République Algérienne Démocratique et Populaire
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

UNIVERSITE KASDI MERBAH OUARGLA

Faculté des Sciences, Technologie et Sciences de la Matière - STSM

Département de Mathématiques & Informatique

N°d'ordre :…….
N°de Série :…….

Mémoire

En vue de l'obtention du diplôme de

Magistère en Informatique

(Option: Réseaux et Systèmes d'Information Multimédia)

Présenté par:

Wassila KORICHI

THEME

Partage de données en environnements mobiles Ad hoc

Dirigé par :
Dr. MOUSSAOUI Samira

Soutenu devant le

Pr. BILAMI Azeddine, Professeur à l'Université de Batna	**Président**
Dr. MOUSSAOUI Samira, Maître de Conférences à l'Université de l'USTHB	**Rapporteur**
Pr. BENMOHAMMED Mohamed, Professeur à l'Université de Constantine	**Examinateur**
Dr. BELATTAR Brahim, Maître de Conférences à l'Université de Batna	**Examinateur**
Dr. KORICHI Ahmed, Maître de Conférences à l'Université de Ouargla	**Examinateur**

Remerciements

Après avoir remercié Dieu,

J'adresse mes remerciements à Monsieur BILAMI Azeddine, Professeur à l'Université de Batna d'avoir accepté la charge de président de jury.

Mes remerciements vont également à Monsieur BENMOHAMMED Mohamed, Professeur à l'Université de Constantine, à Monsieur BELATTAR Brahim, Maître de Conférences à l'Université de Batna, et à Monsieur KORICHI Ahmed, Maître de Conférences à l'Université de Ouargla, qui ont accepté d'être membres de jury.

Je tiens, tout particulièrement et très sincèrement, à remercier Madame MOUSSAOUI Samira, Maître de Conférences au département d'Informatique de l'Université des Sciences et de la Technologie Houari Boumédiene (USTHB), de m'avoir proposé le sujet et de m'avoir encadré. Son suivi, ses encouragements et ses orientations ont été d'un grand réconfort et d'une aide précieuse. Qu'elle trouve dans ces mots l'expression de ma profonde gratitude.

Je remercie très vivement Mr Abdelhakim HERROUZ et Madame Fatima-Zohra Laallam, Maîtres de conférences à l'Université de Ouargla, leurs efforts et leur disponibilité ont assuré le bon déroulement de nos études de post graduation.

Ma gratitude s'adresse également à l'équipe des enseignants de notre promotion PG informatique (RSIM), à l'Université de Ouargla, pour tout le savoir qu'ils nous ont transmis.

Merci également à tous mes collègues de PG informatique (RSIM) pour leur gentillesse, leur compréhension et l'ambiance.

Mes sincères remerciements vont à mes supérieures et collègues de l'Université de Ouargla pour m'avoir donner toutes les commodités nécessaires au déroulement de mon projet dans les meilleures conditions.

Mes remerciements vont également à tous mes amis pour leurs encouragements, compréhensions et aides.

Enfin, je remercie, de tout mon cœur, tous mes proches pour la confiance, le soutien, la patience et l'aide qu'ils m'ont apporté.

Résumé

Les avancées dans le domaine de l'informatique personnelle et des technologies sans fil a permet l'apparition d'un nouveau type de réseaux qui est "les réseaux mobiles en mode Ad hoc". Ces derniers et comme tous les réseaux sans fil ont rendu les utilisateurs d'aujourd'hui mobiles et indépendants de la localisation, en leur offrant de nombreuses opportunités qui ne sont pas toujours prises en compte par les systèmes distribuées existants.

Les réseaux mobiles en mode Ad hoc sont caractérisés par des deconnexions fréquentes ce qui peut engendrer des problèmes de partage de données surtout qu'il s'agit des réseaux (Ad Hoc) où il n'existe aucune infrastructure.

La disponibilité des données est un problème fondamental dans tout système, et notamment dans les réseaux mobiles Ad hoc. Une approche possible pour palier à ce problème est de répliquer l'information au niveau des dispositifs mobiles du réseau. Toute fois, cette solution introduit un nouveau problème qui est celui de l'incohérence des copies de données.

Nous proposons un protocole épidémique pour la convergence des copies de données répliquées dans un réseau mobile en mode Ad hoc. Ce protocole est basé sur la journalisation des écritures. Il permet la réconciliation de plusieurs serveurs en même temps ce qui réduit le nombre de message émis et par conséquence la consommation d'énergie.

Nous avons validé notre proposition par simulation, et montré son impact positif sur la cohérence des données, et l'utilisation efficace des ressources.

Abstract

The personal computing evolution and wireless technology enable the development of a new category of networks: the mobile networks in Ad hoc mode. In effect, with these networks without wire connections, the users are mobiles too and independent of any localization. Many other novel opportunities are not always considered by existing distributed systems.

The mobile Ad hoc networks are characterized by the frequent disconnections which can generate network partition problems. As a result, accessibility to shared data is reduced, especially when there isn't any infrastructure.

The availability of the data is a fundamental problem in any system, and in particular in *the Ad hoc mobile networks*. A possible approach to resolve this problem is the data replication on some mobile devices. But, this solution introduces the data inconsistency problem of the replicas.

We propose an epidemic protocol for the data convergence of replicated data on a mobile Ad hoc network. This protocol is based on the journalizing of the writing operations. It allows the reconciliation of several servers at the same time what reduces the number of transmitted message and the consumption of energy too.

We evaluated our proposal by simulation, and showed its positive impact on the coherence of the data, and the effective use of the resources.

Table des matières

La table des figures :

Liste des tableaux :

Introduction générale

Les terminaux légers équipés de modules de communications sans fil, comme les Smartphones (IPhone, HTC Magic) ou les subnotebooks (EEE PC), sont désormais des objets de consommation courante. Les utilisateurs de ces terminaux sont devenus de plus en plus mobiles. La connectivité supportée par ces équipements offre la possibilité de constituer des réseaux de façon spontanée, réseaux que l'on appelle généralement réseaux mobile Ad hoc.

Un réseau mobile Ad hoc est un ensemble de nœuds mobiles connectés par des liens radio sans fil. Ces environnements présentent l'avantage d'un déploiement rapide et peu coûteux puisqu'ils ne nécessitent l'installation d'aucune infrastructure. Les nœuds se déplacent librement et se connectent dynamiquement les uns aux autres. Par ailleurs, la capacité de stockage des terminaux tels que Smartphone et net book est moindre que celle de serveurs ou de stations de travail, et la durée de vie d'une session de travail est limitée par la batterie. Les réseaux mobiles Ad hoc sont utilisés dans de nombreux domaines (opération de secours, travail collaboratif, campagne scientifique, …). Ils doivent leur existence à une complète et incontournable auto organisation. En effet, leurs caractéristiques, dont principalement le manque d'infrastructure, exigent des nœuds de participer activement aux mécanismes de fonctionnement du réseau.

La mobilité dans un réseau mobile Ad hoc est caractérisée par des phases de connexion et de déconnexion fréquentes. Par conséquent, l'accessibilité aux données sur ce type de réseaux est plus faible que celle dans les réseaux fixe. Une façon de supporter le travail mobile consiste à répliquer les données partagées sur le support mobile avant la déconnexion et à réconcilier les données répliquées à la reconnexions. Cette technique est actuellement reconnue comme étant un moyen efficace pour augmenter la disponibilité et la fiabilité des données [HMJ11] [MOU09]. Elle offre aux utilisateurs de meilleures performances et une plus grande disponibilité des données.

Toute fois, cette solution introduit un nouveau problème qui est celui de l'incohérence des copies de données. En effet, la donnée étant répliquée et les copies manipulées indépendamment, les mises à jour parallèles générées par les utilisateurs font que les contenues des copies finissent par diverger et donc deviennent incohérentes.

Notre travail a porté sur l'étude du problème de l'incohérence des données partagées dans un environnement mobile en mode Ad hoc en répondant à la question: « Comment pouvons nous gérer la cohérence des données dans ce genre de réseau mobile et sans infrastructure en présence de déconnexions imprévisibles ? ».

Une approche appropriée de gestion des mises à jour de copies de données sur les environnements mobiles est une stratégie optimiste. Ceci, bien sûre, lorsque l'application tolère une cohérence non forte des copies. Dans cette approche les réplicas acceptent les mises à jour indépendamment les uns des autres, et par la suite les serveurs entrent dans des phases de réconciliation en vue de converger le contenu des copies locales et distantes.

Contrairement à l'approche pessimiste, l'approche optimiste n'exige pas que les réplicas soient connectés tout le temps. Effectivement, la déconnexion d'un replica dans l'approche optimiste n'influe pas sur le fonctionnement du système. Ainsi, la déconnexion d'un nœud mobile ne pose aucun problème pour l'utilisateur qui peut continuer à travailler avec la copie locale. Cette flexibilité est plus appropriée aux réseaux mobiles Ad hoc ou MANETs ("Mobile Ad hoc NETworks")

Notre travail propose un protocole optimiste de convergence des copies adapté aux caractéristiques des réseaux mobile Ad hoc tels que : la bande passante limitée, l'absence d'infrastructure, la faible capacité en énergie, la sécurité physique limitée et la topologie dynamique.

Notre protocole est basé sur une gestion de cohérence optimiste où la cohérence assurée est faible et produit inévitablement des problèmes de conflits de mises à jour. Selon les systèmes ces conflits sont résolus de différentes manières souvent de manière manuelle en demandant à l'utilisateur de faire des choix. Peu de travaux proposent une gestion automatique de ces conflits. D'autres parts, ces conflits et leur résolution sont étroitement liés au type de l'application traitée. Donc, ce point ne fera pas l'objet de cette étude mais constitue l'une des perspectives qui fera à elle seule l'objet d'un autre sujet de recherche à entamer [TPD 95].

Nous nous intéressons particulièrement à la question : "quand et comment exécuter une mise à jour sur l'ensemble des copies afin de les faire converger vers un même contenu ?"

Le document est organisé en cinq chapitres. Le premier chapitre introduit le contexte dans lequel s'inscrit notre travail. Les environnements mobiles et en particulier les environnements mobiles Ad hoc. Il met en évidence les caractéristiques des communications sans fil, leurs conséquences sur le fonctionnement de ces réseaux, et les problèmes que posent ces environnements. Le deuxième chapitre nous donne quelques définitions et concepts de la réplication et de la gestion de la cohérence. Dans le troisième chapitre, nous étudierons quelques systèmes de réplication pour nous inspirer de ces solutions et maîtriser les problèmes posés. Le quatrième chapitre présent la réalisation de notre protocole basé sur la propagation des mises à jour entre les réplicas d'une donnée adapté aux caractéristiques du réseau mobile Ad hoc. Le cinquième chapitre résume les principales mesures d'évaluation par simulation du protocole et les résultats de comparaisons entre la méthode *MRP* et la méthode *GBRm*. Nous concluons ce document en rappelant notre contribution et en énumérant les perspectives envisageables.

Chapitre 1 : Les Réseaux Mobiles

1.1 Introduction :

Les équipements mobiles deviennent de plus en plus petits et puissants en terme de capacité de traitement et de stockage de données. De plus, ils sont dotés d'une multitude de fonctionnalités qui permettent d'assurer différents types d'applications et de services.

Parallèlement, les réseaux sans-fil actuels connaissent un engouement saisissant. Leur flexibilité d'utilisation a fait que de tels réseaux ont été rapidement adoptés tant par les particuliers que par les entreprises. En effet, les réseaux téléphoniques mobiles se sont développés, passant du GSM à l'UMTS en passant par le GPRS. Tandis que ces réseaux sans- fil fournissent une manière spécifique pour les équipements mobiles d'obtenir des services de réseau, ils nécessitent un temps et un coût potentiellement élevé pour installer leurs infrastructures nécessaires. Il y a, en outre, des situations spécifiques où les besoins de connexions des utilisateurs ne sont pas assurés par ces réseaux avec infrastructures dans une zone géographique donnée. Dans cette situation fournir la connectivité est un réel défi. Pour palier à ces problèmes, de nouvelles extensions des réseaux sans-fil ont été proposées. Elles sont basées sur le fait d'avoir des stations mobiles interconnectées les unes aux autres grâce à une configuration autonome, créant ainsi un réseau mobile Ad hoc flexible et performant.

De nos jours, les réseaux mobiles ad-hoc connaissent une explosion significative des activités dues à leurs facilités de déploiement en réponse aux besoins d'applications ainsi que la disponibilité des périphériques à prix réduit (téléphones portables, ordinateurs portables, etc.) équipés des interfaces sans fil.

Dans ce chapitre, les fondements de base des réseaux mobiles Ad hoc sont étudiés. Il décrit, aussi, les problèmes posés par ce type de réseau MANET à savoir : le routage de données et le déploiement des nouvelles applications distribuées.

1.2 Réseaux sans fil :

Les dispositifs mobiles peuvent bénéficier d'une connexion sans fil vers un autre dispositif mobile ou un site fixe quelconque. Cette connexion est temporaire puisqu'elle dépend de la position géographique d'un nœud mobile. Toutefois, ces connexions temporaires et imprévisibles constituent des réseaux dits sans fil. Sur ce type de réseaux, les utilisateurs peuvent bénéficier des services habituels indépendamment de leur position et de leur mobilité.

Il suffit pour cela d'avoir une connexion. Parfois même sans connexion grâce à la mise en œuvre d'opérations en mode déconnecté. Ces réseaux permettent aussi la mise en réseau de nouveaux dispositifs à moindre coût. Les réseaux sans fil se caractérisent par l'architecture qu'ils adoptent et par la technologie réseau utilisée [BAG95]

1.2.1 Technologies des Réseaux sans fil :

Les différentes technologies des réseaux sans fil sont:

- ✓ Les réseaux WPAN (*Wireless Personal Area Network*): Ces réseaux sont d'une taille de quelques dizaines de mètres. Ce type de réseau relie des périphériques ou des PDA à un ordinateur sans liaison filaire. Ils permettent aussi une liaison sans fil entre deux machines peu distantes. Deux technologies permettent de rendre cette communication sans fil possible :

  - **IrDA** (*Infrared Data Association*): Il s'agit d'une communication à infrarouge. Elle a une faible portée et nécessite un contact visuel et une liaison point à point (entre deux entités bien définies).

  - **Bluetooth :** Connu aussi sous le nom norme IEEE 802.15.1. C'est la principale technologie WPAN. Elle permet un débit théorique de 1Mb/s pour une portée de 30 mètres.

- ✓ Les réseaux WLAN (*Wireless Local Area Network*): Ces réseaux ont une portée de quelques centaines de mètres. Il existe plusieurs technologies pour ce type de réseaux, parmi lesquelles:

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

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

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

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

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

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

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

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

1.2.2 Architecture des réseaux sans fil :

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

Les réseaux mobiles avec infrastructure ou réseaux cellulaires sont composés de deux ensembles d'entités primordiales, interagissant les unes avec les autres : Les sites fixes d'un réseau de communication filaire, et les sites mobiles [IMB94]. Certains sites fixes, appelés stations support mobile (Mobile Support Station) ou station de base (SB) sont munis d'une interface de communication sans fil pour la communication directe avec les sites ou unités mobiles (UM), localisés dans une zone géographique limitée, appelée cellule (voir figure1.1).

A chaque station de base correspond une cellule à partir de laquelle des unités mobiles peuvent émettre et recevoir des messages. Alors que les sites fixes sont interconnectés entre eux à travers un réseau de communication filaire, généralement fiable et d'un débit élevé. Les liaisons sans fil ont une bande passante limitée qui réduit sévèrement le volume des informations échangées [DNF92].

Dans ce modèle, une unité mobile ne peut être, à un instant donné, directement connectée qu'à une station de base. Elle peut communiquer avec les autres sites à travers la station à laquelle elle est directement rattachée. L'autonomie réduite de sa source d'énergie, lui occasionne de fréquentes déconnexions du réseau. Sa reconnexion peut alors se faire dans un environnement nouveau voire dans une nouvelle localisation dirigée par une nouvelle station de base.

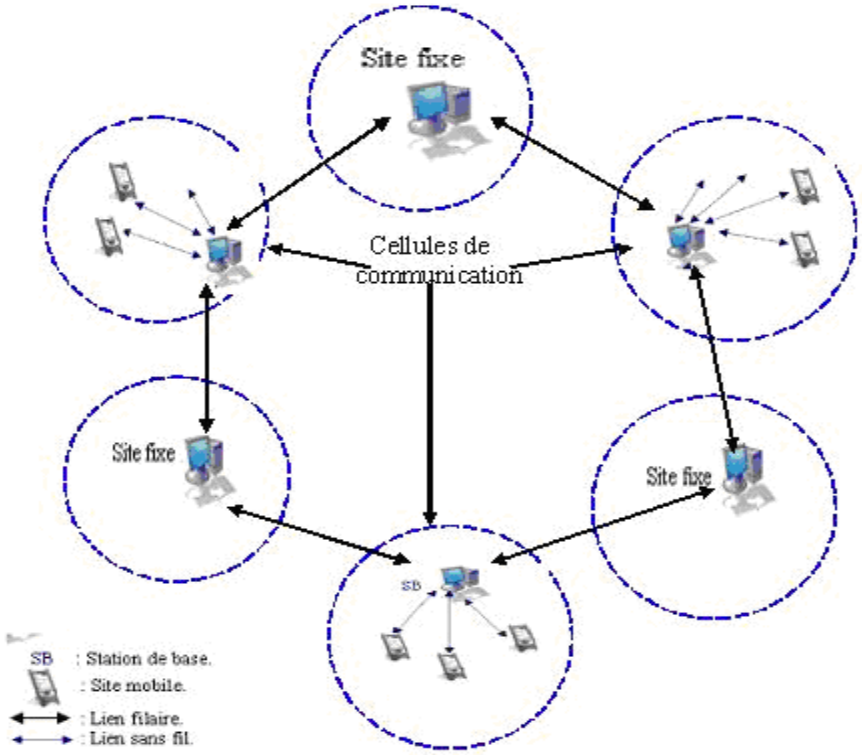


Figure 1.1 : Réseau Cellulaire

Les infrastructures cellulaires offrent une certaine simplicité d'emploi, qui a permis à des réseaux tel que le réseau GSM de la téléphonie mobile de fonctionner correctement. Mais il reste que la mobilité qu'ils offrent est limitée aux zones géographiques possédant une connexion avec les stations de base, une entité mobile ne pourra donc pas faire partie du réseau n'importe où Les réseaux mobiles avec infrastructure n'offrent aux sites mobiles qu'une liberté modérée.

- **Communication cellulaire :**

Initialement, le nombre de cellules existant dans un système cellulaire permet de satisfaire ses utilisateurs. Mais, si par la suite le nombre d'utilisateurs augmente, la bande de fréquence affectée à chaque cellule risque de ne pas suffire. Pour palier à ce problème, il faut accroître la capacité du système, en divisant les cellules en un nombre de micro cellules où chaque micro cellule adjacente a une fréquence différente**.** Chaque micro cellule issue de la division porte sur un champ d'action plus réduit et dispose d'une station de base, cela implique plus de relais d'une plus faible capacité.

- **Transfert intercellulaire (Handoff):**

En se déplaçant, une unité mobile risque de s'éloigner de la station de base de la cellule où elle se trouve ; dans ce cas, pour poursuivre sa communication, l'unité doit faire appel à une nouvelle station de base. Pour faire appel à la station de base la plus

appropriée, l'unité mobile se sert d'un signal transmis par ces dernières. Le signal réceptionné le plus intensément par les unités est celui qui correspond à la station de base la plus proche, et donc la plus appropriée.

1.3 Les réseaux mobiles sans infrastructure *(Ad Hoc) :*

Les réseaux mobiles ad-hoc [CEP00][CBS04] sont apparus dans les années soixante-dix. Ils correspondent à une catégorie de réseaux mobiles qui peuvent fonctionner sans infrastructure fixe en établissant des communications directes ou multi-sauts [HMM02] entre les équipements. Nous pouvons caractériser de manière générale un réseau mobile ad-hoc par la définition suivante :

> *Un réseau mobile ad-hoc (MANET, Mobile Ad-Hoc Network) est un réseau auto-organisé formé spontanément à partir d'un ensemble d'entités mobiles communicantes (ordinateurs portables, téléphones mobiles, assistants électroniques) sans nécessiter d'infrastructure fixe préexistant*e.

Les réseaux mobiles sans infrastructure utilisent, pour matérialiser leur communication, des technologies de transmission sans fils telles que les ondes radio ou l'infrarouge, mais jamais de câble, ce qui limite le coût de tels réseaux.

Les réseaux mobiles Ad hoc peuvent être déployés facilement et rapidement en permettant des échanges directs entre stations mobiles. Ainsi, le fonctionnement du réseau repose sur les stations mobiles elles-mêmes : celles-ci interviennent à la fois comme terminaux pour communiquer avec les autres usagers et comme routeurs afin de relayer le trafic pour le compte d'autres utilisateurs. En particulier, lorsque deux stations ne peuvent communiquer directement parce qu'elles ne se trouvent pas dans le même voisinage, une communication multi-sauts est initialisée en utilisant les nœuds intermédiaires : les paquets sont transmis de station en station avant d'atteindre la destination souhaitée.

En effet, chaque nœud a sa propre portée de communication et tous les nœuds se trouvant dans ce domaine peuvent échanger directement des informations avec lui. Lorsque deux nœuds distants veulent communiquer de l'information, ils doivent d'abord passer par des nœuds intermédiaires avant que l'information ne soit réceptionnée (les nœuds intermédiaires jouent le rôle de relais pour le transport de l'information). Si un nœud se trouve isolé, sans aucun autre nœud à sa portée, il est déconnecté du réseau.

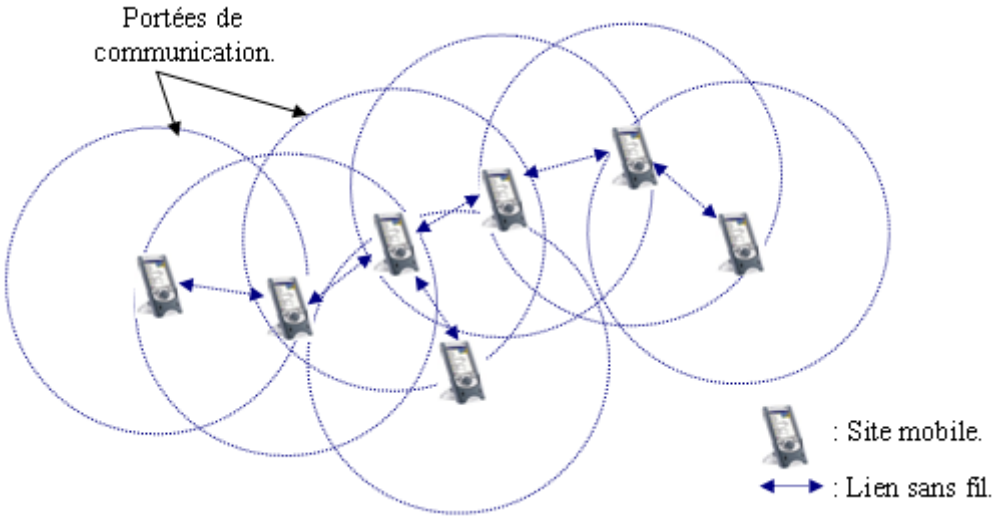


Figure 1.2 : Réseaux mobiles Ad hoc

1.4 La communication dans les réseaux mobiles Ad hoc :

Comme nous l'avons vu précédemment, les réseaux mobiles Ad hoc sont des réseaux sans infrastructure. Ils ne disposent d'aucune administration centralisée et chaque unité mobile a sa propre interface de communication sans fils. Dans ce type d'environnements, la question qui se pose est : comment se fait l'acheminement des paquets ?

Si un nœud désire émettre un paquet, ce paquet est réceptionné par tous les autres nœuds de même portés mais, seul le nœud destinataire, parvient à le déchiffrer. Les autres nœuds sont victimes d'interférences ou de collisions.

Dans un tel environnement mobile, la liaison bidirectionnelle entre deux nœuds n'est pas toujours assurée. Dans le cas le plus simple, par exemple, deux nœuds qui se trouvent dans la portée de communication l'un de l'autre peuvent trouver des difficultés à communiquer. Si le signal provenant d'un des nœuds est atténué par un obstacle ou autre, la communication directe entre les deux nœuds devient unidirectionnelle, et le lien entre les nœuds asymétriques.

1.5 Modélisation d'un réseau mobile Ad hoc :

Un réseau mobile Ad hoc peut être modélisé par un graphe [LEM00] Gt = (Vt,Et) où : Vt représente l'ensemble des nœuds (i.e. les nœuds mobiles) du réseau et Et modélise l'ensemble des connexions qui existent entre ces nœuds. Si e = (U, V) ∈ Et, cela veut dire que les nœuds U et V sont en mesure de communiquer directement à l'instant t.

La figure suivante représente un réseau mobile Ad hoc de 7 unités mobiles sous forme d'un graphe :

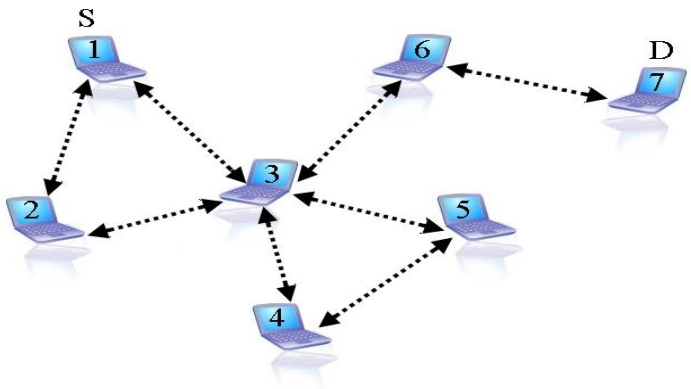


Figure 1.3 : la modélisation d'un réseau mobile Ad hoc.

1.6 Topologie :

La topologie du réseau peut changer à tout moment, elle est donc dynamique et imprévisible ce qui fait que la déconnexion des unités soit très fréquente [LEM00].

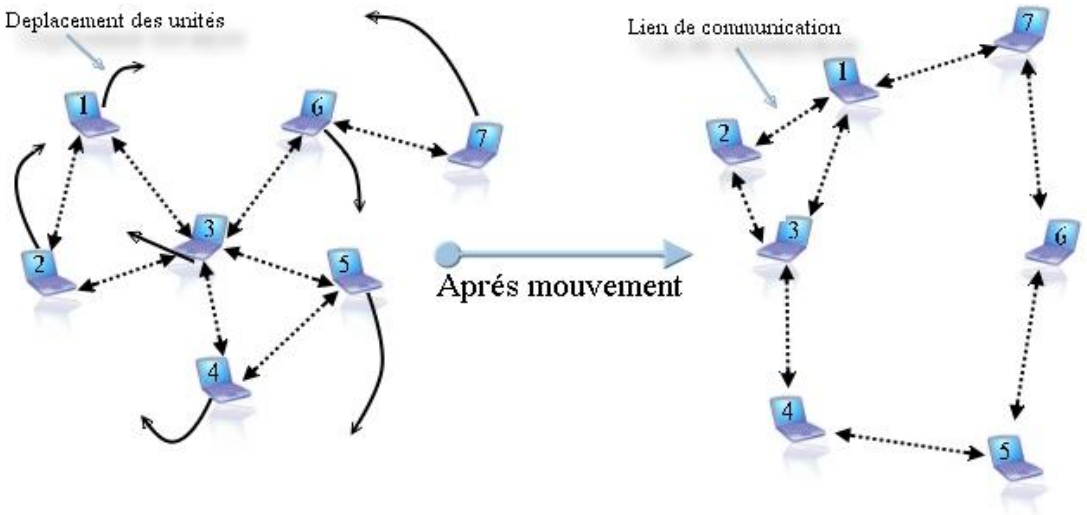


Figure 1.4 : Le changement de la topologie des réseaux mobiles Ad hoc

1.7 Les métriques dans les MANETs :

La plupart des applications informatiques requièrent généralement une configuration matérielle minimale pour bien fonctionner. Les applications déployées dans un réseau spécifient en plus les contraintes que le réseau doit satisfaire pour que leur exécution corresponde à un niveau de qualité bien défini. Du point de vue du réseau, ces critères se traduisent en général en termes de capacité, de disponibilité, de latence et de fiabilité.

Selon le type de réseau, ces concepts abstraits sont traduits en grandeurs mesurables identifiées sous le terme général de métrique.

Le tableau 1.1 suivant donne un aperçu des différentes métriques utilisées pour définir des contraintes de qualité de service liées aux applications courantes [CHC05].

Type d'application	Détails	Qualité de service requise
Diffusion vidéo (télévision Internet, vidéo à la demande)	MPEG2, MPEG4	Débit : 64 kbit/s à 2 Mbit/s Délai, gigue : compensables par la mémoire tampon
Diffusion audio (radio Internet)	MP3, AAC, OggVorbis	Débit : de 32 kbit/s à 256 kbit/s Délai, gigue : compensables par la mémoire tampon
Téléphonie (VolP)	PCM-MIC, G.711	Débit : 6,4 kbit/s à 64 kbit/s Latence : 150 ms à 300 ms Gigue : 0 à 50 ms Pertes : < 0,1 %
Vidéoconférence	Architecture H.323 Résolution vidéo (H.263) : de SUB-QCIF (128 × 96) à 16CIF (1 408 × 1 152)	Débit : 64 kbit/s à 1 920 kbit/s Délai : de l'ordre de 150 ms à 300 ms Gigue : 0 ms à 50 ms Pertes : < 0,1 %
Session interactive	SSH, telnet, VNC, T120	Débits variables Délai de l'ordre de 600 ms Pertes nulles

Tableau 1.1 : Quelques applications multimédias et leurs contraintes de QoS

Les stations d'un réseau mobile Ad hoc concentrent toutes les fonctionnalités liées au routage. De ce fait, toutes les métriques généralement prises en compte par les routeurs pour assurer le routage doivent pouvoir être prises en compte par toute station dans un MANET. Aux métriques qui caractérisent les liaisons sans-fil en général, peuvent s'ajouter des métriques liées à la spécificité des réseaux mobiles Ad hoc. En effet, les stations d'un MANET sont en général caractérisées par des ressources limitées. L'état interne de la station, notamment son autonomie énergétique peut influer directement sur son aptitude à participer au routage. La mobilité des nœuds induit également la nécessité de considérer des métriques telles que la stabilité d'une liaison ou sa durée de vie

1.7.1 Les métriques caractérisant les stations :

Certaines métriques sont utilisées soit pour décrire l'état interne d'une station, soit pour caractériser cette dernière vis-à-vis de ses voisins ou de l'ensemble du réseau :

• **La charge résiduelle de sa batterie** ou **l'autonomie énergétique** d'une station peut être associée à une métrique permettant de l'évaluer par rapport aux autres stations.

Dans le cadre du protocole OLSR par exemple, une caractéristique liée aux stations et identifiée sous le terme de WILLINGNESS [CTJ03] traduit l'aptitude d'une station à retransmettre du trafic en provenance d'autres stations. Dans [CTJ03], il est proposé de corréler le Willingness avec la charge de la batterie. Ainsi, dans un MANET où les stations utilisent OLSR comme protocole de routage, les stations disposant d'une autonomie énergétique plus importante seront plus souvent choisies pour retransmettre les messages diffusés dans le réseau. Le niveau d'énergie des stations est un facteur tellement important dans les MANET que plusieurs travaux y sont consacrés [SWR98][TCK01] [MAB02] [DDB06][PPM07]. En effet, l'épuisement de la batterie d'une station entraîne sa déconnexion du réseau, ce qui peut engendrer la disparition de plusieurs liaisons. Les conséquences peuvent aller jusqu'à la déconnexion d'une partie du réseau si la station concernée était le seul pont qui la reliait au reste du MANET. La **durée de vie** d'une station est également une métrique parfois associée à l'autonomie énergétique de cette station.

- **La capacité disponible du buffer de transmission d'une station** [SMS03] est une métrique liée au trafic généré ou retransmis par une station. Elle permet par exemple de déterminer les stations les moins chargées pour la retransmission des paquets.

- **La mobilité ou la stabilité d'une station par rapport à son voisinage** [NNB01] a également inspiré des métriques permettant de caractériser les stations d'un MANET.

- **La capacité d'adaptation d'une station face à la mobilité ou à la densité dans son voisinage** peut également être associée à des métriques susceptibles de jouer un rôle dans le choix des nœuds retransmetteurs de paquets [YPD07].

1.7.2 Les métriques caractérisant les liaisons :

La grande majorité des métriques étudiées dans les MANET caractérisent plutôt les liaisons entre les stations :

- **Le débit minimum disponible** sur une liaison ou sur une route est l'une des métriques ayant fait l'objet d'un très grand nombre d'études [GKL03][BHA05] [LCL99] [NDA09]. Il correspond à la capacité d'une route entre une station source et une station destination.

Bien que la règle permettant de calculer cette capacité sur une route constituée de plusieurs liaisons soit assez claire, la valeur obtenue dépend des valeurs des capacités de chaque liaison. Le problème est que cette valeur dépend de tellement de facteurs dans les réseaux sans fil en général, et dans les MANET en particulier, qu'aucune des méthodes d'estimation proposées jusqu'à présent ne fait l'unanimité.

• **Le délai de bout-en-bout** est le temps qui s'écoule entre le moment où un paquet est enregistré pour un envoi par la source et le moment où il est intégralement reçu par la destination. Cette métrique est également l'une des plus étudiées et elle est utilisée notamment dans [CSN99].

• **La gigue** est la variation entre la valeur maximale observée sur le délai de bout-en bout et le délai de transmission minimum observé [BAE05][WMK05][BML04].

• **La robustesse d'une liaison** qui est généralement associée à la valeur estimée ou effectivement mesurée **du taux de perte** de paquets sur une liaison [BKS03][AAZ06].

• **La stabilité d'une liaison** ou d'une route est également une métrique intéressante. On peut aussi bien la corréler à la stabilité des stations impliquées qu'à leur durée de vie. Rubin et Liu [RIL03] proposent un modèle pour la qualité de service dans les MANET qui utilise la durée de vie des liaisons comme métrique.

D'autres contraintes, concernent l'accès au support, peuvent être rajouté dans le cas d'un réseau mobile ad hoc basé sur un protocole d'accès avec détection de porteuse comme le Wi-Fi, à s'avoir : le problème de la station cachée et celui de la station exposée.

➢ **Problème de la station cachée :** Il s'arrive que deux stations A et C se trouvent hors de portées respectives mais leurs zones de transmission ne sont pas disjointes (Figure.1.5). Quand ces deux stations A et C envoient des informations à la station B simultanément ceci provoque, bien évidemment, une collision. Ce problème est appelé problème de la station cachée.

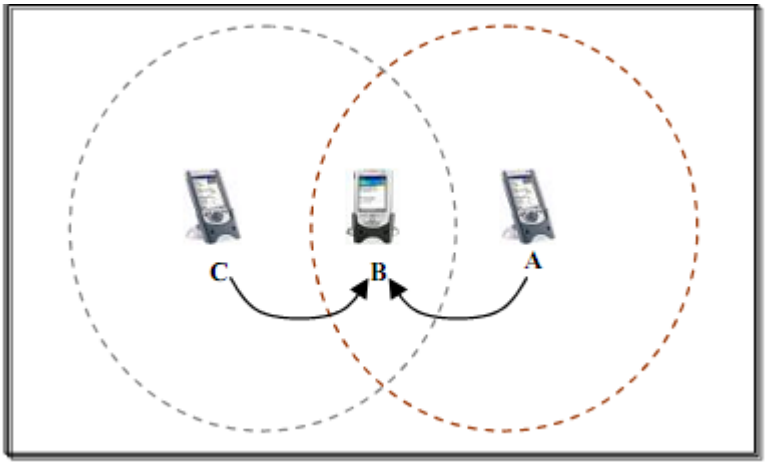


Figure 1.5 : Problème de la station cachée

➢ **Problème de la station exposée** : La figure suivante (Figure. 1.5) présente un scénario typique de ce problème. En effet, supposons que les deux stations A et C peuvent entendre les transmissions de B, mais que la station A n'entend pas C (et vice-versa). Supposons aussi que B est en train d'envoyer des données vers A et que, au même moment, C veut communiquer avec D. Selon la technique CSMA, la station C va commencer par déterminer si le support est libre. A cause de la communication entre B et A, C trouve le support occupé et il retarde son envoie bien que celui-ci n'aurait pas causé de collision.

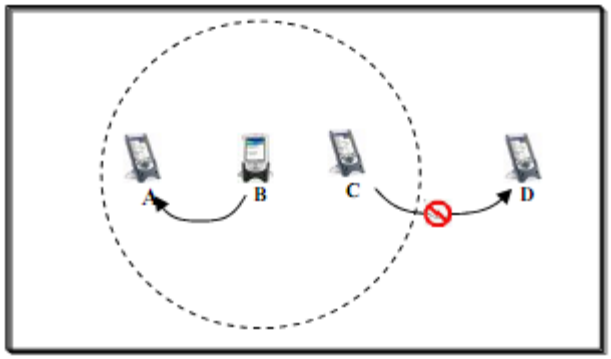


Figure 1.6 : Problème de la station exposée

1.8 Les applications des réseaux mobiles Ad hoc :

Les réseaux mobiles Ad hoc sont utilisés dans toute application où le déploiement d'une infrastructure réseau filaire est trop contraignant, soit parce que c'est difficile à mettre en place, soit parce que la durée d'installation du réseau ne justifie pas de câblage Parmi ces applications on trouve :

- Applications militaires**:** On peut bénéficier des avantages des réseaux mobile Ad hoc dans les applications militaires, car ces réseaux leurs permettent de se déplacer et de communiquer entre eux dans les champs de bataille.

- Contrôle d'environnement: Des petits véhicules équipés de caméras et des détecteurs peuvent être utilisés dans une région déterminée. Afin de collecter un ensemble d'information et de l'envoyer à travers un réseau mobile Ad hoc à une station qui traite ces informations. On peut avec cette méthode, par exemple, prévoir la pollution de l'eau [ERC99].

- Opérations de secours : les opérations de secours nécessitent toujours l'échange d'information, et si par exemple le sauvetage se déroule dans une zone où l'infrastructure de base pour la mise en place d'un réseau est endommagée ou

n'existe pas, dans ce cas, une installation d'un réseau mobile Ad hoc peut résoudre le problème.

- Utilisation privée : Les réseaux mobiles Ad hoc peuvent être utilisés dans les Home Networks où les différents équipements échangent des informations (son, vidéo, alarme). Une des applications qui est dans ce contexte est : un réseau de robots dans une maison qui font plusieurs travaux (nettoyage, assurance de la sécurité) [MFP00].

- Les conférences : Par exemple un réseau mobile Ad hoc qui relie les différentes unités portables (lap top, PDA, notebook) peut être utilisé pour propager et partager les informations parmi les participants dans une conférence [MFP00]

Exemple : réseau véhiculaire.

Les VANets, ou Vehicular Ad hoc Networks, sont des réseaux constitués de véhicules. Le projet StreetSmart [SDA07] propose de constituer les voitures sur une autoroute en VANet, afin de véhiculer des informations sur l'état du trafic. Une forte densité de véhicules avançant à faible vitesse permet de détecter une congestion. Cette information remonte ensuite le flot de véhicules et peut être utilisée par les systèmes de guidage GPS pour optimiser les trajets.

Le nombre de terminaux impliqués dans ces communications est élevé, puisque le réseau couvre l'ensemble du segment d'autoroute. Cependant, l'information ne circule que dans un sens, à l'inverse du flot de voitures, et les communications ne se font pas entre entités spécifiques. Il n'y a donc pas réellement de routage, et on doit simplement connaître les positions relatives des terminaux à un saut afin de décider comment les messages doivent être diffusés. Enfin, aucune information n'a à être stockée après avoir été transmise.

Il n'y a pas de problème d'autonomie ou de stockage. La sécurité n'est pas abordée.

Exemple : réseau de capteurs mobiles, application scientifique.

Dans le projet ZebraNet [POW02], une équipe de chercheurs veut étudier le comportement d'un troupeau de zèbres. Chaque zèbre est équipé d'un collier capteur GPS faisant régulièrement un enregistrement de position.

Périodiquement, un des chercheurs passe à proximité du troupeau afin de récupérer les informations. C'est le seul moment dans l'absolu où les terminaux ont à communiquer, ce qui utilise leur batterie, mais des communications sont tout de même maintenues en dehors des périodes où l'information est recueillie.

La création de nouvelles données est périodique. Si on connaît la fréquence de passage du chercheur, on peut donc prédire les capacités de stockage nécessaires. Cependant, la capacité de stockage d'un capteur étant faible, les données recueillies sont agrégées afin de ne pas occuper tout l'espace mémoire. La durée de vie d'un capteur est par ailleurs limitée

par ses batteries. Quand les batteries d'un collier sont vides, il faut capturer à nouveau l'animal portant ce collier pour les remplacer ou changer le capteur. Les données sont donc répliquées sur différents capteurs, afin qu'aucune ne soit perdue.

Le nombre de terminaux ici est de l'ordre de la centaine. Les terminaux se déplacent en groupe, et les communications sont faites de manière opportuniste pour sauvegarder les informations. L'autonomie des terminaux étant faible, ce système cherche donc à la maximiser.

Exemple : réseau à vitesse humaine, jeu collaboratif

Dans le projet Transhumance [AAI08], un jeu de piste est organisé sur la Butte-aux - Cailles. Les utilisateurs sont organisés en groupes, chaque membre ayant un PDA. Au début d'une session de jeu, un questionnaire est distribué sur les terminaux des joueurs. La première équipe à trouver tous les éléments et revenir à l'arbitre a gagné. A défaut, l'équipe ayant trouvé le plus d'éléments à la fin du temps imparti gagne.

La durée d'une session de jeu pour un groupe est limitée par la batterie de ses terminaux. Différents utilitaires sont proposés aux joueurs, comme un chat, un vote, ou un partage de fichier. Ici, les joueurs sont organisés en groupes collaborant. Même s'ils peuvent se séparer, par exemple pour que chacun aille chercher un indice individuellement, ils se retrouvent par la suite et leurs feuilles de réponses doivent être synchronisées.

Les communications peuvent être chiffrées, pour que les membres des équipes adverses ne puissent pas connaître les réponses, cependant sécuriser les données n'est pas ici critique.

1.9 Protocoles de routage mobile Ad hoc :

La standardisation des protocoles mobiles Ad hoc s'opère essentiellement au niveau de la couche réseau IP afin d'offrir une meilleure interopérabilité entre les réseaux et de permettre des technologies hétérogènes dans les couches basses. Une multitude de protocoles de routage a été développée spécifiquement pour les réseaux mobiles Ad hoc ces dernières années. L'absence de coordination centrale et de stations de base rend la tâche de routage plus complexe que dans les réseaux fixes. De plus, le plan de routage doit être capable de s'adapter à la mobilité de ces réseaux tout en consommant le moins de ressources possibles car les capacités des terminaux sont généralement fortement contraintes. Le groupe de travail MANET de l'IETF [WEB04] a en charge la standardisation des protocoles de routage à destination des réseaux mobiles Ad hoc. Ces protocoles peuvent être classifiés en fonction du mécanisme de mise à jour des informations de routage : routage réactif et routage proactif [LEM00].

1.9.1 Routage réactif **:**

Le protocole établit une route uniquement lorsqu'elle est demandée par un nœud mobile, en initialisant un mécanisme de découverte. Dans une approche réactive, les nœuds mobiles ne maintiennent pratiquement pas d'informations sur la topologie du réseau. Le nœud source émet une requête de route qui est diffusée dans le réseau mobiles Ad hoc jusqu'au nœud destination. Les nœuds intermédiaires sont découverts et mémorisés durant cette phase de diffusion. Lorsque le nœud destination reçoit la requête, il utilise le chemin inverse pour contacter le nœud source et lui transmettre les informations de routage. Un exemple typique de protocole réactif est le protocole AODV (*Ad-hoc On-Demand DistanceVector*) spécifié par la RFC 3561.

1.9.2 Routage proactif :

L'approche proactive consiste, à l'inverse, à ce que chaque nœud du réseau maintienne une table de routage. Les mises à jour de la table sont obtenues par échange périodique d'informations de topologie entre les nœuds. Un exemple typique de protocole proactif est le protocole OLSR (*Optimized Link State Routing Protocol*) spécifié dans la RFC 3626. Ce protocole est une optimisation des protocoles à état de liens adaptés à la nature des réseaux mobiles Ad hoc. Chaque nœud réalise deux opérations principales :

Il détermine la liste des voisins directs en évaluant le voisinage par émission périodique de messages HELLO et il échange les informations de topologie avec les autres nœuds en diffusant des messages TC de contrôle de topologie. L'heuristique du protocole OLSR permet de réduire la charge de trafic durant la phase de diffusion en ne sélectionnant qu'un sous-ensemble de nœuds appelés relais multi-points pour la retransmission des messages de contrôle.

Ces deux approches de routage ne présentent pas les mêmes propriétés et le choix de l'une ou l'autre doit être réalisé en fonction du type d'applications envisagé. Un protocole réactif permet un meilleur passage à l'échelle car il réduit de manière significative la charge de trafic dans le réseau puisque les informations de topologie ne sont pas diffusées périodiquement. Cependant, une approche proactive offre un meilleur temps de latence pour établir une route puisqu'elle ne nécessite aucun mécanisme de découverte au préalable. Des protocoles de **routage hybride** tels que le protocole ZRP (*Zone Routing Protocol*) combinent proactivité et réactivité. Ils permettent d'offrir un compromis entre charge de trafic et temps de latence. La topologie des nœuds qui se trouvent dans une zone proche est maintenue dans une table de routage à travers une approche proactive. *A contrario*, les nœuds situés en dehors de cette zone de voisinage (à plus d'un certain nombre de sauts) sont atteignables par découverte de route grâce à une approche réactive.

1.10 Avantages et inconvénients des réseaux mobiles Ad hoc :

Les réseaux mobiles Ad hoc sont utiles quand la connexion filaire n'est pas disponible, par exemple lors d'une opération militaire, et plus généralement quand le déploiement rapide d'un réseau est nécessaire. Dans ce cas, les nœuds communiquent en acheminant les messages par routage « multi-sauts ».

Indépendamment du fait de disposer ou non d'une infrastructure, le mode Ad hoc multi-sauts a de nombreux avantages en comparaison des modes de communication avec stations de base :

- ✓ **Pas de câblage** : L'une des caractéristiques des réseaux mobiles Ad hoc est l'absence d'un câblage, et ce en éliminant toutes les connexions filaires qui sont remplacées par des connexions radio ou infrarouge.

- ✓ **Déploiement facile** : Contrairement aux réseaux cellulaires qui requièrent un important effort de planification pour leur déploiement, les réseaux mobiles Ad hoc peuvent être déployés facilement et rapidement en permettant des échanges directs entre stations mobiles.

- ✓ **Permet la mobilité** : Comme l'indique leur nom, les nœuds peuvent se déplacer librement à condition de ne pas s'éloigner trop les uns des autres pour garder la connectivité du réseau.

- ✓ **Extensible** : Un réseau mobile Ad hoc peut s'étendre dans sa taille et ceci d'une manière facile. En effet, l'ajout d'un nouveau nœud nécessite quelques configurations pour qu'il fonctionne en sein du réseau.

- ✓ **Coût** : Il n'y a aucun coût d'installation des stations de bases ou de câblage. La simple présence des hôtes possédants des interfaces de communication radio forme le réseau.

Néanmoins, les réseaux mobiles Ad hoc présentent des inconvénients peuvent être résumés par les points suivants :

- × **Topologie non prédictible** : Le changement rapide de leur topologie dû au aux déplacements des nœuds rendent leur étude très difficile.

- × **Capacités limitées (puissance de calcul, mémoire, énergie)** : Dans un tel réseau, il faut trouver un équilibre entre la connexité du réseau et la consommation énergétique. En effet, il faut que la portée d'émission des nœuds

soit suffisante pour assurer la connexité du réseau. Mais plus on accroît la portée des mobiles, plus les communications demandent de l'énergie.

- **Taux d'erreur important** : Ce taux d'erreur dépend des collisions qui augmentent avec le nombre de nœuds partagent le médium radio.

- **Sécurité** : Les réseaux sans fil sont par nature plus sensibles aux problèmes de sécurité. Pour les réseaux mobiles Ad hoc, le principal problème ne se situe pas tant au niveau du support physique mais principalement dans le fait que tous les nœuds sont équivalents et potentiellement nécessaires au fonctionnement du réseau. Les possibilités de s'insérer dans le réseau sont plus grandes, la détection d'une intrusion ou d'un déni de service plus délicate.

1.11 Conclusion :

Dans ce premier chapitre, nous avons fait une brève connaissance avec les réseaux sans fil. En particulier, les réseaux mobiles en mode Ad hoc sont d'un grand intérêt. L'étude sur les réseaux mobiles Ad hoc nous a permis de conclure que ces derniers sont rapides à mettre en œuvre, leur mobilité est indépendante de toute infrastructure fixe et qu' ils peuvent couvrir une large. Ces caractéristiques reposent sur des protocoles de routage spécifiques et sur des transmissions radio.

Nous avons vu dans ce chapitre dans quelles circonstances un réseau mobile Ad hoc peut s'avérer utile: les premières applications étaient liées aux situations d'urgence, par exemple, suite à une catastrophe, ou dans une situation militaire. Mais, on peut aussi les utiliser pour un usage civil quand les infrastructures sont inexistantes, ou compliquées d'accès.

Nous avons, ensuite, vu trois projets de recherche utilisant les MANets dans des contextes très distincts.

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

Chapitre 2 : Partage et réplication de données

2.1 Introduction :

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

La gestion des données sur les réseaux mobiles Ad hoc est un axe de recherche en plein essor vu la demande croissante en accessibilité et en disponibilité de l'information. Le défi consiste à fournir aux nœuds mobiles une continuité de service et d'accès malgré les contraintes de ces environnements.

Une approche possible pour ce problème est de copier une partie ou la totalité des données sur le support mobile. C'est ce qu'on appelle la réplication, cette dernière offre aux utilisateurs de meilleures performances et une plus grande accessibilité aux données.

Toutes fois la réplication est confrontée a un problème de divergence de copies en effet : durant la période de déconnexions les différentes copies (les réplicas) évoluent en parallèles et des écritures concurrentes peuvent alors être effectuées ce qui fait diverger les réplicas d'où la nécessité d'établir un mécanisme pour maintenir la cohérence entre les copies.

Dans notre étude nous nous intéressons principalement aux applications qui acceptent une réplication optimiste car cette dernière consiste à autoriser la manipulation concurrente et indépendante de la donnée répliquée même si le nœud n'est pas connecté au réseau. Par exemple dans le système de fichiers distribué Coda [JKS91], qui supporte les opérations même quand un pair est déconnecté du serveur, ou le système de version Subversion [MBS08].

2.2 La réplication :

2.2.1 Définition :

La réplication ou duplication, comme le nom l'indique, consiste à dupliquer un objet en plusieurs images ou bien copies de telle manière que cet ensemble de copies représente un seul objet logique qui est l'objet initial avant duplication. En d'autre terme, 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. La création de copies consiste à reproduire la structure et l'état des entités dupliquées. La copie d'une base de données est une autre base de données de même contenu. La copie d'un programme est un autre programme qui exécute le même code.

2.2.2 Objectif de la réplication :

Les objectifs visés peuvent être résumés en :

Disponibilité : les ressources sont disponibles et le système est prêt à fonctionner et apte à accomplir sa fonction de manière fiable. Si une donnée se trouve localisée sur plusieurs sites, il n'est pas forcément nécessaire de s'adresser à un site distant afin de la consulter (localité des accès en consultation).

Fiabilité : aptitude à accomplir sa fonction sans défaillance dans des conditions données pour une durée déterminée. En répliquant les composants, le système sera capable de continuer à rendre un service malgré la panne d'un ou de plusieurs de ses composants.

Maintenabilité: possibilité d'être maintenu ou rétabli en un temps donné dans un état d'aptitude à accomplir sa fonction.

2.2.3 Intérêts et objectifs de la réplication de données :

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

- **Accessibilité**:
Dans le cas où une donnée risque d'être non disponible ou inatteignable par une partie des nœuds du réseau, la présence de plusieurs copies de la donnée est le meilleur moyen de permettre aux nœuds qui la recherche de la découvrir. La disponibilité d'une donnée pour un nœud est exprimée par le fait qu'un nœud puisse atteindre au moins

une quelconque des multiples copies. Le perpétuel mouvement des nœuds, les déconnexions et le partitionnement font de la réplication un allié essentiel des environnements mobiles Ad hoc.

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

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

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

2.2.4 Inconvénients de la réplication :

De part sa définition, un réseau mobile Ad hoc requiert une approche de réplication complètement différente de celle déjà développée pour des réseaux plus conventionnels. En effet, sur ces systèmes, certains inconvénients des méthodes connues sont retrouvés avec une ampleur souvent plus importante. Quoique la réplication soit avant tout un outil pour résoudre des problèmes divers sur les réseaux mobiles Ad hoc, elle présente les complications suivantes [PAP 04] :

- Divergence des copies:
La réplication d'une donnée sur plusieurs machines peut faire évoluer les copies d'une manière parallèle et indépendante. Ces copies finissent par avoir des contenus qui sont dans des états différents et incohérents. L'intégration d'un algorithme de convergence est alors indispensable [XAV 98][QIL 96][EPB 95][WIE 90**].**

- Surcoût en communication:
Les opérations de re-localisation de copies de données et l'évaluation de certains paramètres (connectivité, degré de réplication, …) de l'algorithme de réplication

produisent un surcoût en trafic, en traitement, et en consommation énergétique. Sur un réseau mobile Ad hoc, cet "overload" peut vite devenir important car l'algorithme de réplication doit s'adapter en permanence aux changements dynamiques de la topologie.
Cet effort d'adaptation peut entraîner un surcoût considérable. La conception de toute solution de réplication doit veiller à modérer ce surcoût.

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

2.3 La cohérence :

2.3.1 Notions de cohérence:

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

La relation de cohérence est assurée par synchronisation entre copies. Considérant que les copies d'une entité se partagent son état, la synchronisation définit des règles d'accès à cet état.

Elle peut être vue comme une définition indirecte de la cohérence où cette dernière n'est plus définie en termes de différences entre copies mais en termes de règles de contrôle de ces différences [JUD98] [MAR00] [WNT98].

2.3.2 La classification des modèles de cohérence :

2.3.2.1. Type de cohérence :

Les protocoles de gestion de cohérence peuvent être classés en deux grandes catégories selon la vitesse de mise à jour [OLD 00] [JUD 98]:

- **Cohérence forte (Strong consistency) :**

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

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

Ce type de cohérence ne convient pas :

- Lorsque l'efficacité est un critère prépondérant.
- Lorsque la disponibilité des objets doit être très grande.
- Lorsque les temps de communication sont importants.
- Lorsque la sémantique des objets n'est pas forcément très stricte et que l'on accepte de laisser des copies diverger c'est ce qui est toléré en cas de la cohérence faible qui sera étudiée par la suite.

- **Cohérence faible (Weak consistency):**

En cas de cohérence faible [OLD00], la lecture d'une copie locale ne reflète pas forcément toutes les écritures antérieures. Cependant, elle garantit que ces dernières soient toutes répercutées sur la copie au bout d'un temps fini.

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

La cohérence faible, consiste à laisser évoluer chaque replica de façon autonome, puis de faire des contrôles a posteriori. Chaque replica informe les autres copies de la même donnée des modifications qu'il a effectuées.

Le contrôle a posteriori permet la convergence ou la resynchronisation des copies après une phase dite de réconciliation. Une réconciliation est un protocole exécuté pour faire converger les copies vers un même contenu. Souvent ce protocole est périodique. Si la réconciliation n'est pas possible dans le cas ou des écritures conflictuelles ont été produites sur plusieurs copies. Par exemple, deux enseignants ont réservé la même séance pour deux modules différents. Le premier enseignant effectue la mise à jour m1 sur la séance S1 et le deuxième effectue la mise à jour m2 sur la séance S1. Ces mises à jour génèrent ce qu'on appelle un *conflit* qui sera détecté dans la phase de synchronisation.

Dans ce cas, il faut défaire les modifications ("Roll back") et en référer à l'arbitrage d'un proposent des procédures automatiques pour la résolution des conflits. La détection automatique est basée sur des définitions de pré conditions sémantiques.

Cette approche optimiste permet un plus haut degré de parallélisme et une meilleure disponibilité des données que ne le permet l'approche pessimiste offrant une cohérence forte. Son efficacité dépend toutefois du profil d'accès des utilisateurs qui partagent les informations répliquées : il est d'autant plus efficace qu'il y a peu d'accès conflictuels. Ensuite, quelle que soit la méthode de synchronisation adoptée, plusieurs stratégies peuvent être envisagées.

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

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

2.3.2.2. Stratégies de réplication :

- **Réplication passive (asymétrique) :**

La réplication passive distingue deux comportements d'un composant répliqué : la copie primaire et les copies secondaires.

La copie primaire est la seule à effectuer toutes les opérations d'écritures. Les copies secondaires passives, surveillent la copie primaire. En cas de défaillance de la copie primaire, une copie secondaire devient la nouvelle copie primaire [LAM01].

En d'autres termes, seule la copie primaire est accessible à la fois en lecture et en écriture, les copies secondaires ne sont accessibles qu'en lecture.

Le site hébergeant la copie primaire est appelé maître, les autres sites sont appelés esclaves. La réplication asymétrique obéit donc à une architecture maître/esclave. Une requête de mise à jour est toujours orientée vers le site maître, celui-ci se charge de la propager vers les sites esclaves.

Pour que la cohérence soit assurée, la copie primaire diffuse régulièrement son nouvel état à tous les secondaires, assurant ainsi un point de reprise (checkpoint) : en effet, seule la copie primaire, exécute (traite) la requête. Les autres sont mises à jour en diffusant le nouvel état. Dans le cas où la copie primaire est défaillante, un protocole d'élection d'une nouvelle copie primaire parmi toutes les copies secondaires est nécessaire. Ce nouvel état peut être soit :

- Identique à l'ancienne copie primaire donc aucun mécanisme n'intervient.
- Différent de l'ancienne copie primaire (représente un état antérieur), dans ce cas des requêtes doivent être réexécuter (roll back) pour obtenir le meilleur état.

Avantages

• Écritures rapides (une seule copie à mettre à jour) [SEB03].
• Consomme moins d'énergie par rapport à la réplication active [XAV98].

Inconvénients :

• Pas de lecture en parallèle.
• Moins de localité des données.
• Possibilité de perdre la copie primaire [SEB03].
• La perte de la copie primaire n'est pas transparente à l'utilisateur : Car si la copie primaire défaille avant de répondre à l'utilisateur, ce dernier doit être informé (la nouvelle copie primaire) pour qu'il puisse exécuter sa requête une autre fois [XAV98].
• Temps de réponse très grand surtout dans le cas de défaillance de la copie primaire.

La figure 2.1 illustre ce principe à l'aide d'un diagramme temporel. Les flèches horizontales représentent l'exécution de quatre composants : Client, S1, S2 et S3. Les Si sont les copies du composant répliqué S. Elles appartiennent à un même groupe. Le client envoie la requête uniquement à la copie primaire (traits en flèche continue) S1. Celle-ci traite la requête (petit rectangle horizontal), construit un point de reprise et l'envoie à l'aide d'un multicast fiable assurant l'ordre FIFO, aux copies secondaires S2 et S3 en précisant le changement de son état (message update). Après la mise à jour de leurs états, les copies secondaires envoie un "*ack*" à la copie primaire. La copie primaire envoie la réponse au client après réception des "*ack*" de toutes les copies secondaires. Le point de reprise permet de synchroniser l'état des copies secondaires avec celui de la copie primaire puisque celle-ci est la seule qui communique avec le reste du système [LAM01].

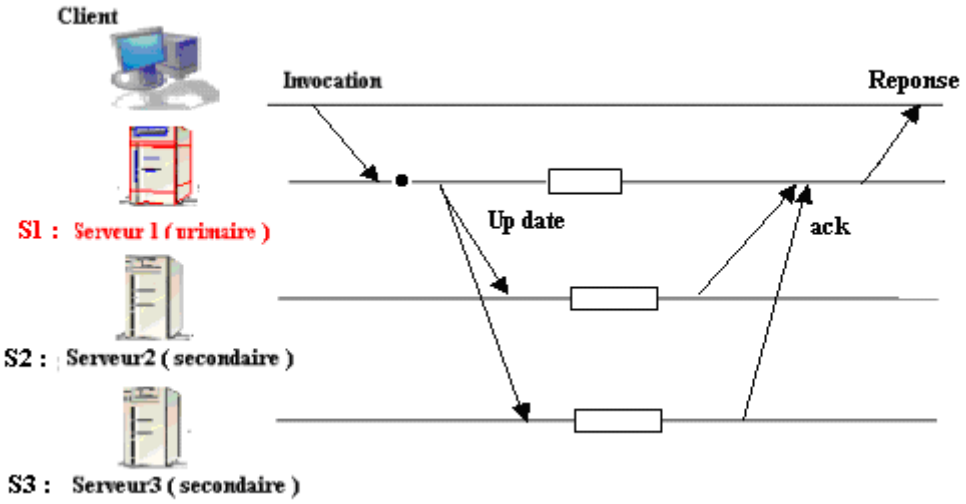


Figure 2.1 : principe de la réplication passive

- **Réplication active (symétrique) :**

Le but de ce type de stratégie est de fournir une copie cohérente à chaque consultation (lecture) pour cela, ce type est approprié à des applications critiques qui n'admettent en aucun cas la divergence entre copies comme les systèmes de gestion de base de données bancaire par exemple car, dans ces derniers, il faut fournir une copie cohérente à chaque lecture.

Dans cette approche chaque copie joue un rôle identique à celui des autres [LAM01] c.à.d qu'il y a une symétrie des comportements des copies d'un composant répliqué, en effet, toutes les copies reçoivent la même séquence des requêtes des clients dans le même ordre, la traite d'une manière déterministe (par conséquent toutes les copies produisent le même résultat pour une requête donnée) et y répondent par une même séquence de réponse.

La figure 2.2 illustre ce principe à l'aide d'un diagramme temporel. Les flèches horizontales représentent l'exécution de quatre composants : Client, S1, S2, S3. Les Si sont les serveurs sur lesquels se trouvent les copies du composant répliqué. Lorsque le client invoque la donnée, il envoie une requête à tous les serveurs Si à l'aide d'une diffusion. Chaque Si traite la requête (petit rectangle horizontal) et renvoie une réponse au client.

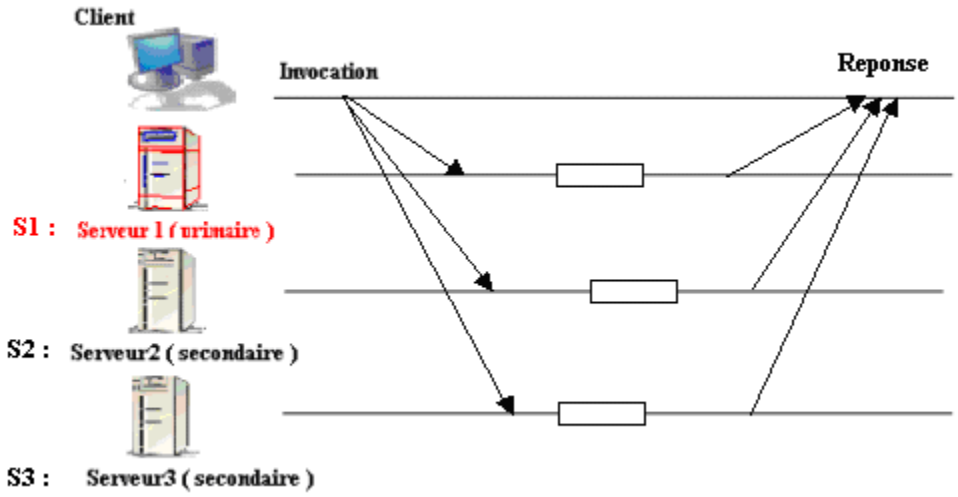


Figure 2.2 : principe de la réplication active.

Le projet *SOLIDOR* propose une à réplication active [SOL 99] où chaque copie est répliquée et les opérations de lectures/écritures sont exécutées sur la donnée simultanément sur n machines distinctes. Les réplicas doivent synchroniser leurs exécutions à chaque fois qu'une requête est reçue ou envoyée afin d'assurer que tout replica réalisant un même calcul reçoive les requêtes provenant des autres processus dans le même ordre. Cette dernière tâche n'est pas évidente et il faut rajouter un traitement de contrôle pour assurer cette synchronisation.

La réplication active présente les avantages suivants :

- Toutes les copies sont à jour.
- Possibilité de lectures parallèles sur un ensemble de copies [SEB03].
- Bonne localité des données.
- Temps de réponse moins élevé par rapport à la réplication passive qui sera étudiée ci-dessous.

En plus de ces avantages, la défaillance d'une copie est masquée par le comportement des copies non défaillantes. Comme chaque copie joue un rôle identique. La défaillance de l'une d'entre elle ne perturbe pas le service fourni par le composant.

Cependant ce type de stratégie a des inconvénients qui sont :

- Écriture lente : car toutes les copies effectuent la même séquence des opérations d'écritures (requêtes clients).

- Encombrement réseau [SEB03] : dû respectivement à la réception de la même séquence des requêtes par toutes les copies et l'émission de la même séquence de réponse aussi par toutes ces copies. En lisant ces inconvénients, on voit la nécessité d'établir un autre type de stratégie dans lequel on pallie à ces inconvénients.

- **Réplication semi active :**

La réplication semi active ce situe à mi-chemin entre la réplication active et la réplication passive, elle est active au sens où chaque copie traite la requête, et elle est passive dans le sens où seul la copie primaire envoi la réponse au client.

Donc :

- Toutes les copies reçoivent le même ensemble de requêtes.
- Toutes les copies traitent toutes les requêtes.
- La copie primaire (appelée leader) traite une requête dès qu'elle la reçoit ; par contre, une copie secondaire (appelée suiveur) doit attendre une notification de la copie primaire pour pourvoir traiter une requête.
- La copie primaire est la seule à émettre les réponses. [LAM01].

La figure 2.3 : illustre un exemple de ce principe.

- Le client envoie une requête à tous les Si.
- La copie primaire S1 envoie une notification (*notify*) aux secondaires et commencent le traitement de la requête.
- Les copies secondaires S2 et S3 ne commencent à traiter la requête qu'après avoir reçu la notification de la copie primaire.
- Si tôt le traitement terminé, S1 envoie la réponse au client.

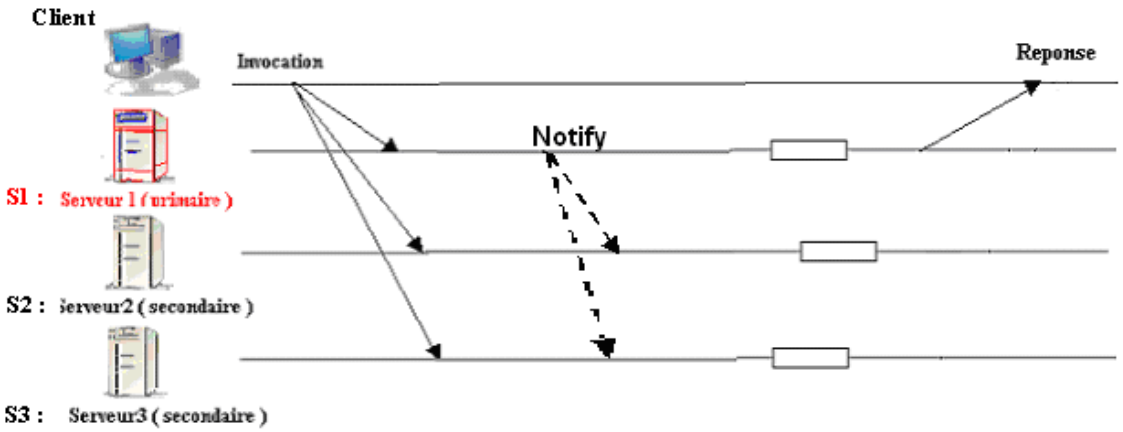


Figure 2.3 : Exemple sur la réplication semi active

Comme toutes les copies reçoivent la requête, le client n'a pas besoin de remettre la requête lorsque le primaire S1 tombe en panne. La nouvelle copie primaire S2 remet automatiquement la réponse au client. Deux situations peuvent se présenter suivant que la défaillance du primaire S1 est détectée par les copies secondaires avant ou après la notification.

Si la défaillance de la copie primaire est détectée avant la réception de la notification, la nouvelle copie primaire S2 envoie une notification concernant la première requête présente dans sa queue et la traite normalement. Si la défaillance de S1 est détectée après la

réception de la notification, S2 traite la requête correspondante sans envoyer de notification et envoie la réponse au client.

2.3.3 Protocoles de contrôle de cohérence :

Il existe plusieurs scénarios qui peuvent rendre une donnée incohérente dans un système distribué [BUS 07].

Des modifications simultanées sur la même donnée sont possibles entrainant un état divergeant des répliques dans le réseau, un mécanisme de « **contrôle de concurrence** » est donc nécessaire afin d'éviter la perte des mises à jour concurrentes.

Les modifications effectuées sur une copie peuvent ne pas être propagées au reste des répliques sur le réseau, des copies qui n'ont donc pas encore été mises à jour peuvent être accessibles, là encore, un mécanisme de « **contrôle de répliques** » est donc nécessaire au maintien de la cohérence dans le réseau.

La survenue d'une panne pendant une opération de modification peut engendrer une incohérence de la copie mise à jour. Un mécanisme de « **contrôle d'intégrité** » permet de rétablir la réplique à un état cohérent.

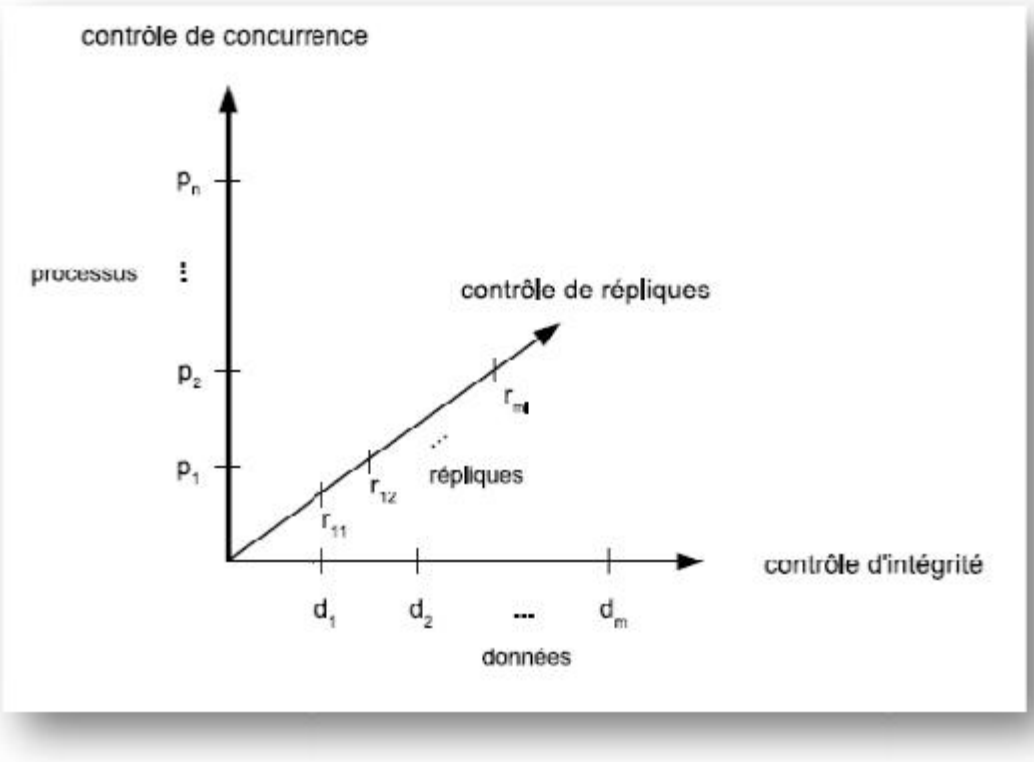


Figure 2.4 : Axes du problème de la cohérence [BUS 07].

1) Protocoles de contrôle de répliques :

La disponibilité d'une donnée dans un système distribué notamment les systèmes reposant sur des architectures P2P est à la source de beaucoup de travaux de recherches. Des méthodes de réplications [RAN O2] permettent actuellement d'assurer une disponibilité et une accessibilité accrue aux données.

La présence de plusieurs copies d'une même donnée pose néanmoins problème dans le cas des mises à jour. En effet, assurer la convergence des copies de tout le réseau et la cohérence des données répliquées est une opération complexe.

Selon qu'on privilégie la disponibilité des données répliquées ou alors la cohérence des répliques d'une même donnée, on peut adopter deux protocoles de contrôle de répliques à savoir les protocoles dits pessimistes ou alors optimistes.

a) Protocoles de contrôle de répliques pessimistes :

Les protocoles de contrôle de répliques dits pessimistes privilégient la cohérence des répliques d'une donnée. Ces dernières sont considérées comme une occurrence unique d'une donnée. Il existe trois grandes catégories de protocoles de contrôles de répliques pessimistes [RAN 02].

• Protocoles à copie primaire :

Le protocole à copie primaire est un protocole à deux niveaux de répliques. Lorsqu'une mise à jour est générée, elle est envoyée au premier niveau de répliques (*primary tier*), la copie primaire est donc chargée de recevoir les modifications effectuées, elle doit ensuite propager ces modifications aux répliques du second niveau (*secondary tier*). La propagation des modifications peut-être synchrone on parle alors de réplication active, ou alors asynchrone on parle alors de réplication passive [MOH07] dans ce dernier cas, le protocole à copie primaire est n'est plus un protocole pessimiste.

• Protocoles à base de quorums :

Dans les protocoles à base de quorums, les opérations de mises à jour sont effectuées sur des sous-ensembles de serveurs S intersectés deux à deux constituant ce qu'on appelle un quorum Q. Le principe d'intersection des sous-ensembles garantit la propagation des mises à jour à toutes les répliques [DRA 00].

• Diffusion atomique :

Les protocoles à diffusion atomique se basent sur une réplication active, en effet, les répliques de la donnée constituent ce qu'on appelle un *groupe de communication*, les processus envoient ensuite les mises à jour dans le même ordre à chaque réplique grâce à une primitive de communication de groupe multicast appelée *diffusion atomique.*

b) **Protocoles de contrôle de répliques optimistes :**

Les protocoles de contrôle de répliques optimistes privilégient la disponibilité et les performances d'accès aux données. Dans ce type de protocole, un degré d'incohérence est toléré au sein du réseau. En effet, une lecture peut ne pas renvoyer les dernières mises à jour effectuées, la seule garantie offerte est que les mises à jour soient propagées aux répliques d'une manière asynchrone (différée) et que les copies convergent à la longue, au bout d'un temps FINI.

Les protocoles épidémiques sont des protocoles de contrôle de répliques optimistes.

1) Protocoles de contrôle de concurrence :

A l'instar des protocoles de contrôle de répliques, il existe des stratégies pessimistes et optimistes en ce qui concerne les accès concurrents.

a) Protocoles de contrôle de concurrence pessimiste:

Ces protocoles ont pour principe d'éviter tout risque de conflit lors d'accès simultanés à une donnée partagée, les plus répandus sont ceux qui se basent sur des principes de verrou ou d'exclusion mutuelle. On peut citer principalement les deux protocoles suivants :

- **Protocoles à jeton :**

  Pour accéder en section critique, un processus doit au préalable acquérir le seul et unique jeton numérique qui circule entre les différents processus, l'unicité de ce jeton assure l'exclusivité d'accès à la section critique, à la fin de la section, le processus relâche le jeton et le donne au prochain demandeur.

  Ces protocoles sont caractérisés par un faible nombre de messages échangés et un délai de synchronisation minime, cependant, ils ne sont pas très robuste du fait de la potentielle perte du jeton et ils nécessitent de connaître tous les processus en concurrence (bien que des variantes opérantes à la demande soient apparus pour atténuer).

- **Protocoles à permission :**

  L'entrée en section critique se fait par demande de permission d'accès à un ensemble de serveurs déterminés, ces derniers gardent en mémoire la liste des processus demandeurs et leur attribut la permission à tour de rôle.

Comparativement aux protocoles à jeton, les protocoles à permission ont un temps de synchronisation et un nombre de messages plus élevés, néanmoins, le nombre de processus concurrents n'est pas limité [CAR 05].

2) Protocoles de contrôle de concurrence optimistes :

Contrairement à la stratégie pessimiste, la stratégie optimiste ne cherche pas à empêcher les accès compétitifs, mais vise plutôt à les détecter ou à les résoudre après que ces derniers se soient produits.

• **Détection de conflit :**

Il existe deux politiques de détection de conflits, la détection syntaxique et la détection sémantique.

La détection syntaxique s'effectue en identifiant les accès concurrents grâce à l'utilisation de vecteurs de versions, par exemple, ce mécanisme permet de détecter plus de conflits qu'il en existe en réalité dans le cas où la donnée est composée de parties indépendantes et que la mise à jour a concerné des parties distinctes.

La détection sémantique utilise par contre les caractéristiques sémantiques des mises à jour telles que la commutativité ou l'idempotence. Elle permet ainsi de ne détecter que les véritables conflits, elle est de ce fait plus efficace. Cependant, ces propriétés doivent être mentionnées par l'application ou par l'utilisateur lui-même [BUS07].

• **Résolution de conflit :**

Après la détection des conflits, l'utilisateur espère que la résolution se fasse de manière automatique et transparente. Néanmoins, ceci impliquerait, comme pour la détection sémantique, la connaissance des propriétés éponymes, mais aussi de l'application et du but de l'utilisateur.

A défaut de cela, la plupart des protocoles de résolution se contentent d'informer l'utilisateur du conflit en attente d'une résolution manuelle, d'autres annulent tout simplement une des mises à jour conflictuelles. Cette décision est la plupart du temps inévitable aussi bien dans le cas automatique que manuelle. Les mises à jour ne sont de ce fait que des « *tentatives*» [BUS 07].

2.3.4 Protocole de Gestion de Cohérence :

Un protocole de gestion de cohérence entre les différentes copies d'un même objet a pour objectif d'améliorer la fiabilité des données et/ou les performances (tant en écriture qu'en lecture) du système. Malheureusement, ces deux objectifs sont antagonistes (figure2.5). Pour obtenir une bonne fiabilité, il est nécessaire d'avoir une cohérence forte, ce qui pénalise les performances.

A l'inverse, pour obtenir de bonnes performances, il est nécessaire de relâcher la cohérence, ce qui pénalise la fiabilité. Cela est d'autant plus vrai que l'on augmente le nombre de copies. En effet, dans le cas de la cohérence forte, un site accède toujours au

dernier élément de la séquence globale des écritures. Ceci peut être réalisé par un protocole basé, par exemple, sur une diffusion atomique des valeurs, ou par le verrouillage global de la donnée avant l'ajout d'un élément à la séquence. Ces méthodes se basent sur un ordre total des écritures. Par contre dans le cas de la cohérence faible, on n'assure plus que la valeur lue sur un site (la dernière valeur de la séquence locale) est bien la dernière de la séquence globale. Cela est généralement réalisé par la construction d'un ordre partiel sur les écritures. Il est donc nécessaire de faire des compromis.

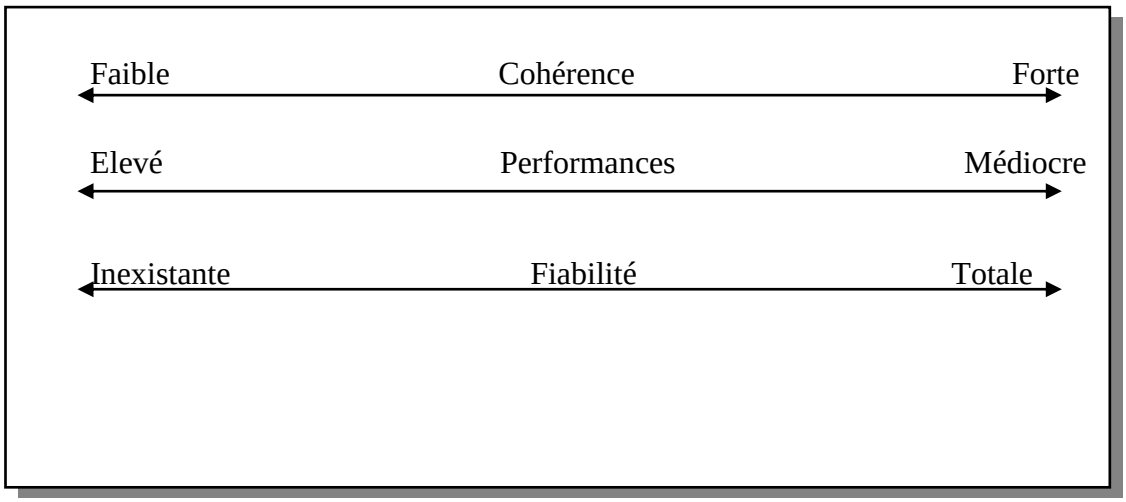


Figure 2.5 : Compromis Performance / Fiabilité et répercussion sur la cohérence

2.3.5 Conditions de synchronisation :

Les mises à jour des différentes copies peuvent se faire simultanément sur toutes les copies ou d'abord sur une et ensuite sur les autres. [MGA95] propose une classification des différents moments de déclenchement de la synchronisation sous le terme condition de cohérence. Sept classes de condition de cohérence sont définies :

Conditions sur le délai : Ces conditions portent sur le temps. Elles expriment la durée maximale que le protocole peut attendre avant la propagation d'une mise à jour. Par exemple, pour un délai maximum de 60 secondes, toutes les mises à jour d'une copie x doivent être propagées vers la copie x' avant l'expiration de celui-ci. Avec cette approche, seule la dernière valeur de x peut être propagée vers x'.

Conditions sur la périodicité : Ces conditions portent également sur le temps. Elles spécifient qu'une copie x' de x doit être mise à jour avec la dernière valeur de x toutes les m unités de temps, que x ait été modifiée ou non. L'avantage de cette approche sur la première est que les pertes de messages de mise à jour dues au réseau ou aux pannes de site sont écartées.

Conditions sur le moment : Ces conditions, introduites par [WIE87] et [WIE90], sont un cas particulier des conditions de périodicité. Elles spécifient qu'une copie x' de x doit être mise à jour à un moment donné avec la dernière valeur de x, par exemple tous les jours à 8h.

Conditions sur la version : Ces conditions spécifient le nombre de modifications pouvant avoir lieu sur la copie x avant que la copie x' soit mise à jour.
Conditions numériques : Si la valeur d'un objet est numérique, ces conditions permettent de borner la déviation entre les valeurs des différentes copies. Il est possible de considérer des différences absolues, relatives ou exprimées en pourcentage. Pour vérifier ces conditions, il faut connaître la valeur des deux copies.

Conditions sur les objets : Ces conditions portent sur la structure des objets. Trois types de conditions sont définies : x' doit être mise à jour avec la dernière valeur de x lorsque :

(a) au moins i sous objets de la copie x ont été modifiés,
(b) au moins q pourcent des sous objets de x ont été modifiés
(c) le sous objet a de x a été modifié, depuis la dernière mise à jour de x'.

Ces conditions sont particulièrement adaptées aux systèmes à objets, mais aussi à d'autres modèles, par exemple, les bases de données relationnelles. En effet, ces relations objets sous objets existent aussi entre les relations et les attributs, entre les relations et les n-uplets ou encore entre les n-uplets et les attributs. Par exemple, pour une copie x' qui est une relation en lecture seule servant pour faire des calculs statistiques, il est raisonnable de ne mettre à jour x' que si plus de q% des n-uplets de x ont changé.

Conditions événements : Finalement, le déclenchement des mises à jour des copies peut être dirigé par des événements. Cette classe de conditions est la plus générale. Un modèle d'événements assez riche permettrait d'exprimer les conditions précédentes.

2.3.6 Initiative de la mise à jour :

Les copies possédant l'information permettant de faire une mise à jour sont appelées copies sources, alors que les copies sur lesquelles doivent être propagées les modifications sont appelées copies cibles. Deux approches pour le rafraîchissement des copies sont possibles. Soit la copie source est l'initiatrice des propagations (figure 2.6 (a)), soit les copies cibles demandent les mises à jour à une (aux) copie(s) source(s) (figure 2.6 (b)).

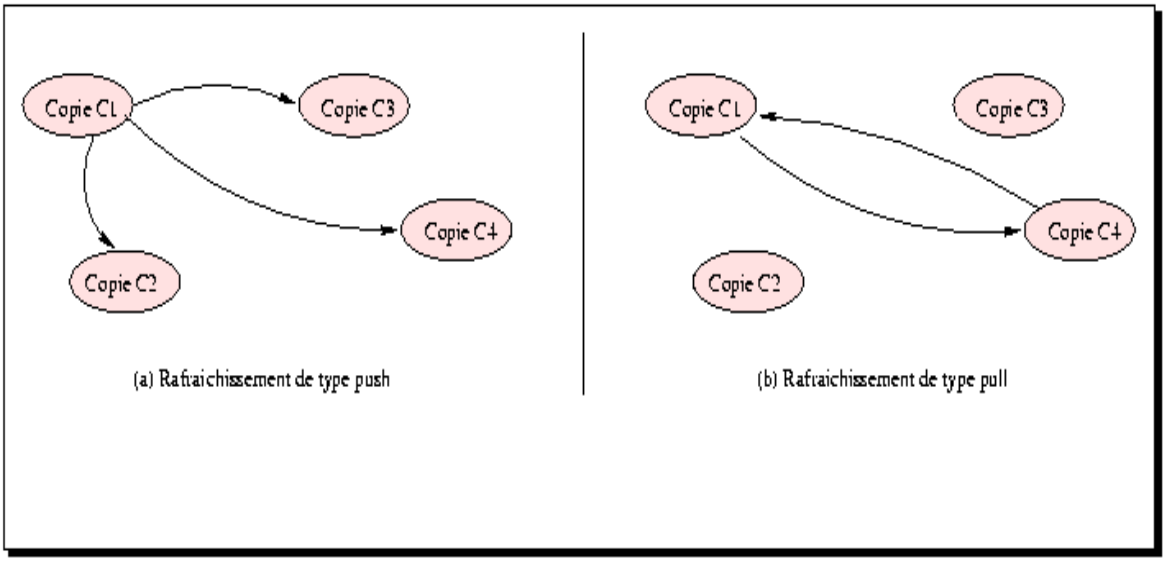


Figure 2.6 : Rafraîchissement de type push et pull

Définition : Rafraîchissement de type push

Propagation des mises à jour à l'initiative de la copie source vers les copies cibles.

Définition : Rafraîchissement de type pull

Propagation des mises à jour à l'initiative de la copie cible.

Avec la première approche, la copie source diffuse à toutes les copies cibles les mises à jour. Dans certains cas, cela peut poser des problèmes de passage à l'échelle. Des messages inutiles de synchronisation peuvent être envoyés à des copies qui ne seront pas consultées.

L'approche de type pull permet de réduire la charge réseau en ne propageant que les dernières modifications ou en les regroupant. Par contre, les copies cibles ne savent pas si une mise à jour est nécessaire. De plus, si elles interrogent trop souvent la copie source, cette approche peut s'avérer inintéressante.

2.3.7 Nature des mises à jour :

La nature des mises à jour désigne le type d'information envoyé aux différentes copies lors de la synchronisation. Dans le cas où le terminal ayant modifié la donnée propage ses modifications (Push), il a deux protocoles de propagation : les protocoles à diffusion des écritures et les protocoles à invalidation [SJE89].

Définition : Protocole à diffusion des écritures

Un protocole à diffusion des écritures envoie les modifications faites sur une copie aux autres copies.

Définition : Protocole à invalidation

Suite à la modification d'une copie, un protocole à invalidation envoie aux autres copies une notification les informant qu'elles sont invalides et ne doivent plus être accédées. Avant de pouvoir de nouveau être utilisée, les copies invalidées doivent être resynchronisées.

Le choix de Protocole à diffusion des écritures ou de l'invalidation dépend du type des données et de leur utilisation.

Pour les protocoles à diffusion des écritures, deux cas sont envisageables : soit il y a transfert des états, soit la procédure exécutée sur la source peut être propagée vers le site cible pour y être exécutée (duplication des opérations). Dans certains cas, la deuxième approche évite le transfert d'importants volumes de données.

Les protocoles à invalidation permettent de limiter la taille des messages échangés entre les différentes copies, alors que les protocoles à diffusion des écritures augmentent le trafic et les risques d'engorgement sur le médium de communication. De plus, le travail que doit fournir chaque copie est plus important. Celles-ci devant installer les mises à jour. Cependant, elles sont moins en retard que dans le cas des protocoles à invalidation.

Pour des données de grande taille, ou auxquelles ont fait peu d'accès, il est plus intéressant d'utiliser l'invalidation. Ainsi, on sauvegarde les ressources qui auraient été utilisées pour envoyer des données non lues. Au contraire, pour les données de petite taille, ou pour les données fréquemment utilisées, les mises à jour sont plus intéressantes : une donnée de petite taille prend peu de place par rapport à un message d'invalidation. Par ailleurs on économise les messages nécessaires pour obtenir la nouvelle version dans le cas d'une invalidation.

Certains systèmes utilisent une propagation hybride : certains sites, choisis sur des critères de fréquence d'accès, ou de capacités, reçoivent une mise à jour, tandis que les autres reçoivent une invalidation. Quand l'un de ces sites veut accéder à la donnée, il s'adresse au possesseur de la dernière mise à jour le plus proche [HDH10].

2.3.8 Topographie de la synchronisation :

Bien souvent, la copie devant propager une mise à jour le fait par diffusion à toutes les copies (figure2.7 (a)). D'autres protocoles propagent les mises à jour de copie en copie (figure 2.7 (b)) ou vers un ensemble de copies où chacune d'elles les propagent à son tour vers un autre ensemble (figure 2.7 (c)). Certains protocoles construisent, entre les copies, des chemins particuliers que doivent suivre les mises à jour (figure 2.7 (d)). On peut ainsi imaginer différentes topographies de propagation des mises à jour entre les copies.

Ces différents protocoles se justifient quand des copies (ou des groupes de copies) doivent être mises à jour avant d'autres ou qu'il est avantageux de s'appuyer sur la topographie du réseau.

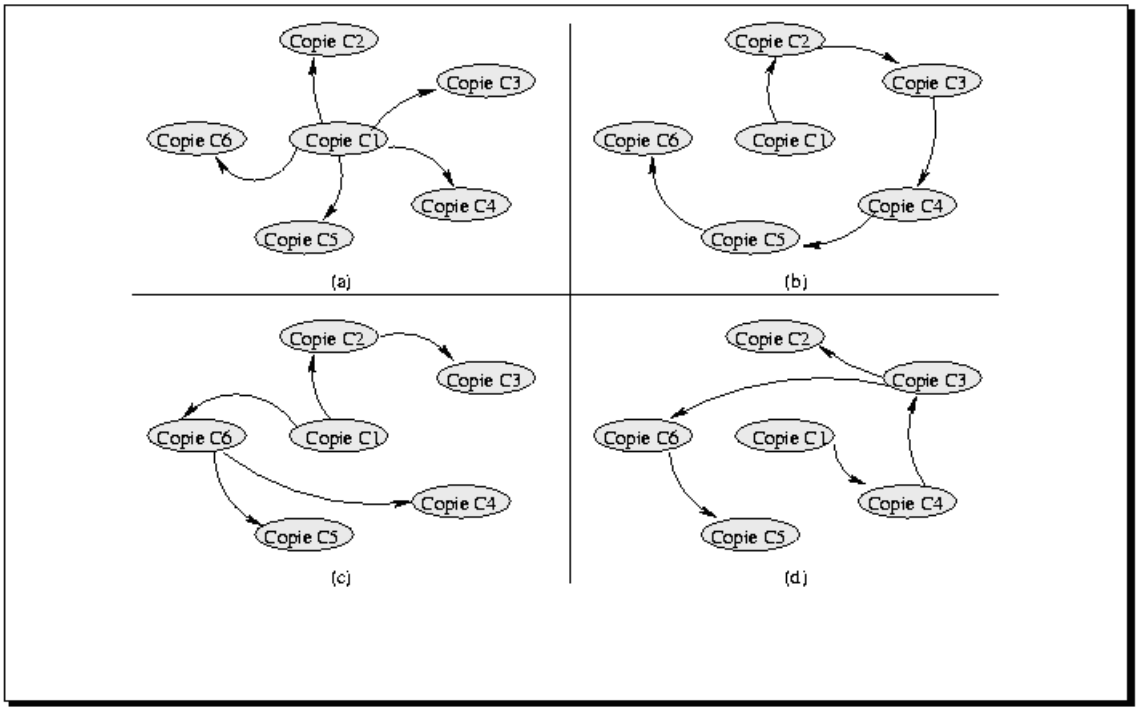


Figure 2.7 : différentes topographies de propagation des mises à jour

2.3.9 Protocoles de synchronisation :

Les protocoles de gestion de cohérence optimistes :

La cohérence lâche est réalisée par les algorithmes optimistes. L'efficacité et surtout la disponibilité sont jugées beaucoup plus importantes qu'une cohérence forte est comme trop pénalisante.

Dans cette classe le protocole on trouve :

- **Les protocoles à incohérence bornée :**

  Ces protocoles garantissent que les copies seront toutes mises à jour « au bout d'un certain temps ». Les modifications n'ont donc pas à être propagées de manière synchrone, ce qui permet de gagner en performance, puisque les propagations pourront avoir lieu de manière asynchrone et ce sans trop charger le système de communication. Les protocoles appartenant à cette famille sont par exemple :

« Epsilon_serializability » [WEB05], L'anti_entropie estampillée (''Timestamped anti_entropy'')[PST97] et la réplication paresseuse (''Lazy Replication'') [BOU03a].

- **Les protocoles incohérents :**

Ces protocoles se contentent de propager à faible coût les mises à jour, mais aucune garantie n'est fournie quant à la cohérence des données. Ces protocoles sont généralement utilisés par les applications du type serveur de noms. En effet, les informations fournies par un serveur de noms sont utilisées comme des indicateurs de localisation éventuellement erronés mais offrant de bonnes performances et à un coût de remise à jour faible.

Les protocoles pessimistes :

La cohérence stricte est réalisée par les protocoles pessimistes, qui interdisent les modifications d'objets sur deux copies au même moment. Ces algorithmes préfèrent réduire la disponibilité au profit d'une cohérence strict jugée indispensable pour le type d'application visée.

En résumé, les protocoles pessimistes se caractérisent par une synchronisation des accès aux copies avant les mises a jour, alors que les protocoles optimistes réalisent cette synchronisation après les mises a jour.

Les protocoles de cette classe sont dit cohérents : ces protocoles rendent visibles les modifications des données à tous les clients qui y accèdent «au même moment ». La propriété « au même moment » peut être prise au sens d'un temps global ou d'un temps virtuel.

2.4 Conclusion :

La réplication des données est devenue un outil indispensable qui permet de résoudre beaucoup de problèmes de disponibilité d'accès aux bases de données réparties et aux fichiers répartis et à tout type de donnée ou objet partagé. La réplication augmente la disponibilité des données ainsi que les performances de ces systèmes. Elle offre aussi une solution pour résister aux pannes et un nouveau mode de fonctionnement sur les machines deconnectables.

Néanmoins, la réplication a un coût dont principalement celui de gestion de la cohérence. Ce coût doit rester conforme par rapport aux services qu'offrent la réplication et le but de performance visé.

La cohérence des données partagées est maintenue à différents degrés selon les besoins des applications, parce que celle-ci n'ont pas les mêmes exigences concernant la cohérence des réplicas. Effectivement, dans un système de gestion de base de données

bancaire, assurer une cohérence forte est très important car, ce type d'application est critique et n'admet en aucun cas une divergence entre les réplicas. Par contre, dans l'informatique mobile (nomade) où les clients veulent travailler en mode déconnecté, assurer une cohérence faible est une solution alors qu'assurer une cohérence forte exige que le client soit connecté tout le temps pour bénéficier de l'accès aux données, sans quoi il ne peut continuer à travailler sur des données partagées chose difficile à assurer sur un système de réseaux mobile Ad hoc.

Chapitre 3 : Les systèmes de réplication

3.1 Introduction :

Ce chapitre décrit les travaux de recherche effectués dans le domaine des réseaux mobiles Ad hoc, qui concernent certaines des problématiques évoquées dans le chapitre précédent: réplication des données et maintien de la cohérence.
Dans ce chapitre nous présentons des solutions conçues spécifiquement pour supportent le travail en mode déconnecté.

Dans ce chapitre nous nous intéressons aux systèmes de partages de données et modifiables pour MANets.

Nous allons tout d'abord présenter des systèmes de partage de fichier pour MANET à savoir AdhocFS, BleuFS, HaddockFS, et le système de gestion de fichiers pour un environnement distribué à grande échelle CODA. Ensuite nous présentons un système de partage de données pour MANET: XMiddle, et le Protocole de réplication GBR, et nous terminaisons par un système de partage de données dans les environnements mobiles BAYOU. Ce dernier sera étudié en détail afin de pouvoir s'inspirer de la solution déjà existante dans les réseaux statiques.

3.2 AdhocFS :

AdhocFS [BOU03a] [BOU03b] est un système qui supporte le partage de fichier distribués au dessus d'un réseau mobile en mode Ad hoc. Il est constitué de trois couches au dessus d'un système d'exploitation. La première, correspond à la gestion de groupe, elle comprend les fonctionnalités de découverte et d'initialisation. La deuxième couche, correspond aux fonctionnalités de gestion de la cohérence et de la disponibilité des données et la troisième couche, correspond à l'interface d'AdhocFS.

Dans AdhocFS la notion de groupe se base sur celle du domaine de sécurité (DS), qui permet de sécuriser le partage des ressources entre entités du groupe. Un DS est définie par rapport à un centre d'intérêt professionnel ou privé. Concrètement, un domaine de sécurité délimite un ensemble d'entités mobiles et stationnaire par la possession d'un certificat numérique qui permet à ces entités mobiles de s'authentifier les unes auprès des autres. Ces certificats sont attribués après vérification appropriée de l'identité de l'entité. Ainsi, une entité peut appartenir à plusieurs DS suivant qu'elle possède les certificats numériques correspondants.

L'architecture d'AdhocFS considère l'existence d'une infrastructure stationnaire de confiance en plus de la collection de nœuds mobiles qui peuvent se déconnecter fréquemment de cette infrastructure. La gestion du groupe dans AdhocFS est réalisée de telle sorte à minimiser la consommation des ressources et particulièrement d'énergie au niveau des entités mobiles. Dans la Figure3.1 le groupe (1) ne pourra pas communiquer avec le groupe (4) car ils ne sont pas dans la portée de communication. Une entité peut appartenir à deux groupes de domaines d'intérêts différents (cas de l'entité appartenant aux groupes (1) et (2)). Aussi, une entité peut être un singleton [a].

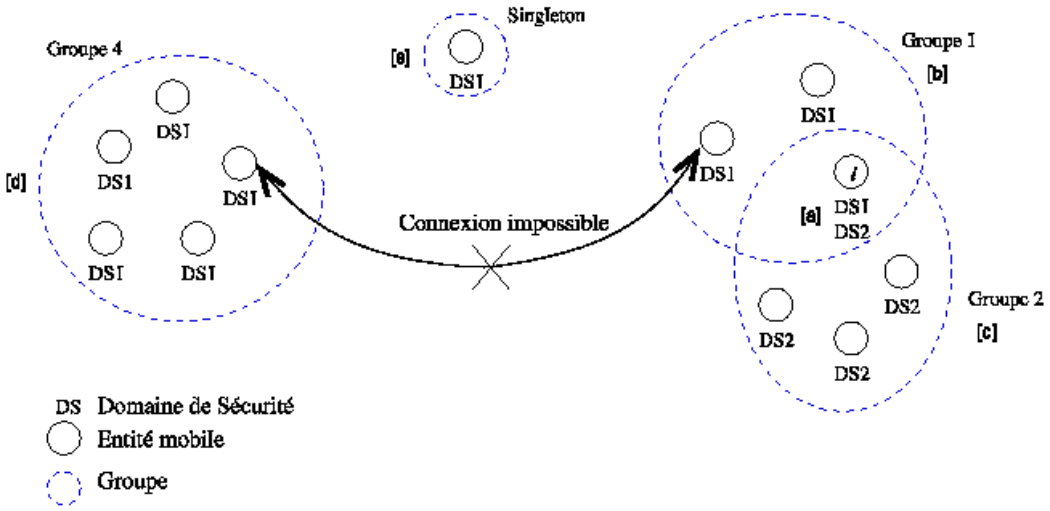


Figure 3.1 : Différentes configurations de groupes dans AdhocFS

Pour favoriser le partage des ressources en environnements mobiles la notion de groupe mobile au niveau d'un domaine de sécurité est définie comme étant un ensemble d'entités mobiles vérifiant les conditions suivantes:

1- Elles sont dans la portée de communication directe les unes des autres.

2- Elles appartiennent au même DS.

Dans AdhocFS, les entités qui appartiennent à un groupe communiquent selon le modèle pair à paire via un réseau sans fils en mode Ad hoc. La découverte des entités mobiles constitue la première étape du processus de création de groupe. A l'issu de cette étape, chaque entité détient la liste des entités avec lesquelles la création d'un groupe est possible (les entités qui sont dans sa porté de communication et appartiennent au même domaine de sécurité). La deuxième étape, consiste à authentifier les entités mobiles se trouvant dans la portée de communication par rapport à son DS. L'initialisation d'un groupe permet de créer le groupe. L'étape de découverte des entités allant former un groupe est complètement décentralisée. En revanche, pour l'initialisation du groupe, les entités devant former ce groupe désigne l'une entre elles comme étant le leader. Le leader du groupe est l'entité dont l'identificateur ID, est un extremum (minimum ou maximum) des autres IDs

du groupe. Le rôle du leader du groupe consiste à coordonner les tâches nécessaires pour l'initialisation du groupe.

3.2.1 Gestion de la cohérence :

AdhocFS adopte la gestion de la cohérence hybride, C'est à dire qu'au sein d'un groupe il maintien la cohérence forte. Par contre, au niveau du DS AdhocFS maintient la cohérence faible. La cohérence au sein d'un groupe mobile est gérée selon le protocole de l'écrivain unique. Toutes les opérations d'écritures se déroulent en exclusion mutuelle, par contre, les opérations de lectures sont partagées. Suivant le protocole de l'écrivain unique, une seule entité est autorisée à écrire sur un replica de donnée partagée. Ces mises à jours seront propagées aux autres réplicas de la même donnée stockés sur des entités du groupe au moment des accès effectifs à ces derniers suivant une propagation paresseuse des mises à jours c'est-à-dire qu'une entité ne recevra les mises à jour d'une autre entité que lors de l'accès effectif.

L'unicité de l'écrivain est assurée par la gestion d'un jeton unique pour l'ensemble des réplicas d'une donnée au sein d'un groupe. Ce jeton attribut le droit d'écriture à l'entité qui le possède.

Un replica au sein d'un groupe peut être dans l'un des cinq états suivants: Fresh, Read, Update Read, ReadWrite, Invalid [BOU03a]. La synchronisation des réplicas permet de détecter les mises à jour divergentes puis de résoudre les conflits générés par ces derniers. Seules les mises à jour effectuées dans des groupes disjoints peuvent être divergentes. En effet, dans un groupe la gestion de la cohérence forte permet de prévenir de ce type de mises à jour. De plus toutes les mises à jours effectuées sur un replica dans un groupe sont propagées, certes de manière paresseuse mais dans le même ordre, à toutes les entités ayant un replica local de la même donnée. Toutes les mises à jour sont répertoriées dans le CCL (Control Coherency List) associé au réplica.

Chaque CCL contient deux éléments :

(i) Une estampille qui indique avec quelle version de la copie de référence la dernière synchronisation a été effectuée (Quand).

(ii) Un historique qui indique par qui et comment la copie a été modifiée.

La détection des conflits consiste à comparer ces CCLs. Elle est réalisée au moment de l'accès effectif à un replica dans un groupe.

3.3 BlueFS :

BlueFS (Blue File System) [BNF] est un système de fichier distribué désigné pour l'informatique mobile« *mobile computing* » qui adresse les nouvelles opportunités et les défis créés par l'évolution du dispositif de stockage mobile.

Comparé au système de fichier Coda, BlueFS réduit l'utilisation d'énergie du système de fichier jusqu' à 55% et fournit un accès jusqu'à trois fois plus rapide aux données répliquées sur le dispositif de stockage portable.

BlueFS utilise « lire de n'importe qui et écrire à plusieurs stratégies ». Un module critique (de noyau) réoriente les appels du système de fichiers à un démon d'un niveau utilisateur « ***user-level*** », appelé *Wolverine*, ce dernier décide quand et où accède t-il aux données. La figure suivante montre l'architecture de BlueFS.

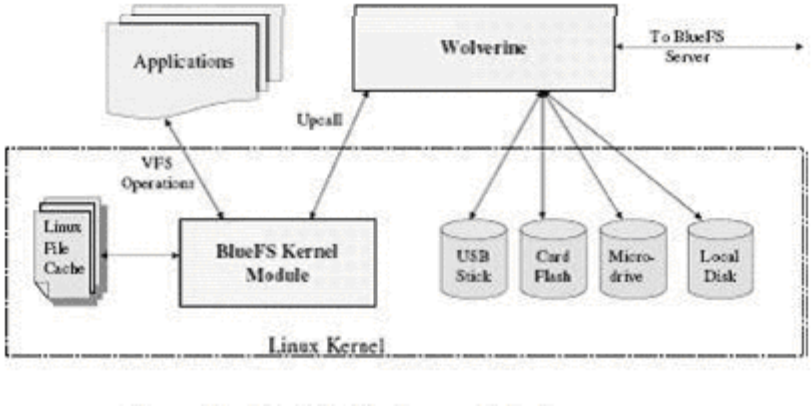


Figure 3.2 : architecture de BlueFS

3.3.1 Cohérence du cache dans BlueFS :

Pour les appels qui concernent la lecture des données, *Wolverine* ordonne des dispositifs connectés par l'exécution prévue et le coût énergétique de collection de données puis essaye de lire les informations à partir du dispositif le moins coûteux en terme d'énergie lors de la recherche des données. Ceci laisse BlueFS s'adapter quand des dispositifs de stockage portables sont insérés ou enlevés ou quand les conditions de réseaux sans fil sont variables.

Pour les appels qui concernent l'écriture des données, *Wolverine* réplique les modifications d'une manière asynchrone à tous les dispositifs connectés à un ordinateur mobile. *Wolverine* rassemble les modifications dans les files d'attente du dispositif. Ces modifications sont enfilées périodiquement au niveau des dispositifs de stockage.

BlueFS emprunte plusieurs techniques de Coda comprenant un support pour les opérations déconnectées et la réintégration asynchrone des modifications au fichier de serveur.

Le système BlueFS maintient chaque dispositif portable à jour avec les derniers fichiers dans l'ensemble de fonctionnement d'un utilisateur et avec la dernière version des fichiers.

Puisque toutes les modifications écrites à un dispositif portable sont écrites au fichier serveur également, BlueFS ne perd pas les données en cas ou un dispositif

portable est mal placé ou bien volé. BlueFS présente une image simple du système à travers tous les ordinateurs et les dispositifs de stockage portable.

- Le replica primaire de chaque bloc de données réside dans le fichier du serveur
- Le client et le dispositif de stockage portable stockent les replicas secondaires qui sont utilisés uniquement pour améliorer l'exécution et pour réduire l'utilisation d'énergie, par conséquence aucune donnée n'est perdue quand un dispositif portable est perdu, volé ou bien endommagé car le replica primaire réside toujours sur le serveur.

Contrairement à AdhocFS et Coda, BlueFS maintient des rappels de service «*callback* » par dispositif pour chaque fichier. Les rappels de service sont un mécanisme dans lequel le serveur se rappelle qu'un client a caché un objet et informe le client quand l'objet est modifié par un autre client. En cas de modification, le serveur envoie une invalidation au client auquel le dispositif portable est actuellement connecté. Si le dispositif est déconnecté, l'invalidation est enfilée et délivrée quand le dispositif sera connecté prochainement à un client. À la réception d'une invalidation, le client met à jour un objet appelé « *l'objet enode* » pour indiquer que les données et/ou le métadata ne sont plus valides. D'autres invalidations sont placées sur la file d'attente d'inscription du dispositif. Quand les files d'attente sont défilées, les objets invalidés sont supprimés.

On peut dire que la cohérence du cache dans BlueFS repose sur deux parties de travaux antérieurs: L'utilisation de Coda du contrôle de la réplication optimiste et l'utilisation de AdhocFS des rappels de service pour la cohérence du cache.

BlueFS ajoute deux modifications importantes. D'abord, BlueFS maintient des rappels de service sur une base de par dispositif, plutôt que sur une base de par client. En second lieu, le serveur de BlueFS enfile des messages d'invalidation quand un dispositif est déconnecté. Ces modifications ont laissé BlueFS soutenir efficacement des médias portables et des clients qui hibernent (reposent) fréquemment pour économiser de l'énergie.

Le contrôle de la réplication optimiste fournit une grande disponibilité et améliore l'exécution en permettant à des clients de lire ou écrire des données sans obtenir des verrous. Quand deux clients modifient concurremment un fichier, le conflit write/write est détecté au niveau du serveur et marqué pour une résolution manuelle. Comme Coda, BlueFS sauvegarde un numéro de version dans le metadonnée (*métadata*) de chaque objet.

Chaque opération qui modifie un objet incrémente également son numéro de version. Avant de commettre une opération de mise à jour, le serveur vérifie que le nouveau numéro de version de chaque opération de modification est exactement plus grand que le numéro de version précédent. Si ce contrôle échoue, on en déduit que deux clients ont fait des modifications conflictuelles à l'objet. L'utilisateur doit résoudre de tels conflits en choisissant une version de sauvegarde. Dans l'absence de déconnexion, de tels conflits se produisent seulement quand deux clients différents mettent à jour le même fichier en 30

secondes d'intervalle de temps (c.-à-d. la quantité du temps où une mise à jour peut résider dans la file d'attente d'inscription du serveur). Une expérience antérieure des autres systèmes de fichiers distribués a prouvé que de tels conflits sont rares [JKS92].

3.3.2 Gestion des conflits :

Il faut Noter que BlueFS ne peut pas éliminer toutes les incohérences parce qu'il permet les opérations déconnectées et écrit les modifications d'une manière asynchrone pour améliorer l'exécution et l'efficacité énergétique. Les mises à jour concourantes sur le même fichier peuvent créer des conflits où deux versions du fichier existent.

Comme dans beaucoup d'autres systèmes qui supportent la cohérence faible, de tels conflits sont automatiquement détectés et marqués pour la résolution manuelle ou automatique.

3.4 Haddock

Haddock FS [JPF04] est un système de fichiers distribué pour réseau mobile Ad hoc Dans Haddock FS, un terminal hébergeant une donnée est appelé gestionnaire de réplique (*replica manager*). La politique de réplication n'est pas présentée, on suppose qu'elle est faite à la demande.

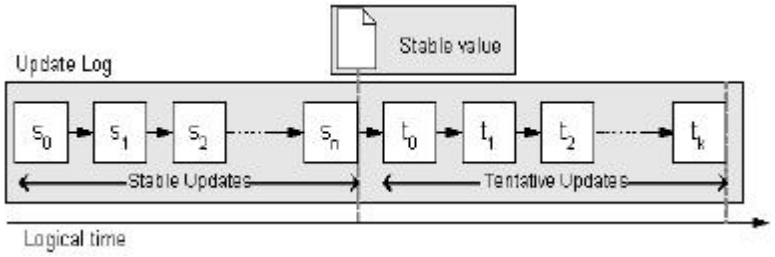


Figure 3.3 : Journalisation des mises-à-jour [JPF04]

Haddock FS propose un protocole de cohérence hybride. Au sein d'un groupe fortement connecté, HaddockFS met tout d'abord en place un protocole de cohérence pessimiste basé sur un algorithme d'exclusion mutuelle à jeton. La détection d'un groupe fortement connecté et la mise en place de l'exclusion mutuelle n'est pas décrite.

Entre les groupes, la cohérence optimiste est mise en place en journalisant les modifications. Celles ci sont triées par ordre causal. Dans son journal, chaque gestionnaire de réplique stocke donc des mises à jour, ainsi que la version stable de la donnée, c'est à dire la dernière version sur laquelle; il sait que tous les gestionnaires de répliques se sont accordés. Les mises à jour plus anciennes que la version stable sont appelées mises à jour

stables (*stable updates*), celles plus récentes sont appelées mises à jour provisoires (*tentative updates).*

La réconciliation est faite entre couple de gestionnaires de répliques, dès que ceux ci sont en contact. Cette réconciliation vise à construire une liste de mises à jour totalement ordonnée suivant l'ordre causal (*un ordre partiel*). Quand il y a plusieurs modifications en parallèle, une seule sera donc choisie. Avec une granularité de l'ordre du fichier, c'est un choix très contraignant, puisque cela empêche la modification concurrente des fichiers, même dans des cas où un algorithme tel que diff-3[SKB07], capable de construire la différence minimale entre deux chaînes de caractères ayant un ancêtre commun, ne détecterait pas de conflit.

Pour obtenir une version stable, chaque donnée a une copie primaire, la première version à avoir été créée. C'est cette version qui décide au final en cas de conflit, et qui est autorisée à créer des versions stables. Le statut de copie primaire peut être transféré, mais crée un point de centralisation.

3.4.1 Stockage et mise à jour

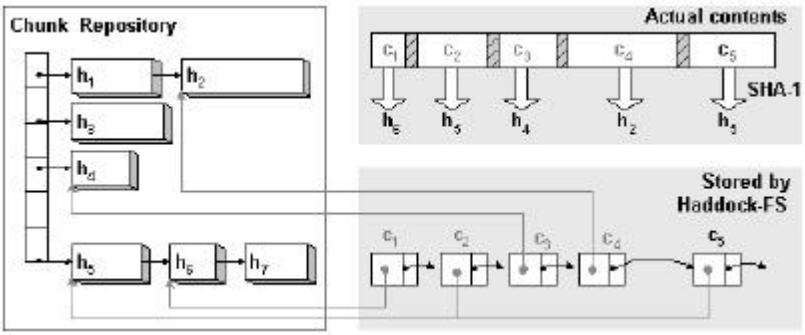


Figure 3.4 : Stockage des *chunks*

HaddockFS cherche à limiter l'occupation mémoire. Il faut donc réduire la taille des journaux de mises à jour stockés en mémoire, et la taille de la mise à jour elle-même échangée entre deux pairs. Pour diminuer la taille des échanges, Haddock FS utilise la fonction de hash SHA-1. Celle ci prend en entrée une chaîne de caractères et retourne une chaîne de caractères statistiquement unique, et de plus petite taille.

Chaque donnée est tout d'abord découpée en morceaux, appelés *chunk* (morceau) ; comme dans la figure 3.4. Un pair indexe ensuite les données qu'il héberge par leur valeur hachée. Quand la donnée est modifiée, un nouveau *chunk* est créé et inséré dans la liste de *chunk* existants.

Au moment de la propagation des modifications, les valeurs hachées sont tout d'abord échangées afin de vérifier s'il est nécessaire de propager les modifications elles mêmes. Pour limiter la place prise par la journalisation des modifications, HaddockFS élimine d'abord les mises à jour antérieures à la version stable de la donnée (obtenu grâce à la copie primaire), et pour lesquelles, il est certain que tous les pairs ont appliqués ces mises à jour. Le choix d'éliminer des mises à jour n'ayant pas encore abouti à une version stable est laissé à l'utilisateur et dans ce cas les mises à jour les plus anciennes sont éliminées, car ce sont celles qui ont le plus de chance d'avoir déjà été propagées.

3.5 CODA :

CODA est un système de gestion de fichiers pour un environnement distribué à grande échelle qui a été adapté aux utilisateurs mobiles [KIS91] [SAT 02] CODA a été développé avec les hypothèses que la plupart des accès sont des lectures et que les accès en écriture concurrents sont pratiquement inexistants. C'est pourquoi, CODA peut utiliser la réplication ce qui rend le système plus fiable et robuste.

Dans ce dernier, les objets répliqués sont des fichiers et des répertoires qui sont dupliqués à partir des disques du serveur sur les caches des clients pour de meilleures exécutions et performances. Ces systèmes sont basés sur une architecture client/serveur et ils utilisent des politiques de cohérence optimistes. De telles options de conception sont appropriées aux réseaux fixes, où l'infrastructure de serveur devrait être toujours accessible aux clients liés. Le partitionnement de réseau et les pannes de serveurs sont des évènements rares. Dans ce contexte, le problème de disponibilité peut être différencié par des politiques de cohérence plus fortes qui assurent la sémantique des fichiers locaux. La popularité et l'utilisation large de ces systèmes dans des domaines des réseaux fixes sont des indices de leur efficacité.

Cependant, quand on prend en compte les clients mobiles, la partition de réseau entre ces clients et l'infrastructure des serveurs n'est plus négligeable [JHE99]. Pour réaliser une meilleure disponibilité en présence de tels clients potentiellement débranchés, le système de fichiers répartis Coda [SAT 02] permet à des clients d'accéder aux fichiers stockés dans le cache tout en étant déconnecté des machines serveurs [KIS 91]. En plus des opérations en mode déconnecté des clients, Coda permet également des opérations de lecture/écriture optimistes sur les copies des serveurs afin de réaliser une meilleure tolérance aux fautes face aux pannes des serveurs.

CODA utilise une technique de réplication optimiste pour les fichiers. 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 system 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.

En résumé, en mode connecté, les clients sont reliés à l'infrastructure de serveurs et le protocole de cohérence pessimiste du cache d'AdhocFS (Andrew File System qui est un système de fichier) est utilisé. Quand un client devient déconnecté des serveurs, il adopte un mode de fonctionnement distinct. Une stratégie optimiste de cohérence est alors employée pour permettre au client déconnecté de lire et de mettre à jour le contenu des fichiers et des répertoires locaux en cache.

3.5.1 Les objectifs de CODA :

CODA offre un accès transparent à la localisation à un espace de nom Unix partagé qui est situé sur un ensemble de serveurs de fichiers [MES 95]. Mais, Coda représente une avancée substantielle par rapport aux anciens systèmes de gestion de fichiers répartis car il supporte beaucoup mieux les pannes de serveurs ou de réseau. L'amélioration en disponibilité est réalisée en utilisant les techniques complémentaires de réplication des serveurs et de fonctionnement en mode déconnecté.

Donc, on peut résumer les objectifs de Coda dans :
Viser une disponibilité permanente de l'information chez le client, y compris dans les cas suivants :

- Déconnexion volontaire
- Déconnexion accidentelle
- Faible connectivité :
    - Connexion intermittente (communication mobile)
    - Connexion à très faible débit.

En utilisant les principes de réplication des serveurs et mode déconnecté.

3.5.2 Mode déconnectée et mode faiblement connecté :

Le mode déconnectée est le premier concept introduit par Coda [KIS 91]. C'est une étape importante dans la gestion des déconnexions. Dans le mode déconnecté, le client continue d'avoir accès aux données dans son cache pendant les déconnexions intermittentes.

La possibilité de fonctionner en mode déconnecté peut être utile même lorsque la connexion est disponible, par exemple, pour prolonger la vie de la batterie ou réduire les dépenses de transmission.

La transparence est préservée du point de vue de l'application, puisque c'est le système qui porte la responsabilité de propager les modifications, de détecter les conflits et de faire les mises à jour quand la connexion est restaurée.

Dans les applications client serveur mobiles, un client en mode déconnecté souffre de beaucoup de limitations telles que :

1. La mise à jour n'est pas visible à tous les utilisateurs.
2. L'absence des données dans le cache peut empêcher l'utilisateur de travailler sur son terminal mobile.
3. Les mises à jour peuvent conduire à la perte ou à l'endommagement des données.
4. Les conflits de mises à jour deviennent de plus en plus probables.

Pour traiter ces conflits, Coda [Mum96] exploite la faible connectivité pour deux raisons principales :

- Faire une validation rapide du cache après une faute intermittente.
- Opérer une propagation des modifications "goutte à goutte" en tâche de fond pour ne pas dégrader les performances.

3.5.3 Connexion :

La disponibilité est un facteur critique pour le travail en mode déconnecté. Si, au moment de la déconnexion, l'ensemble de fichiers en cache n'inclut pas les fichiers sur lesquels l'utilisateur mobile travaillera pendant la phase de déconnexion, l'absence dans le cache se produira, et l'exploitation normale du système est perturbée.

Afin de choisir l'ensemble des fichiers qui devraient être en cache pendant la phase de déconnexion, CODA combine deux stratégies distinctes. La première est basée sur l'information implicite recueillie les fichiers à usage récent en utilisant la politique <<moins récemment utilisé>> de substitution de cache.

Complémentairement, l'information est déduite d'une liste qui contient les noms des fichiers qui intéressent chaque utilisateur mobile. Ces fichiers ont une première priorité de cache qui est plus haute que les autres fichiers.

3.5.4 Détection et résolution de conflit :

Une conséquence de l'opération de déconnexion est que les mises à jour concurrentes qui se produisent tandis que certaines copies sont déconnectées peuvent créer des cas d'incohérence. Ceci soulève le problème de détection et de résolution des conflits qui caractérisent n'importe quelle approche de réplication optimiste. Ces problèmes sont traités pendant la phase de réintégration. Le système de fichiers Coda doit traiter deux types de base d'objets répliqués : répertoires et fichiers. Coda résout le problème de conflit de répertoire en employant une approche sémantique qui détecte et résout automatiquement les conflits.

Un exemple de tel conflit est quand deux clients déconnectés, créent de nouveaux fichiers sur un répertoire commun réplique. Ceci cause un conflit détecté à la phase de réintégration des clients, puisque les copies de répertoires ont été concurremment mises à jour. Dans le cas de répertoires, Coda implémente un protocole pour résoudre les conflits en utilisant la connaissance sémantique. Coda résout facilement ce conflit en incluant tous les nouveaux fichiers dans le répertoire.

Dans le cas des fichiers, aucune connaissance sémantique n'est disponible au sujet de leur contenu. Pour cette raison une approche syntaxique est adoptée basé sur le vecteur de version pour résoudre ce genre de conflit.

Quand un fichier est mis à jour, le serveur incrémente le numéro de version du fichier et celui du volume qui contient le fichier. Quand la connexion est restaurée, le client présente les numéros de volume pour la validation. Si le numéro de volume est valide alors Coda valide tous les fichiers cachés de ce volume. S'il est incorrect alors il doit valider fichier par fichier. D'autre part, si la validation individuelle d'un fichier n'est pas possible à cause de la faible connectivité, le client peut supposer que tous les éléments dans le volume sont incorrects ou reporter la validation jusqu'à la prochaine utilisation de l'élément.

Coda ne possède pas la connaissance sémantique suffisante pour résoudre des conflits de fichier, il offre un mécanisme pour installer et appeler d'une manière transparente un résolveur spécifique à l'application (ASR). L'ASR est un programme qui encapsule la connaissance détaillée et nécessaire à l'application pour distinguer les incohérences des différentes réconciliations. Si l'ASR ne peut pas résoudre les conflits, l'incohérence est exposée à l'utilisateur pour la réparation manuelle.

3.5.5 Avantages et inconvénients :

Avantages :

- CODA fournit un accès global à une base de fichiers répartie sur un ensemble de sites en assurant la déconnexion (Réintégration des données du client déconnecté).
- Il fait un usage intensif des possibilités offertes par les caches des fichiers sur les stations clientes afin de bénéficier de bonnes performances
- Solutions de disponibilité efficace et fortes pour les mobiles ou système faiblement cohérents basés sur une infrastructure de serveur.
- Résout des conflits de serveurs et gère aussi les échecs réseaux provenant des partitions serveurs.

Inconvénients :

- Activités de collaboration dans les réseaux mobiles Ad hoc rejetés : La propagation des mises à jour dépend de l'infrastructure de serveur.

- Nécessité des espaces de stockage élevé en mode déconnecté.
- Les applications doivent être prêtes à voir leurs mises à jour abandonnées en raison de la résolution des conflits pendant la phase de réintégration.

3.6 XMiddle :

XMiddle [CLS02] est un système de partage de données éditables destiné aux MANETets. Il a été développé en 2002 à l'University College à londres.

La particularité de XMiddle est que les données partagées sont structurées sous forme d'arbre XML. Cela permet de partager des sous-arbres: un utilisateur n'a donc pas besoin de répliquer l'intégralité d'une donnée. Par ailleurs, l'utilisation de balises XML permet d'introduire des informations sémantiques sur les données. Elles sont utilisées par XMiddle dans le cadre de la recherche de données.

3.6.1 Cohérence :

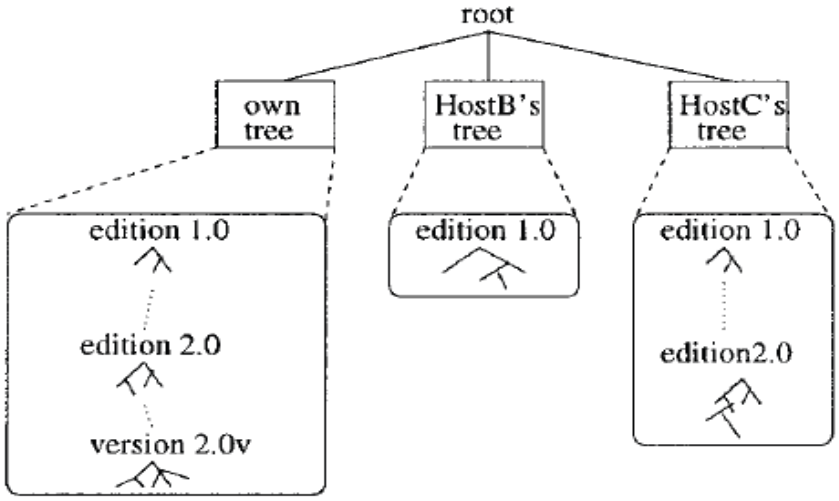


Figure 3.5 : Arbre des versions dans XMiddle [CLS02]

XMiddle offre un modèle de cohérence hybride: le propriétaire du document héberge la copie primaire et ordonne séquentiellement les accès à la donnée dans la partie du réseau à sa portée. Hors de portée du propriétaire du document, les différentes répliques d'une donnée peuvent être éditées en parallèle. Les différentes versions de la donnée sont conservées jusqu'à ce qu'une réconciliation avec le propriétaire soit validée, comme illustré par la figure 3.5. Les terminaux non propriétaires effectuent des réconciliations intermédiaires entre eux. Un algorithme de fusion d'arbre XML permet de réconcilier automatiquement les différentes versions.

3.6.2 Réplication :

Dans XMiddle, la réplication est laissée au soin de l'utilisateur: une donnée est répliquée à la demande. Il n'y a donc aucune réplique superflue de créée. Cependant, ce mode de réplication ne garantit pas la continuité d'accès à la donnée dans le réseau si tous les terminaux l'ayant répliquée deviennent hors de portée.

3.7 Bayou :

Bayou est un projet de recherche du laboratoire Xerox Parc. C'est [DPW94] un système qui offre à un ensemble d'utilisateurs mobiles le partage de données se trouvant sur plusieurs serveurs répliqués. Les applications concernées par ce partage ne nécessitent qu'une connexion intermittente comme les agendas partagés, les bases de données partagées, etc. [EPT97]

Bayou met en place une stratégie de réplication optimiste de type read-any/write-any. Cette stratégie autorise les clients à lire et à écrire sur n'importe quels serveurs. Les serveurs propagent ensuite ces modifications entre eux suivant un protocole de réconciliation incrémental (anti-entropy protocol) [PST96] prenant en compte d'éventuelles corruptions des serveurs [SDT97] [SDT99].

3.7.1 Architecture :

Dans Bayou, chaque collection de données est répliquée dans plusieurs serveurs de données (figure 3.6). L'application cliente communique avec le serveur à travers une interface spécifique à Bayou. Cette interface est implantée dans la souche du client et elle fournit deux opérations de base de lecture et d'écriture.

Bayou réalise un schéma de réplication Read-any/Write-any, c'est-à-dire Bayou utilise un modèle de cohérence faible sur les réplicas. Un problème de l'approche Read-any/Write-any est que les incohérences peuvent apparaître même lorsque seulement un utilisateur ou une application fait des modifications sur les données. Par exemple, un client écrit sur une base de données d'un serveur, et après un moment, lit des données d'un autre serveur. Le client peut voir des résultats, contradictoires à moins que les deux serveurs aient mis à jour leurs bases de données de façon cohérente.

Pour atteindre le maximum de cohérence entre les différentes copies de la base de données, Bayou utilise un protocole dit anti-entropique pour la propagation des mises à jour [PST 97]. Le protocole anti-entropique assure que toutes les copies d'une base de données sont convergentes vers le même état. En plus, le protocole anti-entropique permet à deux machines quelconques qui peuvent communiquer de propager périodiquement leurs mises à jour entre elles.

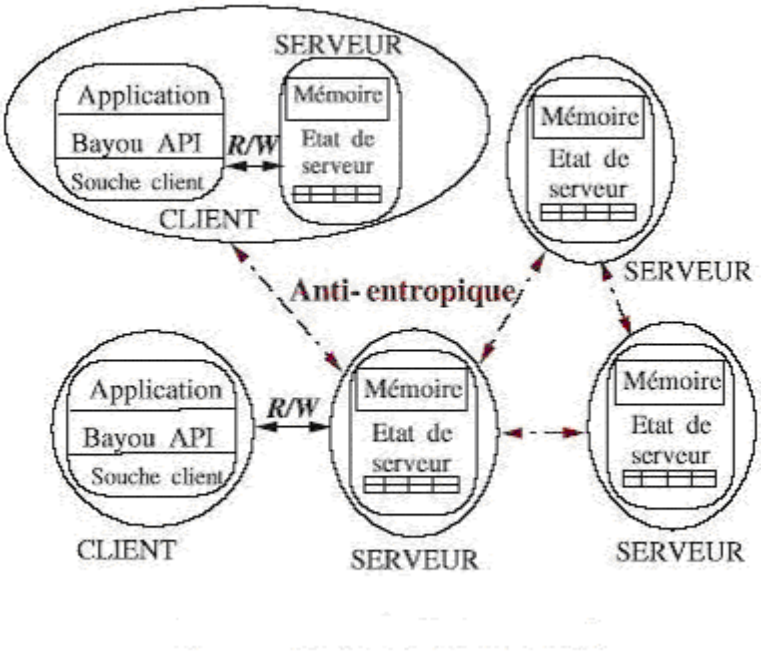


Figure 3.6 : L'architecture de Bayou

Dans le système Bayou, une copie de la base de données se trouve sur chaque serveur, les applications (clients) interagissent avec cette base de données à travers une interface de programmation Bayou API, (Application Programming Interface). Cette API permet à l'utilisateur d'exécuter deux types d'opérations sur la base de données, il s'agit des opérations de lecture et d'écriture. [BOU03a][BAG95].

La conception de Bayou concentre sur le support de mécanisme spécifique à l'application pour détecter et résoudre les mises à jour conflictuelles qui sont naturelles dans un système pareil, assure que les copies convergent vers une cohérence éventuelle, et définit un protocole par lequel la résolution des conflits est stabilisée.

3.7.2 La gestion de cohérence des réplicas :

Quand un replica est tenu par deux serveurs différents en même temps, son contenu peut diverger car, par exemple, ces deux serveurs ont reçu et exécuté différents écritures pour la même copie, une propriété fondamentale du système Bayou est qu'il garantie que :

- Chaque serveur éventuellement reçoit toutes les écritures à travers le protocole d'anti-entropie.

- Deux serveurs ayant le même ensemble d'écriture, vont avoir le même contenu de donnée.

Bayou ne peut pas imposer des limitations strictes aux délais de propagation des mises à jour puisque ceux-ci dépendent des caractéristiques de la connexion au réseau, facteurs que Bayou ne peut pas contrôler.

Afin de garantir une cohérence de toutes les copies du serveur, le modèle Bayou présente deux caractéristiques importantes : la première consiste à exécuter toutes les opérations d'écriture dans le même ordre sur tous les serveurs. La deuxième est que les procédures de détection et de résolution de conflits sont déterministes ce qui entraîne une relation de bijection entre le conflit détecté et la résolution choisie. Ainsi, pour chaque conflit détecté, on a une et une seule procédure de résolution exécutée.

3.7.3 Le protocole d'anti-entropie de Bayou :

Dans un système de réplication à cohérence faible, il est nécessaire d'avoir un protocole qui sert à propager les mises à jours acceptées dans un replica à tous les autres replicas en vu de réconcilier leurs états. Il s'agit du protocole d'anti-entropie de Bayou qui sera décrit dans ce paragraphe :

a. Le protocole d'anti-entropie de base :

a.1. Principes du protocole d'anti-entropies :

Le but de ce protocole est que deux copies se mettent à jour. Dans Bayou chaque copie est dans un endroit appelé serveur. Ce serveur comporte un journal des mises à jour (écriture) *ordonné* et une base de données qui est le résultat de l'exécution de ces écritures ordonnées. Le journal des écritures contient toutes les écritures qui ont été reçues par ce serveur à partir d'une application ou d'un autre serveur.

Quand un serveur de Bayou reçoit d'abord une écriture d'une application cliente, il lui assigne de manière croissante une unique estampille d'acceptation (un numéro d'acceptation). L'estampille d'acceptation peut être une estampille horloge ou bien un simple compteur. Au moment de la propagation, chaque écriture porte son estampille d'acceptation et l'identificateur du serveur qui a établie l'estampille. L'estampille d'acceptation définie un ordre total dans l'ensemble globale des écritures acceptées par un serveur et un ordre partiel qui s'appelle l'ordre d'acceptation entre toutes les écritures du système. L'ordre d'acceptation de l'écriture A précède celui de l'écriture B si elles sont acceptées par le même serveur et l'écriture A est acceptée avant l'écriture B. Les écritures sont ordonnées dans le journal des écritures selon leur ordre d'acceptation.

Le protocole anti-entropique permet la réconciliation point à point entre n'importe quel couple de serveurs indépendamment des autres. Ce qui fait que chaque serveur peut choisir son partenaire d'une manière arbitraire ou selon une prêt connaissance sans se soucier de la topologie du réseau ou du type de la connexion (sans fil par exemple).

Le protocole anti-entropique a été conçu pour être unidirectionnel où l'un des serveurs met l'autre à jour en envoyant les mises à jours inconnues par le récepteur. L'avantage de ce choix est que cette procédure nécessite un échange initial d'information d'état ensuite la communication continue dans un seul sens, de l'émetteur au récepteur en utilisant toute la bande passante.

Le protocole d'anti-entropie est basé sur l'échange des opérations d'écritures au lieu du contenu de la base de donnée. Il établie une contrainte dite la contrainte de **préfixe** qui peut être exprimée ainsi :

Un serveur qui a une mise à jour estampillée par W_i qui a été initialement acceptée par un serveur X, aura toutes les mises à jour acceptées par X avant la date W_i.

Cette contrainte de préfixe permet l'utilisation d'un vecteur de version pour représenter de manière compacte l'ensemble des écritures connues par un serveur. Le vecteur de version du serveur R comporte: l'identité des serveurs qui existe dans le journal des écritures de R et pour chaque serveur l'estampille la plus grande des mises à jour reçues par ce serveur.

a.2. caractéristique du protocole d'anti-entropie:[PST97]

Le protocole d'anti-entropie à faible cohérence de Bayou pour la propagation des mises à jour entre les copies est basé sur la communication par paires et la propagation des opérations d'écriture. Ce protocole fournit un bon support pour différents environnements de réseau et scénarios d'utilisation, en respectant l'ordre d'échange des opérations de mise à jour sauvegardées dans les journaux des copies. Il a aussi plusieurs avantages et fonctionnalités :

- **Supporte des topologies de communication arbitraire:** Le protocole assure la propagation des mises à jour entre n'importe quels paires de replicas quel que soit la topologie. En suite, la théorie d'épidémie assure que les mises à jour seront propagées par transitivité dans tout le système. [DGT87].

- **Fonctionnement en faible bande passante du réseau :** La réconciliation est basée sur l'échange de mise à jour non connues par le replica récepteur.

- **Une progression incrémentale:** Le protocole permet une progression incrémental même s'il est interrompu, par exemple: à cause d'une déconnexion de réseau involontaire

- **Une consistance sûre:** Chaque mise à jour atteint chaque replica, et les replicas ayant les mêmes mises à jour ont le même contenu de la base de données.

- **Une gestion dynamique des copies :** Le protocole supporte la création et la suppression d'une copie par une seule communication avec une copie disponible,

- **Choix de politique arbitraire:** N'importent qu'elles choix de politiques, sont supportées par le mécanisme d'anti-entropie. La politique a besoin d'assurer qu'il y a un chemin éventuel de communication entre n'importe qu'elle paires de replicas.

3.7.4 Gestion du journal des mises à jour :

a. Stabilité des mises à jour :

La cohérence des réplicas nécessite un ordre global des mises à jour pour qu'ils convergent vers le même contenu des copies. Une écriture est dite stable si sa position dans le journal est fixe et son ordre d'exécution ne changera pas. Bayou utilise un mécanisme de stabilisation des mises à jours qui nécessite de mettre un replica comme primaire (serveur primaire). Ce dernier valide la mise à jour en leur donnant un numéro séquentiel (CSN : **C**ommit **S**equence **N**umber) définissant un ordre globale dans le système et une position fixe dans le journal des mises à jours lors de la première réception de cette mise à jour.

Règle :

Une écriture A précède une écriture B si : CSN (A) > CSN (B)
Ou bien : *A et B ne sont pas stable, et ils étaient acceptées par le même serveur et A a été accepté avant B.*

Dans cet ordre les écritures stables sont toujours totalement ordonnées entre elles. Elles sont rangées dans le journal avant toutes les tentatives d'écriture selon l'ordre du CSN. Les écritures tentatives ont un CSN égale à l'infinie.

Le replica primaire est un serveur comme tous les autres, ayant le rôle de stabiliser les écritures et de leur attribuer le numéro de séquence global CSN.

b. Troncature du journal :

Le journal des mises à jour est la source principale de dépassement mémoire dans un système de réplication optimiste basé sur la journalisation des écritures. C'est particulièrement critique dans les environnements mobiles, où les ressources en mémoire sur les dispositifs mobiles sont en général rares. Par conséquent, une condition fondamentale est qu'un tel journal soit optimisé. Pour atteindre cet objectif, les serveurs doivent pouvoir supprimer les mises à jour inutiles de leurs journaux afin de limiter leur taille. Bayou permet à n'importe quel serveur de supprimer des mises à jour, quand il désire ou quand il est dans le besoin d'espace, à condition qu'elles soient stables.

3.7.5 Création et retirement de serveurs :

Bayou permet au nombre et l'emplacement des replicas de changer dans le temps. Tandis que l'emplacement et le moment de création ou de retirement sont gérés par l'application, le mécanisme de la création de nouveaux replicas et du retirement des anciens est approprié à Bayou.

Bayou permet de créer un nouveau replica par le clonage de n'importe quel replica disponible. Le gestionnaire de données du nouveau replica contacte un replica existant pour avoir le schéma de la base. Il crée une base de données locale, ensuite exécute la réconciliation avec un replica existant (peut être un autre que celui contacté en premier). Les informations sur le nouvel replica sont ensuite propagées en insérant une écriture de création « creation write », puisque l'anti-entropie assure la propagation, tous les autres réplicas vont savoir de l'existence du nouveau. De même le retirement d'un replica se fait en diffusant « un retirement write », le replica à retirer peut être supprimé après qu'il effectue au moins une réconciliation avec un autre.

3.7.6 La détection et la résolution des conflits :

Le faite que bayou utilise une stratégie de réplication optimiste les copies d'une même donnée étant partagées en lecture et en écriture par différents serveurs en même temps, par conséquence des conflits peuvent apparaître à la fonction de ces copies.

Pour détecter et résoudre ces conflits chaque écriture dans Bayou est composé de : la mise à jour, d'un contrôle de dépendance (*dependency check*) et d'une procédure de fusion (*merge procedure*). Le contrôle de dépendance permet de détecter les conflits c'est-à-dire qu'il permet aux utilisateurs de préciser, pour chaque opération d'écriture, comment le système doit détecter si la mise à jour résultante de cette opération pourra être en conflit avec la base de données. Si c'est le cas la procédure de fusion est exécutée pour le résoudre respectant la sémantique de l'application [BOU03a] [BAG95].

Il faut noter que le contrôle de dépendance et la procédure de fusion sont fournit par l'application ce qui rend le système de Bayou adaptable à une large gamme d'application.

3.7.7 Critique :

Le système Bayou offre:

- La disponibilité des données : grâce à son modèle (Read anay / Write anay).
- L'adaptabilité**.**
- L'Économie de la bande passante.
- Le Passage à l'échelle : Toutes les opérations dans Bayou telle que la lecture d'une donnée, l'écriture d'une donnée et la propagation des mises à jours entre les réplicas n'entraînent pas plus de deux réplica. Les données échangées dans une session d'anti-entropie entre deux serveurs est le vecteur de version plus les opérations d'écritures non connues par l'un des serveurs. La taille du journal des

replicas dépend de la fréquence des mises à jour de la base de données et de la fréquence des périodes d'anti-entropie.

Chaque replica maintient un vecteur de version, la taille du vecteur de version est relativement petite car il contient un identificateur unique et une estampillé [PST96].

Cependant :

- Bayou peut être utilisé avec les applications qui ne nécessitent pas une cohérence forte (stricte) des données, donc qui tolèrent une certaine divergence des copies, ce qui n'est pas permis dans une grande catégorie d'application.
- La communication entre les serveurs dans le protocole d'anti-entropie nécessite l'envoi du vecteur de version, si la taille de ce dernier augmente, le coût d'échange des vecteurs de version dominera l'opération d'anti-entropie.
- Si les opérations de mise à jour (écriture) augmentent plus vite que l'opération de stabilisation la taille du journal augmente ce qui influe sur les ressources système**.**
- L'application de la politique du choix arbitraire du serveur à réconcilier dans un réseau mobile Ad hoc entraînera la surcharge du réseau dans le cas ou les deux serveurs sont éloignés.

3.8 *Protocole de réplication GBR* [MOU07a] :

Les protocoles HBR, IBR, TRB et GBR proposé dans [MOU07a] sont basés sur une réplication en deux phases pour le partage des données importantes : une phase de réplication préventive, et une autre de réplication adaptative.

La première phase de réplication est une réplication *préventive* et primaire. Le but de cette étape est un déploiement optimal des copies (en nombre de copies et en temps d'accès). Seules les copies utiles sont effectuées et placées à des positions qui favorisent l'accessibilité à ces données. Cette phase est exécutée à l'arrivée d'une nouvelle donnée sur le réseau. C'est-à-dire, à la création ou au chargement de la donnée à partir d'une station fixe (en cas d'une éventuelle connexion avec la station fixe). L'algorithme de réplication dissémine la donnée sur tout le réseau par une distribution de ses copies. Celles ci sont réparties uniformément à une distance de ***k*** sauts les unes des autres [MOU06].

La réplication *préventive* a pour objectifs:

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

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

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

La deuxième phase de l'approche de réplication proposée est une réplication *adaptative*. Elle tente d'ajuster la localisation des copies aux changements dynamiques de la topologie et des besoins des utilisateurs [MOU08b]. Il s'agit de réaliser de nouvelles

copies ou de les remplacer certaines copies existantes par d'autres selon les changements des besoins et des demandes des utilisateurs.

L'algorithme de réplication *adaptative* considère:

(i) La fréquence des accès aux données. On distingue deux types de fréquences d'accès:

les fréquences d'accès internes (accès aux copies de données locales) et les fréquences d'accès externes (accès aux copies de données distantes).

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

(iii) Les temps de réponse aux requêtes générées.

La première méthode ***HBR*** proposée est une application des deux phases de réplication pour les données importantes. Deux autres méthodes ***IBR*** et ***TBR*** sont des améliorations de la première. Dans ***IBR***, la phase de réplication *préventive* est adaptée au degré d'importance de la donnée [MOU07e]. Une notion de degré d'importance est définie selon l'intérêt de la donnée pour les utilisateurs qui risque d'en avoir besoin (les annonces des ordres d'un supérieur sont des données importantes pour des troupes militaire en déplacement, par exemple). Pour les données pas très importantes, la dissémination seule des descriptions des données (MétaDonnées) est réalisée. Cette discrimination du niveau d'importance permet de réduire le coût de la réplication selon les contraintes de l'environnement.

TBR est une méthode qui prend un paramètre supplémentaire pour améliorer l'accessibilité: le temps d'accès aux données [MOU07d][MOU08a]. Dans la majorité des cas, les temps d'accès sont améliorés à travers la sélection de chemins d'accès courts. Cependant, un paquet de données peut traverser un chemin assez long, en un temps beaucoup plus rapide que le parcours d'un chemin plus court. L'encombrement du chemin et la surcharge en calcul des nœuds empruntés influencent les délais de propagation. Un flux d'information important peut retarder considérablement les communications.

La quatrième méthode ***GBR*** adopte la même approche. Mais, elle se distingue par une réplication sur une architecture logique en groupes de nœuds. Ce concept est très utilisé sur les réseaux mobiles ad hoc [Huang 06]. ***GBR*** utilise un algorithme simple de construction de groupes, non centralisé et complètement distribué. L'approche de réplication préventive et adaptative est alors adaptée à cette structure en groupes, afin d'en acquérir tous les avantages possibles. Aucune restriction sur le nombre de nœuds du réseau n'est obligatoire à priori. Le passage à l'échelle peut donc être envisagé et étudié. Cette méthode propose un protocole d'accès aux données mises à jour. Ce protocole est de type protocole à invalidation adapté aux réseaux mobiles ad hoc. Il traite, en particulier, la déconnexion du serveur primaire. La solution a été de prédire la déconnexion du primaire et de permettre son remplacement. Cette prédiction n'est pas étendue au cas de panne imprévisible du primaire.

3.9 Conclusion :

Dans ce chapitre nous avons vu quatre systèmes de partage de données sur réseaux mobile Ad hoc.

XMiddle demande l'utilisation d'un modèle de donnée spécifique (un arbre XML), ce qui limite son utilisation à des applications particulières. Cependant, l'utilisation d'informations structurées en XML est courante, et elle permet à XMiddle de fournir un algorithme de réconciliation applicable à toutes les données partagées. Même avec la journalisation des modifications, la réconciliation n'est donc pas forcément toujours possible. XMiddle laisse la réplication à la charge de l'utilisateur. Il n'offre donc aucune garantie de disponibilité sur la donnée en cas de partition. AdHocFS au contraire garantit la survivance d'une copie en cas de disparition. On ne sait cependant pas à quelles données est accordée la préférence s'il n'y a pas assez de terminaux pour tout conserver. Le modèle de réplication dans HaddockFS n'est pas suffisamment précisé.

Ces quatre systèmes proposent un modèle de cohérence hybride où certains terminaux travaillent en cohérence pessimiste alors que d'autres travaillent en cohérence optimiste. Le choix du modèle de cohérence n'est pas laissé au développeur. Il pourrait pourtant être parfois nécessaire pour certains documents d'appliquer un modèle de cohérence forte même entre différentes parties du réseau, alors que pour d'autres, la cohérence pourrait être optimiste au sein d'une même partie. La cohérence dans BlueFS repose sur deux travaux antérieurs. L'utilisation de Coda pour le contrôle de la réplication optimiste et l'utilisation de AdhocFS pour les rappels de service pour la cohérence du cache. Par ailleurs, malgré la mise en place d'une cohérence pessimiste, une réconciliation manuelle peut être nécessaire.

Dans ce chapitre, nous avons étudié aussi le système de réplication: Coda, le Protocole de réplication GBR, et le système de réplication Bayou afin de s'inspirer de ces solutions et pour pouvoir concevoir notre protocole de gestion de cohérence qui fera l'objet de prochain chapitre.

Chapitre 4 :

MRP un protocole de convergence de copies

4.1 Introduction :

Apres une étude des différents aspects de la réplication et de quelques systèmes de gestion de cohérence déjà existants, nous avons constaté que deux types de stratégies de réplication sont utilisés dans les systèmes actuels. Les stratégies pessimistes consistent à restreindre l'accès à une donnée lors des accès en écriture assurant une cohérence forte. La deuxième par contre, est une stratégie optimiste qui consiste à autoriser la manipulation concurrente et indépendante de la donnée répliquée même si le nœud n'est pas connecté au réseau. Ces protocoles favorisent la disponibilité sur la cohérence. Ils supposent que les partitionnements du réseau sont rares afin d'assurer la condition de convergence des copies [MOU07c][MOU07b].

Notre choix s'est porté sur une stratégie de réplication essentiellement optimiste même si une stratégie pessimiste est plus simple à mettre en œuvre. Cette dernière reste toujours inintéressante dans le cas des réseaux sans fil car elle limite considérablement la disponibilité de l'information et n'assure pas le passage à l'échelle.

Nous avons concentré nos travaux sur la façon de propager les mises à jour entre les réplicas. Cette technique a pour objectif de limiter le coût en messages des écritures sur les copies. Tout en assurant une mise à jour de toutes les copies d'une donnée dans les meilleurs délais possibles. Cette proposition est basée sur le principe d'épidémie des échanges de mises à jour [DGT87][MOU07a] propagées afin de réaliser cette objectif tout en prenant en considération les caractéristiques des réseaux mobiles Ad hoc.

Le protocole proposé est basé sur le principe de journalisation des écritures. Les principales réflexions de notre protocole de gestion de cohérence ont porté sur l'instant de déclenchement d'une phase de réconciliation de copies (propagation des mises à jour d'une copie vers les autres) et le choix des réplicas (copies) à réconcilier. Cette technique tient compte des caractéristiques des MANETs (Mobile Ad hoc Networks) et essaye d'en tirer profit afin de combler leurs contraintes.

4.2 Modèle d'environnement et hypothèses:

Un système de réplication est étroitement lié au type de l'application, auquel il est destiné surtout dans un environnement mobile, car un seul système ne peut résoudre tous les problèmes de ce type de réseaux (engendrés par la mobilité, la topologie dynamique, la taille,…etc.) et être adapté à toutes les situations. Pour ces raisons, nous devons définir un environnement applicatif où de travail auquel notre système sera adapté.

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

Dans notre travail, nous supposons que :

- A l'installation de l'application l'administrateur défini le serveur ***primaire*** (le plus performant et le plus atteinable).

- Notre système est destiné aux applications où les mises à jour (les écritures) nécessitent d'être appliquées dans un même ordre sur chaque copie [WIE90] [XAV98] [BAL02]. Ceci assure qu'à un moment donné, toutes les copies convergentes vers le même contenu. Cet ordre n'est pas forcément l'ordre réel. On ne s'intéresse pas à la méthode avec laquelle la création et le placement des copies sont faits [BAL94].On suppose que les données (copies) existent déjà au niveau des serveurs, et l'application des mises à jour est prise en charge par une autre couche du system de réplication.

- Nous supposons qu'il existe un mécanisme de "Roll back" [GAD08]. Il est utilise lorsque l'ordre d'acceptation est diffèrent de l'ordre global établi par le ***primaire***. Dans ce cas, il faut défaire les modifications ("Roll back") et les réexécutera dans l'ordre donné par le ***primaire***.

4.3 Le protocole MRP(Mobil Réconciliation Protocol):

Quand les utilisateurs accèdent en écriture à une même donné partagée, des divergences entre les copies sur les différentes serveurs peuvent se produire. Donc les copies doivent, à un moment donné, se resynchroniser. Pour se faire, nous proposons un protocole qui permet la convergence entre les copies dans un réseau mobile Ad hoc en s'inspirant de la solution existante du système Bayou [PST97] dédié aux réseaux statiques à grandes échelles.

Le but de protocole ***MRP*** est de mettre à jour un ou plusieurs serveurs à partir d'un autre serveur. Pour se faire, on propage les mises à jours vers les autres réplicas et non pas leur contenu qui peut être de taille importante. En effet, la transmission de données de taille importante conduirait à la surcharge du réseau. Ainsi, l'utilisation de la bande passante, dépend de l'activité des mises à jour et non pas de la taille entière de la donnée qui peut être importante [MOU07a].

Cette propagation se fait d'une manière épidémique, c'est-à-dire, que chaque serveur transmet aux serveurs qu'il réconcilie ses mises à jour et les mises à jour reçues à partir des autres serveurs à condition que ces mises à jour soient non connues par le(s) récepteur(s)

par souci d'optimisation. De plus chaque réconciliation est constituée de plusieurs phases. Une réconciliation est déclenchée périodiquement (suivant deux conditions qui seront décrites dans le paragraphe suivant)

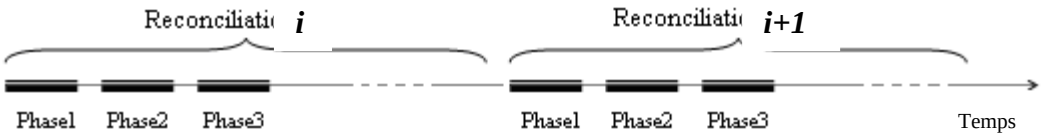


Figure 0.1 : périodes et phases de réconciliation

Une phase de réconciliation ***Pk*** pour un serveur S_i est une phase où S_i choisit le(s) serveur(s) qu'il réconcilie (suivant une politique qui sera détaillée dans la prochaine section) pour propager les mises à jours non connues par ces derniers.

Chaque serveur a un journal contenant les mises à jour reçues à partir d'un client ou d'un autre serveur et un vecteur de version qui comporte une description de l'ensemble des mises à jour connues par ce serveur ainsi que l'estampille correspondante [FID 91].

Quand un serveur reçoit une demande d'écriture, il lui affecte un ordre d'acceptation (**estampille d'acceptation**) et met à jour son vecteur de version. Le serveur ***primaire*** est le serveur qui détient la copie originale d'une donnée. Quand une mise à jour arrive au niveau du serveur ***primaire***, il lui affecte un ordre d'exécution unique sur tous les serveurs (***CSN* : C**ommit **S**equence **N**umber). Cet ordre unique permet de dater les évènements qui se produisent en parallèle sur les différents nœuds.

Pour déclencher le protocole ***MRP***, nous avons pensé initialement à lancer les réconciliations périodiquement, c'est-à-dire, qu'on fixe une période ***t***, durant laquelle le serveur reçoit des mises à jour à partir des clients. Quand la période est expirée, le serveur lance une réconciliation pour propager les mises à jour qu'il a stocké dans son journal. Cependant, cette politique pose un problème. Si la période est trop lente, le nombre de mises à jour reçues à partir des clients devient important. Ce qui engendre une grande divergence entre les copies. Et, le fait qu'on empêche la propagation de ces mises à jour aux autres réplicas durant cette période, fait que la taille du journal devient importante et ne peut pas être optimisé car ces mises à jour n'arrivent pas au serveur ***primaire*** qui doit les stabiliser et par conséquent, elles ne peuvent pas être supprimées car seules les mises à jour stables peuvent être supprimées. (L'optimisation de la taille de journal sera détaillée dans la section 4.7.). De plus la réconciliation n'est lancée qu'après l'expiration du délai même si le nombre de modifications est important, ce qui mène à une grande divergence entre les réplicas. De ce fait, nous avons essayé de fixer un seuil ***S*** de nombres de mises à jour, et à chaque fois qu'un serveur atteint ce seuil il lance la réconciliation. Cependant, il se peut que ce seuil ne soit atteint qu'après un délai important, ce qui augmente le temps de divergence entre les copies.

Nous utilisons, donc, une méthode hybride, où nous intégrons les deux conditions. Le premier est une fonction du temps et la deuxième est une fonction du nombre de modifications effectuées sur le réplica. Le principe est de fixer une période ***t*** et un seuil ***S***. Si la période, expire sans que le seuil ne soit encore atteint, le protocole lance, dans ce cas, une nouvelle réconciliation $\boldsymbol{rec_j}$. Dans le cas où, seul le seuil est atteint, notre protocole lance directement la réconciliation même si la période n'est pas terminée. Nous offrons un compromis entre la cohérence et le temps de réconciliation (voir figure 4.2).

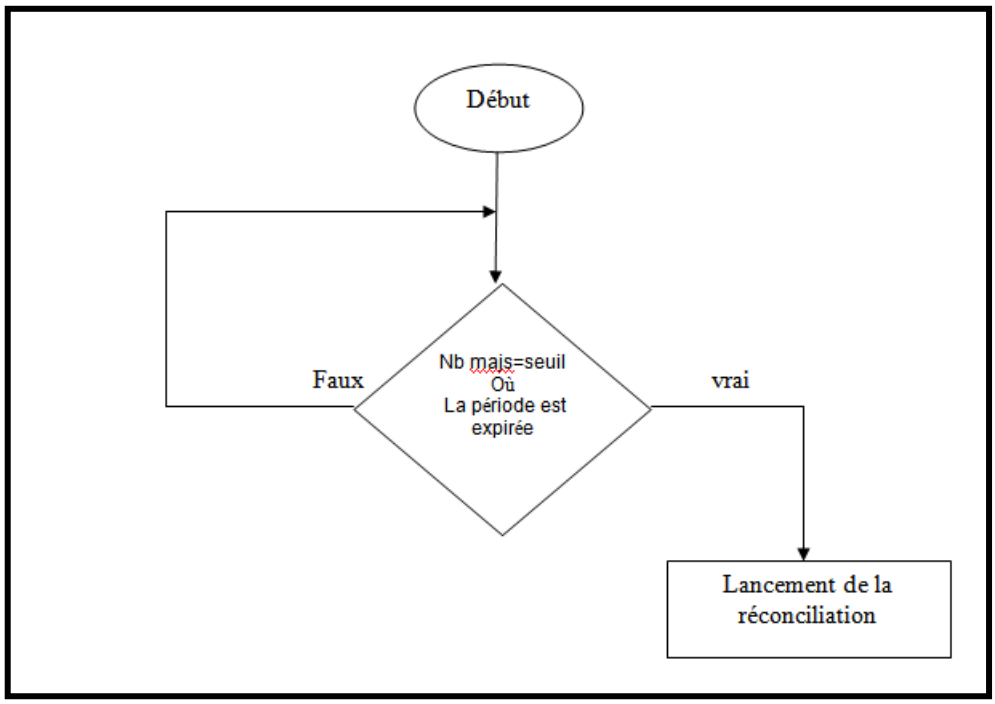


Figure 0.2 : Conditions de lancement de la réconciliation.

4.4 Election des serveurs à réconcilier :

Dans notre approche, à chaque phase de réconciliation $\boldsymbol{rec_j}$, le serveur $\boldsymbol{S_i}$ doit déterminer un ou plusieurs serveurs à réconcilier comme suit :

- D'abord, réconcilier d'éventuels serveurs voisins immédiat (à un hop) en bénéficiant du principe de diffusion dans les réseaux mobiles Ad hoc où un message est reçu par tous les nœuds qui se trouvent dans la portée d'onde radio même si certains ne sont pas destinataires.

- Puis, en ce qui concerne les serveurs qui se trouvent à plus d'un hop, la réconciliation se fait par pair en choisissant en premier les serveurs les plus proches (en terme de nombre de hop) à chaque phase de réconciliation. En effet l'application de la méthode de choix arbitraire de serveur à réconcilier

(méthode appliquée ans le système Bayou par exemple) serait très coûteuse et inappropriée dans un réseau mobile en mode Ad hoc car dans cette politique (de choix arbitraire), il se peut qu'on choisisse un serveur éloigné alors qu'il existe des serveurs proches (en terme de nombre de hop). Ces derniers pourront à leur tour propager les mises à jour à des serveurs plus éloignés par épidémie.

- Cependant, dans ce dernier cas, on risque d'avoir un problème de divergence de copies (une partie de réseau n'a pas les mises à jour d'une autre partie) à cause de la non sélection d'un ou de certains serveurs. D'où l'intérêt de marquer les serveurs avec lesquels nous avons fait une réconciliation. nous utilisons une table des serveurs au niveau de chaque nœud serveur. Cette table donne une description de chacun des serveurs connus par le nœud hôte. A la prochaine réconciliation nous choisissons le serveur le plus proche non marqué. Lorsque tous les serveurs de la table locale sont marqués, le bit de marquage est réinitialisé pour tous.

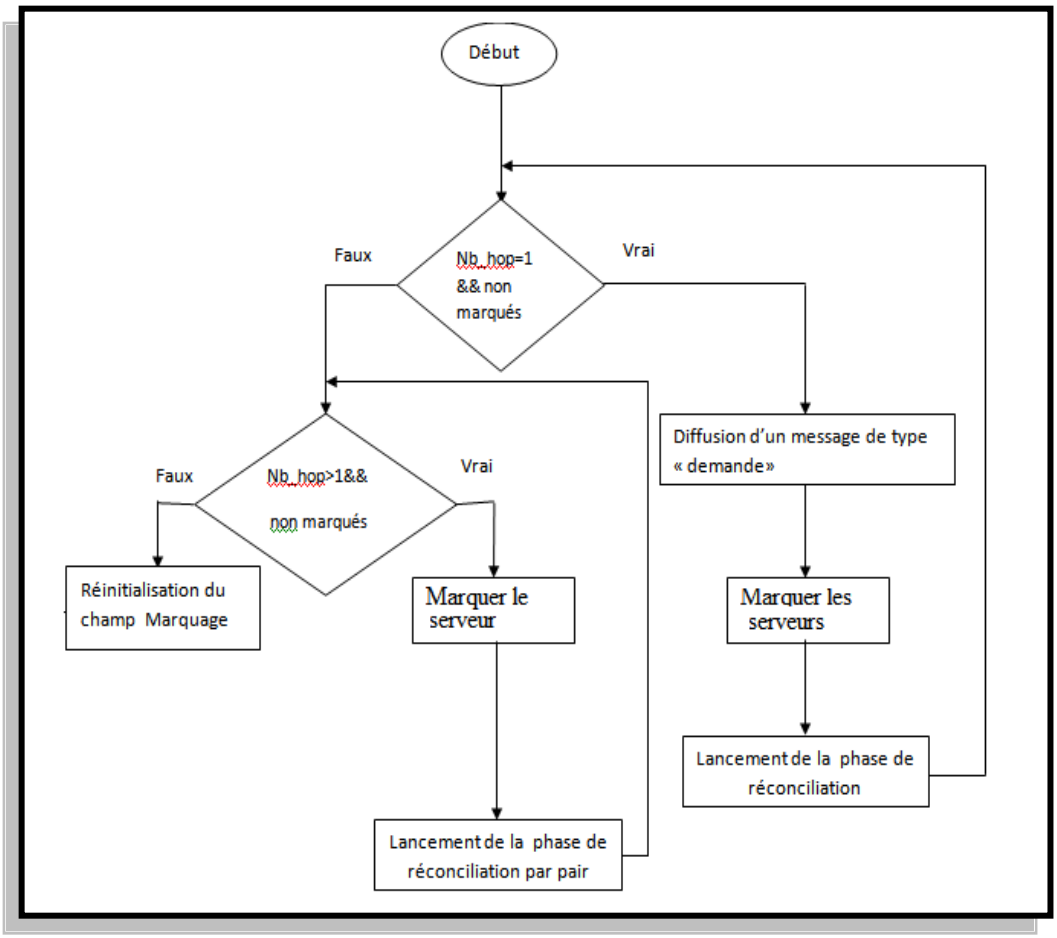


Figure 0.3 : Election des serveurs à réconcilier.

Exemple :

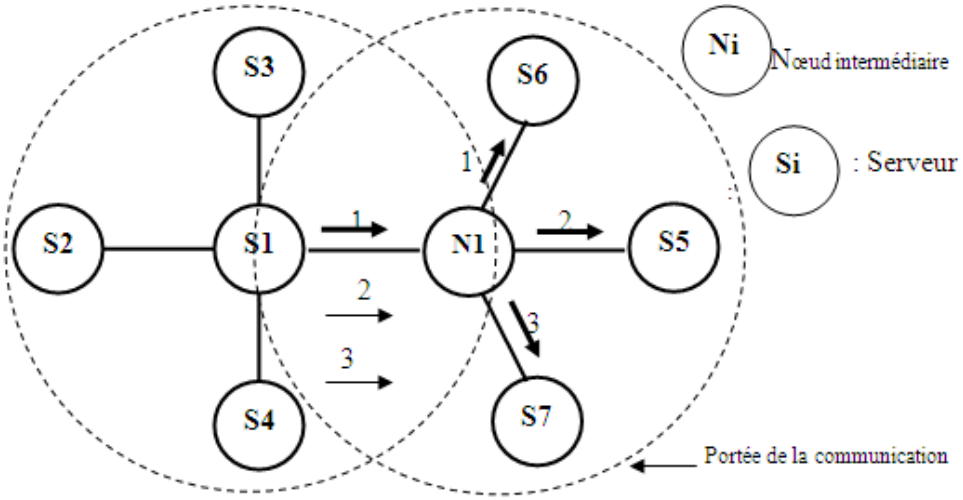


Figure 0.4 : Exemple de réconciliation par pair de serveurs à plus d'un hop

L'exemple de la figure 4.4 montre que

- Le serveur S_1 lance une phase de réconciliation avec les serveurs à un hop S_2, S_3, et S_4.
- En suite, il lance une deuxième phase de réconciliation (par pair) avec S_5, puis avec S_6 (à condition que dans le chemin il n y a pas de serveur, ce dernier assurerait alors cette tâche de réconciliation), et ainsi de suite c'est-à-dire pour que S_1 puisse réconcilier les serveurs S_2, S_3, S_4, S_5, S_6, S_7 il doit lancer quatre phases de réconciliation.

Notons que la réconciliation est épidémique, c'est à dire que chaque serveur transmet aux serveurs qu'il réconcilie ses mises à jour et les mises à jour reçues à partir des autres serveurs à condition que ces mises à jour soient non connues par le(s) récepteur(s) par souci d'optimisation. L'exemple suivant illustre ce principe :

Exemple

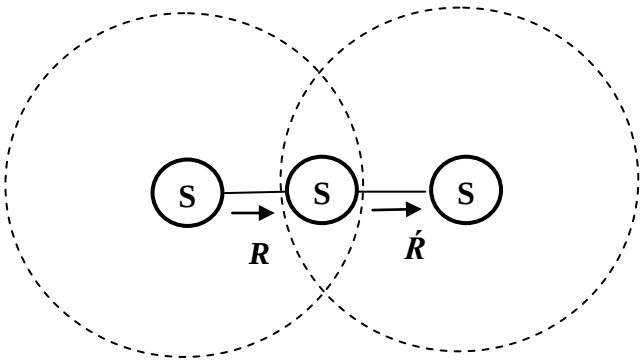


Figure 0.5 : la réconciliation épidémique

Supposons que :

S_i a les mises à jour $\mathbf{m_{i.0}}$, $\mathbf{m_{i.1}}$, $\mathbf{m_{i.2}}$.

S_j a les mises à jour $\mathbf{m_{j.0}}$, $\mathbf{m_{j.1}}$. $\mathbf{m_{i.0}}$

S_k n'a aucune mise à jour.

- Quand S_i entre dans une phase de réconciliation R avec S_j, il transmet au serveur S_j les mises à jour $\mathbf{m_{i.1}}$, $\mathbf{m_{i.2}}$ (les mises à jour non connues par S_j).
- Quand S_j entre dans une phase de réconciliation $\acute{R}$ avec S_k, il transmet au serveur S_k les mises à jour $\mathbf{m_{j.0}}$, $\mathbf{m_{j.1}}$. $\mathbf{m_{i.0}}$ $\mathbf{m_{i.1}}$, $\mathbf{m_{i.2}}$. (Ces mises à jour sont non connues par le serveur S_k)

S_i ne réconcilie pas le serveur S_k car il existe un serveur S_j sur le chemin qui relie S_i à S_k (suivant le principe de notre algorithme de réconciliation qui sera détaillé dans le paragraphe suivant) et par conséquent les mises à jour de S_i vont êtres propagées au serveur S_k par épidémie.

4.5 Principe de l'algorithme de réconciliation de protocole MRP :

Quand un serveur S_i veut lancer une réconciliation (dans le cas où le seuil est atteint ou la période est expirée), il procède de la manière suivante :

- Il diffuse un message de type « demande » qui signifie que le serveur demande à initier une phase de réconciliation. Ce message est reçu par tous les nœuds qui se trouvent dans la portée d'onde radio (à 1 hop) de S_i.

- Il attend un délai limité (ce délai est calculable en fonction de la portée d'onde radio c'est-à-dire qu'on attend un délai supérieur à $2x$ au moins tel que x est le temps nécessaire pour que le message arrive au serveur destinataire) pour qu'il puisse recevoir les vecteurs de version à partir des serveurs qui ont reçu le message « demande ». Ce délai peut être estimé à partir de la qualité du signal de réception, de la position du serveur et de la taille du message. Il peut aussi être évalué par expérimentation (nous faisons des tests pour estimer une valeur appropriée à l'environnement de développement).

- A la réception des vecteurs de version, le serveur S_i :

  * Met à jour sa table des serveurs (sera détaillée dans la prochaine section) plus précisément le champ marquage pour chaque entrée ID_SER_i tel que ID_SER_i est le serveur à réconcilier

  * Parcourt les opérations d'écritures dans son journal et vérifie pour chacune d'elles :

Si l'écriture ou la mise à jour n'est pas connue par l'un des serveurs à réconcilier, la mise à jour est diffusée.

Concernant les serveurs à plus d'un hop la réconciliation est faite comme suit :

- S_i parcourt sa table des serveurs
- Il cherche les serveurs **non marqués** qui ont ***nb_hop***>1.
- Parmi ces serveurs, il vérifie pour chaque serveur S_j s'il existe un serveur intermédiaire dans le chemin joignant S_i à S_j
- Dans le cas où ce serveur intermédiaire existe, la phase de réconciliation ne sera pas déclenchée et les mises à jour envoyées par S_i sont propagées par épidémie au serveur S_j afin de diminuer le nombre de phase de réconciliation ainsi que le trafic réseau engendré.
- Dans le cas où le chemin qui relie S_i à S_j est constitué que de clients, S_i lance la phase de réconciliation par couple avec S_j en procédant de la manière suivante :
    - S_i envoi un message de type « demande » en parallèle (à tous les serveurs qui ont ***nb_hop***>1 et dont le chemin qui les connectent à S_i ne contient pas un serveur) sans attendre un délai.
    - Le serveur S_j dont le vecteur de version a été reçu par le serveur S_i sera choisi par le serveur S_i pour une phase de réconciliation.
- A partir du vecteur de version reçu, S_i vérifie dans sont journal des mises à jour toutes les mises à jours que S_j n'a pas et les envoie à ***Sj***.

4.6 Etat d'un serveur :

L'état d'un serveur est déterminé par un ensemble de structures de données :

4.6.1 La table des serveurs :

Cette table contient les informations que détient un serveur concernant les serveurs existants dans le système (c'est-à-dire dont le serveur a pris connaissance, y compris le (s) serveur (s) déconnecté (s)), sa structure est illustrée par le tableau suivant :

ID_SER	NBR_HOP	Marquage	Next	Date	Serv/Clien
ID_SER1	entier	(1/0)	Nœud X		
ID_SER2	entier	(1/0)	Nœud Y		
:	:	:	:		
ID_SERi	N	(1/0)	Nœud k		

Tableau 0.1 : table des serveurs

***ID_SER* :** identificateur de serveur qui doit être unique pour chaque serveur dans le réseau, l'utilisation de l'identificateur de la carte réseau installée au niveau d'un nœud mobile pourrait assurer cette unicité.

NBR_HOP : c'est un entier qui indique deux informations en même temps.
Si ce champ est à zéro alors le serveur ID_SER_i est déconnecté,
Sinon (si ***NBR_HOP***>0) alors le serveur ID_SER_i est connecté, et le nombre de sauts qui le séparent du serveur est ***NBR_HOPi***. Cette information est délivrée par la couche routage

Marquage* :** ce champ peut prendre seulement deux valeurs "1''*** ou "***0''***. Si ce champ est égal à "***1***'' cela indique au serveur qui possède cette table que le serveur ayant ID_SER_i a été déjà sélectionné pour une phase de réconciliation sinon (si ce champ est égal à ***0***) ce serveur n'a pas été encore choisit pour une phase de réconciliation.
Lorsque tous les serveurs de la table locale sont marqués, ce champ est réinitialisé pour tous. Ce mécanisme permet d'éviter de négliger certains serveurs tout en privilégiant les serveurs les plus proches.
Un serveur qui commence une période de réconciliation, réconcilie les serveurs à un hop en premier par groupe et par diffusion et ensuite il réconcilie les serveurs à plus d'un hop par pair.

Next* :** Il indique le prochain nœud voisin sur le chemin vers le serveur ***ID_SER. Cette information est aussi délivrée par la couche routage.
S'il existe plusieurs chemins qui mènent à un même serveur ***S*** nous ne maintenons qu'un seul (le choix du chemin sera détaillé dans le paragraphe « mise à jour de la table des serveurs»). Ainsi, nous réservons une seule entrée dans la table des serveurs pour le serveur S à travers l'un des nœuds intermédiaires.

Date* :** Il indique l'instant a partir duquel le nœud ***ID_SER a été détecté à l'état déconnecté (c.à.d. ***NB_HOP*** =0). Ce champ est initialisé à 0. Il est mis à jour par le serveur S_i quand il reçoit un message découverte du voisinage à partir de nœud ***ID_SER***). Ce champ est utilisé pour la troncature du journal des mises à jour. La troncature consiste à éliminer les opérations qui ne sont plus nécessaires à la sauvegarde c.à.d. qui sont stabilisées.

Serv/Client : il indique au serveur qui possède cette table que dans le chemin qui le relie avec l'entrée ID_SER_i il existe un serveur ou pas. Ce champ est initialisé à la valeur ***"client*** ".

Exemple : à partir de la figure 4.6 nous avons représenté la table des serveurs correspondante au serveur S_1 avant et après la réconciliation.

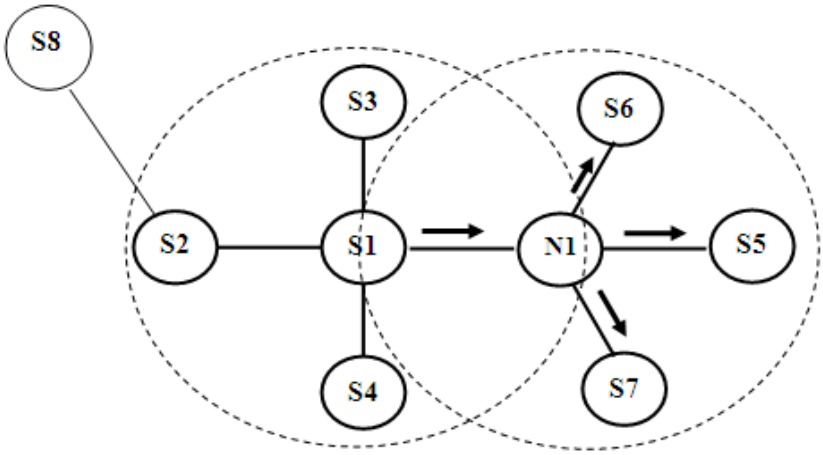


Figure 0.6 : Exemple de mise à jour de la Table des Serveurs

ID_SER	NBR_HOP	Marquage	Next	Date	Serv/client
S2	1	0	S2		serveur
S3	1	0	S3		serveur
S4	1	0	S4		serveur
S5	2	0	N1		client
S6	2	0	N1		client
S7	2	0	N1		client
S8	2	0	S2		serveur

Table de S_1 avant les phases de réconciliation

ID_SER	NBR_HOP	Marquage	Next	Date	Serv/client
S2	1	1	S2		serveur
S3	1	1	S3		serveur
S4	1	1	S4		serveur
S5	2	0	N1		client
S6	2	0	N1		client
S7	2	0	N1		client
S8	2	0	S2		serveur

Table de S_1 après la 1ère phase de réconciliation

ID_SER	NBR_HOP	Marquage	Next	Date	Serv/client
S2	1	1	S2		serveur
S3	1	1	S3		serveur
S4	1	1	S4		serveur
S5	2	1	N1		client
S6	2	1	N1		client
S7	2	1	N1		client
S8	2	1	S2		serveur

Table de S_1 après les trois phases de réconciliation parallèle de S_5, S_6 et S_7

Le serveur S_1 ne lance pas une phase de réconciliation avec le serveur S_8 (car il existe un serveur S_2 intermédiaire sur le chemin,). Donc les mises à jour du serveur S_1 vont être propagées par le serveur S_2 quand il lance une phase de réconciliation avec S_8 et cela grâce à l'épidémie

Remarque :
L'algorithme correspondant au choix du chemin optimal sera décrit dans le prochain paragraphe.

- **La mise à jour de la table des serveurs :**

Vu les caractéristiques des réseaux mobiles en mode Ad hoc tel que la déconnexion et le changement fréquent de la topologie, il est nécessaire de prendre en considération les variations de la topologie dynamique pour maintenir la table de serveur à jour. Pour cela, nous avons besoin initialement d'une phase d'initialisation pour la construction des tables des serveurs de tous les nœuds du système. Dans cette phase, chaque nœud envoie un message de type « découverte du voisinage » qui a la structure suivante :

ID_SERi	***nb_hop***	***Next***	***Serv/client***

Chaque nœud qui reçoit ce message, insère le paquet reçu dans la table des serveurs du récepteur après l'incrémentation du nombre de hop. De plus, si le nœud récepteur est un serveur alors il met à jour le champ ***Serv/client*** correspondant à l'entrée ***ID_SERi*** par la valeur «***serveur*** » sinon, il laisse ce champ à « ***client*** ». A partir de ces messages de découverte, on construit sur chaque nœud S_i la table des serveurs connus du nœud.

A l'expiration de la période d'initialisation, les nœuds dont on ne connaît pas la position (déconnectés ou bien ceux desquels on n'a pas encore reçu de message découverte de voisinage) leur champ ***nb_hop*** prend la valeur 0 et le champ date est initialisé à la date actuelle.

Le premier nœud qui reçoit ce message (***nb_hop***=0), met à jour le champ ***next***.

En suivant cette procédure de construction de la table des serveurs, nous aurons les informations concernant tous les nœuds du système et tous les chemins possibles qui mènent à chaque nœud comme le montre l'exemple suivant :

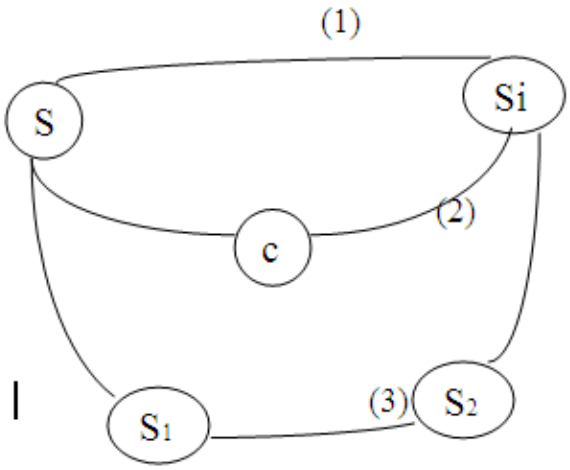


Figure 0.7 : Sélection du chemin optimale

Pour accéder à S_i nous avons trois chemins possibles :

Le premier à travers le serveur i (qui est à un hop par rapport à S_i).

Le deuxième chemin à travers le client ***C*** (qui est à deux hop par rapport S_i).

Le troisième chemin à travers serveur S_1 (qui est à trois hop par rapport S_i).

ID_SER	NBR_HOP	Marquage	Next	Date	Serv/client
S_i	1	0	S_i		Serveur
S_i	2	0	***C***		Client
S_i	3	0	S_1		serveur

Tableau 0.2 : liste des chemins possibles vers S_i

D'où pour construire la table des serveurs qui contient une seule entrée pour chaque serveur et qui est l'entrée optimale, on procède comme suit :

- A la réception d'un message ***découverte*** par un serveur S_i, il compare son contenu (le nombre de hop) avec l'entrée correspondante au même ***ID_SER*** dans la table des serveurs et nous sauvegardons le chemin minimal.

Dans cet exemple, nous prenons le premier chemin (numero1), c'est le chemin le plus optimal.

Supposons maintenant que nous avons le cas suivant :

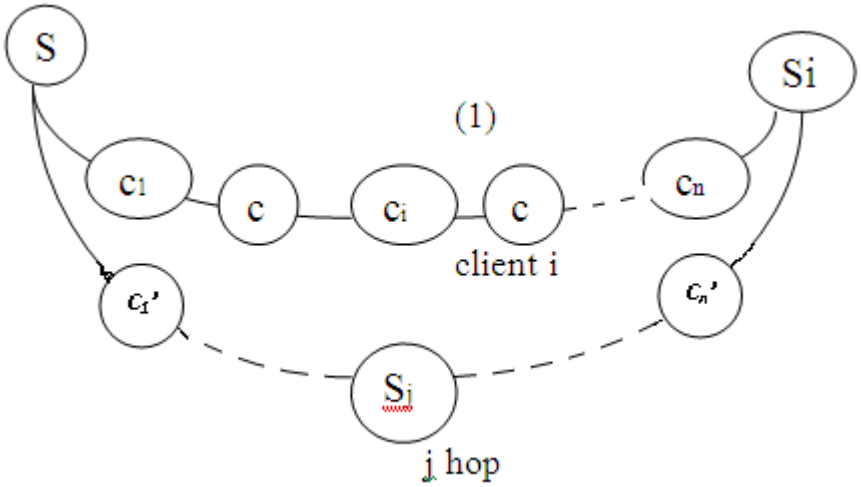


Figure 0.8 : L'influence du nombre de hop sur le choix du chemin

Sur la figure 4.8, nous avons la représentation des deux chemins reliant les nœuds ***S*** et S_i :

-Le premier chemin à n hop comportant que des clients

-Le deuxième chemin à (n'+j)hops >n hops mais il contient au moins un serveur S_j à j hop.

La table des serveurs de ***S*** avant l'optimisation est la suivante :

ID_SER	NBR_HOP	Marquage	Next	Date	Serv/client
S_i	n	0	C_1		client
S_i	n'+j	0	C_1'		Serveur S_j

Tableau 0.3 : La table des serveurs de *S*.

Pour un nombre n de hops assez grande, et avec (n'+j)hops >n hops, ***S*** va choisir le premier chemin (toujours en procédant de la même manière pour choisir le chemin). Comme les deux serveurs sont distants le temps de divergence peut être rallongé.

Pour diminuer le temps de divergence, ***S*** devrait choisir le deuxième chemin qui contient un serveur à j hops. Ce serveur (S_j) sera choisi par le serveur ***S*** pour être réconcilie. Après cette réconciliation et quand S_j rentre dans une phase de réconciliation avec Si (qui est à n'+j hop), il transmet ses mises à jours et les mises à jour reçues à partir des autres serveurs (y compris celle de ***S***) grâce au principe de propagation épidémique.

A partir des remarques déduites de l'exemple précèdent, nous pouvons donner l'algorithme de choix de chemin optimale en suivant la démarche suivante :

- ✓ Si le nombre hop minimum (nb_hop_min) correspond au chemin constitué uniquement de clients et si on est dans le cas ou nb_hop_min est très grand nous fixons un certain seuil pour nb hop min qu'il ne faut pas dépasser. Car, si le chemin minimal est très grand en termes de nombre de hops le temps de divergence peut être augmenté.

- ✓ Si nb_hop_min dépasse le seuil, nous choisissons une autre entrée dans la table des serveurs qui correspond au chemin qui a un nb_hop_min supérieur au nb_hop_min précèdent et qui contient au moins un serveur sur le chemin. Nous procédons de la même manière que pour le cas précédent.

- ✓ A la fin de l'exécution de cet algorithme, la table des serveurs contient pour chaque serveur S_i une seule entrée qui est l'entrée optimale (les autres entrées sont éliminées).

Il faut noter que les messages découverte (message Hello) du voisinage sont périodiques. Dans le cas où un serveur émetteur n'existe pas dans la table du serveur récepteur, ce dernier l'ajoutera directement à sa table des serveurs et un nouveau serveur sera donc connu implicitement.

- Mise à jour des structures :

La taille de la table des serveurs peut ainsi augmenter considérablement et nous risquons d'avoir des entrées inutilisées dans la table des serveurs dont les nœuds associés ne sont plus des serveurs (dans le cas où le serveur n'a plus la donnée ou si ça donnée n'est plus valide et il ne peut plus l'utiliser, donc ses serveurs deviennent uniquement clients). Pour palier à ce problème, nous fixons une période ***t*** (multiple de la période de réconciliation), quand cette période est atteinte, nous déclenchons la phase de nettoyage de la table des serveurs afin d'optimiser sa taille et par conséquence elle devient dynamique.

La phase de nettoyage de la table des serveurs se fait en suivant les étapes suivantes :

1. A la réception d'un message ***découverte*** à partir d'un client par un serveur ***S***, il parcourt sa table des serveurs,
2. s'il trouve l'identificateur de ce client, cela signifie que ce client était un serveur et il ne l'est plus maintenant (car toutes les entrées de la table des serveurs sont des serveurs) et par conséquence, il faut supprimer cette entrée de la table des serveurs.

Le choix d'une période assez large pour déclencher la phase du nettoyage revient au fait que le changement d'état de serveur vers client est relativement rare, d'autre part si la phase du nettoyage est lancée à chaque réception de message découverte, la table des serveurs est parcourue plusieurs fois, ce qui en résulte beaucoup de traitements qui peuvent être inutiles.

4.6.2 Le journal de mises à jour :

Du moment que nous considérons un système de réplication optimiste, nous permettons aux réplicas d'accepter les mises à jour indépendamment les uns des autres, puis ces réplicas rentrent dans des phases de réconciliation dans le but de converger leurs contenus d'où la nécessité d'avoir des mécanismes qui permettent de récupérer les différents changements effectués sur chaque réplica. Il s'agit du journal des mises à jour. Ce dernier contient l'historique des mises à jour effectuées sur chaque réplica. Au niveau de chaque serveur, nous avons un historique.

Chaque mise à jour du journal est représentée par la structure de données contenant les éléments :

Text : la mise à jour elle-même, c'est-à-dire, la description de l'opération de mise à jour effectuée.

ID_SER : l'identificateur du serveur à l'origine de cette Maj.

Stamp : l'estampille d'acceptation affectée par un serveur lors de l'arrivée d'une mise à jour à partir d'un client. Cette estampille est affectée par ce serveur juste lorsque cette mise à jour est acceptée par lui et elle (c.à.d. la valeur d'estampille) sert à indiquer l'ordre dans lequel la mise à jour a été effectuée par le serveur.

CSN : numéro affecté par le serveur ***primaire*** à une mise à jour lorsque cette dernière arrive au niveau du serveur ***primaire*** qui va donner un ordre d'exécution unique sur tous les serveurs (CSN). Tant que la mise à jour n'est pas encore arrivée au serveur ***primaire***, ce champ est égal à l'infini.

❖ **Structure du journal des mises à jour :**

Le journal des mises à jour est décomposé en deux parties, la partie stable et la partie tentative ou non encore stable.

La partie stable est constituée des mises à jour qui sont validées par le serveur ***primaire*** en leurs donnant un numéro séquentiel qui est le ***CSN*** définissant un ordre global dans le système et une position fixe dans le journal pour la mise à jour.

En rappelant que les écritures ou les mises à jours stables sont toujours totalement ordonnées entre elles et elles sont rangées dans le journal avant toute mise à jour tentatives (***CSN***=∞) selon l'ordre du ***CSN.***

La partie tentative est constituée des Majs acceptées par les différents serveurs mais qui ne sont pas encore stabilisées par un ***CSN*** attribué par le serveur ***primaire*** en les recevant. Cela signifie que leurs ***CSN*** sont égaux à l'infinie.

L'exemple suivant montre la structure du journal des mises à jour :

Partie Stable du journal	$TEXT_1$	ID_SER1	Stamp1	CSN1
	$TEXT_2$	ID_SER2	Stamp2	CSN2
	$TEXT_3$	ID_SER3	Stamp3	CSN3
	:	:	:	:
	$TEXT_n$	ID_SER_n	$Stamp_n$	CSNn
Partie Tentative du journal	$TEXT_{n+1}$	ID_SER_{n+1}	$Stamp_{n+1}$	∞
	$TEXT_{n+2}$	ID_SER_{n+2}	$Stamp_{n+2}$	∞
	$TEXT_{n+3}$	ID_SER_{n+3}	$Stamp_{n+3}$	∞
	:	:	:	:
	$TEXT_{n+k}$	ID_SER_{n+k}	$Stamp_{n+k}$	∞

Figure 0.9 : Structure de journal des mises à jour

Les écritures tentatives sont ordonnées selon la règle suivante :

Au niveau d'un serveur, une écriture A précède une écriture B si les deux sont acceptées par le serveur, et A a été acceptée avant B.

4.6.3 Le vecteur de version:

Chaque serveur possède un vecteur de version qui contient pour chacun des autres serveurs la plus grande valeur d'horloge d'estampillage qu'il a reçu et la plus grande valeur de ***CSN*** connu. Ce vecteur est mis à jour au cours des phases de réconciliation et il ne peut être utilisé que lorsqu'on respecte la propriété préfixe défini par Bayou [PST97] qui peut être exprimée ainsi :

Un serveur qui a une Maj estampillée par Wi qui a été initialement acceptée par un serveur X, aura toutes les Majs acceptés par X avant la date Wi.

Exemple : supposons qu'un serveur ***S*** à le journal suivant **:**

TEXT	ID_SERVEUR	STAMP	CSN
$TEXT_{1.1}$	SERV1	1	CSN1
$TEXT_{1.2}$	SERV1	2	CSN2
$TEXT_{2.1}$	SERV2	1	CSN3
$TEXT_{1.3}$	SERV1	3	CSN4
$TEXT_{3.1}$	SERV3	1	CSN5
$TEXT_{2.2}$	SERV2	2	∞
$TEXT_{3..2}$	SERV3	2	∞
$TEXT_{5.1}$	SERV5	1	∞
$TEXT_{3.3}$	SERV3	3	∞

Figure 0.10 : journal du serveur *S*

Le vecteur de version correspondant à ce journal est le suivant

ID_Serveur	CSN MAX	Plus grande estampille
SERV1	CSN 4	**3**
SERV2	CSN3	**2**
SERV3	CSN5	**3**
SERV5	∞	**1**

← ***S*** connaît toutes les Majs de SERV1 qui ont une estampille inférieure ou égale à 3

Figure 0.11 : Vecteur de version du serveur *S*.

- **Utilité d'un vecteur de version :**

En observant le vecteur de version (le tableau ci-dessus) on conclut que dans une phase de réconciliation un serveur S_i (qui a lancé la phase de réconciliation) peut connaître facilement les mises à jour non connues par un serveur S_j (celui à réconcilier) en faisant l'extraction des Majs non couverte par (S_j .V). Cela est illustré dans l'exemple suivant :

Exemple :

Soient ***S*** et ***S'*** deux serveurs qui ont les journaux ***S.J***, ***S'.J*** et les vecteurs de version ***S.V***, ***S'.V*** respectivement (voir les tableaux suivants).

-Quand ***S*** lance une réconciliation avec ***S'***.

- il (***S***) envoi un message de type demande de réconciliation à ***S'***.

- ***S'*** envoie son vecteur de version à ***S***.

- ***S*** détermine alors que ***S'*** n'a pas les Majs $TEXT_{1.2}$ $TEXT_{1.3}$ $TEXT_{2.1}$ $TEXT_{4.2}$, et ***S'*** ne connaît pas que la Maj $TEXT_{3.1}$ a été stabilisée par le serveur ***primaire***.

- ***S*** envoie alors les Majs $TEXT_{1.2}$ $TEXT_{1.3}$ $TEXT_{2.1}$ $TEXT_{4.2}$ et une notification de stabilisation de la Maj $TEXT_{3.1}$ à ***S'***.

Journal du serveur S (S. J)			
TEXT	ID_SERVEUR	STAMP	CSN
$TEXT_{1.1}$	SERV1	1	1
$TEXT_{1.2}$	SERV1	2	2
$TEXT_{3.1}$	SERV3	1	3
$TEXT_{1.3}$	SERV1	3	4
$TEXT_{4.1}$	SERV4	1	∞
$TEXT_{2.1}$	SERV2	1	∞
$TEXT_{4.2}$	SERV4	2	∞

Vecteur de version du serveur S (S.V)		
SERV1	4	3
SERV2	∞	1
SERV3	3	1
SERV4	∞	2

journal du serveur (S'. J)			
TEXT	ID_SERVEUR	STAMP	CSN
$TEXT_{1.1}$	SERV1	1	1
$TEXT_{3.1}$	SERV3	1	∞
$TEXT_{4.1}$	SERV4	1	∞
$TEXT_{5.1}$	SERV5	1	∞

Vecteur de version du serveur S' (S'.V)		
SERV1	1	1
SERV3	∞	1
SERV4	∞	1
SERV5	∞	1

⇩ ← **Après Réconciliation** → ⇩

Journal du serveur S' (S' J)			
TEXT	ID_SERVEUR	STAMP	CSN
$TEXT_{1.1}$	SERV1	1	1
$TEXT_{1.2}$	**SERV1**	**2**	**2**
$TEXT_{3.1}$	**SERV3**	**1**	**3**
$TEXT_{1.3}$	**SERV1**	**3**	**4**
$TEXT_{2.1}$	**SERV2**	**1**	∞
$TEXT_{4.1}$	SERV4	1	∞
$TEXT_{4.2}$	**SERV4**	**2**	∞
$TEXT_{5.1}$	SERV5	1	∞

Vecteur de version du serveur S' (S'.V)

SERV1	4	3
SERV2	∞	1
SERV3	3	1
SERV4	∞	2
SERV5	∞	1

4.7 Optimisation de la taille du journal :

Le protocole de réconciliation que nous avons proposé est basé sur la journalisation des mises à jour dans un système de réplication optimiste. Pour se faire, chaque serveur a un journal des mises à jour dans lequel il garde l'historique de chaque mise à jour reçue (comme on a déjà vu précédemment). Cependant ce journal de mises à jour est la source principale de dépassement de capacité de la mémoire surtout dans les environnements mobiles Ad hoc où les ressources mémoire des dispositifs mobiles sont généralement limitées. D'où la nécessité d'optimiser la taille du journal. Ceci est d'autant plus important, qu'il faut tenir compte des capacités de stockages des sites mobiles. Pour cela nous avons pensé au début de notre conception à supprimer la partie stable du journal de mises à jour. Mais dans ce cas nous pouvons avoir un problème qui est le suivant :

Par exemple, un nœud déconnecté qui rejoint le réseau, ne peut jamais récupérer les mises à jour supprimées (cas où ce nœud a besoin des mises à jour supprimées qui n'ont pas été récupérés par lui). Par conséquence, le contenu des copies existantes du système ne pourra jamais converger vers un même état.

Pour cette raison nous avons éliminé cette idée et nous avons considéré la possibilité de faire une troncature du journal des mises à jour suivant la démarche suivante :

- Réception du plus grand ***CSN*** de chaque serveur (par un serveur S_i qui veut faire une troncature) : Lorsque le serveur S_i lance une phase de réconciliation, il récupère ces ***CSNs*** à partir du vecteur de version associé à chaque serveur.

- Parmi tous ces ***CSN***s reçus, le minimum «min (max (***CSNi***)) » est choisi.
- Supprimer les mises à jours qui ont un ***CSN***<=min et à chaque fois, on garde la date de troncature : si un nœud était déconnecté et pendant cette période de déconnexion, le serveur S_i a fait une troncature du journal**,** il prend les mises à jours de ce nœud et les exécute puis il lui envoie toute la copie.

Cette démarche de troncature permet d'optimiser la taille d'un journal. Cependant dans le cas où un nœud était déconnecté et la taille de son journal a atteint le seuil (Ce seuil est l'espace mémoire maximum que peut prendre le journal.), alors que toutes les mises à jour qu'il a dans son journal sont tentatives, on a un problème de débordement.

Car ce journal, contient que des mises à jours tentatives ce qui empêche de faire une troncature du journal qui ne peut concerner que la partie stable du journal.

4.8 Algorithmes de convergence des copies :

4.8.1 Principe général :

Chaque serveur ***S*** accepte les mises à jour indépendamment des autres serveurs. A chaque mise à jour sollicitée par un client, ***S*** lui affecte une estampille d'acceptation, son ***ID*** et met son ***CSN*** à l'infinie puis il la range dans son journal.

A l'expiration de la période de réconciliation ou si le nombre des mises à jour acceptées par ***S*** après la dernière phase de réconciliation est égal au seuil, ***S*** choisit le(s) serveur(s) les plus proche(s) $\boldsymbol{R_i}$ et lance l'algorithme de réconciliation.

S envoie les Majs non connues par les serveurs $\boldsymbol{R_i}$ ayant été choisis.

- **Structures de données** :

- ***CSN*** : Horloge logique # Plus grande estampille des Majs stables vue par un serveur ***S***.
- ***Journal*** # Contient les Majs acceptées par ***S*** et éventuellement par d'autres serveurs.
- ***S.V*** : Tableau [1.. N]. # C'est le vecteur de version du serveur ***S***, avec N est le nombre des serveurs connu par ***S***. Chaque entrée est de la forme : ***ID*** : ***Stamp***, avec ***ID*** est l'identificateur du serveur créateur et ***Stamp*** est la plus grande estampille des mises à jour générées par *ID* et vue par ***S***.
- ***Stamp*** : Horloge.# Horloge logique locale de ***S***, elle est incrémentée par le serveur ***S*** lors de l'arrivée d'une mise à jour à partir d'un client.

4.8.2 Réconciliation à un hop:

- **Algorithme de base**

Réconciliation (S, R_1, R_2, ...R_n) :
{ # n est le nombre de serveur à mettre à jour.
Var :
E : Ensemble des serveurs initialisé à vide
Récupérer R_1.V, R_2.V, ... Rn.V Depuis R_1, R_2, ...Rn

```
            # Ri.V est le vecteur de version du serveur Ri.
W= Première écriture dans S.Journal


TANQUE (W) FAIRE
        POUR chaque Ri  FAIRE
            SI W.STAMP > Ri.V (W.ID_SERVEUR)
              ALORS              # Ri ne connaît pas W .
              E= E + Ri
            FINSI
        FAIT
  Envoyer (E, W)
  E= Vide
  W= Prochaine écriture dans S.Journal
FAIT
      }
```

Dans cet algorithme, le serveur ***S*** obtient les vecteurs de version des serveurs R_i (le nombre de serveur peut être 1 ou plus). ***S*** parcours les écritures dans son journal et vérifie si elles sont connues par les R_i. Dans le cas où une écriture n'est pas connue par un serveur, ***S*** l'envoie à ce serveur. Il est important de préciser que l'algorithme parcours le journal des écritures une seule fois.

L'envoie incrémentale des écritures c'est-à-dire qu'on envoie à un serveur les écritures qu'il ne connaît pas, tolère les déconnexions pendant le processus de réconciliation car, le serveur récepteur sauvegarde toute mise à jour dés sa réception. En cas de déconnexion, seules les écritures reçues seront couvertes par le vecteur de version du serveur récepteur, celles qui sont non reçues vont être réexpédiées lors d'une autre session.

- **Stabilisation des mises à jour :**

Une extension de l'algorithme de base permet de considérer la stabilisation des opérations d'écritures qui sont envoyés. Durant la réconciliation, l'émetteur doit d'abord envoyer les écritures stables disponibles, si une écriture n'est pas connue comme tentative par le récepteur seule une notification de stabilisation est envoyée. Ensuite, le protocole continue l'opération de base (l'envoie des écritures tentatives). Pour savoir quelles sont les écritures stables pour un serveur, ce dernier garde le plus grand numéro de séquence qu'il connaît ; ***S.CSN***, pour chaque écriture W_i telle que $W_i.CSN < S.CSN$ est connue par ***S***.

Réconciliation (S, R_1, R_2, ...R_n) {
Var :
E_{ecrit} : Ensemble des serveurs destinataire d'une opération d'écriture, initialisé à vide
E_{notif} : Ensemble des serveurs destinataire d'une notification de stabilisation, initialisé à vide
E_{stable} : Sous ensemble de serveurs destinataire des écritures stables, initialisé à vide

```
E : Ensemble des serveurs initialisé à vide
# n est le nombre de serveurs à mettre à jour.
# Ri.V est le vecteur de version du serveur Ri.
# Ri.CSN est le CSN du serveur Ri.
# Construction de Estable

Récupérer Ri.V et Ri.CSN Depuis Ri avec i : 1..n

POUR chaque Ri FAIRE
    SI Ri.CSN < S.CSN
        ALORS Estable = Estable + Ri
    FINSI
FAIT
SI Estable est non vide
    ALORS                          # Il y a des écritures stables qu'il faut envoyer
    W= première écriture stable dans S.Journal
    TANQUE (W) FAIRE
        POUR chaque Ri FAIRE
            SI W.STAMP > Ri.V (W.ID_SERVEUR)
                ALORS              # Ri ne connaît pas W.
                Ecrit = Ecrit + Ri
                SINON              # Ri a l'écriture mais il ne sait pas qu'elle est stable
                Enotif = Enotif + Ri
            FINSI
        FAIT
    Envoyer (Ecrit, W)
    Envoyer (Enotif, W.STAMP, W.ID_SERVEUR, W.CSN)
    Ecrit = vide
    Enotif= vide
    W= prochaine écriture stable dans S.Journal
    FAIT
FINSI

SI (w.csn= ∞)
    Alors                          # Envoie les écritures tentatives
    W= Première écriture tentative dans S.Journal
    TANQUE (W) FAIRE
        POUR chaque Ri
            FAIRE
                SI W.STAMP > Ri.V (W.ID_SERVEUR)
                    ALORS          #Ri ne connaît pas W
                    E = E + Ri
                FINSI
        FAIT
        Envoyer (E, W)
        E = vide
        W= prochaine écriture dans S.Journal
    FAIT
FSI
    }
```

4.8.3 Réconciliation à plus d'un hop :

Dans cet algorithme le serveur ***S*** parcourt sa table des serveurs, il cherche les serveurs **non marqués** qui ont ***nb_hop***>1.

Le serveur ***S*** lance la phase de réconciliation par couple et en parallèle avec tous les serveurs R_i qui ont ***nb_hop***>1 et **non marqués**

Pour chaque ***Ri***, ***S*** parcours les écritures dans son journal et vérifie si elles sont connues par le R_i sélectionné, en cas où les écritures ne sont pas connues par ***Ri***, ***S*** regroupe toutes les mises à jours qu'il n'a pas et les lui envoie par paquet.

```
Réconciliation (S, R1, R2, ...Rn) {
Var :
Wens : Ensemble des écritures initialisé à vide
Récupérer R1.V, R2.V, ... Rn.V Depuis R1, R2, ...Rn
# n est le nombre de serveur à mettre à jour.
# Ri.V est le vecteur de version du serveur Ri.
W= Première écriture dans S.Journal
POUR chaque Ri FAIRE
    TANQUE (W) FAIRE
        SI W.STAMP > Ri.V (W.ID_SERVEUR)
            ALORS                          # Ri ne connaît pas W
            Wens = Wens + W
        FINSI
    FAIT
    Envoyer (Ri, Wens )
    Wens = Vide
    Ri = Prochain serveur à réconcilier
FAIT
}
```

- **Stabilisation des écritures :**

Réconciliation (S, R_1, R_2, ...R_n) {

Var :

W_{ecrit} : Ensemble des écritures à envoyer, initialisé à vide

W_{notif} : Ensemble des notifications de stabilisation, initialisé à vide

W_{stable} : Ensemble des écritures stables, initialisé à vide

Var W_{ens}: Ensemble des écritures initialisé à vide

Récupérer R_i.V et R_i.CSN Depuis R_i avec i : 1..n

n est le nombre de serveurs à mettre à jour.

R_i.V est le vecteur de version du serveur R_i.

R_i.CSN est le CSN du serveur R_i.

POUR chaque R_i **FAIRE**

SI R_i.CSN < S.CSN

ALORS stable=faux

FINSI

SI stable=faux

ALORS # Il y a des écritures stables qu'il faut envoyer

W= première écriture stable dans S.Journal

TANQUE (W) **FAIRE**

SI W.STAMP > R_i.V (W.ID_SERVEUR)

ALORS # R_i ne connaît pas W

$W_{ecrit} = W_{ecrit} + W$

SINON # R_i a l'écriture mais il ne sait pas qu'elle est stable

$W_{notif} = W_{notif} + W$

FINSI

W= prochaine écriture stable dans S.Journal

FAIT

Envoyer (R_i, W_{ecrit})

Envoyer (R_i, W_{notif})

R_i = Prochain serveur à réconcilier

Stable=vrai

FSI

```
   FAIT

W_ecrit = vide
W_notif= vide
 SI (w.csn =∞)
    ALORS                                  # Envoie des écritures tentatives
      W= Première écriture tentative dans S.Journal
      POUR chaque R_i FAIRE
            TANQUE (W) FAIRE
                  SI W.STAMP > R_i.V (W.ID_SERVEUR)
                     ALORS                 #R_i ne connaît pas W
                      W_ens = W_ens + W
                      FINSI
                W= prochaine écriture dans S.Journal

              FAIT
       Envoyer (R_i, P)
       W_ens = vide
       R_i = Prochain serveur à réconcilier
       FAIT
  FSI
    }
```

4.9 Traitement de la déconnexion du serveur primaire:

Avoir recours à un serveur ***primaire*** permet d'assurer la cohérence des données. L'utilisation d'un serveur ***primaire*** est un moyen incontournable pour l'ordonnancement global des mises à jour et pour assurer une convergence des copies mais il représente aussi une contrainte. Il peut arriver qu'un serveur ***primaire*** tombe en panne ou se déconnecte. Cela empêchera toute nouvelle modification des données sur les serveurs (car les serveurs peuvent appliquer les mises à jour sur les données seulement dans le cas où elles sont stabilisées par le serveur ***primaire***) Ainsi plusieurs nœuds seront plantés dans leur traitements. Ce qui peut induire des débordements des journaux des autre nœuds (car seule la partie stable du journal peut être tronquée).

Pour pallier à ce problème, une stratégie de maintenance du serveur s'avère indispensable. Nous proposons une méthode qui fait en sorte qu'il y ait toujours un serveur ***primaire*** dans le réseau même en cas de déconnexion.

Le serveur ***primaire*** déclenche une procédure périodique qui a pour fonction de détecter les paramètres de risque de déconnexion sont :

- niveau d'énergie,
- et, le nombre de connexions stables : un serveur ***primaire*** doit pouvoir atteindre un maximum de serveurs secondaires, d'où l'intérêt du nombre et de la qualité des connexions. Une connexion entre deux nœuds S_i et S_j est considérée stable pendant une période de temps **T**, si le temps t_{ij} pour que la distance séparant deux nœuds devienne plus grande que la portée de leur liaisons radio est supérieur a **T**. Dans ce cas, la valeur de stabilité du lien reliant S_i à S_j est égale à 1, sinon si $t_{ij} \leq \mathbf{T}$ (le lien va se couper avant la fin de la période **T**) la valeur de stabilité du lien est estimée par $t_{ij}/\mathbf{T}$ [HAR03] :

Nous supposons deux seuils pour ces paramètres. Un premier seuil de déconnexion SD_1 où le serveur ***primaire*** détecte un risque de déconnexion, et un second SD_2 où la probabilité de ce risque devient importante [MOU07a].

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

Dans ce cas, il envoie son état au ***primaire*** (son identité, ses propriétés et son vecteur de version). Le ***primaire*** reçoit les informations des différents serveurs. Lorsque le serveur ***primaire*** N_i atteint le seuil SD_2:

- Il diffuse un message ***Deconnect***(N_i) sur le réseau pour avertir les noeuds du réseau qu'il risque de se déconnecter. Ce message est nécessaire dans le cas où un nœud prévoit de lancer une phase de réconciliation au moment où il va se déconnecter.
- Ni envoi un message ***Primaire*****(Ni,Si)** au serveur S_i
  Ni : l'ancien serveur ***primaire***
  Si : le serveur qui a l'état le plus sûr (c'est-à-dire les meilleures performances en énergie et en nombre de connexions stables).
- Á la réception du message, ce serveur devient ***primaire***. Puis il diffuse l'information sur le réseau.
- Ni lance une phase de réconciliation avec le serveur Si afin que le nouveau ***primaire*** ait la dernière version de la donnée.

❖ ***Procédure de déconnexion du primaire :***

1 - Lorsque le *primaire* atteint le premier seuil :

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

Début
  Diffuser message ***Deconnect***(N_i, 1) # déconnexion probable
Fin.

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

Début
  Si première réception
    Alors
      Si capacité énergétique et liaisons satisfaisantes
        Alors
          Envoyer ***Etat***(identité, énergie, connexions stables, vecteur de version) au ***primaire***
      Fsi
  Fsi
Fin.

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

Début
  Pour chaque serveur N_j
    Faire comparer l'état de serveur avec l'état de serveur précédant
      Stocker le meilleur état.
  Fait
Fin.

2 - Lorsque le *primaire* atteint le deuxième seuil :

Lorsqu'un serveur ***primaire*** N_i détecte sa future déconnexion:

Début
  Diffuser ***Deconnect***(N_i,2) aux nœuds du réseau # Déconnexion
  Rec(Ni,Nj) # N_j Nouveau *primaire*
Fin.

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

Début
  Maj(table des serveur**)**
Fin.

Lorsque la phase de réconciliation est terminée avec Nj :

Début #N_j Nouveau serveur *Primaire*
  Diffuser ***Nou_Prim***(N_i, N_j) à tous les nœuds du réseau.
Fin.

4.10 Conclusion :

Au cours de ce chapitre nous avons essayé de concevoir un protocole de convergence de copies pour les réseaux mobiles Ad hoc. Nous avons essayé au maximum de prendre, au mieux, en considérations les caractéristiques de ces réseaux mobile Ad hoc. Pour cela nous avons tenté en élaborant notre protocole de diminuer le nombre de messages propagés durant la phase de réconciliation afin de diminuer le trafic et l'énergie consommée tout en maintenant une propagation efficace des mises à jour.

L'utilisation d'un serveur principal est une contrainte mais aussi un moyen incontournable pour l'ordonnancement globale des mises à jours et pour assurer une convergence des copies (à part l'utilisation du GPRS qui permet d'obtenir une horloge globale utilisée pour dater les mises à jours provenant des clients; ce qui permet d'ordonner les mises à jours dans le journal. Cependant, cette solution simple présente un inconvénient de coût du matériel). Mais Il peut arriver qu'un serveur ***primaire*** tombe en panne ou se déconnecte. Nous avons proposé une méthode qui fait en sorte qu'il y ait toujours un serveur ***primaire*** dans le réseau même en cas de déconnexion.

Enfin, la performance de tout protocole est liée à des paramètres relatifs à l'environnement externe comme la mobilité, la connectivité, le nombre des nœuds participants, le trafic engendré, l'énergie…etc., que seul une simulation avancée peut évaluer. Ceci fera l'objet du prochain chapitre dans le quel on va essayer d'analyser le comportement du protocole élaboré en conjonction avec les paramètres précédemment énumérés.

Chapitre 5 : Evaluation des performances

5.1 Introduction :

Il existe différentes techniques d'évaluation de performances d'un système sur un réseau mobile Ad hoc. Parmi elles, on peut citer la modélisation analytique, les mesures obtenues à partir d'expériences réelles ou expérimentation, et la simulation

La modélisation analytique consiste à représenter les conditions réelles de façon formelle à l'aide d'outils mathématiques.

La validation de protocoles et d'applications pour MANets par des expériences réelles est complexe à mettre en œuvre. En effet, la mise en œuvre de MANets pour la validation est problématique pour plusieurs raisons :

– les expérimentations sont difficilement reproductibles. Les communications étant radio, elles peuvent être perturbées par des signaux extérieurs sur lesquels l'expérimentateur n'a pas de contrôle.

– L'étude du passage à l'échelle, de la variation de la vitesse et du modèle de mobilité des utilisateurs est complexe à réaliser.

La simulation consiste à modéliser l'ensemble du système étudié et à le simuler numériquement à l'aide d'environnements provenant de mesures sur un système réel ou de modèles probabilistes. L'intérêt de la simulation est de pouvoir travailler sur des systèmes non disponibles. Par exemple, lors de l'étape de conception, il est beaucoup moins coûteux de réaliser une simulation préalable des alternatives envisagées. De plus, la simulation est un moyen très souple pour étudier un problème. Cette technique permet des réexécutions de programmes avec changement de paramètres et une prise de trace d'exécution sans les perturbations imprévisibles d'un environnement réel.

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

Les réseaux mobiles peuvent être simulés à l'aide de langages de programmation à utilisation générale tels que C, C++ et JAVA ou de langages de simulation tels que MODSIM III [RFB90], SIMSCRIPT II.5 et SLAM II. Dans la seconde moitié des années 90, avec l'élaboration de plusieurs normes pour les réseaux sans fil à portée limitée, un certain nombre de simulateurs ont été développés conjointement. Nous citons, par exemple, Network Simulator 2 [WEB02], OPNET [WEB01] et GloMoSim [WEB03]. NS-2 est

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

Dans ce chapitre nous allons évaluer les performances du protocole conçu et étudier son comportement. Les métriques les plus significatives sont évaluées en faisant varier plusieurs paramètres tels que : la surface, le nombre de nœuds, la vitesse, etc.

5.2 Le simulateur GloMoSim :

Glomosim (Global Mobile Information System Simulator) est un environnement de simulation pour les réseaux sans fil et filaires. Il a été conçu à l'université californienne UCLA, dans le laboratoire UCLA PARALLEL COMPUTING LABORATORY [BZG99].

GloMoSim utilise le langage PARSEC ("Parallel Simulation Environment for Complex Systems"). C'est est une bibliothèque de simulation d'évènements parallèles discrets basée sur le langage C. Le noyau de la simulation PARSEC permet à GloMoSim d'avoir une grande vitesse d'exécution et un bon passage à l'échelle.

Glomosim a une structure en couches proche de celle du modèle OSI [DAR 99]. 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 au niveau de chacun des nœuds. A la réception d'un message, une couche exécute la tâche 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 standard (Application Programming Interface) est spécifié sous forme de messages échangés entre les différentes couches du simulateur. Ces facilités autorisent l'intégration rapide et aisée des modèles et des protocoles développés sur les différentes couches par des développeurs distincts. La figure 5.1 représente la pile des couches de protocoles implantés sur le simulateur.

Figure 5.1 : Architecture en couches de Glomosim

5.3 Environnement et Paramètres de simulation :

Cette section présente la configuration, que nous avons utilisé pour notre environnement de simulation.

5.3.1 Le choix du modèle de mobilité :

Parmi les modèles de mobilité offerts par le simulateur Glomosim et qui peuvent représenter la mobilité des nœuds dans un réseau mobile Ad hoc, on trouve le modèle Random Waypoint et Random Druken [NWC04]. Dans le premier modèle un nœud choisit aléatoirement une destination sur le terrain de la simulation et il se déplace vers cet endroit en utilisant une vitesse choisie arbitrairement entre une vitesse maximale et une vitesse minimale spécifiées auparavant par l'utilisateur. Lorsqu'il atteint sa destination, le nœud fait une pause pendant un temps défini. Dans le cas de Random Druken, 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) dans le terrain physique. Tout en suivant une vitesse donnée. Parmi ces deux modèles, nous avons choisi le modèle Random Waypoint, car il est plus proche du mouvement réel des nœuds par rapport aux autres modèles.

5.3.2 Couche application :

Dans cette couche chaque nœud peut générer des mises à jour sur des données. La mise à jour des données est définie comme une application dans le fichier de configuration de Glomosim comme suit :

UPDATE : <Source> <Idf_Data> <Temps de début>.

5.3.3 Couche MAC :

Le protocole que nous avons utilisé dans cette couche, est le protocole IEEE 802.11 qui est très utilisé dans les réseaux sans fil. Dans ce protocole chaque paquet transmis doit être suivi par un acquittement de réception. L'absence de cet acquittement après plusieurs transmissions indique la défaillance du lien entre le nœud émetteur et le nœud récepteur. Dans ce protocole une vérification du canal doit précéder son utilisation ce qui permet d'éviter les collusions [RCM05].

5.3.4 Modèle de propagation :

Parmi les modèles de propagation que nous avons trouvé disponibles au niveau de Glomosim, nous avons choisi le modèle Free space [RBG98] qui est un modèle simple et de propagation libre. Dans ce modèle le signal se propage de l'émetteur vers le récepteur en négligeant tous les obstacles entre eux, et le signal faiblit en fonction de la distance entre le nœud source et le nœud destination.

5.4 Paramètres de simulation :

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

Nous considérons des nœuds dont la portée de communication est la même. Cette portée recouvre une région dans un cercle de rayon ***R***. Nous simulons différentes densités du réseau. La variation de la densité du réseau et la variation du rayon ***R*** de transmission (50m à 250m) influencent le degré de connectivité. Chaque simulation s'exécute pendant 6x60 secondes.

Les valeurs par défaut pour certains paramètres de l'environnement sont exprimées ci-dessous:

Paramètres	**Valeur**
Surface	terrain carré de 1000 m^2
Phase d'initialisation	50 secondes.
Fréquence des mises à jour	1 mises à jour / 4 sec
Temps de simulation	6 minutes
Bande passante	2 Mbits/s
Rayon d'un nœud	180m
Vitesse des nœuds	10m/s
Temps de pause	5 secondes
Nombre de nœud	50
Nombre de nœud serveur	25
Période de réconciliation	30 secondes
Seuil (Nombre de Majs à atteindre pour lancer la réconciliation)	30

Tableau 5.1 : Paramètres de simulation

5.5 Mesures évaluées :

Un protocole de gestion de cohérence est jugé meilleur qu'un autre selon son efficacité et son coût. Plus le taux de cohérence est élevé plus l'efficacité est dite meilleure. Il est aussi nécessaire que les ressources utilisées par un protocole soient minimales tel que la consommation d'énergie, l'utilisation de la bande passante et la capacité de stockage. C'est pour cela, que nous avons choisi les paramètres suivants pour mesurer les performances de notre protocole : Taux de propagation, Taux de cohérence, trafic engendré, énergie consommée, … etc.

➢ **Taux de propagation** :

Ce paramètre nous donne une vue sur le niveau de cohérence des copies du réseau. En d'autres termes, il nous informe sur le pourcentage des mises à jour reçues par les nœuds. Plus ces mises-a-jours atteignent de nœuds, plus nos copies convergeront vers une même donnée.

Pour calculer ce taux, nous utilisons les deux formules suivantes :

$$TP\ (N_i) = \frac{Nbre_{Majs(N_i)}}{Nbre_{Majs(Syst)}} \times 100$$

$$TP = \frac{\sum_1^n TP\ (N_i)}{n}$$

Où :

TP(N_i) : taux de propagation du nœud N_i .

TP: taux de propagation du système.

n : le nombre de nœuds
Nbre $_{Majs(Ni)}$: le nombre de mises à jour arrivées au nœud i.
Nbre $_{Majs(Syst)}$: le nombre de mises à jour dans le système.

- **Taux de cohérence** :

Ce paramètre nous donne une vue sur la cohérence des copies du réseau. En d'autres termes, il nous informe sur le pourcentage des mises à jour stables reçues par les nœuds. Le protocole de gestion de cohérence est d'autant plus efficace que le taux de cohérence est important.

Pour calculer ce taux, nous utilisons les deux formules suivantes :

$$TC(N_i) = \frac{Nbre_{Majs_{Stable(N_i)}}}{Nbre_{Majs_{Stable(Syst)}}} \times 100$$

$$TC = \frac{\sum_1^n TC\,(N_i)}{n}$$

Où :

TC(N_i) : taux de cohérence du nœud N_i .

TC: taux de cohérence du système.

n : le nombre de nœuds

Nbre_Majs_Stable (N_i) : le nombre de mises à jour stables arrivées au nœud i.

Nbre_Majs _stable(Syst.) : le nombre de mises à jour stables dans le système.

- **Trafic généré :**

A travers ce paramètre, le coût en messages émis est estimé en nombre de messages circulant dans le réseau durant toute la simulation.
La formule de calcul du trafic est la suivante :

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

T_i : le nombre de messages transmis par un nœud i durant toute la durée simulation.

n : le nombre de nœuds ayant participé à la constitution du réseau (les nœuds serveurs et clients).

- **L'énergie consommée** :

L'un des éléments qui ont guidé notre contribution consiste en l'optimisation de la consommation d'énergie au niveau des terminaux mobiles dont les batteries ont une faible autonomie, en réduisant les échanges de messages.

Dans nos mesures, nous nous intéressons à l'énergie consommée pour l'envoi de message via l'interface sans fil en mode Ad hoc. Pour l'envoi d'un message, deux étapes sont nécessaires : l'acquisition du support de transmission (le canal de transmission) et l'envoi des paquets proprement dit. L'acquisition du support de transmission génère une consommation d'énergie très élevée. Par conséquent, l'envoi de messages de petite taille génère une consommation d'énergie, disproportionnellement élevé par rapport à leur taille. De plus, le mode Ad hoc génère une consommation d'énergie au niveau des terminaux mobiles en dehors des envois/ réceptions des messages. En effet, les terminaux en mode Ad hoc sont constamment à l'écoute du support de transmission puisqu'ils doivent jouer aussi le rôle de routeurs. Cet état génère une consommation qui s'élève à 741µW en mode Ad hoc contre 48µW en mode infrastructure pour un débit de 11Mbps dans un réseau basé sur le protocole IEEE 802.11b (c-à-d., un terminal à l'écoute du support de transmission consomme en mode Ad hoc 741µW.sec quand 11Mb sont échangés) [LMF01].

Notre modèle pour mesurer la consommation de l'énergie au niveau des entités mobiles s'inspire des équations présentées par Feeney [LMF01]. La consommation de l'énergie est modélisée par l'équation linéaire suivante :

$$\boldsymbol{Energy = m \times size + b \ (\mu W.sec)}$$

La partie constante de cette équation (b) est associée à l'acquisition du support de transmission et au changement de l'état de l'entité mobile (par exemple, de l'état d'écoute à l'état d'émission). La partie incrémentale de l'équation (***m*** x ***size***) est proportionnelle à la taille ***size*** des paquets émis / reçus. Les constantes ***m*** et ***b*** dépendent de l'interface utilisée ainsi que du protocole de transmission. Le tableau 5.2 donne les équations correspondantes à l'énergie consommé pour émettre un message, le recevoir et l'ignorer par les entités non destinataires, en utilisant une interface sans fil 11Mbps de débit et le protocole IEEE802.11b (Wi-Fi) en mode Ad hoc [LMF01].

	µW.sec / *Octet*		*µW.sec*
Point à point (Emettre)	0.48	x***size***	+431
Broadcast (Emettre)	2.1	x***size***	+272
Point à point (Recevoir)	0.12	x***size***	+316
Broadcast (Recevoir)	0.26	x***size***	+50
Point à point (Ignorer)	0,11	x***size***	+66

Broadcast (Ignorer)	Non définie

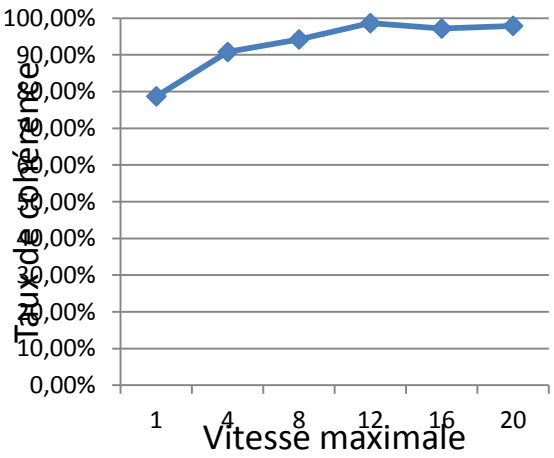
Tableau 5.2 : Modèle linéaire de consommation d'énergie pour l'envoi et la réception des messages

Dans la suite, nous présenterons l'évaluation du coût de notre protocole, en termes d'énergie consommée. Pour le calcule de ce paramètre nous avons voulu étudier la variation de la consommation d'énergie pour tous les nœuds du système.

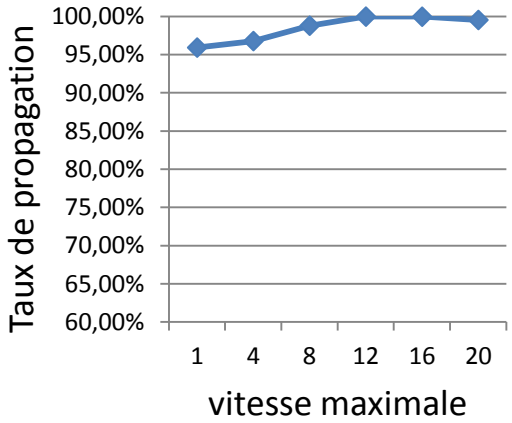
5.6 Résultats de simulation

5.6.1 Effet de variation de la vitesse de déplacement :

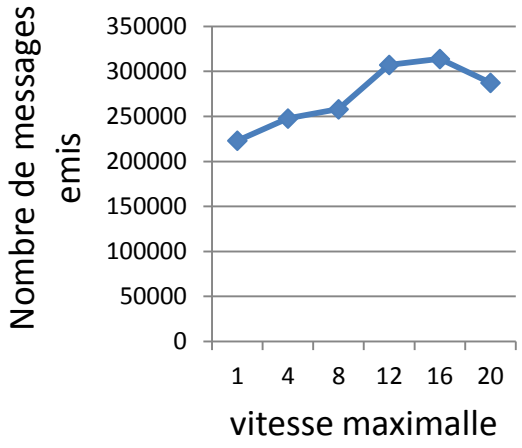
L'effet des variations de la vitesse maximale de déplacement des nœuds sur le taux de cohérence est représenté sur la figure 5.2(a). Cette courbe montre que l'accroissement de la vitesse influence positivement le taux de cohérence. Nous constatons que la vitesse n'a pas un impact négatif important sur le taux de cohérence. La vitesse n'est pas le seul paramètre définissant une mobilité. La mobilité est définie par la vitesse et aussi par le sens de déplacement des nœuds les uns vis-à-vis des autres. Le sens de déplacement étant un paramètre difficile à contrôler, nous étudions la variation de la vitesse qui reste un facteur prépondérant du niveau de mobilité d'un système.



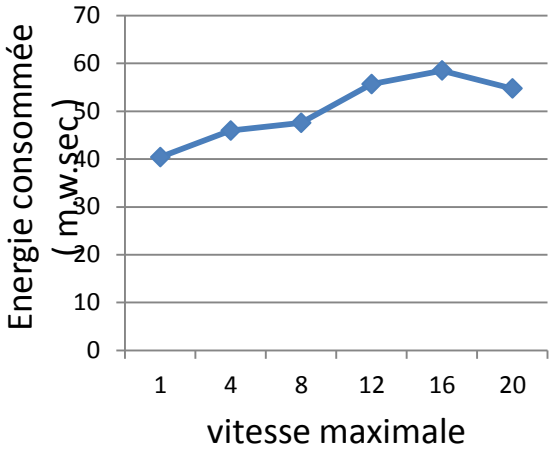
(a)



(b)



(c)



(d)

Figure 5.2 : Effet de variation de la vitesse de déplacement.

Une mobilité importante des nœuds entraîne des changements dynamiques de plus en plus fréquents de la topologie du réseau. Dans ce cas, les serveurs ont plus de chance d'avoir de nouveau voisins donc le principe d'épidémie est plus efficace puisqu'il réussit à se connecter à plus de nœuds. De plus, le changement fréquent de la topologie augmente la chance pour un nœud d'avoir le ***primaire*** comme voisin. En conséquence, les mises à jour sont échangées par ces serveurs, le taux de cohérence devient élevé (figure 5.2 (a)) et il peut atteindre 99%. Par contre, si la mobilité des nœuds est petite, les voisins d'un nœud restant presque les mêmes ou changeant rarement, nécessite plus de temps pour que les Majs puissent être propagées à un maximum de serveurs si les connexions existantes n'y sont pas favorables. Ce qui explique la diminution du taux de cohérence qui peut atteindre 78%.

La figure 5.2 (c) nous renseigne sur l'impact de mobilité des nœuds sur le coût en nombre de messages émis, tel que nous pouvons remarquer que plus la mobilité est petite plus le nombre de messages émis est petit, et vis versa. En effet, une grande mobilité entraîne des changements dynamiques de plus en plus fréquents de la topologie du réseau. D'où plus de chance pour un nœud d'avoir de nouveaux voisins. Donc, plus de messages émis.

Sur la figure 5.2 (d), nous remarquons que l'énergie consommée croît chaque fois que la vitesse de déplacement augmente. Cela est dû au faite que le nombre de messages échangés augmente puisque chaque nouvelle connexion engendre un protocole d'échange de données ou d'opération de mises à jour.

5.6.2 Effet de variation de la densité :

En maintenant une même surface de simulation et, en variant le nombre de nœuds mobiles, nous obtenons une variation de la densité.

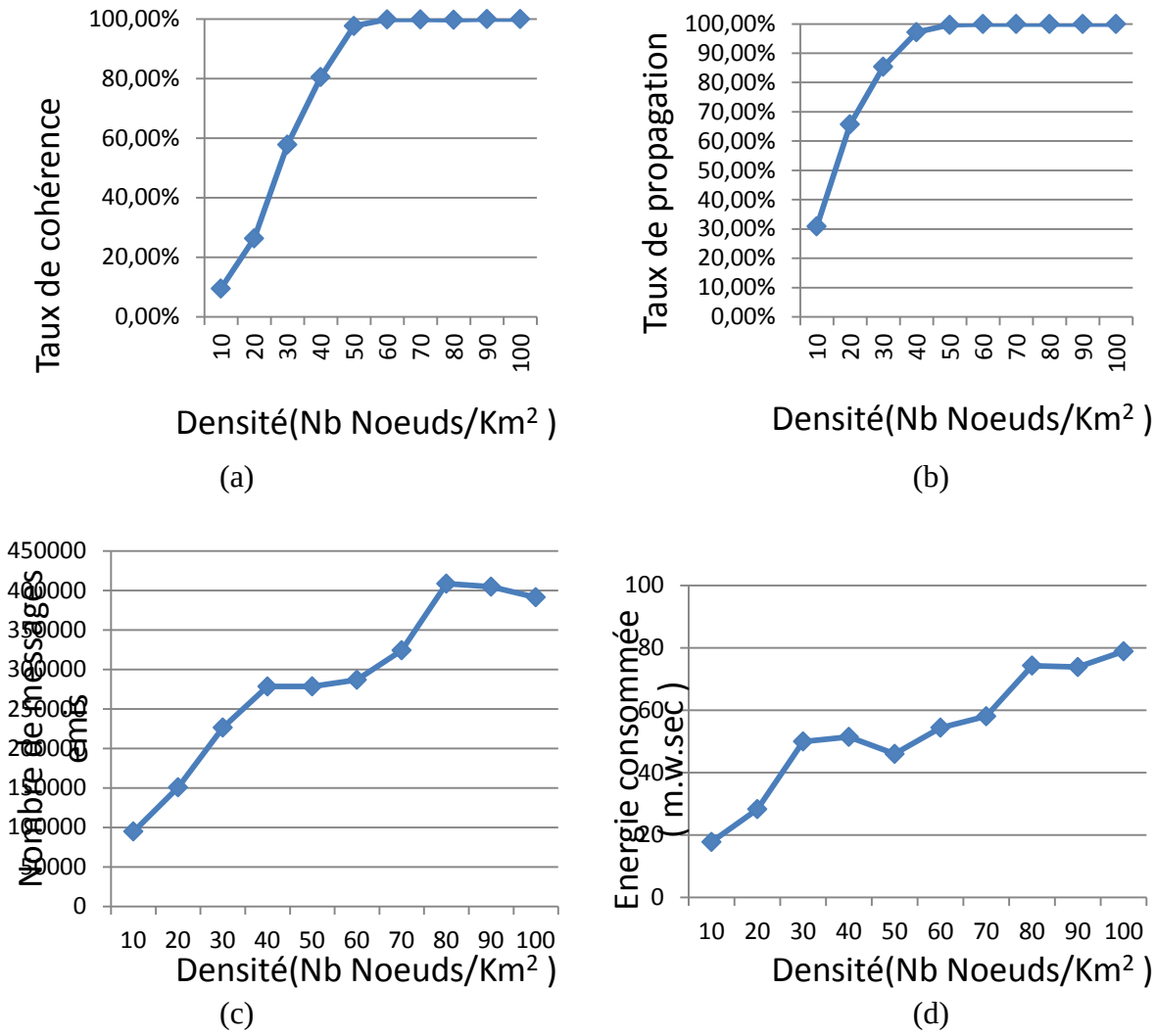


Figure 5.3 : Effet de variation de la densité.

Dans la figure 5.3 (a), nous avons remarqué qu'à chaque fois que la densité augmente, le taux de cohérence augmente. En effet, dans le cas d'une forte densité, les serveurs sont proches les un des autres et les connexions entre les serveurs sont assurées. Dans le cas contraire, certains serveurs ne seront pas atteignables d'où une chute du taux de cohérence. Une faible densité du réseau favorisera même le phénomène de partitionnement du réseau qui partage le réseau en deux sous réseaux disjoints. Une partie des serveurs se voit alors dans l'impossibilité d'atteindre le serveur ***primaire***. Par conséquence, avec une densité convenable les mises à jour sont propagées et reçues par ces serveurs dans de bonnes conditions; il suffit que cette densité assure une connectivité correcte du réseau. Le taux de cohérence devient alors élevé et il peut atteindre les 100%. Mais, si la densité du réseau est faible, le taux de cohérence diminue et il atteint juste les 10% à cause des déconnexions des serveurs qui s'éloignent, de l'isolement du serveur ***primaire*** et des partitionnements du réseau.

Nous remarquons (figure 5.3 (b)) que le taux de propagation diminue aussi avec la chute de la densité du réseau. Cependant, ce taux diminue de manière moins significative relativement au taux de cohérence. Alors que le taux de cohérence diminue jusqu'à atteindre la valeur de 10%, le taux de propagation n'atteindra pas les 30%. En effet, plus la densité est faible, plus il y a des déconnexions et des isolements du ***primaire***; les serveurs peuvent recevoir les mises à jour à partir de leurs voisins mais ils n'arrivent pas à recevoir l'ordre de stabilisation à partir du serveur ***primaire***. Donc, baisse du taux de cohérence.

Le trafic augmente avec l'accroissement de la densité du réseau (figure 5.3 (c)). Car, une grande densité produit plus de connexions. En effet, si la densité est faible, les nœuds peuvent être soit :

- déconnectés et dans ce cas l'échange de messages entre ces serveurs diminue,
- soit éloignés et, par conséquent les serveurs, ce qui fait que nous appliquons la politique de réconciliation serveur par serveur où les mises à jour sont rassemblées et envoyées par paquets pour optimiser le nombre de messages échangés.

Par contre, dans le cas où la densité des serveurs croit, le protocole favorise la réconciliation de plusieurs serveurs plus rapidement en moins d'étapes. Mais les mises à jour sont envoyées une par une, et donc le nombre de messages émis augmente. Ce qui explique l'augmentation du nombre de messages émis pour une densité comprise entre 50 et 100 noeuds/km^2. Malgré que nous ayons obtenu la même valeur de taux de cohérence, le nombre de message émis est plus grand.

La consommation d'énergie (figure 5.3 (d)) par tous les nœuds du système augmente avec l'accroissement de la densité du réseau. Car, une grande densité produit plus de connexions. En conséquence, plus d'échange de messages.

5.6.3 Effet de variation de la portée de communication :

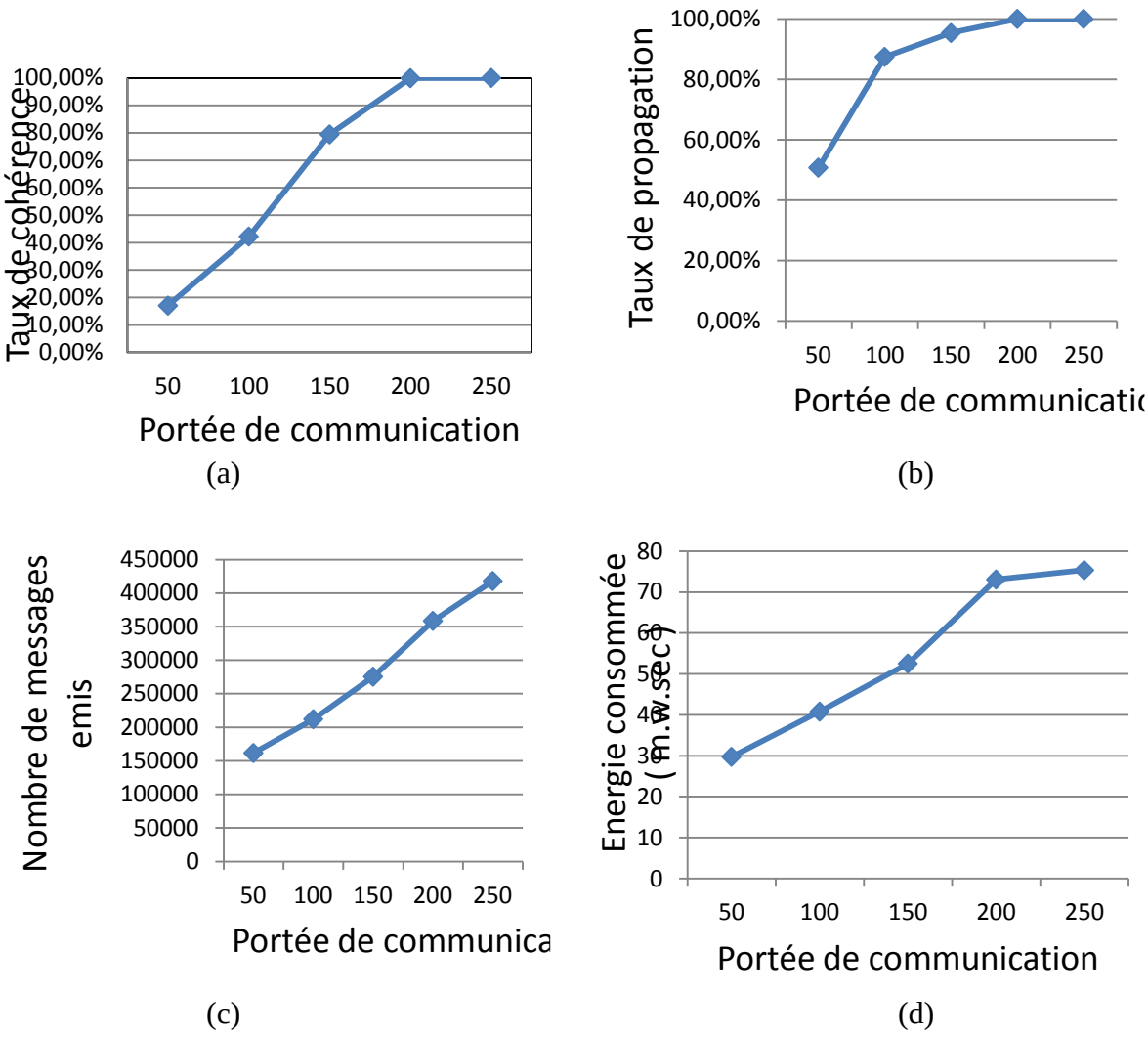


Figure 5.4 : Effet de variation de portée de communication.

Nous remarquons dans la figure 5.4 (a) que le taux de cohérence augmente à chaque fois que la portée de communication augmente ceci est dû au fait que les serveurs deviennent plus proches. De plus, le nombre de voisins augmente à chaque fois que la portée de communication augmente ce qui rend le taux de cohérence élevé. Ces constations sont faites en se référençant aussi aux résultats associes à la figure 5.3.

Nous pouvons remarquer (figure 5.3(c)) que plus le rayon est petit plus le nombre de messages émis est petit, et vis versa. En effet, un petit rayon de communication entraîne une densité du réseau moindre.

5.6.4 Effet de variation de la charge :

La charge du réseau est exprimée par la fréquence des mises à jour. Plus cette fréquence est importante plus la charge est importante.

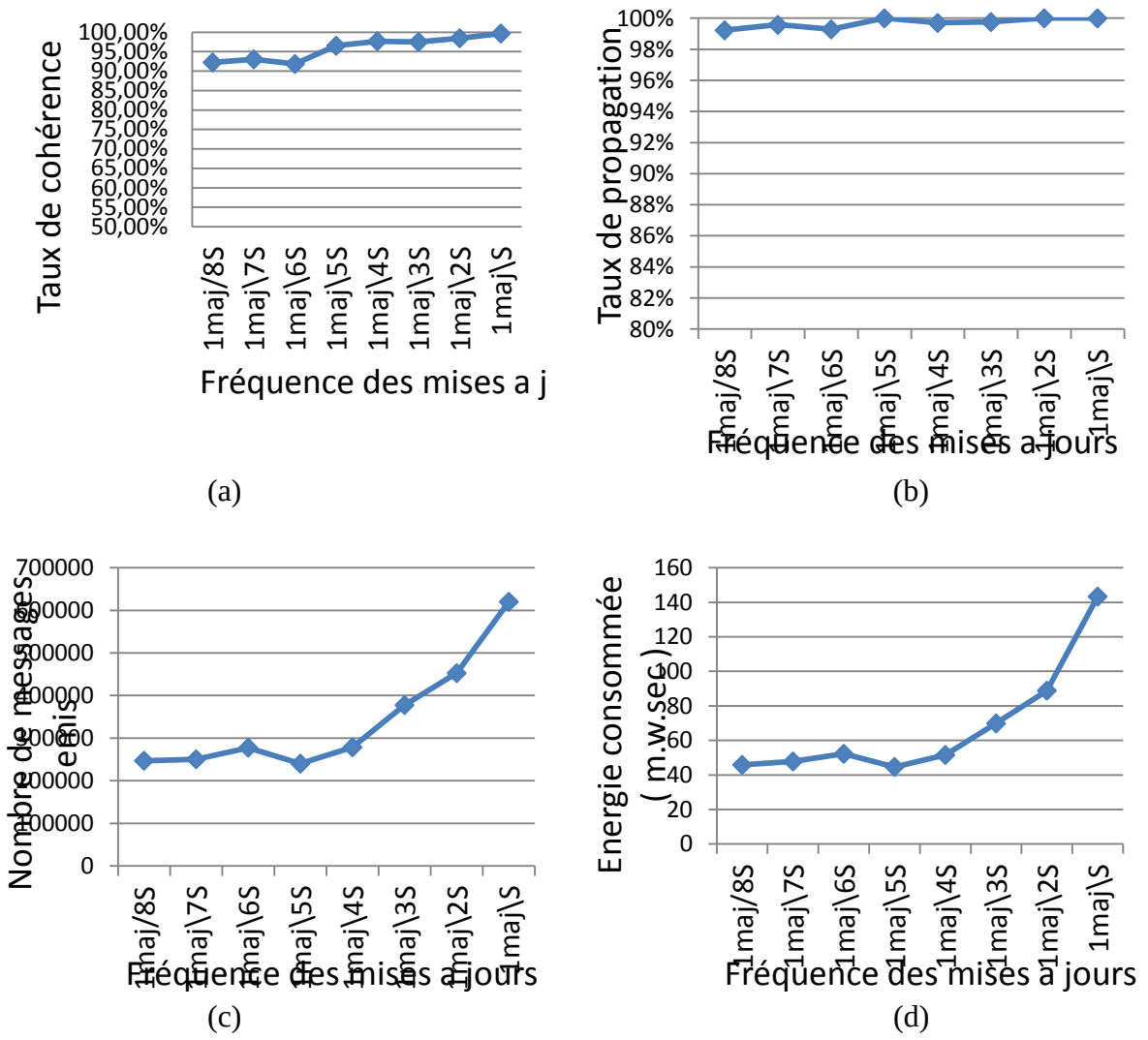


Figure 5.5 : Effet de variation de la charge.

Sur la figure 5.5 (a), nous remarquons que le taux de cohérence diminue chaque fois que la fréquence des mises à jour (Majs) diminue. Cela est dû au faite que le nombre des réconciliations diminue c'est-à-dire la deuxième condition de déclenchement de la réconciliation sera vérifiée moins souvent. Donc, les réconciliations sont déclenchées seulement si la période de réconciliation est expirée. La propagation des mises à jour nécessite, par conséquent, plus de temps pour qu'elles puissent arrivées au niveau des serveurs. Ce qui explique la diminution du taux de cohérence de la valeur de 99% jusqu'à 91% soit une réduction de 8%.

L'augmentation de la fréquence des mises à jour implique l'existence de plusieurs mises à jour nouvelles dans le système qui doivent êtres propagées (figure 5.5 (c)). Ce qui nécessite plus de messages échangés, en plus, le fait qu'il existe plusieurs mises à jour nécessitant une diffusion, le nombre de phases de réconciliation augmente de même que le nombre de messages émis. Par contre, si la charge diminue, le nombre de messages émis diminue car il existe moins de mises à jour à propager et, par conséquent, moins de phases

de réconciliation à déclencher. L'augmentation du nombre des messages échangés nécessite une consommation d'énergie plus élevé (figure 5.5 (d)).

5.6.5 Effet de variation du nombre de serveurs :

L'effet de variation du nombre de serveurs a été étudié en maintenant le nombre de nœuds à 80 nœuds et, en variant le pourcentage des serveurs de 10% à100%.

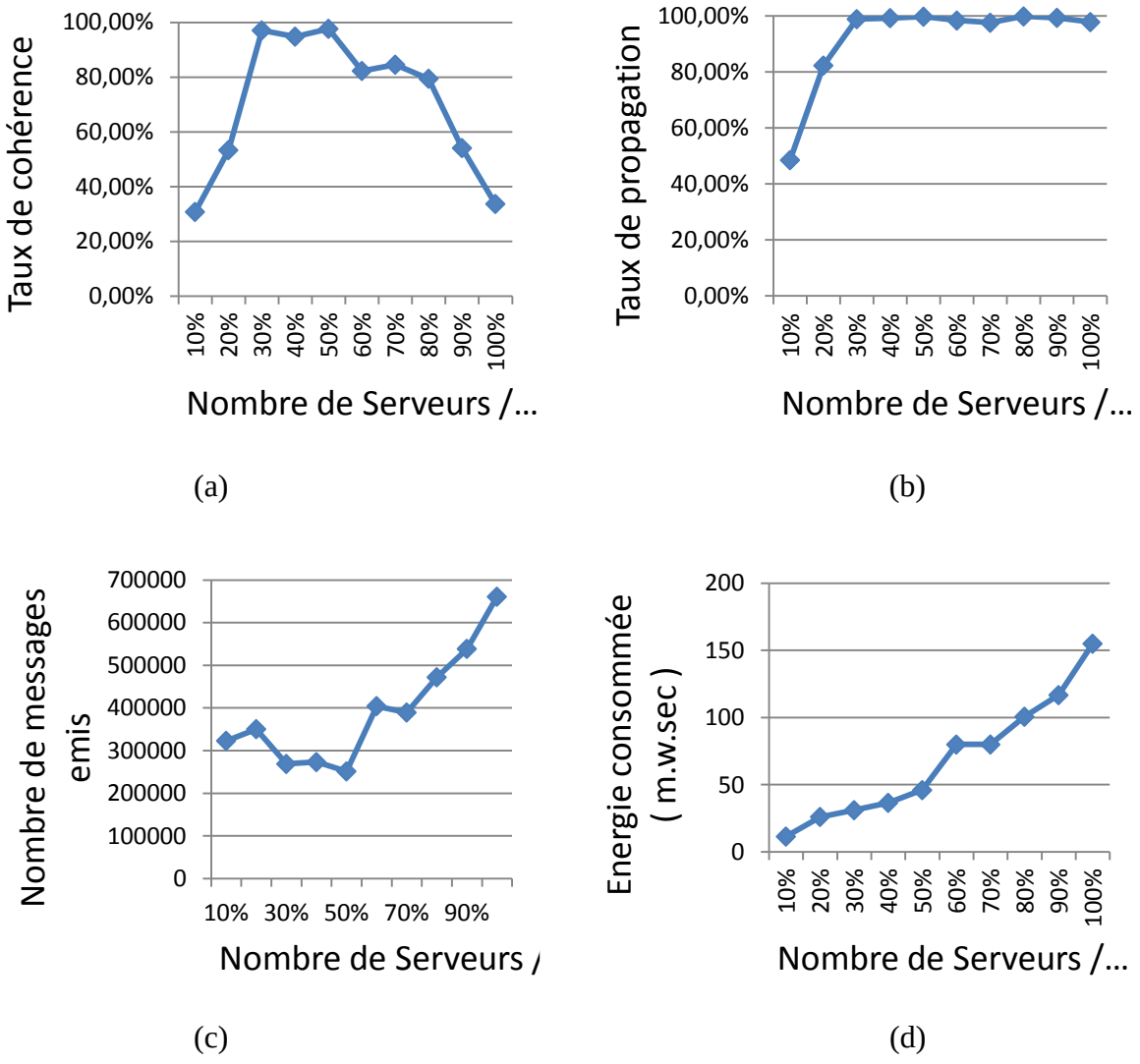


Figure 5.6 : Effet de variation de nombre de serveurs.

Nous observons sur la figure 5.6 (a) que le taux de cohérence croît avec l'augmentation du nombre de serveurs pour ensuite décroître progressivement pour un pourcentage de nombre de serveurs important (au-delà de 50%). L'amélioration de départ du taux de cohérence pourrait s'expliquer par le fait qu'un nombre plus grand des serveurs augmente la chance pour un serveur d'avoir un autre serveur ou le serveur ***primaire*** lui même comme voisin; d'où une propagation plus efficacement des mises à jour. Par contre, la diminution du taux de cohérence s'explique par le fait que lorsque le nombre de serveurs dépasse les

50%, les phases de réconciliation par couple diminue; il existe dans la plupart des cas un serveur intermédiaire entre le serveur qui va lancer la phase de réconciliation et le serveur à réconcilier. Ainsi, la phase de réconciliation par couple ne sera pas déclenchée et les mises à jour envoyées seront propagées par épidémie, d'où un rallongement de la durée de propagation des mises à jour aux autres serveurs. Ce qui explique la diminution de taux de cohérence jusqu'à 34% lorsqu'on à les 100% des nœuds sont des serveurs car dans ce cas les phases de réconciliation par couple ne sont jamais déclenchées.

L'augmentation du nombre de serveurs implique qu'il existe plus de mises à jour nouvelles dans le système qui doivent êtres diffusées. Ce qui nécessite plus de messages échangés (figure 5.6 (c)). Aussi, le fait qu'il existe plusieurs mises à jour à propagées fait que le nombre de phases de réconciliation augmente de même que le nombre de messages émis. Par contre, si le nombre de serveurs diminue, le nombre de messages émis diminue car il existe moins de mises à jour à propager et par conséquent moins de phases de réconciliation à déclencher. L'augmentation du nombre des messages échangés nécessite une consommation d'énergie plus élevée (figure 5.6 (d)).

5.6.6 Effet du passage à l'échelle :

Pour étudier le passage à l'échelle, le nombre de nœuds est augmenté en même temps que l'augmentation de la surface et en maintenant le nombre de serveur à 25% du nombre des nœuds totale.

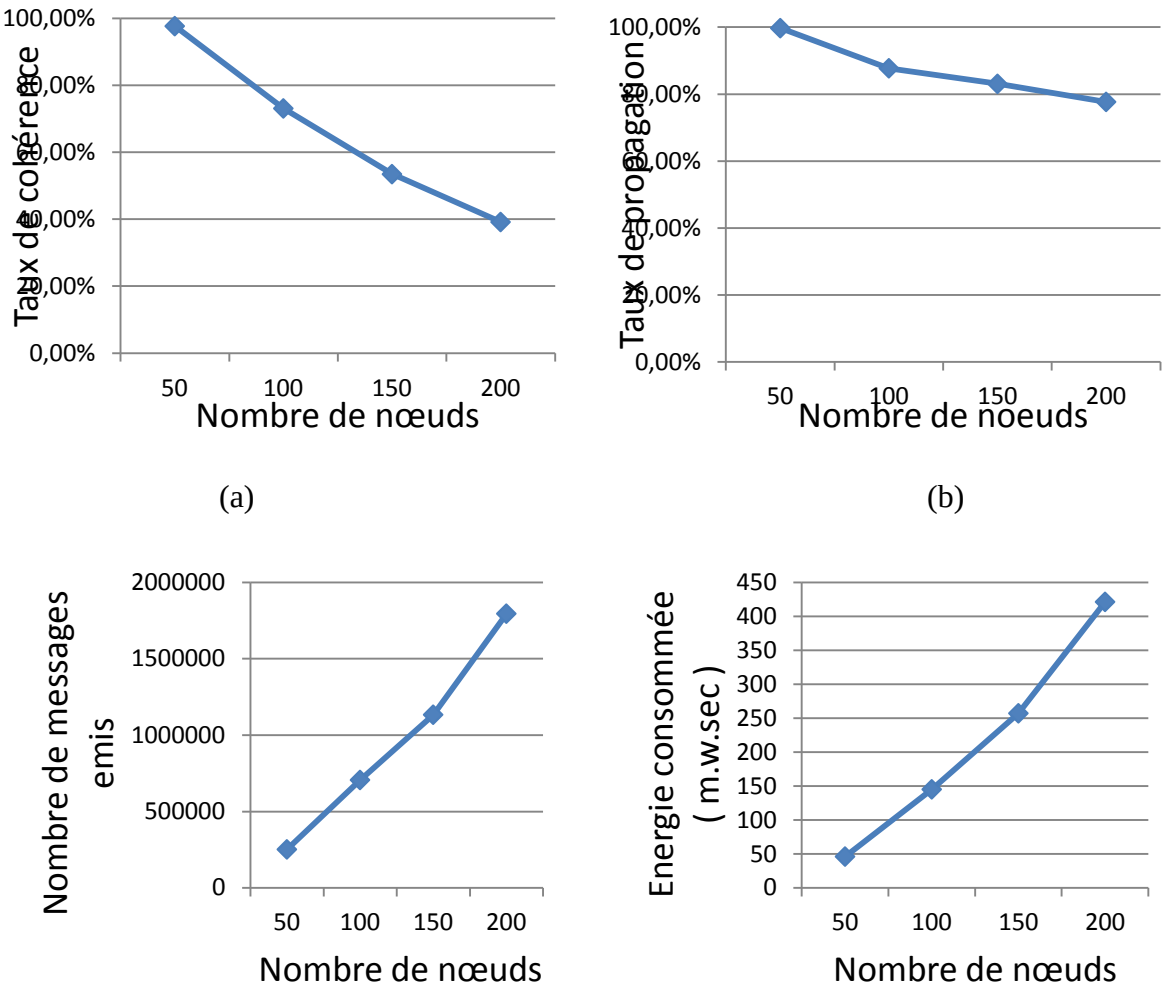


Figure 5.7 : Effet de passage à l'échelle

Nous observons sur la figure 5.7(a) que le taux de cohérence décroît avec l'augmentation du nombre de nœuds. La diminution du taux de cohérence s'explique par la fréquence des partitionnements qui deviennent plus importante avec le passage à l'échelle.

Lorsqu'on a un faible nombre de nœuds qui se déplacent dans une petite surface, il y a plus de chances pour un nœud d'avoir le serveur ***primaire*** comme voisin. Cependant, un grand nombre de nœuds peut aussi affecter les messages. En effet, pour atteindre le ***primaire***, un message peut effectuer plusieurs sauts, donc il peut être atténué ou même perdu. Ceci explique la diminution du taux de cohérence.

Nous remarquons dans la figure 5.7(c) que le nombre de messages émis est proportionnel au nombre de serveurs, c'est à dire que le nombre de messages émis augmente à chaque fois que le nombre de serveurs augmente. Ceci était prévisible à partir

des résultats précédents et on peut l'interpréter en disant que le fait d'augmenter le nombre de serveurs, implique une augmentation du nombre de mises à jour ainsi que du nombre de réconciliation; ce qui nécessite plus de messages échangés. L'accroissement du nombre de messages échangés entraine la consommation de plus d'énergie par les nœuds du système.

5.7. Résultats de la comparaison entre la méthode GBRm (GBR avec mise à jour) et la méthode MRP :

Pour montré l'efficacité de notre solution, nous l'avons comparé à la méthode *GBR avec mise à jour* proposé dans (notons la GBRm) [MOU07a]. Ce choix est dû au fait que notre solution traite le même problème. Cependant, pour GBRm, le taux de cohérence a été calculé pour certaines paramètres, pour lesquelles nous avons pu faire la comparaison telle que : la densité, la vitesse et le passage à l'échelle mais pour la prise en compte d'autres paramètres, les conditions d'évaluation n'ont pas pu être prises en compte car l'environnement et les conditions de développement sont différents.

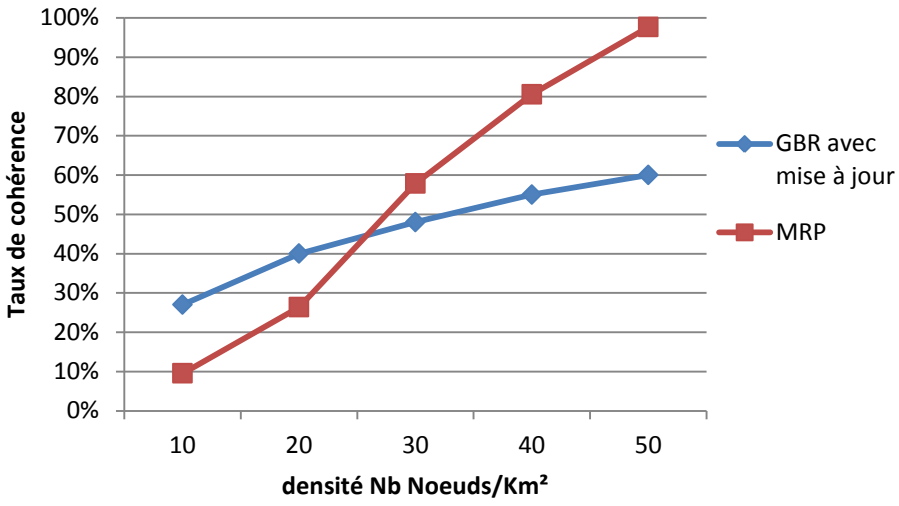


Figure 5.8 : Comparaison des deux méthodes par rapport à la densité.

La méthode MRP répond plus favorable à la densité puisque avec une forte densité nous obtenons une cohérence de 100% pratiquement. Par contre la méthode GBRm avec mise à jour donne des résultats en taux de cohérence bien moins intéressent (environ de 60%). Ceci montre l'intérêt de notre solution qui consiste à utiliser une propagation épidémique des mises à jour. Ce type de propagation est favorisé par une bonne densité. Par contre, avec une densité faible notre solution donne de moins bons résultats puisque la propagation dépend beaucoup de la disponibilité des connexions. Ceci montre, que notre proposition est intéressante pour les réseaux à densité moyenne et forte. La méthode GBRm donne un taux de cohérence légèrement plus grand (autre de 20%) pour une densité faible car, elle propose de transmettre toutes les mises à jour vers le serveur primaire

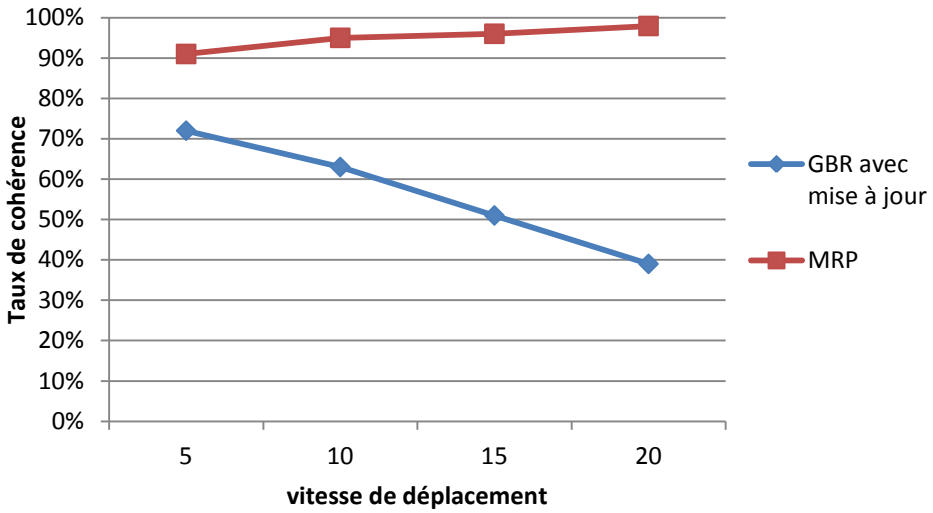


Figure 5.9 : Comparaison des deux méthodes par rapport à la mobilité.

Sur la figure 5.9, nous observons que l'accroissement de la vitesse de déplacement influence négativement le taux de cohérence pour la solution GBRm. Nous pouvons l'expliquer par le fait qu'une mobilité plus importante influence la topologie par des changements plus fréquents. Dans cette solution, lorsqu' un serveur veut modifier une donnée, il envoie sa requête au serveur *primaire* sur un chemin, mais vu la mobilité importante du réseau, la probabilité que ce chemin ne sera pas coupé est faible. Par conséquent, la requête de mises à jour ne peut pas atteindre le serveur primaire et elle ne peut pas être effectuée.

Par contre, la mobilité a un impact positif sur le taux de cohérence pour la méthode MRP, car pour une mobilité importante les serveurs ont plus de chance d'avoir de nouveau voisins. Donc, le principe d'épidémie semble être plus efficace puisqu'il réussit à joindre plus de nœuds serveurs de manière directe et indirecte.

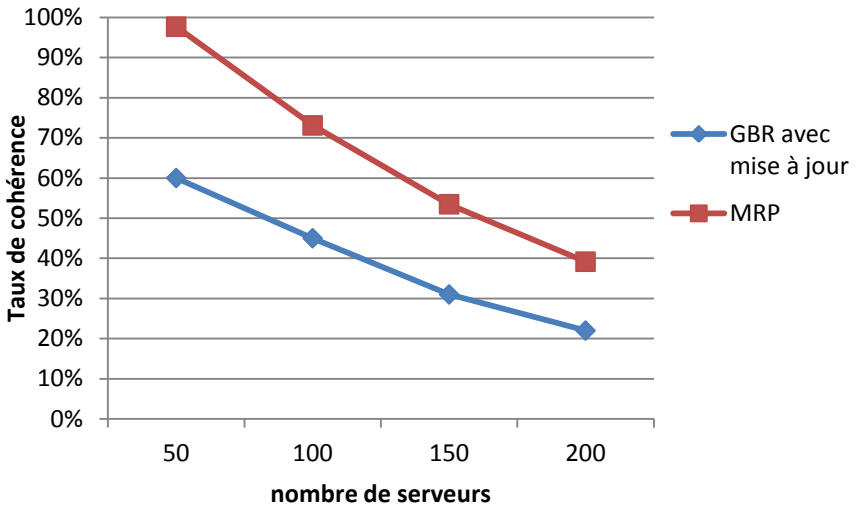


Figure 5.10: Comparaison des deux méthodes par rapport au passage à l'échelle.

Pour étudier le passage à l'échelle, le nombre de nœuds est augmenté en même temps que l'augmentation de la surface. Pour les deux solutions, nous observons que le taux de cohérence décroît avec l'augmentation du nombre de serveurs. Mais le taux de cohérence pour notre solution est meilleur de 30% en moyenne puisque les mises à jour arrivent au niveau des serveurs progressivement en exploitant toutes les connexions qui peuvent exister. Par contre, pour la méthode GBRm, toute mise à jour doit absolument attendre le serveur primaire. Si ce dernier est inatteignable, la mise à jour ne sera pas effectuée.

Ces comparaisons montrent que, relativement à la méthode GBRm, la méthode MRP permet d'avoir des résultats plus intéressants dans le cas de passage à l'échelle, de mobilité important, et de réseau dense,. Cependant, la méthode GBRm donne des résultats meilleurs dans le cas d'un réseau à faible densité.

La méthode GBRm nécessite la connexion du serveur primaire lors des mises à jour et sa déconnexion implique que ces requêtes restent sans réponse. De même, les requêtes d'accès en lecture à une version plus récente ne sont plus satisfaites pour les clients qui ne peuvent atteindre le serveur primaire. La méthode MRP supporte mieux la déconnexion du serveur primaire pour une durée limitée.

La méthode GBRm nécessite l'échange de la totalité de la donnée, qui peut être de taille importante. Ce qui augmente le nombre des paquets envoyés et influence négativement le trafic et la consommation d'énergie.

Chaque méthode peut être utilisée pour un type d'application et d'environnement bien définie. Ceci dépend de la fréquence des modifications (opérations d'écriture), du nombre de serveurs et de la localisation des utilisateurs. Pour la méthode MRP, elle peut être utilisée pour les applications qui favorisent la disponibilité des données sur la cohérence. Elle est aussi intéressante dans le cas des applications où les mises à jour (les écritures) nécessitent

d'être appliquées dans un même ordre sur chaque copie et dans le cas où les écritures nécessitent d'être appliquées en parallèle sur les données. Par contre, la méthode GBRm peut être utilisée dans le cas où les données de l'application ne sont pas de taille important et en cas de modifications peu fréquentes.

5.8. Conclusion :

Nous avons présenté dans ce chapitre le simulateur GlomSim utilisé pour évaluer la stratégie de gestion de cohérence des répliques que nous avons proposée. Nous avons décrit la configuration et les différents paramètres utilisés pour la simulation. Ensuite, nous avons montré l'efficacité du protocole proposé suivant les métriques suivantes : taux de cohérence, taux de propagation et le surcoût (messages émis et énergie consommée). Les mesures effectuées ont révélé des résultats très intéressants. Ces résultats ont montré un taux de cohérence et un taux de propagation qui est voisin de 100%. D'autre part, une consommation énergétique très faible. Ceci est dû au faible surcoût du protocole.

Nous avons remarqué que le taux de propagation converge vers une valeur proche de 100% plus rapidement que le taux de cohérence, par rapports à la densité, à la charge, à la variation de la vitesse de déplacement, et à la variation du taux serveurs. Et même lorsque le taux de propagation diminue avec le passage à l'échelle, il reste supérieur à 75%, lorsque le taux de cohérence avoisine les 40%. Ceci est dû aux déconnexions et à l'isolement du ***primaire***; les serveurs peuvent recevoir les mises à jour à partir de leurs voisins mais ils n'arrivent pas à recevoir l'ordre de stabilisation à partir du serveur ***primaire***.

Conclusion générale

La communication sans fil est devenue une part intégrante de la vie quotidienne de millions de personnes dans le monde. Les utilisateurs cherchent à être toujours libres de leurs mouvements.

Pour cela plusieurs groupes de recherches ont concentré leurs efforts pour améliorer la qualité des services des réseaux sans fil, ce qui a contribué à la naissance des réseaux mobiles en mode Ad hoc.

Un réseau mobile en mode Ad hoc est un réseau sans infrastructure qui comprend des unités mobiles qui sont caractérisées par : la bande passante limitée, l'autonomie en énergie, la sécurité physique limitée et le changement fréquent de la topologie. Cette dernière est due au déplacement libre des unités portables ce qui conduit à des déconnexions qui se produisent fréquemment causant des partitionnements fréquents des réseaux.

Pour palier à ce problème, les machines mobiles sont souvent obligées de copier localement des données utilisateurs, c'est ce qu'on appelle la réplication. Cette dernière résout le problème des disponibilités des données mais crée de nouveaux besoins comme la gestion de la cohérence des données. La gestion de la cohérence regroupe les stratégies de gestion de cohérence forte et les stratégies de gestion de cohérence faible. Cette dernière semble la plus appropriée aux contraintes des environnements mobiles lorsque l'application le tolère.

Dans ce cadre, différentes techniques de convergence des copies existent. Certaines procèdent à la propagation des mises à jour effectuées, d'autres par contre propagent tous le contenu des copies ce qui est très coûteux dans les environnements mobiles et surtout si la taille des objets répliqués est importante (base de données par exemple).

Pour cela, et au cours de notre travail, nous avons essayé de proposer un protocole optimiste de convergence des copies adaptées aux caractéristiques des réseaux mobiles Ad hoc. Ce protocole est basé sur la journalisation des écritures. Il permet la réconciliation de plusieurs copies en même temps, ce qui permet de diminuer le coût en messages émis ainsi que l'énergie consommée par les unités mobiles tout en assurant une convergence plus rapide des copies.

Nous pouvons résumer les caractéristiques de notre protocole en ce qui suit :

- Propagation des mises à jour au lieu du contenu des copies rendant l'utilisation de la bande passante dépendante de la fréquence des mises à jour et non de la taille des copies répliquées.

- L'utilisation de l'historique des mises à jour et l'estampille associée à chaque mise à jour permet d'avoir un ordre total des mises à jour sur les copies des données répliquées.

- La propagation des mises à jour se fait d'une manière épidémique ce qui optimise le nombre de phases de réconciliation afin de converger le plus rapidement possible vers un état cohérent.

- Toutes les mises à jour (écritures) diffusées ne sont pas connues aux serveurs récepteurs. Par conséquent, un serveur va recevoir l'écriture une et une seule fois, ce qui optimise l'utilisation de son énergie (batterie).

- En cas de déconnexion, seules les mises à jour, reçues durant une phase de réconciliation, seront connues selon le vecteur de version du serveur récepteur. Les mises à jour non reçues vont êtres renvoyées lors d'une autre phase de réconciliation et dés le rétablissement de la connexion.

- Notre protocole comporte un algorithme de troncature du journal des mises à jour qui permet de palier au problème de dépassement de capacité du journal.

- Notre solution traite le cas de déconnexion du serveur ***primaire*** et la réélection de nouveau ***primaire***. Nous proposons une méthode qui fait en sorte qu'il y ait toujours un serveur ***primaire*** dans le réseau même en cas de déconnexion.

- Le nombre de messages émis (messages redondants de réconciliation) a été réduit surtout dans le cas d'un réseau connexe et ceci suite à l'utilisation du principe de diffusion dans les réseaux mobiles Ad hoc pour un nombre de hop égal à un.

- Convergence rapide : le faite que les demandes d'initialisation d'une phase de réconciliation sont envoyées en parallèle, le système converge vers un état cohérent le plus rapidement possible. De plus, du moment que notre algorithme offre la possibilité de réconcilier plusieurs serveurs en même temps en bénéficiant de leurs écoute sur l'onde radio. Les écritures propagées vont être reçues par un ensemble de serveurs en même temps pendant une phase de réconciliation au lieu qu'elles arrivent à un seul serveur.

- L'utilisation de la réplication optimiste permet une grande disponibilité des données.

Après avoir évalué les performances de notre protocole, nous avons vu que l'augmentation de la vitesse de déplacement des nœuds a entrainé l'augmentation du taux de cohérence. D'un autre côté, nous avons constaté que la densité du réseau a un impact très important sur la progression du taux de cohérence qui augmente à chaque fois que la densité du réseau augmente. Ainsi, nous avons montré que le taux de cohérence diminue chaque fois que la fréquence des mises à jour (Majs) diminue mais cette diminution n'est pas importante et elle varie de 99 % à 92%.

Nous avons constaté que le taux de propagation converge (vers une valeur proche de 100%) plus rapidement que le taux de cohérence ; ceci dans les cas de variation de la densité, de la charge, de la vitesse de déplacement, et du taux de serveurs. Et même lorsque le taux de propagation diminue avec le passage a l'échelle le taux de propagation de descend pas en dessous de 75% alors que le taux de cohérence diminue jusqu'à 40%.

les comparaisons entre la méthode MRP et GBRm ont montrent que la méthode MRP permet d'avoir des résultats plus intéressants en cas de passage à l'échelle, de mobilité important, et de réseau dense, par rapport à la méthode GBRm. Cependant, la méthode GBRm donne des résultats meilleurs dans le cas d'un réseau à faible densité.

La simulation et l'évaluation ont montré que ce protocole donne des résultats intéressants encourageant la poursuite de cette étude et l'amélioration de la proposition.

L'intérêt du problème abordé par ce mémoire en fait un thème d'actualité pour la recherche. Les diverses questions qui peuvent faire l'objet d'une extension à

ce travail sont sans doute nombreuses. Néanmoins, nous optons pour les premières perspectives suivantes:

- Dans ce travail nous avons traité les déconnexions prévisibles du serveur ***primaire*** et non le cas des déconnexions ou pannes subites. Ce dernier point n'a pas fait l'objet de cette étude mais constitue l'une des perspectives.

- Concevoir un module de gestion de conflit des mises à jour. En effet, le fait d'utiliser une gestion de cohérence optimiste où la cohérence assurée est faible, produit inévitablement des problèmes de conflits de mises à jour qui sont résolus selon les systèmes de différentes manières et souvent de manière manuelle. D'autres parts, ces conflits et leur résolution sont étroitement liés au type de l'application traitée.

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

Bibliographie

[AAI08]: Annie Gentes, Aude Guyot-Mbodji, and Isabelle Demeure,
«*Gaming on the move: urban experience as a new paradigm for mobile pervasive game design*»,
In MindTrek '08 : Proceedings of the 12th international conference on Entertainment and media in the ubiquitous era, pages 23–28, New York, NY, USA, 2008. ACM.

[AAZ06]: Abdrabou, A. and Zhuang, W,
«*A position-based qos routing scheme for UWB mobile Ad hoc networks*»,
IEEE J. Select. Areas Commun, vol. 24, pp. 850-856, (2006).

[BAE05]: Bashandy, A. R. Chong, E. K. P. and Ghafoor, A,
«*Generalized quality-of service routing with resource allocation*»,
IEEE J. Select. Areas Commun, vol. 23, pp.450-463, (2005).

[BAG95]: ALINE BAGIO,
« *Environnement mobile : Etude et synthèse bibliographique* »,
Rapport Interne, INRIA ROCQUENCOURT. Septembre 95.

[BAL02]: R.Baldoni,M.Raynal,
« *Fundamentals of Distributed Computing:A Practical Tour of vector Clock Systems*»,

[BHA05**]:** Badis, H. and Al Agha, K,
«*QOLSR, QoS routing for Ad hoc Wireless Networks Using OLSR*»,
In European Transactions on Telecommunications, vol. 15, n° 4 (2005).

[BKS03]: Barolli, L. Koyama, A. and Shiratori, N,
«*A QoS routing method for ad-hocnetworks based on genetic algorithm*»,
In Proceeding 14th Int. Wksp. Database and Expert Systems Applications, pp. 175-179, (Sept. 2003).

[BML04]: Benaissa, M. and Lecuire, V,
«*A New Smoothing Jitter Algorithm for Voice over Ad hoc Networks*»,
MWCN 2004: 167-178 (2004).

[BNF]: Edmund B. Nightingale and Jason Flinn,
«*Energy-Efficiency and Storage Flexibility in the Blue File System*»,
Department of Electrical Engineering and Computer Science University of Michigan

[BOU03a]: Malika BOULKENAFED,
«*Gestion de l'accès aux données dans les réseaux sans fil en mode Ad hoc* »,
Thèse Doctorat, Université de paris 6 Novembre 2003.

[BOU03b]: Boulkenafed M. and Issarny V,
«*AdHocFS : sharing files in WLANs*»,
In Proceedings of Network Computing and Applications, pages 156–163, 2003.

[BUS07]: Jean-Michel Busca. Pastis,
«*Un système pair-à-pair de gestion de fichiers*»,
Thèse de Doctorat, UPMC-Paris, 2007.

[BZG99]: R. Bagrodia, X. Zeng, and M. Gerla,
«*GloMoSim - A Library for Parallel Simulation of Large-scale Wireless Networks*»,
Computer Science Departement, University of California at Los Angles, 1999.

[CAR05]: Eric Cariou,
«*Cours Algorithmique distribuée : Exclusion mutuelle*»,
Université de Pau et des Pays de l'Adour, 2005.

[CBS04]: C. Siva Ram Murthy and B.S. Manoj,
«*Ad-hoc Wireless Networks : Architectures and Protocols*»,
Number ISBN 0-13-147023-X. Prentice Hall (Eds.), New Jersey, USA, 2004.

[CEP00]: C. E. Perkins,
«*Ad-hoc Networking »*,
Number ISBN 0-201-30976-9. Pearson Education,Addison-Wesley (Eds.), New Jersey, USA, 2000.

[CHC05]: Chaudet C,
«*Protocole IEEE 802.11: qualité de service*»,
Techniques de l'ingénieur, Télécoms A. 2005, vol. TEA2, n° TE7379 (2005).

[CLS02]: Cecilia Mascolo, Licia Capra, Stefanos Zachariadis, and Wolfgang Emmerich,
«*Xmiddle : A data-sharing middleware for mobile computing*»,
Wirel Pers. Commun, 21(1):77–103, 2002.

[CSN99]: Chen, S. and Nahrstedt, K,
«*Distributed quality-of-service routing in Ad hoc networks*»,
IEEE J. Select. Areas Commun., vol. 17, pp. 1488-1505 (Aug. 1999)

[CTJ03]: Clausen, T., and Jacquet, P,
«*Optimized Link State Routing Protocol (OLSR) »*,
IETF, RFC 3626 (2003).

[DAR99] :DARPA ATO Sponsored Research,
«*Design of Mobile Adaptive Networks Using Simulation And Agent Technology*», UCLA, 1999.
Concepts et technologies DUNDO1981.

[DDB06]: Djenouri, D. and Badache, N,
«*New power-aware routing protocol for mobile Ad hoc networks*»,
Int. J. Ad hoc and Ubiquitous Computing, Vol. 1, N° 3, 126-136 (2006).

[DGT87]: Alan Dcmers. Dan, Greene, Carl Hauser, Wes Irish, John. Larson. Scott Shenkcr, Howard Sturgis, Dan Swinehart, and Doug Terry,
«*Epidemic Algorithms For Replicated Database Maintenance*»,
Xerox Palo Alto Research Center 1987.

[DNF92]: D.Duchamp and N.F.Reynoldr,
« *Measured performance of wireless LAN* »,
Technical Report, Computer Science Department, Columbia University, NY, United States,1992.

[DPW94]: A. J. Demers, K. Petersen, M. J. Spreitzer, D. B. Terry, M. M.Theimer Et B. B. Welch,
« *The Bayou Architecture : Support for Data Sharing among Mobile Users* »,
Proceedings of the IEEE Workshop on Mobile Computing Systems and Applications, p. 2.7, Santa Cruz, California, USA, décembre 1994,

[DRA00]: Stéphane Drapeau,
«*Modèles de Cohérence*»,
France Télécom R&D, 2000.

[EPB 95]: E. Pitoura and B. Bhargava,
« *Maintaining Consistency of Data in Mobile Distributed Environments* »,
Proceeding of IEEE International first Conference on Distributed Computing Systems (ICDCS'95), pp. 404-413, 1995.

[EPT97]: W. K. EDWARDS, E. D. MYNATT, K. PETERSEN, M. J. SPREITZER, D. B. TERRY and M. M. THEIMER,
« *Designing and Implementing Asynchronous Collaborative Applications with Bayou* »,
Proceedings of the 10th ACM Symposium on User Interface Software and Technology (UIST'97), p. 119.128, Banff, Alberta, Canada,

[ERC99]: Elizebeth M. Royer,C-K Toh,
«*A Review of Current Routing Protocols for Ad hoc Mobile Wireless Networks*»,
IEEE Personal Communications,pages 46-55, April 1999.

[FID91]: Colin Fidge,
« *Logical Time in Distributed Computing systems* »,
IEEE Computer,24(8):28-33, August1991.

[HAR03]: Takahiro Hara, Yin-Huei Loh and Shojiro Nishio,
«*Data ReplicationMethods Based on the Stability of Radio Links in Ad hoc Networks*»,
Department of Multimedia Engineering, Osaka University, in Proceeding DEXA IEEE 2003.

[HDH10]: Hoa Dung Ha Duong,
«*Partage de données en mode pair à pair sur réseaux mobiles Ad hoc*»,
Thèse de doctorat de l'École Nationale Supérieure des Télécommunications de Paris, le 24 septembre 2010.

[HLC06]: Huang Jiun-Long and Chen Ming-Syan,
«On the Effect of Group Mobilityto Data Replication in Ad Hoc Networks»,
IEEE Transactions on Mobile Computing,vol. 1, n°5, May 2006.

[HMJ11]: Hyun-Gul Roh, Myeongjae Jeon, Jin-Soo Kim, and Joonwon Lee,
«*Replicated abstract data types: Building blocks for collaborative applications*»,
Journal of Parallel and Dist. Comp., (To appear) 2011.

[HMM02]: H. Li, M. Lott, M. Weckerle, W. Zirwas, and E. Schulz,
«*Multihop Communications inFuture Mobile Radio Networks*»,
In Proceeding of the 13th IEEE International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC'02), Lisboa, Portugal, September 2002.

[GAD08]: Gopi Kandaswamy, Anirban Mandal, and Daniel A.Reed,
«*Fault Tolerance and Recovery of Scientific Workflows on Computational Grids*»,
IEEE Computer Society, 2008.

[GKL03]: Ge, Y., Kunz, T. and Lamont, L,
«*Proactive QoS Routing in Ad hoc Networks*»,
2nd International Conference on Ad hoc networks, LNCS 2865, Springer, Berlin (2003).

[IMB94]: Imienlinksi T . and Badrinath B.R,
« *Mobile Wireless computing: solutions and challenges in data management*»,
CACM,37(10),pp18-28, October1994.

[JHE99]: Jin Jing, Abdelsalam Sumi Helal, and Ahmed Elmagarmid,
«*Client-server computing in mobile environments*»,
ACM Computing Surveys, 31(2):117– 157, 1999.

[JKS91]: J. J. Kistler and M. Satyanarayanan,
«*Disconnected operation in the Coda file system*»,
In Thirteenth ACM Symposium on Operating Systems Principles, volume25, pages 213–225, Asilomar Conference Center, Pacific Grove, US, 1991.

[JKS92]: J. J. Kistler and M. Satyanarayanan,
«*Disconnected operation in the Coda file system*»,
ACM Transactions on Computer Systems, 10(1), February 1992.

[JPF04]: João Pedro Faria Mendonça Barreto,
«*Haddock-FS : A distributed file system for mobile ad-hoc networks*»,
Master's thesis, Instituições portuguesas – UTL Universidade Técnica de Lisboa – IST-Instituto Superior Técnico – -Departamento de Engenharia Informática, 2004.

[JUD 98]: Judge A., Nixon P.,Cahill V . Et Al,
«*Overview of Distributed Shared memory »*,
TCD-CS-1998-24,Trinty College Dublin,1998.p.44.

[KIS91]: James J. Kistler and M. Satya-narayanan,
«*Disconnected operation in the Coda file system*»,
In Proceedings of 13th ACM Symposium on Operating Systems Principles, pages 213–25, Association for Computing Machinery SIGOPS, October 1991.

[LAM01]: Lambert SONNA MOMO,

« *Réplication et Durabilité dans les systèmes répartis* »,
Projet de semestre, 19 février 2001.

[LCL99]: Lin, C. R. and Liu, J.-S,
«*Qos routing in Ad hoc wireless networks*»,
IEEE J.Select. Areas Commun, vol. 17, pp. 1426-1438, (1999).

[LEM00]: Tayeb Lemlouma,
«*Le routage dans les réseaux mobiles Ad hoc*»,
Rapport de mini projet de fin de 1ere année post de graduation institut d'Informatique,USTHB, octobre 2000.

[LMF01]: Laura Marie Feeney,
«*An energy-consumption model for performance analysis of routing protocols for mobile Ad hoc network*»,
To a paper in journal of mobile networks and applications, 2001.

[MAB02]: Misra, A. and Banerjee, S,
«*MRPC: Maximizing network lifetime for reliable routing in wireless environments*»,
In Proc. IEEE Wireless Communications and Networking Conf., Orlando, Florida (2002).

[MAR00]: Mark C. Little, Santosh K. Shrivastava,
«*Integrating Group Communication with Transactions for Implementing Persistent Replicated Objects* » ,
LNCS, vol.1752, Springer-Verlag, pp.238-253.2000.

[MBSO8]: C. Michael Pilato, Ben Collins-Sussman, and Brian W. Fitzpatrick.
«*Version Control with Subversion*»
O'Reilly Media, 2 edition, September 2008.

[MES95]: Lily B.Mummert,Maria R.Ebling,and M.Sayanarayanan,
«*Exploiting weak connectivity for file access*»,
ACM SIGOPS,pages 143-155,December 1995,

[MFP00]: Magnus Frodigh, Per Johansson, Peter Larsson ,
«*Wireless Ad hoc networking,the art of networking without a network* »,
Ericsson Review No.4 , pp,248-263, 2000.

[MGA95]: M. Gallersdörder and M. Nicola,
«*Improving Performance in Replicated Data- bases through Relaxed Coherenc*»,
In Proc. of the VLDB Conf., Zürich, Switzerland, 1995.

[MOH07]: Moh'd A. Radaideh, Hayder Al-Ameed,
«*Architecture of Reliable Web Applications Software*»,
Edition IGI Global, 2007.

[MOU06]: Samira Moussaoui, Mohamed Guerroumi and Nadjib Badache,
«*Data Replication in Mobile Ad Hoc Networks*»,
Proceeding of Int. Conf. Mobile Ad Hoc and Sensors Networks MSN'06, Springer LNCS 4325, pp. 685-698, HongKong, December 2006.

[MOU07a] : Samira Moussaoui,
«*Contribution à l'Amélioration de l'Accessibilité des Données sur Réseaux Mobiles Ad hoc* »,
Doctorat d'état, USTHB,2007.

[MOU07b]: Samira Moussaoui, Mohamed Guerroumi and Nadjib Badache,
«*sharing Data Access on Mobile Ad hoc Networks*»,
Proceeding of International conference on Wireless Communication and Mobile Computing M-WCMC 2007,Amman Jordan, September 6-8,2007.

[MOU07c]: Samira Moussaoui and Nadjib Badache,
«*Replicas updates on Mobile Ad hoc Networks*»,
Proceeding of conference ICMTD'07, 27-30 December,2007. (To appear)

[MOU07d]: Samira Moussaoui, Mohamed Guerroumi and Nadjib Badache,
« Replication Approach for MANETs »,
Int. Journal System and Information Sciences Notes, SIWN (Systemics and Informatics World Network), ISSN 1753-2310, Vol.1, N°3, pp. 255-262, July 2007.

[MOU07e]: Samira Moussaoui, Mohamed Guerroumi and Nadjib Badache,
«Improving Data Accessibility in Mobile Ad hoc Networks»,
International Journal AJIT, Special Issue: Cross Layer Design of Multihop Wireless Networks, ISSN: 1449-2679, 2007 (queued for editing).

[MOU08a]: Samira Moussaoui, Mohamed Guerroumi and Nadjib Badache,
«Two phases replication approach on Mobile Ad hoc Networks»,
The International Journal for Ad Hoc and Ubiquitous Computing (IJAHUC). (Acceptable for publishing –under final revision).

[MOU08b]: Samira Moussaoui, Mohamed Guerroumi and Nadjib Badache,
«Data Access in Mobile Ad hoc Networks»,
International Arab Journal of InformationTechnology. (Accepted for publication).

[MOU09]: S. Moussaoui, M. Guerroumi and N. Badache,
«*Two phase replication approach for MANETs*»,
Int. J. Ad Hoc and Ubiquitous Computing, Vol. 4, No. 5, 2009

[MUM96]: L.B. Mummert,
« *ExploitingWeak Connectivity in a Distributed File System*»,
PHD thesis, September 1996.

[NDA09]: Nguyen, D-Q. and Minet, P., Adjih, C., and Plesse, T,
«*Implementation and performance evaluation of a quality of service support for OLSR in a real MANET*»,
SimuTools (2009).

[NNB01]: Nikaein, and N. Bonnet, C,
«*Hybrid Ad hoc routing protocol –HARP»* ,
in Proc.Int. Symp, Telecommunications (2001).

[NWC04]: Navidi, W. and Camp, T,
«*Stationary Distributions for the Random WaypointMobility Model*»,
IEEE Transactions on Mobile Computing 3(1), pp. 99-108 (2004).

[OLD00]: Olivier Dedieu,
« *Réplication optimiste pour les applications collaboratives asynchrones* »,
Thèse de Doctorat de l'Université de Marne-la-Vallée. 14 septembre 2000.

[PAP 04]: Cécile Le Pape, Stéphane Gançarski,
«*Fraîcheur et validité de données répliquées dans des environnements transactionnels*»,
Laboratoire d'informatique de Paris 6, 2004.

[POW02]: P. Juang, H. Oki, Y. Wang, M. Martonosi, L. S. Peh, and D.Rubenstein,
«*Energy efficient computing for wildlife tracking: Design tradeoffs and early experiences with zebranet*»,
In ASPLOS-X : Proceedings of the 10th International Conference on Architectural Support for Programming Languages and Operating Systems, pages 96–107,New York, NY, USA, 2002. ACM Press.

[PPM07]: Patnaik, P. K. and Mall R,
«*Power and Battery Aware Routing in Mobile Ad hoc Networks*»,
Icfai Journal of Computer Sciences, Vol. 1, N° 1 (2007).

[PST96]: K. PETERSEN, M. J. SPREITZER, D. B. TERRY et M. M. THEIMER,
« *Bayou : Replicated Database Services for World-wide Applications* »,
Proceedings of the 7th ACM SIGOPS European Workshop (EuroSIGOPS'96), p. 275.280, Connemara, Ireland, septembre 1996.

[PST97]: Karin Petersen, Mike J. Speitzer Douglas B. terry, Marvin M. Theimer, and Alan j.Demers,
« *Flexible Update Propagation for Weakly Consistent Replication*»,
In Proc, 16th symposium on operating systems principles (sosp,97),1997 .

[QIL 96]: Qi Lu,
«*Improving Data Consistency for Mobile File Access Using Isolation-Only Transactions»,*
PhD Thesis, CMU-CS-96-131, School of Computer,1996.

[RAN02]: Ranjita Bhagwan, David Moore, Stefan Savage et Geoffrey M. Voelker,
«*Replication Strategies for Highly Available Peer-to-Peer Storage*»,
International Workshop on Future Directions in Distributed Computing, Bertinoro, Italy, Juin 2002.

[RBG98]: X. Zeng, R. Bagrodia, and M. Gerla,
«*GloMoSim : a library for parallel simulation of largescale wireless networks»,*
ACM SIGSIM Simulation Digest 28 (1998), no. 1, 154–161

[RCM05]: R.Bruno, C.Chaudet, M.Conti, and E.Gregori,
«*A novel fair medium access control for 802.11-besed multi-hop Ad hoc networks*»,
In 14th IEEE workshop on local and metropolitan area networks (LanMan), September 2005.

[RFB90]: Ronald F. Belanger,
«*MODSIM II — a modular, object-oriented language*»,
(tutorial session) Proceeding WSC' 90 Proceedings of the 22nd conference on Winter simulation, ISBN:0-911801-72-3, IEEE Press Piscataway, NJ, USA 1990.

[RIL03]: Rubin, I. and Liu, Y.-C,
«*Link stability models for QoS Ad hoc routing algorithms»,*
In Proc. 58th IEEE Vehicular Technology Conf., vol. 5, pp. 3084-3088,(2003).

[SAT02]: M. Satyanarayanan,
« *The evolution of Coda*»,
ACM Transactions on Computer Systems (TOCS), 20(2):85–124, 2002.

[SDA07]: Sandor Dornbush and Anupam Joshi,
«*Streetsmart traffic : Discovering and disseminating automobile congestion using VANET's»,*
In VTC Spring, pages 11–15. IEEE,2007.

[SDT97]: M. J. Spreitzer, M. M. Theimer, K. Petersen, A. J. Demers And D. B. Terry,
«*Dealing with Server Corruption in Weakly Consistent, Replicated Data Systems*»,
Proceedings of the 3rd Annual ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom'97), p. 234.240, Budapest, Hungary.

[SDT99]: M. J. Spreitzer, M. M. Theimer, K. Petersen, A. J. Demers,and D. B. Terry,
« *Dealing with Server Corruption in Weakly Consistent, Replicated Data Systems*» ,
Wireless Networks, vol. 5, n_5, 1999.

[SEB03]: Sébastien Monnet,
«*Détection et tolérance aux fautes dans JuxMem*» ,
IRISA /PARIS, Lyon.2003.Presentation powerpoint

[SJE89]: S. J. Eggers and R. H. Katz,
«*The effect of sharing on the cache and bus performance of parallel programs* »,
In Proceedings of the International Conference on Architectural Support for Programming Languages and Operating Systems, 1989.

[SKB07]: Sanjeev Khanna, Keshav Kunal, and Benjamin C. Pierce,
«*A formal investigationof diff3*»,
In Arvind and Prasad, editors, Foundations of Software Technology and Theoretical Computer Science (FSTTCS), December 2007.

[SMS03]: Sheng, M. Li, J. and Shi, Y,
«*Routing protocol with QoS guarantees for ad-hoc network*»,
Electronics Letters, vol. 39, pp. 143.145 (2003).

[SOL99]: projet INRIA : SOLIDOR,
«*Construction de systèmes et d'applications distribués* »,
Rapport de recherche 1999.

[SWR98]: Singh, S., Woo, M. and Raghavendra, C. S,
«*Power-aware routing in mobile Ad hoc networks*»,
In Proc. of Mobile Computing and Networking, pp. 181-190 (1998).

[TCK01]: Toh, C.-K,
«*Maximum battery life routing to support ubiquitous mobile computing in wireless Ad hoc networks*»,
IEEE Trans. Commun, vol. 39, no. 6, pp. 138-147 (2001).

[TPD 95]**:** D.B. Terry, M.M. Theimer, K. Petersen, A.J. Demers, M.J. Spre-itzer and C.H.Hauser,
«*Managing Update Conflicts in Bayou, A Weakly Connected Replicated Storage System*»,
In Proceedings of the 15th Symposium on Operating Systems Principles,
pp. 172-183, Copper Mountain Resort, Colorado, ACM, December 1995.

[WEB01]: OPNET Modeler, http://www.opnet.com/products/ modeler/home.html.

[WEB02]: The Network Simulator–ns-2, http://www.isi.edu/nsnam/ns/index.html.

[WEB03]: GloMoSim, Global Mobile Information Systems Simulation Library.
Availabe from <http://pcl.cs.ucla.edu/projects/glomosim/>.

[WEB04]: «Mobile Ad hoc Networking (*MANET*): Routing Protocol Performance», *www.ietf.org/rfc/rfc2501.txt*

[WEB05]: « *Divergence Control Algorithms for Epsilon Serializability* »,
Disponible sur www-poleia.lip6.fr/~gancarsk/papers/LG-ISI05.pdf.

[WIE87]: G. Wiederhold and X. Qian,
«*Modeling asynchrony in distributed databases »*,
In proceedings of the 3rd International Conference on Data Engineering, pages 246-250, 1987.

[WIE90]: G. Wiederhold and X. Qian,
«*Consistency control of replicated data in federated databases »*,
In Proceedings of the 1st Workshop on the Management of Replicated Data, pages 130-132, Houston, November 1990.

[WMK05]: Wang, M. and Kuo, G.-S,
«*An application-aware QoS routing scheme with improved stability for multimedia applications in mobile Ad hoc networks»*,
In Proc.IEEE Vehicular Technology Conf., pp. 1901-1905, (2005).

[WNT98]: WEBRE S, NIXON P,TANGNEY B,
«*A Flexible Framework for Consistency management in Object Oriented »*,
Distributed Shared memory. TCD-CS-1998-21.Trinity College Dublin.1998.p.21

[XAV98]: Xavier Défago André Schiper Nicole Sergent,
« *Semi passive réplication »*,
livre: Symposium on Reliable Distributed systems ,page 43-50 1998,Département informatique Ecole polytechniques Fédérale de Lausanne Suisse.

[YPD07]: Yawut, C., Paillassa, B. and Dhaou, R,
«*Mobility versus Density Metric for OLSR Enhancement»*,
3rd AINTEC, LNCS Volume 4866, Springer, Berlin (2007).