Gestion de l'accès aux données dans les réseaux sans fil en mode ad hoc

Malika Boulkenafed

N° Ordre
de la thèse :

THESE DE DOCTORAT DE L'UNIVERSITE PARIS 6

Spécialité
Informatique

présentée par

Malika BOULKENAFED

pour obtenir le grade de

DOCTEUR de l'UNIVERSITE PARIS 6

Sujet de la thèse :

Gestion de l'accès aux données dans les réseaux sans fil en mode *ad hoc*

Soutenue le 24 / 11 / 2003 devant la commission d'Examen

Composition du jury :

M.	Guy	PUJOLLE	Président
Mme.	Anne-Marie	KERMARREC	Rapporteurs
M.	Stéphane	UBEDA	
Mme.	Cecilia	MASCOLO	Examinateurs
M.	Titos	SARIDAKIS	
Mme.	Valérie	ISSARNY	

Dédié à mes parents.

Remerciements

Je remercie Guy Pujolle, professeur à l'Université Paris 6, d'avoir accepté le rôle de président du jury.

Je suis très reconnaissante à Anne-Marie Kermarrec, chargé de recherche à Microsoft Research, et à Stéphane Ubéda, professeur à l'INSA-Lyon, d'avoir rapporté ce document de thèse. Leurs remarques pertinentes sur une première version de ce document m'ont permis d'en améliorer la clarté et de considérer des perspectives à mon travail de recherche.

Je remercie également Cecila Mascolo, *lecturer* à *University College London*, et Titos Saridakis, *Principal Scientist* à *Nokia Research Center*, d'avoir bien voulu juger ce travail. Les remarques pertinentes de Mme. Mascolo, ont permis d'apporter plus de précision à ce document de thèse. La collaboration avec M. Saridakis dans le cadre du projet européen VIVIAN a permis d'établir les prémisses de ce travail de recherche.

Je remercie chaleureusement Valérie Issarny, directrice de recherche à l'INRIA-Rocquencourt, qui a dirigé cette thèse, pour son expérience et son dynamisme qui m'ont été très profitables, ainsi que pour ses conseils, sa disponibilité et sa patience, qui m'ont permis d'avancer tout au long de ce travail. J'ai beaucoup appris en travaillant avec elle.

En plus des personnes avec lesquelles j'ai pu coopérer sur ce travail, je remercie celles qui ont assisté à la soutenance.

Je remercie particulièrement Pascal Nassif pour avoir patiemment lu et relu les premières versions de ce document de thèse et d'en avoir décellé les si nombreuses fautes d'orthographes.

Merci à mes parents proches et amis, pour leur soutien, leur présence, et leur compréhension qui m'ont permis d'accomplir ce travail dans de bonnes conditions.

Résumé

La flexibilité et la diversité des réseaux mobiles sans fil offrent de nombreuses opportunités qui ne sont pas toujours prises en compte par les systèmes distribués existants. En particulier, la prolifération des utilisateurs mobiles et l'utilisation des réseaux mobiles sans fil en mode *ad hoc* promeut la formation de groupes collaboratifs pour le partage des ressources.

Nous proposons une solution pour la gestion de l'accès aux données qui intègre les fonctionnalités nécessaires pour le partage des ressources en environnement mobile, en exploitant les différentes connectivités offertes par les réseaux mobiles sans fil. Nous proposons une gestion hybride de la cohérence en fonction de la connectivité du réseau. Cette gestion nous permet de supporter aussi bien une collaboration synchrone qu'asynchrone, où nous considérons que la collaboration synchrone dépend de la proximité géographique. Enfin, nous proposons une gestion de la disponibilité des données basée sur une gestion adaptative de la réplication qui permet d'anticiper les déconnexions des entités participant à une collaboration synchrone. Notre solution tient compte de la disponibilité des ressources au niveau des entités mobiles et tente de réduire la consommation d'énergie, le tout sans intervention explicite de l'utilisateur.

Abstract

The combination of short-range wireless networking and today's mobile device capabilities offer great potential, but is not yet well supported by existing distributed systems. In particular, the widespread deployment of lightweight wireless computing devices, and the use of mobile ad hoc wireless networks promote resource sharing within collaborative groups.

We provide a solution for data access management that includes needed functionalities for resource sharing in a mobile environment, and takes advantage of different connectivities offered by today's mobile wireless networks. Our solution allows collaborative data sharing among ad hoc mobile groups that are dynamically formed according to the connectivity achieved by the ad hoc WLAN. Furthermore, we enhance data availability within mobile ad hoc collaborative groups by transparently anticipating devices disconnections. Our solution is based on a new adaptive data replication protocol, combining both optimistic and conservative schemes so as to best deal with synchronous/asynchronous collaboration, network connectivity and devices' local resources while minimizing energy and storage consumption. Finally, we present the integration of our solution in a distributed mobile files system. This enabled us to validate non functional properties of our proposal.

Table des matières

Table des figures

Liste des tableaux

I Introduction

Dans les années 70 et le début des années 80, les systèmes et applications informatiques développés étaient conçus pour des ordinateurs stationnaires. Vers la fin des années 80, les ordinateurs, dits *portables*, firent leur apparition. En réalité, ces premiers ordinateurs *portables* étaient à peine *transportables*. Leur coût prohibitif comparé aux performances qu'ils affichaient les rendait peu attractifs pour l'utilisateur, surtout que leur *mobilité* restait très restreinte. Ainsi, les ordinateurs stationnaires ont trôné jusqu'au début des années 90, où la miniaturisation des composants informatiques a donné naissance à des ordinateurs réellement portables tout en offrant des performances acceptables comparées aux ordinateurs stationnaires. Mais, il n'en reste pas moins que les ordinateurs portables affichent toujours des ressources comparativement plus faibles que les ordinateurs stationnaires, qui eux n'ont aucune contrainte relative à leurs poids et taille. A titre d'exemple, les capacités de stockage des ordinateurs de poche sont nettement inférieures à celles des ordinateurs stationnaires. De plus, les ordinateurs portables doivent avoir recours à leur batterie lors des périodes de mobilité, ce qui leur octroie une autonomie relativement limitée.

Parallèlement au développement du matériel informatique pour la mobilité, au début des années 80, les réseaux d'accès sans fil *via* les fréquences radio ont instillé l'idée du transport d'information *via* les réseaux sans fil. Cependant, ces réseaux, tels que les téléphones de résidence sans fil, offraient un accès dans une zone géographiquement restreinte à quelques dizaines de mètres et n'étaient pas assez efficaces pour transporter d'autres formes d'informations que la parole telles que les données informatiques ou la vidéo [1]. Vers le début des années 90, les réseaux de deuxième génération (2G) tels que le GSM (Groupe Spécial Mobile puis *Global System for Mobile communication*) ont offert aux utilisateurs des systèmes de téléphonie mobile universels. Ils ont également été les acteurs de la convergence des mondes de l'informatique et des télécommunications. En effet, les standards de téléphonie mobile évoluent sans

cesse pour réaliser du transfert de données. Une évolution majeur du GSM est normalisée sous le nom de GPRS (*General Packet Radio Service*) qui introduit une architecture réseau basée sur la transmission de paquets. Par ailleurs, la norme EDGE (*Enhanced Data for GSM Evolution*) offre des débits supérieurs par l'introduction d'une modulation plus efficace et applicable aux GSM et GPRS. Les normes GPRS et EDGE sont considérées comme intermédiaires entre les systèmes 2G et 3G. Les réseaux 2G ont été victimes de leur succès auprès des utilisateurs. C'est pourquoi, vers les années 2000, les réseaux 3G firent leur apparition et notamment le réseau UMTS (*Universal Mobile Telecommunications System*). Leur apparition a pour origine, à la fois la saturation des réseaux 2G, et le besoin d'une couverture universelle ainsi que de services évolués tendant vers ceux offerts par les infrastructures câblées [1].

Cependant, ces réseaux, qu'ils soient 2G ou 3G, utilisent des bandes de fréquence sous licence. Ainsi, pour en bénéficier, les utilisateurs doivent s'abonner auprès des opérateurs de téléphonie mobile. C'est en 1999 que le groupe de travail IEEE 802.15 intitulé WPAN (*Wireless Personal Area Network*) normalise des réseaux qui utilisent une bande de fréquence libre (2.4-2.5 GHz). Ainsi, des groupements industriels, basés sur ces standards, se sont mis en place, tels Bluetooth et HomeRF (*The Home Radio Frequency*), pour permettre le déploiement rapide de réseaux sans fil à faible coût sur des zones géographiquement limitées. Le groupe HomeRF, dédié à la domotique[1], a été dissolu au profit des standards Bluetooth et IEEE 802.11 publié vers 2001. IEEE 802.11 est le premier standard international pour les réseaux locaux sans fil (WLAN pour *Wireless Local Access Network*). Son but est de fournir une connectivité sans fil à des ordinateurs stationnaires ou mobiles qui nécessite un déploiement rapide dans une zone géographiquement limitée. La bande de fréquence choisie à l'instar de celle des WPANs est libre. De plus, les WLANs offrent un débit nettement supérieur à celui des réseaux 3G (11 Mbps pour IEEE 802.11 contre 2 Mbps prévu pour l'UMTS). Ils bénéficient également d'une interface réseau très flexible qui leur permet de fonctionner en mode infrastructure, mais également en mode *ad hoc* qui supporte la communication directe sans recours à une infrastructure réseau. Parallèlement, divers protocoles de routage pour les réseaux *ad hoc* ont été élaborés au sein du groupe MANET (*Mobile Ad hoc NETworks*) pour étendre la couverture des réseaux en mode *ad hoc*. Les réseaux locaux sans fil connaissent actuellement un engouement important du fait de la flexibilité de leur interface et sont en cours de déploiement sur une grande

[1] l'appellation domotique regroupe l'ensemble des technologies de l'électronique, de l'informatique et des télécommunications utilisées dans les domiciles.

échelle. Par ailleurs, la technologie Wi-Fi (*Wireless Fidelity*), nom commercial pour le standard IEEE 802.11b, est envisagée comme une alternative aux réseaux 3G. L'idée est de permettre de téléphoner et d'accéder à des services multimédias à des débits largement supérieurs à ceux des réseaux 3G et à un coût nettement inférieur [112, 80].

L'évolution des réseaux sans fil et l'avènement de machines légères ont eu un impact important sur le comportement des utilisateurs des systèmes informatiques qui sont devenus de plus en plus mobiles. Par ailleurs, les systèmes distribués conçus pour des environnements stationnaires sont, d'une part, devenus inadaptés aux contraintes apportées par les environnements mobiles et, d'autre part, ne permettent pas d'exploiter les nouvelles fonctionnalités offertes pas ces environnements. Les contraintes apportées par les environnements mobiles concernent particulièrement les caractéristiques des ordinateurs mobiles relatives à leurs ressources locales et à celles du support de transmission en l'occurence les ondes hertziennes. En effet, les ordinateurs mobiles possèdent des ressources intrinsèques (espace de stockage, puissance de calcul, autonomie de la batterie) de capacités très variables, pouvant varier de ressources comparables à celles des PC, notamment dans le cas des ordinateurs portables, jusqu'à des ressources nettement inférieures, notamment dans le cas des ordinateurs de poche. De plus, le support de transmission sans fil offre une bande passante réduite comparée au support de transmission câblé, ainsi qu'une portée limitée qui se traduit par l'atténuation du signal générant, entre autres, une connectivité intermittente. Cependant, les environnements mobiles constituent un support adéquat à de nombreuses facilités inexistantes dans les environnements stationnaires, telles que la mobilité des utilisateurs, l'accès ubiquitaire aux ressources distantes et la création d'un réseau local sans fil temporaire grâce au mode *ad hoc*. De ce fait, de nombreux travaux ont porté sur la conception de systèmes distribués adaptés aux environnements mobiles pour tenter d'offrir aux utilisateurs et à leurs applications des fonctionnalités qui leur permettent d'évoluer dans ces environnements.

I.1 Systèmes distribués pour environnements mobiles

Un système distribué est un ensemble de composants logiciels localisés sur un ensemble d'ordinateurs autonomes reliés entre eux par un réseau informatique. L'ensemble des ressources locales des ordinateurs, telles que, l'espace de stockage ou la

puissance de calcul, et l'ensemble des ressources externes relatives au réseau sous-jacent, telles que la connectivité ou la bande passante constituent le contexte d'exécution du système distribué. Indépendamment de leur contexte d'exécution, les systèmes distribués doivent offrir des fonctionnalités qui permettent [85, 29] :

- La gestion des ressources *accessibles*, qu'elles soient matérielles (espaces de stockage, *etc.*) ou logicielles (bases de données, systèmes de fichiers, *etc.*).
- La gestion des interactions entre les ressources du système.

Par ailleurs, les systèmes distribués doivent garantir la qualité des fonctionnalités qu'ils implémentent. Citons notamment la disponibilité qui concerne en particulier les données et la sécurité des communications qui est souvent réalisée au détriment des performances du système.

Dans le cas des systèmes distribués dédiés aux environnements mobiles, leurs fonctionnalités sont adaptées aux caractéristiques des ordinateurs mobiles et à la variation de la bande passante, incluant la connectivité intermittente. La gestion des ressources accessibles suppose d'abord l'identification de ces dernières dans l'environnement mobile. Ainsi, on retrouve dans la littérature différents protocoles pour la découverte de l'environnement qui servent à identifier automatiquement les ressources accessibles à un composant du système et qui correspondent à ses besoins [10]. En outre, les systèmes distribués existants distinguent entre la gestion des ressources logicielles et celle des ressources matérielles. Du fait de la connectivité intermittente dans les environnements mobiles, la réplication des ressources logicielles et en particulier des données est utilisée pour accroître leur disponibilité et permettre aux utilisateurs mobiles de travailler hors connexion. Plusieurs techniques ont été proposées dans la littérature pour la gestion de la réplication des données. Certains systèmes se sont basés sur des techniques de préchargement [61, 62, 43, 92]. D'autres préconisent la réplication selon les accès effectifs. Enfin, des systèmes proposent d'anticiper les changements du contexte d'exécution, notamment dus aux déconnexions, grâce aux indications des utilisateurs [77]. Par ailleurs, la gestion de la cohérence des réplicas dans les systèmes distribués mobiles est basée sur des protocoles de cohérence faible qui permettent de manipuler de manière indépendante et concurrente les réplicas d'une donnée. En ce qui concerne la gestion des autres ressources locales des ordinateurs mobiles, certains systèmes adaptent la réplication des données à la disponibilité de l'espace de stockage en offrant différentes granularités de réplication [109, 107, 78]. D'autres systèmes permettent aux applications d'intervenir au niveau de la gestion des ressources [118]. Toutefois, aucun des systèmes existants ne pro-

pose un modèle de réplication qui permet d'anticiper les changements du contexte d'exécution de manière transparente à l'utilisateur. De plus, rares sont les systèmes distribués qui adaptent leurs fonctionnalités de gestion des ressources à l'énergie disponible ou tentent de réduire sa consommation.

La gestion des interactions dans un système distribué repose principalement sur la définition du modèle de communication associé (clients/serveur stricte, clients/serveur étendu, pair-à-pair, *publish/subscribe*, *etc.*). Certains modèles de communication sont plus adaptés aux environnements mobiles. Dans les modèles centralisés, les communications doivent être adressées à une entité centrale (p.ex., serveur), ce qui n'est pas approprié en la présence d'une connectivité intermittente. Par conséquent, les systèmes distribués mobiles basés sur ces modèles se focalisent sur la prise en charge de la connectivité intermittente, en ayant principalement recours à la réplication des données et/ou des calculs sur les entités mobiles. Les modèles décentralisés s'avèrent plus adaptés aux environnements mobiles, de par la possible communication entre n'importe quelle entité. L'absence d'entités centrales favorise, en outre, l'extensibilité du système.

I.2 Motivations

La flexibilité et la diversité des réseaux mobiles sans fil offrent de nombreuses opportunités qui ne sont pas toujours prises en compte par les systèmes distribués existants. La prolifération des utilisateurs mobiles et l'utilisation des réseaux mobiles sans fil en mode *ad hoc* promeut notamment la formation de groupes collaboratifs [7]. Les formes de collaboration sont très variées mais elles impliquent toujours un groupe de personnes partageant un intérêt commun. Ces groupes de personnes forment des communautés qui partagent certaines ressources incluant les données. Les systèmes distribués mobiles proposés dans la littérature considèrent pour la plupart qu'une entité en situation de mobilité ne doit avoir recours qu'à ses ressources locales si la couverture du réseau ne permet pas d'établir une connexion sans fil à des ressources distantes prédéterminées. Ces systèmes n'exploitent pas le partage des ressources dans des groupes d'entités mobiles appartenant à une communauté formée autour d'un centre d'intérêt. Cependant, un tel partage des ressources permet d'avoir recours à des ressources mises à disposition par des entités paires qui se trouvent dans la portée de communication du réseau local en mode *ad hoc*.

A titre d'exemple considérons le scénario suivant : une réunion de travail se tient dans une salle de réunions. Les participants à ce type de réunion apportent généralement leurs données sur leurs ordinateurs portables ou leurs *PDAs*. Ainsi, dans la salle de réunion se trouvent des ordinateurs de taille, de capacité de stockage et d'autonomie de batterie, très hétérogènes. De plus, ils stockent localement sur leur ordinateur les données liées à leur contribution à la réunion. Pendant la réunion, les participants ont besoin d'accéder aux contributions de chacun. Ceci peut être réalisé en utilisant une disquette, un stick mémoire, ou encore le *e-mail*, afin de récupérer des copies des données nécessaires, mais ne permet pas le maintien de la cohérence des données ainsi répliquées. Les participants doivent pouvoir échanger leurs données *via* le réseau sans fil, moyennant quelques garanties quant à la sécurité des communications, même si aucune infrastructure de réseau sans fil n'est disponible. La solution consiste à utiliser un réseau local sans fil en mode *ad hoc*. Par ailleurs, les participants à la réunion ont des emplois du temps très variés. Certains d'entre eux peuvent quitter la réunion avant la fin, risquant de compromettre la disponibilité des données nécessaires pour la suite de la réunion. L'indisponibilité des données peut être également provoquée par des déconnexions involontaires suite, par exemple, à une panne de batterie. Ainsi, un système de gestion des données partagées est nécessaire afin de supporter la collaboration dans des groupes de travail. Il doit permettre d'augmenter la disponibilité des données partagées sans contraindre la mobilité des membres du groupe et sans intervention explicite de l'utilisateur. Ces fonctionnalités doivent, en outre, être implémentée en veillant à réduire la consommation des ressources critiques des terminaux mobiles.

Bien que l'exemple décrit ci-dessus soit simple, aucun des systèmes distribués mobiles existants n'offrent toutes les fonctionnalités requises. Le support de la collaboration synchrone qui nécessite une gestion forte de la cohérence, et la réplication préventive des données sans intervention de l'utilisateur afin d'anticiper les changements du contexte d'exécution ne sont, en particulier, pas abordées par ces systèmes. Par ailleurs, aucun des systèmes distribués existants n'a traité la gestion de l'accès aux données en se souciant de la disponibilité de l'énergie qui est une ressource critique quant il s'agit d'ordinateurs de poche de faible autonomie. Au regard de ces limitations et des opportunités offertes par les réseaux mobiles sans fil, en particulier par le mode *ad hoc* de ces réseaux, nous établissons les fonctionnalités principales qu'un système distribué mobile doit intégrer pour exploiter au mieux les ressources disponibles dans l'environnement mobile ainsi que les différentes connectivités pos-

sibles. Ces fonctionnalités relèvent de :

- la gestion transparente du réseau local en mode *ad hoc*, qui permette : (i) la création spontanée d'un réseau local formé à partir d'un ensemble d'entités *autorisés* (appartenant à la même communauté) et (ii) la prise en charge de la dynamicité du réseau de telle sorte, que ses membres puissent se déconnecter, ou que des entités *autorisés* puissent s'y connecter à tout moment,
- la gestion de l'accès aux données adaptée aux contraintes de la mobilité et des réseaux sans fil qui permette : (i) une collaboration synchrone ou asynchrone selon la connectivité du réseau, (ii) une gestion de la cohérence des données adaptée à la connectivité du réseau, et (iii) une réplication préventive des données afin d'anticiper les déconnexions volontaires ou involontaires,
- la gestion des ressources, qui adapte les fonctionnalités précédentes à la disponibilité des ressources et qui permette de réduire la consommation de l'énergie.

I.3 Contribution

Etant données les motivations décrites dans la section I.2, nous proposons une solution pour la gestion de l'accès aux données qui intègre les fonctionnalités nécessaires pour le partage des ressources dans un environnement mobile en exploitant les différentes connectivités offertes par les réseaux sans fil. Notre contribution tient compte des caractéristiques des terminaux mobiles dans le but de réduire la consommation de leurs ressources critiques. Elle est en outre adaptée aux applications qui nécessitent une collaboration synchrone ou asynchrone.

Notre solution est basée sur le modèle de communication pair-à-pair et introduit la notion de groupe pour le partage des ressources. Un groupe définit un ensemble d'entités mobiles, choisies selon un critère déterminant leur appartenance à une communauté et formant un réseau local en mode *ad hoc*. Par ailleurs, nous proposons une gestion hybride de la cohérence en fonction de la connectivité du réseau. Cette gestion nous permet de supporter aussi bien une collaboration synchrone qu'asynchrone, où nous considérons que la collaboration synchrone dépend de la proximité géographique. Enfin, nous présentons la gestion de la disponibilité des données à travers une gestion adaptative de la réplication. En plus de la réplication des données selon les accès effectifs, à l'instar des solutions existantes, nous augmentons la disponibilité des données au moyen d'une réplication préventive qui permet d'anticiper les

déconnexions des entités qui participent à un travail de collaboration synchrone sans intervention explicite des utilisateurs. Notre solution tient compte de la disponibilité des ressources au niveau des entités mobiles et tente de réduire la consommation d'énergie en réduisant les échanges de messages *via* les communications sans fil.

I.4 Structure du document

Le document comprend sept chapitres. Le chapitre II détaille les fonctionnalités qu'un système distribué dédié aux environnements mobiles doit offrir, considérant notamment l'impact des réseaux mobiles sans fil sur la gestion de l'accès aux données. Le chapitre III étudie les solutions proposées dans la littérature par rapport aux exigences décrites dans le chapitre précédent. Nous présentons ensuite notre contribution à la gestion de l'accès aux données dans des environnements mobiles, qui fait ressortir les fonctionnalités nécessaires qui font défaut aux différents systèmes distribués de l'état de l'art. Dans le chapitre IV, nous décrivons notre gestion de groupe qui permet à des entités mobiles, partageant un intérêt commun, de former des groupes de façon spontanée dès que la connectivité le permet, n'importe où, n'importe quand et cela sans recours aux infrastructures réseaux existantes. Ce chapitre fait ressortir les fonctionnalités qui offrent un support pour la collaboration dans les environnements mobiles et qui plus ait sont adaptées aux réseaux mobiles sans fil en mode *ad hoc*. Dans le chapitre V, nous décrivons notre gestion hybride de la cohérence qui s'adapte à la connectivité du réseau sous-jacent ainsi qu'aux besoins des applications de collaboration. Dans le chapitre VI, nous introduisons notre contribution pour l'augmentation de la disponibilité des données où nous proposons un modèle de réplication adaptative qui prend en compte la disponibilité des ressources de chaque entité mobile. Dans le chapitre VII, nous proposons une évaluation de notre contribution en termes des temps de réponses, de la consommation d'énergie et de l'espace de stockage à travers son implémentation dans un prototype de système de fichier dédié aux environnements mobiles. Enfin, nous concluons ce document par un résumé des contributions de notre travail, ainsi que par les perspectives de recherche qu'il suscite.

II Systèmes distribués mobiles : caractéristiques et fonctionnalités

La mobilité peut être considérée de différents points de vue. Elle peut être sociale, physique, matérielle ou logique [63, 103, 38]. Dans le cadre de notre travail, nous nous intéressons à la mobilité physique où les entités mobiles[1] se déplacent dans un espace physique.

Les utilisateurs mobiles sont équipés d'ordinateurs qui facilitent la mobilité physique, c.-à-d., des machines de petite taille, légères et facilement transportables. Les utilisateurs mobiles ont également tendance à changer d'ordinateur en fonction de la situation dans laquelle ils se trouvent. Un utilisateur peut travailler sur un ordinateur portable au bureau, sur son PC à domicile et *vice versa* et prendre uniquement son PDA lors de ses déplacements. Cette mobilité matérielle est de plus en plus fréquente. Ainsi, les systèmes distribués dédiés aux environnements mobiles sont confrontés à des ordinateurs hétérogènes tant par leur taille que par leurs ressources. De plus, les capacités de stockage des terminaux mobiles tels que les PDAs ou les téléphones intelligents, restent insuffisantes par rapport aux besoins toujours croissants des utilisateurs, ainsi qu'aux différents types de données (multimédia, 3D, *etc.*) rendus accessibles grâce au progrès enregistré dans les technologies de transmission sans fil. Un autre point faible des terminaux mobiles réside dans l'autonomie relativement faible de leur batterie. Sachant que les utilisateurs en situation de mobilité n'ont pas toujours l'occasion de brancher leur ordinateur à des prises électriques, ceci incite au développement de protocoles permettant l'économie de l'énergie, tant au niveau des couches réseaux qu'aux niveaux plus élevés comme le système d'exploitation, le *middleware* ou l'applicatif.

[1] Le terme *entité mobile* fait référence au couple (utilisateur, ordinateur mobile associé).

Les réseaux mobiles sans fil sont composés de plusieurs équipements comprenant les satellites, les antennes, les commutateurs, les terminaux, *etc.*, ainsi que des interfaces entre ces équipements pour assurer la communication. La notion de réseau mobile fait référence à la possibilité de déplacement d'une entité mobile d'un réseau vers un autre tout en gardant la capacité de communiquer au sein du nouveau réseau. A ce jour, la mobilité n'est autorisée qu'au niveau de réseaux partageant un même standard, tel que le réseau GSM. La notion de réseau sans fil est, quant à elle, étroitement associée au support de transmission (infrarouge, ondes hertziennes, *etc.*). La communication dans les premiers réseaux sans fil était organisée autour de cellules qui définissaient une zone de couverture relative à une station de base. A présent, la communication directe ou par routage entre terminaux mobile est également possible.

Dans ce chapitre, nous abordons les caractéristiques des systèmes distribués dans le contexte des environnements mobiles. Nous mettons en avant les fonctionnalités que ces systèmes doivent fournir pour répondre aux exigences des applications dédiées à de tels environnements. Dans la section II.1, nous analysons les caractéristiques des réseaux mobiles géographiquement limités que nous considérons comme les plus avantageux de par leur facilité de déploiement et leur débit élevé. Dans la section II.2, nous nous concentrons sur les fonctionnalités que tout système distribué dédié aux environnements mobiles doit fournir. Ces fonctionnalités relèvent de la la découverte de l'environnement, du modèle de communication, de l'accès aux données, de la gestion des ressources locales et de la sécurité informatique. Enfin, nous concluons ce chapitre par une discussion, dans la section II.3, qui met en avant l'impact des terminaux mobiles ainsi que celui des réseaux mobiles sans fil sur les fonctionnalités des systèmes distribués mobiles, en particulier sur la gestion de l'accès aux données.

II.1 Réseaux mobiles sans fil

Dans cette section, nous abordons les caractéristiques des réseaux mobiles sans fil, et plus spécifiquement celles des réseaux géographiquement limités, car nous considérons que ces réseaux constituent une alternative privilégiée aux réseaux 3G pour l'accès aux données dans un contexte ubiquitaire. En effet, ces réseaux supportent l'accès à des services multimédias à des débits pouvant atteindre les 108 Mbps (dans

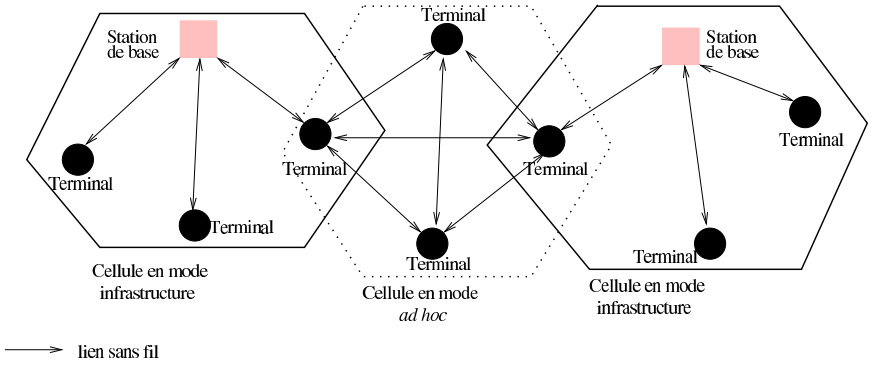

Figure II.1 – Schéma de connexion dans Wi-Fi.

le cas du WLAN IEEE 802.11a turbo), contre 2 Mbps prévus pour l'UMTS. De plus, ils utilisent des bandes de fréquence libres (sans abonnement) et leurs interfaces sont intégrées dans tous les terminaux mobiles récents à des coûts réduits [112, 80], ce qui les rend particulièrement attractifs pour les utilisateurs. Les réseaux Wi-Fi, Bluetooth et les réseaux *ad hoc* sont considérés, à ce jour, comme des exemples typiques de ces réseaux.

II.1.1 Wi-Fi

L'architecture d'un réseau Wi-Fi, basé sur le standard IEEE 802.11b, est cellulaire. Dans ce type de réseau local sans fil, les communications peuvent se faire soit directement, de terminal à terminal, soit en passant par une station de base. Ainsi, le standard IEEE 802.11b offre deux modes de fonctionnement, le mode infrastructure et le mode *ad hoc*. Le mode infrastructure est défini pour fournir aux différents terminaux mobiles des services spécifiques sur une zone de couverture déterminée par les stations de base. Ces dernières peuvent par exemple fournir une passerelle d'accès vers un réseau fixe, tel que l'Internet. Un réseau en mode *ad hoc* est un réseau de terminaux mobiles communiquant sans le support d'une quelconque infrastructure. Chaque terminal peut établir une connexion directe avec n'importe quel autre terminal dans sa portée de communication. Ce mode de fonctionnement est très utile pour mettre en place facilement, rapidement et à moindre coût un réseau sans fil. La figure II.1 illustre les deux modes de connexion possible entre terminaux dans un réseau Wi-Fi. Dans cette figure, deux cellules sont en mode infrastructure et une cellule est en mode *ad hoc*.

Dans les réseaux basés sur le standard IEEE 802.11, les terminaux peuvent se déplacer d'une cellule vers une autre sans interruption (*handover*). Quand un terminal veut accéder à une station de base, il la choisit d'après la puissance du signal, le taux d'erreur ou la charge du réseau. Ce choix peut s'effectuer, soit à travers l'écoute passive, soit à travers l'écoute active. En écoute passive, le terminal attend de recevoir un message *Beacon Frame* venant d'une station de base. Ce mode favorise l'économie d'énergie. En écoute active, le terminal envoie un message de demande d'association à la station de base la plus appropriée et attend qu'elle lui réponde pour s'associer. Ce mode affiche des performances meilleures que l'écoute passive en terme de latence nécessaire pour s'associer à une station de base, mais consomme plus d'énergie.

En ce qui concerne la prise en charge de la sécurité des communications dans le standard IEEE 802.11, elle est renforcée par un mécanisme d'authentification des terminaux auprès des stations de base. Les mécanismes d'économie d'énergie ou de sécurité des communications sont proposés uniquement aux terminaux en mode infrastructure [56].

II.1.2 Bluetooth

Le but de Bluetooth, dont la portée initiale était d'une dizaine de mètres, est de réaliser des connexions sans fil entre différents terminaux à un coût réduit et en abaissant la consommation d'énergie. Par ailleurs, Bluetooth ne cesse d'être amélioré, notamment en ce qui concerne les débits et la portée de communication. Il offre également à ces utilisateurs la possibilité de superposition avec des réseaux basés sur le standard IEEE 802.11.

Bluetooth définit deux schémas de connexion qui reposent sur le modèle de communications maître-esclave. Le premier schéma correspond à un réseau unique, appelé *piconet* (Fig. II.2-[a]), qui peut prendre en charge un maître et sept terminaux au statut d'esclave. Le terminal maître gère les communications avec les différents esclaves. La communication entre les terminaux esclaves transite obligatoirement par le terminal maître. Le second schéma consiste à interconnecter des réseaux *piconet* formant ainsi un *scatternet* (Fig. II.2-[b]). Pour permettre le routage entre différents *piconet*, un maître peut être esclave dans un autre *piconet* (Fig. II.2-[c]) et un esclave peut être connecté à plusieurs maîtres (Fig. II.2-[d]).

Les terminaux Bluetooth peuvent être dans différents états du point de vue de

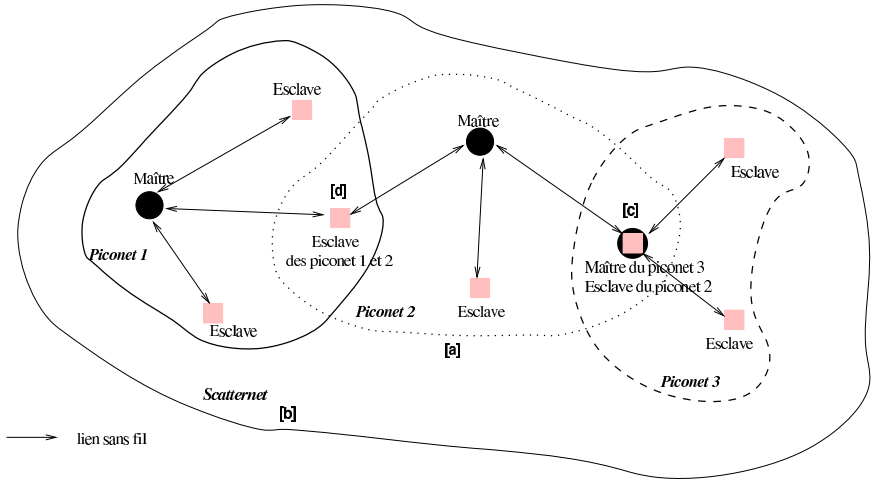

Figure II.2 – Schéma de connexion dans Bluetooth.

la communication, afin de réduire la consommation d'énergie. L'état *park* indique que le terminal ne peut ni recevoir ni émettre, mais il consulte ponctuellement les messages émis par son maître. L'état suspendu (*hold*) indique que le terminal ne peut recevoir que des communications synchronisées. Dans cet état, le terminal se met en veille entre les instants synchrones de réception des messages. L'état de repos actif (*snif*) permet au terminal de déterminer les tranches de temps pendant lesquelles il est actif et celles où il est à l'état de repos.

II.1.3 Réseaux ad hoc

Dans les réseaux dits mobiles, tels que Wi-Fi en mode infrastructure, le GSM, l'UMTS, *etc.*, seuls les terminaux sont soumis au mouvement. Le réseau défini par l'ensemble des stations de base est figé. Dans les réseaux dits *ad hoc*, le réseau en lui même est mobile. En effet, des terminaux mobiles équipés d'interfaces sans fil peuvent dès leur initialisation former un réseau sans recours à aucune infrastructure. Dans les réseaux *ad hoc*, l'acheminement des messages peut être limité à l'envoi directe (un saut) ou étendu au routage (multi-sauts).

Suivant l'envoi direct (Fig. II.3-[a]), notamment supporté par les réseaux Wi-Fi en mode *ad hoc*, l'émetteur envoie ses données directement au terminal destinataire. Les terminaux mobiles doivent être dans la portée de communication directe (suffisamment proches) les uns des autres pour que le signal reçu ne soit pas trop atténué.

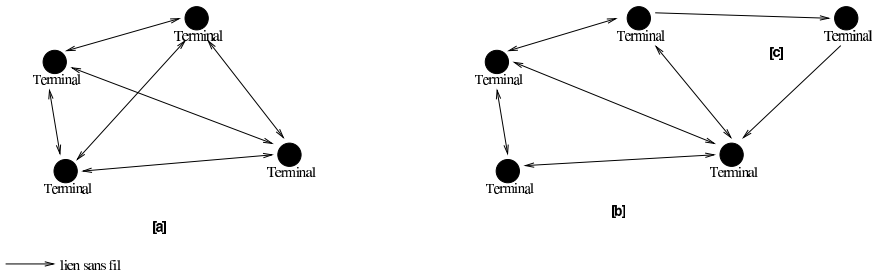

Figure II.3 – Schéma de connexion dans les réseaux ad hoc.

Suivant l'envoi par routage (Fig. II.3-[b]), les communications peuvent avoir lieu de manière indirecte, *via* d'autres terminaux. Dans ce type de configuration, les terminaux mobiles jouent à la fois le rôle de récepteur et de routeur relayant les messages vers leur destination finale.

Les réseaux *ad hoc* offrent un avantage indéniable par rapport à la facilité et la rapidité de leur installation, mais également par rapport au coût réduit du matériel nécessaire. Toutefois, un certain nombre d'obstacles doit être surmonté pour bénéficier de ces avantages. D'une part, certains liens dans les réseaux *ad hoc* sont asymétriques, c.-à-d., les communications sont à sens unique entre un émetteur et un récepteur (Fig. II.3-[c]). D'autre part, du fait que tous les terminaux sont constamment à l'écoute du support de transmission, les interférences détériorent les communications et augmentent le taux d'erreur ainsi que la consommation d'énergie au niveau de tous les terminaux. Enfin, la mobilité des terminaux modifie la topologie du réseau et transforme les tracés de routage, ce qui nécessite, entre autres, la mise à jour régulière des tables de routage au niveau des entités mobiles.

Divers protocoles de routage pour les réseaux *ad hoc* ont été élaborés au sein du groupe MANET (*Mobile Ad hoc NETworks*). On retrouve deux grandes familles de protocoles, les protocoles réactifs et les protocoles proactifs.

- Les protocoles réactifs n'échangent aucun paquet de contrôle pour construire les tables de routage. Ils procèdent à une inondation. Pour l'envoi d'un message, le terminal source l'envoie à tous ses voisins, qui à leur tour, le renvoient à leurs voisins jusqu'à ce que le message ait atteint sa destination. Le terminal destinataire reçoit le message en plusieurs exemplaires. Dans cette famille on retrouve les protocoles AODV [99], TORA [95, 96] et DSR [54].
- Les protocoles proactifs maintiennent des tables de routage, au niveau de chaque terminal, en échangeant de façon continue et à intervalles réguliers des mes-

sages de contrôle. Lorsqu'un terminal envoie un message vers une destination, les tables de routage permettent de déterminer la route optimale. Dans cette famille on retrouve les protocoles DSDV [98], WRP [87, 88] et OLSR [51, 24].

Dans des réseaux denses, les protocoles réactifs sont très coûteux du fait de l'inondation. En revanche, dans des réseaux épars, les protocoles proactifs échangent inutilement des messages de contrôle pour le maintien des tables de routage qui varient légèrement. Des protocoles hybrides ont été proposés afin d'offrir un bon compromis entre protocoles réactifs et protocoles proactifs ; citons par exemple ZRP [97] et CBRP [52]. Dans le protocole ZRP, chaque terminal mobile maintient proactivement des tables de routage concernant ses voisins proches (déterminés par un nombre de sauts fixe). Pour les messages destinés à des terminaux qui se trouvent en dehors de la zone de voisinage, on fait appel à une forme d'inondation contrôlée où les terminaux de la périphérie de cette zone vérifient si le terminal destination appartient à leur zone et ainsi de suite. Ainsi, ZRP est localement proactif et globalement réactif. Le protocole CBRP, quant à lui, divise l'ensemble des terminaux mobiles en *clusters* qui peuvent être disjoints ou interconnectés. Au niveau de chaque *cluster*, un terminal mobile est désigné pour maintenir les tables de routage concernant les autres terminaux du *cluster*. Les tables de routage inter-*cluster* sont construites dynamiquement en exploitant les tables locales à chaque *cluster*. Il existe également des protocoles de routage *multicast* pour les réseaux *ad hoc* parmi lesquels on peut citer : MAODV [115], AMRIS [127], AMRoute [14] et MCEDAR [120].

Les réseaux en mode *ad hoc* sont très adaptés aux communications locales (dans la limite de un ou deux sauts) du point de vue de leurs performances et de leur coût réduit. Mais, au delà des communications locales, le recours aux stations de base est préférable car les protocoles de routage *ad hoc* qu'ils soient proactifs ou réactifs voient leurs performances amoindries. L'un des arguments forts pour les réseaux *ad hoc* est leur utilisation dans des situations où l'infrastructure est inexistante. Cependant, il est intéressant de combiner protocoles de routage *ad hoc* et mode infrastructure de sorte que les messages soient relayés par le biais de ces protocoles jusqu'à ce qu'une station de base soit accessible et prenne en charge l'acheminement des messages. Nous considérons par conséquent que les systèmes distribués mobiles doivent s'appuyer sur une architecture de réseau hybride comprenant des sous réseaux en mode *ad hoc* interconnectés *via* une infrastructure [23].

II.2 Fonctionnalités des systèmes distribués mobiles

Dans cette section, nous présentons les fonctionnalités que tout système distribué dédié aux environnements mobiles doit fournir à ses applications. Ces fonctionnalités concernent en particulier la découverte de l'environnement mobile, le modèle de communication, l'accès aux données distantes, la gestion des ressources et la sécurité.

II.2.1 Découverte de l'environnement mobile

La mobilité physique génère des situations où une entité mobile se retrouve dans des environnements inconnus ne possédant aucune information sur les ressources matérielles et logicielles accessibles. Il est essentiel que ces entités soient informées en permanence des différentes ressources qu'offre l'environnement dans lequel elles se trouvent. La découverte de ressources est de plus en plus répandue grâce aux différents protocoles proposés dans la littérature (p.ex., SLP, JINI, UPnP). Ces protocoles servent à identifier automatiquement les ressources accessibles à une entité et qui correspondent à ces besoins. Pour ce faire, chaque ressource doit faire une annonce en y incluant des détails sur ses caractéristiques et les informations nécessaires pour y accéder. L'entité mobile, quant à elle, lance une requête de découverte de ressources en spécifiant la nature de la ressource ainsi que les caractéristiques recherchées (p.ex., une entité cherche l'imprimante couleur la plus proche).

La découverte de ressources joue un rôle essentiel dans les réseaux sans fil en mode *ad hoc* où elle permet à toutes les entités mobiles formant ce réseau de se faire connaître les unes auprès des autres, ou plus précisément de faire connaître les ressources qu'elles offrent. La nature dynamique de ce type de réseau exclut une localisation centralisée, où le répertoire des ressources disponibles dans le réseau sous-jacent est géré par une même entité.

II.2.2 Modèle de communication

Chaque système distribué intègre un modèle de communication qui est déterminant pour la mise en oeuvre des fonctionnalités du système distribué et en particulier la gestion de l'accès aux données. Dans la littérature, on recense deux modèles de communication : (i) le modèle centralisé où les communications sont centralisées au

niveau d'une entité (p.ex., le modèle client/serveur [53]) ; et (ii) le modèle décentralisé où les communications sont possibles entre n'importe quelles entités (p.ex., le modèle *publish/subscribe* [35, 34], le modèle pair-à-pair [75] ou le modèle client/serveur étendu [118]). Ces modèles sont plus ou moins adaptés aux environnements mobiles. Le type de réseau mobile sans fil auquel le système distribué est destiné s'avère déterminant dans le choix du modèle de communication le plus adapté.

Pour les systèmes distribués qui intègrent un modèle centralisé, les réseaux mobiles qui supportent la communication directe (mode *ad hoc*) sont inadéquats. L'avantage d'établir des connexions directes entre entités mobiles devient un inconvénient dans le cas où l'entité qui centralise les communications n'est pas dans la portée de communication directe, ce qui rend tout échange impossible. En revanche, les réseaux basés sur une infrastructure s'accommodent mieux d'un modèle de communication centralisé. Toutefois, la connectivité intermittente dans les réseaux mobiles peut rendre les stations de base inaccessibles et par conséquent empêcher tout échange entre clients et serveur. De manière générale, les modèles de communication centralisés sont inadaptés aux environnements mobiles. En revanche, les modèles décentralisés sont plus adaptés car les communications sont possibles entre n'importe quelles entités mobiles. Toutefois, les réseaux à base d'infrastructure qui nécessitent que les messages soit relayés par des stations de bases réduisent considérablement les performances dans le cas où la communication s'effectue entre deux entités se trouvant dans la portée de communication directe l'une de l'autre. Dans ce cas de figure, les réseaux en mode *ad hoc* sont plus adaptés.

De manière générale, les entités mobiles évoluant dans un environnement mobile doivent pouvoir communiquer entre elles sans être contraintes par l'existence d'un serveur et/ou d'une infrastructure, c'est à dire, de manière décentralisée.

II.2.3 Accès aux données

La manipulation des données accessibles dans les environnements mobiles, qu'elles soit locales ou distantes, doit être adaptée à ce type d'environnement. Cette adaptation concerne en particulier : la réplication des données et le maintien de la cohérence des réplicas ainsi générés.

La connectivité dans les environnements mobiles est de nature intermittente, ce qui rend inaccessible, pour des périodes de temps indéterminées, des données né-

cessaires aux utilisateurs mobiles. De ce fait, la réplication des données au niveau des caches locaux des entités mobiles devient une nécessité incontournable. La réplication augmente la disponibilité des données mais doit prendre en compte, en plus des besoins des utilisateurs, la disponibilité des ressources au niveau des terminaux mobiles. En effet, une réplication massive ou systématique des données compromet les ressources de stockage souvent réduites au niveau des ordinateurs de poche. De même, les communications sans fil nécessaires pour ces réplications augmentent la consommation d'énergie au niveau des terminaux mobiles.

Le maintien de la cohérence des réplicas doit également prendre en compte la nature intermittente des connexions dans ce type d'environnement. Par conséquent, les protocoles de cohérence forte (protocoles conservatifs) sont en général à proscrire car ils supposent une connexion permanente entre les différents réplicas. Du fait de la connexion intermittente, les protocoles de cohérence forte génèrent des situations où aucun utilisateur n'a le droit de modifier son réplica. Les protocoles qui maintiennent une cohérence faible (protocoles optimistes) ont l'avantage d'autoriser des mises à jour concurrentes sur tous les réplicas déconnectés et procèdent à leur synchronisation au moment de la reconnexion. Ils doivent être accompagnés d'un mécanisme de détection de conflits lors des synchronisations et de procédures capables de résoudre les conflits *a posteriori*, tout en préservant le critère de cohérence à terme.

La prolifération des utilisateurs mobiles promeut des applications de nature collaborative, notamment entre les utilisateurs physiquement proches (p. ex. jeux multijoueurs, édition collaborative, CAO, *etc.*) [7]. Ces applications nécessitent d'avoir accès à la même version des données répliquées sur différents sites. Dans ce cas, la gestion de la cohérence faible n'est pas adaptée. Ainsi, il est judicieux d'offrir une gestion d'accès aux données à la fois adaptée à la connectivité du réseau sous-jacent et aux applications, et qui réduise la consommation des ressources. Notre contribution porte précisément sur cette problématique.

II.2.4 Gestion des ressources locales

Les ordinateurs mobiles dépendent pour leur fonctionnement, de leur batterie, souvent de faible autonomie et dont le rechargement n'est pas toujours possible dans les périodes de mobilité. De même, leur espace de stockage devient vite insuffisant du fait du nombre sans cesse croissant de données exploitées par les utilisateurs.

Les fonctions de communication et d'accès aux données sont celles qui contribuent majoritairement à l'épuisement de ces ressources critiques.

Les communications sans fil génèrent une plus grande consommation d'énergie comparée aux communications câblées. Or, les utilisateurs mobiles communiquent essentiellement *via* le réseau sans fil. De plus, la consommation d'énergie générée par les communications diffère selon le type du réseau sans fil. Dans les réseaux à base d'infrastructure, seuls les terminaux qui émettent ou reçoivent des messages sont concernés par les dépenses d'énergie. En revanche, dans les réseaux en mode *ad hoc*, tous les terminaux sont affectés, à des degrés divers, par les dépenses d'énergie. Ceci est dû au fait que dans le mode *ad hoc*, les interfaces réseau de tous les terminaux sont à l'écoute du support de transmission. Dès qu'un message est émis, tous les terminaux dans la portée de communication directe le reçoivent, mais seul le terminal destinataire du message le valide, tandis que les autres terminaux l'ignorent. Ainsi, même les terminaux non destinataires du message dépensent de l'énergie qui est toutefois relativement négligeable par rapport à l'énergie nécessaire pour émettre ou effectivement recevoir un message. Plusieurs tentatives existent au niveau des protocoles de routage pour diminuer la consommation d'énergie dans les réseaux mobiles en mode *ad hoc* [113, 74].

En ce qui concerne la gestion de l'accès aux données, les ressources affectées sont l'espace de stockage ainsi que l'énergie. Il est donc nécessaire d'adapter la gestion de l'accès aux données en fonction de la disponibilité de ces ressources, tout en réduisant leur consommation. A notre connaissance, aucune proposition n'a été faite au niveau *middleware* ou système, pour gérer l'accès aux données en fonction de la disponibilité des ressources tout en minimisant leur consommation.

II.2.5 Sécurité informatique

La sécurité dans un environnement mobile reste un véritable défi qui nécessite un certain nombre de compromis. En effet, pour assurer la sécurité d'un système distribué, il est nécessaire d'intervenir à plusieurs niveaux. Tout d'abord, il est indispensable de recourir à la cryptographie pour chiffrer et signer les communications et authentifier les utilisateurs. Il faut en outre s'assurer qu'il n'existe pas de failles dans la sécurité des composants informatiques du système qui permettraient, par exemple, de dérober les clés de chiffrement. Très schématiquement, ces vérifications

doivent intervenir à trois niveaux distincts :

- Au niveau de l'architecture générale du système pour vérifier que les divers protocoles de communication ne présentent pas de failles de sécurité.
- Au niveau des éléments logiciels du système pour vérifier l'absence de failles involontaires ou de portes dérobées (*back-doors*).
- Au niveau des éléments matériels du système pour s'assurer qu'aucune faiblesse ne permet l'entrée dans le système par un voie détournée (consommation électrique, fonctionnalités liées à la maintenance, *etc.*). Dans certains cas, il faut s'assurer qu'aucun tiers mal intentionné ne puisse modifier le système en vue de l'affaiblir.

Il est impossible d'assurer une sécurité absolue et sans faille. Par conséquent, les concepteurs des systèmes distribués en général et des systèmes mobiles en particulier doivent faire un certain nombre de compromis. Le fait est que le maintien de la sécurité se fait au détriment des performances et des ressources critiques. Il est alors important de déterminer le degré de sécurité qui convient selon l'utilisation du système et les caractéristiques des ordinateurs utilisés.

II.3 Discussion

Les systèmes distribués dédiés aux environnements mobiles sont confrontés aux éléments qui caractérisent les infrastructures de ces environnements, à savoir les terminaux mobiles et les réseaux mobiles sans fil. Chacun de ces éléments apporte des contraintes dont les systèmes distribués doivent tenir compte dans les fonctionnalités qu'ils offrent aux applications. Ces contraintes sont essentiellement liées à la faible autonomie des entités mobiles, à leur capacité de stockage réduite et à la connectivité intermittente dans les réseaux mobiles sans fil. Ainsi, la découverte de l'environnement mobile doit permettre d'identifier automatiquement les ressources accessibles à une entité mobile et qui correspondent à ses besoins, dans des environnements *a priori* inconnus, dynamiques et n'offrant pas nécessairement une infrastructure réseau. Les communications entre entités mobiles doivent en outre être basées sur des modèles décentralisées, pour ne pas être contraintes par l'accessibilité d'un serveur ou une station de base. Par ailleurs, au niveau réseau, les communications locales (dans la portée de communication) doivent être directes (mode *ad hoc*), alors que les communications distantes sont plus performantes si elles reposent sur des stations de base pour relayer les messages. En ce qui concerne la gestion de l'accès aux données,

qu'elles soient locales ou distantes, elle doit être adaptée à la connectivité du réseau sous-jacent et aux exigences des applications tout en tenant compte de la disponibilité des ressources critiques.

La prolifération des utilisateurs mobiles et l'utilisation des réseaux mobiles sans fil en mode *ad hoc* promeut notamment la formation de groupes collaboratifs [7]. Les formes de collaboration sont très variées mais elles impliquent toujours un groupe de personnes partageant un intérêt commun. Cet intérêt fédérateur peut être professionnel (p.ex. les membres d'un projet de recherche Européen) ou privé. Ces groupes de personnes forment des communautés qui partagent certaines ressources incluant les données. Les modes les plus rudimentaires de collaboration se limitent à des échanges de simples messages textuels (forums, listes de diffusion). Les applications nécessitant une collaboration ont des exigences spécifiques sur la gestion de l'accès aux données réalisés par les systèmes distribués mobiles. Selon le type d'applications, les contraintes sur les données partagées sont différentes. Si l'application nécessite de voir rapidement les mises à jour, on privilégiera la gestion de la cohérence forte des données partagées. Si, au contraire, la propagation des mises à jour n'a pas une forte incidence sur le bon fonctionnement de l'application, on choisira la gestion de la cohérence faible. De plus, le support de la collaboration dans des environnements mobiles nécessite la gestion de groupes collaboratifs formés dynamiquement, suivant la connectivité du réseau, par des entités mobiles appartenant aux même communautés.

Enfin, quelque soit le type d'application, toutes les fonctionnalités qu'un système distribué offre doivent tenir compte de la disponibilité des ressources critiques et oeuvrer pour réduire leur consommation.

III Systèmes distribués mobiles : état de l'art

Dans ce chapitre, nous présentons les solutions offertes par les différents systèmes distribués qui ont pour objectif de permettre l'accès aux données, dans un environnement mobile. Nous analysons par ailleurs ces solutions par rapport aux exigences décrites dans le chapitre II.

Dans la section III.1, nous présentons comment la découverte de l'environnement mobile a été abordée dans la littérature. Puis, dans la section III.2, nous considérons les modèles de communication adoptés par les systèmes distribués mobiles. La section III.3 aborde la gestion de l'accès aux données à travers les solutions proposées pour la gestion de la cohérence et celle de la réplication. Nous abordons également comment le support pour la collaboration a été traité par ces systèmes. Dans la section III.4, nous analysons les solutions apportées pour la gestion des ressources locales autres que les données. Enfin, dans la section III.5, nous abordons les défis ouverts posés par la gestion de la sécurité dans un environnement mobile et en particulier pour le support de la collaboration. Notre étude de l'état de l'art met en avant un certain nombre de lacunes dans les solutions existantes, précisées dans la section III.6. Ceci nous conduit à proposer une nouvelle solution à la gestion de l'accès aux données, introduite dans la section III.7 et décrite dans les chapitres suivants.

III.1 Découverte de l'environnement mobile

Dans la littérature, on retrouve un grand nombre de protocoles de découverte de services (ou ressources). Parmi les plus connus, citons : SLP (*Service Location Protocol*) [41, 42, 58, 130], JINI (*Java Intelligent Network Infrastructure*)[83, 100, 5, 91, 2], Salutation

[25], UPnP (*Universal Plug and Play*) [105, 28, 27], et SDP (*Bluetooth Service Discovery Protocol*) [40]. Ce grand nombre de protocoles de découverte de services est, entre autres, dû au nombre important de groupes de travail et de consortiums industriels qui contribuent à leur développement.

Le protocole SLP est conçu pour les réseaux TCP/IP et supporte des réseaux de grande taille. Il comprend trois composants logiciels :

- Le *User Agent* (UA) dont le rôle consiste à lancer les requêtes de découverte de service au niveau de l'utilisateur (client).
- Le *Service Agent* (SA) dont le rôle consiste à annoncer l'existence d'un service ainsi que ses caractéristiques.
- Le *Directory Agent* (DA) est un composant optionnel qui permet de centraliser les informations concernant les différents services publiés par les SAs, dans une base de données et qui répond aux requêtes des UAs en renvoyant l'URL du service concerné ainsi que ses caractéristiques.

Le protocole SLP supporte la découverte décentralisée de services (sans DA). Dans ce cas, les UAs envoient périodiquement leurs requêtes à l'adresse *multicast* de SLP. Tous les SAs sont à l'écoute à cette adresse. Si l'un d'eux offre le service recherché par l'UA, alors il envoie directement la réponse à l'UA concerné. De plus, chaque SA fait une annonce de ses services périodiquement. De cette façon, les UAs à l'écoute sont mis au courant des services accessibles.

Le protocole JINI comprend une architecture et un modèle de programmation basé sur le langage java. Il est nécessaire que les machines clientes (utilisateur) ou celles offrant un service, exécutent une machine virtuelle java. Le *Lookup Server* (similaire au DA dans SLP) gère la base de données des services disponibles sur le réseau de façon centralisée. Au lieu de répondre à une requête de découverte de services par une URL, comme dans le cas de SLP, le *Lookup Server* envoie le code qui permet d'accéder au service en question et qui sera exécuté directement sur la machine virtuelle de l'entité cliente. Si les clients disposent de peu de ressources de calcul ou de mémoire, les performances peuvent être très mauvaises.

Le protocole Salutation est articulé autour de *Salutation Managers* (SLM) qui joue le rôle de coordinateur dans le processus de découverte de services, dans le même esprit que le DA dans SLP. Salutation adopte une approche de configuration statique et par ce fait, il n'est pas adapté pour les réseaux dont la configuration est susceptible de changer fréquemment.

Le protocole UPnP représente une extension de la technologie *Plug and Play* aux cas où les nouveaux services sont accessibles via un réseau TCP/IP. Il a été conçu pour être utilisé sur des réseaux domestiques de petite taille, où il permet la détection et configuration automatique de nouveaux composants. UPnP est complètement décentralisé (pas d'équivalent de DA, de *Lookup Server* ou de SLM). Il utilise le *Simple Service Discovery Protocol* (SSDP) pour la découverte des nouveaux services.

Le protocole SDP permet la découverte de services fournis ou accessibles via Bluetooth. Il permet une recherche de services basée sur le type ou les attributs. SDP ne fournit pas de mécanismes d'accès aux services, mais permet de les obtenir par l'utilisation d'autres protocoles de découverte de services comme SLP et Salutation.

Ces différents protocoles de découverte de services comportent un certain nombre de distinctions. UPnP permet aux clients de faire une recherche de services basée sur leurs caractéristiques et ne supporte que les réseaux TCP/IP. Salutation est indépendant du protocole réseau utilisé, c.-à-d., il peut être exécuté sur de multiples infrastructures. Comme UPnP, Salutation est indépendant du langage de programmation. En revanche, JINI est indépendant de la plate-forme d'exécution ainsi que du système d'exploitation. Mais, il nécessite l'exécution d'une machine virtuelle java sur toutes les entités, ce qui présente une charge supplémentaire au niveau de la mémoire ainsi que du temps CPU. Enfin, JINI permet l'utilisation du *leasing* qui consiste à enregistrer un service pour une période de temps limitée. Ce procédé est utile pour des réseaux très dynamiques. SLP inclut également le *leasing*, et est indépendant du langage de programmation. Il supporte à l'exemple de UPnP un mode de fonctionnement décentralisé.

III.2 Modèle de communication

Dans le chapitre précédent, nous avons argumenté de l'adéquation des modèles de communication décentralisés pour les environnements mobiles sans fil. Toutefois, quelques systèmes distribués mobiles destinés à des réseaux mobiles cellulaires (en mode infrastructure) sont basés sur des modèles de communication centralisés. C'est notamment le cas du système Coda [119] qui est un système de fichiers, fondé sur le système AFS [84], adapté aux utilisateurs mobiles et qui est basé sur le modèle de communication client/serveur strict. Coda apparaît à l'utilisateur comme un système de fichiers Unix partagé traditionnel. Il fait une distinction entre les sites ser-

veurs, qui sont des machines physiquement sûres, fiables et gérés par une équipe technique qualifiée, et les sites clients qui sont hébergés sur des machines dispersées et qui peuvent être mobiles et déconnectées pour de longues périodes. Clients et serveurs communiquent à travers l'appel de procédure à distance (*RPC*). Odyssey [118, 89, 90], successeur de Coda, est également basé sur le modèle de communication client/serveur strict. La communication entre clients et serveurs se fait également *via* le RPC. La différence d'Odyssey par rapport à Coda réside dans le fait qu'il peut être utilisé avec différents serveurs de données tels que des serveurs de fichiers, des serveurs SQL ou des serveurs Web. Rover [55] est un autre système distribué dédié aux environnements mobiles. Il est également basé sur le modèle de communication client/serveur strict. Les clients sont en général hébergés par des ordinateurs mobiles, tandis que les serveurs sont impérativement hébergés sur des machines stationnaires fiables.

La grande majorité des systèmes distribués mobiles est basée sur un modèle de communication décentralisé. C'est notamment le cas de Ficus [43] qui comme Coda est un système de fichiers dédié aux environnements mobiles sans fil, mais qui est basé sur un modèle de communication pair-à-pair (*peer-to-peer*). De même, ses successeurs Rumor [111] et Roam [109, 107] sont basés sur le modèle de communication pair-à-pair. D'autres systèmes distribués adoptent également le modèle de communication pair-à-pair. C'est notamment le cas de Lime [103, 86], Bengal [33] et XMIDDLE [78]. Le système distribué Bayou [31] est pour sa part basé sur le modèle de communication client/serveur étendu. Un serveur est une machine qui détient une copie complète d'une ou de plusieurs bases de données. La figure III.1 illustre le modèle adopté par le système Bayou. Les clients peuvent accéder aux données stockées sur n'importe quel serveur à condition que la communication soit établie. Réciproquement, chaque machine détenant une copie de la base de données peut répondre aux requêtes de lecture et d'écriture d'autres entités avoisinantes.

III.3 Gestion de l'accès aux données

La gestion de l'accès aux données depuis des terminaux mobiles a fait l'objet d'un grand intérêt de la part des chercheurs dans le domaine des systèmes distribués. Plusieurs études ont ainsi été menées dès les années 90 comme suggéré dans la section précédente. Les premiers travaux se sont concentrés sur la gestion de l'accès aux

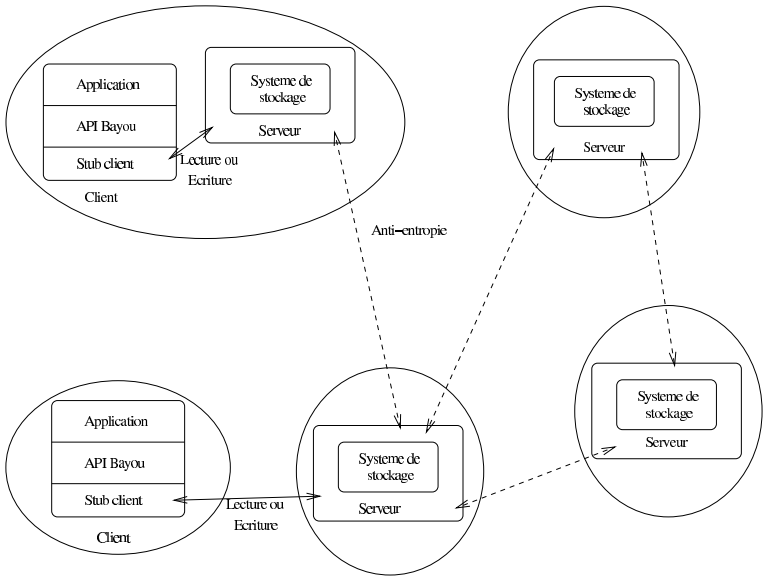

Figure III.1 – Architecture de Bayou

données privées à partir de sites mobiles. Ces travaux ont contribué à l'extension des systèmes clients/serveur traditionnels comprenant uniquement des entités stationnaires, pour inclure des entités mobiles ainsi qu'un support pour les réseaux sans fil à base d'infrastructure. La gestion de l'accès aux données dans les systèmes existants est, en général, abordée à travers la gestion de la cohérence et celle de la réplication.

III.3.1 Gestion de la cohérence

Les protocoles de gestion de la cohérence peuvent être classés en deux grandes catégories : ceux qui garantissent la cohérence des données répliquées à tout moment et sur n'importe lequel des réplicas (*cohérence forte*), et ceux qui font des compromis entre cette garantie et les contraintes de communication (*cohérence faible*). Les systèmes distribués dédiés aux environnements mobiles sans fil préconisent la gestion de la cohérence faible car elle est plus adaptée à la connectivité intermittente de ces environnements.

Dans cette section, nous présentons les solutions existantes pour la gestion de la cohérence forte ainsi que celles proposées dans les différents systèmes distribués mobiles pour la gestion de la cohérence faible.

III.3.1.1 Cohérence forte

La gestion de cohérence forte prévient l'occurrence des incohérences au niveau des données répliquées. Cette garantie est apportée par les protocoles de verrouillage [9], les protocoles dits de copie primaire [3] ou les protocoles à base de quorum [47].

Les protocoles de verrouillage garantissent qu'une donnée ne sera accédée en écriture que sur un seul de ses réplicas à la fois. Pour cela, ils intègrent un gestionnaire de verrous qui assure la pose et la libération des verrous sur les données et prévient l'apparition d'interbloquages. L'utilisation des verrous impose que les utilisateurs puissent identifier à l'avance l'ensemble des données sur lesquelles porteront leurs écritures.

Les protocoles de copie primaire [3] assurent la cohérence des données en centralisant toutes les écritures sur un réplica particulier. Celles-ci sont ensuite propagées sur les réplicas secondaires. Ces protocoles ont l'avantage d'être simple à implémenter et à administrer mais nécessitent une connexion à la copie primaire à chaque écriture. Cette condition peut rendre impossible la modification des réplicas locaux si la copie primaire est inaccessible. Ainsi, seul le détenteur de la copie primaire a le droit d'écriture quand il est déconnecté du réseau.

Les protocoles à base de quorum [47] relâchent le principe de la copie primaire en permettant les écritures et les lectures sur n'importe quel réplica à condition qu'un certains nombre d'autres réplicas (le quorum) y participent. Cette approche relâche la contrainte de la connexion à la copie primaire à chaque écriture, mais nécessite la connexion à un certain nombre d'autres réplicas à chaque écriture. Cette approche ne fait que reporter la contrainte imposée par les protocoles de copie primaire, car aucun utilisateur n'a le droit d'écriture sur son réplica local si un certain nombre (le quorum) de réplicas n'est pas connecté (accessible).

La gestion de cohérence forte introduit un nombre élevé d'échanges de messages pour maintenir tous les réplicas d'une donnée à la même valeur afin de prévenir les éventuelles incohérences. Les protocoles qui permettent une telle gestion de la cohérence ne sont pas toujours compatibles avec les environnements mobiles où les lectures et les écritures doivent avoir lieu, entre autre, sur des réplicas déconnectés.

III.3.1.2 Cohérence faible

Les protocoles de cohérence faible[1], dits également protocoles *optimistes*, ne cherchent pas à assurer la cohérence immédiate des réplicas. Les incohérences constatées disparaîtront progressivement avec l'intégration des nouvelles mises à jour dès que les communications seront rétablies. Dans cette approche, les conflits sont supposés être peu fréquents. Par conséquent, il est moins coûteux de les traiter lorsqu'ils surviennent plutôt que de les éviter. Les protocoles optimistes doivent donc être accompagnés d'un mécanisme de détection des conflits, lors des synchronisations, et de procédures capables de les corriger *a posteriori*, tout en préservant le critère de cohérence à terme[2].

Dans le système Coda, les clients déconnectés des serveurs manipulent librement leurs fichiers locaux. Les mises à jour sont journalisées et transmises dès que les connexions sont rétablies à un ensemble de serveurs répliqués appelés AVSG (*Available Volume Storage Group*). Dans Coda, la propagation des mises à jour entre réplicas stockés sur des sites clients est interdite. Coda propose des mécanismes de résolution de quelques conflits inhérents aux systèmes de fichiers, tels que les conflits de nommage, de suppression/modification et de déplacement de fichiers. Lorsqu'un autre type de conflit apparaît, sa résolution est entièrement laissée à la charge de l'utilisateur.

Dans le système Rover [55], chaque donnée possède également une copie principale stockée sur son serveur de référence. Ainsi les modifications au niveau des réplicas stockés sur les sites clients sont exportées vers les serveurs associés dès que les connexions sont rétablies.

Odyssey [72] propose une gestion de la cohérence faible guidée par l'application de sorte que l'application spécifie le seuil de fraîcheur à partir duquel la mise à jour de chaque réplica est nécessaire. Ainsi, dès que la connexion avec le serveur est rétablie, Odyssey détermine la fraîcheur des réplicas par rapport à ceux stockés sur le serveur et notifie l'application en conséquence. Suite à cela, les mises à jour nécessaires sont entreprises au niveau des réplicas locaux. Il revient également à l'application de spécifier les mécanismes de détection et de résolution des conflits qui souvent, font appel à l'utilisateur.

[1] Nous considérons la cohérence faible causale où les mises à jours sont reçues à terme dans le même ordre au niveau de tous les réplicas.

[2] *eventual consistency*

Ficus [46] base également son protocole de gestion de la cohérence des réplicas sur la rareté des mises à jour concurrentes. Chaque utilisateur, indépendamment de sa connectivité, a le droit de modifier immédiatement les données localement répliquées. Suite à cela, les autres réplicas seront directement notifiés à terme de cette mise à jour. Les réplicas inaccessibles (déconnectés) n'ont aucune garantie quant à la réception de la mise à jour en question, mais seront mis à jour à terme *via* des synchronisations ultérieures. Ficus ne journalise pas les mises à jour. Il utilise, comme son prédécesseur Locus [106, 125], un vecteur de version associé à chaque fichier de données. Périodiquement, le protocole de réconciliation compare les vecteurs de version associés à chaque fichier, pour déterminer si de nouvelles versions du fichier existent (éventuellement des versions divergentes). Comme dans tout système de fichiers Unix, les répertoires sont aussi des fichiers et possèdent, eux aussi, un vecteur de version. Le protocole de réconciliation génère un taux de communication très élevé. De plus, la comparaison périodique des vecteurs de versions génère des communications inutiles dans le cas où aucune modification de réplica n'a été enregistrée. Comme Coda, Ficus propose des mécanismes de résolution de quelques conflits inhérents aux systèmes de fichiers. Rumor [111] et Roam [109, 107] utilisent également les vecteurs de version dans leur gestion de la cohérence faible.

Dans le système distribué Bengal [33], les synchronisations entre différents réplicas sont effectuées lorsque les connexions entre les machines hôtes sont possibles. Un vecteur de version dynamique [108] est utilisé pour détecter les conflits entre les différents réplicas d'une donnée. Quand la comparaison de deux vecteurs de version indique la présence d'un conflit, Bengal fait appel à une série de programmes préconfigurés pour la résolution automatique des conflits. Ces programmes se basent sur des règles spécifiques à chaque application pour déterminer quelle version des réplicas en conflits garder, ou bien comment créer une nouvelle version pour fusionner les deux réplicas en conflit. Les utilisateurs de Bengal doivent spécifier la topologie de la réconciliation (topologie en anneau, en étoile, en arbre, *etc.*). Cette topologie ne contraint pas le choix des réplicas qui doivent se réconcilier entre eux, mais garantit que tous les réplicas recevront à terme toutes les mises à jour.

Le système distribué Bayou [102] utilise le protocole *read-any/write-any* dans son approche à la gestion de la cohérence de bases de données mobiles. Ce protocole a été initialement introduit dans Grapevine [12]. Il permet aux utilisateurs de lire et d'écrire sur n'importe quel réplica de la base de données. Dans Bayou, le client et le serveur peuvent être localisés sur le même hôte, ce qui est notamment le cas des

terminaux mobiles déconnectés. Bayou ne garantit pas le délai de propagation des mises à jour vers les autres réplicas étant donné la possible déconnexion de certains d'entre eux. Les serveurs propagent les mises à jours vers les différents réplicas de la base de données en utilisant le protocole d'anti-entropie [30]. Ce protocole assure que les réplicas convergeront à terme vers un état identique. Pour aboutir à ce résultat, les serveurs ordonnancent toutes les mises à jours reçues de manière cohérente. De plus, pour permettre à chaque couple de serveurs connectés de propager les mises à jour entre eux, le protocole d'anti-entropie a été adapté au modèle pair-à-pair (*peer-to-peer*). Bayou détecte les conflits en associant à chaque requête de mise à jour un ensemble de dépendances composé de requêtes de contrôle spécifiques à l'application, spécifiant des contraintes sur les réponses attendues. Un conflit est détecté au niveau d'un réplica si l'une des requêtes de l'ensemble de dépendances ne donne pas le résultat attendu. Dans ce cas, la résolution du conflit consiste à appeler la procédure de résolution de conflits associée à cet ensemble de dépendances et définie par l'application. OceanStore [66] applique la gestion de la cohérence telle qu'elle a été proposée dans Bayou à un système de fichiers étendu à des serveurs "non sûrs"(*untrusted*) qui traitent uniquement des données chiffrées. XMIDDLE [78] s'inspire également de la gestion de la cohérence proposée dans Bayou et nécessite l'intervention de l'utilisateur afin de spécifier les mécanismes de résolution des conflits pouvant survenir suite à des mises à jour concurrentes.

Dans la suite de ce document, on désignera par synchronisation de réplicas, la détection et la résolution des conflits dues aux mises à jour divergentes. La fréquence de synchronisation des réplicas est un facteur déterminant pour les performances du système. Elle doit résulter d'un compromis entre la fraîcheur des données désirée (la cohérence), la connectivité du réseau et les performances du système en termes de temps de réponse et de consommation des ressources. En effet, plus la synchronisation des réplicas est fréquente plus les réplicas sont à jour et plus les données perçues par les utilisateurs sont fraîches. Mais, ces synchronisations fréquentes mobilisent les ressources du système et par conséquent diminuent les performances de ce dernier. Inversement, plus les synchronisations sont espacées, plus le système est performant. Mais, la cohérence des données est de plus en plus faible. La fréquence des synchronisations peut être établie selon différents critères. Dans AFS [57], ou les systèmes s'y référant, la synchronisation avec le réplica stocké sur le serveur, s'effectue dès la fermeture d'un fichier préalablement ouvert en écriture. Une fois le fichier fermé, le client envoie les mises à jour au serveur qui se charge d'invalider les autres réplicas

du fichier. Cependant, AFS présente deux cas particuliers. Si le fichier a été ouvert en écriture pendant plus de 30 secondes, alors les mises à jour sont transmises au serveur toutes les 30 secondes. L'autre cas particulier survient quand le cache du client est plein. Ce dernier peut alors initier la transmission des mises à jour pour libérer de l'espace avant même que le fichier ouvert en écriture soit fermé. Ces stratégies ne peuvent pas être appliquée dans des systèmes distribués dédiés aux environnements mobiles du fait de la connectivité intermittente. Dans ces systèmes, la fréquence de la synchronisation des réplicas est, en grande partie, déterminée par la connectivité du réseau sous-jacent.

III.3.2 Gestion de la réplication

Les premiers systèmes distribués mobiles n'étaient pas confrontés à des ordinateurs de poche de faible autonomie ou de capacité de stockage réduite tels que les PDAs et les téléphones mobiles intelligents. Par conséquent, ils n'étaient pas concernés par toutes les contraintes rencontrées actuellement dans les environnements mobiles. Leur préoccupation majeure était la connexion intermittente dans l'environnement mobile. Ainsi, la réplication des données devait permettre de rendre les déconnexions transparentes à l'utilisateur en lui assurant la disponibilité de ses données hors connexion. Dans cette section, nous nous intéressons à la gestion de la disponibilité des données dans les environnements mobiles telle qu'elle a été abordée par quelques systèmes distribués existants dédiés aux environnements mobiles.

Dans Coda [61, 62], la disponibilité des données est assurée à deux niveaux : au niveau des serveurs, en répliquant le système de fichiers sur plusieurs serveurs et au niveau des clients où les périodes de déconnexion sont rendues transparentes par le préchargement (*hoarding*) des données nécessaires pour le travail hors connexion. Ainsi, le cache local d'un client joue le rôle d'un serveur pendant la durée de la déconnexion. Le modèle de réplication de Coda distingue les réplicas du serveur, qui sont considérés comme copies primaires, des réplicas sur les sites clients, considérés comme copies secondaires. L'unité de réplication dans Coda est le volume, qui est un ensemble de fichiers et de répertoires formant un sous arbre du système de fichiers. L'ensemble de serveurs qui répliquent un volume est appelé AVSG (*Available Volume Storage Group*). La technique de préchargement se base essentiellement sur la prévision des données allant être accédées lors des périodes de déconnexion. Ainsi, pour éviter les défauts de cache (*cache misses*), les données fréquemment ac-

cédées sont répliquées dans leur intégralité. Ceci présente un inconvénient quand les utilisateurs sont équipés d'ordinateurs de poche de faible capacité de stockage. Ficus [43, 92], implémente également une technique de préchargement pour assurer la disponibilité des données au niveau du cache des clients pendant les périodes de déconnexion. Mais, contrairement à Coda, il ne fait pas de distinction entre réplicas. Odyssey distingue entre les réplicas stockés sur les serveurs et ceux stockés sur les clients mobiles. Mais, les données sont répliquées à partir des serveurs selon les accès effectifs de l'application et non pas suivant des techniques de préchargement. De même, la granularité de la réplication est définie par l'application pour chaque type de données. De manière complémentaire à la gestion de la réplication des données, des travaux sur le préchargement et l'anticipation (prévision) des accès basés sur les relations sémantiques entre réplicas ont été réalisés dans le cadre plus général des systèmes distribués stationnaires [19], et adaptés dans le cadre des environnements mobiles [68, 69, 67].

Bayou [31, 32] aborde la disponibilité des données à travers son modèle de réplication, largement inspiré de celui de Grapvine [12]. La disponibilité des données n'est garantie que par la multiplication des réplicas qui se font à la demande des utilisateurs sans aucune notion d'anticipation ou d'adaptation au contexte d'exécution. De nombreux systèmes se sont inspirés du modèle de réplication tel qu'il est définit dans Bayou [66, 129]. Dans Roam, les réplicas sont créés lors des accès effectifs. Puis, Roam rassemble les réplicas d'une donnée stockés sur des sites localisés dans une zone géographique limitée dans un WARD (*Wide Area Replication Domains*). La granularité de la réplication dans Roam est le volume de données, qui peut être partitionné pour une granularité plus fine dans le modèle de *WARD* étendu. Les membres d'un *WARD* sont sélectionnés selon des critères très restrictifs. En plus de la proximité géographique, les machines doivent être de préférence de puissance équivalente. Chaque *WARD* dispose d'un gestionnaire chargé de la propagation des mises à jour et de la synchronisation avec d'autres *WARDs*. La synchronisation au sein d'un *WARD* se fait selon une topologie en anneau. L'insertion d'un nouveau réplica (stocké sur une nouvelle machine) dans un *WARD* existant est à la charge de l'utilisateur. Dans le cas de la mobilité d'une des machines appartenant au *WARD*, Roam ne prévoit pas de prise en compte automatique. La machine continue de se synchroniser avec le gestionnaire de son *WARD* initial, même à travers une liaison coûteuse, à moins que l'utilisateur ne fasse une déconnexion explicite de son *WARD* initial et rattache sa machine à un *WARD* plus proche géographiquement. Dans XMiddle [78], les réplicas sont créés lors

des accès effectifs. Il gère, par ailleurs, le partage d'arborescences de données structurées de type XML. Chaque utilisateur définit un ensemble de points d'accès vers ses données locales. Ces points d'accès concernent en particulier des branches de l'arborescence de données qui peuvent être accédées et modifiées par les autres utilisateurs. La taille de ces branches varie d'un simple noeud à l'arbre en entier. Ainsi, dans XMIDDLE, la granularité de la réplication peut être adaptée aux besoins des utilisateurs et aux capacités des ordinateurs utilisés. Dans Bengal [33] et 7DS [93, 94] la disponibilité des données est également assurée par la multiplication des réplicas qui sont créés lors des accès effectifs, tout en exploitant les possibilités offertes par le modèle de communication pair-à-pair, dont la proximité d'entités mobiles pairs stockant des données allant être accédées dans la portée de la communication du réseau sous-jacent.

D'autres systèmes proposent d'adapter la gestion de la disponibilité des données à la nature des accès effectués sur ces dernières, à travers la réplication dynamique [126]. Le taux de réplication de chaque donnée est adapté en fonction des accès en lecture/écriture. L'accès à une donnée locale en lecture étant plus rapide qu'un accès distant et ne nécessitant pas de communications externes, le taux de réplication des données accédées seulement en lecture est augmenté. Ceci permet d'accroître le nombre d'accès en local et de diminuer les communications externes. En revanche, la mise à jour d'une donnée répliquée doit être propagée à terme à tous ses réplicas. Ainsi, plus le taux de réplication est élevé, plus la propagation des mises à jour est coûteuse en termes de communications et de résolution de conflits. Dans ce cas, le taux de réplication des données essentiellement accédées en écriture est réduit. Cette solution est applicable dans les cas où l'on connaît d'avance la nature des accès allant être effectués pour chaque donnée répliquée. De plus, cette solution implique un temps de latence (un certain nombre d'accès) avant de pouvoir adapter le taux de réplication à la nature des accès, car la réplication doit se faire sur les sites ayant effectués le plus grand nombre d'accès.

La solution proposée dans [45] propose d'augmenter la disponibilité des données dans le cas du partitionnement des réseaux sans fil provoqué par les connexions intermittentes. La proposition consiste en une répartition des réplicas sur les différentes entités mobiles présentes dans la portée de la communication les unes des autres. Cette solution se base sur la fréquence d'accès à un réplica, ainsi que les fréquences d'accès aux réplicas présents dans le voisinage de chaque entité mobile. Il est clair que cette solution nécessite une connaissance antérieure du schéma d'accès aux données

afin d'établir les fréquences d'accès respectives. De même, cette approche part du principe que les déconnexions sont inévitables. Elle génère, par conséquent, une réplication systématique sans tenir compte de l'occurrence effective des déconnexions.

Dans le cadre des environnements mobiles, le contexte d'exécution subit des changements très fréquents. La disponibilité des données est fortement affectée par ces changements. Par exemple, la baisse d'énergie provoque la déconnexion de l'entité mobile et rend ainsi ses données locales inaccessibles. L'utilisation des techniques de préchargement est intéressante dans ce sens, mais leur déclenchement se fait en fonction des accès et non en fonction de la disponibilité des ressources. De plus, aucun modèle de réplication ne prend en compte la limite d'énergie, qui est un facteur important dans les déconnexions involontaires. La gestion de la disponibilité des données doit anticiper les changements du contexte d'exécution, et s'adapter en répliquant les données susceptibles d'être affectées par ces changements sur des entités mobiles appropriées. On retrouve dans la littérature des tentatives de modèle de réplication adapté au contexte d'exécution, mais aucun de ces modèles ne prend en compte tous les composants du contexte d'exécution. De plus, certains d'entre eux nécessitent l'intervention de l'utilisateur et considèrent qu'il est impossible de concilier transparence vis à vis de l'utilisateur, et adaptation au contexte d'exécution pour l'augmentation de la disponibilité des données.

Afin de répondre aux changements du contexte d'exécution, les systèmes reconfigurables ou réflexifs, qui sont des systèmes capables de s'auto-reconfigurer au moyen d'inspection et/ou adaptation, ont été proposés dans le cadre des environnements mobiles. Ces systèmes bannissent la transparence vis-à-vis de l'utilisateur et/ou de l'application, et considèrent que ses indications sont indispensables afin de répliquer la bonne donnée au bon moment. Le principe de réflexion a été utilisé dans des systèmes dédiés aux environnements mobiles tels que OpenORB[3], OpenCorba [73], dynamic-TAO [64], les travaux de Blair et *al.* [13], MULTE-ORB [104], Flexinet [44], et CARISMA [21]. Par exemple, dans CARISMA, l'adaptation de la gestion de la disponibilité des données se base sur les indications de l'utilisateur ou de l'application. Ainsi, il est nécessaire de spécifier explicitement : (i) *quand*, c.-à-d., les conditions qui doivent être réunis afin de procéder à la réplication des données d'intérêt, comme la baisse de l'énergie ou la baisse de la connectivité ; et (ii) *comment*, c.-à-d., le protocole qui doit être utilisé pour la réplication des données. Ceci permet l'adaptation de la gestion de la réplication au contexte d'exécution, mais nécessite l'intervention de

[3] `http://openorb.exolab.org/openorb.html`

l'utilisateur qui doit être capable de donner une spécification exacte de ses besoins par rapport aux différents contextes d'exécution.

III.3.3 Gestion de la collaboration

Les formes de collaboration sont très variées mais elles impliquent toujours un groupe de personnes qui partagent un intérêt commun et travaillent ensemble. Pour le système collaborer signifie partager des ressources et notamment des données, être au courant des participants qui collaborent et de l'évolution du travail de collaboration [76, 117]. Selon le type d'application, la collaboration peut être synchrone ou asynchrone. Suivant la collaboration synchrone, les utilisateurs participent au travail de collaboration et partagent leurs ressources en même temps. Ainsi, les résultats de chaque utilisateur sont propagés vers les autres instantanément. Ce mode de collaboration est dédié aux applications qui nécessitent de voir rapidement les mises à jour des ressources partagées (p.ex., jeux multi-joueurs, édition collaborative, CAO, *etc.*). Suivant la collaboration asynchrone, les utilisateurs participent à un travail en commun avec d'autres utilisateurs de manière indépendante et à des moments différents. Il n'y a pas de notion de partage de ressources au même moment. Par conséquent, dans ce type de collaboration la propagation rapide des résultats n'est pas nécessaire. Ce mode de collaboration est dédié aux applications où la propagation des mises à jour n'a pas une forte incidence sur leur bon fonctionnement (p.ex., forum, listes de discussions, *etc.*).

La combinaison des modèles de communication décentralisés et des réseaux mobiles sans fil offre un potentiel important pour la collaboration entre utilisateurs mobiles en tout lieu et à tout instant. En effet, de plus en plus d'utilisateurs sont demandeurs de solutions qui leur permettent de collaborer dans des situations de mobilité avec des utilisateurs dans la portée de communication du réseau local sans fil [8, 101, 26]. Certaines propositions considèrent que la clé pour la collaboration dans un environnement mobile sans fil est la possibilité de travailler hors connexion [32, 116]. Or, ces propositions offrent uniquement un support à la collaboration asynchrone. D'autres propositions considèrent que la clé pour la collaboration dans ces environnements consiste dans la possibilité de partager des ressources avec des entités qui se trouvent dans la portée de communication et d'exploiter toutes les connectivités offertes par les réseaux mobiles sans fil et notamment le mode *ad hoc* [7]. Ces propositions permettent la collaboration synchrone. Nous considérons que les deux

propositions sont complémentaires et plus spécifiquement que le support de la collaboration doit être adapté à la connectivité du réseau sous-jacent [18].

Dans la collaboration asynchrone la propagation immédiate des résultats n'est pas nécessaire, ce qui rend la gestion de la cohérence faible des données répliquées adaptée à cette forme de collaboration. Par conséquent, tous les systèmes distribués qui offrent une gestion faible de la cohérence des données répliquées supportent la collaboration asynchrone. En revanche, le support de la collaboration synchrone dans des environnements mobiles sans fil présente un réel défi. Les applications où la collaboration doit être synchrone nécessitent une propagation rapide des évolutions des ressources partagées. Or, la connectivité intermittente constitue un obstacle à de telles propagations. Une solution à ce problème réside dans l'exploitation du mode *ad hoc* des réseaux sans fil qui permet de constituer des groupes d'entités mobiles qui souhaitent collaborer et qui se trouvent dans la portée de communication les unes des autres. Ce choix est de plus conforté par le fait que les utilisateurs qui souhaitent bénéficier d'une collaboration synchrone se trouvent généralement à proximité les uns des autres. Par ailleurs, la collaboration synchrone requiert une gestion de la cohérence forte qui ne contraint pas la mobilité des utilisateurs afin de permettre aux applications d'avoir une vision exacte des ressources partagées, et en particulier les données.

III.4 Gestion des ressources locales

Les systèmes distribués mobiles doivent tenir compte de la faible capacité de stockage des ordinateurs de poches et adapter leurs fonctionnalités en conséquence. Certains systèmes ne prévoient aucune adaptation de leurs fonctionnalités à la disponibilité des ressources. C'est notamment le cas de Coda et Ficus, où la gestion de la réplication n'adapte pas la taille des volumes à répliquer aux capacités de stockage au niveau des sites clients. D'autres systèmes permettent d'adapter la réplication des données aux capacités de stockage des entités mobiles en offrant des granularités variables de réplication. C'est notamment le cas de Roam, où la granularité de réplication qui est par défaut le volume peut être partitionnée pour une granularité plus fine, dans le modèle de *WARD* étendu. De même, le système XMIDDLE permet une granularité de réplication variable et adaptée aux différentes capacités de stockage *via* la manipulation de données structurées de type XML. D'autres systèmes per-

mettent aux applications d'intervenir au niveau de la gestion des ressources. Dans Odyssey, les applications adaptent leur fonctionnement aux ressources disponibles au niveau des sites clients (p.ex., bande passante, espace de stockage, CPU, *etc.*). Pour

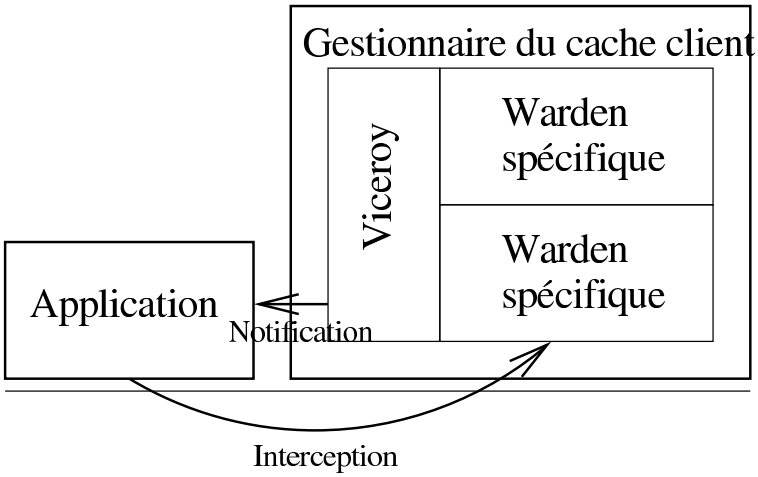

Figure III.2 – Architecture d'un client Odyssey

pouvoir adapter son fonctionnement, l'application doit en premier spécifier les ressources dont elle a besoin. Pour chaque type de ressource, elle doit également définir les seuils supérieurs et inférieurs de disponibilité acceptable, ainsi qu'une procédure de notification qui doit être appelée dès que la ressource en question dépasse l'un des seuils définis. Odyssey définit un composant nommé *viceroy* pour la gestion des ressources sur les sites clients et de la notification des applications lors de changements significatifs, comme illustré dans la figure III.2. Quand l'application est notifiée d'un changement de disponibilité d'une ressource, elle doit adapter son accès en conséquence. Pour chaque type de ressource, un *Warden* est défini ; son rôle consiste à implémenter différentes méthodes d'accès à la ressource et à appliquer la méthode appropriée en fonction des changements constatés. Pour sa part, le système Rover offre aux applications le choix de l'emplacement où les calculs seront effectués (sur le client ou le serveur) en déplaçant les RDOs correspondants (*Relocatable Dynamic Objects*) du client vers le serveur et *vice versa*. Ceci permet d'effectuer des calculs importants sur des machines plus puissantes que les ordinateurs mobiles qui hébergent les clients et de réduire les communications entre les clients et les serveurs. Un RDO est un objet composé de code et de données, avec une interface bien définie. De même, LIME (*Linda In Mobile Environment*) est basé sur la notion d'agent mobile qui permet le choix de l'emplacement de l'exécution des agents selon la connectivité du réseau

et la disponibilité des données qui sont organisées en un espace virtuel de stockage (*Global Virtual Data Structure*) [103, 86].

La gestion des ressources locales associée à celle de l'accès aux données dans des environnements mobiles sans fil se focalise essentiellement sur la gestion de l'espace de stockage qui est directement lié aux données. Rares sont les systèmes qui prennent en compte la réduction de la consommation d'énergie au niveau des sites mobiles, qui se traduit principalement par la nécessité de réduire les communications sans fil. A ce niveau, seuls les protocoles de transmission sans fil intègrent l'économie d'énergie par des processus de mise en veille pour les entités inactives.

III.5 Gestion de la sécurité

La sécurité relève d'un domaine de compétence souvent distinct de celui de la gestion de l'accès aux données. Ainsi, les systèmes distribués mobiles n'intègrent pas de mécanismes spécifiques pour la gestion de la sécurité, mais permettent de faire appel à de tels mécanismes. Cependant, du fait de l'impact négatif de la sécurité sur les performances des systèmes distribués et les ressources des terminaux mobiles, il est important de déterminer le degré de sécurité qui convient selon l'utilisation du système et les caractéristiques des terminaux.

Par ailleurs, la notion de groupes collaboratifs introduit un défi supplémentaire pour la gestion de la sécurité, qui est le calcul de la clé de groupe et le contrôle des admissions. Plusieurs solutions centralisées ont été proposées où un tiers de confiance interne ou externe au groupe gère les clés de chiffrement, les contrôles des admissions et l'authentification des membres du groupe [49, 50, 79]. Cependant, dans les groupes collaboratifs, où le réseau sous-jacent est en mode *ad hoc* et où le modèle de communication est décentralisé, la notion de tiers de confiance interne au groupe n'existe pas car toutes les entités mobiles sont au même niveau (entités paires). De même, les solutions basées sur un tiers de confiance externe au groupe ne sont pas applicables car elles supposent une connexion permanente entre ce dernier et le groupe, ce qui ne peut être vérifié dans un réseau mobile sans fil en mode *ad hoc*. Ainsi, traiter l'absence d'une autorité centralisée au sein des groupes a donné naissance à des solutions qui impliquent tous les membres du groupe pour la mise en accord d'une clé de groupe qui sert pour le chiffrement des communications au sein de ce dernier [60, 4, 128, 20]. Par ailleurs, le contrôle des admissions dans le groupe ou l'authentification des enti-

tés mobiles pouvant faire partie du groupe exige la présence d'un tiers de confiance qui contrôle les identités de ces entités puis leur attribut des certificats numériques.

III.6 Discussion

Les différents systèmes pour la gestion de l'accès aux données dans un environnement où les connexions sont intermittentes, et par conséquent où le partitionnement du réseau est très fréquent et de durée très variable, diffèrent dans leur choix du modèle de communication qui varie du client/serveur strict (Coda, Odyssey, Rover) au client/serveur étendu (Bayou) en passant par le modèle pair-à-pair (Ficus, Rumor, Roam, Bengal, XMIDDLE). Le modèle de communication pair-à-pair est de loin le plus adapté à des situations où la mobilité rend le serveur inaccessible (voir section III.2). De plus, il est particulièrement adapté pour les réseaux mobiles sans fil en mode *ad hoc*, où la communication est directe entre les différentes entités mobiles (sans avoir recours à une infrastructure de stations de base). Le mode *ad hoc* offre une flexibilité aux utilisateurs mobiles en leur permettant d'établir un réseau local n'importe où, n'importe quand à un coût réduit et sans qu'aucune administration du réseau ne soit nécessaire. En outre, les systèmes de gestion de l'accès aux données n'intègrent aucun mécanisme spécifique de découverte de services qui est traité à part dans la littérature (voir section III.1). Ils supposent résolue voir laissent à la charge de l'utilisateur l'identification du contexte d'exécution, incluant la localisation des serveurs pour les modèles client/serveur, des entités paires pour les modèles pair-à-pair. Or, cette connaissance est importante pour gérer au mieux le partage des ressources et en particulier des données. Par exemple, dans le système Roam, la gestion des *WARDs* est à la charge des utilisateurs, et notamment lorsqu'un utilisateur quitte un *WARD* il lui revient de se reconnecter à un autre *WARD* plus approprié. Ceci pose un problème lorsque les utilisateurs se retrouvent dans des environnements nouveaux.

La gestion des données est abordée à travers la gestion de leur cohérence (voir section III.3.1) et celle de leur réplication (voir section III.3.2). Tous les systèmes de gestion d'accès aux données en environnement mobile adoptent un modèle de réplication basé sur une gestion de la cohérence faible où les mises à jour sont effectuées de manière concurrente et parallèle sur les différents réplicas de la même donnée. Ce choix nécessite l'intégration de mécanisme de détection et de résolution des conflits, qui sont soit systématiquement laissés à la charge des développeurs d'applications

(Bayou, XMIDDLE, *etc.*), soit limités à certain types de conflits (Coda, Odyssey, Ficus, *etc.*). De manière générale, tous nécessitent l'intervention de l'utilisateur à un moment ou un autre du processus de réconciliation. La notion de collaboration entre différentes entités mobiles, pour le partage des ressources incluant les données, n'est ainsi abordée que du point de vue de la collaboration asynchrone qui s'accommode d'une gestion optimiste de la cohérence (voir section III.3.3).

Le type des données gérées varie des systèmes de fichiers (Coda, Ficus, Rumor, Roam) aux données structurées comme les bases de données ou les documents XML. Les données structurées ont l'avantage d'apporter une composante sémantique dans les mécanismes de détection et de résolution des conflits, qui permet de retarder l'intervention de l'utilisateur et de résoudre automatiquement un plus grand nombre de conflits. Elles ont également l'avantage de permettre l'utilisation d'une granularité de réplication très flexible qui peut être adaptée aux besoins des utilisateurs et aux capacités de leurs ordinateurs mobiles. En revanche, les systèmes de fichiers ont l'avantage d'englober un large éventail de données et permettent ainsi d'offrir un support à la mobilité pour des applications existantes. Toutefois, des mécanismes de détection et de résolution des conflits plus élaborés sont nécessaires si l'on veut limiter l'intervention des utilisateurs [59].

Aucun des systèmes existants ne propose un modèle de réplication qui permet d'anticiper les changements du contexte d'exécution de manière transparente à l'utilisateur afin de garantir la disponibilité des données. De plus, aucun de ces systèmes n'a abordé la gestion de l'accès aux données en se souciant de la disponibilité de l'énergie qui est une ressource critique quant il s'agit d'ordinateurs de poche de faible autonomie.

De nos jours, les environnements mobiles sans fil offrent des contextes d'exécution très variables : présence/absence d'une infrastructure, ordinateurs mobiles de caractéristiques hétérogènes et connectivité intermittente. Il est donc, important d'offrir aux utilisateurs mobiles des solutions adaptées à chaque contexte d'exécution. Ceci est possible grâce à une approche adaptative par rapport au modèle de communication, à la gestion de l'accès aux données et à la gestion des ressources. Effectivement, un système distribué de gestion d'accès aux données dans les environnements mobiles doit offrir :

- La découverte automatique des ressources logicielles et/ou matérielles accessibles dans l'environnement mobile.
- Une sélection automatique du mode de communication du réseau sous-jacent

(*ad hoc* ou infrastructure) le plus adapté en termes de performances et de coût.
- Une gestion des données qui augmente leur disponibilité tout en tenant compte des ressources disponibles localement et dans le contexte d'exécution.
- L'exploitation de l'ensemble des ressources accessibles dans le contexte d'exécution.
- La possibilité d'établir des communications sécurisées.

III.7 Contributions

Au regard des limitations des solutions existantes, nous avons conçu un système distribué pour la gestion de l'accès aux données adapté aux environnements mobile. Notre système prend en compte l'ensemble des caractéristiques des entités mobiles tout en offrant un support pour les applications qui nécessitent une collaboration synchrone ou asynchrone. Nous avons en outre implémenté un prototype de notre système pour fournir une évaluation de ses propriétés non fonctionnelles, tout aussi cruciales que ses propriétés fonctionnelles pour une utilisation effective.

Dans notre contribution, nous exploitons toutes les connectivités offertes de nos jours par les réseaux mobiles et notamment le mode *ad hoc* ainsi que le modèle de communication pair-à-pair. Le mode *ad hoc* nous permet d'exploiter la proximité des entités mobiles qui partagent un intérêt commun et de définir sur cette base la notion de groupes mobiles. Le but à travers la gestion de tels groupes est de permettre le partage des ressources entre un ensemble d'entités mobiles et offrir ainsi un support aux applications qui nécessitent une collaboration synchrone. Bien que nous ne traitons pas les aspects de la sécurité, notre gestion de groupe présente une base pour l'implémentation de mécanismes de contrôle des admissions dans le groupe et de calcul de la clé de groupe pour le chiffrement des communications. Nous offrons également la possibilité d'avoir recours à un tiers de confiance à travers la notion de domaine de sécurité défini par rapport à une machine de référence. Dans ce cadre, une collaboration a été entreprise avec des experts du domaine de la cryptographie pour fournir une solution efficace en terme de consommation de ressources induite [11].

La gestion de l'accès aux données proprement dit est réalisée à travers la gestion de leur cohérence et la gestion de leur réplication. Nous proposons une gestion hybride de la cohérence en fonction de la connectivité du réseau. Au sein d'un groupe, nous adoptons la gestion de la cohérence forte. En effet, la connectivité du réseau lo-

cal sous-jacent permet d'établir les communications sans fil nécessaires pour le maintien d'une cohérence forte. De plus, cette gestion de la cohérence assure aux applications qui nécessitent une collaboration synchrone, une propagation rapide (suivant les accès effectifs aux données) des mises à jour. En dehors des groupes ou dans des groupes distincts, les entités mobiles bénéficient d'une gestion faible de la cohérence qui leur permet de manipuler les données de manière indépendante et concurrente, d'où la nécessité de mécanismes de détection et de résolution des conflits lors de la synchronisation des réplicas. La gestion de la cohérence faible, couplée au mécanisme de détection et de résolution des conflits, offre un support pour les applications nécessitant une collaboration asynchrone à l'instar des solutions proposées dans la littérature.

Enfin, nous proposons une gestion de la disponibilité des données à travers une gestion adaptative de la réplication. Outre la réplication des données générées par les accès, nous augmentons la disponibilité des données à travers une réplication préventive qui permet d'anticiper les déconnexions des entités qui appartiennent à un groupe et donc qui participent à un travail de collaboration. En effet, les données partagées dans un groupe peuvent devenir inaccessibles du fait de la déconnexion des entités mobiles qui les stockent. Nous proposons ainsi un modèle de réplication adaptative des données qui prend en compte la disponibilité des ressources de chaque entité mobile. Notre modèle de réplication permet de minimiser la consommation des ressources critiques (énergie, espace de stockage) à travers un mécanisme d'anticipation de l'indisponibilité des données et un protocole de choix des entités mobiles allant stocker les nouveaux réplicas qui favorise la préservation de leurs ressources.

Toutes les fonctionnalités offertes par notre système distribué pour la gestion de l'accès aux données en environnement mobile sans fil tiennent compte de la disponibilité des ressources au niveau des entités mobiles et tentent de réduire la consommation d'énergie en réduisant les échanges de messages *via* les communications sans fil. Nous considérons également que l'intervention de l'utilisateur doit être minimisée et notamment lorsqu'il est possible de prendre en compte les changements du contexte d'exécution de manière transparente.

Dans la suite de ce document, nous présentons les trois fonctions principales de notre système qui sont :

– la gestion des groupes mobiles, basée sur un modèle de communication pair-à-pair, intègre la découverte du contexte d'exécution et permet la gestion du

partage des ressources (chapitre IV) ;
- la gestion de la cohérence des données qui réalise une gestion hybride de la cohérence suivant la connectivité du réseau sans fil sous-jacent et offre un support pour les applications qui nécessitent une collaboration asynchrone ou synchrone (chapitre V) ; et
- la gestion de la disponibilité des données qui permet d'anticiper les changements dans la disponibilité des ressources (chapitre VI).

Nous introduisons enfin un prototype de notre système *via* son intégration dans un système de fichiers distribué mobile (chapitre VII). Ceci nous permet de valider notre proposition du point de vue des propriétés non fonctionnelles offertes. Nous montrons notamment que la gestion de groupe offre des temps de réponse proportionnels à la taille du groupe, tandis que les temps de réponse de la gestion d'accès aux données sont proportionnels à la taille des mises à jour nécessaires. Nous avons également évalué le surcoût mémoire généré par notre système qui est négligeable comparer à la taille moyenne des données. Enfin, l'évaluation de l'énergie consommée a montré l'avantage de notre gestion de groupe et d'accès aux donnée par rapport par rapport notamment à la gestion d'accès aux données implémentée dans les systèmes distribués existant.

IV Gestion des groupes mobiles

Notre but est de permettre à des entités mobiles partageant un intérêt commun de former des réseaux locaux sans fil de façon spontanée dès que la connectivité directe le permet, c.-à-d., sans recours aux infrastructures réseaux existantes, afin de partager leurs ressources locales. Nous introduisons ainsi la notion de groupe mobile définie par rapport à un centre d'intérêt [81, 82]. Un groupe mobile possède les propriétés suivantes :

- La création du groupe est spontanée et transparente aux utilisateurs.
- Les entités appartenant à un groupe peuvent se faire confiance et partager leurs ressources.
- Les entités appartenant au groupe ont une vue globale des ressources disponibles au sein de ce dernier et en particulier, les données accessibles, l'espace de stockage que chaque entité met à disposition ainsi que l'énergie dont elle dispose.
- Les entités appartenant au groupe bénéficient d'une gestion des données partagées adaptée aux ressources disponibles dans le groupe ; ceci permet aux entités ayant de faibles ressources de bénéficier des ressources mises à disposition par d'autre entités du même groupe.
- La gestion du groupe n'impose aucune restriction quant à la mobilité des entités ; chaque entité est libre de quitter un groupe, ce qui lui permet, entre autre, d'épargner les ressources locales qui sont affectées par leur partage.
- La gestion du groupe est réalisée de sorte à minimiser la consommation des ressources et en particulier la consommation de l'énergie au niveau des entités mobiles.

Dans la section IV.1, nous introduisons la définition de notre notion de groupe mobile basée sur la notion de domaine de sécurité. A partir de la section IV.2, nous précisons les fonctionnalités qui permettent d'assurer les propriétés susmentionnées

ainsi que les protocoles nécessaires pour leur réalisation. En particulier, nous abordons la découverte décentralisée des entités mobiles, où chaque entité détecte toutes les entités dans sa portée de communication et appartenant à son domaine de sécurité. A l'issue de l'étape de découverte, l'initialisation de groupe, abordée dans la section IV.3, permet la création effective du groupe supportant le partage des ressources. Dans la section IV.4, nous traitons de l'évolution du groupe, liée à la mobilité des entités ainsi qu'à la disponibilité des ressources. Ce chapitre se conclut par une évaluation de la gestion de groupe en terme de complexité théorique, ainsi qu'une discussion qui compare notre approche à la gestion de groupe aux travaux existants.

IV.1 Notion de groupe

Nous définissons dans un premier temps la notion de groupe mobile basée sur celle de domaine de sécurité, qui permet de sécuriser le partage des ressources entre entités du groupe. Puis, nous définissons les configurations de groupes possibles.

IV.1.1 Définitions

Un domaine de sécurité, noté DS, est défini par rapport à un centre d'intérêt professionnel ou privé. Concrètement, un domaine de sécurité délimite un ensemble d'entités mobiles et stationnaires par la possession d'un certificat numérique qui permet à ces entités de s'authentifier les unes auprès des autres. Ces certificats sont attribués après vérification appropriée de l'identité de l'entité par un tiers de confiance. Ainsi, une entité appartient à un domaine de sécurité si et seulement si elle possède le certificat numérique associé. Une entité peut appartenir à plusieurs domaines de sécurité suivant qu'elle possède les certificats numériques correspondants.

Le centre d'intérêt fédérateur d'un domaine de sécurité motive le partage des ressources. De plus, pour favoriser le partage des ressources en environnements mobiles, nous définissons la notion de groupe mobile au niveau d'un domaine de sécurité. Un tel groupe fait référence à un ensemble d'entités mobiles vérifiant les conditions suivantes : (i) elles sont dans la portée de communication directe les unes des autres ; et (ii) elles appartiennent au même domaine de sécurité. Nous considérons également que chaque entité isolée ou ne faisant partie d'aucun groupe est un singleton ou un groupe à un seul élément. Plus formellement, nous modélisons un groupe

mobile par un graphe G_{DS}, tel que $G_{DS} = (N_{DS}, A_{DS})$, où $N_{DS} \subset DS$ désigne les noeuds du graphe identifiés par l'ensemble des entités mobiles formant le groupe et, où A_{DS} est l'ensemble des arcs orientés du graphe représentant les connexions entre ces entités mobiles, tel que :

$$\forall\,(i,\,j) \in N_{DS} \times N_{DS},\; i \neq j,\; (i,\,j) \in A_{DS}\; et\; (j,\,i) \in A_{DS}$$

La notion de connexion entre deux entités mobiles se rapporte à la *disponibilité* d'un lien de communication entre elles. Une telle disponibilité peut dépendre de différents facteurs, notamment la qualité de service, la sécurité ou le coût de la connexion [103]. Dans notre contexte, les connexions sont définies par rapport au réseau mobile sans fil en mode *ad hoc* sous-jacent. Ainsi, si $(i,\,j) \in A_{DS}$ alors i peut établir une connexion directe sans fil (sans recours à des stations de base) avec j, ce qui signifie que i peut envoyer des messages à j. Les entités qui appartiennent à un groupe communiquent selon le modèle pair-à-pair *via* un réseau sans fil en mode *ad hoc*. Dans la section II.1, nous avons vu que les réseaux en mode *ad hoc* présentent des liens asymétriques qui sont tels que si l'entité i est dans la portée de communication directe de l'entité j, l'entité j n'est pas forcément dans la portée de communication de l'entité i. Dans notre définition d'un groupe, nous exigeons que toutes les entités mobiles soient dans la portée de communication les unes des autres. Cette condition est nécessaire à la gestion de groupe basée sur la découverte décentralisée des entités, ainsi qu'à la gestion de l'accès aux données partagées, présentée dans les chapitres V et VI.

Dans la section suivante, nous illustrons les différentes configurations de groupes mobiles possibles au sein d'un domaine de sécurité ou au niveau de plusieurs domaines de sécurité. Dans la suite de ce document, nous utilisons le terme groupe pour désigner un groupe mobile formé de plus d'une entité et le terme singleton pour désigner un groupe formé par une seule entité.

IV.1.2 Configurations de groupes

La définition d'un groupe impose que toutes les entités appartenant à un groupe appartiennent au même domaine de sécurité et soient dans la portée de communication directe les unes des autres. Ainsi, une entité ne peut appartenir qu'à un seul groupe par rapport à un domaine de sécurité. En revanche, une entité peut appartenir à plusieurs groupes relevant de domaines de sécurité différents. A titre d'exemple,

dans la figure IV.1, l'entité i appartient à deux domaines de sécurité DS_1 et DS_2 (Fig. IV.1-[a]), et à deux groupes différents : $Groupe_1$ associé au domaine de sécurité DS_1 (Fig. IV.1-[b]), et $Groupe_2$ associé au domaine de sécurité DS_2 (Fig. IV.1-[c]).

Toutefois, les ressources relevant d'un domaine de sécurité, c.-à-d., sécurisées de par leur appartenance à ce domaine, ne doivent pas être partagées avec des entités appartenant à un domaine de sécurité différent. Ainsi, une entité qui appartient à plusieurs groupes ne doit partager au sein d'un groupe que les ressources dont l'accès est sécurisé par l'appartenance au domaine de sécurité dont relève ce groupe. Nous considérons que les ressources dont l'accès est ainsi sécurisé sont plus spécifiquement les données. A titre d'illustration, considérons la figure IV.1. Soit $j \in DS_1 \; et \; j \in Groupe_1$ tel que j partage la donnée d au sein de $Groupe_1$, toutes les autres entités paires (c.-à-d., appartenant à $Groupe_1$) y ont accès, y compris i qui appartient à $Groupe_1$, mais également à $Groupe_2$ par rapport à DS_2. Supposons que i crée un réplica local de d. Si i ne distingue pas entre les données partagées au sein des différents groupes auxquels elle appartient, d peut être accédée par les entités de $Groupe_2$ qui appartiennent au domaine de sécurité DS_2. Il est par conséquent impératif que la gestion de groupe intègre le partage des ressources par rapport à un domaine de sécurité unique.

Il est également possible d'avoir plusieurs groupes au sein d'un même domaine de sécurité, à condition que ces groupes soient disjoints, c.-à-d., les entités formant ces groupes ne sont pas dans la portée de communication les unes des autre, ou bien, des entités se sont constituées en singleton afin de préserver leur ressources. A titre d'exemple, dans la figure IV.1, deux groupes disjoints $Groupe_1$ (Fig. IV.1-[a]) et $Groupe_4$ (Fig. IV.1-[d]) ainsi qu'un singleton (Fig. IV.1-[e]) relèvent du même domaine de sécurité DS_1. Les cas de connectivité partielle au sein d'un groupe sont résolus par le partitionnement du groupe en sous groupes respectant notre définition. Ils sont traités plus en détail dans la section IV.3.2. En revanche, si les entités formant des groupes appartenant au même domaine de sécurité se trouvent dans la portée de communication les unes des autres (Fig. IV.2-[a]), ces groupes sont fusionnés pour former un groupe unique (Fig. IV.2-[b]).

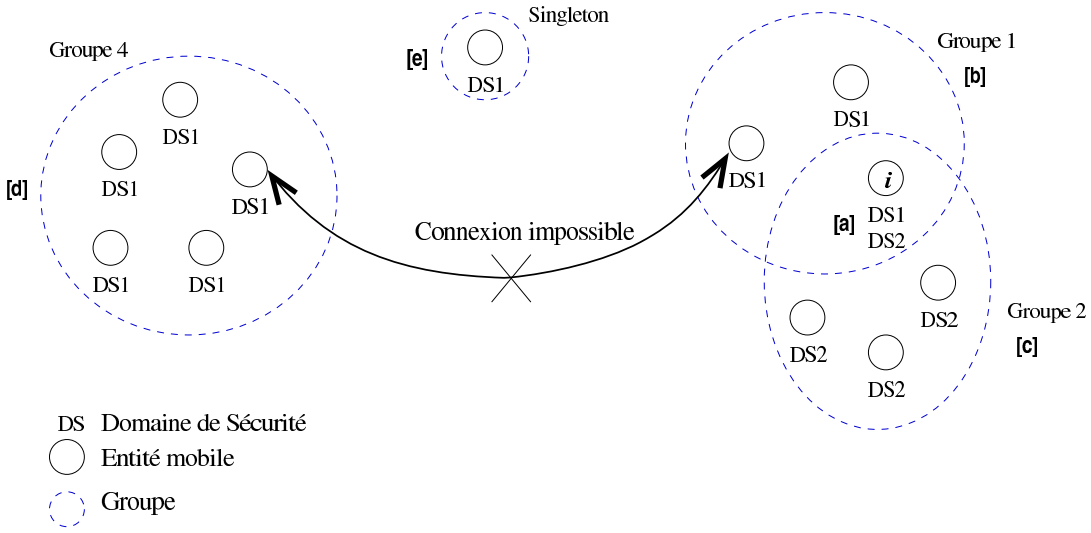

Figure IV.1 – Différentes configurations de groupes possibles.

IV.2 Découverte des entités mobiles

La découverte des entités mobiles constitue la première étape du processus de création d'un groupe. A l'issue de cette étape, chaque entité détient la liste des entités avec lesquelles la création d'un groupe est possible (c.-à-d., celles qui sont dans sa portée de communication et appartiennent aux même domaines de sécurité).

Suite à la discussion de la section III.1, la découverte des entités mobiles est décentralisée. Ainsi, toutes les entités mobiles, souhaitant faire partie d'un groupe, participent au processus de la découverte. Nous supposons, en outre, que la durée de ce processus est bornée au niveau de chaque entité. Au niveau de chaque entité mobile, le processus de la découverte se décompose en deux étapes.

La première étape consiste à découvrir les entités mobiles se trouvant dans la portée de communication. D'après la définition d'un groupe, les communications entre ses entités mobiles se font via des connexions sans fil directes (c.-à-d., sans recours à des stations de base ou à des protocoles de routage pour relayer les messages). Par conséquent, pour que chaque entité puisse établir la liste des entités se trouvant dans sa portée de communication, il lui suffit d'émettre périodiquement un message de *découverte*, puis, de recevoir en parallèle les messages de *découverte* des entités voisines pour enfin, établir une liste des entités paires en fonction des messages reçus.

La deuxième étape consiste à authentifier les entités mobiles se trouvant dans

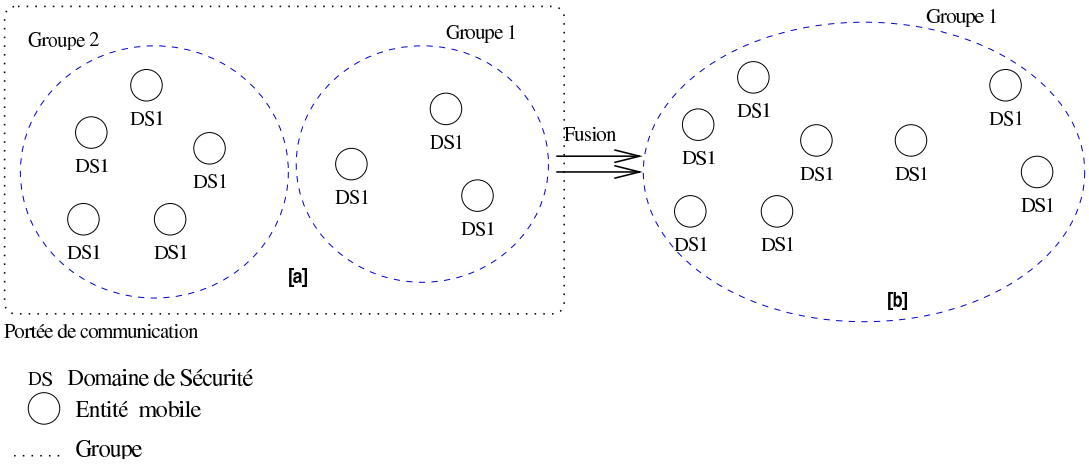

Figure IV.2 – Fusion de deux groupes mobiles.

la portée de communication par rapport à son domaine de sécurité. Pour s'assurer qu'une entité appartient au même domaine de sécurité, il suffit de vérifier qu'elle possède le certificat numérique associé au domaine de sécurité visé. Pour ce faire, chaque entité envoie dans son message de *découverte*, le certificat numérique associé au domaine de sécurité auquel elle appartient. Pour des raison de confidentialité, ce certificat doit être chiffré de sorte que seules les entités appartenant au même domaine de sécurité puissent le déchiffrer. Ainsi, chaque entité est en mesure d'établir la liste des entités avec lesquelles la création d'un groupe est possible.

Dans le processus de découverte, chaque entité mobile détermine sa liste d'entités candidates pour former un groupe sans recours à un coordinateur. Ceci permet de distribuer la charge de calcul et la consommation de ressources relatives à ce processus sur toutes les entités mobiles. Le protocole de découverte des entités mobiles est décrit plus en détail dans la section suivante.

IV.2.1 Protocole

Le protocole de découverte des entités mobiles est basé sur l'échange des messages de découverte *DecouvGp*. Ces messages sont envoyés périodiquement par des singletons voulant intégrer un groupe ou par des entités appartenant à un groupe afin de détecter d'éventuels changements dans la configuration de ce dernier. Les entités qui ne souhaitent plus faire partie d'un groupe ou les singletons qui ne souhaitent

pas intégrer des groupes désactivent l'envoi des messages de découverte *DecouvGp*.

Au niveau de chaque entité i, le message *DecouvGp* contient des informations concernant son identité ainsi que les certificats chiffrés des domaines de sécurité auxquelles elle appartient. Dans le cas d'un singleton, l'entité peut inclure dans son message de *découverte* plusieurs certificats chiffrés selon les domaines de sécurité auxquelles elle appartient. Ceci lui permet de fusionner avec le premier groupe (ou singleton) se trouvant dans sa portée de communication et appartenant à l'un de ses domaines de sécurité. Le choix des certificats à inclure dans le message de *découverte* permet de cibler les domaines de sécurité dans lesquels l'entité mobile souhaite former un groupe. Dans le cas d'une entité appartenant à un groupe, le message de *découverte*[1] contient le certificat associé au domaine de sécurité du groupe. Si une entité appartient à plusieurs groupes, elle émet autant de messages de découverte que de groupes auxquels elle appartient de par la gestion périodique de l'évolution des groupes (voir section IV.4). Notons qu'une entité peut appartenir à un groupe dans un domaine de sécurité et former un singleton par rapport à un autre domaine de sécurité. Dès la réception d'un message de *découverte* émis par une entité j, l'entité i procède à l'authentification des certificats inclus dans le message. Puis, met à jour les listes associées à chaque domaine de sécurité authentifié.

A l'issue de l'étape de découverte, une entité peut obtenir plusieurs listes d'entités candidates pour former des groupes. Toutefois, chaque liste correspond à un domaine de sécurité différent. Ainsi, il est possible qu'une entité participe à plusieurs groupes relevant de domaines de sécurité différents (voir section IV.1.2). La figure IV.3 illustre un tel cas, où à l'issue de l'étape de découverte des entités, l'entité e obtient deux listes de groupes possibles (Fig. IV.3-[a]), l'entité f obtient également deux listes de groupes possibles (Fig. IV.3-[b]) et l'entité g obtient trois listes de groupes possibles (Fig. IV.3-[c]). Etant donné qu'au niveau des entités e, f et g, les listes obtenues correspondent à des domaines de sécurité différents ($DS1$, $DS2$ et $DS3$), il est possible de créer trois groupes différents, chacun correspondant à l'une de ces listes (Fig. IV.3-[d]).

Après l'étape de découverte et d'authentification des entités allant former un groupe, le processus de création d'un groupe se poursuit par l'étape d'initialisation.

[1]Qui sert dans ce cas à détecter d'éventuelles changements dans la configuration du groupe.

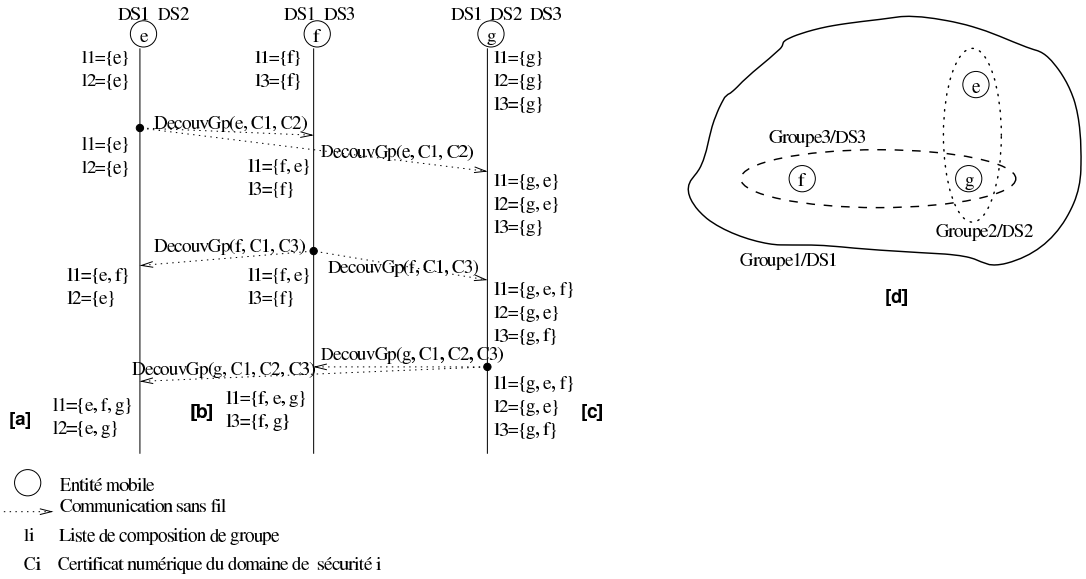

Figure IV.3 – Découverte des entités mobiles

IV.3 Initialisation d'un groupe

L'initialisation d'un groupe permet de créer le groupe proprement dit, afin d'offrir les propriétés décrites en introduction de ce chapitre. L'étape de découverte des entités allant former un groupe est complètement décentralisée. En revanche, pour l'initialisation du groupe, les entités allant former ce groupe désignent l'une d'entre elles comme étant le *leader*. Le *leader* du groupe est l'entité dont l'identificateur, *ID*, est un extrêmum (minimum ou maximum) des autres IDs du groupe. Ainsi, en se basant sur la liste des entités découvertes, chaque entité connaît l'identité du *leader*. Le recours à un *leader* permet de réduire le coût de l'initialisation du groupe en termes d'échanges de messages et donc en terme de consommation d'énergie. Nous décrivons le rôle du *leader* dans la section suivante. Puis, dans la section IV.3.2, nous proposons une solution pour la détection et la résolution d'éventuelles connectivités partielles.

IV.3.1 Rôle du *leader* du groupe

Dans un groupe, le rôle du *leader* consiste à coordonner les tâches nécessaires pour l'initialisation de ce dernier. Ces tâches consistent, dans un premier temps, à cen-

traliser la collecte des informations caractérisant les entités mobiles, appelées méta-données. Ces méta-données sont constituées d'informations concernant :

- Les données partagées localement stockées sur chaque entité appartenant au groupe. Chaque entité envoie au *leader* une structure de données identifiant les données localement stockées, hormis les données proprement dites, ainsi que des informations nécessaires pour la gestion de l'accès aux données, qui sont détaillées dans le chapitre suivant.
- Les ressources, en dehors des données, que chaque entité met à disposition pour le partage au sein du groupe. Dans le cas où l'entité appartient à plusieurs groupes, elle détermine la part de chaque ressource qu'elle souhaite partager au sein de chaque groupe. Les informations concernant la disponibilité des ressources constituent le profil de l'entité, qui sert en particulier pour la gestion adaptative de la disponibilité des données décrite dans le chapitre VI.
- La liste des entités obtenue à l'issue du processus décentralisé de découverte des entités mobiles, qui indique la composition du groupe telle qu'elle est vue par l'entité.

Dans un deuxième temps, le *leader* compare les compositions de groupe reçues, afin de détecter d'éventuelles connectivités partielles (voir Section IV.3.2) et déterminer la composition du groupe. Il fusionne ensuite les méta-données collectées afin de grouper les informations concernant l'ensemble des ressources disponibles pour le partage au sein du groupe. Enfin, le *leader* envoie les méta-données fusionnées à toutes les entités appartenant au groupe. Ceci leur permet d'avoir une vue globale et à jour des ressources (incluant les données) mises à disposition dans le groupe.

Le recours à un *leader* permet de réduire le coût de l'initialisation du groupe en termes d'échanges de messages. Chaque entité envoie un message (contenant les méta-données) au *leader*. Ce message est également reçu par les autres entités du groupe qui l'ignorent. Ceci est dû au mode *ad hoc*, où toutes les entités sont à l'écoute. Toutefois, le coût d'ignorer un message en terme de consommation d'énergie est inférieur au coût de sa réception effective [22, 37]. Ainsi, dans un groupe de n entités mobiles, $2\times(n-1)$ messages sont effectivement reçus lors de l'étape de l'initialisation du groupe. En revanche, une approche décentralisée (sans *leader*) pour l'échange des méta-données aurait induit plus d'envois/réceptions de messages. Dans un groupe de n entités, $n \times (n-1)$ messages seraient effectivement reçus lors de cette étape. Par conséquent, une approche décentralisée aurait induit une plus grande consommation d'énergie.

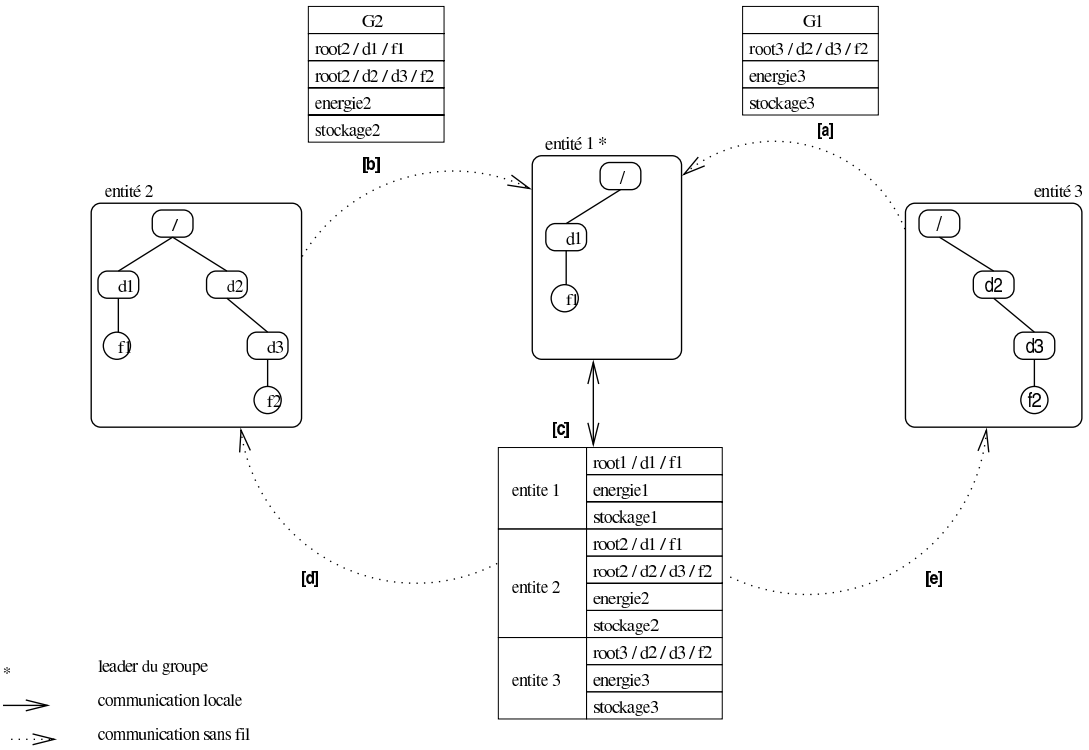

Figure IV.4 – Échange des méta-données dans un groupe

La figure IV.4 illustre un exemple d'échange de méta-données au sein d'un groupe formé de trois entités (*entité 1*, ..., *entité 3*). Après l'étape de la découverte des entités mobiles, *entité 1* est désignée comme étant le *leader* du groupe. Chaque entité lui envoie ses méta-données, (Fig. IV.4-[a]) et (Fig. IV.4-[b]). Ainsi, chaque entité envoie au *leader* une structure de données faisant état des ressources partagées. Le *leader* du groupe procède ensuite à la fusion des méta-données collectées, formant ainsi une structure de données qui contient toutes les méta-données des entités du groupe (Fig. IV.4-[c]). La dernière étape consiste à envoyer les méta-données du groupe à toutes les entités qui lui appartiennent (Fig. IV.4-[d]) et (Fig. IV.4-[e]).

Après l'étape d'envoi des méta-données fusionnées à toutes les entités appartenant au groupe, l'accès aux ressources partagées au sein du groupe devient possible. Mais, avant de procéder à la fusion et à l'envoi des méta-données, le *leader* du groupe doit détecter s'il existe des connectivités partielles au niveau des entités mobiles allant former le groupe. En effet, par définition, toutes les entités appartenant à un groupe doivent être dans la portée de communication directe les unes des autres.

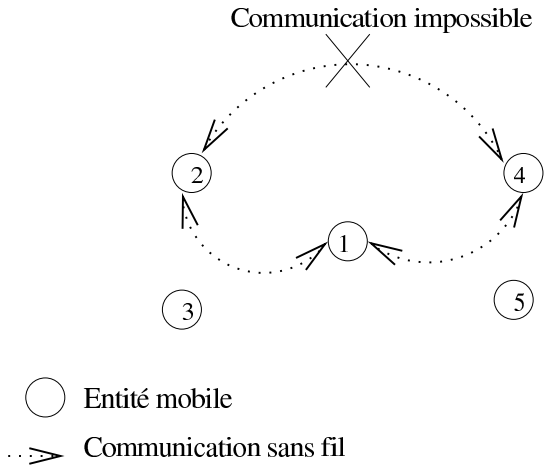

Figure IV.5 – Connectivité partielle.

Nous proposons dans la section suivante un protocole pour la détection et la résolution de la connectivité partielle dans un groupe.

IV.3.2 Connectivité partielle

Les situations de connectivité partielle sont fréquentes dans les réseaux sans fil en mode *ad hoc*. Elles se produisent lorsqu'une ou plusieurs entités sont dans la portée de communication de deux ensembles d'entités mobiles qui eux ne sont pas dans la portée de communication l'un de l'autre. La figure IV.5 illustre un exemple de connectivité partielle où l'entité e_1 est dans la portée de communication des entités e_2 et e_4, mais où les entités e_2 et e_4 ne sont pas dans la portée de communication l'une de l'autre. Ainsi, les entités e_1 et e_2, ou encore e_1 et e_4 peuvent faire partie du même groupe. Toutefois, l'entité e_1 ne peut pas faire partie de deux groupe (voir section IV.1.2). Par ailleurs, les entités e_2 et e_4 ne peuvent pas faire partie du même groupe.

La connectivité partielle dans les groupes mobiles est détectée et résolue pendant l'étape de l'échange des méta-données en comparant les compositions de groupes transmises par les entités mobiles à leur *leader*. La connectivité partielle peut générer des situations où des entités ayant élu un *leader* l_1 sont élues *leader* par d'autres entités qui ne sont pas dans la portée de l_1. Dans ce cas, il revient à chaque entité qui reçoit des méta-données, sachant qu'elle même a élu un autre *leader*, de les traiter.

Notons l l'identificateur du *leader*. Soient G_d la composition de groupe obtenue par l'entité l à l'issue de l'étape de découverte et G_m l'ensemble des identificateurs des entités ayant envoyé leurs méta-données à l'entité l. Nous notons G_l ($G_l = G_d \cap G_m$) l'ensemble des identificateurs des entités ayant été découvertes (authentifiées) par l'entité l et qui lui ont envoyé leurs méta-données. Ainsi, G_l représente un ensemble d'entités mobiles découvertes par l'entité l et qui l'ont élu *leader*. Si une entité appartient à $G_m - G_d$ alors cette entité n'a pas été authentifiée par le *leader* et donc ne sera pas prise en compte dans la création du groupe. D'un autre côté, si une entité appartient à $G_d - G_m$ alors cette entité a été authentifiée par le *leader*, mais elle n'est peut être plus dans sa portée de communication. Par conséquent, elle ne sera pas prise en compte dans la création du groupe. Dans un premier temps, le *leader* du groupe calcule l'intersection des compositions de groupe reçues g_j, telle que $j \in G_l$. Nous notons G_i le résultat de cette intersection. L'identificateur du *leader*, l, appartient à G_i. En effet, $\forall j \in G_l,\ l \in g_j$. Ainsi, G_i représente un ensemble d'entités mobiles qui sont dans la portée de communication directe les unes des autres et qui se sont mutuellement découvertes. Le *leader* calcule également l'union des compositions de groupe reçues g_j, telle que $j \in G_l$. Nous notons G_u le résultat de cette union. L'ensemble G_l est inclus dans l'ensemble G_u ($G_l \subseteq G_u$). En effet, $\forall j \in G_l,\ \{l,\ j\} \subseteq g_j$, ainsi, $\bigcup_{j \in G_l}\{l,\ j\} \subseteq \bigcup_{j \in G_l} g_j$, par conséquent, $G_l \subseteq G_u$. Ainsi, G_u représente un ensemble d'entités mobiles qui sont dans la portée de communication directe des entités appartenant à G_l. La seconde étape consiste à comparer les valeurs G_l, G_i et G_u. Trois cas sont possibles :

- Si $G_l = G_i = G_u$ alors toutes les compositions reçues sont identiques entre elles et égales à G_l. Dans ce cas, le *leader* valide G_l en tant que composition de groupe, fusionne les méta-données et les envoie aux entités appartenant à G_l.
- Si ($G_l = G_i$ et $G_i \neq G_u$) ou ($G_i = G_u$ et $G_i \neq G_l$) alors les entités qui appartiennent à $G_u - G_l$ ne sont pas dans la portée de communication directe du *leader*, mais sont dans la portée de communication directe de quelques entités appartenant à G_l. Dans ce cas, le *leader* valide G_l en tant que composition de groupe, fusionne les méta-données et les envoie aux entités appartenant à G_l.
- Si ($G_l = G_u$ et $G_i \neq G_u$) ou ($G_l \neq G_u$ et $G_i \neq G_u$) alors les entités appartenant à G_i sont dans la portée de communication directe de toutes les entités qui appartiennent à G_l, mais toutes les entités appartenant à G_l ne sont pas dans la portée de communication directe les unes des autres. Notons que le *leader* appartient obligatoirement à G_i. Dans ce cas, la décision revient au *leader*. Une première

option est de former un groupe à partir des entités appartenant à G_i et exclure toutes les entités appartenant à $G_l - G_i$ en leur envoyant le message *ExGp*. Les entités qui reçoivent le message *ExGp* recommencent l'étape de découverte des entités mobiles afin de créer un autre groupe. Notons que cette étape de découverte des entités mobiles ne créera pas d'interférence avec le groupe formé par le *leader*, car les entités formant ce dernier ignoreront tous les messages jusqu'à la prochaine période de mise à jour du groupe (voir section IV.4.1). L'autre alternative qui se présente au *leader* consiste à choisir un sous ensemble de G_l dont les entités sont dans la portée de communication directe les unes des autres pour former un groupe. Les critères de ce choix peuvent être liés à la taille du groupe (p.ex., choisir le groupe le plus grand) ou aux ressources disponibles (p.ex., choisir le groupe avec le plus de ressources disponibles). Ainsi, le *leader* calcule la composition du groupe G, et exclut les entités qui n'appartiennent pas à G en leur envoyant le message *ExGp*.

Si une entité reçoit des méta-données fusionnées et envoyées par un *leader* alors qu'elle appartient déjà à un autre groupe, il lui revient d'ignorer ces méta-données ou de quitter son groupe actuel pour se joindre à celui de ce *leader*. La figure IV.6 illustre les trois cas décrits ci-dessus.

Dans la figure IV.6-[a], les entités 2, 3 et 4 ont élu l'entité 1 comme *leader*, tandis que les entités 5 et 6 ont élu l'entité 3 comme *leader*. Chaque entité envoie à son *leader* ses méta-données. Au niveau du *leader* 1 les variables G_l, G_i et G_u ont les valeurs suivantes : $G_{l1} = \{1,\ 2,\ 3,\ 4\}$, $G_{i1} = \{1,\ 2,\ 3,\ 4\}$ et $G_{u1} = \{1,\ 2,\ 3,\ 4,\ 5,\ 6\}$, ce qui correspond au cas où $G_l = G_i$ et $G_i \neq G_u$. Au niveau du *leader* 3 les variables G_l, G_i et G_u ont les valeurs suivantes : $G_{l3} = \{3,\ 5,\ 6\}$, $G_{i3} = \{3,\ 5,\ 6\}$ et $G_{u3} = \{3,\ 5,\ 6\}$, ce qui correspond au cas où $G_l = G_i = G_u$. Ainsi, le *leader* 1 valide G_{l1} en tant que composition de groupe, fusionne les méta-données et les envoie aux entités appartenant à G_{l1}. De son côté, le *leader* 3 valide son G_{l3} en tant que composition de groupe. Au moment où l'entité 3 reçoit les méta-données envoyées par le *leader* 1, car elle appartient à G_{l1}, il lui revient de les ignorer, c'est notamment le cas dans cette figure, ou de quitter son groupe pour joindre le groupe formé par le *leader* 1.

Dans la figure IV.6-[b], les entités 2, 3 et 4 ont élu l'entité 1 comme *leader*, tandis que les entités 5 et 6 ont élu l'entité 2 comme *leader*. Au niveau du *leader* 1 les variables G_l, G_i et G_u ont les valeurs suivantes : $G_{l1} = \{1,\ 2,\ 3,\ 4\}$, $G_{i1} = \{1,\ 2,\ 3,\ 4,\ 5,\ 6\}$ et $G_{u1} = \{1,\ 2,\ 3,\ 4,\ 5,\ 6\}$, ce qui correspond au cas où $G_i = G_u$ et $G_i \neq G_l$ qui est similaire au cas où $G_l = G_i$ et $G_i \neq G_u$ traité dans l'exemple ci-dessus. Au niveau

du *leader* 2 les variables G_l, G_i et G_u ont les valeurs suivantes : $G_{l2} = \{2,\ 5,\ 6\}$, $G_{i2} = \{2,\ 3,\ 4,\ 5,\ 6\}$ et $G_{u2} = \{2,\ 3,\ 4,\ 5,\ 6\}$, ce qui correspond au même cas. Dans l'exemple de la figure IV.6-[b] l'entité 2 a choisi de se joindre au groupe formé par le *leader* 1.

Dans la figure IV.6-[c], l'entité 5 a élu l'entité 3 comme *leader*, tandis que les entités 2, 4 et 6 ont élu l'entité 1 comme *leader*. Au niveau du *leader* 1 les variables G_l, G_i et G_u ont les valeurs suivantes : $G_{l1} = \{1,\ 2,\ 4,\ 6\}$, $G_{i1} = \{1,\ 2,\ 4,\ 6\}$ et $G_{u1} = \{1,\ 2,\ 3,\ 4,\ 5,\ 6\}$, ce qui correspond au cas où $G_l = G_i$ et $G_i \neq G_u$. Au niveau du *leader* 3 les variables G_l, G_i et G_u ont les valeurs suivantes : $G_{l3} = \{3,\ 5,\ 6\}$, $G_{i3} = \{3,\ 5,\ 6\}$ et $G_{u3} = \{3,\ 5,\ 6\}$, ce qui correspond au cas où $G_l = G_i = G_u$. Ainsi, le *leader* 1 valide G_{l1} en tant que composition de groupe, fusionne les méta-données et les envoie aux entités appartenant à G_{l1}. De son côté, le *leader* 3 valide G_{l3} en tant que composition de groupe, fusionne les méta-données et les envoie aux entités appartenant à G_{l3}. Au moment où l'entité 6 reçoit les méta-données envoyées par le *leader* 3, il lui revient de les ignorer, c'est notamment le cas dans cette figure, ou de quitter son groupe pour joindre le groupe formé par le *leader* 3.

Dans la figure IV.6-[d], les entités 2, 3, 4, 5 et 6 ont élu l'entité 1 comme *leader*. Au niveau du *leader* 1 les variables G_l, G_i et G_u ont les valeurs suivantes : $G_{l1} = \{1,\ 2,\ 3,\ 4,\ 5,\ 6\}$, $G_{i1} = \{1\}$ et $G_{u1} = \{1,\ 2,\ 3,\ 4,\ 5,\ 6\}$, ce qui correspond au cas où $G_l = G_u$ et $G_i \neq G_u$. Dans ce cas, deux choix s'offrent au *leader*, soit il forme un groupe à partir des entités appartenant à G_{i1} et exclut toutes les autres entités (Fig. IV.6-[d1]), soit il selectionne un sous ensemble de G_{l1} selon un critère quantitatif ou qualitatif pour former un groupe et exclut toutes les autres entités (Fig. IV.6-[d2]).

IV.3.3 Gestion des ressources du groupe

Le but de former des groupes par rapport à des domaines de sécurité est d'offrir aux entités appartenant au groupe la possibilité de partager les ressources (données, énergie et espace de stockage) de manière transparente. Ainsi, chaque entité appartenant au groupe a accès à toutes les ressources disponibles dans ce dernier. La figure IV.7 illustre un groupe formé de trois entités mobiles. Les données manipulées sont stockées sous la forme d'un système de fichier répliqué. L'entité $entité_3$ stocke localement une copie de f_2 et sait qu'un réplica de f_2 existe sur l'entité $entité_2$, ainsi que f_1 est disponible sur les entités $entité_2$ et $entité_1$.

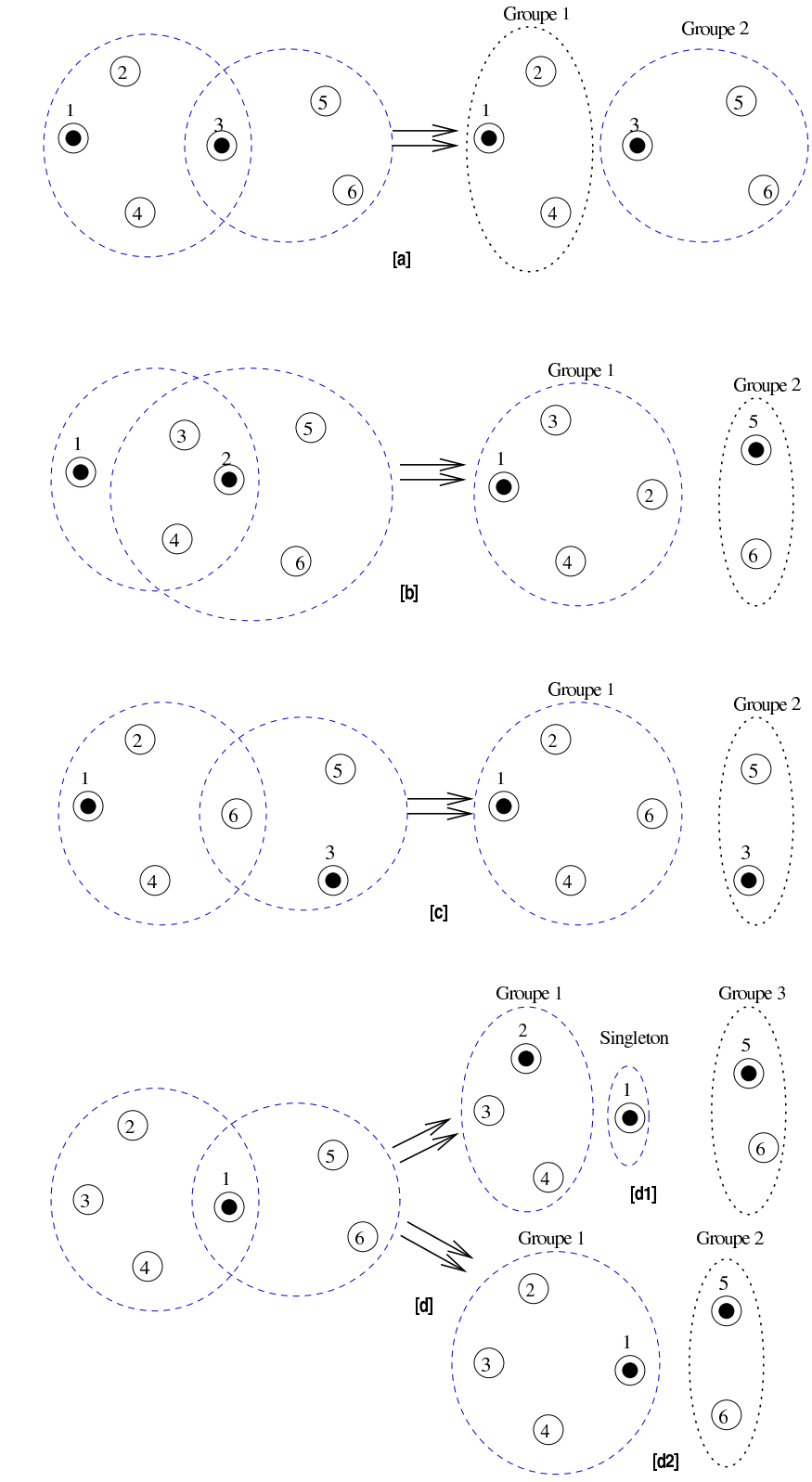

Figure IV.6 – Résolution de la connectivité partielle.

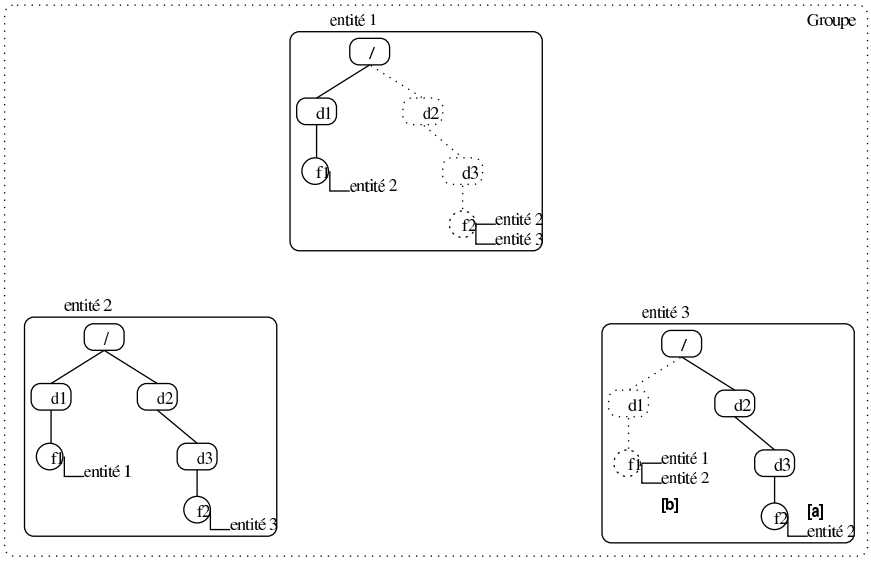

Figure IV.7 – Accès aux données dans un groupe mobile.

Dans notre contribution, nous nous intéressons plus particulièrement à la gestion de l'accès aux données partagées. Par construction du groupe, chaque entité connaît pour chaque donnée locale, la liste des identificateurs des autres entités qui stockent un réplica de cette donnée (Fig. IV.7-[a]). Cette information servira par la suite pour le maintien de la cohérence des données (voir chapitre V), ainsi que pour offrir une disponibilité des réplicas en fonction de l'état des ressources (voir chapitre VI). On associe également à chaque entité du groupe, la liste des données accessibles dans le groupe, mais qui ne sont pas localement stockées, ainsi que les identificateurs des entités ayant des réplicas locaux de ces dernières (Fig. IV.7-[b]). Nous obtenons ainsi une structure de nommage qui inclut les noms de toutes les données stockées dans le groupe, ainsi que les identificateurs des entités ayant des copies locales. Nous rappelons que cette structure de nommage est construite par le *leader* du groupe lors de l'étape de l'échange des méta-données (voir Section IV.3.1).

L'accès à une donnée non locale génère la création d'un réplica local de cette dernière à partir de l'une des entités ayant une copie locale. Ce choix nous permet, en plus d'augmenter la disponibilité des données, de réduire les communications sans fil que les accès distants nécessitent. La création d'un nouveau réplica, au niveau d'une entité au sein du groupe, modifie la structure de nommage construite par le *leader*. Ainsi, la gestion de l'évolution du groupe, décrite dans la section suivante, permet d'intégrer les changements apparus. En ce qui concerne les autres ressources

partagées au sein du groupe, leur gestion qui permet de réduire leur consommation est réalisée au travers de la gestion de l'accès aux données, et en particulier la gestion de la cohérence et de la disponibilité des données (voir chapitre V et VI).

IV.4 Gestion de l'évolution du groupe

Dans les environnements mobiles, la nature nomade des utilisateurs requiert la prise en charge des perpétuels changements qui peuvent survenir dans la composition d'un groupe. Ces changements concernent en particulier les entités quittant le groupe et celles voulant le joindre. Ils doivent être traités de manière à assurer que les entités appartenant au groupe aient une vision globale et à jour des ressources disponibles (et notamment les données accessibles) au sein du groupe. La gestion de l'évolution du groupe doit se faire de manière transparente. Elle doit également se faire en minimisant les communications entre les entités appartenant au groupe.

Les changements dans la composition du groupe sont détectés à l'étape de découverte des entités mobiles, tandis que les changements dans la disponibilité des ressources sont détectés à l'étape suivante où les méta-données sont échangées. L'idéal serait de détecter ces changements instantanément. Ceci implique que les entités du groupe soient constamment à l'affût du moindre changement de configuration (ou de ressources), ce qui consiste à envoyer des messages pour annoncer ces changements. Ces échanges de messages génèrent une importante consommation d'énergie au niveau des entités impliquées. Pour y remédier, nous proposons de rendre la découverte des changements de configuration ou de ressources périodique.

Dans la section IV.4.1, nous proposons une heuristique qui nous permet d'adapter la périodicité de la mise à jour du groupe à la fréquence des changements. La mise à jour du groupe consiste à déclencher périodiquement le processus de reconfiguration du groupe qui est décrit dans la section IV.4.2. Le *leader* du groupe est également changé périodiquement du fait de sa possible mobilité mais également pour distribuer la charge au sein du groupe, comme décrit dans la section IV.4.3.

IV.4.1 Périodicité de la mise à jour d'un groupe

Pour trouver un compromis entre la détection instantanée des changements et la consommation induite d'énergie, nous définissons la période, T_g, de mise à jour

du groupe. Cette période est adaptée à la fréquence des changements de la composition du groupe, qui peut être élevée ou réduite. Ainsi, dans des groupes à faible mobilité/dynamicité, où il y a peu de changements au cours du temps (fréquence des changements réduite), il est plus avantageux de choisir une valeur élevée pour la période de mise à jour du groupe, ce qui permet de réduire et d'espacer la mise à jour du groupe. En revanche, dans des groupes à mobilité/dynamicité élevée, il est nécessaire d'affecter une valeur réduite à la période de mise à jour du groupe, ce qui permet de détecter un plus grand nombre de changements et plus rapidement.

La période de mise à jour du groupe est calculée en fonction des changements antérieurs, et est adaptée en conséquence. Soit T_{last} la dernière valeur affectée à la période T_g. La nouvelle valeur de T_g est calculée en fonction des changements apparus durant la période T_{last} qui vient se s'écouler comme suit, où t dénote la variable de temps. Soit $\gamma = \frac{C_{prev}}{C_{last}}$, où C_{last} est le nombre de changements[2] pendant la dernière

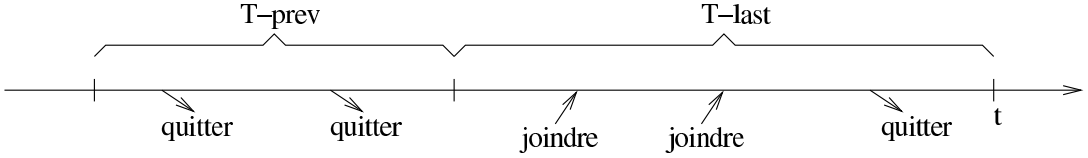

Figure IV.8 – Périodes de détection des changements dans un groupe.

période (c.-à-d., pendant l'intervalle $[t - T_{last}, t]$) et C_{prev} étant le nombre de changements pendant l'intervalle $[t-(T_{prev}+T_{last}), t-T_{last}]$, T_{prev} étant la valeur précédente de T_g, et T_{last} étant la dernière valeur de T_g (cf. figure IV.8). En fonction de la valeur de γ, nous avons deux cas :

- Si γ est inférieur à 1, la valeur de T doit être diminuée car le nombre de changements survenus dans le groupe lors de la dernière période (C_{last}) est supérieur au nombre de changements survenus lors de la période précédente (C_{prev}).
- En revanche, si γ est supérieur à 1, la valeur de T doit être augmentée, car le nombre de changements survenus dans le groupe lors de la dernière période (C_{last}) est inférieur au nombre de changements survenus lors de la période précédente (C_{prev}). Cette nouvelle valeur de T permet de retarder la mise à jour du profil du groupe pour minimiser les communications et ainsi diminuer la consommation d'énergie.

[2]Les changements sont dus à la mobilité des entités mobiles, ce qui génère leur séparation du groupe ou leur fusion au groupe

Dans les deux cas, la nouvelle valeur de T_g devient :

$$T_g \times \frac{C_{prev}}{C_{last}} \tag{IV.1}$$

La valeur de la période de mise à jour du groupe T_g est calculée au niveau de chaque entité mobile. A l'issue de cette période, chaque entité active le processus de mise à jour du groupe.

IV.4.2 Mise à jour du groupe

La mise à jour du groupe doit permettre de détecter les changements apparus dans la composition du groupe (départ/arrivée de certaines entités), ainsi que les changements relatifs aux ressources disponibles au niveau du groupe. Suite à la mise à jour périodique du groupe, les changements apparus avant la fin de la période de mise à jour ne sont détectés qu'à l'issue de celle-ci. De plus, la mise à jour du groupe permet de revoir la durée fixée pour l'étape de découverte, qui est calculée et envoyée par le *leader* du groupe.

A l'issue de la période de mise à jour du groupe, chaque entité appartenant au groupe déclenche la découverte des entités mobiles (voir section IV.2) afin de détecter les éventuels changements dans la composition du groupe. Ainsi, les nouvelles entités voulant faire partie du groupe sont détectées, de même que celles qui ne souhaitent plus en faire partie. A titre d'exemple, soit i un singleton voulant se joindre au groupe G préalablement constitué. Dans ce cas, le singleton émet périodiquement un message de découverte (*DecouvGp*). Ce message est ignoré par les entités appartenant au groupe jusqu'à la fin de sa période de mise à jour (T_g). A la fin de cette période, ces entités déclenchent l'étape de découverte qui se traduit par l'émission d'un message de découverte. Le singleton i reçoit à son tour les messages *DecouvGp*, et répond, s'il souhaite se joindre au groupe, par un autre message *DecouvGp* afin de notifier de sa présence. Ainsi, i est intégré dans le processus de découverte des entités mobiles. Dans le cas où deux groupes G_1 et G_2 appartenant au même domaine de sécurité se trouvent dans la portée de communication direct l'un de l'autre, la décision de fusionner les deux groupe revient à leurs *leaders*. En effet, les périodes de mise à jour diffèrent d'un groupe à l'autre. Par conséquent, les messages de découverte émis par les entités du groupe G_1, par exemple, sont ignorés par les entités du groupe G_2 car en dehors de leur période de mise à jour et *vice versa*. Il revient ainsi au *leader* du

groupe G_2 (resp. du groupe G_1) de traiter les messages de découverte émis par les entités du groupe G_1 (resp. du groupe G_2) et de déclencher la découverte des entités mobiles au niveau de son groupe s'il souhaite la fusion des deux groupes.

Dans le cas où une entité se déconnecte d'un groupe, nous distinguons deux situations : l'entité se déconnecte volontairement, mais reste dans la portée de communication du groupe. Dans ce cas, l'entité déconnectée n'émet plus le message *DecouvGp* à la fin de la période T_g. Ainsi, elle ne sera plus détectée par les entités appartenant au groupe et peut se constituer en singleton. Dans le cas où l'entité se déconnecte involontairement, cette déconnexion est due soit à une panne de batterie, dans ce cas l'entité n'est plus en mesure d'émettre des messages, soit la déconnexion est due à la mobilité de l'entité ou à la connectivité intermittente. Dans ce cas, l'entité n'est plus dans la portée de communication du groupe et par conséquent elle ne sera pas détectée lors de l'étape de la découverte des entités mobiles.

La découverte des entités mobiles nous permet de détecter les changements apparus dans la composition du groupe. Quant aux changements relatifs aux ressources disponibles au niveau du groupe, ils sont détectés par l'étape de l'initialisation du groupe (voir section IV.3). Le *leader* du groupe fusionne les méta-données reçues, puis les envoie aux entités du groupe qui obtiennent ainsi une vue globale et à jour des ressources disponibles.

IV.4.3 Gestion du *leader* du groupe

L'entité désignée comme *leader* du groupe est changée périodiquement du fait de sa possible mobilité mais aussi pour distribuer la charge, en termes de consommation d'énergie et de temps de calcul sur toutes les entités du groupe. La périodicité de l'élection d'un *leader* est adaptée à la fréquence des changements dans le groupe. Elle est calculée selon l'heuristique décrite dans la section IV.4.1 qui permet de déterminer la périodicité de mise à jour du groupe. Le principe de l'élection d'un *leader* consiste à choisir l'entité dont l'identificateur *ID* est un extrêmum (minimum ou maximum) des autres IDs des entités du groupe et qui n'a pas été élue *leader* auparavant. L'élection du *leader* du groupe est décentralisée de sorte que chaque entité appartenant au groupe élit son *leader* indépendamment des autres entités selon l'algorithme suivant.

Soit N l'ensemble des IDs des entités mobiles voulant former un groupe et se trouvant dans la portée de communication directe les unes des autres. Soit i une en-

tité mobile telle que $ID_i \in N$. L'ensemble L_i, défini au niveau de l'entité i, désigne l'ensemble des IDs des entités connues de i qui ont été *leader*. L'ensemble L_i est initialisé au moment où i se joint à un groupe. A cette fin, chaque entité j inclut la valeur de L_j dans son message de découverte (*DecouGp*) au même titre que le certificat du domaine de sécurité (c.-à-d., chiffré de sorte que seules les entités appartenant au même domaine de sécurité puissent le déchiffrer). Dans un singleton s, l'ensemble L_s est égal à l'ensemble vide ($L_s = \phi$). A l'issue de l'étape de découverte des entités mobiles, chaque entité i calcule l'ensemble L_i qui est égal à l'union de tous les ensembles (L_j) reçus dans les messages de découverte (y compris L_i). Ensuite, chaque entité calcule l'ID de l'entité *leader* en choisissant l'extrêmum des IDs appartenant à $G_{di} - L_i$, où G_{di} est l'ensemble des IDs des entités découvertes par i, afin d'éviter la réélection des anciens *leaders*. L'ensemble L_i est par la suite mis à jour en lui ajoutant l'ID du *leader* ainsi élu. Dans le cas où toutes les entités appartenant à G_{di} ont déjà été désignées comme *leader* du groupe ($G_{di} \subseteq L_i$), l'ensemble L_i est réinitialisé à l'ensemble vide.

La mobilité des entités (leur départ du groupe ou leur arrivée dans le groupe) n'affecte pas l'élection du *leader*. En effet, le caractère décentralisé de notre algorithme le rend indépendant de la dynamicité du groupe. Par ailleurs, étant donné que le rôle du *leader* se concentre dans l'étape de l'initialisation du groupe, son départ du groupe avant la fin de la période qui lui est assignée n'affecte pas les fonctionnalités du groupe. Dans tous les cas, une autre entité est élue *leader* à l'issue de la période de mise à jour du groupe. De plus, ce protocole garantit qu'un *leader* unique existe pour chaque période de temps. En effet, sachant que les entités mobiles mutuellement découvertes possèdent toutes le même ensemble L, supposons que deux *leaders* distincts ont été élus. Cela signifie qu'il existe au moins deux entités i, j appartenant à N qui ont élu deux *leaders* distincts. Soient e_i l'extrêmum de $G_{di} - L_i$ et e_j l'extrêmum de $G_{dj} - L_j$, $e_i \neq e_j$ signifie que soit $e_j \notin G_{di}$ et donc que l'entité e_j n'a pas été découverte par l'entité i, soit $e_i \notin G_{dj}$ et donc que l'entité e_i n'a pas été découverte par l'entité j. Dans ce cas, nous sommes en la présence d'une connectivité partielle (voir la section IV.3.2), ce qui contredit l'hypothèse de départ où N est l'ensemble des IDs des entités se trouvant dans la portée de communication directe les unes des autres.

IV.5 Evaluation

La gestion des groupes, avec toutes les fonctionnalités qu'elle supporte, est réalisée dans le souci de minimiser la consommation des ressources. En effet, les échanges de messages entre les entités mobiles, dans le cadre de la gestion du groupe, sont réduits par l'introduction d'un *leader* du groupe. La réduction des communications sans fil a un impact positif sur la réduction de la consommation d'énergie au niveau de toutes les entités appartenant au groupe. Dans la section VII.2.3.1, nous avons évalué cette consommation par l'intégration de la gestion de groupe dans un prototype de système de fichiers distribué mobile. Nous avons en outre montré que l'énergie consommée par une entité mobile pendant l'étape de découverte est comparable à l'énergie consommée par une entité dans l'état d'écoute pendant la même durée. En ce qui concerne le surcoût mémoire, les méta-données générées par la gestion du groupe restent négligeables par rapport à la taille des données, comme nous le montrons dans la section VII.2.1.1. Par ailleurs, les temps de réponse sont évalués dans la section VII.2.2.1 où nous montrons qu'ils sont proportionnels à la taille du groupe.

Le tableau IV.1 indique le nombre de messages émis, reçus et ignorés pour créer, quitter ou joindre un groupe de n entités mobiles. La complexité théorique de la gestion d'un groupe formé de n entités est de l'ordre de $O(N)$. De manière générale, l'entité désignée pour diriger le groupe (le *leader*) envoie/reçoit plus de messages que les autres entités appartenant au groupe. Par conséquent, la charge en termes de temps de calcul ainsi que de la consommation d'énergie est beaucoup plus élevée au niveau du *leader* du groupe, d'où la nécessité de changer de *leader* périodiquement. Dans la section VII.2.3.1, nous avons évalué l'énergie consommée par le *leader* pendant l'étape d'initialisation du groupe. Elle s'élève à environ 32% de l'énergie consommée par la totalité du groupe pendant cette étape.

IV.6 Discussion

Dans un environnement mobile sans fil où les entités mobiles se déplacent dans des environnements inconnus, la création de groupe de manière spontanée et selon certaines contraintes de sélection (appartenir au même domaine de sécurité dans notre cas) ne peut être réalisée sans le recours à la découverte décentralisée des entités mobiles. La découverte des entités mobiles peut être associé à celle de la découverte

Noeud	Etape	Emis	Reçus	Ignorés
Création d'un groupe à partir de n singletons				
Leader	Découverte	1	n-1	0
	Echange M-D	1	n-1	0
Peer	Découverte	1	n-1	0
	Echange M-D	1	1	n-2
Groupe	Découverte	n	n*(n-1)	0
	Echange M-D	n	2*(n-1)	(n-2)(n-1)
***p* entités quittent le groupe**				
Leader	Découverte	1	n-P-1	0
	Echange M-D	1	n-P-1	0
Peer	Découverte	1	n-P-1	0
	Echange M-D	1	1	n-p-2
Groupe	Découverte	n-p	(n-p)(n-P-1)	0
	Echange M-D	n-p	2*(n-p-1)	(n-p-2)(n-p-1)
***p* entités se joignent au groupe**				
Leader	Découverte	1	n+p-1	0
	Echange M-D	1	n+p-1	0
Peer	Découverte	1	n+p-1	0
	Echange M-D	1	1	n+p-2
Groupe	Découverte	n+p	(n+p)(n+p-1)	0
	Echange M-D	n+p	2*(n+p-1)	(n+p-1)(n+p-2)

Table IV.1 – Coût de la gestion de groupe en termes de messages.

de services décrite dans la section III.1. Mais, seuls les protocoles de découverte de services qui supportent le fonctionnement décentralisé, sont adaptés pour la découverte des entités mobiles. En effet, la nature dynamique des groupes et leur autoformation (ou formation spontanée) impose un certain nombre de contraintes. Les groupes sont gérés en mode pair-à-pair. Ce mode permet un fonctionnement sans recours à un coordinateur prédéfini et permet d'équilibrer les charges et la consommation des ressources entre différentes entités. Désigner une entité comme étant un coordinateur pour la découverte de services (p.ex., le **DA** dans SLP) augmente la charge de cette dernière car elle devra être constamment à l'écoute, soit des annonces de services, soit des requêtes de découverte de services. Il pose également le problème de comment désigner un tel coordinateur parmi des entités mobiles dont le comportement est imprévisible.

Dans [114], les auteurs proposent une solution pour prendre en charge la nature imprévisible des réseaux mobiles sans fil en mode *ad hoc* par rapport à la découverte des entités mobiles. Leur solution pour la découverte des entités mobiles est similaire à la notre sauf en ce qui concerne la notion de portée de communication. Dans [114], une entité est considérée être dans la portée de communication d'une autre entité si elles sont à une distance déterminée, nommée distance de sécurité (*safe distance*), l'une de l'autre. Ce critère est déterminé selon l'intensité de son signal des messages reçus. L'intensité du signal d'un message reçu *via* une connexion sans fil directe (c.-à-d., sans recours à des stations de base ou à des protocoles de routage pour relayer les messages) dépend, entre autre, de la distance qui sépare son émetteur du récepteur. Par conséquent, une entité mobile qui reçoit un message peut déterminer, d'après l'intensité du signal, la proximité de son émetteur. Les entités dont les messages sont reçus avec une faible intensité ne sont pas retenues, car il est supposé qu'elles ont une forte probabilité de devenir inaccessibles lors des étapes suivantes. Mais, le contraire peut être vrai, car une entité dont le message a été reçu avec une faible intensité peut se rapprocher et non pas s'éloigner. Notre gestion des groupes mobiles permet de remédier à ce problème *via* la mise à jour périodique des groupes qui permet de détecter les changements dus à la mobilité des entités.

V Gestion de la cohérence

Comme nous l'avons mentionné dans les chapitres précédents, nous nous intéressons particulièrement à la gestion de l'accès aux données en environnements mobiles. L'accès à des données répliquées sur différents sites soulève le problème de leur cohérence. La gestion de la cohérence se décompose en deux grandes catégories, la gestion de la cohérence forte et la gestion de la cohérence faible, comme détaillé dans le chapitre III. La connectivité intermittente dans les environnements mobiles constitue l'argument majeur dans le choix de la gestion de la cohérence faible des données, dans les systèmes distribués existants dédiés à ces environnements. En effet, la grande majorité de ces systèmes est dédiée à des environnements où les réseaux sous-jacents sont des réseaux cellulaires de type GSM. Par conséquent, les communications entre les entités du système distribué sont relayées par des stations de base qui peuvent devenir inaccessibles du fait de la mobilité des entités et générer une connectivité intermittente. Cependant, le mode *ad hoc* des réseaux sans fil permet de constituer des réseaux locaux où la communication directe remédie à la connectivité intermittente rencontrée dans les réseaux à base d'infrastructure. Ce type de réseaux locaux offre un support adéquat pour le partage des ressource et notamment pour la collaboration synchrone, qui par ailleurs, nécessite une gestion forte de la cohérence. De plus, le maintien de la cohérence forte devient possible grâce à la connectivité qu'offrent ces réseaux locaux.

Dans ce chapitre, nous introduisons une gestion hybride de la cohérence adaptée à la connectivité du réseau sous-jacent ainsi qu'aux besoins des applications. Au niveau d'un groupe, nous adoptons la gestion de la cohérence forte afin d'offrir aux applications qui nécessitent une collaboration synchrone la possibilité d'accès à la même version des réplicas d'une donnée. Cependant, au niveau d'un domaine de sécurité, nous adoptons la gestion de la cohérence faible, ce qui permet de manipuler les données répliquées de manière concurrente et indépendante dans des groupes

disjoints. Notre gestion de la cohérence est basée sur le maintien d'un historique des mises à jour au niveau de chaque réplica de données. L'historique des mises à jour inclut les identités des entités qui ont participé aux mises à jour. Ceci nous permet de traquer la mobilité des entités entre différents groupes, sans la contraindre. Du fait de la gestion de la cohérence faible, au niveau d'un domaine de sécurité, nous proposons un protocole pour la détection des conflits qui surviennent lors de la synchronisation des réplicas divergents. La gestion de la cohérence est réalisée de sorte à minimiser la consommation des ressources. En effet, les différents protocoles présentés dans ce chapitre ont été conçus de manière à réduire les échanges de messages entre les entités mobiles [15, 17].

Dans la section V.1, nous définissons les historiques de mises à jour qui servent à déterminer, dans un premier temps, la liste des mises à jour nécessaires pour synchroniser les réplicas, et le cas échéant la divergence des mises à jour. La section V.2 présente notre solution à la gestion hybride de la cohérence. Puis, dans la section V.3, nous présentons le protocole d'accès aux données basé sur notre gestion hybride de la cohérence et la propagation paresseuse des mises à jour. Dans la section V.4, nous abordons la synchronisation des réplicas qui est indispensable du fait de la composante optimiste de notre gestion de la cohérence. Dans la section V.5, nous présentons le cas particulier de la synchronisation avec la copie de référence qui est inhérente à notre découpage en domaines de sécurité et groupes. Enfin, avant de conclure ce chapitre, nous présentons dans la section V.6 une évaluation de notre gestion de la cohérence en termes de complexité théorique à travers le nombre de messages nécessaires pour le maintien de la cohérence ainsi qu'une comparaison avec un protocole utilisé dans la plupart des systèmes distribués mobiles existants. Pour finir, dans la section V.7, nous positionnons notre solution par rapport aux approches proposées dans l'état de l'art.

V.1 Historique des mises à jour

Les utilisateurs des ordinateurs mobiles ont, pour la plupart d'entre eux, un ordinateur stationnaire qui leur sert d'archive pour leurs données privées et/ou partagées. En effet, les utilisateurs en situation de mobilité utilisent des ordinateurs de petite taille et de capacité de stockage réduite (p.ex., PDA). Par conséquent, ils ne copient sur leurs ordinateurs mobiles que les données dont ils auront besoin à court

terme. Ces données sont accédées et modifiées au cours de la mobilité de l'utilisateur. Puis, elles sont synchronisées avec leur copie de référence stockée sur les machines stationnaires. Superposons cette situation avec notre notion de groupes mobiles et de domaines de sécurité.

Les données partagées par des entités appartenant à un même domaine de sécurité ont des copies de référence stockées sur des machines stationnaires et fiables. Les utilisateurs autorisés à accéder au domaine de sécurité manipulent des réplicas de ces données au niveau d'un ou de plusieurs groupes mobiles. Nous nous appuyons sur un historique des mises à jour pour synchroniser les divers réplicas d'une donnée tant au sein d'un groupe qu'avec la copie de référence. Ainsi, l'historique des mises à jour doit indiquer : *quand*, *par qui* et *comment* le réplica a été modifié.

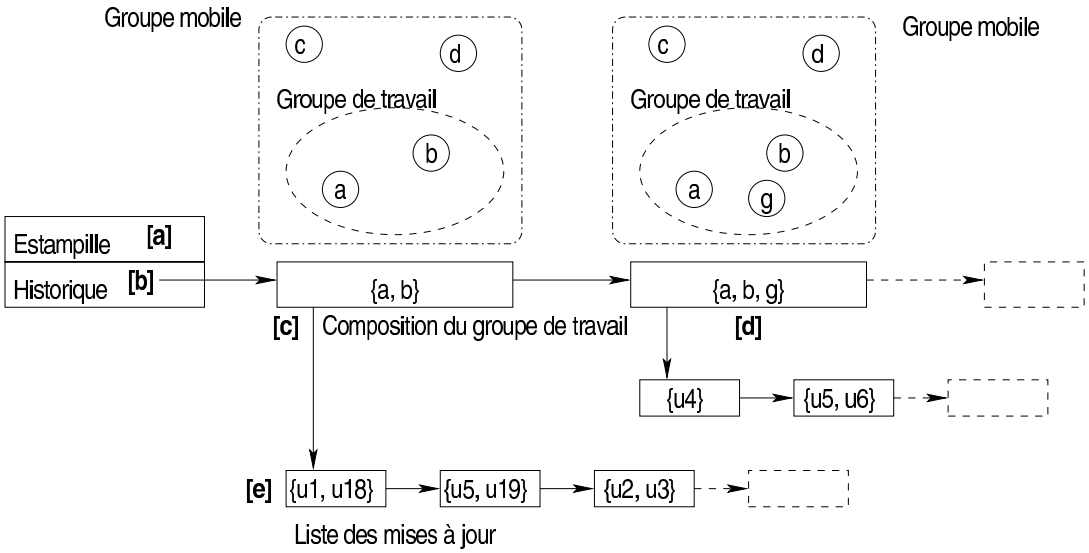

Figure V.1 – CCL attaché à chaque donnée

Nous définissons l'historique des mises à jour associé à un réplica par une structure de données nommée *CCL* (Coherency Control List). Un exemple de CCL est donné dans la figure V.1. Chaque CCL contient deux éléments : (i) une estampille (Fig. V.1-[a]) qui indique avec quelle version de la copie de référence la dernière synchronisation a été effectuée (*quand*) et (ii) un historique (*log*) (Fig. V.1-[b]) qui indique *par qui* et *comment* le réplica a été modifié. L'estampille permet d'établir un ordre chronologique entre deux réplicas par rapport à leur synchronisation avec leur copie de référence, afin de guider le processus de détection et de résolution des conflits. L'his-

torique se présente sous la forme d'une liste où chaque élément indique l'ensemble des identificateurs des entités ayant un réplica local de la donnée modifiée et par conséquent, concernées par les mises à jour (*par qui*). Nous définissons l'ensemble des entités ayant un réplica local de la donnée comme étant un *groupe de travail*, sous ensemble du groupe mobile, associé à la donnée. A chaque *groupe de travail* (Fig. V.1-[c]), nous associons la liste des modifications vues par ce dernier (Fig. V.1-[e]) (*comment*). Si la composition du *groupe de travail* associé à la donnée est modifiée, par l'ajout ou la suppression d'une entité, un nouvel élément est ajouté à l'historique dès qu'une mise à jour se produit, cet élément contient la nouvelle composition du *groupe de travail* associé à la donnée ainsi que les mises à jour vues par celui-ci (Fig. V.1-[d]). Si une entité se déconnecte du groupe, en constituant un singleton, un nouvel élément est ajouté à l'historique de ces CCLs locaux (au moment des mises à jour des données correspondantes) qui contient uniquement son identificateur.

V.2 Gestion hybride de la cohérence

Notre approche à la gestion de la cohérence se distingue en combinant la gestion forte avec la gestion faible de la cohérence selon la connectivité du réseau et les besoins des applications (collaboration synchrone ou asynchrone). Les entités mobiles appartenant à un groupe partagent leurs ressources (incluant des données) pour réaliser une collaboration synchrone. Par conséquent, les données répliquées dans un groupe bénéficient d'une gestion forte de la cohérence. Cette gestion est basée sur le protocole de l'écrivain unique décrit dans la section V.2.1, qui a été optimisé et adapté de sorte à minimiser les communications *via* le réseau sans fil. Par ailleurs, les données répliquées dans des groupes disjoints peuvent être manipulées de manière indépendante et concurrente, grâce à la gestion optimiste de la cohérence au niveau d'un domaine de sécurité décrite dans la section V.2.2.

V.2.1 Gestion de la cohérence forte

Suivant la définition d'un groupe mobile, donnée dans la section IV.1.1, les entités qui y appartiennent bénéficient d'une connectivité telle que le maintien d'une cohérence forte devient possible. La cohérence des réplicas de données au sein d'un groupe mobile est gérée selon le protocole de l'écrivain unique. Toutes les opérations

d'écriture se déroulent en exclusion mutuelle, tandis que les opérations de lectures sont partagées. Cependant, les données locales peuvent être manipulées librement dans des groupes disjoints. Ainsi, les entités qui le souhaitent peuvent se constituer en singleton.

Suivant le protocole de l'écrivain unique, une seule entité à la fois est autorisée à écrire sur un réplica de donnée partagée. Ces mises à jour 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 jour décrite dans la section V.3. Cela nous permet de minimiser les communications, les calculs et par la même occasion l'énergie consommée pour le maintien de la cohérence. 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 attribue le droit d'écriture à l'entité qui le possède. Nous associons à chacun des réplicas d'un groupe, l'un des quatre états définis ci-dessous, en fonction des droits de lecture/écriture que l'entité qui le stocke possède :

- L'état Read signifie que la donnée est verrouillée en lecture et que ce réplica est à jour ; par conséquent, l'entité a le droit de lecture sur son réplica local.
- L'état UpdateRead signifie que la donnée est verrouillée en lecture mais ce réplica n'est pas à jour du fait de la propagation paresseuse des mises à jour ; par conséquent, l'entité doit mettre à jour son réplica local avant de procéder à la lecture.
- L'état ReadWrite signifie que la donnée est verrouillée en écriture et que cette entité possède le jeton associé ; par conséquent, elle a le droit de lecture et d'écriture sur son réplica local.
- L'état Invalid signifie que la donnée est verrouillée en écriture mais que cette entité ne possède pas le jeton associé ; par conséquent, elle n'a pas de droit d'accès à son réplica local. Toutefois, l'utilisateur peut forcer l'accès en lecture, tout en sachant que ce n'est pas la dernière version de la donnée.

Suivant notre protocole, un réplica est considéré à jour par rapport aux réplicas d'un groupe si et seulement si, il est dans l'état ReadWrite ou Read. Par ailleurs, si un réplica est dans l'état ReadWrite, alors tous les autres réplicas de la même donnée dans le groupe sont dans l'état Invalid. Si, en revanche, un réplica est dans l'état Read, tous les autres réplicas de la même donnée dans le groupe sont soit dans l'état Read, soit dans l'état UpdateRead suivant qu'ils sont à jour par rapport aux réplicas du groupe.

Ces états changent en fonction des accès effectués sur les réplicas. Ils sont initialement attribués par le *leader* du groupe lors de l'étape de l'échange des méta-données. Ainsi, lorsque plusieurs entités mobiles se constituent en groupe, l'entité élue *leader* attribue l'état UpdateRead à tous les réplicas partagés dans le groupe. Par ailleurs, si un singleton se joint à un groupe, l'état de ses réplicas locaux est déterminé en fonction de l'état des réplicas de la même donnée existants dans ce groupe. Soient i un singleton qui s'est joint à un groupe et r_i un réplica local, le *leader* du groupe attribue à r_i l'état Invalid si un réplica de la même donnée est accédé en écriture dans le groupe, c'est à dire, qu'il existe au moins un réplica de la même donnée dans l'état Invalid ou s'il existe un réplica dans l'état ReadWrite. En revanche, il attribue à r_i l'état UpdateRead si des réplicas de la même donnée sont accédés en lecture dans le groupe, c'est à dire, qu'il existe au moins un réplica de la même donnée dans l'état Read. Par ailleurs, si aucun réplica de la même donnée n'existe dans le groupe, le *leader* attribue à r_i l'état Read. La fusion de deux groupes est cependant équivalente à la création d'un nouveau groupe. Par conséquent, l'état de tous les réplicas partagés dans les deux groupes devient UpdateRead, tandis que l'état des réplicas partagés dans un seul des deux groupes reste inchangé suite à cette fusion.

V.2.2 Gestion de la cohérence faible

La cohérence faible, mise en avant dans l'état de l'art des systèmes distribués dédiés aux environnements mobiles, est considérée comme étant la plus adaptée aux environnements mobiles où la connectivité est intermittente. En effet, elle permet aux utilisateurs mobiles de manipuler leurs réplicas locaux de manière indépendante et concurrente. Toutefois, cette approche de la gestion de la cohérence des données n'offre pas de support aux applications de collaboration synchrone, qui nécessitent le maintien d'une cohérence forte entre différents réplicas d'une donnée partagée.

Dans notre approche, au niveau d'un domaine de sécurité, plusieurs réplicas d'une même donnée peuvent être partagés entre différents groupes. Sachant qu'au sein d'un groupe, la cohérence forte des réplicas est assurée, les réplicas se trouvant dans des groupes différents peuvent être manipulés de manière indépendante et bénéficient ainsi d'une cohérence faible. Si un utilisateur appartenant à un groupe veut bénéficier d'une gestion faible de la cohérence de ses réplicas locaux, il lui suffit de se constituer en singleton. Ainsi, ses réplicas locaux seront indépendants de ceux du groupe auquel il appartenait. Dans ce cas, cette entité constituée en singleton ne par-

tage plus ses ressources au sein du groupe jusqu'au moment où elle le réintègre.

La gestion faible de la cohérence nécessite la détection et la résolution des conflits qui peuvent survenir lors de la synchronisation des réplicas divergents. Pour ce faire, nous proposons une solution basée sur les historiques des mises à jour décrite dans la section V.4.

V.3 Accès aux données

Dans un domaine de sécurité, plusieurs réplicas d'une même donnée peuvent être partagés dans des groupes disjoints. L'accès à des réplicas appartenant à des groupes disjoints se fait indépendamment au niveau de chaque groupe. En revanche, dans un groupe mobile, les accès aux réplicas partagés nécessitent un contrôle de cohérence afin de s'assurer que le réplica allant être accédé est cohérent avec le reste des réplicas du groupe. Nous effectuons ces contrôles de cohérence au moment de l'accès effectif au réplica, ce qui nous permet de ne pas maintenir la cohérence des réplicas de données qui ne sont pas accédées, et ainsi de minimiser la consommation de ressources. Ces contrôles de cohérence sont réalisés en comparant les historiques des mises à jour attachés à chaque réplica de donnée.

L'accès aux données partagées dans un groupe devient possible à l'issue de l'étape de l'initialisation du groupe décrite dans le chapitre IV. A la fin de cette étape, chaque entité possède une vue globale des ressources partagées dans le groupe grâce aux méta-données envoyées par le *leader* du groupe. Ces méta-données contiennent, entre autre, pour chaque donnée répliquée, l'état associé à chacun des réplicas de la donnée et l'identité de l'entité (si une telle entité existe) qui possède le jeton nécessaire pour la gestion des accès en lecture/écriture. La nature des contrôles de cohérence effectués au moment de l'accès effectif à un réplica dépend de l'état de celui-ci. Si le réplica est dans l'état Read ou ReadWrite alors aucun contrôle de cohérence n'est nécessaire car ce réplica est à jour. Si, en revanche, le réplica est dans l'état Invalid ou UpdateRead alors un contrôle de cohérence est nécessaire et il s'effectue en comparant le CCL associé à ce réplica avec celui d'un réplica à jour (dans l'état Read ou ReadWrite) dans le groupe. A cet effet, notre gestion de la disponibilité de données accroît la probabilité de l'existence d'au moins un réplica à jour au sein du groupe (voir chapitre VI). Suivant la nature de l'accès, la donnée est verrouillée en lecture ou en écriture. La gestion des verrous nécessite la création d'un jeton. Un jeton est

créé, s'il n'en existe pas (déterminé d'après les méta-données), lors de la première demande d'accès à une donnée partagée dans le groupe par l'entité qui demande l'accès. La possession du jeton est indispensable pour les accès en écriture, tandis que les accès en lecture peuvent s'effectuer sur des entités qui ne le possèdent pas. La perte du jeton due, par exemple, à la déconnexion de son possesseur, est détectée et corrigée au moment de la mise à jour du groupe et plus précisément lors de l'étape de l'échange des méta-données (voir section IV.4). Au moment où la perte du jeton est détectée, un nouveau jeton est créé par l'entité qui demandera en premier l'accès à cette donnée. Le protocole d'accès aux données partagées dans un groupe est détaillé dans la section suivante.

V.3.1 Protocole

Pour l'accès à un réplica de données dans un groupe mobile, l'entité qui demande l'accès doit :

Avoir un réplica local de la donnée Comme nous l'avons mentionné dans le chapitre précédent, ce choix nous permet de réduire les communications sans fil que les accès distants nécessitent. Ainsi, si l'entité ayant demandé l'accès ne possède pas un réplica local, alors elle doit demander une copie à l'entité qui possède un réplica à jour notamment au possesseur du jeton associé. Ce dernier lui envoie une copie du réplica ainsi que celle du CCL associé. L'état du réplica créé est déterminé en fonction de l'état des réplicas du groupe. Si la donnée accédée est verrouillée en lecture alors l'état Read est attribué à ce réplica. En revanche, si la donnée accédée est verrouillée en écriture alors l'état Invalid lui est attribué. Si en revanche aucun réplica dans le groupe n'est à jour (le cas d'un premier accès après la création du groupe), l'entité récupère aléatoirement une copie locale (peut être non à jour) en gardant le même état, puis elle procède au contrôle de cohérence.

Contrôler la cohérence de son réplica local Le contrôle de cohérence s'effectue suivant l'état du réplica local. Si le réplica est dans l'état Read ou ReadWrite aucun contrôle de cohérence n'est nécessaire car ce réplica est à jour. Si, en revanche, le réplica est dans l'état Invalid ou UpdateRead alors une synchronisation du réplica local avec un réplica à jour notamment celui du possesseur du jeton est nécessaire. Cette synchronisation permet de détecter d'éventuels conflits et aboutit à une propagation

des mises à jour si nécessaire, elle est décrite en détails dans la section V.4. Toutefois, un cas particulier subsiste et correspond au cas où tous les réplicas du groupe sont dans l'état UpdateRead. C'est notamment ce qui se produit lorsque plusieurs entités mobiles se constitue en groupe (voir section V.2.1). Dans ce cas, l'entité qui demande l'accès se charge de synchroniser son réplica local avec tous les réplicas du groupe, suivant le protocole décrit dans la section V.4.

Poser un verrou Suivant la nature de la demande d'accès, un verrou est posé localement soit en lecture, soit en lecture/écriture. Un protocole d'exclusion mutuelle assure l'attribution et la gestion des verrous en écriture et en lecture. Un verrou posé en lecture permet un accès partagé. L'état du réplica local devient alors Read. Ceci engendre également la modification de l'état des autres réplicas du groupe comme suit : si un réplica était dans l'état ReadWrite il devient dans l'état Read, s'il était dans l'état Invalid il devient dans l'état UpdateRead et s'il était dans l'état Read ou UpdateRead son état ne change pas. Par ailleurs, un verrou ne peut être posé en écriture qu'après avoir récupéré le jeton. Ce verrou assure un accès exclusif à la donnée à l'entité qui en a demandé l'accès en écriture. L'état du réplica accédé en écriture devient ReadWrite, tandis que l'état des autres réplicas du groupe est changé comme suit : si un réplica était dans l'état ReadWrite, Read ou UpdateRead son état devient Invalid, en revanche s'il était dans l'état Invalid son état ne change pas. La nouvelle identité du possesseur du jeton est propagé lors de la propagation des mises à jour.

L'accès effectif en lecture ou en écriture au réplica local peut à présent être effectué. Dans le cas d'un accès en écriture, le CCL associé au réplica local doit être mis à jour. Ainsi, toutes les nouvelles modifications sont répertoriées dans ce dernier. Ces modifications sont propagées aux entités du groupe au fur et à mesure que ces dernières accèdent à la donnée. Cette propagation paresseuse des mises à jour réduit le nombre de messages échangés pour le maintien de la cohérence ainsi que les calculs générés par le protocole de détection et de gestion des conflits. Enfin, le verrou posé en lecture ou en écriture doit être libéré pour permettre aux autres entités du groupe d'accéder à la donnée, en particulier dans le cas d'un verrou en écriture. La libération du verrou ne modifie pas l'état des réplicas du groupe. Ce n'est que lors de la prochaine demande d'accès que les états sont modifiés en conséquence. Par ailleurs, chaque entité traite les demandes d'accès en attente, si elles existent, dans l'ordre *FIFO*. Cependant, lorsqu'un accès en lecture est traité, tous les autres accès en lecture mis en attente le sont également afin de permettre un accès partagé en lecture.

Ce protocole vérifie la propriété de sûreté (*safety*). En effet, à tout instant il y a au plus un écrivain dans un *groupe de travail* par rapport à une donnée partagée. L'unicité du jeton permet de garantir cette propriété, car seul le possesseur du jeton peut accéder en écriture à la donnée. Dans un *groupe de travail* associé à une donnée, l'identité de l'entité qui possède le jeton (s'il existe) est connue. Cette information est propagée dans les méta-données et lors de la propagation des mises à jour. Un jeton n'est créé que si aucune entité du groupe, d'après les méta-données, ne le possède garantissant ainsi son unicité. La perte du jeton est détectée et résolue au moment de la mise à jour du groupe. Cependant, la propriété de vivacité (*liveness*) exige que toute entité appartenant au groupe qui demande un accès à une donnée partagée dans le groupe doit, au bout d'un temps fini, recevoir l'autorisation d'accès. Pour garantir cette propriété, dans la limite de la connectivité du réseau sous-jacent, nous supposons que les accès aux réplicas sont de durée finie. La propriété de vivacité ne peut pas être garantie face à la déconnexion des entités mobiles. En effet, si une entité qui demande l'accès à un réplica non local se déconnecte du groupe avant que le verrou soit libéré, l'accès ne sera pas possible.

Le tableau V.1 résume tous les cas possibles pour accéder une donnée répliquée sur n entités dans un groupe de p entités mobiles en fonction de l'état de l'entité (i) qui demande l'accès (*Etat*), de la nature de l'accès (*Op.*), ainsi que l'existence du jeton au niveau de cette entité (*Jeton*). Lorsque la valeur de *Local* est égale à vrai, le réplica accédé est local, tandis que lorsque qu'elle est égale à faux, le réplica accédé n'est pas localement stockée. Dans ce tableau les états R, RW, UR et I correspondent respectivement aux états Read, ReadWrite, UpdateRead et Invalid. Les opérations (*Op.*) R, W correspondent respectivement aux opérations de lecture et d'écriture. Lorsque la valeur de *Msg* est égale à *n-1*, cela signifie que le message est envoyé à toutes les entités qui ont un réplica local de la donnée, excepté l'expéditeur. Lorsque la valeur de *Msg* est égale à *n-2*, cela signifie que le message est envoyé à toutes les entités qui ont un réplica local de la donnée, excepté l'expéditeur et l'écrivain précédent. Enfin, lorsque la valeur de *Msg* est égale à *p-1*, cela signifie que le message est envoyé à toutes les entités du groupe, excepté l'expéditeur.

V.3.2 Impact de la propagation paresseuse des mises à jour

L'impact de la propagation paresseuse des mises à jour est particulièrement manifeste lors de certaines séquences d'accès. Ces séquences permettent d'illustrer la

Local	*Jeton*	*Etat*	*Op.*	*Msg.*	*En réponse*	*Résultats et actions*
vrai	vrai	R	R	aucun	rien	lecture
vrai	vrai	R	W	n-1	rien	état groupe sauf i ← I et écriture
vrai	vrai	RW	R	n-1	rien	état i ← R, état groupe sauf i ← UR et lect.
vrai	vrai	RW	W	aucun	rien	écriture
vrai	faux	R	R	aucun	rien	lecture
vrai	faux	R	W	écrivain et n-2	jeton rien	état i ← RW, état groupe sauf i ← I et écriture
vrai	faux	I	R	écrivain	CCL et, MAJs	mise à jour du CCL et du fichier, lecture
vrai	faux	I	W	écrivain	Jeton, CCL et MAJs	mise à jour du CCL et du fichier, écriture
vrai	faux	UR	R	écrivain	CCL et, MAJs	mise à jour du CCL et du fichier, état i ← R, lecture
vrai	faux	UR	W	écrivain	jeton, CCL et MAJs	mise à jour du CCL et du fichier, état i ← RW, état groupe sauf i ← I et écriture
faux	faux	-	R	écrivain	fichier et CCL	copier le fichier et le CCL, état i ← R, lecture
faux	faux	-	W	écrivain	jeton, fichier et CCL	copier le fichier et le CCL, état i ← RW, état groupe sauf i ← I et écriture

Table V.1 – Différents scénarios possibles pour l'accès aux données

réduction du nombre d'entités impliquées dans les synchronisations, lors des différentes demandes d'accès. Dans la suite de cette section, nous considérons un réplica r_i stocké sur l'entité i qui appartient à un groupe.

Lectures successives si i demande un accès en lecture sur r_i alors que la donnée est verrouillée en lecture au sein du groupe, deux cas sont possibles : soit r_i est dans l'état Read et donc l'accès en lecture s'effectue sans contrôle de cohérence, soit r_i est dans l'état UpdateRead, et l'accès en lecture nécessite une synchronisation préalable avec un réplica à jour.

Demande d'écriture après une lecture si i demande un accès en écriture sur r_i, dont l'état est Read, l'attribution d'un verrou en écriture ne génère pas *a priori* de mises à jour. Il suffit à l'entité i d'obtenir le jeton et faire passer l'état de tous les autres réplicas du groupe à l'état Invalid, sauf pour r_i, dont l'état devient ReadWrite.

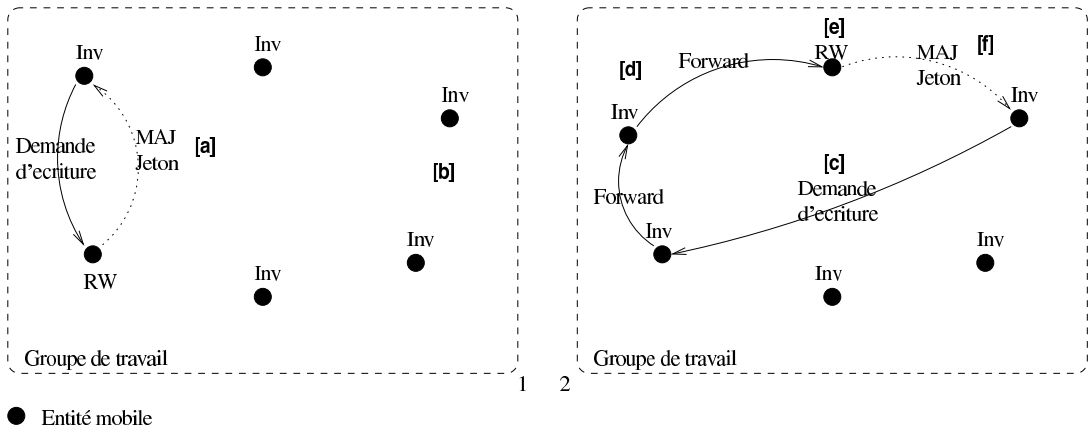

Figure V.2 – Séquence d'écriture successives.

Ecritures successives si i demande un accès en écriture sur r_i tandis qu'il est dans l'état Invalid. La première étape consiste à synchroniser le réplica local r_i avec celui de l'écrivain précédent connu noté j. Cette synchronisation génère une propagation des mises à jour entre ces deux entités sans impliquer les autres entités du *groupe de travail* (Fig. V.2-[a]) qui restent dans l'état Invalid (Fig. V.2-[b]). En plus des mises à jour, le nouvel écrivain (l'entité i) récupère le jeton.

D'après notre protocole d'accès aux données partagées, les demandes d'accès, en particulier les demandes de synchronisation sont toujours adressées au possesseur du jeton car il détient un réplica à jour. Ainsi, si une entité k demande un accès en écriture à son réplica local r_k, elle transmet sa demande à l'entité qui avait effectué le premier accès en écriture au tout début de cette séquence d'écritures successives, à savoir l'entité j (Fig. V.2-[c]). Cette dernière, n'étant plus le dernier écrivain, transmet à son tour la demande d'accès à la seconde entité ayant demandé un accès en écriture, à savoir l'entité i (Fig. V.2-[d]), et ainsi de suite jusqu'à arriver au dernier écrivain de la séquence (le possesseur du jeton) (Fig. V.2-[e]). Ce dernier renvoie le jeton ainsi que les mises à jour pour synchroniser le réplica local de l'entité demandant l'accès (Fig. V.2-[f]).

Cette cascade de messages générée par la transmission de la demande d'accès d'un ancien écrivain à un autre peut devenir un inconvénient si le nombre d'écritures successives est grand. On peut remédier à cet inconvénient en synchronisant toutes les entités du *groupe de travail* après chaque nouvelle demande d'écriture, ce qui est notamment le cas dans le cadre de la gestion faible de la cohérence. Certes, le dernier écrivain sera connu à chaque nouvelle demande d'accès, ce qui évitera les transmissions en cascade, mais le coût de chaque nouvelle attribution de verrou sera beaucoup plus important. Nous avons évalué et comparé les deux approches dans la section V.6 où nous avons démontré que la propagation paresseuse des mises à jour permet de réduire le nombre de messages échangés pour le maintien de la cohérence des réplicas au sein d'un groupe et par conséquent, permet de réduire la consommation d'énergie.

Demande de lecture après une écriture si i demande un accès en lecture sur r_i tandis que la donnée est verrouillée en écriture au sein du groupe alors une synchronisation de r_i avec le réplica de l'écrivain (Fig. V.3-[a]) est toujours nécessaire. Cet accès en lecture génère par ailleurs le changement des états de tous les réplicas du groupe comme suit : l'entité i, après avoir synchronisé son réplica local, ainsi que l'ancien écrivain deviennent dans l'état Read (Fig. V.3-[b]) car leurs réplicas locaux sont à jour et les accès en lecture à ces derniers peuvent être effectués sans mises à jour supplémentaires. Cependant, les autres réplicas qui n'ont pas encore été synchronisés deviennent dans l'état UpdateRead (Fig. V.3-[c]). En effet, les mises à jour destinées à l'entité i, qui demande l'accès en lecture, sont calculées en comparant le CCL associé au réplica local de cette dernière avec celui associé au réplica de l'écrivain. Ces mises

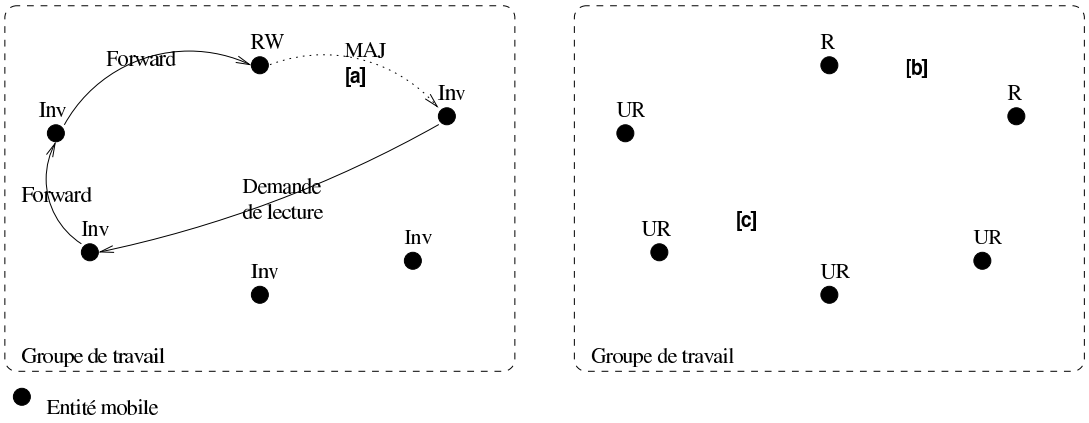

Figure V.3 – Demande de lecture après une écriture.

à jour, bien que réceptionnées par toutes les entités appartenant au groupe, ne sont pas toujours suffisantes pour synchroniser les réplicas locaux de ces dernières. Ceci résulte en partie de la propagation paresseuse des mises à jour. Pour cette raison, lors d'une demande d'accès en lecture après une écriture, les réplicas locaux des entités appartenant au *groupe de travail* (sauf le demandeur d'accès et l'écrivain) passent à l'état UpdateRead.

V.4 Synchronisation des réplicas

La synchronisation des réplicas permet de détecter les mises à jour divergentes puis de résoudre les conflits générés par ces dernières. Seules les mises à jour effectuées dans des groupes disjoints peuvent être divergentes. En effet, au sein d'un groupe la gestion de la cohérence forte nous prémunit de ce type de mise à jour. De plus, toutes les mises à jour effectuées sur un réplica dans un groupe sont propagées, certes de manière paresseuse mais dans le même ordre, à toutes les entités ayant un réplica local de la même donnée. Par ailleurs, toutes les mises à jour sont répertoriées dans les CCLs associés aux réplicas. La détection des conflits consiste à comparer ces CCLs. Elle est réalisée au moment de l'accès effectif à un réplica dans un groupe. La section V.4.1 décrit la détection des conflits en détail. Si la synchronisation des réplicas détecte des mises à jour conflictuelles, la résolution des conflits est traitée comme décrit dans la section V.4.2.

V.4.1 Détection des conflits

Notre protocole de détection des conflits est basé sur la comparaison des CCLs associés à chaque réplica. Au niveau d'un groupe, cette comparaison se fait avec le CCL associé à un réplica à jour (c.-à-d., dont l'état est Read ou ReadWrite), s'il en exite. La comparaison s'effectue au moment de l'accès effectif au réplica, à l'issue de laquelle les mises à jour nécessaires pour la synchronisation des deux réplicas sont déterminées ou les mises à jour conflictuelles sont désignées. Notons que ce protocole de détection des conflits est indépendant du type des données manipulées. Le CCL associé à un réplica stocke la liste des mises à jour dans l'ordre dans lequel elles ont été reçues par ce dernier. Pour détecter des conflits, nous définissons la notion de *préfixe* entre deux CCLs associés à deux réplicas de la même donnée qui nous permet de les comparer. Nous considérons que le CCL ccl_1 est le *préfixe* de ccl_2 si et seulement si :

- les valeurs des deux estampilles sont égales ;
- la liste des *groupes de travail* associée à la donnée dans ccl_1 est préfixe de la liste des *groupes de travail* dans ccl_2 ; et
- chaque liste de mises à jour correspondant à un *groupe de travail* dans ccl_1 est identique à celle qui correspond au même *groupe de travail* dans ccl_2. Cette condition doit être vérifiée pour chaque liste de mises à jour sauf, éventuellement, pour la liste qui correspond au dernier élément de ccl_1. Dans ce cas, les modifications de ccl_1 doivent être un sous ensemble de celles de ccl_2.

La relation *préfixe* détermine quel réplica doit être mis à jour. Si ccl_1 est préfixe de ccl_2, alors le réplica correspondant à ccl_1 doit être mis à jour. La mise à jour s'effectue en propageant les modifications manquantes déterminées selon ccl_2. Si les deux CCLs ne sont pas préfixe l'un de l'autre, alors les deux réplicas correspondants divergent et nécessitent le recours au protocole de résolution des conflits décrit dans la section V.4.2.

A titre d'exemple, considérons les deux CCLs donnés dans la figure V.4. Le CCL ccl_1 est préfixe du CCL ccl_2 car : (i) leurs estampilles sont égales (égale à 1 dans l'exemple), (ii) la liste des *groupes de travail* dans ccl_1 est préfixe de celle dans ccl_2 et (iii) chaque liste de mises à jour correspondant à un *groupe de travail* dans ccl_1 est identique à celle qui correspond au même *groupe de travail* dans ccl_2, sauf la liste qui correspond au dernier élément de ccl_1 qui est un sous ensemble de son équivalente dans ccl_2 ($\{1\} \in \{1, 5\}$). Dans ce cas il suffit de propager les mises à jour $\{5, 6\}$, qui

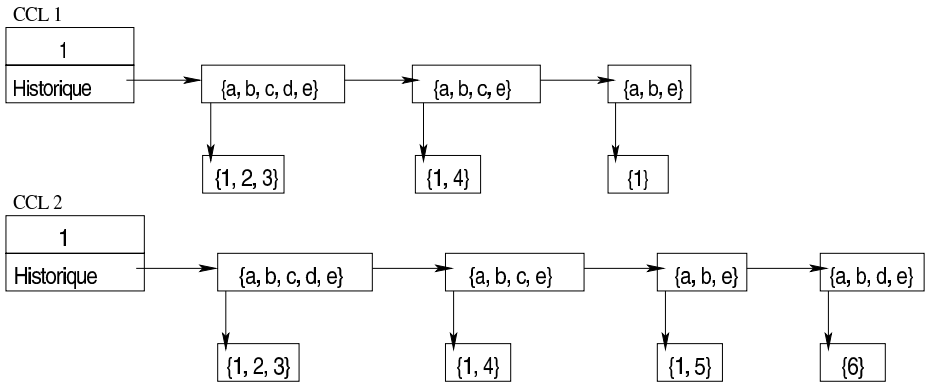

Figure V.4 – Exemple de CCLs. ccl_1 est préfixe de ccl_2.

existent dans ccl_2 vers le réplica associé au ccl_1.

Dans la section suivante, nous proposons un protocole optimisé de détection des conflits qui prend en compte l'impact de la propagation paresseuse des mises à jour.

V.4.1.1 Impact de la propagation paresseuse des mises à jour

L'une des conséquences de la propagation paresseuse des mises à jour dans un groupe (voir Section V.3) est que les entités appartenant au groupe peuvent avoir des vues différentes des mises à jour liées à leurs réplicas locaux. En effet, les mises à jour ne sont propagées aux réplicas qu'au moment de l'accès effectif. A titre d'exemple, considérons un groupe où les entités a, b, c, d et e partagent une donnée répliquée r telle que r_i correspond au réplica stocké sur l'entité i et $i \in \{a,\ b,\ c,\ d,\ e\}$. Les CCLs associés aux entités a et b donnés dans la figure V.5 résultent du scénario suivant :

- L'entité a fait une demande d'écriture sur r_a. La mise à jour $\{1\}$ est enregistrée dans ccl_a.
- L'entité c fait une demande de lecture sur r_c. Elle reçoit la mise à jour $\{1\}$ de l'entité a et l'enregistre dans ccl_c, de même pour les entités b, d et e. Ainsi, l'entité b enregistre la mise à jour $\{1\}$ dans ccl_b.
- L'entité a fait une demande d'écriture sur r_a. Les mises à jour $\{2,\ 3\}$ sont enregistrées dans ccl_a.
- L'entité d quitte le groupe. L'entité c fait une demande d'écriture sur r_c. Elle reçoit les mises à jour $\{2,\ 3\}$ de l'entité a les enregistre dans ccl_c et ajoute un nouvel élément à son CCL car la composition du groupe a changé. Puis, elle enregistre la mise à jour $\{1\}$.

- L'entité a fait une demande de lecture sur r_a. Elle reçoit la mise à jour $\{1\}$ de l'entité c l'enregistre dans un nouvel élément du ccl_a, de même pour les entités b et e. Ainsi, l'entité b enregistre la mise à jour $\{1\}$ dans un nouvel élément du ccl_b.
- L'entité a fait une demande d'écriture sur r_a. La mise à jour $\{4\}$ est enregistrée dans ccl_a.
- L'entité c quitte le groupe. L'entité d rejoint le groupe et fait une demande d'écriture sur r_d. Elle reçoit les mises à jour nécessaires de l'entité a. Puis, elle enregistre la mise à jour $\{1\}$ dans ccl_d.
- L'entité a fait une demande de lecture sur r_a. Elle reçoit la mise à jour $\{1\}$ de l'entité d l'enregistre dans un nouvel élément du ccl_a, de même pour les entités b et e. Ainsi, l'entité b enregistre la mise à jour $\{1\}$ dans un nouvel élément du ccl_b.
- L'entité a fait une demande d'écriture sur r_a. La mise à jour $\{5\}$ est enregistrée dans ccl_a.
- L'entité d quitte à nouveau le groupe. L'entité a fait une demande d'écriture sur r_a. La mise à jour $\{6\}$ est enregistrée dans un nouvel élément du ccl_a.
- L'entité e quitte le groupe. L'entité a fait une demande d'écriture sur r_a. La mise à jour $\{7\}$ est enregistrée dans un nouvel élément du ccl_a.

Ainsi, l'entité b a reçu uniquement la mise à jour $\{1\}$. Bien que ccl_a ne soit pas un *préfixe* du ccl_b, ni inversement, les réplicas correspondants ne sont pas divergents et peuvent être synchronisés par une simple propagation des mises à jour car toutes les mises à jour ont été effectuées dans un groupe qui contient a et b. En effet, les conflits peuvent survenir sur des mises à jour effectuées dans des groupes disjoints et plus particulièrement, si les deux réplicas ont été mis à jour dans des groupes disjoints. Toutefois, si uniquement un des réplicas a été mis à jour dans un groupe qui ne contient pas l'entité qui stocke le deuxième réplica alors les deux réplicas peuvent être synchronisés par une propagation des mises à jour. Par conséquent, la détection des conflits consiste à déterminer si de telles mises à jour existent en comparant les listes des mises à jour telles qu'elles sont vues par les deux entités.

Le protocole de détection des conflits est donné sous sa forme algorithmique dans la figure V.6 où a et b désignent deux entités mobiles stockant chacune un réplica de la même donnée. La première étape pour la détection des conflits entre le réplica stocké sur l'entité a et celui stocké sur l'entité b consiste à établir les listes de mises à jour vues par l'entité a, puis, celles vues par l'entité b. Les listes de mises à jour vues

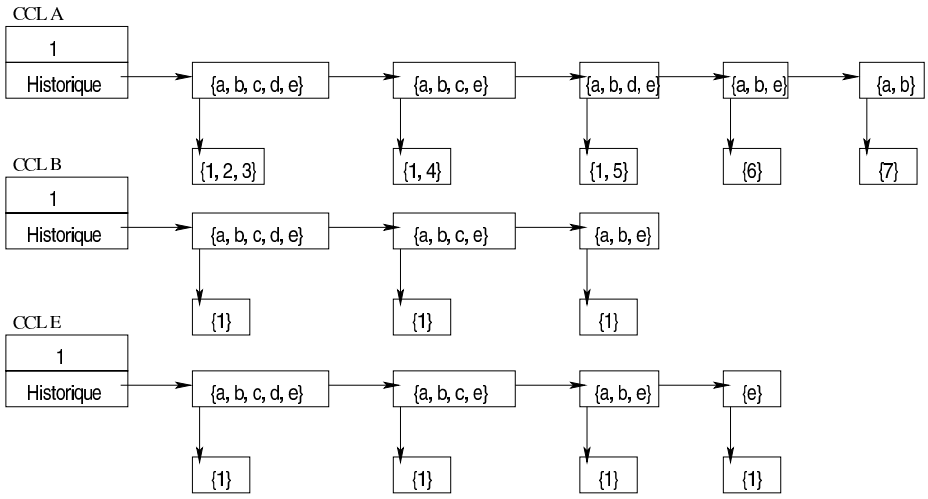

Figure V.5 – Exemple de CCLs

par l'entité a correspondent à des *groupes de travail* qui incluent a et b, puis à ceux qui n'incluent pas l'entité b. De la même manière, les listes de mises à jour vues par l'entité b correspondent à des *groupes de travail* qui incluent a et b, puis à ceux qui n'incluent pas l'entité a. On utilise les notations suivantes : $MAJ_b(a)$ correspond à la liste de mises à jour vue par l'entité a et associée aux *groupes de travail* qui contiennent les entités a et b ensembles ; $MAJ_{\neg b}(a)$ correspond à la liste des mises à jour vue par l'entité a et associée aux *groupes de travail* qui ne contiennent pas l'entité b.

Une fois les listes de mises à jour déterminées, le protocole de détection des conflits considère la liste $MAJ_{\neg b}(a)$ et la liste $MAJ_{\neg a}(b)$. Ces listes correspondent aux mises à jour effectuées dans des groupes qui ne contiennent pas l'une ou l'autre des deux entités et donc ces mises à jour sont potentiellement divergentes. Ainsi, si ces deux listes ne sont pas vides (Fig. V.6-[a]), un conflit est signalé sur les mises à jour correspondantes à l'union de ces deux listes (Fig. V.6-[b]). Dans le cas où l'une des deux listes est vide (Fig. V.6-[c]), la synchronisation des deux réplicas est possible, elle consiste à propager les mises à jour vers le réplica qui n'est pas à jour. Le sens de la propagation des mises à jour est déterminé en comparant $MAJ_b(a)$ et $MAJ_a(b)$.

Si $MAJ_b(a) \subset MAJ_a(b)$ alors le réplica stocké sur l'entité a doit être mis à jour (Fig. V.6-[d]). Les mises à jour à propager sont déterminées en calculant la différence entre $MAJ_b(a)$ et $MAJ_a(b)$ (Fig. V.6-[g]). Les mises à jour déterminées par la liste $MAJ_{\neg a}(b)$, dans le cas où elle n'est pas vide, sont également propagées vers le réplica stocké sur l'entité a (Fig. V.6-[e]). Si, en revanche, $MAJ_{\neg b}(a) \neq \phi$ alors le réplica

stocké sur l'entité b doit être également mis à jour. Elles sont déterminées par la liste $MAJ_{\neg b}(a)$ (Fig. V.6-[f]).

Si $MAJ_a(b) \subset MAJ_b(a)$ alors le réplica stocké sur l'entité b doit être mis à jour (Fig. V.6-[h]). Dans ce cas, les mises à jour à propager sont déterminées en calculant la différence entre $MAJ_b(a)$ et $MAJ_a(b)$ (Fig. V.6-[k]). Les mises à jour déterminées par la liste $MAJ_{\neg b}(a)$, dans le cas où elle n'est pas vide, sont également propagées vers le réplica stocké sur l'entité b (Fig. V.6-[i]). Cependant, si $MAJ_{\neg a}(b) \neq \phi$ alors le réplica stocké sur l'entité a reçoit également des mises à jour. Elles sont déterminées par la liste $MAJ_{\neg a}(b)$ (Fig. V.6-[j]).

Dans le cas où $MAJ_b(a)$ et $MAJ_a(b)$ sont égaux (Fig. V.6-[l]), les mises à jour sont propagées vers le réplica stocké sur l'entité b si $MAJ_{\neg b}(a) \neq \phi$ (Fig. V.6-[m]) et vers le réplica stocké sur l'entité a si $MAJ_{\neg a}(b) \neq \phi$ (Fig. V.6-[n]). Par ailleurs, si $MAJ_{\neg a}(b)$ et $MAJ_{\neg b}(a)$ sont vide, il n'est pas nécessaire de propager des mises à jour car les deux réplicas sont identiques (Fig. V.6-[o]).

A titre d'exemple, considérons les trois CCLs donnés dans la figure V.5, les listes des mises à jour vues par les trois entités a, b, et e sont les suivantes :

$MAJ_b(a) = \{1, 2, 3, 4, 5, 6, 7\}$, $MAJ_{\neg b}(a) = \phi$,
$MAJ_e(a) = \{1, 2, 3, 4, 5, 6\}$, $MAJ_{\neg e}(a) = \{7\}$,
$MAJ_a(b) = \{1\}$, $MAJ_{\neg a}(b) = \phi$,
$MAJ_e(b) = \{1\}$, $MAJ_{\neg e}(b) = \phi$,
$MAJ_a(e) = \{1\}$, $MAJ_{\neg a}(e) = \{1\}$,
$MAJ_b(e) = \{1\}$, $MAJ_{\neg b}(e) = \{1\}$,

Si l'on considère le réplica stocké sur l'entité a et celui stocké sur l'entité b on constate qu'une simple propagation des mises à jour est nécessaire. En effet, $MAJ_{\neg b}(a) = \phi$ et $MAJ_{\neg a}(b) = \phi$, reste à déterminer quel réplica doit être mis à jour. Nous constatons que $MAJ_a(b) = \{1\}$ est un sous ensemble de $MAJ_b(a) = \{1, 2, 3, 4, 5, 6\}$, donc le réplica stocké sur l'entité b doit être mis à jour. Ces mises à jour sont obtenues par la différence $MAJ_b(a) - MAJ_a(b)$.

En revanche, si l'on procède à une détection des conflits entre le réplica stocké sur l'entité a et celui stocké sur l'entité e on constate une divergence. En effet, $MAJ_{\neg a}(e) \neq \phi$ et $MAJ_{\neg e}(a) \neq \phi$ donc la condition (Fig. V.6-[a]) est vérifiée, ce qui signifie que les deux réplicas sont divergents par rapport à la mise à jour $\{1, 7\}$ ($MAJ_{\neg e}(a) \cup MAJ_{\neg a}(e) = \{1, 7\}$). En analysant le ccl_e, nous constatons, en effet, que l'entité e a mis à jour son réplica local en étant un singleton, de même que le réplica stocké sur

[a] **si** (($MAJ_{\neg b}(a) \neq \phi$) et ($MAJ_{\neg a}(b) \neq \phi$))
[b] Conflit sur $MAJ_{\neg b}(a) \cup MAJ_{\neg a}(b)$
[c] **sinon**
[d] **si** ($MAJ_b(a) \subset MAJ_a(b)$)
si ($MAJ_{\neg a}(b) \neq \phi$)
[e] Propager $(MAJ_a(b) - MAJ_b(a)) \cup MAJ_{\neg a}(b)$ vers l'entité a
sinon
si ($MAJ_{\neg b}(a) \neq \phi$)
[f] Propager $(MAJ_{\neg b}(a))$ vers l'entité b
sinon
[g] Propager $(MAJ_a(b) - MAJ_b(a))$ vers l'entité a
finsi
finsi
sinon
[h] **si** ($MAJ_a(b) \subset MAJ_b(a)$)
si ($MAJ_{\neg b}(a) \neq \phi$)
[i] Propager $(MAJ_b(a) - MAJ_a(b)) \cup MAJ_{\neg b}(a)$ vers l'entité b
sinon
si ($MAJ_{\neg a}(b) \neq \phi$)
[j] Propager $(MAJ_{\neg a}(b))$ vers l'entité a
sinon
[k] Propager $(MAJ_b(a) - MAJ_a(b))$ vers l'entité b
finsi
finsi
[l] **sinon**/* $MAJ_b(a) = MAJ_a(b)$ */
si ($MAJ_{\neg b}(a) \neq \phi$)
[m] Propager $(MAJ_{\neg b}(a))$ vers l'entité b
sinon
si ($MAJ_{\neg a}(b) \neq \phi$)
[n] Propager $(MAJ_{\neg a}(b))$ vers l'entité a
sinon
[o] Les deux réplicas sont identiques
finsi
finsi
finsi
finsi
finsi

Figure V.6 – Protocole de détection des conflits

l'entité a a été mis à jour dans un groupe qui ne contient pas l'entité e.

V.4.2 Résolution des conflits

L'automatisation de la résolution des conflits résultant de la gestion de la cohérence faible des données répliquées est une préoccupation récurrente dans la conception des systèmes distribués et particulièrement ceux dédiés aux environnements mobiles. En effet, la résolution totalement automatique des conflits est impossible et nécessite l'intervention de l'utilisateur car elle dépend de la sémantique des données ainsi que de l'intention des utilisateurs.

Plusieurs tentatives pour limiter et guider l'intervention des utilisateurs ont été proposées dans la littérature. Elles consistent à adapter la résolution des conflits selon les applications et le type des données manipulées. C'est notamment le cas de la solution proposée dans le système répliqué Bayou [121, 124, 123, 102, 122] où chaque requête de mise à jour est associée à un ensemble de dépendances (*dependency set*) composé de requêtes de contrôle spécifiques à l'application, spécifiant des contraintes sur les réponses attendues et une procédure de résolution des conflits (*mergeproc*), qui est appelée lorsqu'un conflit est détecté [31]. Un conflit est détecté lorsque l'exécution d'une requête appartenant à l'ensemble de dépendances, au niveau d'un réplica, ne vérifie pas la contrainte associée. Définir l'ensemble de dépendances, ainsi que la procédure de résolution, sont à la charge des développeurs d'applications. D'autres systèmes tels que Bengal [33] ont adopté des solutions inspirées de celle de Bayou. Lorsqu'un conflit est détecté, Bengal fait appel à une série de procédures de résolution de conflits, qui sont supposées résoudre le conflit automatiquement. De plus, Bengal maintient un ensemble de règles configurables qui permet de déterminer quelle procédure de résolution appelée selon la nature du conflit. Ces règles sont testées une à une jusqu'à ce que l'on trouve celle qui correspond au type du conflit, suite à quoi, la procédure de résolution associée est appelée. Si cette procédure ne résout pas le conflit correctement, une autre règle est choisie et ainsi de suite jusqu'à ce que le conflit soit résolu ou que toutes les règles soient testées. Cette solution génère des appels de procédure de résolution en cascade et provoque des goulots d'étranglement.

Les solutions proposées pour les systèmes de fichiers [70, 72, 71, 46, 110] considèrent des conflits particuliers inhérents à la nature des données manipulées. Ces systèmes proposent une classification des conflits pour lesquels des mécanismes de

résolution ont été proposés. Lorsqu'un autre type de conflit apparaît, sa résolution est entièrement laissée à la charge de l'utilisateur. Ces solutions considèrent les types de conflits suivants :

- conflit de modification qui correspond à des écritures divergentes effectuées sur des réplicas du même fichier. Ce type de conflit est résolu par des traitements adaptés à la sémantique des fichiers qui sont à la charge de l'utilisateur.
- Conflit de nommage qui survient lorsque deux fichiers du même répertoire ont le même nom. Ce type de conflit est résolu en renommant les fichiers concernés avec un suffixe unique.
- Conflit de suppression/modification qui survient lorsqu'un fichier est supprimé sur un réplica et mis à jour sur un autre réplica du système de fichiers. Ce type de conflit est résolu en déplaçant le fichier supprimé dans un répertoire particulier.
- Conflit de déplacement qui survient lorsqu'un fichier est déplacé différemment sur deux sites.

De manière générale, les systèmes dont la gestion de la cohérence des données est basée sur le maintien d'un historique des mises à jour, ont recours à des techniques de fusion des historiques selon certaines contraintes telles que l'ordre chronologique, pour la résolution des conflits [32, 102]. Ces solutions ne prennent pas en compte les cas où le réagencement de certaines opérations peut résoudre le conflit. IceCube [59] permet de trouver un tel réagencement des opérations, dans les historiques fusionnés, qui réduit le nombre de conflits non résolus automatiquement. Cependant, générer tous les réagencements possibles conduit à une explosion combinatoire. Pour éviter cela, IceCube réduit l'espace de recherche des différents réagencements en introduisant des contraintes similaires à l'ensemble des dépendances dans Bayou. L'utilisateur peut alors choisir parmi les réagencements proposés, celui qui servira pour la résolution du conflit. Notre gestion de la cohérence est également basée sur le maintien d'historiques des mises à jour. Par conséquent, les techniques de réagencement telles qu'elles sont proposées dans IceCube peuvent être utilisées pour la résolution des conflits.

Par ailleurs, la synchronisation des réplicas s'effectue également, lorsqu'une entité établit une connexion avec sa machine de référence. La synchronisation s'effectue alors entre le réplica localement stocké sur l'entité et sa copie de référence, comme décrit dans la section suivante.

V.5 Synchronisation avec la copie de référence

Dans un domaine de sécurité, la synchronisation des réplicas d'une donnée avec leur copie de référence permet de propager à terme toutes les mises à jour à tous les réplicas de la donnée. Ceci permet également de maintenir un réplica sur une machine fiable. La fréquence de ces synchronisations peut être aléatoire et déterminée par la connectivité du réseau sans fil en mode *ad hoc*. Dans ce cas, le réplica est synchronisé avec sa copie de référence quand l'entité qui le stocke se trouve dans la portée de communication directe de sa machine de référence. Par ailleurs, la fréquence de synchronisation peut être périodique, auquel cas les communications entre l'entité mobile et sa machine de référence devront éventuellement être relayées par des stations de base si elles ne se trouvent pas dans la portée de communication directe l'une de l'autre. Ce deuxième choix est évidemment plus contraignant, notamment quand aucune station de base n'est inaccessible ou quand la connexion aux stations de base est payante.

La synchronisation est effectuée en comparant le CCL associé au réplica avec la liste des mises à jour associée à la copie de référence. En effet, nous maintenons pour chaque copie de référence une liste des mises à jour qui correspond aux différentes synchronisation avec les réplicas de la donnée. Chaque élément de cette liste contient : (i) une estampille qui correspond à un compteur incrémenté à chaque fois qu'une nouvelle synchronisation a été effectuée, (ii) l'identité de l'entité sur laquelle est stocké le réplica synchronisé, et (iii) la liste des mises à jour propagées vers la copie de référence. Un nouvel élément est ajouté à cette liste chaque fois que l'estampille est incrémentée. L'estampille est incrémentée à chaque modification ou mise à jour de la copie de référence, qui correspond, en fait, à une synchronisation avec un réplica qui contient de nouvelles mises à jour. Le maintien de cette liste des mises à jour permet de synchroniser des réplicas dont l'estampille est inférieure à l'estampille actuelle qui correspond au dernier élément de la liste. La figure V.7 montre un exemple de liste de mises à jour associée à une copie de référence. Le protocole de synchronisation avec la copie de référence est détaillé dans la section suivante.

V.5.1 Protocole

La synchronisation avec la copie de référence est à l'initiative de chaque entité mobile. Soit r_i un réplica de donnée stocké sur l'entité mobile i. L'entité i débute la

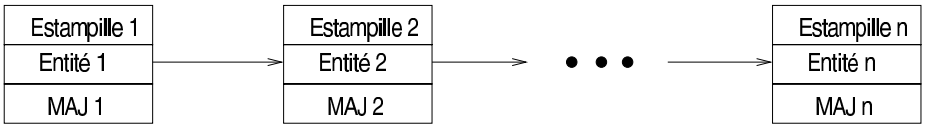

Figure V.7 – Exemple de liste de mise à jour au niveau de la copie de référence

synchronisation en envoyant à sa machine de référence le CCL associé à son réplica local r_i. Puis, la synchronisation se poursuit sur deux niveaux : celui de la machine de référence et celui de l'entité mobile.

Au niveau de la machine de référence, la première étape consiste à vérifier si effectivement une copie de référence existe pour le réplica concerné. Si cette copie de référence n'existe pas, deux cas sont possibles : (i) la copie de référence a été supprimée de la machine de référence ou (ii) il s'agit de la première synchronisation d'une donnée qui a été localement créée sur l'entité i. Dans le premier cas, c'est à l'utilisateur de décider de garder ou non le réplica local sur l'entité i. Le second cas, est distingué du premier par l'attribution de la valeur *NIL* à l'estampille du CCL associé au réplica. On crée alors une copie de référence à partir de r_i ainsi qu'une nouvelle liste de mises à jour, dont l'estampille est égale à un. Suite à cela, la nouvelle valeur de l'estampille est communiquée à l'entité i.

Dans le cas où une copie de référence existe, la liste des mises à jour associée au CCL du réplica r_i est calculée. Cette liste est comparée à la liste des mises à jour effectuées au niveau de la copie de référence à partir de la valeur de l'estampille qui correspond au CCL de r_i. En effet, cette valeur correspond à la dernière synchronisation avec la copie de référence. Si l'estampille associée au réplica r_i est égale à e_i, alors le réplica r_i est à jour par rapport aux mises à jour au niveau de la copie de référence allant de la valeur de l'estampille égale à 1 jusqu'à la valeur de l'estampille égale à $e_i - 1$ (si $e_i = 1$ cela correspond à la première synchronisation avec la copie de référence). La synchronisation doit être effectuée par rapport aux mises à jour allant de la valeur de l'estampille égale à e_i jusqu'à la dernière valeur de l'estampille. La comparaison de ces deux listes s'effectue en calculant leur intersection. Si l'intersection est vide, alors il n'existe aucune mise à jour conflictuelle. La synchronisation des deux réplicas (le réplica r_i et sa copie de référence) s'effectue en propageant les mises à jour déterminées au niveau de la copie de référence vers le réplicas r_i , et vice versa (c.-à-d., propager les mises à jour déterminées au niveau du réplicas r_i vers sa copie de référence). Si l'intersection n'est pas vide, alors des mises à jour conflictuelles ont

		Nombre de messages [min, max]					
		Prop. paresseuse			Prop. non paresseuse		
Op. préc.	**Op. requise**	**émis**	**reçus**	**ignorés**	**émis**	**reçus**	**ignorés**
Ecriture	Ecriture	[2, n]	[2, n]	[2(n-2), n(n-2)]	4	2n	2(n-2)
	Lecture	[2, n]	[n, 2(n-1)]	[(n-2), (n-1)(n-2)]	0	0	0
Lecture	Ecriture	[1, 3]	[n-1, n+1]	[0, 2(n-2)]	4	2n	2(n-2)
	Lecture	[0, 2]	[0, 2]	[0, 2(n-2)]	0	0	0

Table V.2 – Coût de l'accès aux données en termes de messages.

été détectées nécessitent le recours à la résolution des conflits.

Au niveau de l'entité i, la synchronisation de r_i avec sa copie de référence génère la suppression de l'historique log_i et la mise à jour de l'estampille locale, ainsi que la mise à jour du réplica local si nécessaire.

V.6 Evaluation

Notre gestion de la cohérence hybride permet d'exploiter au mieux la connectivité du réseau afin de répondre au besoins des applications sans contraindre la mobilité des entités.

La gestion de la cohérence forte, au niveau d'un groupe, permet l'accès à des réplicas cohérents et à jour, tout en réduisant les échanges de messages nécessaires pour le maintien de la cohérence forte. Comme nous l'avons mentionné, l'envoi de messages *via* le réseau sans fil en mode *ad hoc* induit une importante consommation d'énergie au niveau des entités émettrices et réceptrices ainsi qu'au niveau des entités qui se trouvent dans leur portée de communication [22, 37]. Dans la section VII.2.3.2, nous avons évalué cette consommation par l'intégration de la gestion d'accès aux données dans un prototype de système de fichiers distribué mobile. En ce qui concerne le surcoût mémoire, les méta-données générées par la gestion d'accès aux données restent négligeables par rapport à la taille des données, comme nous le montrons dans la section VII.2.1.2. Par ailleurs, les temps de réponse sont évalués dans la section VII.2.2.2 où nous montrons qu'ils sont proportionnels à la taille des mises à jour, tandis que la taille du groupe les affecte faiblement.

La propagation paresseuse des mises à jour au sein d'un groupe (voir section V.3) permet de réduire considérablement le nombre des entités qui reçoivent/émettent effectivement des messages pour l'accès à des données cohérentes. De cette façon, la consommation d'énergie, par un plus grand nombre d'entités due au mode *ad hoc*, est compensée par le nombre réduit de messages échangés afin d'accéder à des données cohérentes. Dans le cas de la propagation non paresseuse des mises à jour, le nombre de messages échangés pour accéder à des données cohérentes est beaucoup plus élevé. Nous rappelons que la propagation non paresseuse consiste à propager les mises à jour dès qu'elles se produisent et dès que la connectivité du réseau le permet. Le tableau V.2 donne le nombre de messages émis, reçus et ignorés dans un groupe de taille n, pour l'accès à un réplica local, en fonction du mode de propagation des mises à jour (paresseuse et non paresseuse), de l'opération requise et de la dernière opération effectuée sur un réplica de la donnée qui permet de déterminer l'état des autres réplicas stockés dans le groupe.

Le nombre de messages échangés, quand la propagation des mises à jour est paresseuse, varie en fonction de l'état du réplica sur lequel l'opération est requise. Si une lecture est requise alors que le réplica local est dans l'état UpdateRead, alors deux messages sont échangés avant que l'accès en lecture soit effectif. En revanche, si le réplica local est dans l'état Read, aucun message n'est échangé. Le nombre de messages pour l'accès à un réplica dans l'état Invalid dépend de la longueur de la chaîne générée par la recherche du dernier écrivain (voir Section V.3.2). Ce nombre varie de 2 à n messages échangés pour les accès en écriture ou en lecture.

Afin d'illustrer l'avantage de la propagation paresseuse par rapport à la propagation non paresseuse, en termes de nombre de messages échangés (émis/reçus) et de messages ignorés pour l'accès aux données, nous considérons le cas des écritures successives où la propagation paresseuse est la plus utilisée. Soient G un groupe formé de n entités mobiles ($\|G\| = n$), r_i un réplica stocké sur l'entité i, tel que $i \in G$. Supposons que chaque entité appartenant au groupe possède un réplica de r. Considérons le cas de p écritures successives sur des réplicas stockés sur des entités mobiles différentes avec respectivement une propagation paresseuse et non paresseuse des mises à jour. Initialement tous les réplicas de r sont dans l'état Read. Un premier accès en écriture est demandé sur le réplica r_b stocké sur l'entité b. Nous supposons que l'entité b possède le jeton.

Ecritures successives avec propagation paresseuse des mises à jour. Lors du premier accès en écriture, un message est émis par l'entité b et est reçu par les $n-1$ autres entités du groupe. Soit $p = 2 \ldots n$, pour valider le $p^{ième}$ accès en écriture p messages sont émis et reçus par p entités, et $p(n-2)$ messages sont ignorés. En revanche, si $p > n$, pour valider le $p^{ième}$ accès en écriture n messages sont émis et reçus par n entités, et $n(n-2)$ messages sont ignorés. Soit au total, pour p accès en écriture effectués à chaque fois sur des réplicas différents ($p > n$), $(1 + \sum_{i=2}^{n} i + \sum_{i=n+1}^{p} n)$ soit $(\frac{n(2p-n+1)}{2})$ messages sont émis ; $(n - 1 + \sum_{i=2}^{n} i + \sum_{i=n+1}^{p} n)$ soit $(\frac{n(2p-n+3)-4}{2})$ messages sont reçus et $(\sum_{i=2}^{n} i(n-2) + \sum_{i=n+1}^{p} n(n-2))$ soit $(\frac{(n-2)(n+2np-2)}{2})$ messages sont ignorés.

Ecritures successives avec propagation non paresseuse des mises à jour. Dans ce cas, lors du premier accès en écriture et pour le $p^{ième}$ ($p \in \mathbb{N}^+$) accès en écriture, 4 messages sont émis, $2n$ messages sont reçus et $2(n-2)$ messages sont ignorés.

Pour comparer les deux propagations de mises à jour, nous considérons uniquement les messages effectivement émis et reçus. En effet, la consommation d'énergie pour ignorer des messages est négligeable comparée à la consommation d'énergie nécessaire pour émettre ou recevoir un message. Soient $msg_{par} = n(2p-n+2)-2$ la somme des messages émis et reçus avec la propagation paresseuse et $msg_{npar} = 2(np+2p)$ la somme des messages émis et reçus avec la propagation non paresseuse. Nous démontrons que $\forall\, p \in \mathbb{N}^+$ et $\forall\, n \in \mathbb{N}^+$ tel que $n \geq 2$, $msg_{npar} > msg_{par}$. En effet, considérons la différence $msg_{npar} - msg_{par} = 4p + n^2 - 2n + 2$, pour que cette différence soit positive p doit être supérieur à $\frac{-n^2+2n-2}{4}$. Etudiant l'équation $e = -n^2 + 2n - 2$, son determinant est négatif ($\Delta = -4$) ce qui signifie que $\forall\, n \in \mathbb{N}^+, e < 0$. Ainsi, $\forall\, p \in \mathbb{N}^+$ et $\forall\, n \in \mathbb{N}^+, p > 0 > \frac{-n^2+2n-2}{4}$ par conséquent $msg_{npar} - msg_{par} > 0$. Donc, le nombre de messages échangés, lors de la propagation non paresseuse des mises à jour, est supérieur au nombre de messages échangés lors de la propagation paresseuse des mises à jour. Cette constation est confirmée par l'évaluation des temps de réponse lors d'une propagation paresseuse et non paresseuse des mises à jour, présentée dans la section VII.2.2.2.

Pour un exemple numérique (plus parlant), considérons un groupe de 10 entités ($n = 10$) et 10 accès en écriture. Le nombre de messages nécessaires pour effectuer 10 accès en écriture sur des réplicas différents[1], dans le cas d'une propagation pares-

[1]Nous supposons que chaque entité du groupe possède un réplica.

seuse des mises à jour, est de 118 messages. Tandis que dans le cas d'une propagation non paresseuse des mises à jour, ce nombre s'élève à 240 messages.

V.7 Discussion

Les systèmes distribués pour l'accès aux données dans des environnements mobiles sans fil étudiés dans le chapitre III adoptent tous une approche optimiste pour la gestion de la cohérence des données répliquées. Ces systèmes se basent sur la connectivité intermittente dans ce type d'environnement et par conséquent l'impossibilité de maintenir une cohérence forte entre les réplicas. Cependant, la gestion optimiste de la cohérence des réplicas ne favorise pas la collaboration synchrone qui devient de plus en plus demandé par les utilisateurs, car leur mobilité croissante génère de plus en plus de situations où des groupes d'individus équipés d'ordinateurs portables souhaitent travailler ensemble loin de leur lieu de travail conventionnel.

Nous proposons une approche originale à la gestion de la cohérence des données répliquées. En effet, nous adoptons une gestion hybride de la cohérence combinant la cohérence faible et la cohérence forte en fonction du contexte dans lequel se trouve l'entité mobile. Nous permettons ainsi aux entités isolées ou celles qui se trouvent dans des groupes disjoints de bénéficier d'une cohérence faible. En revanche, la gestion de la cohérence forte au sein d'un groupe permet aux applications qui nécessitent une collaboration synchrone d'avoir accès à la même version des données sur toutes les entités du groupe. Toutefois, le maintien de la cohérence forte ne contraint en aucune manière la mobilité des entités. De plus, nous proposons une propagation paresseuse des mises à jour qui permet de minimiser le nombre de messages échangés pour le maintien de la cohérence des réplicas d'un groupe et par conséquent, de réduire la consommation d'énergie au niveau des entités mobiles.

La gestion de la cohérence faible peut générer des conflits lors de la synchronisation des réplicas. De ce fait, tous les systèmes ayant une gestion faible de la cohérence des données proposent des protocoles de détection et de résolution des conflits. Nous détectons les conflits de mises à jour en comparant les historiques (CCLs) associés à chaque réplicas. La comparaison de ces historiques permet de déterminer, dans un premier lieu, la liste des mises à jour nécessaires pour synchroniser les réplicas, et le cas échéant les mises à jour conflictuelles. Le maintien d'historique pour chaque réplicas permet d'employer des techniques de fusion et de réagencement pour la réso-

lution des conflits. Toutefois, la résolution totalement automatique n'est pas possible, et l'intervention de l'utilisateur peut être indispensable dans certains cas.

VI Gestion de la disponibilité des données

Dans les environnements mobiles, l'accès aux données distantes est en général conditionné par la possibilité d'établir une connexion sans fil avec le serveur de données. Ainsi, les entités mobiles déconnectées ne peuvent avoir accès qu'aux données stockées dans leur cache local. Dans le cas des réseaux en mode *ad hoc*, les entités mobiles peuvent, dans certains cas, avoir accès aux données stockées sur d'autres entités mobiles se trouvant dans leur portée de communication. Mais, le problème de la connectivité intermittente de ces dernières reste posé. La disponibilité des données peut être abordée de deux points de vue : (i) du point de vue d'une entité isolée qui doit avoir recours à son cache local lorsque le serveur de données est inaccessible ; et (ii) du point de vue d'un ensemble d'entités mobiles formant un réseau en mode *ad hoc*, où même si le serveur de données est inaccessible, les entités mobiles peuvent avoir recours aux données stockées sur des entités paires dans leur portée de communication. La solution pour augmenter la disponibilité des données, et sur laquelle s'accordent les chercheurs dans le domaine de la gestion de l'accès aux données dans les environnements mobiles sans fil, consiste à répliquer ces données [45, 6, 48]. Au niveau d'une entité isolée, cette solution se traduit par des techniques de préchargement qui consistent à anticiper les déconnexions des entités mobiles en répliquant dans leur cache local les données nécessaires pour la période de déconnexion. Au niveau d'un réseau en mode *ad hoc*, cette solution consiste à avoir des réplicas accessibles dans la portée de communication. En outre, augmenter la disponibilité des données est cruciale pour le bon déroulement d'un travail collaboratif. En effet, dans le cadre d'un groupe de travail, les données sont distribuées entre les différentes entités participants au travail collaboratif en fonction de leurs contributions et de l'espace de stockage disponible sur leurs ordinateurs respectifs. Par conséquent, la dé-

connexion (volontaire ou involontaire) de l'un des participants peut générer la perte des données indispensables pour le reste du groupe. Une réplication excessive ou systématique des données peut compromettre les ressources de certaines entités mobiles et notamment augmenter leur consommation d'énergie et diminuer l'espace de stockage.

Ainsi, pour augmenter la disponibilité des données, tout en évitant ces inconvénients, nous considérons que les données utiles pour le travail collaboratif, entre autre, doivent être répliquées de manière rationnelle et à des moments opportuns en prévention des déconnexions. Nous considérons également que l'intervention de l'utilisateur doit être réduite au maximum, et en particulier lorsqu'il est possible de prendre en compte les changements dans le contexte d'exécution de manière transparente. A cet effet, nous proposons d'anticiper l'indisponibilité des données à travers un modèle de réplication adaptative aux ressources de chaque entité mobile et un protocole de choix des entités mobiles allant stocker les nouveaux réplicas qui favorise la préservation de leurs ressources. Notre modèle de réplication favorise également le partage des ressources au sein de groupes d'entités mobiles paires et offre ainsi un support à la collaboration synchrone. Nous augmentons ainsi la disponibilité des données sans contraindre la mobilité des entités et sans intervention explicite de la part de l'utilisateur [16, 18].

Dans la suite de ce chapitre, nous présentons dans la section VI.1 notre approche adaptative de la gestion de la disponibilité des données. Dans la section VI.2, nous introduisons notre protocole de réplication qui permet d'adapter la réplication des données aux ressources disponibles au niveau des entités mobiles et d'augmenter la disponibilité des données *via* une réplication préventive permettant d'anticiper les déconnexions. Ce chapitre se conclut par une évaluation VI.3 de la gestion de la disponibilité en termes de complexité théorique, ainsi qu'une discussion VI.4 par rapport aux travaux existants.

VI.1 Approche adaptative de la gestion de la disponibilité des données

Comme il a été mentionné précédemment, les protocoles de réplication existants pour les systèmes distribués mobiles, ne prennent pas en compte les contraintes liées aux ressources critiques des entités mobiles (voir chapitre III). Toutefois, il est primor-

dial de faire un compromis judicieux entre la réplication des données, la gestion de leur cohérence et la consommation de ressources au niveau des entités mobiles (en particulier l'énergie). A cet effet, nous adoptons une approche adaptative à la gestion de la disponibilité des données par rapport aux ressources disponibles des entités mobiles et à la connectivité du réseau.

Au niveau du domaine de sécurité, la disponibilité des données est assurée par le maintien de copies de référence sur des machines fiables. En revanche, au sein d'un groupe, la disponibilité des données est assurée par leur réplication sur les entités du groupe. Notre protocole de la gestion de la disponibilité des données prend en compte les ressources disponibles au niveau de chaque entité mobile appartenant au groupe, ainsi que les relations sémantiques entre les différentes données répliquées dans le groupe. Rappelons que d'après la gestion de groupe, toutes les entités mobiles ont une connaissance des données stockées sur les entités paires appartenant au groupe et de l'état de leurs ressources. Ceci permet d'identifier les entités mobiles qui peuvent participer à l'accroissement de la disponibilité des données en fonction de leurs ressources et d'adapter la réplication des données en conséquence. Notre gestion adaptative de la disponibilité des données est construite autour de trois notions que nous définissons dans ce chapitre :

- le profil d'une entité mobile, qu'elle communique au *leader* du groupe pendant l'étape de l'échange des méta-données, indique : (i) l'espace de stockage disponible pour le partage ; (ii) la possibilité de stocker des réplicas de données, autre que les réplicas nécessaires pour l'accès local, en fonction de l'énergie disponible ; et (iii) le temps prévu avant de quitter le groupe, déterminé grâce aux indications de l'utilisateur (p.ex., un utilisateur peut indiquer dans son agenda électronique l'heure à laquelle il doit quitter le groupe) ;
- les Replicas de Travail qui sont stockés en local sur l'entité mobile en fonction des accès effectifs à l'instar des modèles de réplication proposés dans la littérature ;
- les Replicas Preventifs qui permettent d'anticiper l'indisponibilité des données provoquée par les déconnexions des entités mobiles en garantissant la disponibilité d'au moins un réplica à jour par donnée au sein du groupe. Ils sont créés à des moments opportuns sur des entités mobiles choisies en fonction de leur profil.

La gestion des réplicas est déterminée en fonction du profil des entités mobiles que nous le définissons dans la section suivante. Puis, nous définissons le profil d'un

groupe dans la section VI.1.2. Enfin, la section VI.1.3 présente la classification des profils qui permet d'adapter la réplication des données à la disponibilité des ressources au niveau du groupe.

VI.1.1 Profil des entités mobiles

Dans un groupe d'entités mobiles, l'indisponibilité des données, localement stockées sur ces entités, est essentiellement due aux déconnexions de ces dernières. Nous distinguons deux types de déconnexion : volontaires et involontaires. Les utilisateurs ont recours aux déconnexions volontaires soit pour se constituer en singleton afin de préserver les ressources qui étaient partagées au sein du groupe, soit parce qu'ils s'éloignent de la portée de communication du groupe. Les déconnexions involontaires sont causées, soit par des pannes de batterie (dans le cas des entités mobiles non alimentées par le courant électrique), soit par la connectivité intermittente dans les environnements mobiles. Ainsi, nous identifions trois paramètres qui contribuent à réduire la disponibilité des données, soit en favorisant les déconnexions, soit en limitant la réplication : (i) l'éloignement de la portée de communication du groupe, (ii) le manque d'énergie et (iii) l'insuffisance de l'espace de stockage. Mesurer ces paramètres périodiquement nous permet d'anticiper les déconnexions et d'adapter la réplication des données pouvant devenir inaccessibles en conséquence.

Nous définissons la notion de profil d'une entité mobile qui regroupe ces paramètres et permet de détecter l'indisponibilité prochaine de certaines données. Le profil d'une entité mobile indique :

– L'autonomie de la batterie de cette entité mobile. Elle est calculée à partir des fonctions système à travers des interfaces de type ACPI[1] (*Advanced Configuration and Power Interface*) qui permettent la configuration et la gestion de l'énergie des entités mobiles à partir du système d'exploitation et non plus à partir du BIOS comme c'était notamment le cas avec des interfaces de type APM[2] (*Advanced Power Management*), ce qui permet plus de souplesse et d'adaptabilité. Nous estimons l'évolution, dans le temps, de la quantité d'énergie disponible sur une entité mobile, non alimentée par un courant électrique, par la fonction

[1] `http://www.acpi.info/index.html`
[2] `http://www.intel.com`

linéaire décroissante suivante :

$$Energie(t) = init - \frac{init}{batterie} \times t \quad \text{(VI.1)}$$

où t est la variable temps, $init$ est l'autonomie initiale de la batterie au moment de l'initialisation du groupe, et $batterie$ est une estimation de la durée de vie de la batterie définie en fonction de la charge de l'entité mobile. Ces coefficients sont recalculés périodiquement.
$Energie(t)$ permet de déterminer si l'entité mobile peut être sollicitée pour stocker un réplica autre que ceux nécessaires à l'accès local de son utilisateur. En effet, recevoir un nouveau réplica implique une communication sans fil qui génère une importante consommation d'énergie [37] et peut compromettre l'énergie disponible au niveau de l'entité concernée. De plus, $Energie(t)$ permet d'anticiper les déconnexions dues à la baisse d'énergie en générant les Replicas Preventifs nécessaires au maintien de la disponibilité des données qui peuvent devenir inaccessibles au sein du groupe à cause de cette déconnexion.

– La durée prévue dans le groupe, qui est estimée en fonction des indications de l'utilisateur, par exemple son emploi du temps. Nous estimons l'évolution de la durée prévue dans le groupe par la fonction linéaire décroissante suivante :

$$Duree(t) = cst - t \quad \text{(VI.2)}$$

où cst est la durée estimée, pendant laquelle l'utilisateur est supposé être connecté au groupe de travail. $Duree(t)$ permet de générer des Replicas Preventifs, si nécessaire, quand le moment de la déconnexion volontaire de l'utilisateur est proche.

– L'espace de stockage disponible pour le partage. En effet, chaque entité mobile appartenant au groupe partage une partie de son espace de stockage pour le maintien de la disponibilité des données, c'est à dire, pour le stockage des Replicas Preventifs. Nous représentons l'espace de stockage disponible pour le partage par la variable $Stockage$, dont la valeur dépend du taux de réplication et doit être recalculée périodiquement. $Stockage$ permet de déterminer si l'entité mobile peut être sollicitée pour stocker un Replica Preventif.

Maintenir un profil pour chaque entité mobile permet de rendre la gestion de la disponibilité des données transparente à l'utilisateur. D'une part, il n'a plus à se soucier de l'existence d'autres réplicas de ces données locales au niveau du groupe qu'il va quitter, ni à se soucier de quelles données répliquer ni où stocker ces réplicas. D'autre

part, les entités du groupe sont assurées que les données qui leurs sont nécessaires et qui se trouvent sur l'entité allant être déconnectée, seront répliquées sur d'autres entités connectées au groupe.

VI.1.2 Profil du groupe

L'ensemble des profils des entités mobiles définit le profil du groupe. Il est mis à jour périodiquement, par le *leader* du groupe. Cette étape de mise à jour est intégrée dans le processus d'échange des méta-données. Le profil du groupe, à l'instar des autres méta-données, est propagé vers toutes les entités appartenant au groupe (voir chapitre IV). Ainsi, chaque entité a une vision globale des ressources disponibles au niveau du groupe. Ceci permet de choisir les entités appropriées pour participer à l'accroissement de la disponibilité des données (voir section VI.2.2).

La période de mise à jour du profil du groupe découle de l'heuristique donnée dans la section IV.4.1. Elle permet d'avoir un compromis entre la détection instantanée des changements des profils des entités mobiles et la consommation d'énergie générée par les communications sans fil nécessaires à cette détection. De plus, les fonctions $Energie(t)$ et $Duree(t)$ sont paramétrées par la variable temps, ce qui permet d'avoir une approximation de la disponibilité des deux ressources représentées par ces fonctions sur une période de temps déterminée. Par ailleurs, mettre à jour le profil du groupe périodiquement permet d'avoir des approximations fiables de la disponibilité des ressources au niveau des entités mobiles et d'adapter la gestion de la disponibilité des données en conséquence, tout en réduisant la consommation d'énergie et sans aucune intervention de la part de l'utilisateur.

VI.1.3 Classification des profils

Le choix des entités devant participer à l'accroissement de la disponibilité des données est déterminé par la classification de leur profil suivant des critères quantitatifs. Pour ce faire, nous associons aux profils des entités appartenant au groupe les attributs qualitatifs suivants :

- L'attribut *Optimal* qui désigne une entité mobile apte à collaborer activement et à partager ses ressources avec les autres entités mobiles du groupe.
- L'attribut *Acceptable* qui désigne une entité mobile capable de partager ses res-

sources avec les autres entités du groupe, mais constitue un second choix par rapport aux entités dont le profil est *Optimal*.

– L'attribut *Faible* qui désigne une entité mobile qui dispose de peu de ressources, et qui ne peut, par conséquent, pas les partager avec les autres entités du groupe.

Les classifications des profils sont valables pour la période de temps déterminée par la période de mise à jour du groupe T_g. Elles sont obtenues par la classification des fonctions ($Energie(t)$, $Duree(t)$, et $Stockage$) qui définissent un profil, en fonction des seuils définis pour chacune de ces fonctions (e_{th}, t_{th}, et s_{th}). Ces seuils représentent les valeurs minimales pouvant être atteintes par les fonctions correspondantes et en dessous desquelles les risques de perte de données augmentent. Dans la pratique, ces seuils sont fixés par l'utilisateur ou adaptés par l'application. Par exemple, le seuil attribué à la fonction $Energie(t)$ est de l'ordre de 25% de l'autonomie totale de la batterie. Ainsi, pour une période donnée T', dans un groupe $G = (N,\ A)$ de n entités mobiles, nous associons les attributs *Optimal*, *Acceptable*, et *Faible* aux fonctions définissant le profil d'une entité mobile comme suit. Soit $Energie_i(t)$ l'estimation de l'énergie disponible sur l'entité mobile i. On dit que $Energie_i(t)$ est :

– *Optimal*, si la fonction est maximale sur T' et supérieure au seuil e_{th} donné. De manière plus formelle, $Energie_i(t)$ est *Optimal* si et seulement si :

$$\begin{array}{c} \forall t \in [0, T'], \forall j = 1, ..., n \; et \; j \neq i,\ j \in N \\ Energy_i(t) \geq e_{th} \; et \; Energy_i(t) \geq Energy_j(t) \end{array} \tag{VI.3}$$

– *Acceptable*, si la fonction n'est pas *Optimal*, mais reste supérieure au seuil. De manière plus formelle, $Energie_i(t)$ est *Acceptable* si et seulement si :

$$\forall t \in [0, T'], Energy_i(t) \geq e_{th} \tag{VI.4}$$

– *Faible*, si la fonction est inférieure au seuil pour un t donné dans la période T'. De manière plus formelle, $Energie_i(t)$ est *Faible* si et seulement si :

$$\exists t \in [0, T'], Energy_i(t) \leq e_{th} \tag{VI.5}$$

La définition des attributs précédents est appliquée de manière similaire à la fonction $Duree(t)$ en considérant le seuil t_{th}. En ce qui concerne le paramètre $Stockage$, ces attributs sont définis comme suit : $Stockage$ est *Optimal* si et seulement si : $\forall j = 1, ..., n \; et \; j \neq i,\ j \in N \; Stockage_i \geq s_{th} \; et \; Stockage_i \geq Stockage_j$, il est *Acceptable* si et seulement si : $Stockage_i \geq s_{th}$ et *Faible* si et seulement si : $Stockage_i \leq s_{th}$.

En combinant les attributs des fonctions définissant le profil d'une entité mobile, nous classifions le profil d'une entité mobile comme indiqué dans le tableau VI.1. Les

$Energie(t)$	$Duree(t)$	$Stockage$	**Profil de l'entité mobile**
Optimal	Optimal	Optimal	Optimal
Optimal	Optimal	Acceptable	Optimal
Optimal	Optimal	Faible	Acceptable
Optimal	Acceptable	$\forall$	Acceptable
Optimal	Faible	$\forall$	Faible
Acceptable	Optimal	$\forall$	Acceptable
Acceptable	Acceptable	$\forall$	Acceptable
Acceptable	Faible	$\forall$	Faible
Faible	$\forall$	$\forall$	Faible

Table VI.1 – Classification du profil d'une entité mobile

attributs des fonctions $Energie(t)$, $Duree(t)$ et $Stockage(t)$ n'ont pas le même poids dans la classification du profil de l'entité mobile. Les fonctions $Energie(t)$ et $Duree(t)$ représentent l'évaluation des paramètres pouvant générer la déconnexion de l'entité mobile. Ainsi, si l'attribut *Faible* est associé à l'une d'entre elles, le profil de l'entité mobile est considéré *Faible*. En effet, l'attribut *Faible* associé à la fonction $Energie(t)$ signifie qu'elle est inférieure au seuil donné. Par conséquent, l'entité mobile n'a plus assez de batterie pour participer au travail collaboratif du groupe et risque de se déconnecter dans peu de temps. D'un autre côté, l'attribut *Faible* associé à la fonction $Duree(t)$ signifie que l'entité mobile va se déconnecter du groupe dans peu de temps. Ainsi, une entité allant se déconnecter du groupe, que cela soit de manière volontaire ou involontaire, ne peut être sollicitée pour stocker des Replicas Preventifs afin d'accroître la disponibilité des données. De même, les données localement stockées sur cette entités doivent être répliquées sur d'autres entités si cela n'est pas déjà le cas (voir section VI.2.1). Par ailleurs, le paramètre $Stockage$ n'a pas la même influence sur le profil de l'entité mobile. Par conséquent, même si l'attribut *Faible* est associé à ce paramètre, le profil de l'entité mobile reste *Acceptable*, car l'espace de stockage peut toujours être libéré si nécessaire par des techniques de remplacement et de ramasse miettes.

Les attributs associés au profil de l'entité mobile aident à adapter la réplication en fonction des ressources disponibles dans le groupe mobile. De plus, ils permettent d'anticiper les déconnexions et de répliquer en conséquence les données dont la disponibilité est compromise.

VI.2 Protocole de réplication adaptative

Dans notre schéma de réplication adaptative, nous distinguons deux types de réplicas de données : les Replicas de Travail (RT), et les Replicas Preventifs (RP). Lorsqu'une entité mobile se joint à un groupe de travail, les réplicas de données stockés dans son cache local sont considérés comme des Replicas de Travail. De même, les réplicas générés par les demandes d'accès à des données stockées sur d'autres entités du groupe sont considérés comme des Replicas de Travail. La création de ces réplicas peut nécessiter la libération de l'espace de stockage en remplaçant des réplicas choisis selon la technique de remplacement utilisée. Parmi les techniques de remplacement utilisées on peut citer : le LRU (*Least Recently Used*), qui consiste à remplacer le réplica le moins récemment accédé ; le NRU (*Not Recently Used*), qui consiste à remplacer aléatoirement l'un des réplicas les moins récemment accédés ; et le NFU (*Not Frequently Used*), qui consiste à remplacer le réplica le moins fréquemment accédé depuis sa création. Ainsi, des anciens Replicas de Travail ou des réplicas peu accédés peuvent être remplacés à condition qu'il existe un autre réplica de la même donnée dans le groupe. Dans notre approche, nous considérons que l'existence d'au moins un réplica est suffisante pour le maintien de la disponibilité des données au sein d'un groupe. C'est dans ce but que nous définissons les Replicas Preventifs. Ils servent au maintien d'au moins un réplica à jour par donnée au sein du groupe. Ils sont créés quand les Replicas de Travail existants ne sont pas suffisants pour le maintien de la disponibilité des données comme précisé dans la section suivante. Ils permettent ainsi d'anticiper de futures déconnexions qui peuvent compromettre la disponibilité des données.

VI.2.1 Gestion des réplicas

Dans cette section, nous abordons la gestion des réplicas du point de vue de la gestion de la cohérence des Replicas de Travail et des Replicas Preventifs, des situations qui génèrent la création des Replicas Preventifs et également, des interactions qui existent entre les Replicas de Travail et les Replicas Preventifs.

VI.2.1.1 Gestion de la cohérence

La gestion de la cohérence des Replicas de Travail est différente de celle des Replicas Preventifs. Les Replicas de Travail sont accédés par les utilisateurs en lecture ou en écriture ; leur cohérence est gérée selon le protocole de cohérence hybride présenté dans le chapitre V. Ainsi, au niveau d'un groupe, ils bénéficient d'une gestion forte de la cohérence et de la propagation paresseuse des mises à jour entre les Replicas de Travail selon les demandes d'accès.

Un Replica Preventif est par définition le réplica le plus à jour concernant une donnée. Il est créé quand le Replica de Travail le plus à jour est sur le point de devenir indisponible (voire section VI.2.1.2). Ainsi, la gestion de la cohérence des Replicas Preventifs ne se pose pas, car ils sont les réplicas les plus à jour. Toutefois, si un autre Replica de Travail (s'il existe dans le groupe) est mis à jour, le Replica Preventif associé à cette donnée devient un Replica de Travail. De même, si une entité qui stocke un Replica Preventif se constitue en singleton, le Replica Preventif devient un Replica de Travail (voir section VI.2.1.3).

VI.2.1.2 Création des réplicas préventifs

Comme il a été précédemment mentionné, les Replicas Preventifs servent à anticiper d'éventuelles déconnexions pouvant rendre certaines données indisponibles pour le reste du groupe. Les éventuelles futures déconnexions sont détectées en analysant la classification des profils des entités appartenant au groupe. Ainsi, si le profil d'une entité mobile p devient *Faible*, nous distinguons les cas suivants :

- si p stocke la version la plus à jour d'un Replica de Travail, alors elle propage les mises à jour (si ce n'est pas déjà fait) à un autre Replica de Travail stocké sur une entité mobile choisie selon le critère de sélection présenté dans la section VI.2.2. Ainsi, ce Replica de Travail mis à jour devient en même temps un Replica Preventif (RP). Notons que la propagation des mises à jour de manière paresseuse n'est pas respectée dans ce cas, car il représente un cas d'urgence et peut causer la déconnexion de l'entité mobile ainsi que la perte des dernières mise à jour du Replica de Travail en question.
- si p cache le seul Replica de Travail d'une donnée, alors un nouveau réplica doit être créé. Le choix de l'entité mobile qui devra stocker le Replica Preventif est déterminé en combinant son profil avec des relations sémantiques entre les

Dans un groupe ad hoc		Événements	Actions
Plusieurs RTs	seulement un RT est à jour	p devient *Faible*	MAJ du RT qui devient aussi RP
	plusieurs RTs sont à jour	p' devient *Faible*	Aucune
Un seul RT	-	p' devient *Faible*	créer un RP sur une entité mobile appropriée

Table VI.2 – Gestion de la réplication dans un groupe ad hoc, p est l'entité mobile qui cache le réplica à jour et p' l'entité mobile qui cache un RT.

données déjà stockées et le Replica Preventif (voir section VI.2.2).
Le tableau VI.2 résume les différents cas de création des Replicas Preventifs. Notamment le cas où plusieurs Replicas de Travail existent au niveau du groupe, ou bien le cas où un seul Replica de Travail est stocké dans le groupe.

VI.2.1.3 Interactions entre réplicas de travail et réplicas préventifs

Nous venons de voir que des Replicas de Travail peuvent devenir des Replicas Preventifs dans le cas où des mises à jour leur sont propagées sans qu'il y ait eut de demandes d'accès à ces derniers. De même, des Replicas Preventifs peuvent se transformer en Replicas de Travail si :

- Ils ont été créés dans des groupes mobiles distincts. Ils deviennent des Replicas de Travail et sont considérés comme faisant parti du cache local si l'entité mobile qui les stocke se constitue en singleton ou se joint à un autre groupe du même domaine de sécurité.
- Ils sont effectivement accédés en lecture ou en écriture au niveau de l'entité qui les stocke.
- Une entité mobile appartenant au groupe, dont le profil est au moins *Acceptable* (voir tableau VI.1), demande un accès à ce Replica Preventif.
- Une entité mobile appartenant au groupe et stockant un Replica de Travail, non à jour, de la même donnée, demande un accès à ce Replica de Travail.

Dans ces cas, le Replica Preventif n'est plus le seul réplica à jour et de ce fait il devient un Replica de Travail. Dans la section suivante nous allons voir comment la répartition des Replicas Preventifs est déterminée en fonction du profil des entités mobiles

et des relations sémantiques entre les réplicas à créer et les réplicas déjà stockés sur l'entité en question.

VI.2.2 Répartition des réplicas

La répartition des réplicas est basée sur des critères différents, qu'il s'agisse de Replicas de Travail ou de Replicas Preventifs. Les Replicas de Travail sont répartis selon les demandes d'accès des différentes entités mobiles. Les Replicas Preventifs sont répartis selon deux critères : le profil de l'entité mobile et les relations sémantiques avec les réplicas déjà stockés sur cette entité.

Le choix des entités mobiles, appartenant à un groupe de travail, pour stocker des Replicas Preventifs est guidé en premier lieu par leurs profils. On choisit d'abord les entités dont le profil est *Optimal*, puis celles dont le profil est *Acceptable* (s'il n'existe pas d'entités avec un profil *Optimal*). Les entités mobiles dont le profil est *Faible* ne seront jamais choisis pour stocker un Replica Preventif. Une fois que les entités mobiles, avec les profils appropriés, sont sélectionnées, on affine cette sélection en choisissant les entités qui stockent déjà des Replicas de Travail du réplica de donnée concerné. Dans ce cas, le Replica de Travail est mis à jour sans tenir compte de la politique paresseuse de propagation des mises à jour. Ainsi, le Replica de Travail devient également un Replica Preventif. Si aucune entité mobile avec un profil *Optimal* ou *Acceptable* appartenant au groupe ne stocke un Replica de Travail correspondant au Replica Preventif à créer, nous considérons les relations sémantiques entre les réplicas stockés sur l'entité mobile, avec le meilleur profil, et le Replica Preventif à créer.

Le tableau VI.3 recense tous les cas possibles pour la sélection de l'entité mobile qui va stocker le Replica Preventif. La mention *Sélection Optimale* désigne le meilleur candidat pour stocker le Replica Preventif. Si de telles entités mobiles n'existent pas, on choisit une entité mobile parmi celles dont la mention est *Sélection Acceptable*. Si aucune entité mobile dans le groupe de travail ne bénéficie de la mention *Sélection Optimale* ou *Sélection Acceptable*, on choisit alors une entité mobile dont la mention est *Sélection Faible*. Cette mention signifie que l'entité mobile ne stocke pas un Replica de Travail ou bien des données sémantiquement liées au Replica Preventif considéré, mais son profil lui permet de participer à l'augmentation de la disponibilité des données et de stocker des Replicas Preventifs. En revanche, si le groupe n'est constitué que d'entités mobiles dont la mention est *Non Sélectionnée*, la réplication (la création

Profil	un RT existe	Des données sémantiquement liées	Choix de l'entité
Optimal	vrai	$\forall$	Sélection Optimale
Optimal	faux	vrai	Sélection Acceptable
Optimal	faux	faux	Sélection Faible
Acceptable	vrai	$\forall$	Sélection Acceptable
Acceptable	faux	vrai	Sélection Acceptable
Acceptable	faux	faux	Sélection Faible
Faible	$\forall$	$\forall$	Non Sélectionnée

Table VI.3 – Sélection de l'entité mobile pour la création d'un Replica Preventif

du Replica Preventif) ne pourra pas se faire sans compromettre les ressources de l'entité mobile sélectionné, dont le profil est déjà *Faible*.

Dans la section suivante, nous présentons comment les relations sémantiques entre réplicas sont déterminées.

VI.2.2.1 Relations sémantiques entre réplicas

Les relations sémantiques entre réplicas sont utilisées dans le cas où l'on souhaite sélectionner une entité mobile pour stocker le Replica Preventif d'un Replica de Travail unique dans le groupe, ou bien dans le cas où les entités qui stockent les autres Replicas de Travail ont un profil *Faible*.

Soit p_1 l'entité mobile qui stocke la version la plus à jour (par rapport au groupe) du Replica de Travail rt_1. Si le profil de p_1 devient *Faible*, il faut créer un Replica Preventif rp_1. Supposons qu'aucune entité mobile, qui cache localement rt_1, n'a un profil approprié (c.-à-d., *Optimal* ou *Acceptable*). Donc, p_1 doit choisir une entité mobile qui cache localement des données sémantiquement proches de rt_1. Les relations sémantiques entre les données sont relatives au type de données. Par exemple, si nous considérons un système de fichiers, l'ensemble des fichiers sémantiquement proches de rt_1 est choisi comme suit [68, 69] :

– en règle générale, les fichiers stockés sur p_1 ou dans le même répertoire que rt_1 sont les plus sémantiquement liés. Le nom du fichier donne également des indications sur les relations sémantiques. Ainsi, les fichiers avec des noms iden-

tiques, mais des extensions différentes, sont sémantiquement très liés.
- p_1 calcule les distances sémantiques entre rt_1 et les fichiers localement stockés. Ces distances sémantiques permettront de créer des groupes de fichiers sémantiquement liés. Notez, que rt_1 peut appartenir à plusieurs de ces groupes. Une entité mobile qui stocke localement n fichiers appartenant au groupe de rt_1 est considérée comme sémantiquement proche. Le paramètre n est déterminé par l'utilisateur ou par l'application.
- Après la création des groupes de fichiers sémantiquement liés, p_1 classe les entités mobiles suivant leurs éligibilités pour stocker un Replica Preventif et sélectionne celle qui est la plus appropriée comme résumé dans le tableau VI.3.

VI.3 Evaluation

Dans les chapitres précédents, nous avons évalué notre proposition en termes de communications sans fil générées. Dans le contexte des réseaux mobiles sans fil en mode *ad hoc*, les communications ont un impact majeur sur la consommation d'énergie au niveau des entités mobiles qui se trouvent dans la portée de communication. Pareillement, dans cette section, nous évaluons la gestion de la disponibilité des données en termes des communications sans fil générées.

L'élément clé dans la gestion de la disponibilité des données est le maintien d'une vue globale des profils de toutes les entités au niveau d'un groupe. Cela nous permet d'anticiper les déconnexions des entités mobiles pour augmenter la disponibilité des données. Les profils des entités mobiles sont intégrés dans les méta-données qu'elles échangent avec leur *leader* qui se charge de les fusionner et de les envoyer à tout le groupe. Par conséquent, cette étape ne génère pas de communications supplémentaires à celles déjà générés par les fonctionnalités décrites dans les chapitre IV et V.

La détection d'une prochaine déconnexion et le choix des entités allant stocker des Replicas Preventifs est basée sur la classification des profils. Cette étape est totalement décentralisée et réalisée indépendamment au niveau de chaque entité mobile du groupe. Par conséquent, elle ne génère aucune communication entre les entités mobiles.

Enfin, dans le cas où le profil d'une entité devient *Faible* et où la création d'un Replica Preventif est nécessaire, un message est envoyé par l'entité dont le profil est *Faible* à celle qu'elle a choisi pour stocker le Replica Preventif. Ce message contient

soit des mises à jour destinées à un Replica de Travail existant, soit une copie du réplica qui risque de devenir indisponible.

VI.4 Discussion

Dans l'état de l'art, la réplication des données est la solution proposée pour augmenter la disponibilité des données. Elle est utilisée aussi bien dans les systèmes distribués stationnaires que dans les systèmes distribués dédiés aux environnements mobiles sans fil. Dans ce dernier cas, les connexions intermittentes posent le problème de la déconnexion des utilisateurs mobiles pour des durées indéterminées. Ainsi, la réplication a été utilisée pour permettre aux utilisateurs mobiles de travailler hors connexion.

Nous abordons la réplication non pas du point de vue d'entités mobiles isolées qui doivent avoir recours à leur cache local en cas de déconnexion, mais du point de vue d'un groupe d'entités mobiles. Toutefois, notre objectif reste le même, c.-à-d., augmenter la disponibilité des données. Nous considérons le contexte des environnements mobiles où les réseaux sous-jacents sont des réseaux locaux sans fil en mode *ad hoc* et abordons la disponibilité des données du point de vue d'un groupe mobile formé par des entités se trouvant dans la portée de communication les unes des autres et partageant un centre d'intérêt commun, qui les incite à partager leurs données et leurs ressources. La réplication des données au niveau du cache local de chaque entité, afin de permettre aux utilisateurs mobiles de travailler hors connexion, suit le même principe que celui proposé dans les autres systèmes distribués. L'originalité de notre solution réside en deux points. Le premier consiste à offrir une réplication adaptative des données au sein d'un groupe. Cette réplication permet de maintenir la disponibilité des données partagées au sein du groupe qui peut être compromise par les déconnexions de certaines entités. Le deuxième point consiste à adapter la réplication aux ressources disponibles au niveau des entités appartenant au groupe. Ainsi, nous proposons de classifier les entités mobiles en fonction de leur profil qui indique la quantité des ressources disponibles. Les entités ayant un seuil acceptable de ressources sont choisies, puis nous affinons la sélection en considérant les réplicas déjà stockés au niveau de chaque entité. Ceci nous permet de choisir l'entité, pour stocker le nouveau réplica, qui a le plus de ressources ou la plus grande probabilité d'accéder le nouveau réplica dans le futur. Pour ce faire, nous utilisons les relations

sémantiques entre réplicas de données proposées dans les techniques de préchargement existantes dans la littérature.

VII Implémentation et évaluation

Afin, d'évaluer notre contribution du point de vue des performances, nous avons implémenté un prototype de système distribué appelé ADHOCFS qui intègre les fonctionnalités présentées dans les chapitres IV, V et VI dans un système de fichiers.

Dans la section suivante, nous présentons l'implémentation d'ADHOCFS. Puis, dans la section VII.2, nous présentons une évaluation de ses performances en termes du surcoût de données, des temps de réponse et de la consommation d'énergie générés par la gestion de groupe et celle de l'accès aux données.

VII.1 Implémentation

Pour mettre en oeuvre les fonctionnalités présentés dans les chapitres IV à VI, Nous avons implémenté un système de fichiers qui intègre les fonctionnalités décrite dans les chapitres IV à VI, offrant ainsi un système supportant le partage de fichiers distribué au-dessus d'un réseau mobile sans fil en mode *ad hoc* et en particulier offrant un support à la collaboration synchrone. Le choix d'implémenter un système de fichier nous permet d'englober un large éventail de données et d'offrir un support à la mobilité pour des applications existantes. Le développement de notre prototype a été réalisé au-dessus du système d'exploitation Linux Mandrake 7.1 (2.2.15-4mdk) dans le langage OCaml (Objective Caml[1]). OCaml est un langage sûr et fortement typé qui garantit l'intégrité des données manipulées par les programmes. De surcroît, le compilateur OCaml permet de minimiser la taille des exécutables particulièrement adaptée aux ordinateurs de poches.

L'architecture d'ADHOCFS, présentée dans la figure VII.1, est constituée de trois couches au-dessus du système d'exploitation. La première couche correspond à la

[1] http ://caml.inria.fr/index-fra.html

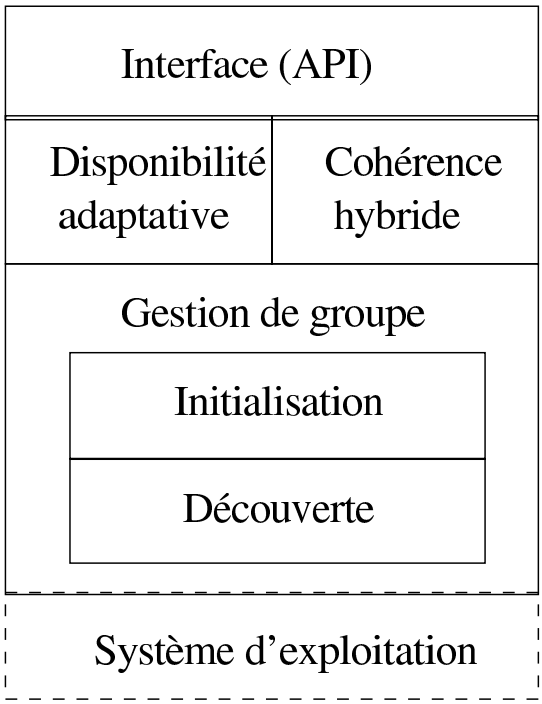

Figure VII.1 – Architecture de ADHOCFS.

gestion de groupe qui 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 gestion de groupe suit les étapes décrites dans le chapitre IV. Les groupes sont gérés sur la base d'une communication pair-à-pair. Pour la découverte des entités mobiles, nous utilisons le protocole SLP. Ce choix est en particulier motivé par son mode de fonctionnement décentralisé (sans *Directory Agent*). Dans ADHOCFS, chaque entité exécute un *Service Agent* qui prend en charge les requêtes de découverte des entités voulant former un groupe. Chaque entité choisit le domaine de sécurité dans lequel elle veut former un groupe en l'enregistrant auprès de son *Service Agent* en utilisant le format URL de SLP [42], à titre d'exemple, pour un domaine de sécurité *D1*, l'annonce sera $adhocfs://adresse\_IP:port/D1$. Chaque entité peut faire autant d'annonces que de domaines de sécurité auxquels elle appartient. Les entités qui ne souhaitent pas participer à des groupes désenregistrent l'annonce de service qui traduit leur appartenance au domaine de sécurité associé.

Parallèlement, les entités exécutent des *User Agent* qui lancent des requêtes de découverte de services basées sur le nom du domaine de sécurité. Dans l'exemple précédent, la requête de découverte émise par le *User Agent* sera de la forme *adhocfs :D1*. En réponse à ces requêtes, chaque entité reçoit l'ensemble des identités des entités ayant annoncé appartenir au domaine de sécurité recherché. Ainsi, chaque *User Agent* est capable de déterminer les entités appartenant au même domaine de sécurité que lui, et d'établir ainsi les listes des entités candidates pour former un groupe suivant le protocole décrit dans la section IV.2.1. Notons qu'ADHOCFS n'implémente pas de mécanisme de sécurité. Après l'étape de découverte des entités allant former un groupe, le processus de création d'un groupe se poursuit par l'étape d'initialisation. Dans cette étape, le choix d'un *leader* du groupe est essentielle pour réduire son coût en termes d'échanges de messages et donc en terme de consommation d'énergie. Le *leader* du groupe est l'entité dont l'identificateur est le minimum de tous les identificateurs dans le groupe. Dans ADHOCFS, chaque entité est identifiée par son adresse IP.

L'initialisation du groupe consiste principalement dans la fusion et l'échange des méta-données. Ainsi, chaque entité appartenant au groupe envoie au *leader* l'arborescence de son système de fichier local (sans les fichiers de données proprement dit) et son profil qui désigne l'état des ressources qu'elle met à disposition au sein du groupe. L'évolution du groupe est ensuite gérée par le déclenchement périodique des étapes de découverte des entités mobiles et l'étape d'initialisation. La période est calculée selon l'heuristique donnée dans la section IV.4.1.

Dans la gestion de la cohérence des données, décrite dans le chapitre V, le contenu de la liste des mises à jour associée au CCL attaché à chaque donnée dépend du type des données manipulées. Dans ADHOCFS, la liste des mises à jour associée au CCL attaché à chaque fichier contient les numéros des blocs de fichier modifiés. Le choix de représenter les numéros de blocs modifiés permet d'éviter la redondance dans le cas de nombreuses mises à jour du même bloc. De plus, lors de la détection des conflits, l'utilisation des numéros des blocs modifiés nous permet de cibler exactement les blocs de fichier sur lesquels le conflit s'est produit. Enfin, l'accès aux données se fait selon le protocole décrit dans la section V.3. La granularité de la réplication dans ADHOCFS est le fichier. L'interface de ADHOCFS, est similaire à celle du système de fichiers UNIX.

VII.2 Evaluations

Dans cette section, nous présentons les résultats de l'évaluation de notre prototype ADHOCFS en termes de surcoût relatif aux méta-données, de temps de réponse et de la consommation d'énergie.

VII.2.1 Surcoût des méta-données

La gestion des groupes, la gestion hybride de la cohérence des données et la gestion de la disponibilité des données génèrent des méta-données qui peuvent être regroupées en trois catégories :

- les méta-données relatives au groupe, qui contiennent la liste des identificateurs des entités qui forment le groupe,
- les méta-données relatives aux données proprement dites, qui contiennent pour chaque fichier de données : des CCLs, la liste des entités qui stockent localement ce fichier, un jeton, les files d'attente en lecture et/ou en écriture, ainsi que des informations nécessaires pour la gestion des verrous,
- les méta-données relatives aux profils des entités, qui contiennent les coefficients des fonctions *Energie, Duree* et l'estimation de l'espace de stockage disponible.

Ces méta-données génèrent un surcoût au niveau de l'espace de stockage local de chaque entité mobile, mais également au niveau des messages transmis pour la gestion du groupe et celle de la l'accès aux données.

VII.2.1.1 Gestion de groupe

Le surcoût des méta-données, nécessaires à la gestion du groupe, introduit au niveau de l'espace de stockage local de chaque entité appartenant à un groupe de n entités est donné par l'équation S_G_{local} :
$S\_G_{local} = Comp\_Gp + Struct\_Nom + Profil \times n$.
$Comp\_Gp$ correspond à la composition du groupe, qui est égale à $ID \times n$, où ID correspond à l'identificateur d'une entité mobile. Dans ADHOCFS, l'identificateur est codé sur 4 octets. $Struct\_Nom$, donnée par l'équation :
$Struct\_Nom = \sum_{i=1}^{F}(nom_i + (ID + Etat) \times Local_i + ID_{jeton})$,
correspond à la structure de données qui indique pour chaque fichier nom_i stocké

dans le groupe les identités ID des entités qui en possèdent un réplica local, l'état du réplica ($Etat$) et l'identité de l'entité qui possède le jeton associé (ID_{jeton}). Par ailleurs, F indique le nombre total de fichiers stockés dans le groupe et $Local_i$ désigne le nombre d'entités qui stockent localement le fichier nom_i. $Profil$ correspond aux coefficients des fonctions *Energie*, *Duree* et l'estimation de l'espace de stockage disponible pour chaque entité, dans notre prototype il est égale à $4 \times int$, soit l'équivalent de 16 octets. Ainsi, dans un groupe de n entités qui partagent F fichiers, tel que le fichier i est répliqué sur $Local_i$ entités, le surcoût local des méta-données, nécessaires à la gestion du groupe (S_G_{local}) est égale à : $(20 \times n + \sum_{i=1}^{F}(5 \times Local_i + 16)$ octets).

De plus, S_G_{local} correspond au surcoût du message envoyé par le *leader* du groupe à l'issue de l'étape de l'initialisation du groupe. Cependant, le surcoût du message envoyé par une entités i du groupe correspond au $1/n$ de celui du leader et est égale à : $S\_E_{message} = Comp\_Gp + \sum_{j=1}^{F_i}(nom_j + Etat_j + jeton_j) + Profil$. où $jeton_j$ est un booléen qui indique si l'entité i possède le jeton associé au fichier nom_j. Ainsi, pour une entité i qui partage F_i fichier dans un groupe de n entités, ce surcoût est égale à : ($4 \times n + 14 \times F_i + 16$ octets)

VII.2.1.2 Accès aux données

Le surcoût des méta-données, résultant de la gestion de l'accès aux données, au niveau de l'espace de stockage local de chaque entité appartenant au groupe correspond aux méta-données nécessaires pour la gestion de la cohérence des données et celle de leur disponibilité. Une partie de ce surcoût est comptabilisée dans le surcoût de la gestion du groupe, notamment $Struct\_Nom$ et $Profil$. En plus de ce surcoût, la gestion de la cohérence génère au niveau d'une entité i du groupe le surcoût donné par :
$S\_A_{local} = \sum_{j=1}^{F_i}(ccl_j + Lect_j + Ecr_j)$, où ccl_j correspond au CCL du fichier nom_j, $Lect_j$ correspond à la file d'attente des demandes d'accès en lecture et Ecr_j correspond à la file d'attente des demandes d'accès en écriture. Le CCL attaché à un fichier f est donné par :
$ccl_f = int + \sum_{j=1}^{G}(ID \times local_j + int \times Modif_j))$, où G correspond au nombre des différentes composition de groupe où le fichier a été modifié, $local_j$ désigne le nombre d'entités qui stockent localement ce fichier et $Modif_j$ un entier qui désigne le nombre de bloc du fichier qui ont été modifiés dans le groupe j.

Le surcoût au niveau des messages transmis pour la gestion de la cohérence d'un

fichier est égale à la taille de son CCL, soit ($4+\sum_{j=1}^{G}(4\times local_j+4\times Modif_j)$) octets). Ce qui est négligeable comparer à la taille moyenne des fichiers de données.

VII.2.2 Temps de réponse

Les temps de réponse évalués dans cette section ont été mesurées en exécutant notre prototype sur une plate-forme de dix ordinateurs portables Pentium III à 500 MHz avec un cache mémoire de 256 KB et 200 MB de RAM. Ces ordinateurs sont équipés d'interface réseau sans fil à 11Mbps. Le réseau mobile sans fil sous-jacent est basé sur le protocole Wi-Fi (IEEE 802.11b) en mode *ad hoc*. Dans la suite de cette section, nous évaluons, dans un premier temps, la gestion de groupe puis l'accès aux données.

VII.2.2.1 Gestion de groupe

Pour évaluer le coût de la gestion de groupe, en termes de temps de réponse, nous considérons la création d'un groupe de n entités mobiles, le cas où une entité quitte un groupe de n entités et le cas où une entité se joint à un groupe de n entités. La figure VII.2 donne, en fonction de la taille du groupe, le coût de créer un groupe, quitter un groupe et joindre un groupe, en termes de temps de réponse.

Le coût de la création d'un groupe est mesuré à partir de l'étape de découverte des entités mobiles jusqu'à ce que l'étape de l'initialisation du groupe soit achevée par l'échange des méta-données. Le coût principal de la création d'un groupe est imputé à cette dernière étape, supervisée par le *leader* du groupe. Les n entités mobiles sont dans la portée de communication directe les unes des autres et ne présentent pas de cas de connectivité partielle.

Le coût de la détection qu'une entité a quitté un groupe de n entités, en termes de temps de réponse, est identique à celui de la création d'un groupe de $n-1$ entités. De même, le coût de la détection qu'une entité a joint un groupe de n entités, en termes de temps de réponse, est identique à celui de la création d'un groupe de $n+1$ entités.

Les courbes données dans la figure VII.2 correspondent à des polynômes quadratiques (c.-à-d., de degré 2). Cela signifie que la taille des groupes a une forte incidence sur le temps de réponse, ce qui est justifié par le fait que la taille des messages lors de cette étape est relativement petite. A partir de ces courbes, nous estimons le coût de la création d'un groupe de n entités, en termes de temps de réponse, par l'équation

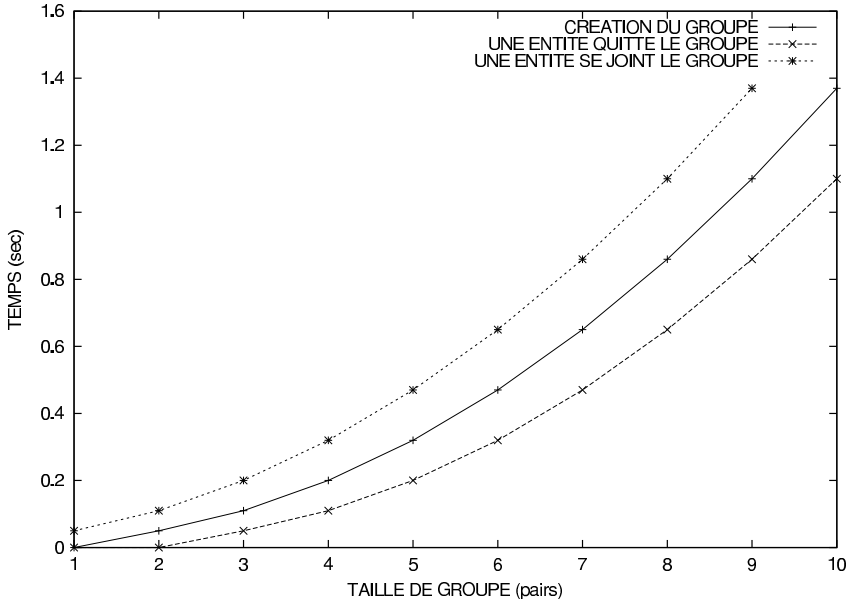

Figure VII.2 – Coût de la gestion de groupe en termes de temps de réponse.

suivante où t représente le temps de réponse en secondes et n la taille du groupe en nombre d'entités :

$$t = 0.01 \times n^2 + 0.02 \times n - 0.03 \quad \text{(VII.1)}$$

VII.2.2.2 Accès aux données

Dans cette section, le temps de réponse correspond à l'intervalle de temps qui s'écoule entre la demande d'accès à un réplica local et l'accès effectif à ce dernier. Nous évaluons le coût de l'accès aux données, en termes de temps de réponse, à travers le coût de l'obtention du jeton et de la notification, le coût de la propagation des mises à jour et l'impact de la propagation paresseuse des mises à jour. Mesurer ces coûts nous permet d'en déduire les coûts des différents accès, comme suit. Le coût de l'obtention du jeton et de la notification correspond au coût d'un accès en écriture à un réplica local dans l'état Read (c.-à-d., à jour) stocké sur une entité qui ne possède pas le jeton associé. Le coût de l'obtention des mises à jour correspond au coût d'un accès en lecture à un réplica local non à jour (c.-à-d., dans l'état Update-Read ou Invalid). Cependant, le coût d'un accès en écriture à un réplica local non à jour, en termes de temps de réponse, correspond à la somme du coût de l'obtention du jeton, en termes de temps de réponse, et celui de la propagation paresseuse des

Groupe (*entités*)	1	2	3	4	5
Temps de réponse (*sec*)	0	0.00949	0.01281	0.01982	0.02129
Groupe (*entités*)	6	7	8	9	10
Temps de réponse (*sec*)	0.02645	0.02977	0.03493	0.03825	0.04341

Table VII.1 – Coût de l'obtention du jeton en termes de temps de réponse.

mises à jour. Par ailleurs, l'accès en lecture à un réplica local à jour ne génère pas de communication, par conséquent son coût, en termes de temps de réponse, est nul.

Le tableau VII.1 donne le coût de l'obtention du jeton (sans les mises à jour) et de la notification, en termes de temps de réponse. Dans un groupe de taille fixe, ce coût est constant. Notons que l'impact d'une entité supplémentaire dans le groupe sur le coût de l'obtention du jeton, en termes de temps de réponse, est de l'ordre de $0.005\ sec$ ce qui est négligeable.

Par ailleurs, la figure VII.3 montre le coût de l'obtention des mises à jour, en termes de temps de réponse. Ce coût est proportionnel à la taille des mises à jour. Suivant notre gestion de l'accès aux données, le coût de l'accès à un réplica local, en termes de temps de réponse, dépend de la taille des mises à jour nécessaires, tandis que la taille du groupe l'affecte faiblement. En effet, ces mises à jour sont propagées par une entité qui stocke un réplica à jour dans le groupe. Dans notre protocole, ces mises à jour sont propagées par le dernier écrivain sur ce réplica.

Dans ADHOCFS, le coût de la gestion de groupe représente un surcoût pour l'accès aux données proprement dit. Cependant, la gestion de groupe permet à toutes les entités d'un groupe d'avoir une vue globale des données stockées dans le groupe. Par conséquent, les demandes de mise à jour ou les demandes de réplicas sont adressées directement à une entités qui possède un réplica à jour qui au lieu d'être adressées à toutes les entités (même à celles qui ne possèdent pas de réplicas) ce qui est le cas sans gestion de groupe. Par ailleurs, dans un groupe, une entité qui demande une mise à jour ou un réplica, reçoit une seul réponse, tandis que sans gestion de groupe elle recevra plusieurs réponses qu'elle devra traiter. Ainsi, le surcoût dû à la gestion de groupe est justifié par la réduction des communications et des calculs lors des accès aux données.

Les figures VII.4 et VII.5 montrent l'avantage de la propagation paresseuse des mises à jour, en termes de temps de réponse. En effet, le temps de réponse lors de la

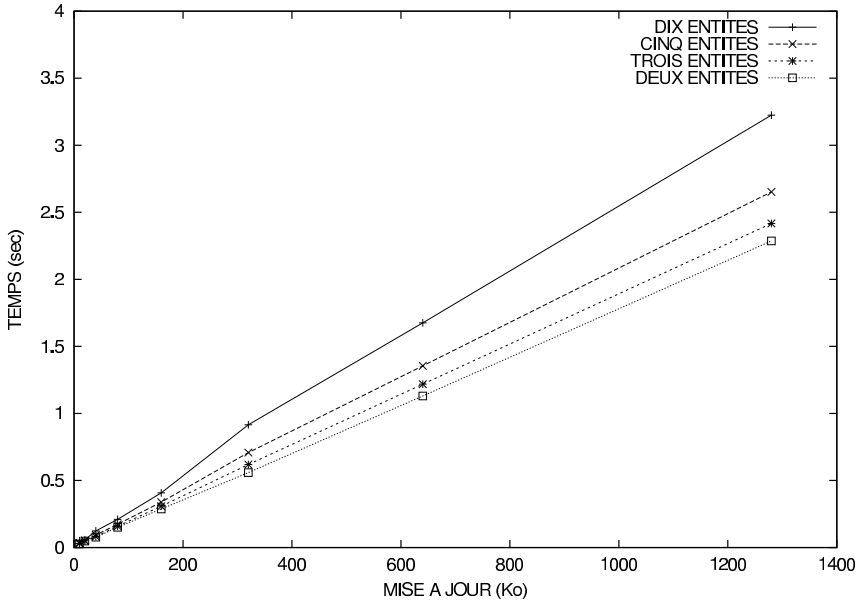

Figure VII.3 – Coût de l'obtention des mises à jour en termes de temps de réponse.

propagation paresseuse des mises à jour varie de $0.0269\ s$ à $3.219\ s$ en fonction de la taille du groupe et des mises à jour (voir figure VII.4). Cependant, le temps de réponse lors de la propagation non paresseuse des mises à jour varie de $0.0361\ s$ à $24.8\ s$ en fonction de la taille du groupe et des mises à jour (voir figure VII.5). Dans l'état de l'art la propagation non paresseuse des mises à jour est utilisée dans les protocoles de gestion de la cohérence faible.

Pour illustrer l'avantage de la propagation paresseuse des mises à jour, considérons le scénario suivant : dans un groupe composé de quatre entités mobiles, chacune ayant un réplica local d'un fichier donné, nous procédons à une opération d'écriture successivement sur chacun des réplicas de sorte à ce que la taille des mises à jour pour chacune de ces opérations soit égale à $640\ Koctets$. Selon notre protocole d'accès aux données, le temps de réponse relatif à la première opération d'écriture est égale à $0.019\ sec$, la deuxième opération d'écriture nécessite $1.20\ sec$, la troisième opération d'écriture nécessite $1.22\ sec$ et la dernière opération d'écriture nécessite $1.27\ sec$, soit $3.709\ sec$ pour les quatre écritures successives. Dans le cas où les mises à jour sont propagées dès qu'elles se produisent, le temps de réponse relatif à une opération d'écriture est égale à $3.639\ sec$, soit $14.56\ sec$ pour les quatre écritures successives.

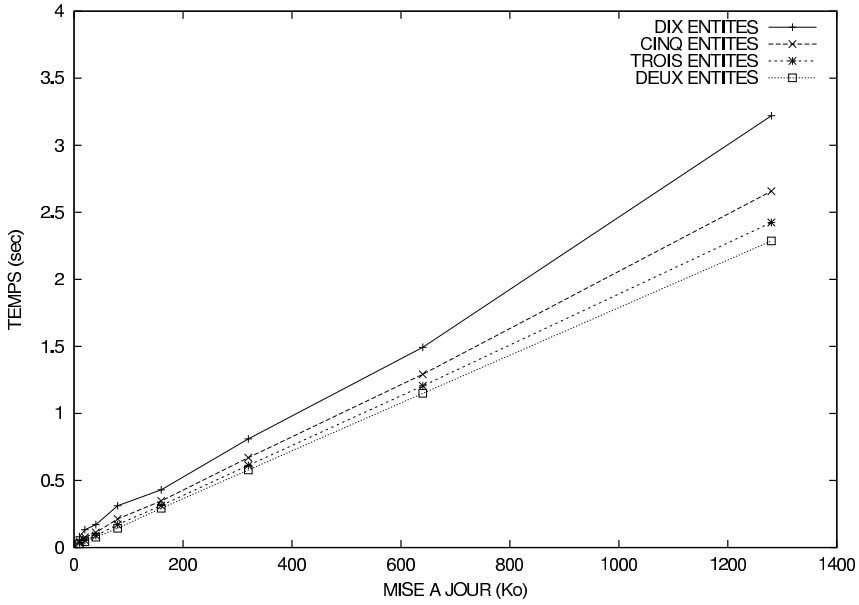

Figure VII.4 – Coût de la propagation paresseuse des mises à jour en termes de temps de réponse.

VII.2.3 Consommation d'énergie

L'un des éléments qui ont guidé notre contribution consiste à diminuer la consommation d'énergie au niveau des terminaux mobiles dont les batteries ont une faible autonomie, en réduisant les échanges de messages. Plusieurs études ont été réalisées sur la consommation d'énergie au niveau d'entités mobile en mode veille [36, 39, 65]. Dans nos mesures nous nous intéressons à l'énergie consommée pour l'envoi de messages *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. Cet état génère une consommation qui s'élève à $741\ mW$ en mode *ad hoc* contre $48\ mW$ en mode infrastructure pour un débit de 11 Mbps dans un réseau basé sur le protocole IEEE 802.11b (c.-à-d., un terminal à l'écoute du support de transmis-

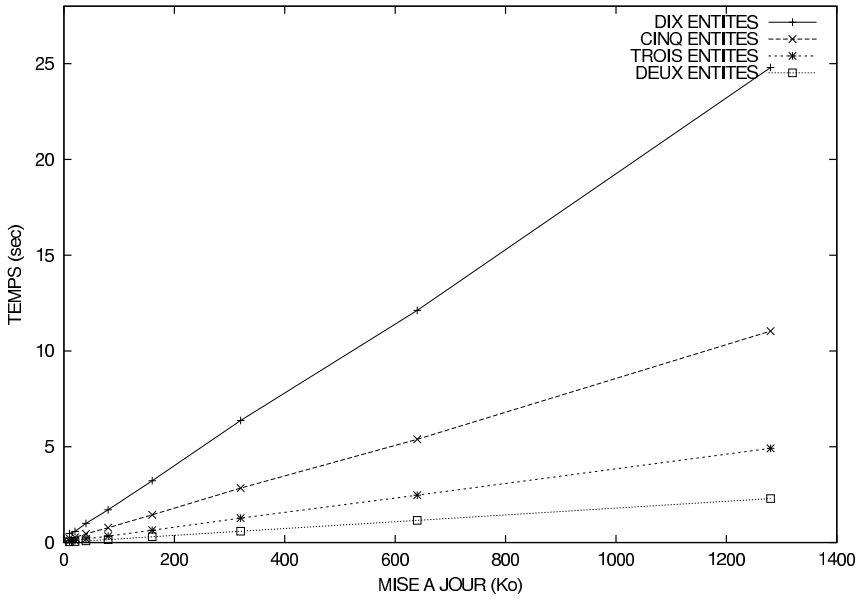

Figure VII.5 – Coût de la propagation non paresseuse des mises à jour en termes de temps de réponse.

sion (*idle*) consomme en mode *ad hoc* $741\ mW.sec$ quand 11 Mb sont échangés).

Notre modèle pour mesurer la consommation de l'énergie au niveau des entités mobiles s'inspire des équations présentées par Feeney et *al.* dans [37]. La consommation de l'énergie est modélisée par l'équation linéaire suivante :

$$Energy = m \times size + b \ \ (\mu W.sec) \tag{VII.2}$$

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 (p.ex., de l'état d'écoute à l'état d'émission). La partie incrémentale de l'équation ($m \times size$) est proportionnelle à la taille, *size*, des paquets émis/reçus. Au sein d'un groupe, l'énergie consommée pour l'envoi d'un message en mode *ad hoc* est égale à la somme des énergies consommées pour émettre le message, le récevoir et l'ignorer par les entités non destinataires. Les constantes m et b dépendent de l'interface sans fil utilisée ainsi que du protocole de transmission. Le tableau VII.2 donne les équations correspondantes à l'énergie consommée pour émettre un message, le récevoir et l'ignorer en utilisant une interface sans fil à 11Mbps de débit et le protocole IEEE 802.11b (Wi-Fi) en mode *ad hoc* [37].

Dans la suite, nous présentons l'évaluation du coût de la gestion de groupe, en

	Emettre	Recevoir	Ignorer
	$\mu W.sec/octet\ \mu W.sec$		
Point à point	$0.48 \times size + 431$	$0.12 \times size + 316$	$0.11 \times size + 66$
Broadcast	$2.1 \times size + 272$	$0.26 \times size + 50$	non définie

Table VII.2 – Consommation d'énergie pour l'envoi d'un message au sein d'un groupe

termes de consommation d'énergie, ainsi que celui de la gestion de l'accès aux données.

VII.2.3.1 Gestion de groupe

Dans cette section, nous évaluons le coût de l'étape de la découverte des entités mobiles, en termes de consommation d'énergie, et celui de l'étape de l'échange des méta-données qui sont les étapes principales dans la création d'un groupe.

Le coût de la découverte des entités mobiles est évalué au niveau d'une entité mobile, puis au niveau du groupe dans sa totalité. Lors de l'étape de découverte, dans un groupe de n entités, chaque entité (sans distinction entre le *leader* du groupe et les autres entités appartenant au groupe) envoie un message de découverte, en mode *broadcast*, et reçoit $(n - 1)$ autres messages de découverte envoyés eux aussi en mode *broadcast*. Ainsi, l'énergie consommée par une entité mobile pendant l'étape de découverte est donnée, d'après le tableau VII.2, par l'équation suivante où n est la taille du groupe et $size$ la taille du message :

$$(1.84 + 0.26 \times n)size + 50 \times n + 222 \ \ (\mu W.sec) \tag{VII.3}$$

La figure VII.6 présente la consommation d'énergie d'une entité lors de l'étape de découverte en fonction de la taille du groupe et celle des messages. Notons que lors de cette étape, pour un seul domaine de sécurité, la taille des messages transmis ne dépasse pas les $256\ octets$.

Par ailleurs, l'énergie consommée par le tout le groupe, pendant l'étape de découverte, est le cumul des énergies consommées par chaque entité appartenant au groupe, comme l'indique l'équation suivante :

$$n \times (1.84 + 0.26 \times n)size + n \times (50 \times n + 222) \ \ (\mu W.sec) \tag{VII.4}$$

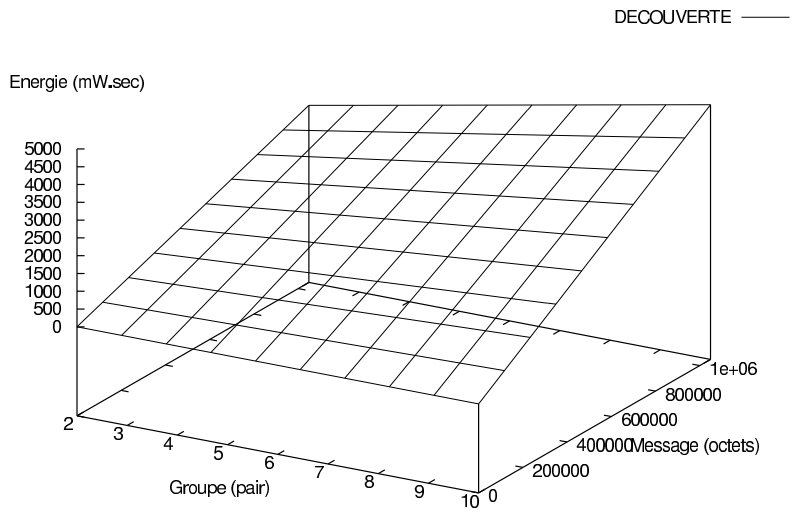

Figure VII.6 – Energie consommée par une entité mobile pendant l'étape de découverte.

La figure VII.7 montre l'énergie consommée par tout le groupe lors de l'étape de découverte en fonction de sa taille et celle des messages échangés. A titre d'exemple, dans un groupe de 10 entités mobiles, l'énergie consommée par une entité pendant l'étape de découverte, pour un seul domaine de sécurité, peut atteindre $1858.64\ \mu W.sec$, tandis qu'au niveau du groupe cette valeur peut atteindre $18586.4\ \mu W.sec$. L'énergie consommée par une entité mobile pendant l'étape de découverte est comparable à l'énergie consommée par une entité dans l'état d'écoute pendant la même durée. En effet, une entité dans l'état d'écoute consomme $741\ mW.sec$ quand 11 Mb sont échangés, soit $0.54\ \mu W.sec/octet$. Ainsi, pendant l'étape de découverte, dans un groupe de 10 entités mobiles, une entité dans l'état d'écoute consomme $1382.4\ \mu W.sec$. Par conséquent, l'étape de découverte génère une consommation d'énergie 1.3 fois plus importante que l'état d'écoute.

Par ailleurs, dans l'étape de l'initialisation du groupe le *leader* supervise l'échange des méta-données. Lors de cette étape, chaque entité envoie au *leader* un message, en mode point à point, contenant ses méta-données et doit ignorer les messages envoyés par les entités paires qui se trouvent dans sa portée de communication, soit $(n-2)$ messages ignorés. Ainsi le *leader* du groupe reçoit $(n-1)$ messages, puis envoie à son tour un message en mode *broadcast* qui contient les méta-données fusionnées

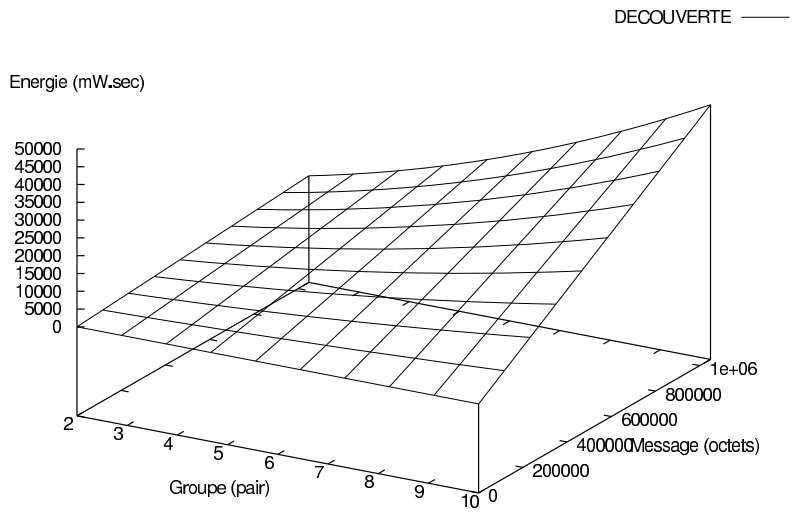

Figure VII.7 – Energie consommée par un groupe pendant l'étape de découverte.

à l'ensemble des entités du groupe. Nous considérons dans nos équations que les tailles des messages ($size$) envoyés par les entités mobiles sont équivalentes et que la taille du message envoyé par le *leader* est égale à ($n \times size$). L'énergie consommée par le *leader* pendant l'étape de l'initialisation est donnée, d'après le tableau VII.2, par l'équation suivante :

$$(2.22 \times n - 0.12)size + 316 \times n - 44 \ \ (\mu W.sec) \tag{VII.5}$$

Par ailleurs, l'énergie consommée par une entité du groupe pendant cette étape est donné par l'équation suivant :

$$(0.26 + 0.37 \times n)size + 66 \times n + 349 \ \ (\mu W.sec) \tag{VII.6}$$

Les figures VII.8 et VII.9 représentent l'énergie consommée par le *leader* du groupe et par une entité appartenant au groupe pour l'échange des méta-données, en fonction de la taille du groupe et celle des messages échangés.

A titre d'exemple, soit un groupe de 10 entité mobiles, si la taille du message émis par une entité du groupe est égale à 256 $octets$ alors la taille du message émis par le *leader* est équivalente 2560 $octets$. Ainsi, lors de l'étape de l'échange des méta-données, le *leader* du groupe consomme environ 8768.48 $\mu W.sec$ alors qu'une autre entité du groupe consomme environ 2022.76 $\mu W.sec$. Par ailleurs, une entité dans

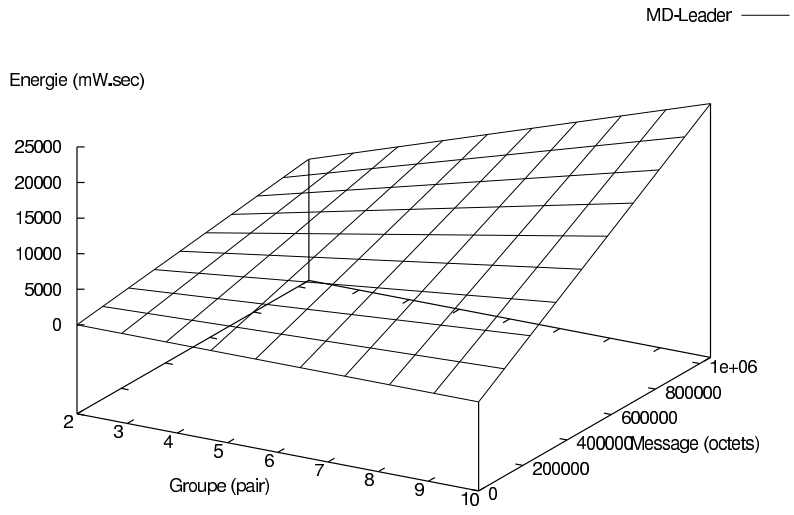

Figure VII.8 – Energie consommée par le *leader* pendant l'échange des méta-données.

l'état d'écoute pendant la durée de l'initialisation de ce groupe consomme $2626.56\ \mu W.sec$, ce qui est comparable à la consommation d'une entité ayant participé à l'initialisation du groupe.

L'énergie consommée par tout le groupe pendant l'échange des méta-données est donné par l'équation suivante, où $size$ est la taille du message émis par une entité du groupe :

$$(0.37 \times n^2 + 2.11 \times n - 0.38)size + 66 \times n^2 + 599 \times n - 393 \ \ (\mu W.sec) \qquad \text{(VII.7)}$$

La figure VII.10 présente l'énergie consommée par un groupe pendant l'étape d'échange des méta-données en fonction de sa taille et celle des messages échangés. Dans l'exemple précédent, la consommation au niveau du groupe s'élève à $26973.32\ \mu W.sec$ où l'énergie consommée par le *leader* représente 32.5%, d'ou l'importance de changer de *leader* périodiquement.

VII.2.3.2 Accès aux données

Dans cette section nous évaluons l'impact de l'accès aux données au sein d'un groupe de n entités mobiles en terme de consommation d'énergie. Ce coût est évalué au niveau du groupe dans sa totalité. Dans la section V.6, nous avons mentionné

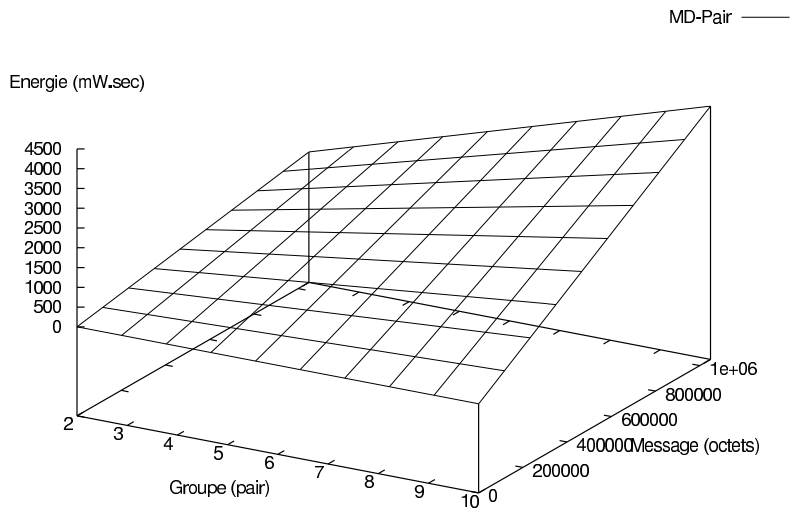

Figure VII.9 – Energie consommée par une entité pendant l'échange des méta-données.

que l'accès en lecture ou en écriture aux données en termes d'échanges de message pouvait varier selon les scénarios d'un nombre minimal de messages à un nombre maximal. Dans notre évaluation, nous tenons compte de ces deux cas extrêmes.

L'énergie consommée lors d'un accès en écriture à un réplica local dépend de son état (Read, UpdateRead, Invalid ou ReadWrite). Dans le cas où l'état du réplica local est Invalid, l'énergie minimale consommée lors de cet accès, qui correspond à l'énergie consommée pour l'obtention des mises à jour si l'identité de l'entité qui détient la version la plus à jour du réplica est connue, est donnée par l'équation suivante :

$$(0.76 + 0.22 \times n)size + 132 \times n + 1230 \ \ (\mu W.sec) \tag{VII.8}$$

Cependant, l'énergie maximale consommée lors de cet accès, qui correspond à l'énergie consommée pour l'obtention des mises à jour si l'identité de l'entité qui détient la version la plus à jour du réplica n'est pas connue, est donnée par l'équation suivante :

$$(0.11 \times n^2 + 0.38 \times n)size + 66 \times n^2 + 615 \times n \ \ (\mu W.sec) \tag{VII.9}$$

Les figures VII.11, VII.12 représentent l'énergie minimale, maximale consommée lors d'un accès en écriture à un réplica local dans l'état Invalid en fonction de la taille du groupe et celle des messages échangés.

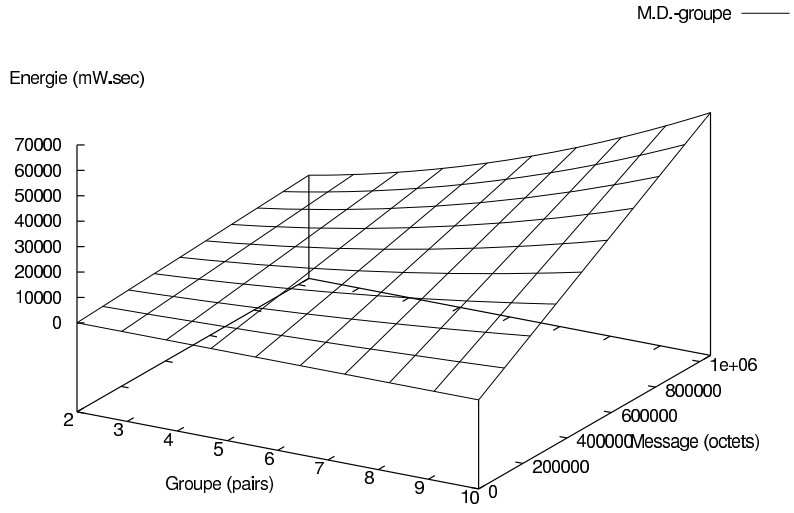

Figure VII.10 – Energie consommée par un groupe pendant l'échange des métadonnées.

Dans le cas où le réplica local est dans l'état Read, l'énergie consommé lors d'un accès en écriture à ce dernier est donnée par l'équation suivante :

$$(1.84 + 0.26 \times n)size + 50 \times n + 222 \ \ (\mu W.sec) \qquad \text{(VII.10)}$$

Cette énergie correspond à l'énergie consommée pour l'obtention du jeton et la notification des entités du groupe qu'un verrou en écriture est posé sur la donnée correspondante au réplica accédé. Par ailleurs, si le réplica local est dans l'état UpdateRead, l'énergie consommé lors d'un tel accès est donnée par l'équation suivante :

$$(2.6 + 0.48 \times n)size + 182 \times n + 1452 \ \ (\mu W.sec) \qquad \text{(VII.11)}$$

Cette énergie correspond à l'énergie consommée pour l'obtention des mises à jour et la notification des entités du groupe qu'un verrou en écriture est posé sur la donnée correspondante au réplica accédé. Les figures VII.13, VII.14 représentent l'énergie minimale (le réplica est dans l'état Read), maximale (le réplica est dans l'état UpdateRead) consommée lors d'un accès en écriture à un réplica local dans l'état Read, UpdateRead en fonction de la taille du groupe et celle des messages échangés.

Dans le cas où le réplica est dans l'état ReadWrite, l'accès en écriture à ce réplica ne génère pas d'échanges de message. Par conséquent, l'énergie consommée lors de cet accès relève uniquement du traitement de cet accès par le système d'exploitation.

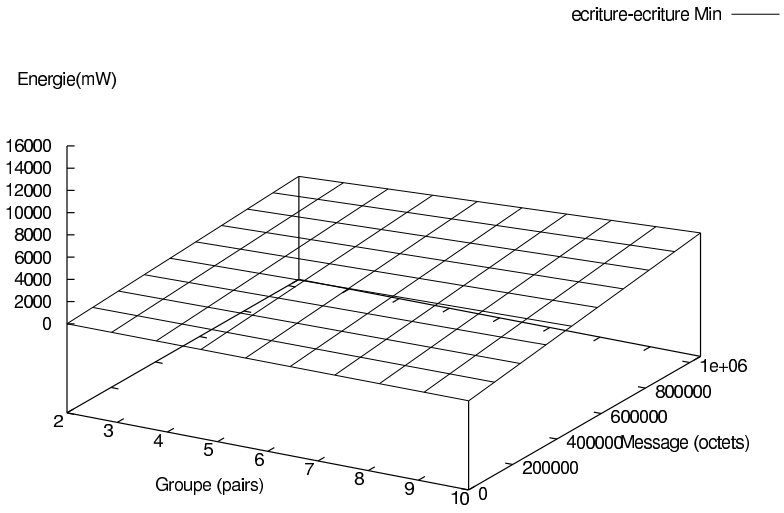

Figure VII.11 – Energie minimale consommée lors d'une écriture précédée par une écriture.

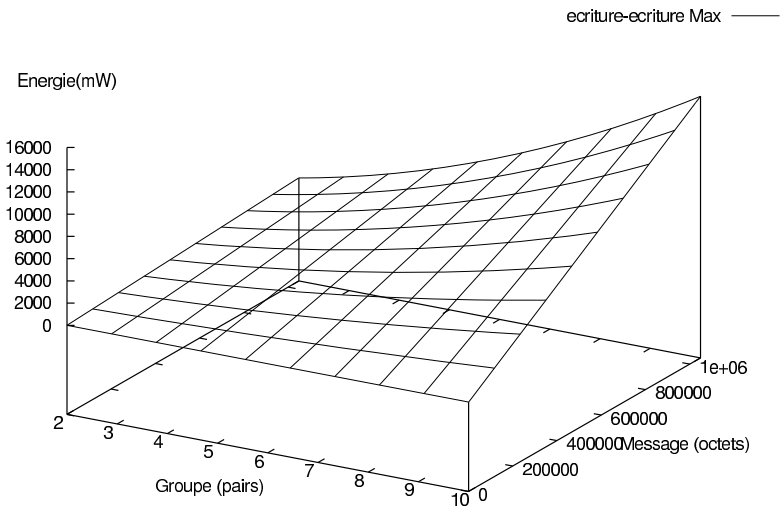

Figure VII.12 – Energie maximale consommée lors d'une écriture précédée par une écriture.

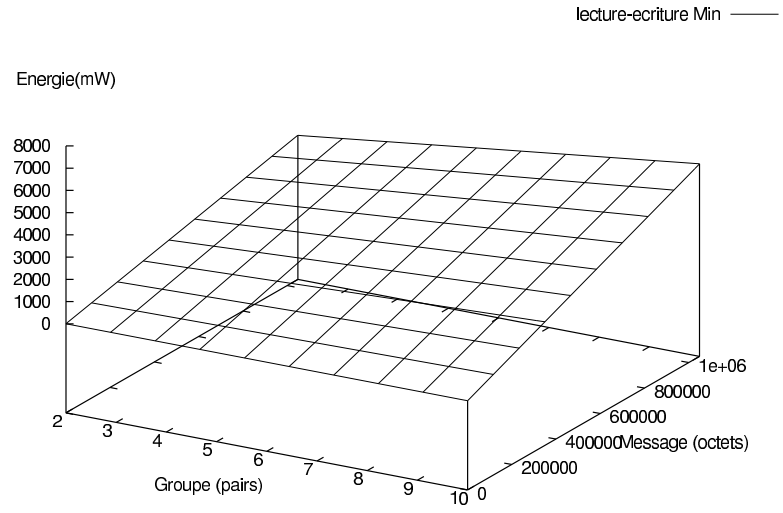

Figure VII.13 – Energie minimale consommée lors d'une écriture précédée par une lecture.

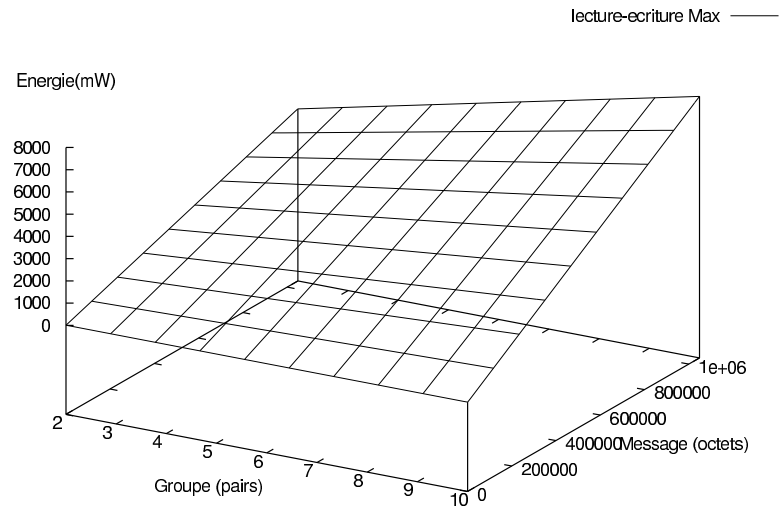

Figure VII.14 – Energie maximale consommée lors d'une écriture précédée par une lecture.

L'énergie consommée lors d'un accès en lecture à un réplica local dépend également de son état. Dans le cas où l'état du réplica local est Invalid, l'énergie minimale consommée lors de cet accès est donnée par l'équation suivante :

$$(2.22 + 0.37 \times n)size + 116 \times n + 837 \ \ (\mu W.sec) \qquad \text{(VII.12)}$$

Cette énergie correspond à l'énergie consommée pour l'obtention des mises à jour si l'identité de l'entité qui détient la version la plus à jour du réplica est connue, et de la notification des entités du groupe qu'un verrou en lecture est posé sur la donnée correspondante au réplica accédé, ce qui permet un accès partagé en lecture à la donnée.

Par ailleurs, l'énergie maximale consommée lors de cet accès est donnée par l'équation suivante :

$$(0.11 \times n^2 + 0.53 \times n + 1.46)size + 66 \times n^2 + 599 \times n - 393 \ \ (\mu W.sec) \qquad \text{(VII.13)}$$

Cette énergie correspond à l'énergie consommée pour l'obtention des mises à jour si l'identité de l'entité qui détient la version la plus à jour du réplica n'est pas connue, et la notification des entités du groupe qu'un verrou en lecture est posé sur la donnée correspondante au réplica. En termes d'échanges de message cet accès génère autant de messages que l'étape de l'échange des méta-données au niveau d'un groupe. Ce qui explique la similarité des graphes correspondants (figures VII.10 et VII.16). Les figures VII.15, VII.16 représentent l'énergie minimale, maximale consommée lors d'un accès en lecture à un réplica local dans l'état Invalid, en fonction de la taille du groupe et celle des messages échangés.

Dans le cas où le réplica local est dans l'état Read, l'accès en lecture à ce réplica ne génère pas d'échanges de message. Par conséquent l'énergie consommée lors de cet accès relève uniquement du traitement de cet accès par le système d'exploitation.

Cependant, l'énergie consommée lors d'un accès en lecture à un réplica dans l'état UpdateRead est donnée par l'équation suivante :

$$(0.76 + 0.22 \times n)size + 132 \times n + 1230 \ \ (\mu W.sec) \qquad \text{(VII.14)}$$

Cette énergie correspond à l'énergie consommée pour l'obtention des mises à jour si l'identité de l'entité qui détient la version la plus à jour du réplica accédé est connue. Elle est identique à celle consommée lors d'un accès en écriture à un réplica local dans l'état Invalid (voir équation VII.8).

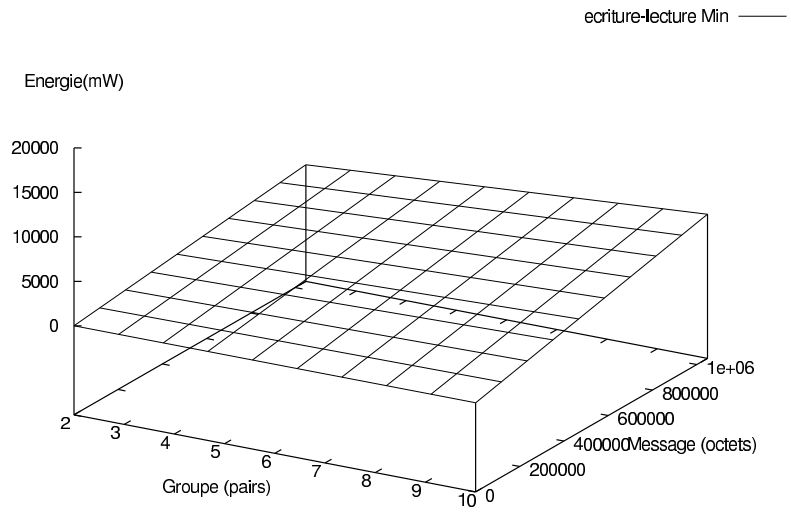

Figure VII.15 – Energie minimale consommée lors d'une lecture précédée par une écriture.

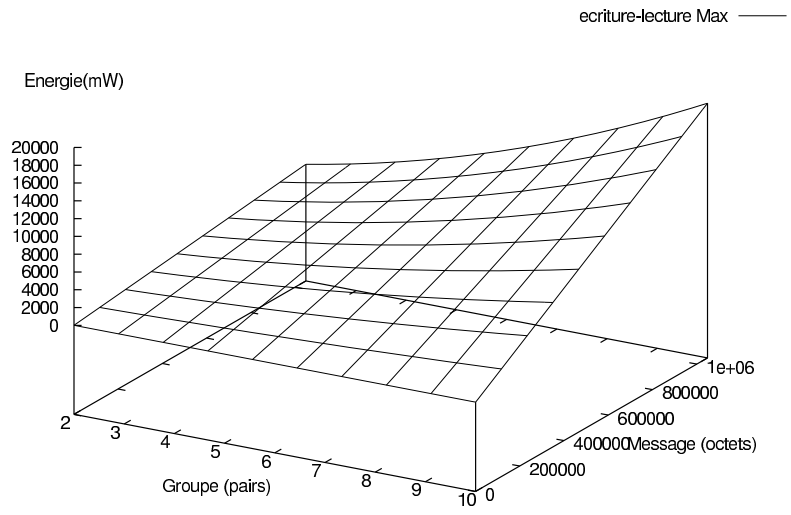

Figure VII.16 – Energie maximale consommée lors d'une lecture précédée par une écriture.

VIII Conclusions

Dans ce chapitre, nous résumons le bilan des environnements mobiles et des fonctionnalités que les systèmes distribués dédiés à ces environnement doivent offrir. Puis, nous résumons notre contribution qui porte sur la gestion du partage des ressources en environnement mobile et en particulier celle des données. Enfin, nous présentons les perspectives de recherche que suscite notre contribution.

VIII.1 Bilan

De nos jours, les environnements mobiles sans fil offrent des contextes d'exécution très variables : présence/absence d'une infrastructure réseau, différents type de réseaux mobiles sans fil, et terminaux mobiles de caractéristiques hétérogènes. Il est donc important d'offrir des solutions adaptées à chaque contexte d'exécution et aux contraintes associées. Ces contraintes sont essentiellement liées à la faible autonomie des terminaux mobiles, aux capacités de stockage réduites de certains d'entre eux et à la connectivité intermittente dans les réseaux mobiles sans fil. Les fonctionnalités du système distribué à adapter concernent en particulier la découverte de l'environnement mobile, le modèle de communication, l'accès aux données, la gestion des ressources et la gestion de la sécurité.

La découverte de l'environnement mobile doit permettre d'identifier automatiquement les ressources accessibles à une entité mobile et qui correspondent à ses besoins, dans des environnements *a priori* inconnus, et ce de façon dynamique et décentralisée. La communication entre entités mobiles doit également être basée sur des modèles décentralisées et non contrainte par l'existence d'un serveur et/ou une infrastructure réseau. Par ailleurs, au niveau réseau, les communications locales sont plus performantes si elles sont directes *via* le mode *ad hoc*, tandis que les communica-

tions qui nécessitent plus d'un saut sont plus performantes si elles reposent sur des stations de base pour relayer les messages. En ce qui concerne la gestion de l'accès aux données, locales ou distantes, elle doit être adaptée à la connectivité du réseau sous-jacent et aux exigences des applications tout en tenant compte de la disponibilité des ressources. De plus, la prolifération des utilisateurs mobiles, et des réseaux mobiles sans fil en mode *ad hoc* promeut le partage des ressources dans des groupes collaboratifs. Les formes de collaboration sont très variées mais elles impliquent toujours un groupe de personnes partageant un intérêt commun. Par ailleurs, le maintien de la sécurité se fait au détriment des performances et des ressources critiques. Il est donc important de déterminer le degré de sécurité qui convient, selon l'utilisation du système et le profil des ordinateurs utilisés.

Aucun des systèmes distribués mobiles existants n'offrent toutes ces fonctionnalités réunies. La découverte de l'environnement mobile est en générale traitée indépendamment des autres fonctionnalités. Les modèles de communication sur lesquels ces systèmes se sont basés varient selon le type de réseau visé. Les modèles centralisés étaient proposés pour des réseaux cellulaires à base d'infrastructure. Les modèles décentralisés, bien qu'ils n'étaient pas initialement destinés aux réseaux en mode *ad hoc*, permettent d'exploiter la communication directe supportée par les réseaux actuels. Tous les systèmes de gestion d'accès aux données en environnements mobiles adoptent un modèle de réplication basé sur une gestion de la cohérence faible où les mises à jour sont effectuées de manière concurrente sur les différents réplicas de la même donnée. Ce choix nécessite l'intégration de mécanisme de détection et de résolution des conflits. La résolution de conflit est soit systématiquement laissés à la charge des développeurs d'applications, soit limité à certains types de conflits. Aucun de ces systèmes ne propose un modèle de réplication qui permet d'anticiper les changements du contexte d'exécution de manière transparente à l'utilisateur afin d'augmenter la disponibilité des données. La notion de collaboration entre différentes entités mobiles pour le partage des ressources, incluant les données, n'est abordée que du point de vue de la collaboration asynchrone qui s'accommode d'une gestion optimiste de la cohérence. Cependant, aucun de ces systèmes n'intègre le support de la collaborationn synchrone qui nécessite une gestion forte de la cohérence. Par ailleurs, aucun des systèmes existants n'a abordé la gestion de l'accès aux données en se souciant de la disponibilité de l'énergie qui est une ressource critique quant il s'agit d'ordinateurs de poche de faible autonomie. Enfin, la sécurité dans les systèmes distribués mobiles reste un véritable défi. Elle est abordée indépendamment

des autres fonctionnalités.

VIII.2 Contributions

Le principal objectif de cette thèse a été de fournir une solution pour le partage des ressources, et en particulier des données, en environnement mobile. Notre solution exploite les différentes connectivités qu'offrent les réseaux sans fil et tient compte des caractéristiques des terminaux mobiles afin de réduire la consommation de leurs ressources critiques. Par ailleurs, notre solution supporte une collaboration synchrone et asynchrone entre les entités mobiles en fonction de la connectivité du réseau sous-jacent et de la proximité géographique.

Dans notre solution, le partage des ressources est basé sur la notion de groupe définie par rapport à un centre d'intérêt partagé par un ensemble d'entités mobiles. La création des groupes, basée sur la découverte décentralisée des entités mobiles, est spontanée et transparente aux utilisateurs. Pour ce faire, nous exploitons les connectivités offertes par les réseaux mobiles et notamment le mode *ad hoc* ainsi que le modèle de communication pair-à-pair. Le mode *ad hoc* nous permet d'exploiter la proximité des entités mobiles qui partagent un intérêt commun et de créer des réseaux locaux. Bien que nous ne traitons pas les aspects de la sécurité, notre gestion de groupe présente une base pour l'implémentation de mécanismes de contrôle des admissions dans le groupe et de calcul de la clé de groupe pour le chiffrement des communications. Nous offrons également la possibilité d'avoir recours à un tiers de confiance à travers la notion de domaine de sécurité.

La gestion de l'accès aux données proprement dit est réalisée, à l'instar des solution existantes, par la gestion de leur cohérence et la gestion de leur réplication. Cependant, nous proposons une gestion hybride de la cohérence en fonction de la connectivité du réseau. Dans un groupe, nous adoptons la gestion de la cohérence forte. En effet, la connectivité du réseau local sous-jacent permet d'établir les communications sans fil nécessaires pour le maintien d'une telle cohérence. De plus, cette gestion de la cohérence offre un support aux applications qui nécessitent une collaboration synchrone. En dehors des groupes ou dans des groupes distincts, les entités mobiles bénéficient d'une gestion faible de la cohérence qui leur permet de manipuler les données de manière indépendante et concurrente, d'où la nécessité de mécanismes de détection et de résolution des conflits lors de la synchronisation des

réplicas. La gestion de la cohérence faible, couplée au mécanisme de détection et de résolution des conflits, offre un support pour les applications nécessitant une collaboration asynchrone, à l'instar des solutions proposées dans la littérature.

Enfin, nous proposons une gestion de la disponibilité des données basée sur une gestion adaptative de la réplication. Outre la réplication des données générée par les accès effectifs, nous augmentons leur disponibilité par une réplication préventive qui permet d'anticiper les déconnexions des entités qui appartiennent à un groupe participant ainsi à une collaboration synchrone. Notre modèle de réplication est basé sur le profil des entités mobiles et permet de minimiser la consommation des ressources critiques (énergie, espace de stockage). Il repose sur un mécanisme d'anticipation de l'indisponibilité des données et un protocole de selection des entités mobiles pour l'accroissement de la disponibilité des données qui favorise la préservation de leurs ressources.

Toutes les fonctionnalités offertes par notre système distribué pour la gestion de l'accès aux données dans des environnements mobiles sans fil tiennent compte de la disponibilité des ressources au niveau des entités mobiles et tentent de réduire la consommation d'énergie en réduisant les échanges de messages *via* les communications sans fil. Nous considérons également que l'intervention de l'utilisateur doit être réduite au maximum et notamment lorsqu'il est possible de prendre en compte les changements du contexte d'exécution de manière transparente.

Nous avons également présenté un prototype de notre système *via* son intégration dans un système de fichiers distribué mobile. Ceci nous a permis de valider notre proposition du point de vue des propriétés non fonctionnelles offertes. Nous montrons notamment que la gestion de groupe présente des temps de réponse proportionnels à la taille du groupe, tandis que les temps de réponse de la gestion d'accès aux données sont proportionnels à la taille des mises à jour. Nous avons également évalué le surcoût mémoire généré par notre système qui est négligeable comparé à la taille moyenne des données. Enfin, l'évaluation de l'énergie consommée a montré l'avantage de notre gestion de groupe et d'accès aux donnée par rapport notamment à la gestion d'accès aux données implémentée dans les systèmes distribués existant. Ce prototype a été en parti réalisé dans le cadre du projet Européen VIVIAN[1] (*Opening Mobile Platforms for the Development of Component-Based Applications*), dont l'objectif était de fournir une plate-forme qui facilite le développement d'applications pour

[1]ITEA 99040, `http://www-nrc.nokia.com/Vivian/index.html`

des ordinateurs de poche en environnement mobile.

VIII.3 Perspectives

Il serait intéressant d'étendre les différentes fonctionnalités proposées pour le partage des ressources dans les WLANs en mode *ad hoc* aux nouveaux/futurs besoins dans le domaine ubiquitaire. La gestion de groupe peut être adaptée à des réseaux *ad hoc* multi-sauts ou à des réseaux hybrides combinant le mode *ad hoc* et le mode infrastructure. Le support de la collaboration synchrone dépend de la proximité géographique. Cependant, il est intéressant de considérer cette adaptation dans le cas où les données sont accédées uniquement en consultation. Par ailleurs, dans la définition de groupe, nous avons considéré la notion de connexion entre deux entités mobiles par rapport à la connectivité du réseau sous-jacent et l'appartenance au même domaine de sécurité. Cette définition peut être étendue pour inclure différents facteurs, notamment la qualité de service ou le coût de la connexion.

Le principe clé de notre approche réside dans l'adaptabilité aux ressources locales des entités mobiles et à la connectivité du réseau. Cette notion d'adaptabilité peut être étendue à tous les niveaux, notamment prendre en compte les profils des entités mobiles pour l'élection du *leader* du groupe. De plus, le profil d'une entité mobile peut inclure d'autres paramètres tels que la diminution de l'intensité du signal d'un message reçu qui permet de détecter l'éloignement d'une entité et par conséquent sa déconnexion du groupe.

L'accroissement de la disponibilité des données basé sur un protocole de réplication préventive peut être amélioré en considérant uniquement les données effectivement accédées dans le groupe. Notre solution ne vise pas un type de données particulier. Cependant, il est intéressant de considérer les spécificités de certains types de données. Par exemple, les données structurées introduisent une dimension sémantique qui peut être exploitée dans l'accroissement de la disponibilité des données de manière plus adaptative et plus flexible. D'un autre côté, notre solution peut être étendue à la gestion des services en environnements mobiles.

Bibliographie

[1] K. Al Agha, G. Pujolle, and G. Vivier. *Réseaux de mobiles et réseau sans fil*. Eyrolles, 2001. *Cité dans la(les) page(s):* 1 , 2

[2] A. Al-Theneyan, P. Mehrotra, and M. Zubair. Enhancing JINI for use across non-multicastable networks. Technical Report NASA/CR-2000-210329 ICASE Report No. 2000-34, Institute for Computer Applications in Science and Engineering Langley Research Center, 2000. *Cité dans la(les) page(s):* 23

[3] P. A. Alsberg and J. D. Day. A principle for resilient sharing of distributed resources. In *Proc. of 2nd International Conference on Software Engineering*, 1976. *Cité dans la(les) page(s):* 28

[4] Y. Amir, Y. Kim, C. Nita-Rotaru, J. Schultz, J. Stanton, and G Tsudik. Exploring robustness in group key agreement. In *Proc. of IEEE 21st International Conference on Distributed Computing Systems (ICDCS)*, 2001. *Cité dans la(les) page(s):* 39

[5] K. Arnold, B. O'Sullivan, R.W. Scheifler, J. Waldo, and A. Wollrath. *The JINI Specification*. Addison Wesley, 1999. *Cité dans la(les) page(s):* 23

[6] D. Barbara and T. Imielinski. Sleepers and workaholics : Caching strategies in mobile environments. In *Proc. of ACM Special Interest Group on Managment of Data (SIGMOD)*, 1994. *Cité dans la(les) page(s):* 99

[7] L. Bartram and M. Blackstock. Designing portable collaborative networks. *Journal of ACM Queue*, May 2003. http ://www.acmqueue.org/. *Cité dans la(les) page(s):* 5 , 18 , 21 , 36

[8] V. Bellotti and S. Bly. Walking away from the desktop computer : Distributed collaboration and mobility in a product design team. In *Proc. of Computer Supported Cooperative Work (CSCW'96)*, 1996. *Cité dans la(les) page(s):* 36

[9] P. A. Bernstein and E. Newcomer, editors. *Principles of Transaction Processing*. Ed. Morgan Kaufmann, 1997. *Cité dans la(les) page(s):* 28

[10] C. Bettstetter and C. Renner. A comparison of service discovery protocols and implementation of the service location protocol. In *Proc. of 6th EUNICE Open European Summer School : Innovative Internet Applications*, 2000. *Cité dans la(les) page(s):* 4

[11] R. Bhaskar. Group key agreement in ad hoc networks. Research Report 4832, INRIA-Rocquencourt, 2003. *Cité dans la(les) page(s):* 42

[12] A. Birrell, R. Levin, R. M. Needham, and M. D. Schroeder. Grapevine : An exercise in distributed computing. *Communication of the ACM*, 25(4), 1982. *Cité dans la(les) page(s):* 30 , 33

[13] G. Blair, G. Coulson, P. Robin, and M. Papathomas. An architecture for next generation middleware. In *Proc. of Middleware*, 1998. *Cité dans la(les) page(s):* 35

[14] E. Bommaiah, A. McAuley, R. R. Talpade, and M. Liu. AMRoute : ad hoc multicast routing protocol. Internet-draft, IETF, 1998. *Cité dans la(les) page(s):* 15

[15] M. Boulkenafed and V. Issarny. Coherency management in ad hoc group communication. In *Proc. of joint VIVIAN-ROBOCOP workshop on Software Infrastructures for Component-Based Applications on Consumer Devices*, 2002. *Cité dans la(les) page(s):* 70

[16] M. Boulkenafed and V. Issarny. Data availability within mobile collaborative ad hoc groups. In *Proc. of 7th CaberNet Radicals Workshop*, 2002. *Cité dans la(les) page(s):* 100

[17] M. Boulkenafed and V. Issarny. ADHOCFS : Sharing files in WLANs. In *Proc. of IEEE 2nd International Symposium on Network Computing and Applications*, 2003. *Cité dans la(les) page(s):* 70

[18] M. Boulkenafed and V. Issarny. A middleware service for mobile ad hoc data sharing, enhancing data availability. In *Proc. of ACM/IFIP/USENIX 4th International Middleware Conference*, 2003. *Cité dans la(les) page(s):* 37 , 100

[19] P. Cao, E. W. Felten, A. Karlin, and K. Li. A study of integrated prefetching and caching strategies. In *Proc. of Measurement and modeling of Computer Systems (SIGMETRICS'95)*, 1995. *Cité dans la(les) page(s):* 33

[20] S. Capkun, J. Hubaux, and L. Buttyan. Mobility helps security in ad hoc networks. In *Proc. of the ACM Symposium on Mobile Ad Hoc Networking and Computing (MobiHOC)*, 2003. *Cité dans la(les) page(s):* 39

[21] L. Capra, W. Emmerich, and C. Mascolo. CARISMA : Context-aware reflective middleware system for mobile applications. *Journal of IEEE Transactions on Software Engineering (TSE)*, November 2003. *Cité dans la(les) page(s):* 35

[22] J.-H. Chang and L. Tassiulas. Energy conserving routing in wireless ad hoc networks. In *Proc. of INFOCOM*, 2000. http ://citeseer.nj.nec.com/chang00energy.html. *Cité dans la(les) page(s):* 53 , 93

[23] G. Chelius and E. Fleury. Ananas : A New Adhoc Network Architectural Scheme. Technical Report 4354, INRIA, 2002. *Cité dans la(les) page(s):* 15

[24] T. Clausen, P. Jacquet, A. Laouiti, P. Muhlethaler, A. Qayyum, and L. Viennot. Optimized link state routing protocol. In *Proc. of IEEE International Multi-topic Conference (INMIC)*, 2001. *Cité dans la(les) page(s):* 15

[25] Salutation Consortium. Salutation architecture overview. White Paper, 1998. http ://www.salutation.org/whitepaper/originalwp.pdf. *Cité dans la(les) page(s):* 24

[26] Intel Corp. It mobility road map : Past, present, and future plans, 2002. http ://www.intel.com/. *Cité dans la(les) page(s):* 36

[27] Microsoft Corp. Understanding universal plug and play. White Paper, 2000. *Cité dans la(les) page(s):* 24

[28] Microsoft Corp. *Universal Plug and Play Device Architecture*, June 2000. Version 1.0. *Cité dans la(les) page(s):* 24

[29] G. F. Coulouris, J. Dollimore, and T. Kindberg. *Distributed systems : concepts and design*. Addison-Wesley International computer science series, 1994. *Cité dans la(les) page(s):* 4

[30] A. J. Demers, D. Greene, C. Hauser, W. Irish, J. Larson, S. Shenker, H. Sturgis, D. Swinehart, and D. Terry. Epidemic algorithms for replicated data base maintenance. In *Proc. of 6th Symposium on Principles of Distributed Computing*, 1987. *Cité dans la(les) page(s):* 31

[31] A. J. Demers, K. Petersen, M. J. Spreitzer, D. B. Terry, M. M. Theimer, and B. B. Welch. The bayou architecture : Support for data sharing among mobile users. In *Proc. of IEEE Workshop on Mobile Computing Systems & Applications*, 1994. http ://citeseer.nj.nec.com/demers94bayou.html. *Cité dans la(les) page(s):* 26 , 33 , 89

[32] W. K. Edwards, E. D. Mynatt, K. Petersen, M. J. Spreitzer, D. B. Terry, and M. M. Theimer. Designing and implementing asynchronous collaborative applica-

tions with bayou. In *Proc. of ACM Symposium on User Interface Software and Technology*, 1997. *Cité dans la(les) page(s):* 33 , 36 , 90

[33] T. Ekenstam, C. Matheny, P. L. Reiher, and G. J. Popek. The bengal database replication system. *Journal of Distributed and Parallel Databases*, 9(3), 2001. http ://citeseer.nj.nec.com/449358.html. *Cité dans la(les) page(s):* 26 , 30 , 34 , 89

[34] F. Fabret, H. A. Jacobsen, F. Llirbat, J. Pereira, K. A. Ross, and D. Shasha. Filtering algorithms and implementation for very fast publish/subscribe systems. *Journal of SIGMOD Record (ACM Special Interest Group on Management of Data)*, 30(2), 2001. *Cité dans la(les) page(s):* 17

[35] F. Fabret, F. Llirbat, J. Pereira, and D. Shasha. Efficient matching for content-based publish/subscribe systems. Technical Report 4089, INRIA, 2000. http ://wwwcaravel.inria.fr/pereira/matching.ps. *Cité dans la(les) page(s):* 17

[36] K. I. Farkas, J. Flinn, G. Back, D. Grunwald, and J. Anderson. Quantifying the energy consumption of a pocket computer and a java virtual machine. In *Proc. of the International Conference on Measurement and Modeling of Computer Systems (ACM SIGMETRICS)*, 2000. *Cité dans la(les) page(s):* 124

[37] L. Feeney and M. Nilsson. Investigating the energy consumption of a wireless network interface in an ad hoc networking environment. In *Proc. of IEEE INFOCOM*, 2001. http ://citeseer.nj.nec.com/feeney01investigating.html. *Cité dans la(les) page(s):* 53 , 93 , 103 , 125

[38] A.L. Murphy G.-C. Roman, G.P. Picco. Software engineering for mobility : A roadmap. In *Proc. of 22nd International Conference on Software Engineering*, 2000. http ://www.cs.rochester.edu/u/murphy/papers/. *Cité dans la(les) page(s):* 9

[39] P. Gauthier, D. Harada, and M. Stemm. Reducing power consumption for the next generation of PDAs : It's in the network interface. In *Proc. of International Workshop on Mobile Multimedia Communications (MoMuC)*, 1996. *Cité dans la(les) page(s):* 124

[40] Bluetooth Special Interest Group. *Bluetooth Specification Part E. Service Discovery Protocol (SDP)*, November 1999. http ://www.bluetooth.com. *Cité dans la(les) page(s):* 24

[41] E. Guttman. Service location protocol : Automatic discovery of IP network services. *Journal of IEEE Internet Computing*, July 1999. *Cité dans la(les) page(s):* 23

[42] E. Guttman, C. Perkins, and J. Kempf. Service templates and service : Schemes. Technical Report RFC 2609, IETF, june 1999. *Cité dans la(les) page(s):* 23 , 116

[43] R. G. Guy. Ficus : A very large scale reliable distributed file system. Technical Report CSD-910018, UCLA Computer Science Department, 1991. http ://citeseer.nj.nec.com/guy91ficu.html. *Cité dans la(les) page(s):* 4 , 26 , 33

[44] O. Hanssen and F. Eliassen. A framework for policy bindings. In *Proc. of IEEE International Symposium on Distributed Objects and Applications*, 1999. *Cité dans la(les) page(s):* 35

[45] T. Hara. Effective replica allocation in ad hoc networks for improving data accessibility. In *Proc. of INFOCOM*, 2001. *Cité dans la(les) page(s):* 34 , 99

[46] J. Heidemann, T. Page, R. Guy, and G Popek. Primarly disconnected operation : Experience with Ficus. In *Proc. of 2nd Workshop on Management of Replicated Data*, 1992. *Cité dans la(les) page(s):* 30 , 89

[47] A. Helal, A. Heddaya, and B. Bhar, editors. *Replication Techniques in Distributed Systems*. Kluwer Academic Publishers, 1996. *Cité dans la(les) page(s):* 28

[48] Y. Huang, P. Sistla, and O. Wolfson. Data replication for mobile computers. In *Proc. of ACM Special Interest Group on Management of Data (SIGMOD)*, 1994. *Cité dans la(les) page(s):* 99

[49] IETF. Multicast security working group. http ://www.securemulticast.org/msec-index.htm. *Cité dans la(les) page(s):* 39

[50] IRTF. Group security research group. http ://www.securemulticast.org/gsec-index.htm. *Cité dans la(les) page(s):* 39

[51] P. Jacquet, A. Laouiti, P. Minet, and L. Viennot. Performance of multipoint relaying in ad hoc mobile routing protocols. In *Proc. of Networking*, 2002. *Cité dans la(les) page(s):* 15

[52] M. Jiang, J. Li, and Y. C. Tay. Cluster based routing protocol CBRP functional specification. IETF Internet Draft, August 1998. http ://www.ietf.org/proceedings/98dec/I-D/draft-ietf-manet-cbrp-spec-00.txt. *Cité dans la(les) page(s):* 15

[53] J. Jing, A. S. Helal, and A. Elmagarmid. Client-server computing in mobile environments. *Journal of ACM Computing Surveys*, 31(2), 1999. *Cité dans la(les) page(s):* 17

[54] D.B. Johnson and D.A. Maltz. *Dynamic Source Routing in Ad Hoc Wireless Networks in Mobile Computing*, chapter 5. Kluwer Academic Publisher, 1996. *Cité dans la(les) page(s):* 14

[55] A.D. Joseph, J. A. Tauber, and M. F. Kaashoek. Mobile computing with the Rover toolkit. *Journal of IEEE Transactions on Computers*, 46(3), 1997. *Cité dans la(les) page(s):* 26 , 29

[56] T. Karygiannis and L. Owens. Wireless network security : 802.11, bluetooth and handheld devices. Technical Report SP 800-48, National Institute of Standards and Technology (NIST), 2002. *Cité dans la(les) page(s):* 12

[57] M. L. Kazar. Synchronization and caching issues in the Andrew File System. In *Proc. of USENIX Winter Technical Conference*, 1988. *Cité dans la(les) page(s):* 31

[58] J. Kempf and J. Goldschmidt. Mobile code for network service access. In *Proc. of INET*, 2000. *Cité dans la(les) page(s):* 23

[59] A.-M. Kermarrec, A. Rowstron, M. Shapiro, and P. Druschel. The IceCube approach to the reconciliation of divergent replicas. *Proc. of ACM 20th Symposium on Principles of Distributed Computing (PODC)*, 2001. *Cité dans la(les) page(s):* 41 , 90

[60] Y. Kim, A. Perring, and G. Tsudik. Simple and fault-tolerant key agreement for dynamic collaborative groups. In *Proc. of ACM Conference on Computer and Communications Security (CCS)*, 2000. *Cité dans la(les) page(s):* 39

[61] J. J. Kistler. Increasing file system availability through second-class replication. In *Proc. of Workshop on the Management of Replicated Data*, 1990. http ://citeseer.nj.nec.com/172126.html. *Cité dans la(les) page(s):* 4 , 32

[62] J. J. Kistler and M. Satyanarayanan. Disconnected operation in the Coda File System. *Journal of ACM Transactions on Computer Systems (TOCS)*, 10(1), 1992. *Cité dans la(les) page(s):* 4 , 32

[63] L. Kleinrock. Nomadicity : Anytime, anywhere in a disconnected world. *Journal of Mobile Networks and Applications*, 1(4), 1997. *Cité dans la(les) page(s):* 9

[64] F. Kon, M. Roman, P. Liu, J. Mao, T. Yamane, L.M. Aes, and R. Cambpell. Monotoring, security, and dynamic configuration with the dynamicTAO reflective ORB. In *Proc. of International Conference on Distributed Systems Platforms and Open Distributed Processing*, 2000. *Cité dans la(les) page(s):* 35

[65] R. Kravets and P. Krishnan. Power management techniques for mobile communication. In *Proc. of MobiCom*, 1998. *Cité dans la(les) page(s):* 124

[66] J. Kubiatowicz, D. Bindel, Y. Chen, P. Eaton, D. Geels, R. Gummadi, S. Rhea, H. Weatherspoon, W. Weimer, C. Wells, and B. Zhao. Oceanstore : An architecture for global-scale persistent storage. In *Proc. of ACM ASPLOS*, 2000. http ://citeseer.nj.nec.com/kubiatowicz00oceanstore.html. *Cité dans la(les) page(s):* 31 , 33

[67] G. Kuenning, W. Ma, P. Reiher, and G. Popek. Simplifying automated hoarding methods. In *Proc. of ACM 5th International Workshop on Modeling, Analysis and Simulation of Wireless and Mobile Systems*, 2002. *Cité dans la(les) page(s):* 33

[68] G. H. Kuenning. *Seer : Predictive File Hoarding for Disconnected Mobile Operation*. PhD thesis, University of California, Los Angeles, May 1997. *Cité dans la(les) page(s):* 33 , 111

[69] G. H. Kuenning and G. J. Popek. Automated hoarding for mobile computers. In *Proc. of Symposium on Operating Systems Principles*, 1997. *Cité dans la(les) page(s):* 33 , 111

[70] P. Kumar. Coping with conflicts in an optimistically replicated file system. In *Proc. of IEEE Workshop on Management of Replicated Data*, 1990. *Cité dans la(les) page(s):* 89

[71] P. Kumar and M. Satyanarayanan. Log-based directory resolution in the coda file system. In *Proc. of 2nd International Conference on Parallel and Distributed Information Systems*, 1993. *Cité dans la(les) page(s):* 89

[72] P. Kumar and M. Satyanarayanan. Supporting application-specific resolution in an optimistically replicated file system. In *Proc. of IEEE 4th Workshop on Workstation Operating Systems*, 1993. *Cité dans la(les) page(s):* 29 , 89

[73] T. Ledoux. OpenCorba : a Reflective Open Broker. In *Proc. of Reflection*, 1999. *Cité dans la(les) page(s):* 35

[74] X.-Y. Li and P.-J. Wan. Constructing minimum energy mobile wireless networks. In *Proc. of ACM Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc)*, 2001. http ://citeseer.nj.nec.com/li01constructing.html. *Cité dans la(les) page(s):* 19

[75] Y. Li, J. Li, K. Yu, K. Wang, S. Li, and Y.-Q. Zhang. A peer-to-peer communication system. In *Proc. of IEEE Pacific Rim Conference on Multimedia*, 2002. *Cité dans la(les) page(s):* 17

[76] D. Margulius. Collaborative challenges. *Journal of InfoWorld*, july 2002. http ://www.infoworld.com. *Cité dans la(les) page(s):* 36

[77] C. Mascolo, L. Capra, and W. Emmerich. Middleware for mobile computing (a survey). Advanced Lectures in Networking, 2002. Editors E. Gregori, G. Anastasi, S. Basagni. Springer. LNCS 2497. *Cité dans la(les) page(s):* 4

[78] C. Mascolo, L. Capra, S. Zachariadis, and W. Emmerich. XMIDDLE : A data-sharing middleware for mobile computing. *Personal and Wireless Communications Journal*, 21(!), 2002. *Cité dans la(les) page(s):* 4 , 26 , 31 , 33

[79] P. McDaniel, A. Prakash, and P. Honeyman. Antigone : A flexible framework for secure group communication. In *Proc. of 8th USENIX Security Symposium*, 1999. *Cité dans la(les) page(s):* 39

[80] B. McFarland and M. Wong. The family dynamics of 802.11. *Journal of ACM Queue*, May 2003. http ://www.acmqueue.org/. *Cité dans la(les) page(s):* 3 , 11

[81] D. Mentré, M. Boulkenafed, and V. Issarny. ADHOCFS : A serverless file system for mobile users. Research Report 4303, INRIA-Rocquencourt, 2001. *Cité dans la(les) page(s):* 45

[82] D. Mentré, M. Boulkenafed, and V. Issarny. Towards a network file system for roaming users. In *Proc. of CaberNet Workshop*, 2001. *Cité dans la(les) page(s):* 45

[83] Sun Microsystems. Technical white paper : Jini architectural overview. Technical report, Sun Microsystems, 1999. http ://www.sun.com/jini/. *Cité dans la(les) page(s):* 23

[84] J. H. Morris, M. Satyanarayanan, M. H. Conner, J. H. Howard, D. S. H. Rosenthal, and F. Donelson Smith. Andrew : a distributed personal computing environment. *Journal of Communication of the ACM*, 29(3), 1986. *Cité dans la(les) page(s):* 25

[85] S. J. Mullender. *Distributed systems*. Addison-Wesley ACM press frontier series, 1994. *Cité dans la(les) page(s):* 4

[86] A. L. Murphy, G. P. Picco, and G.-C. Roman. Lime : A middleware for physical and logical mobility. In *Proc. of 21st International Conference on Distributed Computing Systems (ICDCS)*, 2001. *Cité dans la(les) page(s):* 26 , 39

[87] S. Murthy and J.J. Garcia-Luna-Aceves. A routing protocol for packet radio networks. In *Proc. of ACM MobiCom*, 1995. *Cité dans la(les) page(s):* 15

[88] S. Murthy and J.J. Garcia-Luna-Aceves. An efficient routing protocol for wireless networks. *Journal of ACM Mobile Networks and Applications Journal, Special issue on Routing in Mobile Communication Networks*, 1(2), october 1996. *Cité dans la(les) page(s):* 15

[89] B. D. Noble, M. Price, and M. Satyanarayanan. A programming interface for application-aware adaptation in mobile computing. In *Proc. of 2nd USENIX Symposium on Mobile and Location-Independent Computing*, 1995. *Cité dans la(les) page(s):* 26

[90] B. D. Noble, M. Satyanarayanan, D. Narayanan, J. E. Tilton, J. Flinn, and K. Walker. Agil application-aware adaptation for mobility. *Journal of ACM SIGOPS Operating Systeme Review*, 31(5), 1997. *Cité dans la(les) page(s):* 26

[91] S. Oaks and H. Wong. *JINI in a Nutshell*. O'Reilly, 2000. *Cité dans la(les) page(s):* 23

[92] T. W. Page, R. G. Guy, G. J. Popek, J. S. Heidemann, W. Mak, and D. Rothmeier. Management of replicated volume location data in the Ficus Replicated File System. In *Proc. of USENIX Conference*, 1991. *Cité dans la(les) page(s):* 4 , 33

[93] M. Papadopouli and H. Schulzrinne. Seven degrees of separation in mobile ad hoc networks. In *Proc. of IEEE Global Telecommunications Conference (GLOBECOM)*, 2000. http ://citeseer.nj.nec.com/papadopouli00seven.html. *Cité dans la(les) page(s):* 34

[94] M. Papadopouli and H. Schulzrinne. A performance analysis of 7DS a peer-to-peer data dissemination and prefetching tool for mobile users. In *Advances in wired and wireless communications, IEEE Sarnoff Symposium Digest*, 2001. *Cité dans la(les) page(s):* 34

[95] V.D. Park and M.S. Corson. A highly adaptive distributed routing algorithm for mobile wireless networks. In *Proc. of IEEE INFOCOM*, 1997. *Cité dans la(les) page(s):* 14

[96] V.D. Park and M.S. Corson. Temporally-ordered routing algorithm (TORA) version 1 functional specification. Technical report, IETF MANET Internet Draft, 1997. draft-ietf-manet-tora-spec-00.txt. *Cité dans la(les) page(s):* 14

[97] M.R. Pearlman and Z.J. Haas. Determining the optimal configuration for the Zone Routing Protocol. *Journal of IEEE on Selected Areas in Communications, special Issue on Wireless Ad Hoc Networks*, 17(8), 1999. *Cité dans la(les) page(s):* 15

[98] C.E. Perkins and P. Bhagwat. Highly dynamic destination-sequenced distance-vector routing DSDV for mobile computers. In *Proc. of ACM SIGCOMM Conference on Communications Architectures, Protocols and Applications*, 1994. *Cité dans la(les) page(s):* 15

[99] C.E. Perkins and E.M. Royer. Ad hoc on-demand distance vector routing. In *Proc. of IEEE 2nd Workshop on Mobile Computing Systems and Applications*, 1999. *Cité dans la(les) page(s):* 14

[100] P. J. Perrone and V. S.R.Krishna Chaganti. Jini in the box. *Journal of Embedded Systems Programming*, 12(12), 1999. *Cité dans la(les) page(s):* 23

[101] M. Perry, K. O'Hara, A. Sellen, B. Brown, and R. Harper. Dealing with mobility : Understanding access anytime, anywhere. *Journal of ACM Transactions on Computer Human Interaction (TOCHI)*, 8(4), 2001. *Cité dans la(les) page(s):* 36

[102] K. Petersen, M. J. Spreitzer, D. B. Terry, M. M. Theimer, and A. J. Demers. Flexible update propagation for weakly consistent replication. In *Proc. of Symposium on Operating Systems Principles (SOSP)*, 1997. *Cité dans la(les) page(s):* 30 , 89 , 90

[103] G. Picco, A. Murphy, and G.-C. Roman. Lime : Linda meets mobility. In *Proc. of ACM 21th International Conference on Software Engineering*, 1999. *Cité dans la(les) page(s):* 9 , 26 , 39 , 47

[104] T. Plagemann, F. Eliassen, V. Goebel, T. Kristensen, and H. Rafaelsen. Adaptive QoS aware binding of persistent objects. In *Proc. of IEEE International Symposium on Distributed Objects and Applications*, 1999. *Cité dans la(les) page(s):* 35

[105] Universal Plug and Play Forum. Universal plug and play device architecture, March 2000. Version 0.91. *Cité dans la(les) page(s):* 24

[106] G. Popek, B. Walker, J. Chow, D. Edwards, C. Kline, G. Rudisin, and G. Thiel. Locus : A network transparent, high reliability distributd system. In *Proc. of ACM 8th Symposium on Operating Systems Principles*, 1981. *Cité dans la(les) page(s):* 30

[107] D. Ratner, P. Reiher, and G. Popek. Roam : A scalable replication system for mobile computing. In *Proc. of Workshop on Mobile Databases and Distributed Systems (MDDS)*, 1999. *Cité dans la(les) page(s):* 4 , 26 , 30

[108] D. Ratner, P. Reiher, and G. J. Popek. Dynamic version vector maintenance. Technical Report CSD-970022, UCLA Department of Computer Science, 1997. http ://citeseer.nj.nec.com/ratner97dynamic.html. *Cité dans la(les) page(s):* 30

[109] D. H. Ratner. *Roam : A Scalable Replication System for Mobile and Distributed Computing*. PhD thesis, University of California, UCLA, January 1998. *Cité dans la(les) page(s):* 4 , 26 , 30

[110] P. Reiher, J. S. Heidemann, D. Ratner, G. Skinner, and G. J. Popek. Resolving file conflicts in the ficus file system. In *Proc. of USENIX Conference*. USENI, 1994. *Cité dans la(les) page(s):* 89

[111] P. Reiher, J. Popek, M. Gunter, J. Salomone, and D. Ratner. Peer-to-peer reconciliation based replication for mobile computers. In *Proc. of European Conference on Object Oriented Programming 2nd Workshop on Mobility and Replication*, 1996. http ://citeseer.nj.nec.com/5097.html. *Cité dans la(les) page(s):* 26 , 30

[112] M. W. Ritter. The future of WLAN. *Journal of ACM Queue*, may 2003. http ://www.acmqueue.org/. *Cité dans la(les) page(s):* 3 , 11

[113] V. Rodoplu and T. H. Meng. Minimum energy mobile wireless networks. In *Proc. of IEEE International Conference on Communications (ICC)*, 1998. http ://citeseer.nj.nec.com/rodoplu98minimum.html. *Cité dans la(les) page(s):* 19

[114] G.-C. Roman, Q. Huang, and A. Hazemi. Consistent group membership in ad hoc networks. In *Proc. of 23rd International Conference on Software Engineering (ICSE)*, 2001. *Cité dans la(les) page(s):* 68

[115] E. Royer and C.E. Perkins. Multicast operation of ad hoc on-demand distance vector routing protocol. In *Proc. of ACM/IEEE MobiCom*, 1999. *Cité dans la(les) page(s):* 15

[116] S. Sanborn and C. Moore. Collaboration comes together. *Journal of InfoWorld*, 2001. http ://archive.infoworld.com/. *Cité dans la(les) page(s):* 36

[117] T. Saridakis. Nomadic collaboration management. In *Proc. of 4th International System Administration and Networking Conference (SANE)*, 2002. *Cité dans la(les) page(s):* 36

[118] M. Satyanarayanan. Accessing information on demand at any location : mobile information access. *Journal of IEEE Personal Communication*, 3(1), 1996. *Cité dans la(les) page(s):* 4 , 17 , 26

[119] M. Satyanarayanan, J. J. Kistler, P. Kumar, M. E. Okasaki, E. H. Siegel, and D. C. Steere. Coda : A highly available file system for a distributed workstation environment. *Journal of IEEE Transactions on Computers*, 39(4), 1990. http ://citeseer.nj.nec.com/satyanarayanan90coda.html. *Cité dans la(les) page(s):* 25

[120] P. Sinha, R. Sivakumar, and V. Bharghavan. Mcedar : Multicast core-extraction distributed ad hoc routing. In *Proc. of Wireless Communication and Networking Conference*, 1999. *Cité dans la(les) page(s):* 15

[121] D. Terry, A. Demers, K. Petersen, M. Spreitzer, M. Theimer, and B. Welch. Session guarantees for weakly consistent replicated data. In *Proc. of 3rd International Conference on Parallel and Distributed Information Systems (PDIS)*, 1994. *Cité dans la(les) page(s):* 89

[122] D. B. Terry, K. Petersen, M. J. Spreitzer, and M. M. Theimer. The case for non-transparent replication : Examples from Bayou. In *Proc. of IEEE International Conference on Data Engineering*, 1998. *Cité dans la(les) page(s):* 89

[123] D. B. Terry, M. M. Theimer, K. Petersen A. J. Demers, M. J. Spreitzer, and C. H. Hauser. Managing update conflicts in a weakly connected replicated storage system. In *Proc. of 5th ACM Symposium on Operating Systems Principles*, 1995. *Cité dans la(les) page(s):* 89

[124] M. M. Theimer, A. J. Demers, K. Petersen, M. J. Spreitzer, D. B. Terry, and B. B. Welch. Dealing with tentative data values in disconnected work groups. In *Proc. of Workshop on Mobile Computing Systems and Applications*, 1994. *Cité dans la(les) page(s):* 89

[125] B. Walker, G. Popek, R. English, C. Kline, and G. Thiel. The locus distributd operating system. In *Proc. of ACM 9th Symposium on Operating Systems Principles*, 1983. *Cité dans la(les) page(s):* 30

[126] O. Wolfson, S. Jajodia, and Y. Huang. An adaptive data replication algorithm. *Journal of ACM Transactions on Database Systems*, 22(2), 1997. http ://citeseer.nj.nec.com/wolfson97adaptive.html. *Cité dans la(les) page(s):* 34

[127] C.W. Wu and Y.C. Tay. AMRIS : A multicast protocol for ad hoc wireless networks. In *Proc. of IEEE Military Communications Conference (MILCOM)*, 1999. *Cité dans la(les) page(s):* 15

[128] S. Yi and R. Kravets. Key management for heterogeneous ad hoc wireless networks. Technical Report UIUCDCS-R-2002-2290, UIUC, 2002. *Cité dans la(les) page(s):* 39

[129] H. Yu and A. Vahdat. Design and evaluation of a continuous consistency model for replicated services. In *Proc. of 4rd Symposium on Operating Systems Design and Implementation*, 2000. http ://citeseer.nj.nec.com/yu00design.html. *Cité dans la(les) page(s):* 33

[130] W. Zhao, H. Schulzrinne, and E. Guttman. mSLP- mesh enhanced service location protocol. In *Proc. of International Conference on Computer Communications and Networks (ICCCN)*, 2000. *Cité dans la(les) page(s):* 23