps d'exécution et sur l'utilisation de l'espace mémoire. Pour palier à ce problème, GloMoSim a recours à une approche d'agrégation de nœuds où une seule entité peut simuler plusieurs nœuds du réseau dans le système [Bta00]. Chaque nœud a sa propre structure de données et son code qu'il doit s'exécuter sans interférence ni violation d'accès aux structures de données des autres nœuds. Chaque nœud à son propre état et l'ensemble des états complets de tous les nœuds est maintenu par l'entité. Cette dernière approche permet d'augmenter le nombre de nœud dans une simulation sans augmenter le nombre d'entité. Une entité n'est autre qu'une zone géographique où chaque nœud est déterminé par ses coordonnées dans cette zone.

2.1. Architecture

GloMoSim a une structure de couches proche de celle du modèle OSI [Dar99]. En effet, chaque entité intègre les diverses couches réseaux qui peuvent être simulées sous forme de fonction. Au début de la simulation une fonction d'initialisation est appelée pour chaque couche de chaque nœud. A la réception d'un message, une couche exécute la tache

demandée. A la fin de la simulation, une nouvelle fonction est lancée pour mettre fin à la simulation et collecter les statistiques désirées. Pour faciliter la communication entre les diverses couches réseaux, un ensemble d'API (Application Programming Interface) est spécifié sous forme de messages échangés entre les couches.

Couche Application (CBR, HTTP, Telnet, FTP)
Couche Transport (TCP, UDP)
Couche Réseau (IP avec AODV, Flooding, Bellman-Ford, OSPF, DSR, WRP)
Couche Mac (CSMA, MACA, MACAW, FAMA, 802.11)
Couche Radio (Free Space, Rayleigh, Ricean, SIRCIM)
Couche Mobilité (Random drunken et Random waypoint)

Tab 4.1 : les couches de Glomosim.

2.2. Configuration de la Simulation

GloMosim prend en entrée un fichier décrivant la description de la simulation. Ce fichier (en format .in) contient un ensemble de variables de paramètres de la simulation avec les valeurs correspondantes. Parmi ces variables on trouve :

- Simulation-Time : temps de simulation en jour, heure, seconde, microseconde et nano-seconde. Si on ne précise pas l'unité, la seconde est utilisée par défaut.

- Terrain-Range-X : Largeur du terrain de la simulation en mètre ;
- Terrain-Range-Y : Longueur du terrain de la simulation en mètre ;
- Number-Of-Nodes : nombre des nœuds réseaux dans la simulation ;
- Node-Placement et Node-Placement-File : décrivent les emplacements des nœuds dans le terrain : (aléatoire, uniforme, rangés, en groupe, etc.) ;
- Mobility : spécifie si les nœuds sont mobiles ou pas. Si une mobilité est activée, on spécifie alors la nature de mobilité (aléatoire ou pas), la liste suivante présente les différents types de mobilité :

```
- MOBILITY TRACE
- RANDOM-DRUNKEN
- MOBILITY-TRACE-FILE ./mobility.in
- MOBILITY PATHLOSS-MATRIX
- MOBILITY RANDOM-WAYPOINT
    MOBILITY-WP-PAUSE      30S
    MOBILITY-WP-MIN-SPEED  0
    MOBILITY-WP-MAX-SPEED  7
- MOBILITY NONE
```

La mobilité None implique que la vitesse de déplacement des nœuds est égale à zéro. Dans ce type de mobilité les nœuds ne subissent aucun mouvement.

Pour le model RANDOM-DRUNKEN, si un nœud est actuellement à la position (x, y), il peut probablement se déplacer à (x-1, y), (x+1, y), (x, y-1), et (x, y+1) sur le terrain physique. Pour RANDOM WAYPOI NT, un nœud choisit aléatoirement une destination sur le terrain physique. Il se déplace vers cette destination avec une vitesse uniformément choisie entre MOBILITY-wp-min-speed et MOBILITY-wp-max-speed (meter/sec). Après ait atteint sa destination, le nœud reste là pour une période de temps précisée par le paramètre MOBILITY-wp-pause.

- Power-Range : représente les limites de la portée de communication de chaque nœud.
- Bandwith : représente la bande passante utilisée par les nœuds pour transmettre des messages.
- Application : application à simuler (ftp, cbr, http ou telnet) ;
- Routing-Protocol : protocole de routage utilisé ;
- Transport-Protocol : protocole de transport utilisé (UDP ou TCP) ;
- MAC-Protocol : protocole de la couche Mac (802.11, CSMA, FAMA
(Floor Acquisition Multiple Acces), MACA(Multiple Access Collision Avoidance);
- et les paramètres de statistiques et d'autres paramètres optionnels

3. Paramètres de la simulation :

Les paramètres de la simulation permettent de définir le choix de l'environnement de l'exécution de notre algorithme étant donnée la surface du terrain, le nombre de nœud, le modèle de mobilité des nœuds.

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.

La variation de ces paramètres nous permet d'effectuer plusieurs tests sur notre algorithme.

- **Le modèle de mobilité**

Le simulateur GloMoSim présente deux principaux modèles de mobilité le modèle RANDOM-DRUNKEN et le modèle RANDOM-WAYPOINT.

Dans le modèle RANDOM-DRUNKEN, les nœuds choisissent leurs destinations en fonction de leurs positions actuelle. Si un nœud est dans la position (X, Y), son prochain déplacement s'effectuera vers l'une des quatre positions avoisinantes (x-1, y), (x+1, y), (x, y-1) ou bien (x, y+1).

Le deuxième modèle de mobilité RANDOM-WAYPOINT, qui est le modèle que l'on a choisi car c'est le modèle le plus proche du déplacement réel des nœuds d'un réseau mobile Ad hoc. Les nœuds choisissent au hasard chacun une destination (X, Y) vers laquelle ils vont se déplacer avec une vitesse maximum VMax. Nous l'avons fixé à 5m/s par défaut, et varié entre 2 et 10m/s dans les évaluations par rapport à la vitesse.

Après avoir atteint sa destination, chaque nœud doit prendre un temps de pose que nous avons fixé à 30s.

- **La surface de simulation**

Les dimensions du terrain de simulation qu'on a choisi par défaut sont 1500m×1500m, cette surface peut aller jusqu'à 2000m×5000m pour les graphes de passage à l'échelle (200 nœuds et plus).

La commande qui spécifie la surface du terrain de simulation est la suivante.

TERRAIN-DIMENSIONS (1500, 1500) ;

- **La couche MAC :**

Le protocole de communication qu'on a choisi pour cette couche est le 802.11. Avec un rayon de communication (portée) de 250métres.

- **Le protocole de routage** :

Le simulateur GloMoSim offre la possibilité d'utiliser plusieurs protocoles de routage, notre choix s'est porté sur le protocole AODV.

Le tableau suivant résume l'ensemble des variables utilisées durant la simulation :

Paramètres	Valeur par défaut	Intervalle de variation
Nombre de nœuds	50	25-200
Nombre de données	50	
Surface	1500mX1500m	1500X1500– 2000X5000
Portée (mètres)	250	
Temps de simulation min	10	
Vitesse Max (m/s)	5	2-10
Capacité de stockage	4	2-8
Période de réallocation (s)	60	

- **La charge du réseau**

Ce paramètre a une relation directe avec la consommation de la bande passante et l'augmentation de trafic. Une consommation importante de la bande passante peut dégrader les performances du réseau et augmente le risque de collusion, ce qui agit négativement sur la disponibilité de donnée.

- **Capacité de stockage**

La taille de stockage des nœuds est un paramètre important pour augmenter la disponibilité et réduire le trafic. Nous allons varier ce paramètre (entre 1KO et 50KO) pour observer ses effets sur les métriques choisis ci-après. La taille du cache par défaut est 20KO.

- **La portée de communication**

La portée de communication est délimitée par le rayon de puissance de communication des nœuds. Ce paramètre est accessible en utilisant la variable POWER-RANGE du fichier de configuration. La valeur par défaut de ce paramètre est 50M et l'intervalle de variation est [1m, 300m].

- **Scalability**

Le but de ce paramètre est de suivre le comportement d'un protocole de réplication de données, quand le nombre de nœuds participant augmente. Le protocole de réplication doit toujours donner un taux d'accessibilité acceptable et réduire le nombre de message

transmis. Dans ce paramètre nous allons varier le nombre de nœud, dans un terrain de 5000 M * 5000 M.

4. Paramètres de l'évaluation :

4.1 Taux d'accessibilité aux données :

Ce paramètre représente le taux de succès des requêtes d'accès aux données, après chaque période de réplication, chaque nœud exprime plusieurs requêtes et enregistre le nombre de requête satisfaite. A la fin de la simulation le taux d'accès aux données est calculé par la formule suivante :

$$ACC = \frac{NRS}{NRF}$$

ACC : le taux d'accessibilité

NRS : nombre de requêtes satisfaites durant toute la durée de simulation

NRF : nombre de requêtes formulées durant toute la durée de simulation

4.2 Mise 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 du groupe.

$$MajR = \frac{NbMaiE}{NbMaiT}$$

MajR : est le pourcentage de requêtes de mises à jour réalisées par les noeuds primaires.

NbMajE : est le nombre de mises à jour effectuées.

NbMajT : est le nombre total de mises à jour générées.

4.3. 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 invalide est la suivante :

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

IA : est le pourcentage d'accès invalides (Invalid Access).

NbIA : est le nombre d'accès invalides

SuccR : est le nombre de requêtes satisfaites ou réussies durant la simulation (Succeeded Requests).

4.4 Le trafic

Le trafic est le nombre de message transmis par tous les nœuds pendant la durée de simulation, formellement le trafic est défini comme suit :

$T = \sum_{i=1}^{n} NmesTr_i$ tel que n et le nombre de nœud et $NmesTr_i$ est le nombre de message Transmis par le nœud i durant la simulation.

5. Résultats :

Dans ce qui suit nous allons présenter les différents tests et mesures effectués afin d'évaluer notre protocole d'accès aux données avec mise à jour après la phase réplication basée sur les groupe.

Principalement les tests sont basés sur deux critères : le taux de réussite des accès aux données en lecture et en écriture et la validité des données. Les résultats présentés sont obtenus en fonction de la variation de plusieurs paramètres qui sont :

- La densité du réseau (le nombre de nœuds) : ce paramètre permet de tester le passage à l'échelle de notre algorithme.
- La mobilité des nœuds.
- La charge du réseau, le nombre de requêtes générées par les nœuds.

5.1 Le taux de succès

5.1.1 Variation du nombre des nœuds (passage à l'échelle)

Dans la figure suivante nous avons mesuré le taux de succès des requêtes en variant le nombre des nœuds. Le but de cette mesure est d'évaluer la capacité de passage à l'échelle de notre méthode de réplication et de la comparer à celle de DCG-S1 [Hara03][Hara06b].

Les nœuds peuvent se déplacer dans une surface fixée à 10km² (2000m×5000m) et ont une vitesse maximum de 5m/s, les nœuds ont chacun un espace mémoire pouvant supporter 4 données, et le nombre de données partagées entre les nœuds est de 50.
La portée de transmission est de 250 mètres.

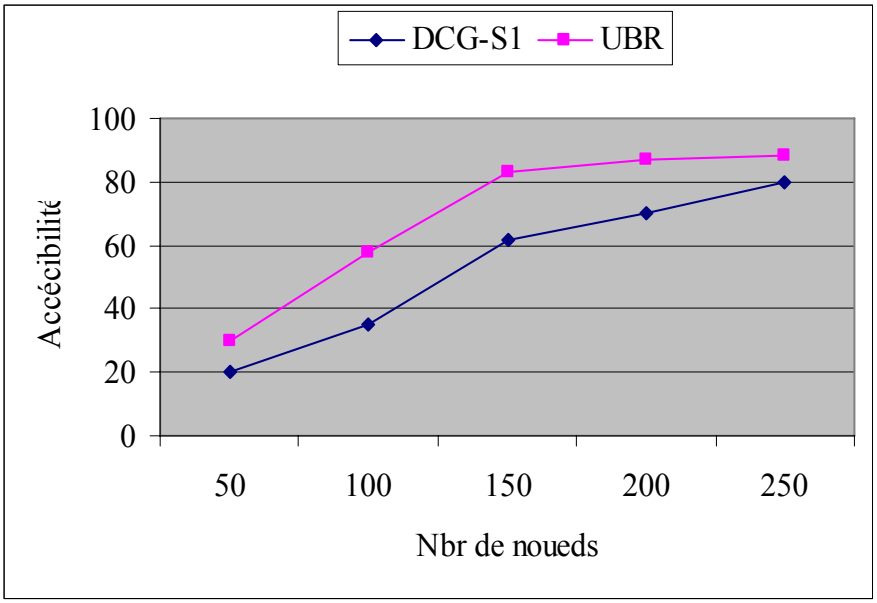


Figure 4.1 : taux de succès en fonction du nombre de nœuds

D'après l'allure des deux courbes, on constate que le taux de succès augmente proportionnellement avec le nombre de nœud, ceci est facilement compréhensible puisque on augmentant le nombre de nœud on augmente la densité du réseau c'est-à-dire le nombre moyen de nœuds par unité de surface donc les nœuds se retrouvent si souvent proche les uns des autres ce qui entraîne une construction des groupes de plus grands effectifs donc un espace de partage de données commun plus important. En effet, la présence de plus de noeuds 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 ne 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

5.1.2 Variation de la capacité de stockage

Dans cette évaluation, nous avons calculé le taux de satisfaction des requêtes en faisant varier la capacité de stockage des nœuds.

Le but de cette mesure est de montrer de quelle manière la disponibilité des données varient avec le changement de la capacité de stockage des nœuds.

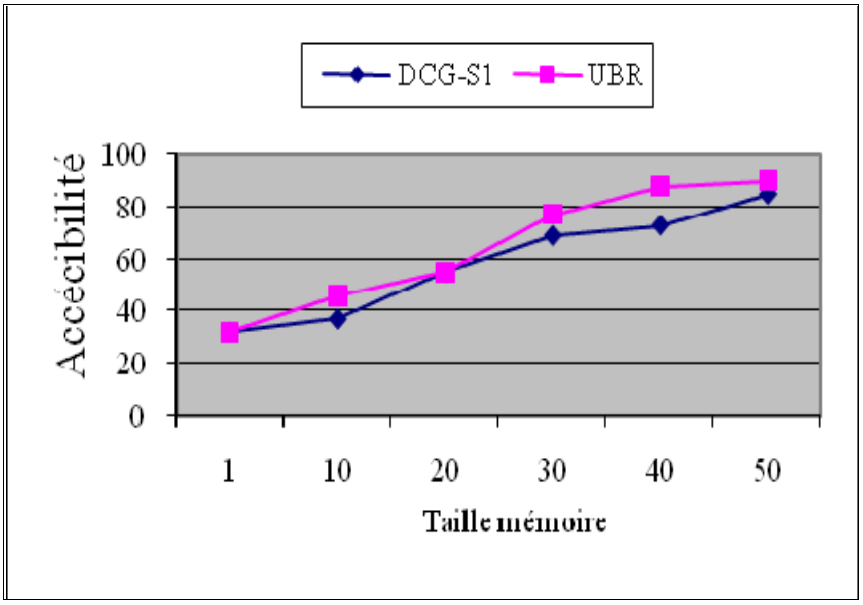


Figure 4.2 : taux de succès en fonction de la taille mémoire

D'après l'allure des deux courbes, on remarque le taux de sucées augmente avec l'augmentation de la capacité de stockage des nœuds.

Ceci est argumenté comme suit :

Lorsque les nœuds ont un espace de stockage faible par rapport au nombre de données dans le réseau, l'espace de partage de données (espace commun de réplication de données) sera faible donc on aura automatiquement un taux d'accessibilité faible ; puis on augmentant cette capacité de stockage, l'espace de partage de données augmente ce qui conduit a un taux de satisfaction important.

5.1.3 Variation de la surface

La densité est un paramètre important qui nous permet d'évaluer l'impact de connectivite sur notre solution, car avec les paramètres par défaut (mobilité, portée radio, nombre de nœuds…etc.), si on augmente la surface de déplacement des nœuds on change la connectivite car les nœuds seront éloigné et auront une liberté de déplacement importante.

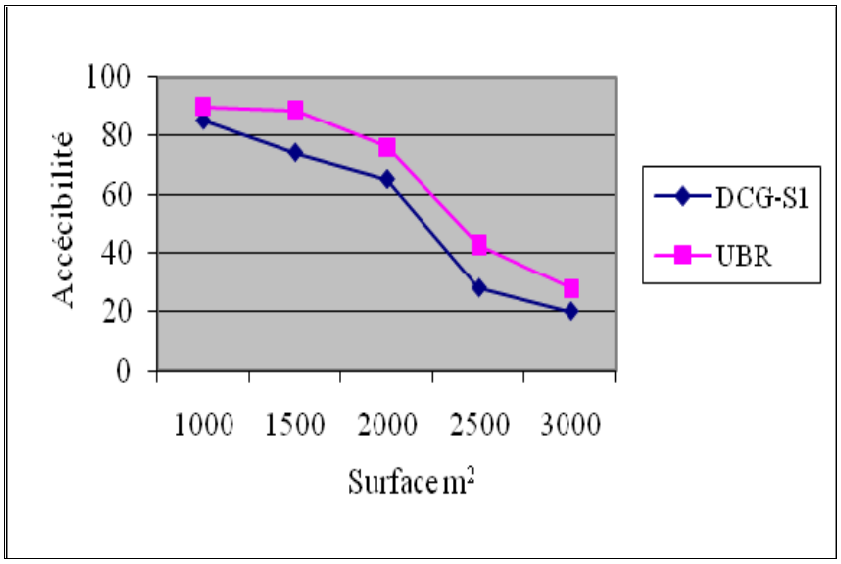


Figure 4.3 : taux de succès en fonction de la surface

L'allure du graphe montre que le taux de succès diminue considérablement avec l'augmentation de la surface ; ceci est totalement vrai car si la surface augmente la densité diminue donc la connectivite s'affaiblie d'où les nœuds vont se retrouver souvent seuls, donc on aura tendance à des singletons qui tente de répliquer individuellement les données dont ils ont le plus besoin ce qui entraînera à un taux de satisfaction des requêtes trop faible.

Il est donc clair qu'une construction de groupes stables est très importante pour une réplication efficace.

5.1.4 Variation de la vitesse

Dans la figure suivante nous avons mesuré le taux de succès aux requêtes par rapport à une vitesse variant d'un minimum de 2m/s jusqu'à 9m/s.

Le but de cette mesure est de tester la capacité de notre méthode à résister à l'augmentation de la vitesse des nœuds.

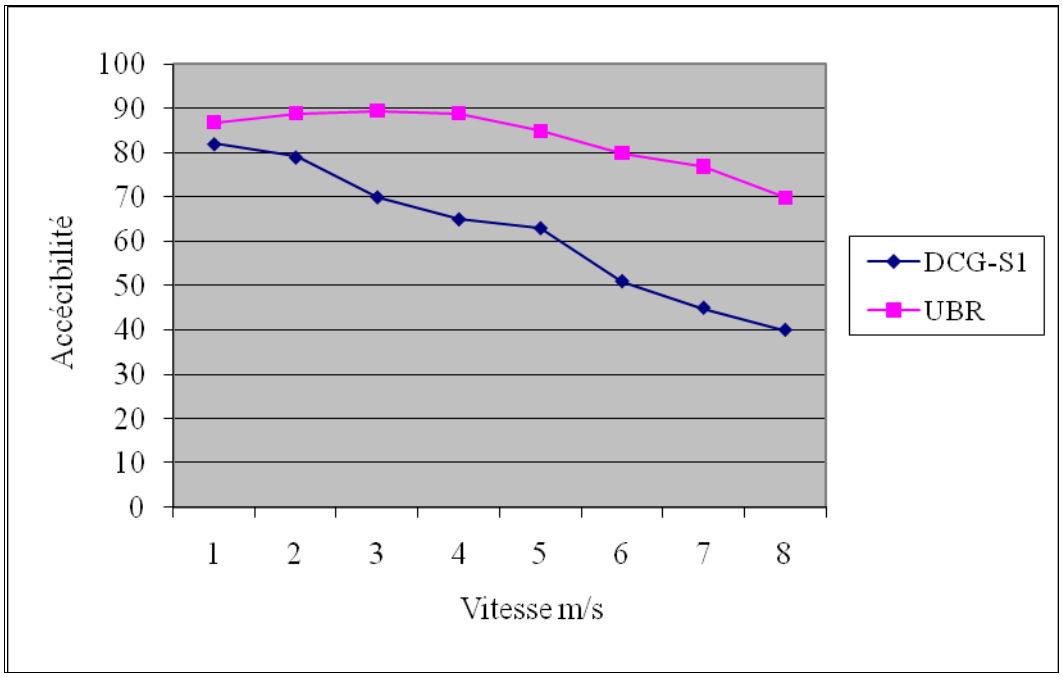


Figure 4.4 : Taux de succès en fonction de la vitesse.

L'image du graphe montre que notre solution résiste mieux à l'augmentation de la vitesse en gardant un taux de succès relativement stable.

Selon ces résultats de simulation, notre méthode permet une plus grande disponibilité par rapport à DCG-S1, et une résistance à la mobilité des nœuds, ceci est dû au fait que notre architecture en groupes hiérarchiques résiste à la mobilité des nœuds puisque un déplacement de certains nœuds n'influent pas trop sur le groupes.

5.2 Le taux de succès des mises à jour

5.2.1. Mises à jour réussies et variation de la densité:

Le premier résultat présenté montre l'impact de la variation de la densité par rapport aux nombre de mise à jour réussies, En observant le graphe nous pouvons constater que le taux de mise à jour réussies augmente par rapport à l'augmentation du nombre des nœuds, ceci est évidant car plus le nombre de nœud augmente moins en aura la chance d'être isolé et donc plus de chance pour atteindre le primaire.

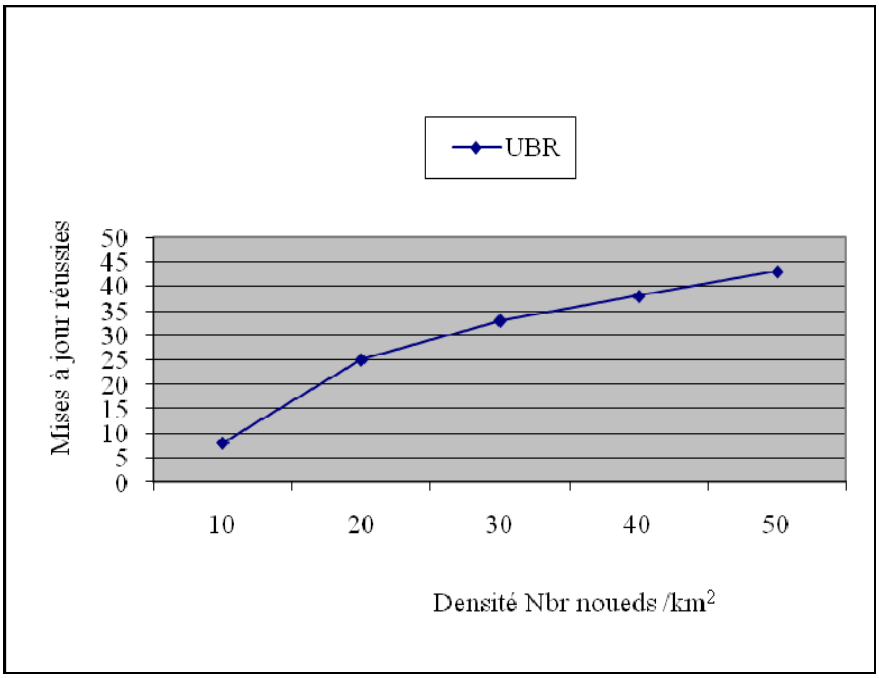


Figure 4.5 : mises à jour réussies en fonction de la densité.

La figure 4.5 représente le pourcentage du nombre de requêtes de mises à jour réussies en fonction de la densité et du passage à l'échelle. 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.

Plus la densité augmente, plus le nombre de mises à jour satisfaites augmente. 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.

5.2.2. Mises à jour réussies et variation de la vitesse:

Nous avons choisis de tester le protocole de mise à jour par rapport à la mobilité des noeuds et cela en variant les intervalles de vitesse comme suit:

- Vitesse faible : dans ce cas la vitesse de déplacement des noeuds varie entre 0 et 2 m/s.
- Vitesse moyenne : la vitesse de déplacement varie entre 2 et 4 m/s.
- Grande vitesse : la vitesse de déplacement varie entre 4 et 6 m/s.
- Très grande vitesse : la vitesse de déplacement varie entre 6 et 8 m/s.

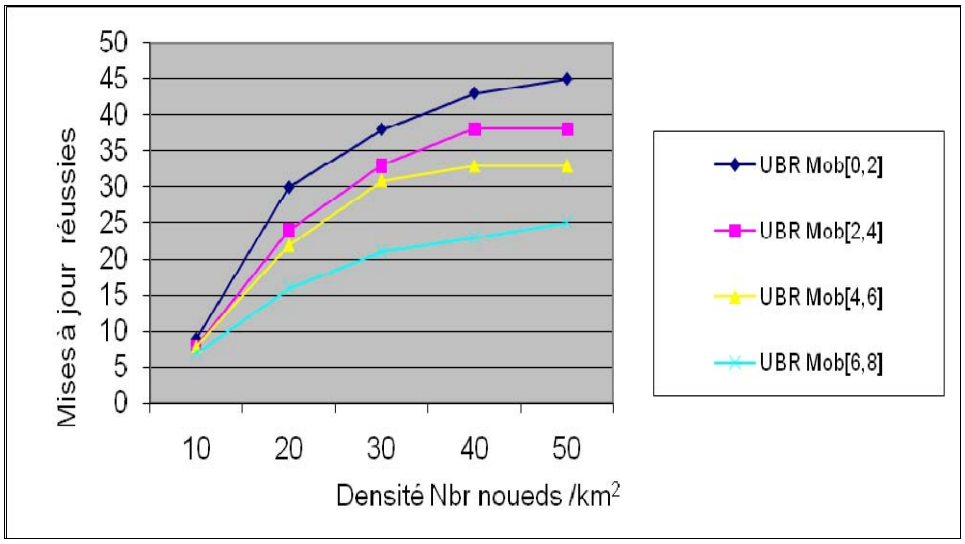


Figure 4.6: mises à jour réussies en fonction de la densité (variation de la vitesse).

La figure 4.6 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 noeuds. Nous observons que plus la vitesse augmente, plus le nombre des demandes de mise à jour effectuées diminuent. Ceci est dû au fait qu'une mobilité plus importante influence la topologie par des changements plus fréquents.

5.2. 3. Accès invalides

Les accès invalides ont été étudiés en fonction du nombre de requêtes par minute dans le réseau. 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 (30%, 20%, et 10 % de requêtes de mise à jour par rapport au nombre total des requêtes), et cela en variant le nombre total de requête.

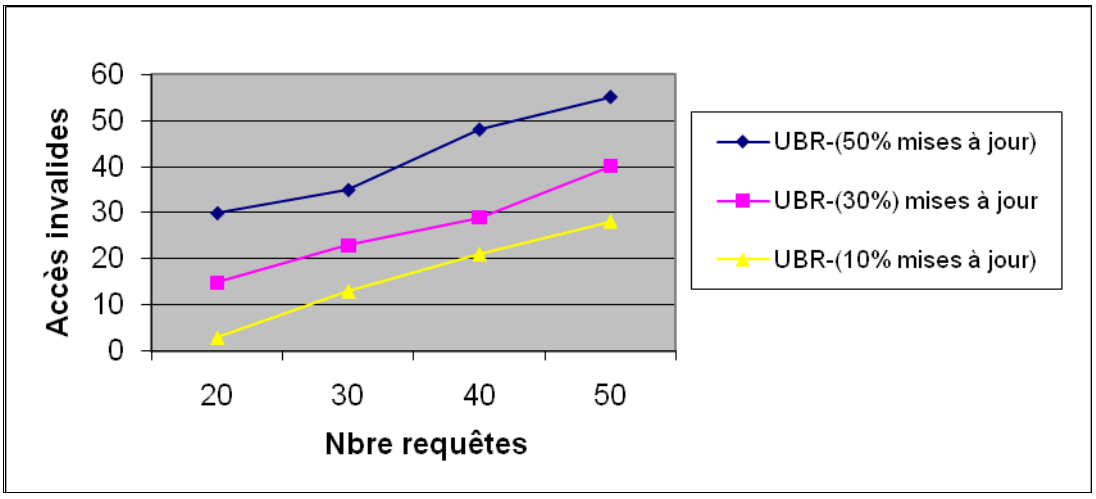


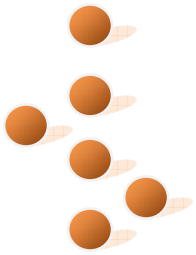
Figure 4.7: Accès invalides/Nbr requêtes.

La figure 4 .7, 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.

6. Conclusion

Les tests des performances de notre protocole montrent des résultats considérables en ce qui concerne l'accessibilité des donnés en lecture et ceci grâce à la notion de groupe qui permet d'avoir une bonne gestion de l'espace de stockage globale et de la réplication, Ces évaluations montrent que la notion de groupes utilisée permet d'avoir de bons résultats en cas de passage à l'échelle ;

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. 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.



Conclusion

Conclusion

L'accès aux données dans les environnements mobiles est un problème majeur dû à la nature de la communication sans fil, la réplication améliore l'accessibilité des données en termes de temps de réponse, de disponibilité et de fiabilité. Ceci ne peut être garanti que par une distribution efficace des données sur l'ensemble des nœuds du réseau et aussi par un algorithme de gestion d'accès en lecture et en écriture qui prend en considération la manière dont les données sont distribuées sur l'ensemble des nœuds. Un tel algorithme à pour but de donner un temps de réponse réduit quant à l'accès aux données et de diminuer la charge du réseau

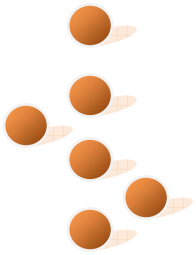
Notre travail consistait à trouver une solution pour la dissémination des données qui offrirait une meilleure disponibilité et cohérence. Nous avons d'abord proposé puis simuler un protocole de réplication dans les réseaux mobiles ad hoc. En premier lieu, nous avons étudié différentes approches de réplication, dont l'approche de réplication par groupe qui semble la plus intéressante. Dans la mesure ou elle offre un plus grand espace virtuel commun pour la réplication des données accédées par plusieurs membres d'un groupe.

Notre approche de réplication par groupe est basée sur la notion des liaisons stables. Cette stabilité des liaisons permet de rentabiliser au mieux la réplication par la limitation des copies de données redondantes.

L'idée du protocole se mesure par le fait de mettre les données qui ont la fréquence d'accès les plus importantes dans l'espace du groupe.

Nous avons aussi apporté une contribution au problème de la mise à jour de données partagées. Cette contribution est donnée à travers un protocole de mise à jour à invalidation que nous avons adopté avec la notion de groupe, cet algorithme s'exécute en deux niveaux internes au groupe l'autre inter les groupes du réseau, ce qui permet d'améliorer le degré de la cohérence de données localement dans un groupe.

Les tests des performances de notre algorithme montrent des résultats considérables en ce qui concerne l'accessibilité des donnés en lecture et ceci grâce à la notion de groupe qui permet d'avoir une bonne gestion de l'espace de stockage globale et de la réplication, Ces évaluations montrent que la notion de groupes utilisée permet d'avoir de bons résultats en cas de passage à l'échelle.



Références

[Bad] B. R. Badrinath, T. Imielinski: Replication and mobility.

[Bel 05] P. Bellavista, A. Corradi, and E. Magistretti. Comparing and evaluating lightweight solutions for replica dissemination and retrieval in dense manets. iscc, 0:43–50, 2005.

[D,Rat98] D. Ratner. Roam: A Scalable Replication System for Mobile and Distributed Computing. PhD thesis, UCLA, January 1998.

[Hara 03] T. Hara. Replica allocation methods in ad hoc networks with data update. Mobile Networks and Applications 8, pages 343–354, 2003.

[Hara,Y 03] T. Hara, Y.-H. Loh, and S. Nishio. Data replication methods based on the stability of radio links in ad hoc networks. dexa, 0:969, 2003.

[Hara 04] T. Hara, N. Murakami, and S. Nishio. Replica allocation for correlated data items in ad hoc sensor networks. SIGMOD Record, 33(1):38–43, Mar. 2004.

[Hung 06] J.-L. Huang and M.-S. Chen. On the effect of group mobility to data replication in ad hoc networks. IEEE Transactions on Mobile Computing, 5(5):492–507, May 2006.

[Har03] Takahiro Hara. "Replica allocation methods in ad hoc networks with data update", ACM-Kluwer Journal on Mobile Networks and Applications, vol 8(4), pp 343-354, 2003.

[Hara 06b] 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.

[Hys03] Takahiro Hara, Yin-Huei Loh, Shojiro Nishio. " Data Replication Methods Based on the Stability of Radio Links in Ad Hoc Networks"**,** Proceedings of the 14th International Workshop on Database and Expert Systems Applications (DEXA'03) 1529-4188/03 2003 IEEE.

[Har04] Takahiro Hara and Sanjay Kumar Madria. "Dynamic Data Replication Using Aperiodic Updates in Mobile Adhoc Networks", 9th international conference, DASFAA 2004

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

[Mco 00] M. Coglianese Optimistic Data Replication for Mobile Applications December 19, 2000.

[Mouss 07] S.Moussaui thèse doctorat: Contribution à l'Amélioration de l'Accessibilité des Données sur Réseaux Mobiles Ad Hoc, 2007

[Rat 03] D.Ratner, J.G.Popek, P.Reiher: The Ward Model: A Replication architecture for mobile environments.

[Rei 01] D. Ratner, P. Reiher, G. J. Popeky, G.H. Kuenning; Replication requirements in Mobile environments.

[Shio] Shiow-yang; Yu-tse Chang: An Active replication scheme for Mobile Data Management.

[Scourias 96] J. Scourias," Overview of GSM: The Global System for Mobile Communications, 1996. URL: citeseer.nj.nec.com/scourias96overview.html.

[Tamori 02] A. D. Joseph, J. A. Tauber, and M. F. Kasshoek. Mobile Computing with the Rover Toolkit.

[Tamori 02] M. Tamori, S. Ishihara, T. Watanabe, and T. Mizuno. A replica distribution method with consideration of the positions of mobile hosts on wireless ad-hoc networks. icdcsw, 0:331, 2002.

[Ter06] D. B. Terry, M. M. Theimer, K. Petersen, A. J. Demers, M. J. Spreitzer, and C. H. Hauser. Managing Update Conflicts in Bayou, a Weakly Connected Replicated Storage System.

[Vai 07] Replication of Location-Dependent Data in Mobile Ad Hoc Networks, Vaidyanathan samany June 2007

[Yu 05] H. Yu, P. Martin, and H. Hassanein. Cluster based replication for large-scale mobile ad-hoc networks. In Proc. 2005 International Conference on Wireless Networks, Communications and Mobile Computing, volume 1, pages 552–557, 2005.

[Phi 94] Techniques de réplication de données pour les systèmes répartis à grande échelle. Philippe Quéinnec 1 avril 1994

[Lam 01] Réplication et Durabilité dans les systèmes répartis. Lambert SONNA MOMO 19 février 2001

[Van 03] Duplication et cohérence configurables dans les applications réparties à base de composants. VANIA MARANGOZOVA Le 17 juin 2003

[Oliv2000] Réplication optimiste pour les applications collaboratives asynchrones. Olivier Dedieu Septembre 2000

[Pib95] E. Pitoura and B. Bhargava, “Maintaining consistency of data in mobile distributed environments”, Proc. IEEE ICDCS’95, pp.404-413, 1995.