N° d'ordre : 09/2007-E/IN

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

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

Faculté d'Electronique et d'Informatique
Département Informatique

THÈSE

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

Spécialité : **Informatique**

par

Nadia NOUALI-TABOUDJEMAT

Thème

Modèle et Techniques de Transactions Adaptables pour les Environnements Mobiles

soutenue le 17 novembre 2007, devant la commission d'examen

Mr.	M. BETTAZ	Professeur	INI/MESRS	Président
Mme.	H. DRIAS	Professeur	INI/USTHB	Directrice de thèse
Mme	M. BOUKALA	Professeur	USTHB	Examinateurs
Mme	A. DOUCET	Professeur	LIP6/UPMC	
Mr.	Z. SAHNOUN	Professeur	UMC	
Mr.	D. ZEGGOUR	Professeur	INI	
Mme	Z. ALIMAZIGHI	Maître de conférence	USTHB	

Dédicaces

A mes parents qui m'ont inculquée très tôt le l'importance des études, de la science et du travail.

A mon mari qui a été à mes côtés et m'a toujours soutenue et encouragée.

A mes deux trésors Maya et Sarah pour le bonheur qu'elles m'apportent.

A mes frères et sœurs.

A la mémoire de mes grands parents.

Those who dream by day are cognizant of many things which escape those who dream only by night.

--Edgar Allan Poe

Remerciements

Je remercie cordialement le professeur Mohammed Bettaz, Professeur à l'INI et Directeur du Département Réseaux au MESRS pour m'avoir fait l'honneur de présider le jury.

Je remercie vivement Mme Zahia Alimazighi, Professeur à l'USTHB et Doyenne de la Faculté d'Electronique et d'Informatique, Mme Malika Boukala, Professeur à l'USTHB et Chef du Département Informatique, Mme Anne Doucet, Professeur à l'UPMC, Directrice Adjointe du LIP6, Mr. Zaidi Sahnoune, Professeur à l'université Mentouri de Constantine et Chef du Laboratoire LIRE ainsi que Mr. Djamel-Eddine Zeggour Professeur à l'INI, pour l'intérêt qu'ils ont porté à mon travail en acceptant de l'examiner.

Je souhaite remercie tout particulièrement Mme Habiba Drias, Professeur à l'USTHB et Directrice de l'INI qui m'a fait confiance en acceptant d'être ma Directrice de thèse. Ses conseils m'ont été d'un très grand apport et ses encouragements m'ont donné de l'énergie.

Je voudrais témoigner ma profonde gratitude à Mme Anne Doucet, Professeur à l'UPMC pour m'avoir accueillie au LIP6 et pour son soutien lors de la préparation de cette thèse.

Je remercier vivement Mr. Abdelkader Khelladi, directeur du CERIST qui a mis à ma disposition tous les moyens dont j'avais besoin et qui m'a tant encouragée pour mener à terme ce projet de thèse.

Je tiens tout spécialement à remercier du fond du cœur mes collègues et amies Mesdames H. Aliane et S. Benmeziane pour leur soutien et leur amitié et pour tout ce que j'ai partagé avec elles.

Je n'oublie surtout pas de remercier mes collègues et membres de mon équipe avec qui j'ai partagé des moments riches d'enseignements humains et scientifiques. Ceci n'aurait pas pu se réaliser sans la confiance que Mr. M. Benhamadi, ex-directeur du CERIST, a placée en moi pour la mise en place de cette équipe de recherche sur le calcul mobile; je tiens à l'en remercier.

Je tiens à remercier mes collègues du CERIST qui m'ont témoigné leur disponibilité, en particulier, Mr. N. Meftouh et l'équipe de la bibliothèque et Mr. M. Makhlouf et l'équipe du DAM.

Je remercie mes parents, mon mari, mes enfants et mes frères et sœurs pour leur soutien inconditionnel. Depuis toujours, ils ont su contribuer à me fournir un environnement exceptionnel.

Que tous ceux qui de près ou de loin m'ont apporté un quelconque soutien trouve ici l'expression de mes sentiments les plus reconnaissants.

Nadia Nouali-Taboudjemat

Résumé

Le concept de transaction a vu le jour dans les années soixante-dix pour répondre aux besoins des systèmes de gestion de base de données. Ce paradigme permet d'assurer la cohérence en présence des accès concurrents à des données partagées et en présence de pannes. Cependant, ce paradigme a gagné une importance dépassant largement le contexte dans lequel il a été développé à l'origine et a vu son spectre d'applications s'élargir au commerce électronique, le travail coopératif, la gestion de workflow, etc.

Traditionnellement, la sémantique de transaction est définie par les propriétés ACID (Atomicité, Cohérence, Isolation et Durabilité) dont le maintien garantit la cohérence grâce à des mécanismes adéquats tels que les protocoles de validation assurant l'Atomicité et les protocoles de contrôle de concurrence assurant l'Isolation. De nombreux modèles transactionnels ont été élaborés pour prendre en charge des applications avancées diverses. Cependant, l'évolution des systèmes distribués vers les environnements sans fils et mobiles a induit de nouvelles contraintes dont le faible débit des communications sans fils et la latence du réseau qui allongent la durée de vie des transactions. De plus, les transactions mobiles requièrent l'utilisation de sites mobiles connectés au réseau par intermittence et évoluant dans un environnement non fiable sujet à de fréquentes déconnexions. La mobilité des unités permet à des transactions en cours d'exécution d'accéder à des systèmes hétérogènes, de manipuler des données de localisation imprécises ou des données dont la localisation change dynamiquement. La participation des unités portables restreintes en ressources (de calcul, mémoire, énergie) et hétérogènes rendent caduques les solutions transactionnelles traditionnelles. Toutes ces contraintes imposées par les caractéristiques de l'environnement mobile, conjuguées aux variations et au changements dynamiques de ces dernières, ainsi que la variabilité des besoins applicatifs en termes de propriétés transactionnelles interpelle à réfléchir sur des solutions offrant plus de flexibilité et d'adaptabilité.

Dans cette thèse, notre contribution a été centrée principalement sur le problème de la validation atomique qui permet d'assurer la propriété d'atomicité. Nous avons donc proposé et évalué le protocole M2PC (Mobile Two-phase Commit Protocol) pour l'architecture de réseau mobile avec infrastructure et le protocole A-D2PC (Ad hoc Decentralized 2PC) pour les architectures ad hoc. Une évaluation et étude comparative des protocoles de validations de transactions en environnement mobile a été réalisée sur une plateforme de simulation permettant de modéliser à la fois les aspects transactionnels et les aspects communication des réseaux sans fil et mobiles. Cette évaluation a permis de mettre en évidence l'apport de chaque approche et les indices de performance de ce type de protocoles. Nous avons ensuite exploré la piste de l'adaptation des protocoles de validation de transactions mobiles aux variations de contexte qui nous a conduit à la proposition et la formalisation d'un modèle de transaction flexible supportant des propriétés adaptables AdapT (Adaptable Transactions) et d'un protocole de validation aTCP (Adaptable Transaction Commit Protocol) qui permet une adaptation aux exigences des applications et du contexte mobile en termes de propriétés transactionnelles et de coût d'exécution. Ensuite, nous avons proposé une architecture middleware context-aware supportant le modèle AdapT ainsi que le protocole aTCP. L'originalité de cette architecture réside principalement dans l'utilisation de l'approche basée sur les politiques.

Mots clés : Calcul mobile, Modèle de transactions mobiles, Propriétés ACID, Protocoles de validation, Adaptabilité, Context-awareness, Réseau mobile et sans fil.

Introduction

There are two mistakes one can make along the road
to truth-not going all the way, and not starting.
-- Buddha

Les progrès réalisés en matière de communication (réseaux haut débit, normalisation des protocoles et des architectures distribués, explosion d'Internet, les communications sans fils…) conjugués à la prolifération des calculateurs ultralégers portables (ordinateurs portables, agendas électroniques, téléphones cellulaires, cartes à puce,…) permettent à un utilisateur d'accéder à des données et d'exécuter des traitements n'importe où, n'importe quand et à partir de n'importe quel terminal par le biais d'applications dites mobiles. Les applications de ce type sont multiples. Il peut s'agir d'applications personnelles dans lesquelles un utilisateur veut pouvoir accéder à tout moment à des données privées de diverses natures (données bancaires, agenda, carnet d'adresse, dossier médical, bookmarks,…). Il peut également s'agir d'applications professionnelles dans lesquelles des employés doivent accéder et partager en permanence et où qu'ils se trouvent des informations relatives à leur entreprise.

La manipulation des données par une application mobile doit, comme dans le cas des applications classiques, permettre l'accès concurrent et simultané à plusieurs utilisateurs. La cohérence de ces données, même en présence de pannes, est assurée dans les systèmes classiques par des mécanismes transactionnels. Dans le contexte de l'environnement mobile, des contraintes particulières rendent caduques l'utilisation des mécanismes existants. Aussi, a-t-il été nécessaire de revisiter les problèmes du traitement transactionnel en tenant compte des caractéristiques imposées par ce nouvel environnement.

Motivations et Objectifs

Le traitement transactionnel est un sujet très important dans les systèmes de base de données et les systèmes d'information en général. En effet, il a gagné une importance dépassant largement le contexte dans lequel il a été développé à l'origine. Le concept de transaction a vu le jour dans les années soixante-dix pour répondre aux besoins des systèmes de gestion de base de données comme un paradigme permettant d'assurer la cohérence des données partagées en présence des accès concurrents et en présence de pannes. Le transfert de fonds entre deux comptes bancaires, la réservation de places d'avion et de chambres d'hôtel, la gestion de productions sont les exemples d'applications transactionnelles classiques. Mais le spectre des applications de ce paradigme s'est rapidement élargi pour inclure la CAO, le commerce électronique, la gestion de workflow, etc.

Une transaction est une séquence d'opérations qui forment une *unité logique* de travail. L'idée principale d'une transaction est l'*indivisibilité*, c'est-à-dire, les opérations de la transaction sont soit toutes exécutées et leurs résultats rendus permanents, soit aucune ne l'est. Traditionnellement, la sémantique de transaction est définie par les propriétés **ACID (Atomicité, Cohérence, Isolation et Durabilité)** dont le maintien garantit la cohérence grâce à des mécanismes adéquats tels que les protocoles de validation assurant l'Atomicité et les protocoles de contrôle de concurrence assurant l'Isolation. De nombreux modèles transactionnels ont été élaborés pour prendre en charge des applications avancées diverses. Cependant, l'évolution des systèmes distribués vers les environnements sans fils et mobiles a induit de nouvelles exigences devant être supportées par les modèles et les techniques transactionnels. *On définit une transaction mobile comme étant une transaction distribuée dont certaines parties s'exécutent sur des sites statiques et d'autres sur des unités mobiles*. Le faible débit des communications sans fils et la latence du réseau allonge la durée de vie des transactions. De plus, les transactions mobiles requièrent l'utilisation d'unités mobiles limitées en ressources, connectées au réseau par intermittence et évoluant dans un environnement non fiable. La mobilité des unités permet à des transactions en cours d'exécution d'accéder à des systèmes hétérogènes et de manipuler des données de localisation imprécises ou des données dont la localisation change dynamiquement.

Dans cette thèse nous nous sommes donc intéressés aux techniques transactionnelles en environnement mobile. Les techniques suggérées pour l'environnement sans fil et mobile durant la dernière décennie ont un support restreint pour les environnements changeant dynamiquement. En dépit de leur habilité à satisfaire des applications ou des caractéristiques spécifiques telles que les déconnexions, elles ne possèdent pas, en général, l'habilité à faire des ajustements et des modifications au moment de l'exécution. Une des principales raisons est qu'une spécification complète d'un modèle de transaction particulier doit être effectuée avant qu'une transaction basée sur ce modèle là ne puisse être exécutée. Ceci n'est pratique que dans les scénarios où toutes les conditions et les besoins transactionnels peuvent être prévus à l'avance. Or, en raison de sa nature, l'environnement de calcul mobile est extrêmement dynamique et sujet à des changements rapides et imprévisibles. De même, les caractéristiques des applications avancées et/ou mobiles font que leurs besoins transactionnels varient. En effet, les propriétés demandées peuvent différer entre les transactions appartenant à des applications différentes ou appartenant à la même application. Ces besoins peuvent aussi changer au cours de la vie d'une application et ne peuvent pas être complètement connus au moment de sa conception. Tout ceci nous a interpellés pour réfléchir sur des solutions offrant plus de flexibilité et d'adaptabilité.

En nous concentrant sur les problèmes de la validation de transactions mobiles *(mobile transaction commitment)*, nous avons étudié les protocoles de validation dans les systèmes distribués classiques et les systèmes mobiles. Nous avons exploré les possibilités d'adaptabilité de ces protocoles tout en nous fixant l'objectif principal qui consiste à permettre une validation flexible des transactions en fonction des changements et de l'hétérogénéité des environnements d'exécution mobiles ainsi que des besoins des applications. Nous considérons que la capacité d'adaptation dépend des critères de qualité définis par les applications. Un critère de qualité est défini comme un compromis entre un coût d'exécution et la qualité des résultats des transactions. Par exemple, une application de commerce électronique peut exiger une atomicité stricte quelque soit le coût d'exécution pour sa transaction de paiement (en termes de communication, etc.). Par contre, une application de réservation de places peut se contenter d'une atomicité relâchée contre un coût d'exécution plus faible vu qu'une information approximative sur les places disponibles est suffisante pour le bon fonctionnement de ce type d'application.

Contributions

Notre étude de l'état de l'art se compose d'une analyse des concepts et des techniques proposées pour les modèles et les techniques de transactions mobiles. Le nombre important des travaux de cette analyse montre le caractère fondamental de ce domaine et son intérêt aussi bien théorique que pratique. Les travaux étudiés ont souvent adapté des techniques et des modèles des systèmes répartis et/ou multibases. Malgré la variabilité de l'état de l'environnement mobile, les solutions proposées se sont souvent intéressées seulement aux déconnexions. Les modèles d'exécution se sont basés sur une exécution complète sur l'unité mobile ou sur le réseau fixe pour des contextes applicatifs spécifiques, ignorant presque tout le temps le modèle d'exécution répartie mobile. Cette analyse nous a permis de nous rendre compte de la nécessité de disposer de techniques transactionnelles offrant la *flexibilité* et l'*adaptabilité* à de nombreux besoins applicatifs et environnementaux. Ces deux propriétés n'ont pas été suffisamment abordées dans les travaux étudiés.

Après avoir étudié le problème de la validation atomique en environnement mobile, nous avons proposé deux protocoles de validation de transaction mobile: M2PC *(Mobile Two-phase Commit Protocol)* pour les architectures à infrastructure et A-D2PC *(Ad hoc Decentralized 2PC)* pour les architecture ad hoc. Le protocole M2PC est une adaptation du protocole standard 2PC *(Two-phase Commit Protocol)* de validation de transactions distribuées qui prend en charge le problème des déconnexions des unités mobiles et leur mobilité. La spécificité de M2PC réside dans le fait qu'il intègre un mécanisme qui lui permet de s'exécuter sur une infrastructure réseau n'offrant pas de support pour la gestion de la mobilité des unités portables. L'évaluation de M2PC a permis de mettre en évidence ses performances face aux caractéristiques de l'environnement sans fil et mobile et aussi par rapport à l'approche de support de mobilité qu'il utilise. Le protocole A-D2PC propose d'utiliser la variante décentralisée du 2PC, en lui associant d'autres mécanismes, afin de lutter contre les contraintes des réseaux ad hoc en particulier le manque de support fixe et les fréquentes déconnexions.

Nous avons ensuite réalisé l'étude et l'évaluation des protocoles de validation de transactions en environnement mobile. L'originalité de ce travail, par rapport à ce qui a été proposé dans le domaine, réside dans le fait qu'à notre connaissance peu ou pas de travaux réalisés sur les protocoles de validation atomique en environnement mobile sont basés sur la simulation. En effet, les évaluations des protocoles proposés dans la littérature ont été limitées à des estimations sur les pires et meilleurs cas. Pour la simulation, nous avons utilisé une plateforme permettant de modéliser à la fois les aspects transactionnels et les aspects communication des réseaux sans fil et mobiles. Cette approche a permis de montrer l'impact important et souvent imprévisible des caractéristiques liées aux communications sans fil et à la mobilité sur les techniques transactionnelles et de mettre en évidence les apports et indices de performance de chaque approche.

Nous avons adopté une approche générale qui ne cible pas d'application ou de contrainte matérielle particulière pour explorer la possibilité d'adaptation dynamique des protocoles de validation de transactions mobiles aux variations de contexte ce qui nous a permis de proposer:

- un modèle de transaction flexible supportant des propriétés adaptables que nous avons spécifié par le formalisme ACTA permettant de définir un modèle de transaction et ses propriétés.

- un protocole de validation *a*TCP (*Adaptable Transaction Commit Protocol)* qui permet une adaptation aux exigences des applications et du contexte mobile en termes de propriétés transactionnelles et de coût d'exécution.

- une architecture d'adaptation pour une implémentation context-aware dont l'originalité réside principalement dans l'utilisation du concept de *politique* pour l'adaptation transactionnelle. Cette architecture supporte le modèle AdapT et le protocole *a*TCP.

Organisation du document

Cette thèse est organisée en sept chapitres:

Le chapitre 1 présente les problèmes induits par les caractéristiques de l'environnement mobile et leur impact sur le traitement transactionnel. Le chapitre 2 présente l'évolution du concept de transaction et les modèles des systèmes traditionnels aux systèmes mobiles. Il donne une analyse et une synthèse des travaux de recherche qui proposent des solutions aux problèmes rencontrés par les transactions en environnement mobile.

Le chapitre 3 introduit l'impact de la mobilité sur la validation des transactions en environnement mobile et présente une étude des protocoles proposés montrant de quelle manière la propriété d'atomicité est assurée en présence des contraintes sur les ressources, les déconnexions et la mobilité des utilisateurs. Le chapitre 4 présente les protocoles M2PC et A-2PC avec une analyse de leur fonctionnement et performances. Le chapitre 5 donne les résultats de l'évaluation et de l'étude comparative d'un ensemble de protocoles de validation de transactions mobiles et conclut sur les indices de performance des différentes approches.

Le chapitre 6 est consacré à la conception d'un modèle de transaction et d'un protocole de validation adaptable et leur mise en œuvre est présentée dans le chapitre 7. Dans ces deux chapitres nous avons présenté la conception de techniques transactionnelles flexibles pouvant s'accommoder à un grand nombre de scénarios d'utilisation et de contexte d'exécution. Nous nous sommes intéressés plus spécifiquement aux protocoles de validation assurant une propriété d'atomicité adaptable.

La conclusion récapitule les contributions de cette thèse et présente quelques perspectives de recherche.

Chapitre I

Impact de la mobilité sur le traitement transactionnel

If I had more time, I could have
written you a shorter letter
--Blaise Pascal

Il est incontesté que la mobilité soulève plusieurs nouveaux défis à la communauté des chercheurs [Wei93, AK93, IB94, PS98, JHE99, Fra01]. Le calcul mobile présente de nouvelles variables qui doivent être prises en compte: localisation des équipements de calcul mobiles changeant dynamiquement, qualité variable de connexion du lien utilisé pour l'accès au réseau fixe, déconnexions fréquentes involontaires ou volontaires, et ressources limitées des unités mobiles qui tendent à avoir de faibles capacités de calcul, des petites batteries et des petits écrans. En outre, le calcul mobile doit faire face à tous les problèmes spécifiques aux systèmes d'information hétérogènes [SL90, GSC96], à l'intégration de base de données [BLN86], à la résolution de conflit [KCGS95], et à la gestion globale de transaction [BGS92].

Afin d'offrir la flexibilité et l'ubiquité d'accès et de manipulation des informations des recherches intenses sont menées dans le domaine de gestion des données qui rencontre de nombreux défis. En effet, les approches traditionnelles ne sont pas toujours directement applicables au contexte mobile et donc les chercheurs ont été amenés à revisiter les problèmes du domaine et à étudier de plus près les nouveaux challenges afin de proposer les solutions les plus adéquates. Le thème des transactions mobiles qui nous intéresse dans cette thèse est l'un des thèmes majeurs de la gestion des données qui a suscité et continue de le faire un grand intérêt depuis le début de l'apparition de l'informatique mobile. Comme pour les autres domaines, la gestion des transactions mobiles doit faire face aux incertitudes des environnements de calcul mobile tout en fournissant une manière viable d'accès concurrents et de traitement des données dans les réseaux constitués de composants statiques et mobiles hétérogènes.

1.1 Caractéristiques de l'environnement mobile

Il est important de comprendre que le but final du paradigme de calcul mobile est de permettre aux utilisateurs d'accomplir leurs tâches à l'aide des dispositifs de calcul, n'importe quand et n'importe où. Pour réaliser ce but, la connectivité réseau est une part essentielle des unités de calcul mobiles. Le réseau sous-jacent dans le calcul mobile est en général sans fil.

1.1.1 Architecture du système mobile

La figure 1.1 montre le modèle d'architecture, largement accepté dans la littérature [DH95, IB94, Bad98], d'un système supportant un environnement de calcul mobile. Un ensemble d'ordinateurs (PCs, stations de travail, etc.) sont interconnectés par un réseau de communication filaire classique à haut débit. Parmi ces ordinateurs certains sont des unités (sites ou hôtes) fixes, UF *(Fixed Hosts, FH)*. D'autres sont des stations base, SB *(Base Stations, BS*) ou stations de support de base *(Mobile Support Station, MSS)* munies d'une interface de communication sans fil, leur permettant de communiquer avec les sites mobiles. Les sites (ou hôtes) mobiles *(Mobile Hosts, MH)* ou unités mobiles, UM *(Mobile Units, MU)* sont connectées à une SB à travers des canaux sans fil. Les UFs ne sont pas équipées pour communiquer directement avec les UMs. Les SBs fournissent une passerelle de communication entre la partie filaire et la partie non filaire du réseau. La littérature consacrée à la gestion des données considère souvent que les stations base ne sont pas limitées à la fonction de routage ou de communication mais peuvent supporter des applications telles qu'un serveur de base de données.

Les UMs peuvent être de configurations diverses; avec ou sans disque, des capacités de mémorisation et de calcul plus ou moins modestes et sont alimentées par des sources d'énergie autonomes (batteries). Elles peuvent se déplacer librement dans une zone géographique limitée, qui pour une meilleure réutilisation des fréquences, est elle-même subdivisée en *cellules* de plus petite taille. Chaque cellule est gérée par une SB particulière. Une UM peut se déplacer d'une cellule à l'autre, mais ne peut être connectée à un instant donné qu'à une seule station base. Le processus de passage d'une cellule à l'autre est assuré par un protocole appelé *Handoff*. Ce processus est transparent aux UMs. Il est responsable du transfert des informations d'état, relatives au calcul mobile en cours, vers la station base de la nouvelle cellule [PB94b, DHB97, WC97, Bad98].

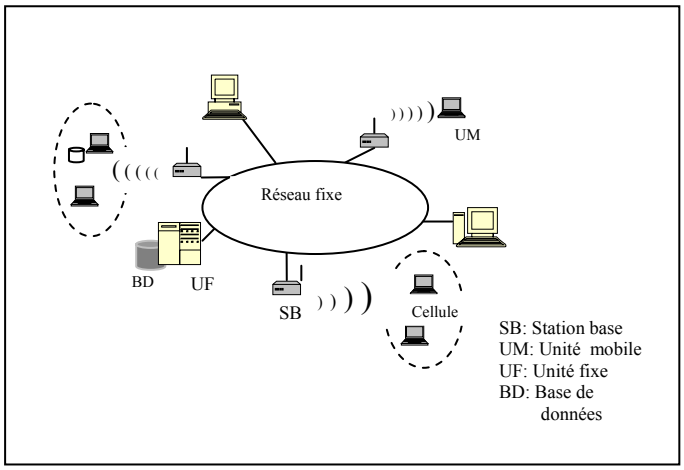


Figure 1.1-Architecture d'un système mobile avec infrastructure

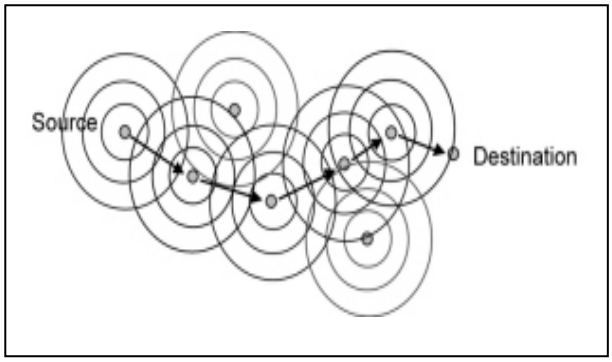


Figure 1.2-Architecture d'un système mobile sans infrastructure (ad hoc)

Le modèle de réseau décrit dans la figure 1.1 est appelé réseau mobile à infrastructure du fait qu'il comporte une partie filaire fixe qui supporte les communications entre les unités mobiles. Le modèle de réseau mobile sans infrastructure, dit réseau Ad hoc *(Mobile Ad Hoc Network, MANET)*, ne comporte que des unités mobiles communiquant d'une manière directe (sans station de base) en utilisant leurs interfaces de communication sans fil (figure 1.2). L'absence de l'infrastructure filaire oblige les unités mobiles à se comporter comme des routeurs qui participent à la découverte et à la maintenance des chemins pour les autres sites mobiles du réseau [Puj01, Gar02].

1.1.2 Les problèmes induits par un système mobile

La mobilité des utilisateurs et donc des unités portables, les caractéristiques physiques de ces dernières ainsi que les communications sans fil induisent les problèmes suivants [Sat96, FZ94, IB94, AK93] :

La mobilité. La localisation d'éléments mobiles et donc de leur point d'attache au réseau fixe change en fonction de leur déplacement d'une cellule à l'autre. Les conséquences de cette mobilité sont nombreuses :

- La configuration d'un système comportant des unités mobiles n'est pas statique. Ceci implique que: (i) la conception d'algorithmes distribués ne peut plus être basée sur une topologie fixe (anneau, arbre, …); (ii) le centre d'activité, la charge du système et la position des sites change dynamiquement.

- Gestion de la localisation : (i) le coût induit par la localisation des unités mobiles s'ajoute au coût de chaque communication les impliquant; (ii) des structures de données efficaces et des algorithmes appropriés doivent être conçus pour la représentation, la gestion et l'obtention de la position des unités mobiles, qui est une donnée qui change rapidement.

- L'hétérogénéité : (i) la connectivité *(connectivity)* varie sensiblement en termes de performance et de fiabilité. Par exemple, à l'extérieur, un client mobile doit compter sur des réseaux à faible bande passante, alors qu'à l'intérieur d'un bâtiment une connexion fiable et à large bande ou même des connexions filaires peuvent être mises à sa disposition. D'autre part, l'existence de zones non correctement couvertes par les services de communication, peuvent conduire à des déconnexions de durées variables; (ii) le nombre d'unités contenues dans une cellule peut varier dans le temps, causant la variation de la charge au niveau de la SB et de la bande passante disponible; (iii) les unités mobiles ne sont pas équipées de la même manière en ressources. Selon le type et les fonctionnalités pour lesquelles elles sont destinées, elles peuvent, par exemple, disposer de plus ou moins de mémoire ou d'une taille d'écran plus ou moins importante.

- La mobilité soulève aussi des problèmes très importants liés à la sécurité et à l'authentification.

Le médium de communication non filaire. Les communications non filaires sont caractérisées par :

- Une connexion faible et intermittente. Les réseaux non filaires sont plus coûteux, offrent une largeur de bande faible et sont moins fiables que les réseaux filaires. Les communications non filaires font face aux imperfections dues aux interférences du signal avec le bruit émanant de l'environnement qui les entoure. Ainsi, alors qu'en environnement filaire d'énormes progrès ont été réalisés en termes de largeur de bande, les produits pour les communications non filaires atteignent quelques Kbps seulement pour les ondes et la téléphonie cellulaire pourrait atteindre quelques Mbps. Du fait que la bande disponible est partagée entre les utilisateurs appartenant à la même cellule, la largeur de bande exploitable par un utilisateur est encore plus faible. Concernant la transmission radio, le taux d'erreur est si élevé que la bande effective reste très limitée. La bande passante est actuellement une ressource rare et malgré les progrès de la technologie, il n'est pas faux de supposer qu'elle restera une limitation importante et une contrainte de performance dont il faudra tenir compte dans la conception d'un système mobile. Les transmissions par satellite sont, quant à elles, très coûteuses et souvent utilisables en réception seulement.

- Une déconnexion facile. La déconnexion d'une unité mobile du reste du réseau peut être volontaire ou involontaire, prévisible ou soudaine, de courte ou de longue durée. En effet,

dans les systèmes distribués classiques, un site est soit connecté, soit totalement déconnecté du réseau. Par contre, dans un environnement mobile, il existe plusieurs modes de liaison allant d'une déconnexion totale à une connexion faible lorsque le débit de la liaison de communication est faible. Par exemple, pour économiser de l'énergie, une unité mobile peut fonctionner en mode veille *(doze mode)* ; dans ce cas, la vitesse du processeur est réduite et aucun calcul n'est effectué. L'unité mobile attend la réception d'un message pour reprendre son fonctionnement normal.

- Une connectivité variante. Les technologies non filaires offrent une gamme de produits ayant une largeur de bande et une fiabilité variables.

- Un support physique de la diffusion. Un canal de diffusion très large partant de la station base vers les unités mobiles est offert. C'est un avantage à exploiter dans le calcul mobile.

- Les tarifs. Pour certains réseaux (exemple la téléphonie cellulaire), l'accès réseau est facturé en fonction du temps de connexion, alors que pour d'autres (exemple les émissions radio), il est facturé en fonction du nombre de messages.

La portabilité des unités mobiles. La mobilité n'aurait pu être possible sans les avancées technologiques dans la conception des unités portables. Cependant, la portabilité est obtenue au prix de certaines limitations dans les ressources de calcul.

- Les unités portables sont pauvres en ressources comparées aux unités ou sites statiques. Les unités mobiles doivent être légères et de petites dimensions afin d'être facilement transportables. De telles considérations conjuguées au coût et niveau de la technologie feront en sorte que ces unités resteront toujours moins équipées (en mémoire, taille de l'écran et la capacité du disque) que les unités statiques. Ceci conduit à une asymétrie entre les deux types de sites existant sur un même réseau ; un autre facteur à ne pas négliger dans la conception des systèmes de calcul mobile.

- Les unités mobiles comptent sur des batteries à capacité limitée pour leur alimentation. Malgré les progrès technologiques, ce problème ne cessera pas d'exister. Un intérêt doit être porté à la consommation d'énergie à tous les niveaux de conception hardware ou software.

- Les unités mobiles sont sujettes aux accidents, au vol et à la perte. Ils sont par conséquent, moins fiables et moins sécurisées que les unités statiques.

1.2 Des systèmes répartis aux systèmes pour environnements mobiles

Pour de nombreuses applications, il n'est pas raisonnable de supposer que tous ses besoins en termes de traitement d'information peuvent être satisfaits à travers un serveur de données unique. En fait, une entreprise moderne dispose d'une architecture largement distribuée qui consiste en une variété de serveurs d'applications et de données. Ces serveurs peuvent être hétérogènes en termes de matériel informatique, d'interface et de protocoles. Ils peuvent également être autonomes ou avoir la capacité d'interopérer avec d'autres serveurs. Ce type d'architecture est connu sous le nom de système fédéré.

Lorsque les données sont distribuées sur un ensemble de serveurs fédérés, deux cas peuvent se présenter et ont un impact direct sur les techniques transactionnelles. Premièrement, les serveurs participant à la fédération peuvent être logiquement une partie d'un seul système ; il est alors possible de supposer qu'ils exécutent la même suite de protocoles. Deuxièmement, les serveurs peuvent être complètement autonomes et indépendants les uns des autres ; ils sont simplement accédés ensemble par une application. Dans ce cas, il n'est pas possible de supposer l'uniformité des protocoles ce qui complique la

situation. Dans le cas des bases de données, l'hétérogénéité concerne, par exemple, le modèle des données, le langage de requêtes, l'implémentation des concepts, etc. Les systèmes fédérés dans lesquels tous les serveurs sont des systèmes de gestion de bases de données (SGBDs) sont appelés systèmes *multibases (SMBD)* ou *(Multi DataBase Systems, MDBS)*.

Il est naturel de considérer que l'environnement de calcul mobile est une extension de l'environnement de calcul réparti. Nous considérons qu'un système mobile comprend une application avec des utilisateurs mobiles ou fixes qui accèdent aux données localisées sur des sites également mobiles ou fixes. Un système mobile peut être réparti à l'échelle d'une zone géographique comme un bâtiment, un quartier, une ville, un département, une région, un pays, etc. Un tel système peut être composé d'un nombre important de serveurs de données, souvent des SGBDs, qui diffèrent par leur niveau d'autonomie et d'hétérogénéité. Nous adoptons ici (voir figure 1.3) la classification proposée dans [DHB97], qui est elle même étendue de la classification de [OV99]. Cette classification consacrée aux SGBDs (mais peut tout à fait s'adapter à différents types de systèmes) est basée sur les caractéristiques d'autonomie, de distribution et d'hétérogénéité. Les axes d'autonomie, de distribution et d'hétérogénéité sont définis comme suit:

Autonomie. Elle indique le degré d'indépendance des composants du système. Ceci concerne, entre autres, la liberté de choisir le type de données utilisées, la manière de les manipuler, la liberté de décider le type d'information à partager avec le système global, ainsi que l'indépendance d'exécution.

Hétérogénéité. Elle fait référence à l'hétérogénéité du matériel informatique mais aussi des gestionnaires de données : langages de requêtes, gestionnaires de requêtes, gestionnaires de transactions, etc.

Distribution. Elle fait référence à la distribution physique des données mais aussi aux fonctions de gestion de données.

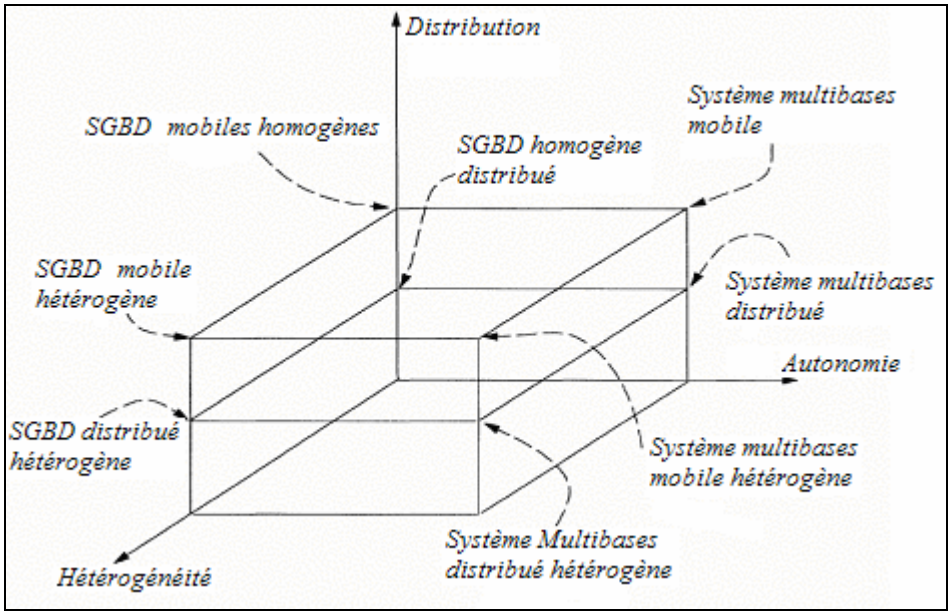


Figure 1.3-Classification des systèmes de bases de données mobiles

La figure 1.3 présente les systèmes mobiles en ajoutant un niveau sur l'axe de distribution vu qu'un système de calcul mobile peut inclure un réseau fixe représentant un système distribué. Un système de calcul mobile peut ainsi être vu comme un type dynamique de système distribué où les liens entre nœuds changent dynamiquement. A partir de cette

catégorisation, il est possible conceptuellement d'avoir un système mobile avec pas du tout d'autonomie ou d'hétérogénéité ; ce qui est peu probable dans les applications modernes. Le cas le plus général est celui qui est constitué de serveurs mobiles, hétérogènes et autonomes. Dans ce type de configuration, un serveur même léger comporte des capacités de gestion lui permettant de travailler de façon autonome lors des déconnexions ou lorsque les capacités de communication sont faibles. En s'intéressant à des techniques mobiles conçues pour ce type d'environnement, cela signifie la prise en charge des problèmes de l'environnement mobile le plus complexe.

De nombreux axes de recherche ont vu le jour dans les systèmes mobiles [SKK+90, IB94, DH95, PS98, NB01, BBO+03, ACPJ03] autour des problèmes de gestion des données. La dissémination d'information (météo ou trafic routier) à un large nombre d'utilisateurs mobiles permet de profiter des capacités de diffusion offertes par les réseaux de communication sans fil [AAFZ96, FZ96, PC99a, PC99b, DVCK99]. La gestion de données spatio-temporelles motivée par la nécessité de localisation des UMs durant leurs déplacements [WXCJ98, GBE+00, NB03] répond aux besoins des applications utilisant des requêtes dépendantes de la localité (par exemple : trouver l'hôtel/restaurant le plus proche). La gestion des données embarquées a pour objectif de minimiser la consommation de la mémoire et des processeurs (exemple : dossier médical ou d'assurance embarqué sur cartes à puce) [PBVB01, ABP03]. La duplication/synchronisation des données dans les applications de type client-serveur où les données peuvent être dupliquées sur les clients mobiles à partir de serveurs fixes offre aux clients la possibilité de manipuler les données localement (lors des déconnexions) pour ensuite les synchroniser[1] avec les serveurs (lors des reconnexions) [VCFS00, BP98].

Les travaux sur la gestion des données peuvent être classés en trois modèles d'architectures adoptés par les solutions existantes:

Le modèle client-serveur est une architecture à deux niveaux avec des données distribuées sur des serveurs. Les serveurs sont responsables de la réplication, du stockage et de la mise à jour des données. Ils sont souvent fixes et résident sur l'infrastructure filaire. Les clients n'ont aucune autonomie car ils dépendent entièrement des serveurs et peuvent être ou ne pas être mobiles et hétérogènes. Ce modèle était l'approche qui a été la première adoptée pour les systèmes de fichiers répartis et de base de données car elle simplifie la logique de gestion des données et permet un déploiement rapide [SKK+90]. Le modèle délègue toute la responsabilité de gestion des données à seulement un petit sous-ensemble de serveurs. En plus, le modèle adresse le problème de mobilité simplement en ne le traitant pas ou en employant des méthodes traditionnelles de délais de garde *(timeouts)*.

Le modèle client-agent-serveur ou **client-proxy-serveur** étend l'approche précédente en introduisant un niveau additionnel dans la hiérarchie. Les données restent distribués sur des serveurs résidant sur une infrastructure filaire. Les clients dépendent toujours des serveurs et peuvent être ou ne pas être mobiles et hétérogènes. Cependant, un agent, résidant sur l'infrastructure filaire, est placé entre les clients et les serveurs. L'agent prend un sous-ensemble des responsabilités des serveurs, incluant la gestion des déconnexions, de cache et du transcodage. En conséquence, les serveurs ne différencient plus les clients mobiles et les clients fixes. Ils peuvent traiter tous les clients uniformément puisqu'ils communiquent avec des entités situées sur l'infrastructure filaire. Les proxies sont alors responsables de fournir les données aux clients et de maintenir des sessions quand les clients changent de localisation [DHB97].

[1] Opération qui consiste à répercuter le résultat des manipulations locales des données sur la (les) copie(s) du (des) serveur (s)

Le modèle pair-à-pair *(peer-to-peer)* adopte une approche complètement différente des deux autres modèles. Ce modèle est fortement autonome car chaque entité de calcul doit pouvoir opérer de manière indépendant. Il n'y a aucune distinction entre les serveurs et les clients et leur responsabilité. Les données peuvent résider sur n'importe quel dispositif, et chaque dispositif peut être hétérogène et mobile. Dans ce modèle, deux dispositifs quelconques peuvent interagir l'un avec l'autre [PAC+02]. De plus, à la différence des approches client-serveur, ce modèle est ouvert dans le sens où il n' y a aucun ensemble strict de conditions que chaque dispositif doit suivre. Ceci peut conduire au fait que la couche de gestion des données peut être implémentée différemment sur chaque dispositif. En conséquence, chaque pair doit se charger des problèmes locaux et globaux de gestion des données ; ces derniers sont pris en charge par des serveurs ou des proxies dans les modèles client-serveurs.

Les architectures client-serveur et client-agent-serveur demeurent les modèles les plus populaires, d'une utilité pratique du point de vue commerciale et non commerciale. Ces deux architectures fournissent aux utilisateurs certaines garanties, telles que la connectivité, largeur de bande fixe et sécurité, en raison de la puissance inhérente des agents et des serveurs. Du point de vue commerciale, elles garantissent des revenus accrus aux fournisseurs d'infrastructure et de services, au fur et à mesure que le nombre d'utilisateurs sans fil augmente. Les architectures pair-à-pair, qui reflètent vraiment le but du calcul n'importe où et n'importe quand, sont en grande partie confinées au milieu universitaire, mais possèdent le potentiel de révolutionner le calcul mobile dans les décennies à venir.

Afin d'offrir un support de calcul cohérent et fiable, la gestion des données doit fournir un support transactionnel, et, c'est le sujet qui nous intéresse dans cette thèse et qui consiste en la gestion des transactions mobiles en présence des contraintes de l'environnement mobile (déconnexions, bande passante faible, mobilité des unités portables, etc.) [GHPS96, PB99, LS95, WC99].

La recherche dans le domaine des transactions mobiles est le centre du chapitre 2. La section suivante présente les concepts de base du paradigme de transaction et ses nouveaux défis dans l'environnement de calcul mobile.

1.3 Transactions : concepts et impact de la mobilité

Le concept de transaction résout d'une façon très élégante le problème de maintient de la cohérence en présence d'un haut degré de concurrence d'accès aux serveurs de données et en présence de défaillances de différentes origines. Ceci est obtenu d'une façon générique transparente à la logique de l'application libérant ainsi le développeur de ces différents problèmes. Généralement, un SGBD est l'instanciation la plus importante de serveur de données transactionnel. Cependant, d'autres types de serveurs tels que les serveurs de messagerie, les serveurs Web ou intranet, les systèmes de gestion de workflow font aussi appel au support transactionnel.

1.3.1 Concepts de base du traitement transactionnel

Une transaction est une séquence d'opérations qui, lorsqu'elle est exécutée seule dans un environnement sans pannes, fait passer le système d'information d'un état cohérent initial à un état cohérent final. Les états intermédiaires peuvent être temporairement incohérents [Gra81]. Une transaction est dite centralisée lorsqu'elle s'exécute entièrement sur un site. Elle est dite répartie lorsqu'elle met en jeu des actions sur des objets situés sur des sites distincts [BBK91].

1.3.1.1 Les propriétés ACID

Les transactions sont caractérisées par les propriétés **ACID (Atomicité, Cohérence, Isolation et Durabilité)** [BBK91, GR93]:

Atomicité. La séquence d'actions d'une transaction est indivisible: soit toutes les actions de la transaction sont exécutées et la transaction est **validée** (le système atteint un nouvel état cohérent), soit aucune ne l'est et la transaction est **rejetée** (annulée ou abandonnée) (le système reste dans l'état cohérent initial). C'est le principe du tout ou rien *(all-or-nothing principale)*.

Cohérence ou Consistance. L'exécution d'une transaction dans un environnement sans concurrence et sans pannes ne doit pas introduire d'incohérences.

Isolation. Les actions d'une transaction sont isolées. Le résultat des actions intermédiaires (état temporairement incohérent) est masqué aux autres transactions. Une autre manière de présenter cette propriété est que le résultat d'une exécution parallèle de plusieurs transactions est équivalent à celui d'une exécution séquentielle. On parle alors de **sérialisation** (en théorie la sérialisation est plus restrictive que l'isolation).

Permanence ou Durabilité. Les résultats d'une transaction qui s'est bien terminée ne peuvent pas être détruits ultérieurement par l'occurrence d'une panne.

Si toutes ces propriétés sont vérifiées, on parle de cohérence forte (ou cohérence stricte). Si une des propriétés n'est pas vérifiée la cohérence est dite dans un état dégradé ou faible.

L'exemple typique utilisé pour illustrer ces concepts est l'opération de virement suivant entre deux comptes bancaires :

Procédure de virement (Nd, Nc, M)

Début-transaction

Nd : numéro du compte à débiter, Nc : numéro du compte à créditer, M : montant

Débiter (Nd, M)

Créditer (Nc, M)

Fin-transaction

Dans l'exemple du virement, l'atomicité garantit que le transfert de fond n'est effectif que si chacune des actions de débit et de crédit est réalisée. La consistance signifie que la somme débitée doit être égale à la somme créditée. L'isolation se traduit par le fait que les deux comptes bancaires soient inaccessibles aux autres transactions durant l'opération de transfert de fonds. La durabilité quant à elle implique que le transfert de fonds est pris en compte de façon définitive après la validation de la transaction.

La propriété de cohérence d'une transaction est assurée par le contrôle d'intégrité, et est généralement du ressort du concepteur de l'application (même si certains systèmes intègrent un vérificateur de cohérence). Les propriétés d'atomicité et de durabilité sont assurées grâce à des techniques **de validation et reprise** ou **recouvrement**. Les mécanismes de **contrôle de concurrence** permettent de maintenir la propriété d'isolation.

1.3.1.2 Contrôle de concurrence

Le but de ces techniques est d'assurer la propriété de sérialisation. Le principe consiste à ordonnancer les actions des transactions concurrentes de telle sorte que le résultat de leur exécution soit équivalent à celui d'une exécution séquentielle. L'ordonnancement est basé sur

la détection des conflits d'accès aux objets partagés par des transactions concurrentes. Il y a conflit d'accès lorsque deux transactions exécutent des actions incompatibles sur le même objet (exemple lire et écrire). L'apparition des conflits crée des relations de dépendance qui peuvent être représentées à l'aide d'un graphe dans lequel les noeuds représentent les transactions et les arcs représentent les dépendances engendrées par les conflits d'accès. Un cycle dans le graphe des dépendances indique une exécution non sérialisable. Il existe deux groupes de techniques de contrôle : le contrôle continu et la certification. Le contrôle continu assure la sérialisation tout le long de la vie des transactions, celles qui provoquent des exécutions non sérialisables sont soit abandonnées soit mises en attente. Ces méthodes dites pessimistes supposent une fréquence élevée de conflits et sont bâties autour de deux techniques de base : le verrouillage à deux phases et l'estampillage. Le contrôle continu utilise des méthodes dites optimistes qui supposent une fréquence faible de conflit et laisse les transactions s'exécuter d'abord. Lorsqu'une transaction atteint le point de validation, le système entame la vérification des conflits. Si un conflit est détecté, la transaction est redémarrée.

a. Verrouillage à deux phases: Le principe de fonctionnement du verrouillage à deux phasesou 2PL *(Two Phase Locking)* consiste à protéger les objets partagés par des verrous. Le système maintient une table d'allocation des verrous aux transactions (table 1.1). Un conflit a lieu lorsqu'une demande de verrouillage, émise par une transaction, est incompatible avec le verrou existant sur l'objet. La relation de dépendance est définie par l'ordre chronologique des opérations sur l'objet partagé. Pour traduire cette dépendance en termes d'exécution séquentielle, on force la transaction à l'origine du conflit à attendre la libération de l'objet par la transaction concurrente. Le graphe des dépendances représente alors le groupe des attentes entre transactions. Un contrôleur de concurrence *2PL strict* est un contrôleur de concurrence 2PL qui détient les verrous d'une transaction jusqu'à sa validation/annulation, les verrous sont libérés tous à la fois.

La mise en attente des transactions n'ayant pas obtenu un verrou, peut provoquer des situations d'interblocage qui peuvent être traitées par un algorithme de prévention ou de détection d'interblocage.

Table 1.1-Compatibilité des verrous

Détenu **Demandé**	**Aucun verrou**	**Lecture**	**Ecriture**
Lecture	Compatible	Compatible	Incompatible
Ecriture	Compatible	Incompatible	Incompatible

b. Ordonnancement par estampillage : Le principe de ces techniques consiste, lorsqu'il y a conflit, à forcer les transactions à s'exécuter selon un ordre de sérialisation pré-établi. En cas de conflit, le système vérifie que les accès concurrents sont effectués dans l'ordre pré-établi entre les transactions à l'aide de l'estampille. Si c'est le cas, l'exécution est acceptée, sinon, la transaction est annulée et reprise. C'est une technique adaptée à la mise en cohérence des copies multiples d'une donnée.

c. Méthodes des contrôles par certification : Cette approche diffère les contrôles à la fin de la transaction (c'est donc un contrôle postérieur). Si une action incompatible avec celles des transactions concurrentes est décelée, la transaction est annulée et reprise. Sinon la transaction est terminée. La transaction est donc exécutée en trois (03) phases [BG81]:

- *La phase de lecture:* une transaction peut lire les valeurs des données qu'elle manipule. Les mises à jour (les opérations d'écriture) ne sont pas directement appliquées sur les données mais sauvegardées temporairement dans l'espace de travail de la transaction.

- *La phase de validation:* le système vérifie d'abord l'existence de conflits avec d'autres transactions afin d'assurer que la propriété de sérialisation n'est pas violée. L'ensemble des données lues par une transaction est noté *readset* et l'ensemble des données écrites par une transaction est noté *writeset*. Durant la phase de validation, le système vérifie si aucun élément de l'ensemble *readset* de la transaction n'existe dans les *writeset* des autres transactions. S'il existe, la transaction est défaite et redémarrée. La validation peut être à l'encontre de transactions validées (on parle de *validation backward*) ou de transaction *actives*[2] (on parle de *validation forward*). Si la propriété d'intégrité des données n'est pas vérifiée pendant la phase de validation, alors la transaction est défaite et redémarrée.

- *La phase d'écriture:* si la transaction est validée, les mise à jour sont appliquées sur la base de données.

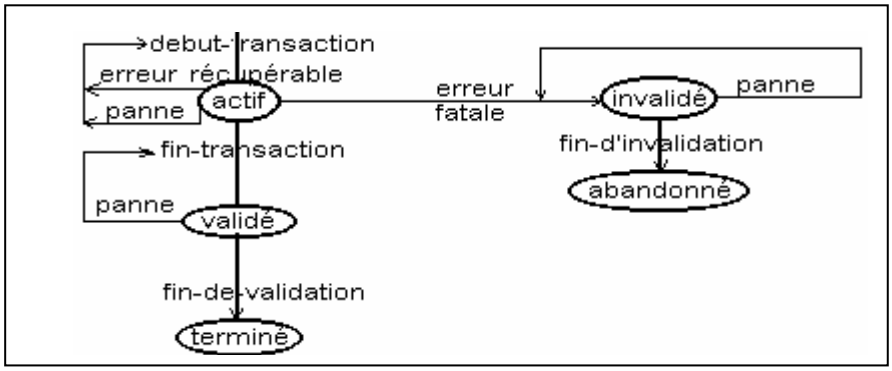


Figure 1.4-Diagramme d'état d'une transaction.

1.3.1.3 Validation et reprise

Il s'agit de garantir les propriétés d'atomicité et de permanence en présence de défaillances telles que celles qui provoquent l'arrêt de la transaction avant sa terminaison (violation de la propriété d'indivisibilité) et celles qui entraînent une disparition de l'information en mémoire secondaire (violation de la propriété de permanence). Afin d'assurer le contrôle d'atomicité à l'aide des deux procédures **défaire** *(undo)* et **refaire** ***(redo)*** exécutées par le gestionnaire de transactions, l'exécution d'une transaction se fait en deux étapes, calcul et validation, et un journal est prévu en mémoire permanente. Pendant l'étape de calcul qui correspond à l'exécution de la transaction, les modifications sur les objets sont enregistrées dans le journal. Pendant l'étape de validation *(commitment)* le système consolide les actions qui ont été exécutées sur les objets pendant la phase de calcul. Les deux formes de journaux les plus répandus sont le journal-avant *(before-journal)* qui stocke les valeurs des objets avant leur modification par une transaction et le journal-après *(after-journal)* qui stocke les valeurs des objets modifiés par une transaction. Il est utilisé pour refaire les actions d'une transaction. La validation consiste à propager les modifications enregistrées dans l'espace de travail de la transaction vers la mémoire secondaire. Les journaux permettent de fiabiliser cette opération.

Lors d'un redémarrage après défaillance, le système transactionnel exécute pour chaque transaction, l'action correspondant à l'état enregistré dans le journal :

- état actif : la transaction est annulée et ses actions sont défaites,
- état validé : la procédure de validation est refaite,
- état invalidé : la procédure d'invalidation est refaite,
- état terminé ou abandonné (annulé) : pas d'action.

[2] Une transaction active désigne une transaction en cours d'exécution et non encore validée.

1.3.2 Impact de l'environnement mobile sur les transactions

Les transactions réparties accèdent à des données réparties (ou dupliquées) sur plusieurs sites et sont capables de coordonner leurs actions. Conceptuellement, les transactions d'un système homogène sont des transactions réparties ordinaires. Par contre, dans un système hétérogène une transaction accédant à plusieurs serveurs peut interférer avec des transactions qu'un serveur spécifique traite localement, nous avons donc des transactions locales et des transactions globales. Les transactions globales sont exécutées sous le contrôle du gestionnaire global de transaction (GTG) *(Global Transaction Manger, GTM)*, alors que les transactions locales sont exécutées sous le contrôle des gestionnaires de transactions locaux (GTL) *(Local Transaction Manager, LTM)*. Dans le cas d'un système réparti hétérogène, chaque GTL peut employer des schémas et des techniques de gestion de transaction différents. Si l'on considère qu'une transaction mobile est une extension à une transaction répartie, alors nous obtenons aussi deux types de transactions mobiles:

-dans le cas simple, une transaction mobile est une transaction répartie s'exécutant dans un environnement *réparti homogène* incluant des sites mobiles communicants via des réseaux sans fil.

-dans le cas le plus général, une transaction mobile est une transaction répartie s'exécutant dans un environnement *réparti hétérogène* incluant des sites mobiles communicants via des réseaux sans fil.

Etant donnée la tendance des applications modernes associant des serveurs de données multiples pour les besoins de leurs traitements d'information, le deuxième type de transaction prend de plus en plus de place.

La nature des calculateurs portables, l'architecture et les propriétés des réseaux sans fil et mobiles ont une influence considérable sur les techniques transactionnelles. En effet, en plus de la fiabilité et de la disponibilité, les performances d'un système transactionnel se mesure principalement par son **taux de productivité (*throughput)*** qui indique le nombre de transactions exécutées avec succès par unité de temps et son **temps de réponse** *(response time)* défini par la durée écoulée entre l'initiation et la validation de la transaction, perçue par le client. L'influence des contraintes de l'environnement mobile et leur variation sur ces performances peuvent se traduire par :

- des annulations fréquentes des transactions, ce qui va réduire le throughput et,

- des coûts d'exécution imprévus et variables (dépense d'énergie, prix de communication, temps d'exécution, etc.).

Le temps d'exécution augmente *(long-lived transaction)* et varie à cause des ressources limitées des unités mobiles ou de la latence des réseaux de communication sans fil et leur débit faible et variable, et aussi à cause des déconnexions. L'augmentation du temps d'exécution, à son tour, prolonge la consommation des ressources locales, en particulier elle entraîne une augmentation de la dépense d'énergie, ainsi que le coût de communication. Le coût de communication sans fils est généralement élevé (en particulier dans le cas des réseaux cellulaires à grande échelle) et peut être très variable selon la technologie de réseau sans fil utilisée et selon la politique économique du fournisseur.

En plus des coupures qu'elle peut induire lors de passage par des zones de non couverture, la mobilité (ou le mouvement) des unités mobiles peut être à l'origine d'autre perturbation concernant l'exécution des transactions lors des changements de cellules *(handoff* ou *handover)*. En effet, la transaction mobile peut s'exécuter séquentiellement sur plusieurs

stations base et éventuellement sur plusieurs ensembles de données dépendant du mouvement des unités mobiles. Le calcul et le contrôle peuvent migrer durant la mobilité de l'utilisateur.

La variabilité des coûts d'exécution des transactions mobiles ne peut être contrôlée que si les techniques transactionnelles conçues pour cet environnement prennent en compte la variabilité des contraintes et des caractéristiques de l'environnement et s'y adaptent en choisissant le schéma d'exécution et les propriétés qui conviennent le mieux à l'application ou à l'utilisateur. Ce problème n'a pas été suffisamment abordé et reste encore ouvert.

Les techniques transactionnelles traditionnelles assurant les propriétés ACID ont été conçues pour un environnement où les défaillances (de communication ou de sites) sont traitées comme des exceptions par les mécanismes de résistance aux fautes et recouvrement. Ces techniques traitent les déconnexions comme des défaillances et conduisent donc à un taux élevé d'annulations. Etant donné que les déconnexions sont des caractéristiques intrinsèques à un système mobile et font parties du fonctionnement normal de celui-ci, les techniques transactionnelles pour un système mobile doivent être conçues pour les supporter. C'est ce qui ressort des contributions dans le domaine des transactions mobiles (chapitre 2).

1.3.3 Modèle d'exécution de transaction mobile

Dans le reste du document, pour plus de clarté et de précision, nous nous plaçons dans un contexte de base de données même si comme nous l'avons signalé plus haut, les mécanismes transactionnels connaissent une utilisation de plus en plus étendue à de nombreux domaines d'applications. La notion d'autonomie locale en environnement mobile est manifestée dans l'habilité d'un poste mobile à continuer d'opérer de façon indépendante lorsqu'il est déconnecté du réseau [Chr93].

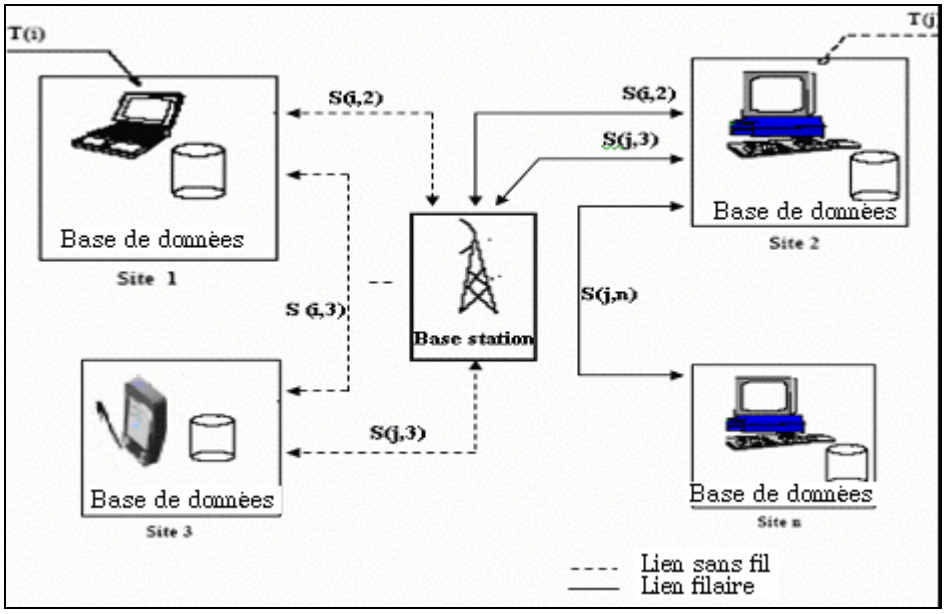


Figure 1.5-Un environnement d'exécution de transaction mobile

Dans un modèle d'exécution de transaction mobile, les postes fixes sont des serveurs caractérisés par un espace de stockage et une capacité de traitement importants et hébergent des bases de données ou des SGBDs. Les postes mobiles sont connectés, de façon intermittente au réseau. Selon la capacité des mobiles, ils peuvent héberger ou non une base de données. Durant son déplacement, l'utilisateur mobile peut envoyer des requêtes et mettre à jour ses données locales (base de données, cache, …).

Par exemple, dans la figure 1.5, la transaction mobile T_i (respectivement T_j) est décomposée en sous-transactions *S(k,l)* pour accéder aux données localisées sur les sites 1, 2 et 3 (respectivement 2, 3 et n); *k* indique l'identifiant de la transaction, *l* indique le site participant de la transaction.

Une transaction mobile peut être définie comme étant une transaction dont l'exécution fait participer au moins une unité mobile. Il en découle les scénarios ou modèles d'exécution suivants [Ser04]:

1. *Exécution complète sur des unités fixes.* La transaction est initiée par une UM mais elle est exécutée sur la partie fixe du réseau. Dans ce cas, des techniques traditionnelles peuvent être utilisés pour la gestion des transactions mobiles [DG00, KK00, YZ94a, YZ94b, DHB97]. Seuls les résultats du traitement de la transaction sont livrés à l'unité mobile.

2. *Exécution complète sur l'unité mobile*. La transaction est initiée et exécutée sur l'UM, on parle d'*exécution autonome* car elle permet de poursuivre les activités sur des données embarquées même durant les périodes de déconnexion. Cependant, des techniques de *synchronisation* ou *réconciliation* des données mises à jour localement avec les copies principales du serveur sont indispensables [LH02, MB01, WC99, WC95, LS95]. De plus, les techniques de gestion de transactions mobiles doivent chercher à optimiser l'utilisation des ressources locales (mémoire, CPU, batterie,…) des unités mobiles [PBVB01, Vin02].

3. *Exécution répartie entre l'unité mobile et des unités fixes*. Dans ce cas, et en plus de l'optimisation des ressources, les caractéristiques de la communication sans fil doivent être prises en compte par exemple en limitant leur utilisation par l'exécution locale de certaines fonctions.

4. *Exécution répartie sur des unités mobiles*. Ce modèle peut offrir une approche pair-à-pair qui permet à une UM de jouer le rôle de serveur pour d'autres UMs qui se trouvent dans son voisinage. On peut imaginer un exemple où deux vendeurs itinérants travaillant dans la même zone partageant des données. Ce modèle nécessite, en plus des techniques de découverte et localisation des voisins mobiles, la répartition dynamique du rôle et des tâches pour chaque mobile. La station base peut jouer un rôle central dans ce modèle, mais si elle n'est pas utilisée, on parle alors d'une approche point à point adaptée aux réseaux de type *Ad hoc* qui nous ramène au scénario 6 ci-dessous.

5. *Exécution répartie entre plusieurs unités mobiles et unités fixes*. Ce modèle convient bien au travail coopératif et aux systèmes multibases de données.

6. *Exécution sur plusieurs unités mobiles en mode ad hoc*. Bien que la majeure partie de nos contributions dans cette thèse concerne les réseaux mobiles avec infrastructure, cependant, nous avons dans certaines parties évoquées, le cas où l'infrastructure de support est ad hoc. Dans ce cas, la transaction est répartie entre plusieurs unités mobiles sans support d'unités fixes ou stations bases.

Une exécution répartie mobile des transactions (modèles 3, 4, 5 et 6) requiert une attention particulière pour l'optimisation des ressources locales restreintes des UMs et aussi la prise en compte des caractéristiques et des contraintes induites par les communications sans fil.

Comme nous allons le voir dans le chapitre 2, les propositions sont pour la plupart basées sur les modèles 1 et 2. Les modèles répartis (3, 4, 5 et 6) permettent de prendre en charge une large gamme d'unités mobiles des plus légères, n'ayant pas des ressources lui permettant de participer dans l'exécution d'une transaction, aux plus riches pouvant jouer le rôle de serveurs à d'autres unités. De même une large gamme d'applications peut être implémentée grâce aux modèles répartis qui offrent la possibilité d'exécutions coopératives.

Selon le modèle, les communications sans fil sont sollicitées différemment. Pendant l'exécution d'une transaction locale ou sur la partie fixe la communication n'est pas obligatoire, par contre pendant une exécution répartie, elle est nécessaire au moins temporairement.

1.4 Conclusion

Dans ce chapitre nous avons présenté les concepts de base du traitement transactionnel afin de faciliter la compréhension de la suite du document. Les problèmes induits par les caractéristiques des environnements mobiles et leur impact sur le traitement transactionnel permettent de mieux comprendre les motivations des recherches menées dans le domaine et des travaux et objectifs de cette thèse. Le chapitre suivant présente justement un état de l'art des travaux de recherche sur les transactions et la mobilité.

Chapitre II

Gestion des transactions en environnement mobile

Faced with the choice between changing one's mind
and proving that there is no need to do so, almost
everyone gets busy on the proof.
--John Kenneth Galbraith

Le besoin d'accès et de traitement de l'information dans les environnements mobiles présente un important défi pour la recherche. En plus des problèmes soulevés par les avancées réalisées en matière de communication sans fil et de terminaux portables, des techniques nouvelles de gestion des données sont nécessaires pour offrir une interaction adéquate entre les unités fixes et les unités mobiles dans un système d'information globale. Un aspect clé du domaine de gestion des données est d'offrir une infrastructure de traitement de transaction flexible et puissante qui prend en compte les spécificités du calcul mobile. Ce chapitre donne un aperçu sur l'évolution des techniques transactionnelles réparties vers la prise en charge de la mobilité en mettant l'accent sur les modèles de transaction vu que ces derniers constituent la plus grande partie de la littérature du domaine depuis le début des années 90s. Dans les sections de ce chapitre, un état de l'art des propositions académiques pour la résolution des problèmes des transactions mobiles ainsi que quelques solutions commercialisées sont présentés. Puis l'analyse a été étendue afin d'inclure les travaux touchant au traitement transactionnel dans des environnements mobiles spécifiques tels que les réseaux ad hoc et les environnements de diffusion. Enfin de ce chapitre nous introduisons l'adaptation dans les sytèmes transactionnels. Le but de la discussion est de situer la terminologie relative à ce domaine ainsi que les objectifs fixés dans cette thèse, leurs motivations et leur positionnement par rapport aux travaux liés.

2.1 Les modèles de transactions

Les systèmes transactionnels sont aujourd'hui au coeur de la majorité des architectures informatiques [BN97, WV02]: systèmes de gestion de bases de données, systèmes d'exploitation répartis, moniteurs transactionnels. La notion de transaction joue un rôle fondamental, aussi bien dans les applications informatiques traditionnelles de télécommunication, de contrôle de processus industriel, de finance, ou encore de gestion d'agences de voyages, que dans les nouvelles applications, telles que le commerce électronique sur le Web et les Workflow. Ces applications sont basées sur des systèmes

d'informations complexes dans lesquels plusieurs programmes (ou utilisateurs) se partagent des informations à travers des réseaux. Dans ce contexte, la notion de transaction répond au besoin fondamental de simplifier le problème du maintien de la cohérence des informations en cas d'accès concurrents ou de défaillances [WV02].

2.1.1 Evolution du modèle de transaction

Un **modèle de transaction** est constitué de l'ensemble des **propriétés structurelles** et **comportementales** qui doivent être respectées par une transaction. Les propriétés structurelles précisent la structure de la transaction et la nature des actions la composant ainsi que la manière avec laquelle elles sont organisées. Les propriétés comportementales sont liées aux exécutions de la transaction et portent sur trois aspects : la manière et le degré de préservation des propriétés ACID, les dépendances d'exécution intra transaction, et les dépendances d'exécution inter-transactions [Med95].

2.1.1.1 Le modèle linéaire

Dans le **modèle linéaire** *(Flat transaction)* ou transaction plate, une transaction est une séquence *indivisible* d'actions élémentaires qui vérifie les propriétés ACID. Les objets manipulés dans une transaction linéaire classique sont élémentaires (de structure simple et de petite taille) qui ne sont manipulables qu'à travers des actions primitives de Lecture/Ecriture (notées R/W). Ces actions ne traduisent pas la sémantique des opérations dont elles sont la représentation. Afin de pouvoir exploiter la connaissance sémantique des opérations d'une transaction, le modèle linéaire a été enrichi donnant naissance **au modèle avec opérations typées**. L'apparition de ce modèle coïncide avec celle des types abstraits de données. Un objet typé, défini comme une instance d'un type abstrait, est caractérisé par des variables constituant son état interne et un ensemble d'opérations spécifiques du type (opérations typées) pour la manipulation de cet état et dont l'exécution se traduit par des opérations R/W accédant à l'état de l'objet et/ou d'autres opérations typées. Une transaction respectant ce modèle peut être vue comme une séquence indivisible d'opérations sur des objets typés. Ces opérations sont donc d'un plus haut niveau sémantique.

Par exemple, le compte bancaire de notre exemple de virement (chapitre 1) peut être défini comme un type abstrait Compte dont l'état contient le couple d'information (Numéro, Solde) et dont l'interface est constituée des opérations Consulter, Débiter, Créditer.

Le modèle avec opérations typées a été motivé par le besoin d'affiner le contrôle des accès concurrents entre transactions [WS92, BBK91]. En effet, les situations de conflit peuvent être redéfinies en fonction de la connaissance sémantique des objets et des opérations sur les objets. Le principe directeur est fondé sur les notions de commutativité des opérations.

Dans l'exemple du compte bancaire, on peut spécifier que l'opération Consulter est compatible avec Débiter et Créditer (elles peuvent s'exécuter de façon concurrente) et que Débiter et Créditer sont des opérations commutatives.

Une autre extension au modèle linéaire a consisté en l'introduction de "points de reprise" ou *(savepoints)* [ABC+79] qui peuvent servir de points de lancement en cas de pannes. Ces points de reprise permettent d'éviter d'avoir à refaire la transaction à partir du début.

2.1.1.2 Les modèles avancés ou étendus

En vue d'une part, prendre en compte les caractéristiques des applications avancées et d'autre part, s'adapter à la répartition, les systèmes transactionnels ont connu de nombreuses propositions de **modèles de transactions avancés/étendus** qui ont été motivés par le soucis de [BBK91, WS92, Med95] :

- fournir un support pour les activités de longue durée de vie,
- modéliser le parallélisme à l'intérieur des transactions (intra- transactions),
- augmenter le parallélisme inter-transactions,
- supporter la coopération entre membres d'un groupe d'utilisateurs d'une application avancée (CAO, applications coopératives, etc.),
- traduire la sémantique des applications supportées,
- mieux s'adapter au modèle de données orientées objet,
- affiner l'unité de rejet/reprise d'une transaction,
- s'adapter aux transactions conversationnelles,
- s'adapter à de multiples systèmes autonomes dans un environnement de SGBDs fédérés.

Nous présentons dans cette section les principaux modèles parmi un grand nombre de propositions qui souvent constituent des variantes à ces principaux modèles.

2.1.1.3 Le modèle emboîté

Le modèle de **transaction emboîtée** et sa variante le modèle **multi niveau** constituent les extensions les plus importantes apportées au modèle transactionnel. Le modèle emboîté *(Nested Transactions)* a été motivé par le souci de mieux intégrer le parallélisme et les points de reprise à l'intérieur d'une transaction [WS92, BBK91]. Dans ce modèle une transaction est découpée en unités plus fines, appelées **sous-transactions**, pouvant s'exécuter séquentiellement ou en parallèle sur un ou plusieurs sites. Chaque sous-transaction est définie de manière récursive comme une transaction de niveau inférieur ; elle s'exécute comme une opération pour le compte d'une transaction de niveau supérieur, et comme une transaction pouvant initier à son tour d'autres opérations de lecture/écriture et/ou d'autres sous-transactions. Cette définition permet de schématiser une transaction emboîtée par un arbre dont la racine désigne le nom de la transaction racine et les feuilles correspondent aux opérations de lecture/écriture.

Le modèle décrit ici présente des propriétés intéressantes ; en premier lieu il permet de structurer une transaction de longue durée de vie et/ou manipulant des objets complexes et de grande taille, en unités de reprise indépendantes tout en assurant les propriétés fondamentales de sérialisation et d'atomicité globale. En outre il est bien adapté à la répartition dans la mesure où les sous-transactions peuvent s'exécuter en parallèle sur des machines différentes.

2.1.1.4 Le modèle emboîté fermé

Dans ce modèle emboîté de base, la validation d'une transaction parent ne peut avoir lieu avant la terminaison de toutes ses sous-transactions. Ce qui signifie que l'atomicité globale s'applique à la transaction la plus externe. L'annulation d'une transaction parent entraîne l'annulation de toute sa descendance, par contre l'annulation d'une sous-transaction, permet à son ascendant d'opter pour la ré- exécution de la même ou pour le choix d'une autre sous-transaction. Il découle de ceci que seule la transaction racine respecte la propriété de durabilité telle que définie par le modèle linéaire. L'isolation est assurée par la racine ; ce qui ne permet pas la visualisation des résultats aux autres transactions. Cependant cette propriété est relâchée entre sous-transactions ayant le même ancêtre. En résumé, les propriétés ACID sont assurées au niveau global, par contre, au niveau des sous-transactions seules les propriétés A et I le sont.

2.1.1.5 Le modèle emboîté ouvert

Ce modèle *(Open Nested)* se distingue par rapport au modèle emboîté fermé principalement par une granularité d'isolation réduite. En effet, les mises à jour effectuées par une sous-transaction sont rendues visibles aux autres transactions du système immédiatement après sa validation et avant la validation de la transaction racine (relâchement de l'isolation globale). D'une part, ceci augmente le degré de concurrence mais alors il n'est plus possible de restaurer l'état des données simplement par le fait de défaire les effets d'une transaction rejetée ; étant rendus visibles, ceux-ci peuvent avoir été utilisés par d'autres transactions. Ce problème a été résolu par l'introduction d'opérations dites de compensation. Une **transaction de compensation** est associée à chaque sous-transaction. La transaction de compensation a pour rôle de défaire d'un point de vue sémantique toutes les actions de la transaction compensée, l'état des données existant avant l'exécution de celle-ci n'étant pas obligatoirement restauré. D'autre part, il est évident que l'atomicité ne peut être assurée sans les transactions de compensation. Cependant, une technique de compensation ne garantit pas l'atomicité d'une transaction car ses résultats peuvent être consultés par d'autres transactions entre le moment de la validation et le moment où sa transaction de compensation démarre. On parle alors d'Atomicité sémantique. Le modèle emboîté ouvert permet aussi de « relâcher » la propriété de durabilité grâce à un attribut associé à chaque sous-transaction indiquant si celle-ci est **durable** ou **non durable**. Contrairement à une transaction non durable, les effets d'une transaction durable validée ne peuvent pas être annulés même si un de ses ancêtres est rejeté. Le « relâchement » de la propriété de durabilité a pour conséquence le « relâchement » de la propriété d'atomicité vu que le rejet d'une transaction ne permet pas l'annulation des effets de ses sous-transactions durables.

2.1.1.6 Le modèle multi niveaux

Comme pour le modèle emboîté ouvert, le modèle **multi niveau**x permet [WS92, SSW95] de prendre en charge plusieurs aspects dont le parallélisme inter- et intra- transactions, la définition de points de reprise à l'intérieur des transactions. Il permet en plus de faciliter la manipulation d'objets complexes dans les systèmes où l'accès à un objet nécessite des accès à d'autres objets. Dans ce type de système une conception modulaire par niveaux d'abstraction successifs peut s'avérer intéressante. Une transaction multi niveaux peut être schématisée par une structure d'arbre (figure 2.1).

Par exemple, un SGBD relationnel peut être considéré comme un système à 4 niveaux (L0, L1, L2, L3) tels que : L2 représente les relations du système dont les opérations sont Insérer, Effacer, Consulter. L1 représente les enregistrements dont les opérations sont Insérer, Effacer, Consulter. L0 désigne les pages du système avec les opérations de Lecture/Ecriture.

Le niveau L3 représente en fait le niveau transactions du système. Soit la transaction T accédant aux tuples de deux relation R1 et R2 :

Début T
Insérer dans R1 (champ 1= a, champ =b) ;
Effacer dans R2 (champ 1= d, champ =d).
Fin T

Cette transaction peut être schématisée par l'arbre abstrait de la figure 2.1 (x, y et z étant des enregistrements vérifiant la condition champ 1 =d et champs 2=d).

Le modèle multi niveaux présente des différences par rapport au modèle emboîté ouvert : l'arbre abstrait d'une transaction multi niveaux est équilibré, le nombre de niveaux d'un

système multi niveaux est fixe et connu à priori, l'état d'un système multi niveaux dépend du niveau considéré.

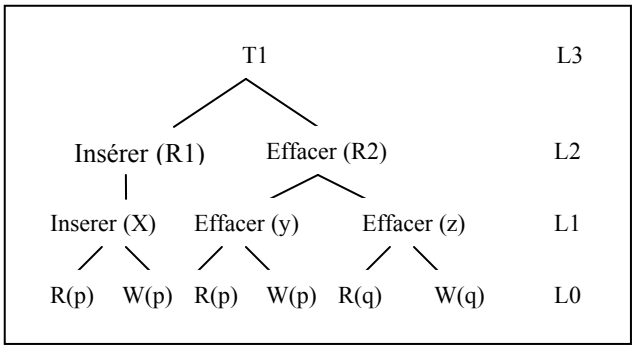


Figure 2.1-Exemple d'une transaction multi niveaux

2.1.1.7 Le modèle Saga

Ce modèle a été conçu spécialement dans le but de pallier aux problèmes que posent les transactions de longue durée de vie. L'idée de base consiste à réduire l'unité d'isolation. Une saga est une transaction emboîtée ouverte à deux niveaux : la racine étant la saga composée au deuxième niveau de transactions simples [GK87]. Les sous-transactions d'une saga peuvent s'exécuter séquentiellement ou en parallèle. L'isolation n'est assurée qu'au niveau des sous-transactions car les résultats partiels peuvent être rendus visibles aux autres sagas. La consistance est vérifiée par chaque sous-transaction. L'atomicité est assurée au niveau de la Sagas, ses sous-transactions ne pouvant qu'être toutes exécutées sinon aucune ne l'est. Cette atomicité est réalisée par l'association d'une sous-transaction de compensation à chaque **sous-transaction.** Une transaction de compensation est associée à chaque sous-transaction. L'annulation d'une sous-transaction entraîne l'annulation de la Saga et l'exécution des compensations dans l'ordre inverse de la validation des sous-transactions correspondantes. Lorsque l'isolation est relâchée et que la compensation est utilisée on parle d'atomicité sémantique.

2.1.1.8 Le modèle flexible

Il s'agit de l'un des modèles conçus pour s'adapter aux systèmes **multibases** de données [ELR90, ZNBB94]. Dans ces systèmes deux types d'application coexistent : les applications locales associées aux systèmes existants, et les applications globales provenant de la multibases de données et accédant à plusieurs de ces systèmes en même temps. D'où les deux types de transactions : locales et globales. Le modèle flexible structure une transaction globale en la décomposant en un ensemble de tâches pouvant être réalisées chacune de manière différente. Ce qui a donné naissance aux **alternatives** : plusieurs sous-transactions fonctionnellement équivalentes sont associées à chaque tâche. Si une sous-transaction ne peut être exécutée, une sous-transaction équivalente sera lancée. Pour une même transaction flexible plusieurs chemins d'exécution (alternatives) sont possibles. Des dépendances d'exécutions (de défaillance, de succès, d'initiation ou externe) entre les sous-transactions peuvent être spécifiées.

Ce modèle propose la **semi atomicité** qui permet de terminer un sous ensemble de sous-transactions avec succès. Des sous-transactions peuvent alors annuler sans causer l'annulation de la transaction flexible et peuvent aussi valider ou non indépendamment de la transaction flexible. En plus de la compensation qui est facultative, des techniques de récupération (refaire, ré- exécuter) peuvent être utilisées pour la reprise après pannes. La compensation et la ré- exécution appellent au relâchement de la sérialisabilité.

2.1.1.9 DOM

Le modèle des transactions DOM *(Distributed Object Management)* [BOH+92] est destiné aux transactions de longue durée dans les systèmes multibases. Une transaction DOM est une transaction composée d'un ensemble de sous-transactions *(appelés top transactions)* emboîtées ouvertes. Elles peuvent être vitales ou non. L'annulation d'une transaction vitale entraîne l'annulation de la transaction DOM, ce qui n'est pas le cas lors de l'annulation des transactions non vitales. Des dépendances de précédence peuvent être définies entre les sous-transactions. Ainsi, les sous-transactions peuvent être exécutées et validées de manière indépendante, sauf si des dépendances de précédence sont spécifiées. A chaque sous-transaction doit être associée une transaction de compensation. Comme pour les Sagas, en cas d'annulation, les transactions de compensation sont exécutées dans l'ordre inverse de validation des sous-transactions associées. Des transactions de contingence peuvent aussi être exécutées lorsqu'une condition de défaillance (annulation ou erreur sémantique) de la sous-transaction correspondante survient.

Au niveau global, les transactions DOM relâchent l'atomicité et l'isolation. Les sous-transactions peuvent valider et rendre visible leurs modifications avant la validation de la transaction DOM.

2.1.2 Analyse et applicabilité à l'environnement mobile

Malgré leurs spécificités, souvent les modèles avancés ont été inspirés par le modèle emboîté ouvert. Tous ces modèles de transaction avancés ont établi un consensus sur le fait de subdiviser une transaction en unités plus réduites, en l'occurrence les sous-transactions. Ceci permet de relâcher et de ramener le support de certaines propriétés au niveau des sous-transactions au lieu de la transaction globale. Les caractéristiques des sous-transactions peuvent être résumées en [ST97] :

- Ouverte vs fermée : ces propriétés sont utilisées pour indiquer si les résultats des sous-transaction sont rendus visibles seulement aux parents ou à toutes les transactions.

- Vitale vs non vitale : une transaction parent peut être dépendante de l'annulation d'une transaction enfant. Si une transaction vitale est annulée, la transaction parent doit aussi être annulée. Par contre, l'annulation d'une transaction non vitale ne conduit pas à l'annulation de son parent.

- Dépendante vs. Indépendante : par analogie, la terminaison d'un enfant peut être dépendante de la terminaison d'un parent. Une transaction enfant dépendante devra être annulée si la transaction parent s'annule.
- Compensable vs. Non compensable : les transactions peuvent être différenciées par le fait d'avoir des effets compensables ou non. Une transaction est compensable s'il existe une autre transaction qui peut défaire sémantiquement ses effets.
- Substituable vs. Non substituable : une transaction est substituable s'il existe une transaction appelée transaction contingente ou alternative qui peut être exécutée à sa place.
- Temporelle : la transaction est tenue d'observer des contraintes de temps, telle qu'une date limite de terminaison.

L'utilisation de transactions emboîtées permet, grâce à une décomposition adéquate des sous-transactions, d'augmenter le parallélisme d'exécution et de réduire l'annulation des transactions. En effet, les sous-transactions soeurs (ou parent et enfant) peuvent être exécutées en parallèle. L'annulation d'une sous-transaction peut être gérée sans entraîner nécessairement l'annulation de la transaction globale. La table suivante (table 2.1) récapitule les propriétés des principaux modèles présentés dans cette section. Rappelons que la propriété d'autonomie est

liée à la capacité des UMs à continuer leur activité de manière indépendante et l'hétérogénéité concerne le type d'UM, d'UF et de gestionnaires de données (chapitre 1).

Table 2.1-Synthèse des modèles de transaction avancés

	Emboîté fermé	**Emboîté ouvert**	**Multi niveaux**	**Saga**	**Flexible**	**DOM**
Ouvert	-	X	X	X	X	X
Fermé	X	-	-	-	-	-
Non vitale	X	X	X	-	X	Variable
Dépendante	X	-	-	X	X	X
Non- dépend.	-	X	X	-	-	-
Compensable	-	X	X	X	Possible	X
Substituable	X	-	-	-	X	X
Atomicité	Racine Sous-tr.	Racine Sous-tr.	Racine Sous-tr.	Racine Sous-tr.	Sous-tr.	Sous-tr.
Isolation	Racine Sous-tr.	Sous-tr.	Sous-tr.	Sous-tr.	Sous-tr.	Sous-tr.
Consistance	Racine	Racine	Racine	Racine	Racine	Racine
Durabilité	Racine	Racine Sous-tr.	Racine .	Racine	Racine	Sous-tr.
Autonomie	-	X	X	X	X	X
Hétérogénéité	Possible	X	X	X	X	X

Dans le modèle de transaction plate il n'y a aucun moyen de valider ou annuler des parties de transactions. En cas de problème, la transaction est annulée et ré- exécutée depuis le début. La première extension au modèle plat a été l'introduction des "points de reprise" *(savepoints)* [WV02]. Lorsqu'une UM participe à l'exécution d'une transaction plate, on peut lui permettre de se déconnecter une fois sa part de l'exécution terminée et sa reconnexion lui permettra de connaître le résultat. Un point de reprise pourrait être inséré là où la portion de transaction se termine sur l'UM; ainsi cette dernière n'aura pas à être contactée pour refaire la portion qu'elle a déjà exécutée.

Dans le modèle emboîté, la décomposition d'une transaction en sous-transactions peut être très avantageuse pour l'environnement mobile. En utilisant ce modèle, une transaction peut, par exemple, exécuter en parallèle une sous-transaction dans une unité mobile et une autre dans une unité fixe. Cependant, dans le cas du modèle fermé, les sous-transactions doivent attendre la validation globale avant de partager leurs modifications avec d'autres transactions. Cette contrainte peut limiter les performances du système mobile, où l'autonomie d'exécution est une nécessité. Cette autonomie s'impose d'elle même car elle concerne à la fois les unités mobiles, qui lors des déconnexions doivent pouvoir continuer leur activité et en plus doivent limiter leurs échanges avec d'autre composants du système pour économiser l'énergie, et les SGBD locaux, qui participent à l'exécution des différentes branches de la transaction mobile. Les transactions emboîtées ouvertes paraissent plus adaptées pour l'environnement mobile du fait que les sous-transactions peuvent être exécutées sur des unités mobiles, valider et partager leurs modifications avec d'autres transactions. Ainsi, les unités mobiles profitent d'une certaine autonomie sans bloquer le système et par conséquent, l'hétérogénéité est également possible.

En tant que modèles de transactions emboîtées ouvertes, les Sagas, les transactions multi niveaux, flexibles et les DOM peuvent être appliquées à l'environnement mobile. Cependant, il faut étudier de plus près leur adaptation aux limitations de l'environnement mobile. Dans Saga, l'exécution des transactions de compensation peut ne pas être possible lors des déconnexions des unités mobiles. Par ailleurs, pour tous les modèles il n'est pas toujours possible de définir ou de garantir l'exécution de transactions de compensation. Les alternatives dans le modèle flexible paraissent intéressantes car elles montrent une grande tolérance aux pannes ce qui peut profiter à l'environnement mobile sujet à de fréquentes

variations. En effet, on peut penser choisir dynamiquement une alternative d'exécution en fonction de l'état actuel des paramètres de l'environnement. Cependant, il faut éviter de traiter les variations du contexte comme des pannes. Les transactions non vitales et les transactions de contingence du modèle DOM pourraient réduire le taux d'annulation des transactions dans les environnements mobiles, mais ce modèle est aussi concerné par la compensation.

La plupart des modèles proposés se focalisent sur l'amélioration du contrôle de concurrence dans le but de supporter les transactions de longue durée de vie dans un environnement multibases. Dans ces modèles la non disponibilité des données (qui peut être fréquente dans l'environnement mobile à cause notamment des déconnexions) n'est pas discutée.

Pour finir, nous pouvons également citer le modèle *Split transactions* [PKH88] qui est proposé pour les applications dont la "fin est ouverte" *(open-ended)* (durée non connue, ayant des actions non prévisibles au début ou ayant des interactions avec d'autres activités). Le principe consiste à diviser une transaction en cours par une opération "Split" en transactions serialisables et à partager ses ressources entres les transactions résultantes de façon à ce que chaque terminaison d'une partie de la transaction, ses résultats et ressources sont libérés et que le travail réalisé ne soit plus affecté par les pannes ultérieures. Une opération inverse appelée *Join* est aussi définie. Ce modèle est adapté à la mobilité mais seulement pour des applications d'un type particulier telle que la CAD.

2.2 Techniques de transaction pour environnements mobiles

Cette section est consacrée à la présentation des travaux de recherche sur les modèles de transactions mobiles et de quelques systèmes commercialisés. Les travaux de recherche sont présentés dans l'ordre chronologique de leur publication.

2.2.1 Les propositions académiques

Généralement, les modèles de transaction présentés dans cette section exploitent les résultats des recherches précédentes dans le domaine des modèles et du traitement transactionnel réparti en les adaptant à l'environnement de calcul mobile.

2.2.1.1 Reporting et Co-transactions

Chrysantis [Chr93] considère qu'un environnement de bases de données mobiles est un système multibases où les transactions exécutées sur une UM sont des sous-transactions. Il présente un schéma pour la structuration des transactions mobiles en se basant sur le modèle **imbriqué ouvert** augmenté des notions de transactions **rapporteuses** *(reporting transactions)* et de **co-transactions** *(co-transactions)*. Une transaction parent peut initier quatre types de sous-transactions pouvant s'exécuter sur des SFs ou sur des UMs. Il s'agit de :

- *transactions atomiques compensables*.

- *transaction non compensable* qui au moment de la validation délègue la responsabilité à la transaction mère.

- *transactions rapporteuses* pouvant partager leurs résultats partiels avec la transaction mère durant l'exécution et avoir des transactions de compensation selon qu'elles délèguent ou non la validation des résultats à la transaction mère. Ces transactions peuvent, après la diffusion de leurs résultats, continuer leur exécution et valider de manière indépendante.

- et enfin, *les co-transactions* qui sont des transactions rapporteuses spéciales se comportant comme des co-routines. Elles sont suspendues par d'autres transactions au

moment du partage des résultats. Elles reprennent leur exécution à partir de ce point aussitôt qu'elles sont reprises tout en conservant leur état à travers les exécutions.

Ce modèle traite de la migration et est conçu pour des UMs connectées en continu au réseau même en mouvement. En effet, les transactions rapporteuses et les co-transactions peuvent échanger des messages et aussi reloger leur exécution d'une station base (SB) à une autre afin de minimiser l'utilisation de la bande passante. La technique de relogement n'a pas été expliquée. L'atomicité sémantique peut être assurée grâce à des transactions de compensation. Ce modèle suppose par ailleurs l'existence d'un SGBD global à chaque station base. L'auteur affirme que ce modèle minimise le coût des communications filaires et non filaires. Vu qu'une connexion continue est exigée, ce modèle ne prend pas en charge les déconnexions. D'autre part, il est clair qu'un mécanisme de réplication doit être considéré afin de permettre les exécutions sur l'UM.

2.2.1.2 Le modèle de validation de transaction par Batch (par lot)

Dans [Nar94] on s'intéresse à la faiblesse de la bande passante et à la latence des communications sans fils. On propose alors d'exécuter la transaction entière localement sur l'unité mobile moyennant des copies cachées des données. Pour sa validation, la transaction entière est envoyée à un serveur de bases de données situé sur un site fixe pour le reste des opérations. Le serveur négociera la validation avec les autres serveurs du système. Un accusé de réception positif est envoyé à l'unité mobile si la validation a lieu. Si la transaction venait à être annulée pour avoir utilisé une version périmée de l'une des données manipulées, le serveur envoie la dernière version de la donnée en question à l'unité mobile. Cette dernière peut ré- exécuter la transaction. Cette méthode est optimiste car elle suppose qu'il y a peu de chances pour que les transactions ne soient pas validées après leur transfert sur le serveur. On peut diminuer les chances d'avoir des conflits et de voir trop de transactions annulées en réduisant la granularité des objets manipulés.

2.2.1.3 Isolation-Only Transactions (IOT)

Le modèle IOT [LS94, LS95] a été développé pour les conflits de lecture-écriture de fichiers UNIX à partir d'UMs par un ensemble d'opérations structurées en transactions. Ce modèle développé dans le contexte du système de fichiers CODA [SKK+90] permet des exécutions déconnectées sur l'UM sur des copies de fichiers grâce à des transactions dites de seconde classe. Les résultats de ces transactions ne sont rendus visibles au serveur qu'après la validation. A la reconnexion, ces transactions sortent de l'état d'attente et leurs résultats sont validés selon les critères désirés (sérialisabilité locale, globale ou certification). Le serveur vérifie leur serialisabilité avec toutes les transactions validées. Si la validation est confirmée, alors l'UM valide ses résultats vers l'UF, sinon les conflits doivent être résolus automatiquement ou manuellement par l'UM avant de retenter la validation. D'autres transactions dites de première classe ne s'exécutent pas sur des données partitionnées grâce au maintient de la connexion avec le serveur pour chaque fichier accédé. Ces transactions valident immédiatement après avoir été exécutées. IOT est un modèle linéaire ou plat qui ne satisfait que l'isolation. Les déconnexions sont supportées grâce au cache, mais la mobilité n'est pas prise en charge.

2.2.1.4 Multi Database System Transaction Processing Manager (MDSTPM)

L'approche MDSTPM [YZ94a, YZ94b] propose une architecture permettant de soumettre les transactions à partir des UMs pour leur exécution sur un système multibases hétérogène. Une SB est désignée à l'avance en tant que coordinateur global *(Global Transaction Manager Coordinator, GTMC)* de l'UM. Une fois qu'elle reçoit la requête d'initiation d'une transaction

globale émanant du mobile, elle se charge de la gestion et de la coordination de l'exécution de la transaction pendant que l'UM est libre de se déconnecter. Un mécanisme de file de messages permet de gérer les échanges entre l'UM et la SB de façon asynchrone. Lorsque l'UM se reconnecte elle récupère les résultats chez la SB correspondante. Si la transaction s'exécute sur plusieurs SBs, ces dernières sont nommées Participant Globaux *(Global Transactions Manager Participant, GTMP)*. Le MDSTPM fait interface entre l'UM et les SGBDs locaux qui sont responsables de la gestion des sous-transactions locales et de leurs propriétés. La mobilité n'est pas traitée de façon explicite, l'UM se déconnecte après soumission de sa requête vers la SB coordinatrice et doit se reconnecter sur cette même SB pour retrouver ses résultats.

2.2.1.5 Clustering

Ce modèle [PB94a, PB94c] est conçu pour assurer la cohérence de la base de donnée dans un système réparti. Il propose un schéma de duplication afin de permettre à l'UM de traiter des transactions en s'adaptant aux variation des communications. La base de données est divisée en grappes *(clusters)* regroupant chacune des données sémantiquement liées ou localisées sur des sites voisins. Lorsque l'UM se déconnecte elle devient une grappe. La cohérence stricte est exigée pour les données internes à un cluster et des degrés de cohérence sont définis sur des données dupliquées et localisées sur des clusters différents. Le degré de cohérence peut dépendre de la qualité de la liaison entre une unité mobile et sa station base. Lorsque la connexion est faible, les utilisateurs peuvent tolérer un degré d'incohérence plus élevé (voir aussi [IB93]). La transaction mobile sera décomposée en transactions faibles et en transactions strictes. La décomposition devra se faire sur la base du degré de consistance requise. Les opérations des transactions strictes s'exécutent lorsque la connexion est forte à l'intérieur d'une grappe. Les transactions faibles s'exécutent et sont validé en local lorsque la connexion est faible ou lorsque l'UM est déconnectée et travaille sur la copie locale. Les mises à jour sont réconciliées à la reconnexion sur les autres clusters après une validation globale. L'accès se fera globalement (en dehors du cluster) à des données cohérentes à l'aide de transactions strictes. Le concept de transactions suppléantes (transactions proxies) a été introduit pour assurer la durabilité des effets du calcul effectué sur l'UM vu sa vulnérabilité. A chaque transaction exécutée sur un site mobile, une transaction duale est définie et exécutée sur la station base correspondante. Cette transaction suppléante est considérée comme une sous-transaction de la transaction originale. Ainsi, à chaque soumission d'une transaction à l'UM, sa transaction suppléante est soumise à sa station base. Les transactions suppléantes n'incluent que les mises à jour de leurs transactions d'origine. Le déplacement de l'unité mobile implique que le relogement des transactions suppléantes peut être envisagé. Les transactions suppléantes pourraient également être utilisées pour réaliser des opérations de sauvegardes périodiques (correspondant à la notion de points de reprise) [PB94a].

2.2.1.6 Semantic-based

Ce modèle considère une transaction mobile comme étant une transaction de longue durée de vie caractérisée par des délais de transmission longs et des déconnexions non prévisibles. Dans [WC95] on propose d'exploiter la sémantique des objets (en particulier la commutativité des opérations) afin d'améliorer la gestion des accès concurrents sans pour autant compromettre la propriété de sérialisation (isolation) et ceci pour une meilleure autonomie des UMs. Le traitement de transactions mobiles est ramené au problème de concurrence et de cohérence du cache. La fragmentation des objets permet de découper les objets de grande taille ou complexes en fragments plus petits afin d'offrir une meilleure concurrence et de pallier aux limitations de stockage sur l'UM. Une base de données située sur la partie statique envoie des fragments vers l'UM à sa demande. Deux paramètres sont inclus dans cette

demande ; l'un indique les données à charger sur l'UM et l'autre spécifie les contraintes nécessaires à la préservation de la cohérence de l'objet entier. Le fragment chargé sur l'UM est une copie exclusive qu'une transaction peut manipuler localement. A la terminaison de la transaction, l'UM retourne les fragments au serveur. Ces fragments sont regroupés de nouveau par une opération de fusion exécutée au niveau du serveur. Lorsque les fragments peuvent être combinés dans un ordre quelconque ils sont dits « recomposables » *(reordable objects)*. Du moment qu'un serveur unique de base de données est présupposé, les propriétés ACID peuvent être maintenues. Les ensembles, les structures de données de type pile ou file d'attente sont des exemples d'objets fragmentables.

2.2.1.7 Two-tiers Replication

Two-tier replication [GHPS96] suggère un schéma de réplication et une méthode d'exécution des transactions pour des MUs qui se connectent de façon occasionnelle. Une analyse des schémas de réplication impatients *(eager)* et paresseux *(lazy)* a conduit les auteurs à conclure que les schémas impatients ne conviennent pas aux environnements mobiles principalement parce qu'il n'est pas possible de permettre des déconnexions. Two-tier réplication est un mécanisme de duplication paresseux. Chaque UM a deux versions pour chaque donnée accédée ; une *version maître* représentant la valeur la plus récente reçue à partir de l'UF n'ayant pas encore été modifiée par les transactions locales et une *version tentative* représentant la modification locale la plus récente qui va subir une validation dès la reconnexion. Deux types de transactions sont supportés : **transactions** *(Base Transactions)* et **tentatives** *(Tentative Transactions)*. Les transactions bases accèdent les versions maîtresses et sont exécutées par des MUs connectées. Les transactions tentatives accèdent les versions tentatives (copies locales) pendant la déconnexion. A la reconnexion, les transactions tentatives sont ré- exécutées comme des transactions bases afin d'obtenir une cohérence globale, sous la coordination de la station base courante. En cas d'échec de ces ré- exécutions, l'MU et l'utilisateur sont informés de l'échec et de sa cause. Dans ce cas, les tentatives sont annulées et les versions tentatives des données sont remplacées par les données originales de la version maîtresse. L'acceptation des résultats de la ré- exécution peut être soumise à des critères tolérant un degré de divergence contrôlée, ce qui permet la persistance des mises à jours locales.

2.2.1.8 Twin-Transactions model

Le modèle Twin-Transaction (TT) [RZ96, RZ98] est proposé pour la gestion de cohérence des copies répliquées entre une UM et les UFs. Il suggère de capturer le processus d'exécution des transactions sur les données répliquées dans un graphe d'histoire qui servirait ultérieurement en conjonction avec le graphe de précédence à la détection des conflits. Le principe consiste à répliquer le processus d'exécution des transactions. A chaque transaction, impliquant un nœud mobile dans son exécution, est associée une image miroir et deux transactions équivalentes (d'où le nom de jumelles) sont alors crées. L'une des transactions s'exécutera sur le site d'où émane la transaction et l'autre sera envoyée sur un *méta objet* qui est une vue logique de l'ensemble des copies de l'objet accédée par la transaction. Si le *méta objet* se trouve sur une UM qui n'est pas disponible (déconnectée, en déplacement) la transaction jumelle sera sauvegardée dans un journal d'histoire et sera exécutée à la reconnexion. Les mises à jour émanant de différents sites possédant une copie de la donnée seront éventuellement reçues. Le *méta objet* est responsable de la synchronisation des répliquas moyennant un ensemble de règles de résolutions de conflits. Le résultat de la synchronisation peut conduire à l'annulation/ compensation de certaines transactions. Le site initiateur d'une transaction peut s'informer sur la probabilité de succès de sa transaction ou exiger une certaine probabilité de succès sans quoi il peut décider d'annuler sa transaction. Le

calcul de la probabilité de succès est fonction de la probabilité de conflit qui peut être déduite à partir des modèles de conflits précédents. Il est clair que le succès de ce modèle dépend du mécanisme d'apprentissage des modèles de conflits et du calcul de la probabilité de conflit qui devraient accompagner cette proposition.

2.2.1.9 Kangaroo

Le modèle **Kangourou** [DHB97, DK99] est le premier modèle qui a été conçu avec le souci de modéliser le **mouvement des transaction**s (leur mobilité ou migration d'une cellule à l'autre). Ce modèle est construit sur une approche multibases de données où la gestion des transactions est toujours réalisée au niveau de la station base (ou d'un site spécifié sur la partie fixe du réseau). Ce modèle est basé sur les modèles de transaction emboîtée ouverte [WS92] et *split* [PKH88]. Un agent d'accès aux données *DAA (Data Access Agent)* est installé sur chaque UM pour s'interfacer entre les UMs et les gestionnaires globaux du système multibases. Le DAA gère l'exécution de la transaction globale mobile ainsi que son mouvement. Le DAA se trouvant sur la SB courante est chargé de la coordination jusqu'à ce que la transaction change de cellule et de SB de rattachement, le DAA de la nouvelle SB prend alors le relais. Ce mouvement entraîne le découpage de la transaction actuelle en deux transactions dites *Joey Transactions (JT)*. Une JT étant une partie de la transaction globale peut elle même être composée de sous-transactions pouvant être locales ou globales. La validation des JTs se fait indépendamment les unes les autres. Deux modes d'exécution sont permis, l'un assurant une atomicité globale annulant la transaction Kangoroo lorsqu'une JT est annulée puis compensée. L'autre appelé mode *Split* requiert la terminaison de la transaction Kangoroo suite à l'annulation d'une JT sans avoir à refaire les JTs déjà validées ; ce mode ne garantie ni l'atomicité ni l'isolation.

2.2.1.10 Pro-Motion

Pro-motion [WC99, WC97] aborde le problème du cache et de la cohérence des données sous les contraintes de déconnexion et de mobilité. Pour permettre l'exécution locale, le concept de ***compact*** comme unité de cache et de contrôle est introduit. Le compact est une abstraction qui encapsule des données, des méthodes (opérations sur les données), des règles de cohérence, des obligations *(deadlines)*, des informations sur l'état courant du compact et des méthodes d'interface qui permettent les interactions entres les compacts et l'UM. Un compact sert d'unité de réplication et de cohérence pour un contrat entre l'UM et les SGBDs locaux. Un Agent Compact est responsable de la gestion des transactions sur l'UM. Le Gestionnaire de Mobilité situé sur la SB est chargé de retrouver les données requises à la demande de l'agent Compact. Un gestionnaire de compacts, localisé sur le serveur de données, interagit avec le gestionnaire de mobilité et les serveurs de données qui le considèrent comme un client courant exécutant une transaction de longue durée. Pro-motion utilise des transactions *emboîtées ouvertes* et *split* [Chr93, RC96]. Les sous-transactions pouvant elles mêmes être des transactions emboîtées, s'exécutent et valident sur les UMs grâce au protocole 2PC. A la reconnexion, les compacts sont soumis au gestionnaire de compact sur l'UF qui crée une *split-transaction* pour la validation du compact. La compensation sera utilisée en cas de non validation d'une transaction validée localement.

2.2.1.11 Pre-serialisation ou Toggle

La technique de gestion de transaction Pre-serialization [DG98, DG00a, DG00b] est basée sur le modèle de transaction multi niveaux *(multi-level)* [Wei91] qu'il étend pour prendre en charge les déconnexions prévisibles ainsi que la migration des transactions. De façon similaire à de nombreuses approches, il suppose l'existence d'un système multibases où les

bases de données locales sont indépendantes l'une de l'autre. Les transactions mobiles sont considérées comme des transactions globales de longue durée (à cause des déconnexions et de la migration) composées par des sous-transactions compensables *(appelés Site Transactions)*. Afin de minimiser les exécutions longues, Pre-serialization permet aux *transactions sites* de valider unilatéralement de manière indépendante de la transaction globale. Ceci relâche les ressources locales au fur et à mesure que les transactions sites terminent leur exécution. Une couche interface de service est fournie au dessus de chaque SGBD local. Cette couche offre les services qui peuvent être invoqués par les transactions. Un gestionnaire de transaction du site *STM (Site Transaction Manager)* supervise les transactions exécutées sur un nœud. Un coordinateur de transaction global *GTC (Global Transaction Coordinator)* réside sur chaque SB. Son rôle est de gérer les déconnexions et les migrations des UMs ainsi que la soumission des requêtes vers le STM correspondant. Les déconnexions sont traitées par l'introduction de nouveaux états de transaction aux états habituels (active, validée, annulée). L'état *Déconnectée* est associé à une déconnexion prévisible et permet à la transaction de continuer son exécution à la reconnexion. L'état suspendu est associé à une déconnexion due à une panne et permet la suspension de la transaction sans pour autant l'annuler jusqu'à ce qu'elle entrave l'exécution d'autres transactions. Ceci permet de réduire des annulations inutiles. Lorsque la transaction globale manifeste sa volonté de valider, son GTC initie un algorithme de validation basé sur un *Graphe Partiel de Sérialization Globale*. Si un cycle est détecté, l'algorithme tente de briser les cycles par l'annulation de ses sous-transactions non vitales ainsi que les sous-transactions suspendues des autres transactions globales. Si un cycle ne peut être résolu, alors la transaction globale est annulée. Sinon elle est validée ou t*oggled*. Le *toggeling* signifie la pré- validation des résultats d'une transaction. Une transaction pré-validée (*toggled)* peut commencer seulement des sous-transactions non vitales, et n'est annulée que dans le cas d'une déconnexion non prévisible. Le GTM offre la possibilité de mouvement entre les SB en maintenant des informations sur des tables adéquates. Les états *suspendue* et *déconnectée* permettent de traiter une déconnexion comme un état légal plutôt que de le considérer comme une panne. La majeure partie des traitements est exécutée sur les UFs et les SBs ce qui fait que le modèle convient bien aux terminaux mobiles à ressources très limitées. L'atomicité des transactions est basée sur le concept d'atomicité sémantique autorisant les transactions non vitales, quant à l'isolation elle est assurée sur chaque noeud grâce à la méthode des Tickets [GRS91].

2.2.1.12 Terminaux Alternatifs

[BZSH00] esquisse une architecture qui permet à des transactions d'être commencées sur une UM et puis d'être reprises plus tard sur une autre UM. Il est supposé que les UMs sont toujours atteignables par le système de communication. Par conséquent, aucun mécanisme n'est prévu pour les déconnexions. De plus, les hand-offs ne sont pas pris en compte car il est considéré que l'UM évolue dans une localité géographique réduite. L'architecture est basée sur une approche classique de Client-Agent-Serveur, où l'agent est un composant "intergiciel" *(middleware)* résidant sur l'UF. L'agent route toutes les requêtes d'un client (UM) vers la ressource de données (sur l'UF) et contrôle les transactions qui sont considérées de longue durée de vie. Le traitement transactionnel est fait par l'UM. L'agent garde une reproduction de l'état d'exécution de la transaction. Un échec de l'UM ne mène pas à une annulation car l'exécution peut être reprise sur une autre UM en utilisant l'information stockée par l'agent. Pour réduire la charge des communications, la copie sauvegardée sur l'agent n'est pas mise à jour à chaque changement sur l'UM mais seulement pendant les points de contrôle. Après un échec un nouveau client reprendra la transaction à partir du dernier point de contrôle.

2.2.1.13 Moflex

Moflex [KK00] s'intéresse à l'accès aux systèmes multibases hétérogènes à partir d'UMs. Moflex supporte la mobilité des UMs, et permet une flexibilité dans la définition et l'exécution des transactions mobiles. Cette approche est une extension du modèle de transaction Flexibles [ELR90] où les transactions sont une collection de sous-transactions liées par un ensemble de dépendances d'exécution de succès, de défaillance ou externes (de temps, de coût, etc.). L'UM crée une transaction Moflex et la soumet au gestionnaire de transaction mobile *MTM (Mobile Transaction Manager)* résidant sur la SB courante. Le MTM envoie les étapes de la transaction Moflex vers les SGBDs locaux et coordonne leur validation globale. Le moniteur d'exécution local *LEM (Local Execution Monitor)* agit comme une interface entre les MTM et les SGBDs locaux pour la détection des conflits. Une transaction Moflex est définie par un ensemble de dépendances, d'objectifs acceptables et de règles. Les dépendances entre sous-transactions indiquent les conditions d'exécution des sous-transactions : une sous-transaction est exécutée seulement si une sous-transaction qui lui est liée se termine avec succès *(success dependancy)* ou si cette dernière a échoué *(failure dependancy)*, ou bien si une condition externe, tels que le temps, le coût et/ou la localisation, est vérifiée *(external condition)*. Les objectifs acceptables sont l'ensemble des états finaux d'exécution de sous-transaction que l'utilisateur considère qu'ils sont corrects pour son application. Concernant les règles, elles sont de deux types mais étroitement liées ; les règles de contrôle de la migration (ou *Hand-over Control rules)* qui déterminent la politique d'exécution de la sous-transaction lors du déplacement de l'UM d'une cellule à l'autre (par exemple découper (split), continuer ou recommencer) et la politique de combinaison des exécutions des sous-transactions découpées lors du handover. Moflex permet la définition de transactions dépendantes de la localisation [DK98]. La compensation est utilisée pour assurer l'atomicité sémantique.

2.2.1.14 Prewrite

Prewrite [MB01] vise à réduire la charge des traitements sur l'UM et à augmenter la disponibilité des données sur les UMs en cas de déconnexions. Il propose deux types de données, **pré- écrite** et **écrite**. La variante pré- écrite reflète l'état futur de la donnée mais sa structure peut être légèrement différente de celle de la version écrite. En effet, l'opération de pré- écriture n'effectue aucune mise à jour mais rends visible l'état futur que la donnée aura après la validation de la transaction sur l'UF. Une fois que la transaction a déclarée toutes ses opérations de pré- écriture, elle pré- valide sur le MU et rend visibles les résultats sur l'UM et l'UF. La valeur pré- écrite a besoin d'une capacité de stockage moins importante sur l'UM. Par exemple, la valeur pré- écrite d'un objet de type document est le résumé et la valeur écrite est le document complet (résumé, corps et bibliographie). Dans l'approche Prewrite, l'exécution de la transaction est répartie entre l'UM et le serveur de bases de données. Le gestionnaire de transaction sur l'UM exécute la transaction mais les modifications permanentes sont faites par le gestionnaire de données localisé sur le serveur. Prewrite garantie que la délégation de la responsabilité d'écrire sur le serveur de données, limite le traitement des transactions sur l'UM et réduit le nombre de messages échangés sur le réseau sans fil. Le gestionnaire de transactions peut exécuter des pré- lectures, des pré- écritures et des pré- validations. Les lectures ordinaires et les écritures permanentes sont faites par le gestionnaire de données. La SB a des capacités de journalisation et coopère étroitement avec le gestionnaire de données. En mode connecté, le gestionnaire de transaction demande à la SB les verrous nécessaires à l'exécution de la transaction. Le gestionnaire de données fournie les verrous à la SB qui une fois transmis à l'UM permet à cette dernière de se déconnecter. Lorsque le gestionnaire de transactions termine l'exécution, une validation locale (pré- validation) est effectuée et

reportée à la SB. Le gestionnaire de données rend les pré- écritures permanentes et valide la transaction mobile. Prewrite considère les transactions mobiles comme de longue durée et leur implantation peut être faite avec des transactions emboîtées et split.

2.2.1.15 Team Transaction Model

Gupta et al. ont présenté dans [GGG01] le modèle Team Transaction. A cause de leur longue durée de vie couplée au déconnexions fréquentes, les auteurs considèrent que la fiabilité, et non pas la disponibilité, est le problème majeur auquel doit faire face tout modèle de transaction mobile. Aussi ont-ils concentré leurs travaux sur la procédure de journalisation et de recouvrement. Comme les transactions emboîtées, ils ont modélisé la nature distribuée de la transaction en une structure d'arbre hiérarchique. Mais, à la différence du modèle emboîté, Team Transaction offre une certaine flexibilité en supportant la mort involontaire d'une transaction et la capacité d'un nœud parent à tuer son nœud enfant.

Une *Team Transaction* consiste en trois entités: la racine qui est le coordinateur, les enfants qui sont des sous-transactions initiées par le coordinateur et l'agent d'accès aux données (Data Access Agent; DAA). Le DAA est chargé aussi de la journalisation nécessaire au recouvrement après un crash du coordinateur dont il suit la trace afin de déceler rapidement sa panne éventuelle et déléguer sa fonction à un autre nœud. Le modèle peut être organisée en clusters et sous clusters, chaque noeud enfant pouvant redistribuer son travail en formant un nouveau team transaction à un niveau plus bas. Les nœuds enfants rapportent leurs résultats périodiquement et délèguent la validation au coordinateur qui prend la décision finale de valider ou non.

La philosophie du modèle Team Transaction fournie la possibilité de recouvrement d'activités de longue durée grâce à sa procédure de recouvrement qui permet de restaurer autant de travail réalisé avant le crash d'un noeud (en panne de batterie ou sortie du champ) que possible. L'algorithme de recouvrement est similaire dans sa première partie aux phases d'analyse et de "reprise" *(redo phase)* de l'algorithme ARIES[3] [MHL+89, MHL+92]. Cependant, à la différence de ARIES les effets des transactions incomplètes ne sont pas défaits vu que l'objectif est de restaurer l'état de la base de données au moment du crash. Ainsi, dans sa seconde partie, l'algorithme de recouvrement du modèle Team Transaction choisie un nouveau noeud coordinateur et relance la transaction coordinateur après restauration des états et des valeurs les plus récentes des objets de la base au moment du crash [GGG01, Var02].

2.2.1.16 HiCoMo

HiCoMo [LH02] (High Commit Mobile Transactions) a pour objectif de garantir un haut taux de validation en dépit des déconnexions à des applications de prise de décision. Cette approche est applicable à des données dérivées tels que les agrégats, les résumés ou autres statistiques (moyenne, addition, minimum, maximum, etc.). Il suppose l'existence de tables bases sur le réseau fixe et un entrepôt de données (agrégats des tables bases) sur l'UM. L'UM est supposée déconnecté pour la plupart du temps. De manière similaire à Clustering et Two-tier réplication, deux types de transactions sont considérés: les transaction HiCoMo s'exécutant sur les agrégats sur l'UM et les transactions bases initiées à partir du réseau fixe. Les transactions HiCoMo s'exécutent sur des tables d'agrégation pendant les déconnexions et

[3] Algorithm for Recovery and Isolation Exploiting Semantics: Une méthode de recouvrement supportant un verrouillage à granularité fine et des retours arrière partiels (Partial Rollbacks) utilisant un journal avant (Write-Ahead Logging).

les modifications qu'elles apportent sont reflétées sur les tables base grâce aux transactions bases. Ainsi, lors d'une reconnexion, une transaction HiCoMo est transformée en une transaction base. La transformation se base sur une analyse complexe qui utilise de l'information sémantique des données et des opérations. Ce processus est un facteur clé dans cette proposition. Afin d'obtenir un haut taux de validation, seules des opérations commutatives (addition et soustraction) sont considérées pour les transactions HiCoMo, de plus comme dans Two-tier réplication, une marge d'erreur des résultats des transactions HiCoMo et bases est tolérée.

2.2.1.17 Le modèle AMT

Le modèle de transaction AMT *(Adaptable Mobile Transaction model)* [SRAL05] permet de définir des transactions avec plusieurs alternatives d'exécution associées à un contexte particulier. Le but principal est d'adapter l'exécution de transaction aux variations de contexte. Ce modèle permet la description des transactions mobiles ayant une ou plusieurs *alternatives d'exécution* (EA_k), ayant chacune un *descripteur d'environnement* (ED_k) *associé*. Le descripeteur d'environnement est une représentation des caractéristiques variables qui peuvent affecter l'exécution des transactions. Ces caractéristiques sont spécifiques au contexte de l'application; elles dépendent du réseau utilisé, des UMs et des UFs, la localisation, ou encore des habitudes de l'utilisateur. Par exemple, considérer le prix de communication peut être intéressant dans les réseaux où le prix varie selon l'endroit ou l'horaire de communication. Les applications dépendantes de la localité doivent également prendre en compte la localisation des utilisateurs.

Lorsqu'une transaction T_{AMT} est initiée, l'alternative d'exécution appropriée est choisie en fonction de l'état courant de l'environnement mobile. Il est suggéré dans cette proposition, que l'adaptation des exécutions de transaction améliore le taux de validation, les coûts d'exécution, les temps de réponse et la disponibilité de l'application ; un mot la qualité de service de l'application. Les alternatives d'exécution EA_k peuvent prendre la forme d'un des modèles d'exécution suivants: la transaction mobile (1) est initialisée par une UM et entièrement exécutée sur des UFs, (2) est initialisée par une UM/UF et entièrement exécutée sur une UM, (3) l'exécution est distribuée entre des UMs et des UFs, et (4) l'exécution est distribuée entre plusieurs UMs. Donc, une large variété de transactions mobiles est prise en compte. Une alternative EA_k contient un ensemble de *transactions composantes* (t_{ki}) qui doivent respecter les propriétés ACID. Elles peuvent être des transactions traditionnelles *plates* (*flat model)*, distribuées ou imbriquées fermées. Des *transactions de compensation* peuvent être associées aux transactions composantes. Elles peuvent être exécutées en cas de pannes. Une EA_k peut être annulées si une transaction composante annule ou si l'environnement mobile change et que le nouvel état ne satisfait pas le descripteur d'environnement. Les *EA*s et la T_{AMT} sont des unités de coordination; l'accès aux données est réalisé par les transactions composantes. Si on fait l'analogie avec les multibases de données, les transactions composantes sont des *transactions locales* participant aux *transactions globales*.

De plus, un middleware TransMobi est proposé pour la surveillance de l'environnement et l'implémentation du modèle AMT avec des protocoles appropriés tels que le protocole de validation CO2PC (chapitre 3) qui offre l'atomicité sémantique pour les alternatives d'exécution en permettant aux participants de réaliser la validation locale optimiste ou la validation non optimiste.

2.2.1.18 Discussion

Dans la table 2.2 un résumé des caractéristiques des différents modèles décrits ci-dessus est donné. L'accent y est mis sur la prise en charge des principaux problèmes liés à l'environnement mobile à savoir les déconnexions, la mobilité, les ressources limitées.

Tous les modèles, à l'exception de Reporting et AMT considèrent que les transactions mobiles sont demandées par des UMs. Dans Reporting et AMT, les transactions peuvent être demandées par n'importe quelle unité. Twin-Transaction par contre ne comporte aucun site fixe et donc les transactions sont initiées et exécutées sur des UMs. L'utilisation des modèles d'exécution (chapitre 1) est répartie majoritairement entre le premier modèle (complète exécution sur les UFs) qui est appliqué par Reporting, MDSTPM, Kangaroo, Toggle, Pre-serialization, Moflex et le deuxième modèle (complète exécution sur une UM) est utilisé par IOT, Clustering, Semantics-based, Two-tier replication, Pro-motion, Prewrite et HiCoMo. Le troisième (exécution repartie entre une UM et des UFs) est utilisé par Batch commit, Clustering, Two-tier replication, Twin Transactions et terminaux alternatifs. Le quatrième modèle (exécution répartie entre plusieurs UMs) est utilisé par AMT. Ce dernier modèle, en plus d'accepter les quatre premiers, se trouve être le seul (avec Reporting) à accepter le cinquième modèles d'exécution, où plusieurs UMs et UFs font partie de l'exécution. Team Transaction est basé sur une infrastructure ad hoc et supporte donc le modèle 6.

En ce qui concerne les modèles transactionnels, Reporting a un apport original. Les auteurs étendent le modèle de transactions emboîtées ouvertes en appliquant des techniques de délégation afin de permettre la visibilité et la délégation de certaines responsabilités des UMs aux UFs. Reporting acceptent les modes d'exécution 1, 3 et 5.

Pour le support de transactions mobiles, les contributions proposent des variantes par rapport au mode d'exécution mais elles ne modifient pas la structure des modèles transactionnels.

Nous avons aussi constaté que de nombreux modèles exigent une capacité de traitement non négligeable sur l'UM. En effet, l'accent est mis sur la recherche d'une solution aux problèmes des déconnexions par l'autorisation d'une activité autonome, en local, pendant ces périodes. Ce qui ramène à la nécessité de disposer d'au moins une partie des données répliquées et des traitements transactionnels sur l'UM. Le problème dans ce cas reste la consommation d'énergie qu'on a tenté de réduire en minimisant les échanges entre la partie fixe et la partie mobile car ce sont les envoies de messages qui sont les plus consommateurs. Les modèles, comme Moflex ou pré-sérialisation, considérant des UMs à faibles capacités ne tirent pas profit des UMs qui en disposent. Idéalement, la solution devrait aussi bien s'adapter aux terminaux de très faibles capacités qu'à ceux de hautes capacités comme le fait le modèle AMT qui acceptent d'ailleurs les différents modèles d'exécution.

Notons que les modèles Pro-Motion et Moflex offrent une large gamme de propriétés qui permettent plus de flexibilité et de puissance au traitement transactionnel. Pro-Motion est un framework assez complet car en plus de la prise en charge des déconnexions et de la mobilité, il offre des fonctions de "profilage d'utilisateurs" *(user profiling)* qui ne sont pas considérées dans les autres modèles. En effet, disposer des profiles des utilisateurs permet d'optimiser le chargement des données sur le cache en fonction du désir des utilisateurs (les références [CFZ01a, CGFZ03, PAC+02, PJFY04] utilisent la notion de "user profile pour la gestion des données). Si ces fonctions sont ajoutées à Moflex, avec Pro-Motion ils peuvent être de bons candidats à plus d'investigations et de développement. Pro-Motion requiert de bonnes capacités de calcul sur l'UM alors que Moflex peut être implémenté sur des UMs avec de faibles capacités de calcul vu que la plupart des traitements sont effectués sur les UFs.

L'approche avec des clients alternatifs diffère des autres approches par le fait qu'elle traite de la mobilité de l'utilisateur (et de la transaction) d'un terminal à l'autre plutôt que d'une localisation à l'autre. Elle propose donc une solution à un problème différent des autres problèmes considérés jusque là dans la littérature des techniques transactionnels. Une transaction peut alors être exécutée partiellement sur un client, suspendue, puis reprise sur un autre client où elle peut continuer son exécution.

Le modèle Twin-Transactions a la particularité de proposer de répliquer les transactions en plus des données afin de résoudre les problèmes de cohérence des copies et d'assurer la synchronisation à la reconnexion. La notion de *proxy-transaction* introduite dans [PB94a] également réplique les opération de mise à jour d'une transaction s'exécutant sur une UM sur la SB durant la phase de connexion à des fins de recouvrement uniquement ce qui la distingue des Twin-Transactions.

Le modèle AMT apporte une nouveauté par rapport à la flexibilité en proposant la notion de modèle adaptable ou (context-aware) aux variations dynamiques de l'environnement mobile. Il propose de choisir l'alternative d'exécution en fonction des valeurs d'un certains nombre de paramètres évalués au moment d'exécuter une transaction de façon à refléter l'état courant de son contexte d'exécution. Cependant, ce modèle ne fait qu'une adaptation "statique" au moment du lancement de la transaction; or, les variations peuvent survenir à tout moment durant l'exécution d'où le besoin d'une adaptation "dynamique".

Le modèle Team Transactions est le premier modèle que nous avons rencontré s'intéressant à l'environnement ad hoc. De se fait, ce modèle ne compte sur le soutien d'aucun support fixe. Or, la plupart des autres modèles intègrent dans leur conception la partie fixe du réseau en particulier la SB (en plus des sites fixes hébergeant les SGBDs locaux). En effet, souvent la SB héberge la fonction de coordination et le processus de gestion de mobilité lorsqu'elle est fournie. D'autres propositions concernant le traitement des transactions en environnement ad hoc et en environnement de calcul "diffus"[4] ou *(pervasive computing)*[5] [SM03, Wei91] sont présentées dans la section 2.3.

La table 2.2 montre que les déplacements et *hands-off* n'ont pas été abordés autant que les déconnexions et même quand les modèles se prêtent bien au traitement du *handoff* ils ne donnent pas de détails. Le modèle kangaroo est le premier modèle à avoir été conçu spécialement pour gérer la mobilité. La gestion de mobilité dans ce modèle est réalisée par la migration du contrôle avec la transaction. Moflex prends en compte deux aspects de la mobilité: les *hands-off* et l'exécution des transactions dépendantes de la localisation (le seul à considérer ce dernier aspect). De même que Kangaroo, la coordination suit la transaction dans son mouvement. Le mécanisme des files d'attente et d'échange asynchrone de messages permet à une UM de s'enquérir de l'exécution de sa transaction à partir de n'importe quelle localisation. Par construction Pro-motion et Pre-serialisation peuvent supporter la migration de transactions même si les détails sur la gestion de mobilité n'ont pas été donnés dans les articles examinés.

---

[4] http://www.ece.rutgers.edu/~parashar/Classes/02-03/ece572/readings-perv.html

[5] *Ubiquitous computing* de synonymes *ubicomputing* ou *pervasive computing* est traduit en Français par *"informatique omniprésente"* dont les synonymes sont *"informatique diffuse"* ou *"informatique envahissante ou ambiante"* fait référence à la tendance vers l'informatisation, la connexion en réseau, la miniaturisation des dispositifs électroniques (processeurs minuscules et capteurs communiquant spontanément les uns avec les autres) et leur intégration dans n'importe quel objet du quotidien jusqu'à devenir presque invisibles pour les utilisateurs, favorisant ainsi l'accès aux informations dont on a besoin partout et à tout moment.

A propos de la gestion de mobilité, [DK99] propose une analyse de l'impact des stratégies de gestion de mobilité sur le traitement des transactions mobiles. En examinant trois options de gestion des transactions mobiles à savoir : la coordination fixée sur l'UM initiatrice de la transaction, fixée sur une station base ou se déplaçant d'une station base à l'autre en même temps que l'UM. Il est montré que la performance de la stratégie de gestion de transactionS mobileS employée dépend de la nature du mouvement de la transaction et est meilleure si elle s'adapte à l'environnement d'exécution. D'autre part, la stratégie de gestion de localisation (dans la couche réseau) a aussi son impact sur cette performance. Il est suggéré dans cette étude de prendre en considération cet aspect lors de l'évaluation des techniques de gestion des transactions mobiles, ce qui n'a pas été le cas dans les propositions étudiées.

Un bon nombre des propositions analysées sont basées sur un environnement multibases de données qu'elles étendent pour inclure la mobilité par l'ajout d'un gestionnaire de transaction mobile qui est défini au dessus des SGBDs existants et comptant sur le support des stations bases.

Dans les schémas optimistes, les objets en cache sur l'UM peuvent être mis à jour sans coordination mais ces modifications doivent être propagées et validées au niveau des SGBDs pour que la transaction soit validée. Ce schéma peut conduire à un taux d'annulation de transactions mobiles important à moins que les conflits soient rares. Vu qu'on s'attend à ce que les transactions mobiles soient longues à cause des déconnexions et la latence du réseau sans fil, la probabilité des conflits augmente dans cet environnement. Dans les schémas pessimistes où les objets en cache doivent être exclusivement verrouillés, les transactions mobiles peuvent valider localement. Ces schémas conduisent à d'inutiles blocages de transactions vu que les UMs ne peuvent pas libérer les objets cachés tant qu'elles sont déconnectées. Le compromis entre ces approches peut être dicté par plusieurs paramètres dont certains sont liés à la nature même de l'application tel que son besoin en termes de degré de cohérence et d'autres à l'environnement d'exécution de l'application tel que la fréquence des déconnexions ou le type de terminal. Par exemple, les méthodes pessimistes sont nécessaires si l'application désire obtenir les propriétés ACID strictes. De plus, la taille du cache étant limitée sur l'UM et la bande passante étant limitée pour les transferts des objets vers le cache, ces deux paramètres ne doivent pas être négligés dans le choix de la stratégie.

Majoritairement, les modèles ont essayé d'étendre les modèles de transactions avancées qui par ailleurs n'ont pas connu un grand succès en termes d'implémentation dans les systèmes commercialisés classiques [BP96, Ram01, WW04] et cela reste, actuellement, vrai pour le mobile (comme nous allons le voir à travers l'analyse des solutions mobiles commercialisées, section 2.2.2). Les extensions ont porté principalement sur l'adaptation des modes d'exécution de ces modèles afin de prendre en charge les déconnexions et la mobilité proprement dite. Pour cela il a été nécessaire de relâcher les propriétés ACID et principalement l'Atomicité et l'Isolation. La validation est généralement réalisée en deux étapes, la validation locale sur l'UM et la validation globale sur le serveur de base de données ou sur la SB. Cette approche optimiste donne la possibilité de travailler durant les déconnexions sans causer le blocage du reste du système. Concernant l'isolation (ou le contrôle de concurrence), les résultats locaux même s'ils n'ont subi qu'une validation locale sont souvent rendus visibles sur l'UM. De plus, la technique de verrouillage notamment 2PL est la plus utilisée pour le contrôle locale. Pour le contrôle global, ce sont plutôt des approches optimistes qui sont favorisées.

Les modèles étudiés utilisent l'information sémantique pour préserver la propriété de Cohérence. La notion de compact du modèle Pro- Motion et la notion d'objet fragmentable de

Table 2.2-Table récapitulative des modèles de transaction académiques

	Type de transaction	Propriétés	Modèle d'exécution	Gestion des données	Charge sur l'UM	Déconnexions	Mobilité
Reporting et Co-transactions	Transactions emboîtées ouvertes, Co-transactions, reporting, compensable ou non.	Atomicité Sémantique. Isolation (serialisabilité)	Exec. sur UM et SB.	Non indiquée mais nécessite réplication	Elevée	Non Connexion continue	Oui (Pas de détails)
Batch commit	Transactions plates	ACID	Init. sur UM Validation sur SFs	Versions d'objets en cache	Elevée	Oui	-
IOT	Transactions plates de 1ère et 2nd classe	Isolation	Init. UM 1ère classe sur l'UM en mode connecté. 2nd classe exéc. sur UM en mode déconnecté et validée sur SF	Réplication et cache	Elevée	Oui	-
MDSTPM	Multitransactions	Selon Multidabases	Init. UM Exec. SB et FSs	-	Faible	Oui	-
Clustering	Weak et Strict	Atomicité sémantique	Init. UM Weak exec. Valide sur UM en déconnecté. Stricte exécute sur UM en mode connecté	Clusters en cache	Elevée	Oui	-
Semantic-based	Transactions de longue durée	ACID	Init. UM Exec. UM Reintégration sur SF.	Fragments en cache	Elevée	Oui	-
Two-tiers Replication	Transactions Bases et Tentatives	Atomicité Cohérence	Init. UM Tentative sur UM. Base sur UM connectée.	Versions en cache	Elevée	Oui	-
Twin-Transactions	Transactions jumelles	Atomicité sémantique Cohérence	Init. quelconque Exéc. Sur le site initiateur	Copie en cache	Elevée	Oui	Oui (Pas de détails)
Kangaroo	Transactions emboîtées ouvertes et Split	Atomicité sémantique	Init. MU Exec. SB and FS	-	Faible	-	Oui
Pro-Motion	Transaction de longues durée Emboîtées et split	Atomicité sémantique	Init. MU Exec. MU Synchronisation sur SF	Compact en cache	Elevée	Oui	Oui (Pas de détails)
Pre-serialisation	Multitransactions	Atomicité sémantique Isolation	Init. MU Exec. SB et SF	-	Faible	Oui	Oui (Pas de détails)
Terminaux Alternatifs	Transactions plates	ACID	Init. MU Exec. MU et FS	Réplication	Elevée	Oui	Mobilité du terminal
Moflex	Multitransactions et transactions dépendantes de la localisation	Atomicité sémantique	Init. MU Exec. SB et SF	-	Faible	-	Oui
Prewrite	Transactions de longue durée, emboîtées et split	-	Init. MU Exec. MU and FS	Réplication	Elevée	Oui	-
HiCoMo	Transactions Base et HiCoMo	-	Init. sur MU Exéc. sur UM et SF	Agrégats et autres en cache	Elevée	Oui	-
AMT	Transactions emboîtées	Atomicité Sémantique et Atomicité Stricte	MU, SB et SF	Réplication	Variable	Oui	-
Team	Transactions réparties	ACID	MU	Base de données locales	Elevée	Oui	-

Semantic-based sont des exemples de cette approche. Généralement la propriété de Durabilité est assurée après la validation globale sur le serveur de base de données. Certains modèles offrent comme même la durabilité localement; c'est le cas de HiCoMo, Semantic-based et Pre-write, mais avec des inconvénients. HiCoMo est restreints aux opérations commutatives en plus d'exiger une réintégration complexe. Semantique-based réduit la disponibilité vu que les fragments peuvent être gardés au niveau de l'UM pour une durée pouvant être relativement longue. Les coûts de communication dans Pre-write peuvent s'élever à cause des échanges de messages nécessaires à l'acquisition des verrous sur la station base.

Le nombre important de modèles et techniques transactionnels proposés montre qu'en fait jusqu'ici aucune proposition n'est suffisamment satisfaisante pour susciter un consensus. Il nous paraît clair que la propriété d'adaptation est la clé de succès des modèles ou des techniques transactionnels que l'on peut proposer. En effet, les besoins sont si divers d'un contexte d'exécution à l'autre, d'une application à l'autre et d'un utilisateur à l'autre et ces besoins peuvent varier au cours d'une même exécution. Il est alors difficile d'imaginer une solution pouvant satisfaire toute cette diversité si elle n'est pas suffisamment flexible pour pouvoir s'adapter. Dans la section 2.4 nous introduisons quelques approches d'adaptation qui ont été proposé dans le contexte du traitement transactionnel.

2.2.2 Les approches commerciales

Cette section introduit quelques produits commerciaux qui proposent des solutions de bases de données pour les environnements mobiles, mais avant, nous résumons la terminologie du domaine. Les bases de données mobiles disponibles dans le commerce utilisent presque la même architecture de réplication de données: une ou plusieurs bases de données distantes sur les UMs ont des répliquas de la base de données principale stockée dans l'UF. Des transactions simples peuvent être exécutées sur les deux bases de données distante et principale. Les mises à jour sont répercutées localement à chaque base de données. Une fois reconnecté au réseau fixe, l'UM peut lancer une synchronisation qui réconcilie les copies des bases de données en créant un état cohérent unique de base de données sur les bases de données distante et principale. La synchronisation peut être exécutée de deux manières:

• *Session-based*: une base de données de l'UM se connecte directement à la base de données de l'UF. Pendant la connexion, la synchronisation a lieu. Quand la synchronisation est complète, l'UM et l'UF peuvent se déconnecter en ayant un état cohérent de base de données. Cette approche est aussi désignée sous le nom de *connection-based-synchronisation*.

• *Message-based*: un système de transmission de messages (tel que le E-mail) transfère l'information requise pour effectuer la réconciliation. La synchronisation est exécutée quand l'UM et les UFs sont déconnectés. Cette stratégie est également connue en tant que *store-and-forward* ou *file-based synchronisation*.

En raison de l'importance que la synchronisation a dans le calcul mobile, et spécialement sur les terminaux portables de type Palm, un consortium industriel de plusieurs sociétés de logiciels a défini un protocole normalisé de synchronisation appelé *SyncML* [Syn00]. Le but de ce protocole est de fournir un format de synchronisation standard entre UMs et UFs indépendant de toutes plateforme ou de tout format. *SyncML* travaillent avec un ensemble standard de messages représentés comme documents XML. *HotSync* de Palm [CFZ01a] est un exemple de protocole de synchronisation largement répandu (partageant beaucoup d'idées avec SyncML). *HotSync* supporte une synchronisation bi- directionnelle qui permet à des mises à jour d'être exécutées sur l'UM et l'UF.

La dissémination *(data dissemination)* [Fra96] est une technique de distribution de données à partir d'un ensemble de producteurs vers un grand nombre d'utilisateurs. Elle peut être réalisée via des services "*Publish/subscribe"*, des systèmes de notification ou par la diffusion d'information *(broadcasting)* [AAFZ95]. Ces techniques souvent appelées *Push-based* sont intéressantes à cause de la propriété d'asymétrie caractérisant les environnements mobiles. En effet, la réception consomme moins d'énergie que l'émission dans le sens UMs vers les UFs et la bande passante est meilleure dans le sens UFs vers UMs. Le *pull*, qui est la méthode classique d'accès aux données par des requêtes explicites émises par les UMs, peut être combinée en une approche hybride avec le *push* avec l'accès *pull* réservé au cas où l'UM désire obtenir des données en urgence. Nous ne nous étalerons pas sur une autre technique de gestion de données appelées *Data Recharging* [CFZ01b] dont l'idée consiste à combiner la synchronisation avec la dissémination et d'offrir un processus similaire au rechargement de batterie en énergie. L'UM pourrait se connecter à un réseau pour charger les données en fonction de son profil.

2.2.2.1 Informix Cloudscape

La solution d'*Informix* [Inf99] pour le calcul mobile se compose de *Cloudscape*, une base de données relationnelle orientée objet, légère résidant sur l'UM et les UFs, Cloudconnector, un framework de serveur supportant la connexion et résidant sur l'UF, et *Cloudsync*, un composant chargé de la synchronisation entre les transactions de l'UM et de l'UF. Des transactions sont exécutées seulement sur l'UM pendant les déconnexions. *Cloudscape* emploie l'architecture commune où l'UF détient la base de données principale (également appelé source), et les UMs détiennent les répliquas (appelées la cibles) de la base de données principale. Une base de données source peut avoir beaucoup de cibles, alors qu'une base de données cible a seulement une base de données source. La réplication peut contrôler des objets tels que les tables complètes ou seulement les colonnes et les rangées choisies, les vues et les index, les schémas et fichiers de code Java (JAR), des agrégats définis par l'utilisateur. *Cloudscape* emploie une technique de synchronisation qui envoie des données de la base de données de l'UF, et renvoie les événements d'affaires des bases de données de l'UM. Cette technique s'appelle "LUCID", signifiant "*Logic Up, Consistent Information Down*". L'UM traite les données de sa base de données en utilisant des appels à des méthodes Java. De telles méthodes sont appelées chacune unité de travail. La phase Logic Up se produit quand l'application de l'UM est connectée au réseau. A ce moment là, l'UM exécute la synchronisation avec l'UF en expédiant ses unités de travail à l'UF qui les exécute de nouveau. Après que les unités de travail aient été ré- exécutées avec succès dans la base de données de l'UF, ce dernier publie ses résultats aux bases de données de l'UM, lançant ainsi la phase Consistent Information Down. Cette phase met à jour toutes les bases de données mobiles. Un état cohérent unique de base de données est ainsi réalisé. La ré- exécution sur le serveur permet de résoudre les conflits potentiels entre différentes unités de travail soumises by plusieurs UMs. Toutes les transactions exécutées fonctionnant sur les cibles sont conditionnées par le succès de l'application sur la base source après la reconnexion. En cas de succès la transaction devient durable. Si la transaction est rejetée sur la base de données source, elle est défaite sur la cible.

2.2.2.2 Sybase Anywhere

Adaptive Server Anywhere (ASA) [Syb00] est une base de données relationnelle appropriée pour être employé sur de petits dispositifs. ASA est dotée de la synchronisation basée sur la session et celle basée sur les messages. Des transactions sont exécutées seulement sur l'UM pendant les déconnexions. Sybase Anywhere fournit trois technologies de réplication:

• *SQL Remote*: destinée à la réplication bi-direcionnelle basée sur les messages des transactions impliquant un serveur de données et un grand nombre de bases de donnée distantes. Les temps de latence pour la réplication dépendent de l'implémentation et peuvent être de l'ordre de secondes, minutes, ou heures.

• *MobiLink*: destinée à la réplication bi- directionnelle basée sur la session. Elle supporte des bases de données non Sybase et est conçue pour la réplication de données entre une base de donnée centrale et un grand nombre de bases distantes. A la terminaison de chaque session de synchronisation les bases de données sont cohérentes.

• *Replication Server*: destinée à la réplication bi- directionnelle basée sur la connexion des transactions. Elle convient à la réplication de données entre un petit nombre de bases d'entreprise connectées par un réseau haut débit. La latence est de quelque seconde offrant un système de réplication pratiquement temps réel.

MobiLink et *SQL Remote* sont plus appropriées pour les environnements mobiles.

Les transactions peuvent utiliser des *savepoints* qui sont tous libérés quand la transaction se termine. Les transactions peuvent être emboîtées et ou s'exécuter sur des serveurs distants.

2.2.2.3 Microsoft SQL Server CE

Le *Microsoft SQL Server CE* (SSCE) (SSCE) [Mic01] est la base de données fonctionnant sur l'UM; c'est la version mobile de la base de données de *SQL Server* fonctionnant sur l'UF. L'agent client sur l'UM se connecte (HTTP) au serveur IIS *(Internet Information Server)* résidant sur l'UF. L'agent serveur agit en tant qu'interface sur l'UF pour le client s'exécutant sur l'UM. Les transactions sont exécutés seulement sur l'UM pendant les déconnections. La réplication dans *SQL Server CE* est basée sur la réplication *Microsoft SQL Server 2000*. *SQL Server CE* peut répliquer les tables entières, un sous-ensemble de rangées et de colonnes, mais pas les vues et les fonctions définies par l'utilisateur. La réplication est toujours lancée et contrôlée par l'application CE de Windows. La réplication et la synchronisation de données sont réalisées par un modèle *publish subscribe* avec un protocole de communication basé sur les messages. La publication (notamment une collection d'articles où un article est une table dont la réplication est permise), est créée sur l'UF. Les mises à jour émanant des abonnés sont envoyées au *publisher* (éditeur) qui fusionne ces mises à jour avec les mises à jour de tous les autres abonnés. Par la suite, l'éditeur propage les mises à jour de nouveau aux abonnés pour obtenir une cohérence des bases de données repliées sur l'UM. Des publications sont conçues en fonction des groupes spécifiques d'utilisateurs, ainsi une UM recevra uniquement réceptrice les mises à jour auxquelles a souscrit. SSCE permet l'utilisation des *savepoints* et des transactions emboîtées jusqu'à une profondeur de cinq niveaux. Les mises à jour faites dans la transaction emboîtée ne sont pas être visibles à la transaction supérieure. Les résultats deviennent visibles à la sous-transaction parent après que le sous-transaction emboîtée ait validé. Les changements ne sont pas visibles en dehors de la transaction supérieure jusqu'à ce que cette transaction valide. Les transactions sont fermées et sont exécutées séquentiellement.

2.2.2.4 WebSphere Everyplace et DB2 Everyplace

IBM a développé la famille de produits *WebSphere Everyplace* pour la gestion de données dans les environnements mobiles. *WebSphere Everyplace Access* [IBM02b] permet aux UMs d'accéder à des données (email, PIM et applications de données d'affaires). Il délivre des pages web et des applications électroniques d'affaires (e- affaires) aux téléphones cellulaires et aux PDAs. Il contient un service de perception de localisation qui fournit des informations sur la localisation aux applications mobiles. IBM propose également *DB2 Everyplace* [IBM02a], une base de données relationnel hors-ligne mono utilisateur avec une empreinte de 180 Kb qui

facilite le stockage sur UM. Il est disponible pour les systèmes d'exploitation Palm OS, Symbian OS, Windows CE, Windows 95/98/NT/2000/XP, QNX Neutrino, Linux, et Linux embarqué. *DB2 Everyplace* supporte un sous-ensemble du standard SQL et fournit l'indexation. Il contient un serveur de synchronisation bi- directionnelle entre les données relationnelles de l'UM et les données sources de l'entreprise (DB2, Oracle, Informix, Sybase, MS SQL Server et Lotus Domino). *DB2 Everyplace* fournit la gestion de transactions plates (opérations commit, auto-commit et rollback). Il supporte des connexions (ODBC/JDBC) aux bases de données de manière sérialisable. En mode connecté les UMs peuvent demander l'exécution de requêtes et procédures stockées sur le serveur de données.

2.2.2.5 Oracle9iAS Wireless et Oracle9i Lite

Oracle prend en charge l'accès mobile aux données avec le serveur d'application *Oracle9iAS Wireless* [Ora02b]. Ce dernier rend une application web (localisée sur des unités fixes) accessible à partir d'UMs. Il inclut des services mobiles comme PIM *(Personal Information Manager)*, email basés sur la localisation des UMs (pour des applications sensibles à la localisation) et des services de push -via SMS *(Short Message Service)*, WAP *(Wireless Application Protocol)*, émail et voix. En particulier, *Oracle9iAS Wireless* fournit des transactions mobiles sécurisées, orientées commerce électronique, à partir de n'importe quelle UM. Afin de permettre une exécution hors-ligne, Oracle propose Oracle9i Lite [Ora02a] (une extension d'Oracle9iAS Wireless), une base de données relationnelles avec une empreinte de 50 Kb à 1 Mb. Elle s'exécute sous les systèmes d'exploitation Windows CE, Windows 95/98/NT/2000, Palm OS ou Symbian EPOC. *Oracle Lite* est une plateforme logicielle pour développer, déployer, et gérer les applications mobiles *off line*. C'est le noyau du produit mobile d'Oracle, sa technologie de réplication consiste en un ensemble de services qui permettent une synchronisation fiable des données et des applications entre *Oracle Lite* et les serveurs de base de données [Vie01]. *Oracle9i Lite* supporte des transactions plates ACID et les quatre niveaux d'isolation de l'ANSI SQL92[6]. L'accès concurrent est contrôlé avec un mécanisme de verrouillage par lignes. Des connexions JDBC ou ODBC Multiples sont supportées. Il est possible de créer des copies *(snapshots)* pour des UMs à partir de sites maîtres Oracle. Les *snapshots* peuvent être modifiés en mode déconnecté. Une synchronisation bi- directionnelle (Mobile Sync) avec les bases de données principales est faite (en mode connecté) par le site maître. Mobile Sync synchronise plusieurs UMs simultanément.

2.2.2.6 PointBase

PointBase [Poi02] est une base de données relationnelle pour les environnements mobiles. Elle est directement embarquée *(embedded)* dans les applications "hors ligne" ou *off-line* et peut s'exécuter sur des UMs comme les ordinateurs portables, les tablettes PC et les PDAs. *PointBase* est entièrement écrite en Java avec une empreinte (fichier Jar) entre 45 Kb et 90 Kb pour la version micro et 1 Mb pour la version embarquée. Elle peut être exécutée sur toute plate-forme qui supporte une machine virtuelle Java (JVM). *PointBase* supporte les connexions multiples à partir d'une seule application s'exécutant sur la même JVM. *PointBase* micro inclut un support transactionnel pour des transactions plates et l'interface de programmation JDBC. La version embarquée de *PointBase* supporte des transactions réparties (validation à deux phases 2PC), elle utilise le verrouillage par ligne et fournit les quatre niveaux d'isolation de l'ANSI SQL92 (read uncommitted, read committed, repeatable read,

[6] SQL est devenu le plus populaire langage de requête pour bases relationnelles. Le nom "SQL" est une Abréviation de *Structured Query Language* L'American National Standards Institute (ANSI) avec l'ISO sont à l'origine de la publication de ce standard.

serializable). Par l'intermédiaire de *PointBase UniSync*, une base de données PointBase peut être synchronisée bi- directionnellement avec une base de données comme Oracle ou Microsoft SQL Server. Pour stocker des données sur des UMs, *PointBase UniSync* utilise un mécanisme de *publish-subscribe*. Ce mécanisme permet aux applications clientes de souscrire aux données publiées (sous-ensemble de lignes et tables) sur le serveur.

2.2.2.7 FastObjects j2

FastObjects j2 [Fas02] est une base de données à objets. Elle fournit des composants base de données embarquées dans un paquet Java de 450 Kb. *FastObjects j2* est un composant mono site pour des applications embarquées s'exécutant sur tout environnement Java. Elle a une architecture modulaire avec un noyau qui fournit les fonctions essentielles de bases de données comme la gestion du cache ou la gestion de transactions. *FastObjects j2* utilise un verrouillage par objet et fournit les quatre niveaux d'isolation définis dans l'ANSI SQL92. L'architecture modulaire permet l'incorporation de particularités comme la gestion de versions, des événements, le support de XML, la journalisation (récupération basée sur la technique défaire) et la duplication si elles sont demandées par les applications. *FastObjects j2* permet un accès hors-ligne aux bases de données et fournit des transactions ACID (plates), emboîtés et parallèles. Elle supporte JDOQL *(Java Data Objects Query Language)* et elle est conforme aux standards JDO et ODMG *(Object Data Management Group)*. *FastObjects j2* offre un mécanisme de duplication de style "base de données ombre" *(shadow database)*. Les copies peuvent être modifiées par des transactions, les modifications sont journalisées et appliquées à la base de données primaire.

2.2.2.8 Discussion

La table 2.3 résume les principales caractéristiques des systèmes commercialisés que nous avons décrits. De manière générale, les approches commerciales supportent des applications simples telles que des applications mono- utilisateur de gestion de carnets d'adresse. Leurs solutions sont basées sur des techniques de synchronisation simples et n'offrant pas les outils nécessaires pour la prise en charge de la sémantique transactionnelle des applications. Peu des résultats développés dans les modèles académiques ont été implémentés dans les produits. Par exemple, les concepts de transaction de compensation ou alternative ne sont pas offerts. Les notions de transactions emboîtées fermées et de savepoints ont été intégrées dans Microsoft SQL Server et Sybase Anywhere. Oracle Lite intègre les savepoints. Les problèmes de déconnexion ont été résolus grâce à la réplication et à l'exécution en mode déconnecté (*off-line)*. Mais la mobilité n'a pas été considérée dans les produits étudiés. Les systèmes décrits utilisent le deuxième modèle d'exécution (complète exécution sur une UM) uniquement.

Table 2.3-Récapitulatif des approches commercialisées

	Gestion des données	Support des transactions	Support des déconnexions
Informix Cloudscape	- BD relationnelle -Publish-subscribe -Synchronisation par ré-exécution des actions.	-Transacions plates, SQL92 et extensions. -Tentative sur l'UM, validée sur l'UF.	-Exécution off line sur les données répliquées.
Sybase Anywhere	-BD relationnelle -Publish-subscribe -Synchro. Bidirect.	-Transacions emboîtées.	-Exécution off line sur les données répliquées.
Microsoft SQL Server CE	-SQL server DB for UMs. -Publish/subscribe -Synchro. Bidirect.	-Transacions emboîtées.	-Exécution off line sur les données répliquées.
WebSphere Everyplace et **DB2 Everyplace**	-Services email, e-affaire, service de localisation. -Bd relationnelle -Empreinte 180Kb -Publish-subscribe -Synchro. bidirect. par SyncML	-Transactions plates, sous-ensemble de SQL.	-Accès on line à une application Web. -Exécution off line sur les données répliquées.
Oracle9iAS Wireless **Oracle9i Lite**	-Services email, WAP, Push, sevice basé sur la localisation -Bd relationnelle -Empreinte 50Kb à 1Mb -Synchro. Uni ou bidirect. des copies (snapshots)	-Transaction e-commerce. -ACID plates tentative sur le mobile, quatre niveaux d'isolation SQL92	-Accès on line à une application Web. -Exécution off line sur les données répliquées.
PointBase	-Bd relationnelle -Empreinte 45-90Kb et 1Mb -Plateformes Java -Publish-subscribe -Synchro. bidirect. (avec Oracle ou Microsoft SQL)	-*Micro*: Plates et partiellement SQL92 *-Embeded.: Réparties avec 2PC, quatre niveau d'isolation SQL92*	-Exécution off-line sur les données répliquées.
FastObjects j2	-Bd objets -Empreinte 450 Kb -Plateformes Java -Bd ombre pour garder les mises à jour de l'application	*-ACID plates, parallèles et emboîtées, Quatre niveaux d'isolation SQL92*	-Exécution off-line sur les données répliquées.

2.3 Autres techniques de traitement transactionnel

Mis à part le modèle Team Transaction, les approches présentées dans la section 2.2.3 basées sur l'approche client-serveur ou client-proxy-serveur, comptent sur une infrastructure réseau mobile composée d'un ensemble d'unités mobiles se connectant à un réseau filaire -comportant des unités fixes- par le biais de liens sans fil (figure 2.1), appelé réseau mobile à infrastructure *(infrastructure-based mobile network)*. Par ailleurs ces approches ont occupé la part du lion dans la littérature consacrée au traitement des transactions mobiles. Dans cette section, nous allons présenter d'autres approches pour le traitement des transactions en environnement mobiles. La première partie de cette section est consacrée aux approches basées sur les réseaux sans fil sans infrastructure appelés réseaux MANETs *(Mobile Ad hoc Networks)* qui possède d'autres contraintes s'ajoutant à celles déjà existantes dans le cas d'un réseau mobile à infrastructure. Dans la deuxième partie nous allons présenter des travaux consacrés aux environnements de diffusion *(broadcast environment)*. Ces approches, tentent d'exploiter les avantages offerts par la propriété de *diffusion (broadcast),* qui est une caractéristique intrinsèque de l'environnement de communicationradio.

2.3.1 Traitement transactionnel en environnement ad hoc

Dans l'architecture traditionnelle de calcul mobile, les connexions entre unités mobiles sont réalisées via un réseau à infrastructure fixe. Par exemple, dans les réseaux cellulaires, une station base supporte tous les nœuds mobiles de sa cellule. Lorsqu'un hôte mobile migre d'une cellule à une nouvelle cellule, il s'attache à une nouvelle SB. Dans un réseau ad hoc

(Mobile Ad hoc Network, MANET) tous les noeuds (PDAs, laptops, téléphone mobiles, senseurs) sont sans fil et mobiles. Ils peuvent librement et dynamiquement s'auto organiser en topologies réseau arbitraires et fréquemment changeantes sans aucun support fixe (aucune SB n'est utilisée). Les hôtes ou unités mobiles (UMs) peuvent s'interconnecter les uns et les autres dans leur rayon de transmission par auto configuration. L'énergie d'une UM a aussi un impact sur le rayon de transmission; lorsque la puissance décroît, le rayon diminue. Les MANETS sont typiquement utiles dans les situations où aucune infrastructure n'est demandée ou n'est pas disponible. Ils sont aussi très utiles quand le déploiement temporaire et rapide de réseau est demandé. Des exemples de telles situations sont les champs de bataille, l'application de loi, les opérations de secourisme en cas de catastrophe (en zone de tremblement de terre) et aussi dans les applications éducatives et commerciales et de surveillance de l'environnement. Une catégorisation des applications existantes et potentielles et des services de MANETs peut être trouvée dans [YCG+03, CCL03].

Plusieurs modèles de transaction nouveaux basés sur des concepts variables ont été proposés dans le cas d'architectures à infrastructure (section 2.2). Cependant, en comparaison aux réseaux à infrastructure, peu de travaux existent sur sur ce sujet pour l'architecture MANET[7]. Dans cette section nous présentons les travaux rencontrés dans ce domaine mais avant nous identifions les problèmes spécifiques à cet environnement ayant motivé ces travaux.

2.3.2 Motivation et challenges pour les techniques transactionnelles dans les MANETs

Des contraintes similaires existent dans des deux architectures réseau; celle avec infrastructure et les MANETs. Cependant, la dernière soulève des spécificités liées aux caractéristiques de son architecture et aux propriétés de ses applications.

2.3.2.1 Effet de l'environnement sur les modèles de transaction

La recherche sur le traitement transactionnel dans le contexte de l'architecture à infrastructure, s'est souvent chargée de la mobilité des utilisateurs/clients alors que dans les réseaux ad hoc **tous les nœuds** (clients et serveurs) **peuvent être mobiles**. Les modèles de transaction pour réseaux ad hoc doivent prendre en compte les problèmes de localisation des utilisateurs et des sources de données. Le **changement de topologie** signifie que les noeuds mobiles peuvent se connecter et se déconnecter et des coupures peuvent apparaître. Cette perte de connectivité fréquente et imprévue est très commune. Les problèmes de fiabilité et de recouvrement deviennent plus complexes. Les modèles de transaction pour les réseaux à infrastructure compte sur la robustesse de la partie fixe pour garder disponibles les données de recouvrement afin de pallier au haut degré de vulnérabilité de l'environnement. Dans un MANET, le modèle de transaction doit compter sur la robustesse du mécanisme de recouvrement pour fournir la fiabilité.

Dans l'architecture à infrastructure, seule l'énergie du client est considérée. Dans les MANETs, les clients et les serveurs ont des ressources rares; par exemple, **tous les nœuds sont alimentés à la batterie**. Les modèles de transaction et/ou les techniques pour cet environnent doivent tenir compte de l'énergie. Une approche est que les noeuds doivent s'informer les uns les autres sur leurs niveaux d'énergie. Vu que cette action consommera elle aussi de l'énergie, il faudrait décider comment et à quelle fréquence elle doit s'effectuer et la politique à adopter en fonction du niveau des noeuds (par exemple, choisir un serveur avec un niveau d'énergie élevé pour exécuter la transaction ad hoc).

[7] Ces travaux sont généralement plus récents [MA 05] et [BBG+05].

2.3.2.2 Effets des applications sur les modèles de transaction

En plus de leur utilisation dans le domaine militaire, les applications civiles des MANETs sont en expansion [CCL03]. Le comportement naturel de ces applications peut avoir un effet sur la conception des modèles de mécanismes transactionnels. Les applications des MANETs sont parfois temps réel. Par exemple, dans le cas d'un désastre, les ordinateurs portés par les secouristes peuvent être utilisés pour collecter des informations à partir des hôtes localisés dans l'hôpital mobile, sur les équipements médicaux et demander l'accueil d'un blessé trouvé sur le site [GB01]. Il y a un besoin pour la conception d'ordonnanceurs *(schedulers)* temps réel et de protocoles de validation qui minimisent le nombre de transactions annulées à cause de la violation des délais limites *(deadlines)*.

Les applications des MANETs peuvent être, de par leur nature, distribuées [Var02], de longue durée de vie et/ou coopératives. Il est vital pour ces applications d'être capables de reprendre leur exécution à partir du point précédant immédiatement une déconnexion volontaire ou involontaire ou un crash. Un exemple typique que nous avons emprunté à [GGG01] décrit une transaction qui représente une tâche impliquant plusieurs parties et qui peut s'étaler sur plusieurs jours. Dans une zone touchée par un tremblement de terre, une équipe peut être déployée pour compter le nombre des victimes et préparer un rapport détaillé sur les pertes de biens. L'équipe peut consister en plusieurs membres avec un coordinateur d'équipe et après la division de la zone affectée en plusieurs segments, chaque membre peut être envoyé à chacun de ses segments pour collecter l'information désirée. Le segment peut être suffisamment grand pour que la tâche prenne plusieurs jours. D'autres applications où des transactions longues et coopératives peuvent être utilisées sont les compagnes de contrôle démographique, une enquête menée par une compagnie pour un feedback sur ces produits, etc. Ces exemples montrent aussi combien il est important que les données collectées par chaque UM participant à une tâche globale puissent être transférées à un lieu sûr aussitôt que possible. De plus, dans le cas d'une panne d'UM, il est vital de restaurer la plus grande part possible du travail réalisé pour éviter de le refaire. Ceci montre encore une fois l'importance du processus de recouvrement pour le traitement des transactions en environnement ad hoc [GGG01].

2.3.2.3 Le modèle Team Transaction

Ce modèle [GGG01] a été décrit précédemment dans la section 2.2, nous le citons ici pour rappeler qu'il s'agit d'un modèle adapté à une architecture ad hoc. Ce modèle a mis l'accent sur les mécanismes de recouvrement des transactions de longue durée de vie.

2.3.2.4 Déconnexions planifiées

[HAE00] propose une méthode pour supporter le travail d'équipes mobiles en utilisant des bases de données résidant sur les UMs des membres d'une équipe sans l'appui d'une quelconque UF. Le traitement transactionnel peut fonctionner localement à chaque UM si elle est déconnectée. Localement, un mécanisme de gestion des conflits d'accès comme 2PL est utilisé. Un enregistrement de pré- validation est propagé à d'autres UMs par une technique *épidémique*[8] d'échange de messages. Toutes les UMs exécutent alors la synchronisation nécessaire et communiquent leurs actions aussi par les messages épidémiques. Deux UMs ou

[8] L'idée de base des algorithmes épidémiques consiste pour chaque processus à stocker tous les messages qu'il obtient et à les retransmettre un nombre limité de fois vers ses voisins. Il s'agit d'une technique de diffusion à grande échelle.

plus peuvent également traiter des transactions tant qu'elles sont connectées l'un à l'autre. Il se peut qu'une ait besoin de se déconnecter tandis que d'autres UMs continuent à traiter des transactions. Dans ce cas-ci, une procédure *sign-off*, est employée. Si une UM décide de se déconnecter, elle accorde à une autre UM les droits d'agir en tant que *proxy* (procuration). Le proxy-UM représente l'UM dans le traitement transactionnel qui continue entre les UMs restantes. L'UM qui s'est déconnectée peut alors exécuter des transactions de lecture. A sa reconnexion, elle contactera directement le proxy-UM afin de récupérer les mises à jour et rejoindre de nouveau le réseau de traitement des UMs.

2.3.2.5 Gestion de transaction temps-réel en conservant l'énergie *(power-aware)*

Le Gruenwald et Shankar M. Banik se sont intéressés aux contraintes temps-réel ainsi qu'à la mobilité et la conservation d'énergie [GB01]. Les auteurs considèrent que dans l'environnement ad hoc les applications sont temps-réel (en considérant les applications militaires et les catastrophes) et que le client et les serveurs sont mobiles. Ainsi, un gestionnaire de transaction *(Transaction Manager (TM))* au niveau du serveur mobile doit considérer la mobilité des UMs initiatrices ainsi que les *deadlines* des transactions. De plus, le TM devrait tenir compte les restrictions d'énergie sur les UMs. En équilibrant la consommation d'énergie entre les noeuds, les UMs avec une faible batterie ne perdront pas rapidement leur réserve et le nombre des déconnexions peut alors être réduit. Les UMs fonctionnent en trois modes pour réduire leur consommation: le mode *actif*, le mode *veille* et le mode *sleep* [SWR98]. Les UMs sont classées en *petites unités (Small UMs (SMHs))* avec une mémoire de stockage, des capacités de calcul et une énergie réduites et *grandes unités* (Large UMs (LMHs)) qui comportent un SGBD entier. Les SMHs ne comportent que des modules du SGBD pour l'interrogation de leurs propres données, soumettre des transactions aux LMHS et recevoir les résultats.

Les transactions sont de deux types *firm* et *soft*. Les transactions de type *firm* doivent être annulées dès qu'elles ratent leurs deadlines alors que les transactions s*oft* peuvent être exécutées après l'expiration de leurs deadlines. Le facteur le plus important pris en compte pour lestransactions *firm* est le temps et celui des transactions *soft* est l'énergie. Ainsi, les SMH soumettent une transaction f*irm* au plus proche LMH et une transaction s*oft* au LMH qui dispose de la plus grande réserve d'énergie. L'architecture de base de données proposée compte sur le GPS pour avoir les informations de localisation des clients et des serveurs. Lorsqu'elle recoit une transaction, si le LMH est active (dans le mode veille elle se réveille seulement pour une transaction *firm*), elle utilisera un algorithme d'ordonnancement dynamique temps-réel qui conserve l'énergie (a real-time energy-efficient dynamic scheduling algorithm) pour programmer l'exécution de la transaction. Cet algorithm prend en compte le type de la transaction *(firm, soft)*, les deadlines et les limitations d'énergie. Si toutes les données ne sont pas disponibles dans la LMH, il divise la transaction globale en sous-transactions et distribue le deadline entre elles avant de les soumettre aux LMHs qui contient les données demandées. Les LMHs participantes informent le coordinnateur LMH sur la terminaison des sous-transactions. Le coordinateur LMH utilise un algorithme de construction de graphe de sérialisation globale partielle *(Partial Global Serialization Graph (PGSG) algorithm)* [DG00] pour vérifier les propriétés d'Atomicité et d'Isolation de la transaction globale. Puis, il soumet les résultats au SMH demandeur qui envoie un acquittement [GB01]. L'objectif de cette technique est de réduire le pourcentage des transactions manquant leur deadlines tout en conservant la consommation d'énergie des UMs.

2.3.2.6 Un système de transaction auto-organisant pour les réseaux ad hoc

Le papier [PA02] presente la conception et l'implémentation d'un système qui supporte le développent de services commerciaux électroniques. Le système incorpore dynamiquement la propriété transactionnelle dans les nœuds mobiles. Il offre un système transactionnel auto-organisant permettant à des groupes de nœuds de coopérer dans l'exécution de transactions dans un environnement ad hoc. Deux objectifs sont visés dans le système proposé. Le premier objectif est d'éviter une infrastructure fixe en rendant les noeuds capables de s'auto organiser en un moniteur de traitement transactionnel (TP-Moniteur)[9]. Pour atteindre cet objectif chaque noeud est conçu comme un mini TP-Moniteur. Le second objectif est l'adaptabilité du système à différents environnements en offrant l'habilité d'attacher ou de détacher la fonctionnalité du mini TP-Moniteur à un nœud au moment de l'exécution. Un scénario typique présenté dans le papier [PA02] considère un ensemble de noeuds (PDAs, Pockets-PCs, laptops, etc.) dans une foire commerciale où des fournisseurs, des fabricants, des détaillants, et des clients se rencontrent pour l'exposition et la ventes des derniers produits parus. Comme le réseau est spontané, les clients sont libres d'acheter chez différents vendeurs et les vendeurs chez différents fournisseurs d'une manière dynamique. Clients et marchants peuvent librement entrer et quitter la foire participer dans une transaction. Si un noeud change de localisation, par exemple, arriver à une foire différente, il peut recevoir le code nécessaire à l'exécution d'une transaction à partir d'une station base s'il y en a une ou à partir d'un autre nœud. Lorsqu'il quitte la foire, le code est enlevé.

Pour garantir la correction transactionnelle sans composant centralisé, [PA02] adopte CheeTah [PA00], un TP-Moniteur très léger résident sur chaque noeud et qui traite les appels distants comme des sous-transactions d'une transaction racine. Cependant, pour offrir une adaptabilité dynamique à un système spontané comme celui décrit dans l'exemple ci-dessus, les auteurs ont utilisé PROSE, une plateforme dynamique de programmation orientée aspect *(Aspect Oriented Programming (AOP))* pour Java [PGG02], qui permet d'ajouter dynamiquement CheeTah comme un composant aux nœuds joignant un réseau spontané. AOP est l'une des approches permettant l'extension d'applications existantes. PROSE offre l'adaptabilité en combinant le traitement transactionnel avec la logique originale de l'application dans une JVM. Les appels distants entrants au et sortants du noeud sont redirigés via le code du TP-Moniteur pour qu'ils soient transformés en appels transactionnels de manière automatique et transparente [PA02]. La caractéristique d'adaptabilité visée ici permet le chargement d'un nouveau programme au moment de l'exécution pour définir une fonctionnalité additionnelle du système dépendante de la sa localisation et des services de proximité.

L'approche proposée ne définie pas un modèle de calcul distribué ou une nouvelle infrastructure de service, elle utilise des services existants (tel que Jini) pour intégrer le comportement distribué, collaboratif et dynamique d'un environnement ad hoc pour un système transactionnel auto-organisant.

2.3.3 Traitement transactionnel pour applications commerciales dans un MANET

L'objectif principal du projet CoCo/Da[10] est de supporter la cohérence des échanges d'information dans les environnements ad hoc. Il est supposé un modèle de système où des UMs communiquent directement entre elles d'une manière pair à pair. L'architecture du

[9] Un moniteur de traitement transactionnel est chargé de créer, exécuter et gérer les applications tranactionnelles dans un environnement distribué (serveur d'applications).

[10] http://www.inf.fu-berlin.de/inst/ag-db/projects/cocoda/projects-overview.html

système introduite dans le rapport technique [BHPS05] permet le traitement de transaction "équitable" *(fair)* et atomique dans les MANETs. L'idée principale de l'architecture est le SLS *(Shared Log Space)* qui est une mémoire distribuée consistant en l'union de tous les LLS *(Local Log Spaces)* des nœuds participants. L'idée du SLS est d'établir un journal virtuel partagé, qui peut être accédé par chaque nœud du MANET. Le SLS est formé collectivement par l'ensemble des LLS *(Local Log Space)* résidents sur chaque UM. Un journal écrit sur le LLS est disséminé entre tous les voisins. Un nœud se reconnectant au réseau après une panne peut obtenir des informations sur le contenu du SLS en synchronisant sont LLS avec ceux de ses voisins. Ce SLS est diffusé vers autant de noeuds que nécessaire pour assurer le recouvrement.

Un exemple d'application pour le traitement transactionnel "équitable", consiste en l'échange de jetons électroniques. Il n' y a pas d'échange de marchandise dans le protocole. Les parties impliquées conclue un contrat leur permettant d'échanger les droits de possession des marchandises électroniques qui seront acquises ultérieurement. De plus des messages échangés, les logs sur l'état de la transaction en cours sont écrits dans le SLS. La stratégie de recouvrement et de synchronisation proposée permet la conclusion de contracts de façon atomique même si les partenaires de ce contracts sont déconnectés et qu'ils ne peuvent pas communiquer directement ultérieurement. Sans l'utilisation du SLS, la transaction serait bloquée ou annulée.

La notion "d'*équitabilité*" (*fairness*) est considérée comme la principale exigence de ce type d'applications (commerciales). Un échange de marchandise est considéré "équitable" si les deux parties ont reçu ce qu'elles s'attendent à recevoir ou si aucune des deux ne reçoit rien. Il est montré que seule une forme "d'équitabilité faible" ou "*weak fairness*" pouvait être garantie en permettant à chaque partie de prouver sa bonne foi et éventuellement le comportement incorrect de l'autre partie devant une tierce partie de confiance. Pour cela des jetons de non- répudiation sont utilisés et aussi enregistrés dans le SLS. Notons que la solution proposée dans ce projet a été réalisée avec en tête un scénario d'application bien spécifique. Des extensions pour affiner et généraliser la solution sont envisagées dans le projet.

2.3.4 Le modèle de Transaction Neighborhood pour l'informatique diffuse

Les environnements de calcul "pervasifs" étendent le concept traditionnel des réseaux mobiles [PACJY02, PJFY04]. Les unités de calcul mobile se composent des ordinateurs portables, des ordinateurs des dits *"wearable",* des ordinateurs embarqués dans des véhicules, des ordinateurs incorporés dans l'infrastructure physique, et des senseurs. Ces unités satisfont les requêtes de l'utilisateur en comptant sur des répertoires locaux de données et les données disponibles dans d'autres unités situées à proximité. En plus, chaque unité est équipée des technologies ad hoc à courte portée telle que Bluetooth. La technologie ad hoc permet aux unités mobiles d'interagir spontanément avec les autres unités, fixes ou mobiles qui se trouvent dans leur voisinage. Par exemple, deux voitures qui se déplacent sur la route peuvent établir une connexion réseau et échanger des données tant qu'elles sont dans le rayon de couverture l'une de l'autre. À la différence du calcul mobile traditionnel, l'environnement "pervasif" ne différencie pas entre les clients mobiles et les serveurs situés dans une infrastructure fixe. Toutes les unités sont modélisées comme des pairs et peuvent s'engager dans une transaction. Il n'y a aucun appui d'infrastructure, le voisinage de chaque unité est susceptible de changer dans l'espace et le temps, il n'y a aucune garantie que les unités souhaitant exécuter des transactions peuvent être disponibles au même moment et que deux dispositifs déconnectés puissent se rencontrer une deuxième fois. Ceci limite non seulement la disponibilité des données mais également la possibilité de reconnexion entre les noeuds exécutant une transaction. En conséquence, les pairs doivent avoir confiance l'un de l'autre ou

compter sur un tiers. Pour adresser ce problème, un modèle de transaction appelé *Neighborhood-Consistent Transaction* (NC-Transaction) est conçu pour maintenir la cohérence dans le voisinage *(neighborhood consistency)* parmi des nœuds proches [PJYF03]. Il n'assure pas la cohérence globale, cette tâche étant souvent impossible puisqu'il n'y a aucune garantie que deux pairs déconnectés puisse jamais se re-contacter. *NC-Transaction* définit des *témoins (witnesses)* parmi les voisins qui acceptent de surveiller l'état de la transaction. Chaque témoin peut émettre un vote pour valider ou annuler la transaction. Le modèle utilise également *un protocole de vote épidémique (epidemic voting protocol)*[11] [GKW+02, ABG+06] pour la dissémination des votes.

Traditionnellement, les transactions se composent de trois phases - *début*, *exécution* et *fin*. De même, *NC-Transaction* se compose aussi de trois phases : (i) négociation, (ii) exécution et (iii) terminaison. Dans la phase initiale, l'initiateur de transaction négocie d'abord les tâches que chacun devrait exécuter, la durée prévue d'une transaction et l'ensemble des témoins actifs. Pendant la phase de *négociation* les témoins sont choisis parmi les nœuds voisins. Chaque noeud à proximité est présenté avec une liste de nœuds partenaires, de tâches à exécuter et la durée prévue. Un témoin potentiel évalue les tâches et décide de les accepter ou de les rejeter. Les participants déterminent la liste des voisins qui sont d'accord pour jouer le rôle de témoins et en choisissent au plus *n*, (*n* est un paramètre variable pour le modèle).

Dans la phase d'exécution, chaque participant commence à exécuter les lectures et écritures négociées sur ses données locales. Quand un participant doit valider ou annuler sa part de transaction, il essaye d'en informer autant de témoins que possible. En revanche, chaque témoin emploie la phase d'exécution pour rassembler ces intentions de validation/annulation. Une fois qu'un témoin a réuni suffisamment de données pour décider des résultats, il tente d'envoyer un message de validation ou d'annulation à tous les nœuds participants.

Dans la phase de terminaison chaque nœud validera ou annulera selon qu'il ait réuni ou non un quorum de votes positifs des témoins. Ce quorum de votes représente un pourcentage prédéfini des votes diffusés par les témoins. L'avantage d'employer des témoins actifs et un protocole de vote épidémique est que la terminaison de transaction ne dépend pas d'un point central. En plus, l'utilisation d'un protocole de vote épidémique n'exige pas de toutes les entités impliquées d'être simultanément reliées à tout moment et, en conséquence, ceci doit permettre de pallier à la nature dynamique des environnements.

2.3.5 Discussion

La table 2.4 résume quelques caractéristiques des solutions analysées dans cette section. Il est clair que les approches sont différentes. Les principales idées qui apparaissent dans cette étude sont les notions de "délégation", les protocoles épidémiques et d'espace virtuel partagé (SLS). La délégation des tâches d'une UM à un autre en cas de déconnexion du premier se retrouve dans le modèle Team [GGG01], la méthode des déconnexions planifiées [HAE00] et même un peu dans le modèle Neighborhood [PJYF03] à travers les "witnesses" à qui on délèguent une partie de la responsabilité de la prise de décision. L'utilisation des algorithmes épidémiques est proposée dans [HAE00] et dans [PJYF03] et implicitement dans [BHPS05]. Ces mécanismes ont pour objectif de pallier à la nature dynamique du réseau, notamment les déconnexions. L'idée du SLS sert les mêmes objectifs mais présente l'avantage d'un potentiel

[11] La diffusion épidémique a é été introduite en 1987 pour la mise à jour de bases de données [DGH+87] [HSAA03]. Elle a trouvé récemment un regain d'intérêt avec le développement des systèmes de grande taille, des systèmes pair à pair (exemple ad hoc) et des réseaux de capteurs.

de champs d'application plus large dans les environnements ad hoc et ne se limite pas au problème des transactions. En effet, elle offre un moyen d'avoir une vue plus ou moins précise de l'état d'un système très dynamique où le simple échange de messages ne suffit pas pour collaborer dans la réalisation d'une tâche commune.

Table 2.4-Characteristiques des solutions pour le cas ad hoc et pervasif

Modèle	Structure du modèle	Problèmes traités	Méchanismes adoptés	Propriétés des transaction
Team Transaction	-Hiérarchie (inspiré du modèle imbriqué)	-Longue durée d'exécution -Tolérance aux panes (fiabilité) des UMs -Comportement distribué et cooperatif	- Logging périodique -- algorithme de recouvrement special comptant sur un noyau de nœuds fiables	Prpriétés ACID globales
Planned disconnections	Transactions plates	-Déconnexions prévisibles	-Délégation à un proxy	Atomicité Cohérence globales
Power-aware Real-time Transaction Management	Sous-transactions à un niveau	- Transactions temps reel -Constrainte d'énergie -Déconnexion et mobilité	- Transaction *firm/soft* - Scheduling temps réel avec conservation d'énérgie - Rapertir la consummation d'énergie entre les noeuds - Suspension de transaction	Atomicité Isolation globales
Self-organizing Transaction System	Transaction imbriquée ouverte	Auto-organisation dynamique	- Instantiationdynamique de composant (technique PROSE) - Service distribute cooperative (Cheetah TP-monitor)	Prpriétés ACID globales
Transaction équitable	Transaction plate	-Répudiation -Déconnexions -Configuration dynamique	-Espace virtuel partagé (SLS) -Jeton de non-répudiation	Atomicité *Equitabilité*
Modèle Neighborhood		-Déconnexion -Configuration dynamique	- Concept de Témoins -Algorithme épidémique	Cohérence dans la localité ou voisinage

D'un côté, la nature distribuée et coopérative des applications, la nécessité de mettre en commun les ressources restreintes de l'infrastructure ainsi que le désire de pallier à la non fiabilité et changement dynamiques de l'environnement plaident en faveur d'une approche basée sur la collaboration de plusieurs nœuds du réseau pour la mise en œuvre de la fonctionnalité transactionnelle. D'un autre côté, l'autonomie est aussi requise pour le travail hors connexion "off line". De plus, les équipements étant dotés de ressources variables, leur participation à l'exécution d'une transaction peut aussi varier du simple rôle de relais à un rôle de moniteur transactionnel. Il faudrait alors disposer d'une solution pouvant concilier et s'adapter à cette hétérogénéité des matériels et des besoins applicatifs. Un des éléments de cette solution pourrait être, un système de découverte de service qui permettrait à des unités d'exporter leurs fonctionnalités transactionnelles aux nœuds qui n'en disposent pas ou compléter ceux qui désirent collaborer. La "négociation de contrats" utilisée dans le modèle Neighborhood entre les participants et les témoins peut être généralisée pour la gestion de la collaboration entres unités mobiles désirants coopérer. Le contrat pourrait contenir des clauses sur les périodes de déconnexion, le verrouillage de certaines ressources pour une période de temps, etc…).

2.3.6 Les contributions dans les environnements de diffusion

Dans un réseau mobile sans fil, le serveur peut avoir une bande passante relativement plus large dans le sens descendant (émission vers les mobiles) que dans le sens ascendant (des clients mobiles vers le serveur). De tels environnements asymétriques de communication [AAFZ95] rendent les mécanismes conventionnels de traitement transactionnel inapplicables

parce que ces mécanismes exigent une communication bi-directionnelle considérable entre le serveur et les clients mobiles et le temps requis pour la transmission des messages peut être intolérablement long. Un autre problème est la cherté de la transmission de données par l'air car la largeur de bande des clients mobiles vers le serveur est une ressource rare [PB94b]. Un temps prolongé de communication peut être trop coûteux. En outre, le nombre élevé des clients mobiles peut surcharger le serveur par la soumission des transactions. D'autres contraintes sont la capacité limitée de traitement et de la batterie sachant que l'émission est très consommatrice d'énergie pour un mobile. Par conséquent, un des objectifs de conception des solutions est de réduire au minimum l'utilisation de la largeur de bande ascendante.

Afin exploiter l'abondance relative des communication descendantes du serveur vers les clients mobiles, la diffusion de données devient un mode important de transfert de l'information dans le calcul de mobile et les environnements sans fil [AK93, IB94, ZAFA94]. Les disques d'émission ou *(broadcast disks)* [AAFZ95] sont une forme de tels systèmes de diffusion de données. Le serveur diffuse sans interruption et de façon répétitive tous les objets de données de la base. Les clients mobiles voient cette diffusion comme un disque sur lequel ils lisent les valeurs des données émises. Un programme périodique de diffusion est construit pour programmer l'émission des objets de données cycliquement selon certains critères de popularité. Des slots inutilisées dans chaque cycle d'émission peuvent être employées pour émettre des informations de contrôle que les clients mobiles peuvent utiliser pour exécuter quelques fonctions sur leurs transactions locales.

La plupart des systèmes et des applications dans les environnements de calcul mobiles comportent une grande proportion de transactions de lecture *(read-only)*. Ces transactions ne modifient aucune donnée [GW82]. Les exemples incluent des systèmes de diffusion d'information sensible au temps telle que les cours des actions, le trafic routier et la météo. Dans des applications de e-commerce, telles que les marchés boursiers et les enchères, on s'attend à ce que le nombre d'acheteurs ou de soumissionnaires soit relativement petit comparé au nombre de spéculateurs qui lisent fréquemment les prix. Le grand nombre des transactions read-only fait que leur traitement soit un critère de performance important dans ces applications [KKS98]. Bien que, les transactions read-only puissent être traitées avec des algorithmes conventionnels de traitement transactionnel, dans beaucoup de cas il est plus efficace de les traiter avec des algorithmes spéciaux qui tirent profit du fait qu'elles n'exécutent que des lectures [GW82, LSLH98]. Par conséquent, un des objectifs de conception des protocoles est d'exploiter la sémantique des transactions read-only au niveau des clients mobiles en les traitant séparément des transactions de mise à jour au niveau du serveur.

2.3.6.1 L'approche de diffusion multiversion

Dans un autre travail complet [PS98, PC99], un certain nombre de méthodes de diffusion sont présentées pour garantir la correction des transactions de lecture. L'approche de diffusion multiversions diffuse un certain nombre de versions pour chaque donnée élémentaire avec les nombres de version. Cette méthode augmente la taille du cycle d'émission et en conséquence du temps de réponse. D'ailleurs, l'ordre de sérialisation est fixé au début de la transaction; il est souvent trop restrictif et manque de flexibilité. Dans le cas de la diffusion d'invalidation *(invalidation-only)*, une transaction de lecture est annulée si une des donnée qu'elle aurait lue est mise à jour au niveau du serveur, ce qui réduit la concurrence. Dans la méthode *conflit-serializability*, les clients mobiles et le serveur doivent maintenir une copie du graphe de sérialisation pour la vérification des conflits. Le maintient du graphe est trop coûteux. L'intégration des mises à jour dans la copie locale du graphe et la détection de cycle peuvent être des processus trop exigeants en termes de calcul pour des ordinateurs portables.

2.3.6.2 L'approche des rapports de certification

Dans [Bar97], le concept des rapports de certification (*Certification Reports)* est présenté comme une manière de supporter les transactions en environnement mobile. La technique utilise le canal de diffusion pour aider les clients mobiles à effectuer une partie du travail de vérification sur les transactions en cours pour décider de leur annulation ou non. Les clients mobiles n'ont pas de cache local et le serveur diffuse un rapport, appelé un rapport de certification. Ce rapport contient la liste des éléments de données lues *(readset)* et écrites *(writeset)* par les transactions actives qui ont déclaré leur intention de valider sur le serveur pendant la période précédente et après avoir été certifiées avec succès. Tandis qu'un client mobile exécute une transaction, il écoute chaque rapport de certification émis par le serveur. Le client mobile agit en tant que certificateur local pour sa propre transaction en vérifiant dans chaque période si le *readset* et le *writeset* courants de sa transaction ont des intersections avec le rapport courant de certification. La décision d'annuler une transaction est prise si son *readset* a une intersection non vide avec le *writeset* d'une transaction qui valide ou si le *writeset* de la transaction a une intersection non vide avec le *readset* ou le *writeset* d'une transaction qui valide (ceci est inspiré par les algorithmes optimistes de la gestion des conflits d'accès OCC [KR81]). Aucune transaction en conflit avec une transaction concurrente validée n'arrivera au serveur. Sinon, quand la transaction est prête à valider, le client envoie au serveur les ensembles *readset* et *writeset* avec les valeurs à écrire dans la base de données. Le serveur fait la vérification finale pour vérifier si deux transactions exécutées sur différents clients sont en conflit. Le serveur informe les clients sur l'acceptation ou le rejet de leurs transactions par le biais du canal de diffusion. Un des avantages cruciaux de ce protocole est qu'il réalise la majeure partie du travail de validation des transactions sur les clients. D'ailleurs, les transactions qui finalement seraient rejetées sont annulées tôt au niveau du client et peuvent être resoumises promptement. Cependant, les transactions mobiles subiront un retard potentiellement intolérable à cause de leur annulation. En outre, des transactions read-only ne sont pas distinguées des autres dans ce protocole.

2.3.6.3 Approche de validation partielle sur le client

Victor C.S. Lee et ses co-auteurs ont beaucoup de papiers sur le traitement des transactions read-only dans les systèmes répartis temps réel (voir par exemple, [LYL96, LHLH98]) et plus récemment dans les environnements sans fil de diffusion [LLK99, LLSC02, LLK04]. Leurs papiers récents [LLSC02] et [LLK04] présentent leurs principales approches de traitement transactionnel dans des environnements de diffusion. Ils proposent d'adopter une approche optimiste intégrée qui effectue la validation des transactions sur les clients mobiles et sur le serveur. La validation partielle sera effectuée sur les clients mobiles. Au niveau du serveur, un algorithme sophistiqué de validation est conçu pour valider des transactions soumises par des clients mobiles. Les protocoles qu'ils proposent PVTO (la validation partielle avec l'ordonnancement basé sur l'estampillage) ou (*Partial Validation with Timestamp Ordering²*) [LLSC02] et le FBOCC (validation en avant et partielle en arrière OCC) ou *(Forward and Partial Backward validation OCC)* [LLK04] peut détecter des conflits de données à une étape précoce sur les clients mobiles. Le premier protocole utilise l'ajustement dynamique de l'ordonnacement par estampillage pour résoudre des conflits de données avec souplesse et pour réduire les relancements inutiles de transaction. Ces relancements sont introduits par la validation sur le serveur adoptée par les protocoles conventionnels qui supposent implicitement que les transactions validées doivent précéder la transaction en cours dans l'ordre de sérialisation. En employant le mécanisme d'estampillage, l'ordre provisoire de sérialisation entre des transactions en cours d'exécution peut être dynamiquement ajusté et enregistré tant que la cohérence des données n'est pas violée. Dans le dernier protocole, la validation en avant d'une transaction est faite sur le serveur, en confrontation avec des

transactions en cours d'exécution, y compris des transactions mobiles et des transactions du serveur. Sur les clients mobiles, la validation en arrière partielle d'une transaction est faite en confrontation avec des transactions validées au début de chaque cycle de diffusion.

A la fin de leur exécution, les transactions mobiles read-only peuvent être validées localement et les transactions mobiles de mise à jour sont envoyées au serveur pour la validation finale. Ces transactions de mise à jour ont une meilleure chance de validation parce qu'elles sont passées par la validation partielle en arrière. La détection tôt de conflit de données économise des ressources de traitement et de communication, alors que l'ajustement dynamique par les estampilles permet d'éviter des annulations inutiles de transaction. La combinaison des principes des deux papiers est proposée comme une perspective de recherche dans [LLK04].

2.3.6.4 L'approche OCC avec estampille de mise à jour et la pré-déclaration

SungKeun Lee a également effectué une recherche importante au sujet du traitement transactionnel dans les environnements de diffusion. Il a d'abord proposé un protocole optimiste appelé l'OCC-UTS (gestion des conflits d'accès optimiste avec des estampilles de mise à jour) ou *(Optimistic Concurrency Control with Update Time Stamp)* [LHY99] (voir également une évaluation analytique d'OCC-UTS dans [PB99]), pour offrir une cohérence transactionnelle du cache dans un environnement de calcul sans fil et mobile en utilisant l'invalidation par la diffusion. Contrairement à PVTO et à FBOCC, ce protocole préfère la validation en arrière. Cependant, il n'utilise pas une validation en arrière pure où le *readset* d'une transaction en cours de validation est confronté au *writeset* des autres transactions qui ont déjà validé. OCC-UTS vérifie si l'exécution d'une transaction mobile tentant de valider est serialisable ou non en comparant les estampilles de chaque donnée élémentaire accédée par la transaction à sa dernière estampille de mise à jour qui est maintenue sur le serveur. Ainsi, le contrôle de cohérence des données accédées et le protocole de validation sont implémentés de façon distribuée comme une partie intégrale du processus d'invalidation de cache, le plus gros travail du contrôle de cohérence étant effectué sur les clients mobiles comme c'est le cas dans PVTO, FBOCC et OCC-UTS.

SungKeun et ses co-auteurs ont proposé une recherche plus récente [Lee04, LK04a] basée sur une notion de traitement transactionnel basé sur la pré-déclaration [LHK03, LK04b]. L'idée centrale est d'utiliser la pré-déclaration du *readset* afin de réduire au minimum le nombre de cycles de diffusion différents à partir desquels les transactions lisent leurs données. Ainsi un client recherche les éléments de données dans l'ordre dans lequel elles ont été diffusées, plutôt que dans l'ordre dans lequel elles ont été demandées. Plus récemment il y a eu des tentatives dans la recherche à utiliser efficacement ce qui s'appelle une diffusion hybride ou *hybrid broadcast*; c'est-à-dire ; une diffusion avec en plus un canal ascendant. Il est supposé que les données que le serveur maintient sont divisées en *Push_Data* pour la radiodiffusion périodique et *Pull_Data* pour le service sur demande. C'est-à-dire que les clients doivent explicitement demander des objets de données dans le cas de *Pull_Data*. De nouveaux algorithmes de traitement transactionnel pour des émissions hybrides doivent être conçus [KYL04].

2.3.6.5 Discussion

Le mécanisme OCC laisse les transactions s'exécuter sans encombre jusqu'à ce qu'elles atteignent leur point de validation [KR81]. Cette approche convient à la propriété asymétrique de largeur de bande de communication dans des environnements de diffusion. Le serveur peut employer la grande largeur de bande descendante pour diffuser les données aux transactions s’exécutant sur les clients mobiles où elles peuvent être traitées localement sans envoyer

chaque demande au serveur. Après qu'une transaction finit de lire et pré-écriture tous les objets demandés sur le client mobile, la transaction ainsi que les informations nécessaires sont envoyées de nouveau au serveur pour la validation. Cette extension basique et directe de l'approche optimiste aide à soulager la bande ascendante limitée de communication.

Cependant, cette approche souffre d'un certain nombre de surcoûts. Premièrement, un grand nombre de demandes de validation peuvent surcharger le serveur et le serveur doit garder un long enregistrement de l'historique des transactions validées. Deuxièmement, les transactions mobiles causant des conflits avec des transactions validées sur le serveur, sont autorisées à s'exécuter jusqu'au bout et sont envoyées au serveur pour la validation où elles seront finalement annulées. Le traitement de ces transactions est inutile et gaspille la largeur de bande ascendante. Ces surcoûts peuvent éventuellement mener à des transactions mobiles ratant leurs dates limites. Pour résoudre ces problèmes, les propositions ont suggéré des variantes d'OCC qui détectent des conflits de données et relancent les transactions à une étape précoce sur le client mobile. En conséquence, on peut éliminer le traitement et la communication inutiles. D'ailleurs, les transactions peuvent avoir une plus grande chance de se terminer avant leurs dates limites après leur relancement.

La composante clé dans des protocoles OCC est la phase de validation, où le destin d'une transaction est décidé. Avant de décider si une transaction doit valider ou annuler, sa serializabilité avec des transactions concurrentes est vérifiés. La serialisabilité de transaction peut être effectuée fondamentalement de deux manières: validation en arrière et la validation en avant [Hae84]. Tandis que dans le schéma en arrière, le processus de validation est fait contre des transactions validées, dans le cas en avant, il est effectué contre des transactions en cours d'exécution. La validation en avant est considérée plus efficace dans les environnements mobiles. Sur le serveur, il y a des transactions soumises pour la validation par différentes sources comprenant les clients mobiles. Il n'est pas souhaitable de relancer une transaction mobile en cours de validation en raison du coût élevé de relancement. Par conséquent, la validation en avant est un meilleur choix pour le serveur en raison de la flexibilité de choisir de relancer les transactions qui sont actives et qui sont en conflit avec la transaction en cour de validation. En outre, le schéma en avant généralement détecte et résout la validation de conflits de données plus tôt que ce que fait le schéma en arrière, et par conséquent il perd moins de ressources et de temps [LLSC02, LLK04].

De plus, seulement l'ensemble des données écrites *(write set)* d'une transaction est exigé pour la validation en avant, il permet aux transactions de lecture d'être validées localement et de façon autonome sur les clients mobiles. Cependant, la validation en avant est considérée sous optimale si le serveur n'est pas d'état *(stateless)* ; il ne connaît pas l'état des transactions mobiles. Dans ce cas-ci la validation en arrière serait préférée car les informations sur les transactions validées peuvent être obtenues à partir des journaux du serveur [LHY99].

Le meilleur choix semble être une solution hybride où les clients mobiles doivent déterminer si leurs transactions mobiles peuvent être validées en détectant n'importe quels conflits de données avec les transactions validées auparavant grâce à une validation en arrière *(backward validation)*. Puis, la validation en avant *(forward validation)* doit être effectuée sur le serveur. En conséquence, la performance du système est améliorée. Bien qu'il soit possible d'effectuer la validation en arrière à la fin d'une exécution de transaction [LHY99]], elle cause des relancements retardés de transaction et le gaspillage des ressources sur les clients mobiles. Ainsi, la validation en arrière partielle est introduite au début de chaque cycle de diffusion [LLK04]. S'il s'avère qu'une transaction a lu des données incohérentes, elle est relancée immédiatement.

Pour garantir la serializability, des informations de contrôle doivent être diffusées par le serveur aux clients mobiles. La taille et la complexité de ces informations affectent l'exécution du système. Tout d'abord, l'envoi des paramètres consomme de la bande passante. Si la taille des informations est comparable à la taille de la base de données à diffuser, il y aura inefficacité en termes d'utilisation de largeur de bande. Deuxièmement, la taille des information de contrôle affecte la longueur du cycle d'émission, qui a son tour a un grand impact sur la satisfaction des contraintes temporelle des transactions sur les clients mobiles. Il y a deux composants principaux à ce fait. Le premier est le temps d'attente pour les données. Le temps d'attente augmente avec la longueur du cycle de diffusion. Le second est attribué aux relancements de transaction. Des relancements de transaction sont principalement provoqués par des conflits de données pendant un cycle de diffusion. Quand la longueur du cycle augmente, le nombre de transactions validées sur le serveur par cycle augmente. Ceci en retour augmente la probabilité de conflits de données, qui mène à un taux plus élevé de relancement de transaction.

2.4 L'adaptation dans les systèmes mobiles

Les environnements mobiles devenant de plus en plus courants, leur utilisation se doit de devenir aussi naturelle et performante que possible. En dépit de tous les problèmes de cet environnement, un utilisateur souhaitera se déplacer librement et continuer à travailler le plus normalement possible sur son unité portable. Le besoin de continuité de service dans un environnement mobile soulève de nombreux problèmes. Tout d'abord, il est indispensable de gérer la disponibilité des données malgré les déconnexions. Par ailleurs, il est également indispensable que le système et les applications soient réactifs aux changements de l'environnement. Par exemple, lorsque la bande passante disponible diminue, une application se doit de réduire sa consommation réseau.

La réactivité ou adaptation aux changements de l'environnement a été le sujet de nombreux travaux qui ont suivi différentes approches [BNB05]. La première approche, appelée aussi adaptation *"application-transparent"*, masque complètement la mobilité à l'application comme le fait le système de fichier CODA [KS92] qui masque les déconnexions. Le besoin d'impliquer l'application dans le processus d'adaptation s'est très vite fait ressentir, ce qui a donné naissance à l'approche *"Application-aware"*. Les applications doivent tenir compte des limitations engendrées par la contrainte de portabilité et de mobilité en étant plus flexibles et en modifiant leurs dépendances aux ressources selon leurs disponibilités, ajustant leurs comportements de façon dynamique. Cette approche consiste à faire coopérer le système et les applications pour qu'ils s'échangent des informations. Le système fournit des informations sur l'état de l'environnement, par exemple la bande passante disponible. Les applications fournissent des indications sur les ressources de l'environnement qu'elles considèrent comme critiques, et utilisent les informations du système pour choisir la forme d'adaptation qui leur convient le mieux. Odyssey est un des premiers exemples de cette approche qui propose une solution pour l'adaptation au niveau système d'exploitation [SNK+94, SNT+97]. Le système surveille les niveaux des ressources, notifie les applications en cas de changement, en tenant compte des fenêtres de tolérance qu'elles auraient préalablement spécifiées. Chaque application décide, indépendamment, de l'adaptation qui lui convient et c'est le système qui coordonne et applique les décisions effectives d'allocation des ressources. Les systèmes Gaia [CHR01, RC01], MoLèNE [AS99, SA00], Cadmium [Bag98, Bag99], AeDEn [MA01], ISAM [YAB02] et CARISMA [CEM01, CEM03] sont un échantillon d'une telle approche.

Dans le domaine du traitement transactionnel, nous avons noté dans la section 2.2 que sur l'ensemble des solutions étudiées, seul le modèle AMT [SRAL03, SRAL05] a proposé cette

notion de modèle de transaction mobile adaptable ou "contexte-aware" *("sensible au contexte" ou "conscient du contexte")*. Le calcul context-aware a été défini pour la première fois par Schilit et Theimer [ST94] : *"un logiciel qui s'adapte selon le lieu de son utilisation, l'ensemble des objets et personnes proches, et les changements de ces objets à travers le temps"*. Depuis cette première interprétation, plusieurs définitions et catégorisations des applications context-aware ont été données. Parmi lesquelles, la définition de Schilit *et al*. [SAW94] décrit les applications context-awares comme étant des applications qui "*adaptent dynamiquement leurs comportements selon le contexte utilisateur et le contexte de l'application"*, sachant que le contexte[12] inclus dans le cas qui nous intéresse ici les caractéristiques du réseau (bande passante, état de connectivité, coût de communication, …), les caractéristiques du terminal (mémoire, écran, batterie, ...), la localisation (position, ressources proches, …), etc. L'approche d'adaptation dynamique et context-aware a en effet suscité un grand intérêt dans le domaine du calcul mobile vu que cet environnement est caractérisé par une nature très dynamique où de nombreux paramètres sont sujets à des variations plus ou moins fréquentes. Dey [Dey01] a essayé de donner une définition générale du context-aware computing : « *Un système est context-aware s'il emploie le contexte pour fournir l'information et/ou des services appropriés à l'utilisateur, et dont la pertinence dépend de la tâche de l'utilisateur* ».

Les nombreux travaux de recherche qui existent sur l'adaptation et dont certains ont été cités ci-dessus, ont été abordés dans le domaine de recherche sur les middlewares pour le calcul mobile contexte-aware (des travaux récents sont : [ECDF01, CEM03, LeM03, Jan05, Dav05, KMS05]) et ne sont donc pas spécifiques au traitement transactionnel. Cependant, il existe quelques travaux relatifs au traitement transactionnel qui se sont intéressés à l'adaptation mais pas dans le contexte du calcul mobile; nous les présentons dans ce qui suit.

Depuis la sérialisabilité jusqu'à la cohérence causale[13], plusieurs critères de cohérence sont envisageables pour les données gérées dans les systèmes répartis. Le choix d'un critère particulier relève à la fois de la nature des données et de l'application qui les manipule. L'article [TR96] propose une approche modulaire pour mettre en oeuvre toute une famille de critères de cohérence. Il définit un ensemble de mécanismes généraux à partir desquels peuvent être implantés des critères particuliers [Mat88]. Puis il montre comment un critère de cohérence particulier peut être défini par un ensemble de règles de synchronisation. L'interprétation de ces règles par les mécanismes précédents offre alors une mise en oeuvre particulière du critère concerné. Un avantage de l'approche proposée réside dans la possibilité de changer statiquement ou dynamiquement le critère de cohérence.

Un des domaines qui a le plus nécessité un modèle de transaction adaptable est celui du "travail coopératif assisté par ordinateur" ou *Computer Supported Cooperative Work* (CSCW). Les efforts consentis pour développer des modèles de transactions coopératifs ont contribué à renforcer la tendance au développement de frameworks pouvant être ajustés à différentes situations. La littérature est riche dans ce domaine nous citons [RN04, Ram01]. Ce qui nous a intéressé dans ce travail est l'intérêt qu'il porte aux problèmes liés à la nature

[12] Le mot « contexte » est un mot très vaste qui peut avoir plusieurs sens et peut être utilisé de plusieurs façons dans différents domaines. Cependant dans notre travail nous sommes concernés par le contexte utilisé par les applications dans un environnement mobile. De manière générale, on peut avoir le contexte de calcul (bande passante, ressources de calcul,…), le contexte utilisateur (localisation, situation sociale, personnes proches,…), le contexte physique (éclairage, les niveaux de bruit, l'état du trafic, la température,…) et le contexte temps (heure de la journée, la semaine, le mois, et la saison de l'année,…).

[13] La cohérence causale et la sérialisabilité causale sont des critères de cohérence faibles dans lesquels la relation de causalité entre les opérations de lecture et d'écriture sur des objets partagés jouent un rôle central [TR96].

dynamique et hétérogène du travail coopératif. La solution propose un support transactionnel adaptable qui non seulement peut accepter des situations différentes mais peut aussi être modifié suivants les changements pouvant survenir dans l'environnement réel pendant l'exécution du travail. Entre autre, les caractéristiques intéressantes des modèles existants ont été identifiées puis étendus pour la conception du framework appelé CAGISTrans. Ce dernier prend en charge les environnements dynamiques en permettant des améliorations durant l'exécution. Un système de gestion de transaction a été construit sur le principe du middleware pour permettre l'interopérabilité et l'indépendance de la base de données; ce qui permet de pallier aux problèmes d'hétérogénéité.

RTF *(Reflective Transaction Framework*) [BP96, BP97, Bar99] est un framework dont le but est la spécification et l'implémention des modèles de transaction avancés au dessus de moniteurs transactionnels commercialisés. Les composants de base sont des *adaptateurs de transaction*-i.e., des modules ajoutés, fournissant des services transactionnels pour les applications avancées. RTF est comme son nom l'indique basé sur le concept de "*réflexion" (reflection)* lui permettant de s'ouvrir pour s'adapter aux différents modèles mais ne supporte pas le changement dynamique du comportement des transactions.

Par définition, le principe de réflexion permet à un programme d'accéder, de raisonner sur et d'altérer sa propre interprétation de façon dynamique. Ce concept a été adopté tout d'abord dans les langages de programmation [Smi82], puis a attiré l'attention des chercheurs ces deux dernières décennies pour son applicabilité dans les systèmes d'exploitation [Yok92], les systèmes distribués [Str93] et enfin dans les middlewares [BCRP98, CEM03]. Dans [KJ03, Kar03, JK04] l'approche réflexive a été utilisée pour la conception d'une plateforme middleware pour des services de transaction qui devraient satisfaire la diversité des besoins des applications comme le workflow, le travail coopératif, le multimédia et les applications mobiles. On propose alors une architecture composée d'un ensemble de services transactionnels spécialisés offrant chacun des garanties transactionnelles différentes. Dans cet environnement des services nouveaux ou modifiés, offrant des propriétés différentes peuvent être inclus au besoin. Un service est vu comme un composant logiciel développé indépendamment et livré à cet environnement. Un service peut être soit complet soit être lui-même assemblé à d'autres composants spécialisés; par exemple, le recouvrement, le verrouillage ou la gestion des ressources. Une importante caractéristique du service transactionnel sont les propriétés qu'il garantie. Pour un service traditionnel, ce sont les propriétés ACID. Pour un service avancé l'atomicité sémantique peut être garantie. Des transactions différentes peuvent demander des services différents ce qui conduit à une exécution concurrente de ces services. Un gestionnaire de services est chargé d'assurer la synchronisation entre services hétérogènes et incompatibles, il contrôle aussi le déploiement et la modification de ces services.

Le besoin de flexibilité et d'adaptation est aussi partagé par les recherches portant sur le support transactionnel pour le wokflow. Les auteurs de [LP98] proposent une approche pour l'implémentation de gestion et de restructuration dynamique d'activités workflow avec un support pour les propriétés transactionnelles distribuées. Entre autres, des opérations Split et Join sont à la base de leur proposition. En fait, les opération Split et Join ont été introduites par Kaiser et Pu pour implémenter la notion de dynamisme dans la gestion des accès concurrents [PKH88] (voir la section 2.1).

Dans [WW04] il est constaté que les services Web *(Web Services)* stimulent des projets d'interopérabilité des systèmes qui soulèvent des problèmes dus à l'hétérogénéité et la grande variété des modèles de transaction distribués. Or, le traitement transactionnel joue un rôle clé dans la garantie d'une coordination fiable et cohérente des opérations à travers les services web. Cette coordination non seulement nécessite les propriétés ACID traditionnelles mais

aussi une coordination plus flexible de services multiples et hétérogènes conduisant encore une fois à la relaxation des propriétés ACID. Malgré les efforts investis dans les modèles étendus, aucun d'entre eux n'est suffisamment flexible pour supporter à lui seul toutes les applications. Alors, [WW04] exprime le besoin d'une architecture de système de traitement transactionnel adaptable. Les technologies des services web semblent offrir une base pour cet objectif en permettant aux systèmes transactionnels existants d'être encapsulés en web services et être intégrables avec d'autres systèmes [WW04]. En effet, l'architecture des services web ignore les détails d'implémentation qui se cache derrière le service, elle ne s'occupe que du transfert de données structurées entre les parties ce qui offre une solution prometteuse possible pour l'interopérabilité. Sur la base de ces constats, un framework transactionnel est proposé pour le support de l'adaptation et de l'interopérabilité dans le domaine des services web *(Web Services)* [WW04]. Il réalise une extension du Activity Service Framework (ASF) de OMG CORBA qui est un modèle de transaction orienté objet vers un modèle de transaction orienté Web. L'ASF fournie une infrastructure capable de supporter la construction de modèles de transactions étendus variés, cependant, il n'est pas adaptable pendant l'exécution aux changements dynamiques des besoins des applications. Les extensions apportées proposent de pallier à cet inconvénient en permettant au système de traitement transactionnel de s'auto évaluer et d'appliquer les modèles et les optimisations les plus appropriées.

2.5 Conclusion

Il est clair que vu la croissance sans précédent que connaît l'utilisation des terminaux et ordinateurs mobiles, le travail mené sur les ordinateurs fixes sera progressivement transféré sur les ordinateurs portables. Les utilisateurs ressentiront de plus en plus le besoin de réaliser des tâches plus complexes et sophistiquées sur leurs portables. De ce fait, des avancées dans le domaine du traitement transactionnel sont nécessaires.

Comme on l'a vu à travers ce chapitre, les transactions dans l'environnement mobile diffèrent des transactions des systèmes centralisés ou répartis. En effet, pour supporter les transactions mobiles, les modèles de transaction et leurs techniques doivent s'accommoder de nouvelles contraintes, telles que la non fiabilité des communications, la vie réduite de la batterie, la faible bande passante et les capacités de calcul et de stockage réduites. Les calculs mobiles doivent minimiser les annulations de transaction dues aux déconnexions. Les exécutions sur les données partagées doivent assurer la correction des transactions. Le blocage de l'exécution des transactions doit être minimisé pour réduire le coût de communication et accroître la concurrence. L'autonomie doit être offerte pour permettre aux transactions de s'exécuter ou d'être validées malgré les déconnexions temporaires.

L'autre caractéristique importante dont il faut tenir compte est la variation dynamique des caractéristiques qui nécessite une approche basée sur l'adaptation context-aware. En outre, le besoin d'adaptation des modèles et des propriétés transactionnelles variables a été exprimé déjà dans les environnements répartis. D'où ce double objectif que nous avons abordé dans les chapitres 6 et 7 de cette thèse.

Les concepts théoriques développés jusqu'ici pourront servir de base pour la conception de solutions et la production de prototypes qui seront testés sur des applications du monde réel. En effet, l'étude que nous avons proposée dans ce chapitre nous a permis de constater qu'un décalage significatif existe entre les approches académiques et les approches commerciales ce qui prouve qu'il y a encore beaucoup à faire dans ce domaine avant d'atteindre les résultats escomptés. Dans cette thèse, nous nous sommes fixé comme objectif d'examiner en profondeur les problèmes liés à la validation des transactions mobiles. Cette démarche qui

consiste à traiter séparément les problèmes transactionnels de modèles, de validation, de contrôle de concurrence, etc., a aussi été adoptée dans les systèmes centralisés et répartis. Les chapitres 3, 4 et 5 suivants sont donc consacrés à la validation des transactions en environnement de calcul mobile. Puis les chapitres 6 et 7 sont consacrés à la proposition d'un modèle et d'un protocole de validation adaptables avec une architecture context-aware de support.

Chapitre III

La validation atomique

Basic research is what I'm doing when I
do not know what I'm doing.
--WernherVon Braun

Les transactions sont des abstractions puissantes qui offrent des garanties en termes de fiabilité aux systèmes de bases de données et aux systèmes distribués en présence de la concurrence et des pannes. Ces garanties sont obtenues par l'implémentation des propriétés ACID (Atomicité, Cohérence, Isolation et Durabilité). La propriété d'atomicité d'une transaction distribuée ne peut être assurée que par un protocole de validation atomique. Vu son impact important sur les performances des systèmes distribués, le problème de la validation de transaction a fait l'objet d'une recherche intensive depuis les années 70s et qui continue d'être active jusqu'à nos jours. En effet, le caractère fondamental et l'intérêt aussi bien théorique que pratique de ce problème conjugué aux développements récents réalisés sur les environnements mobiles ont renouvelé la motivation des travaux de recherche sur ce sujet.

Après avoir introduit une définition formelle du problème de la validation atomique, ce chapitre présente un rappel sur les travaux qui lui ont été consacrés dans le cadre des systèmes répartis; le protocole de validation à deux phases étant au centre de la discussion. Différentes optimisations et variantes du protocole à deux phases ainsi que les notions liées aux protocoles non bloquants (c'est-à-dire, garantissant la terminaison de la transaction quel que soit le type de pannes) dans le cadre des systèmes synchrones et asynchrones ont été abordées. Cette présentation permet de familiariser le lecteur avec les problèmes et la terminologie liés à ce thème. L'impact de la mobilité sur les protocoles de validation est ensuite introduit avant d'aborder et d'analyser les travaux de recherche réalisés dans le cadre de l'environnement mobile et sans fils.

3.1 La validation de transaction dans les systèmes distribués

Cette section donne un aperçu sur les protocoles assurant l'atomicité des transactions réparties. On parle de transaction *répartie* ou *distribuée* pour qualifier une transaction dont les opérations manipulent des données distribuées sur plusieurs sites. Une transaction répartie est décomposée en autant de *branches* que de sites auxquels elle accède. Pour terminer une transaction, tous ces sites doivent coordonner leurs actions afin de garantir *l'atomicité* globale.

3.1.1 Définition de la validation atomique

La propriété d'*atomicité*, appelée aussi propriété du *tout ou rien (all-or-nothing)*, garantit que soit toutes les mises à jour faites par la transaction sur l'ensemble des sites sont rendues permanentes (on dit que la transaction est *validée*) soit elles sont toutes annulées (on dit que la transaction est *annulée* ou *abandonnée*). Chaque site accédé par une branche de transaction distribuée est appelé *participant*. Les différents participants se mettent d'accord sur l'issue d'une transaction en exécutant un *protocole de validation atomique (Atmic Commit Protocol, ACP)*. Chaque participant révèle sa capacité à valider en émettant un *vote*. Un vote *oui* (ou *prêt*) implique que le participant s'engage à rendre permanentes les mises à jour effectuées par la transaction même s'il tombe en panne. Un vote *non* (ou *abandon*) indique qu'il ne pourra pas valider suite à un problème interne. Le problème de la validation atomique est un problème d'*accord* entre les participants sur l'issued'une transaction. Formellement, le problème de la validation atomique est défini par les quatre propriétés suivantes, qui doivent être satisfaites par tout protocole de validation atomique:

• **Accord Uniforme:** tous les participants qui décident (i.e., de valider ou d'annuler), prennent la même décision.

• **Validité:** si un participant décide de valider, alors tous les participants doivent avoir voté *oui*.

• **Intégrité:** chaque participant décide au plus une fois.

• **Non Trivialité:** si tous les participants votent *oui* et aucune panne ne se produit, alors tous les participants décident de valider. La propriété de non trivialité [BT93] exclut la solution triviale au problème dans laquelle les participants décident toujours d'annuler la transaction.

3.1.2 Le protocole de validation à deux phases

Le protocole le plus communément utilisé pour résoudre le problème de la validation atomique dans un système distribué est le **protocole de validation à deux phases** *(two-phase commit (2PC) protocol)*. Les standards tels que OSI-TP [ISO92], DTP de X/OPEN [X/OP96] et OTS de CORBA [OMG94] incluent la spécification de services offrant la sémantique du protocole 2PC (figure 3.1).

Le protocole 2PC se déroule en deux étapes ou phases: la phase de vote et la phase de décision. Dans la première phase, le site où la transaction a été initiée *(generalement appelé le coordinateur)* envoie des messages à tous les sites impliqués *(participants)* dans la transaction, leur demandant de se préparer à la valider la transaction *(message prepare)*. Si, pour une raison ou une autre, dont un problème de contrôle de concurrence ou de panne de stockage, un des participants repond *No*, le coordinateur décide d'annuler la branche locale de la transaction et envoie des messages *Abort* à tous les participants. Si toutes les réponses reçues sont *Yes*, le coordinater décide alors de valider la branche locale et en informe tous les sites participants par des messages *commit*. Un vote *Yes* indique que les opérations locales ont été exécutées avec succès et que les mises à jours peuvent être rendues permanentes ou durables même si une panne survient. Un participant qui vote *No* peut annuler sa branche de transaction de façon unilatérale, tandis que un participant qui vote *Yes* doit attendre la décision du coordinateur avant de soit annuler, soit valider sa branche. Les participants envoient des accusés de réception pour la décision du coordinateur.

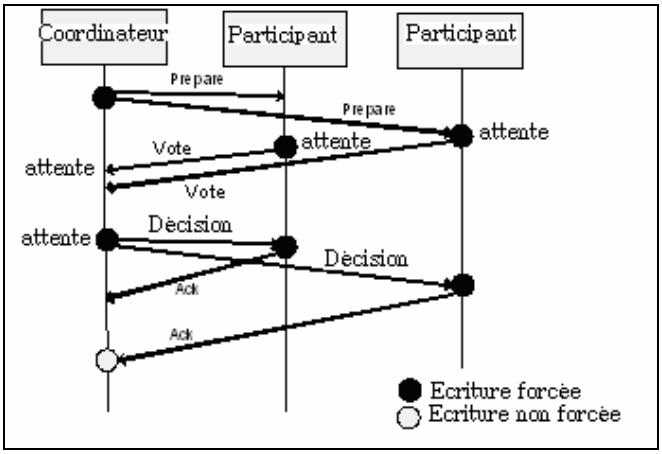


Figure 3.1-Le protocole 2PC

Un protocole de validation atomique doit être tolérant aux pannes (pannes de sites ou pannes de communication). Pour ce faire, les différentes étapes du protocole 2PC sont enregistrées dans des *journaux (logs)* maintenus au niveau du coordinateur et des participants. Les journaux ou *logs* contiennent des enregistrements des données de contrôle de transaction (prepare, commit, abort, end transaction) et des données de manipulation (mises à jour, informations pour défaire *(undo)* et/ou pour refaire *(redo)*). Le journal du coordinateur contient, en plus de ces enregistrements, les identités de tous les participants. Les journaux sont utilisés durant le recouvrement. Comme les pannes peuvent apparaître à tout moment, certains enregistrements sont immédiatement sauvegardés par une écriture forcée sur un support de stockage fiable (*force-written;* written by blocking I/O). Une écriture est dite *non forcée* si l'enregistrement correspondant est mémorisé dans un journal puis reporté de façon asynchrone sur un espace de stockage stable. Une écriture est dite *forcée* si elle impose le report immédiat du contenu du journal sur un espace de stockage stable. Le résultat d'une écriture forcée n'est jamais perdu en cas de panne. Cette distinction est importante, car elle détermine le nombre d'entrées/sorties disque bloquantes nécessaire au bon déroulement du protocole.

3.1.3 Les optimisations et les variantes du protocole 2PC

Le protocole générique 2PC, aussi appelé “sans presemption” ou “presumed nothing” et noté PrN, a les inconvénients suivants: son coût en messages; *4n* messages sont necessaries pour valider une transaction s’excusant sur *n* sites, et sa probabilité élevée de blocage en cas de panne du coordinateur. De nombreuses optimisations et variantes ont été apportées au protocole 2PC [Tra92, LAE94, BCF+97]. Ces optimisations portent sur trois facteurs : la réduction du nombre de messages, la réduction du nombre d'écritures forcées dans les journaux du coordinateur et des participants et enfin la réduction de la latence du protocole. La latence d'un protocole de validation est le temps écoulé entre le moment où l'application décide de terminer une transaction et le moment où cette terminaison devient effective pour chaque participant. Réduire la latence d'un protocole de validation permet de réduire la durée d'une transaction et de ce fait, de libérer plus rapidement les ressources qu'elle détient sur chaque participant. Un état de l’art sur la validation atomique dans les systèmes répartis est présenté dans [AP98] et des résultats d’expérimentation sont donnés dans [LAE94].

Les optimisations *Read-Only* et *One-Phase Commit* sont les plus courantes et les plus simples. Elles sont intégrées dans les standards comme OSI-TP de l'ISO. Lorsqu’un participant qui n’a effectué que des lectures reçoit la demande de vote du coordinateur, il répond par le message "Read-Only" plutôt que par un vote et il termine unilatéralement sa

branche de transaction. Ceci permet le relâchement prématuré des ressources locales à ce participant. L'optimisation One-Phase Commit permet à une transaction qui n'a accédé qu'à un seul site de ramener le protocole au simple envoi de la décision par le coordinateur à l'unique participant, sans vote préalable. Le *2PC décentralisé (D2PC)* conserve la même structure que le 2PC traditionnel, c'est à dire une validation à deux phases, mais il réduit le nombre d'étapes de communication. Dans le 2PC traditionnel, qualifié de centralisé, le coordinateur diffuse une demande de vote à tous les participants et centralise les réponses avant de prendre sa décision. Dans le 2PC décentralisé, le coordinateur diffuse une demande de vote à tous les participants et chaque participant diffuse son vote à tous les autres. En l'absence de pannes, chaque participant peut ainsi prendre une décision de façon décentralisée. L'optimisation *presumed commit (PrC)* [MLO86] réduit le nombre de messages échangés ainsi que le nombre d'écritures forcées dans les journaux dans le cas où une transaction est validée et l'optimisation *presumed abort (PrA)* [MLO86] réduit le nombre de messages échangés ainsi que le nombre d'écritures forcées dans les journaux dans le cas où une transaction est abandonnée.

Les protocoles *Early prepare (EP)* [SC90], *Coordinator Log (CL)* [SC90] et *Implicit-Yes Vote* [AC95] sont des protocoles à une phase. L'idée de One-phase Commit *(1PC)* suggérée dans [Gra78] a pour principe d'éliminer la phase de vote du 2PC qui permet au coordinateur de vérifier si oui ou non les participants de la transaction peuvent garantir localement les propriétés ACID de leur branche de transaction au moment de la validation. 1PC garantie les propriétés ACI avant le moment de la validation. Pour cela, chaque participant se met dans l'état "prêt à valider" après l'exécution de chaque opération d'une transaction et avant d'acquitter cette opération en effectuant une sérialisation continue et un contrôle d'intégrité immédiat. De ce fait, la première phase du protocole 2PC, c'est à dire la phase de vote, est intégrée à l'exécution même de la transaction. Si au moins une opération échoue pendant l'exécution de la transaction, celle-ci est abandonnée, si toutes les opérations sont exécutées avec succès, le coordinateur enregistre la décision de valider par une écriture forcée dans son journal, puis diffuse cette décision à tous les participants sans attendre d'acquittement et enfin oublie la transaction. Ces protocoles imposent des contraintes fortes. Tout d'abord, ces protocoles ne s'appliquent qu'à des participants utilisant un protocole de contrôle de concurrence pessimiste et continu. D'autre part, ces protocoles étendent la fenêtre de vulnérabilité à toute la durée d'exécution d'une transaction. En effet, dès lors que la validation s'effectue en une seule phase, la panne du coordinateur laisse l'ensemble des participants dans un état indécis, car l'abandon unilatéral d'une branche de transaction devient impossible sans l'avis du coordinateur.

3.1.4 Le blocage des protocoles de validation

Un protocole de validation est dit *bloquant* si, en cas de panne d'un ou de plusieurs sites, les sites opérationnels peuvent se retrouver bloqués jusqu'à la reprise des sites défaillants avant de pouvoir terminer la transaction. Le protocole 2PC est bloquant car une panne du coordinateur peut bloquer les participants qui sont en attente de sa décision. Des protocoles de terminaison ont été proposés pour résoudre cette situation de blocage, mais ces protocoles ne peuvent pas débloquer toutes les situations [Gra78, Ske81]. Ceci a donné naissance au problème de la *validation atomique non-bloquante* qui a pour but de garantir, en plus de la propriété d'atomicité, la propriété de *terminaison*. Cette propriété permet à tous les sites qui ne sont pas en pannes d'aboutir à une décision uniforme concernant une transaction, malgré la panne d'un ou de plusieurs autres sites. Un protocole de validation atomique non-bloquant satisfait les quatre propriétés citées ci-dessus et la propriété suivante.

Terminaison: tous les participants corrects qui exécutent le protocole de validation atomique finissent par prendre une décision.

Pour l'étude de la validation non-bloquante, il est supposé que les canaux de communication entre les sites sont fiables. En effet, il a été démontré qu'un système distribué à communication non fiable n'admet pas de solution au problème de la validation atomique non-bloquante [Gra78, Ske81]. A partir de cette hypothèse, le problème a été considéré pour les systèmes distribués *synchrone* et les systèmes distribués *asynchrones.* Un système distribué est dit *synchrone* s'il existe une borne supérieure sur le délai d'acheminement des messages sur un canal de communication. Cette constante doit être connue et peut être exploitée par tous les sites du système. Un système réparti *asynchrone* est caractérisé par l'absence de borne supérieure sur le temps d'acheminement des messages. Dans ce contexte, il est impossible de différencier un site qui est en panne d'un site avec lequel les communications sont extrêmement lentes.

Le premier protocole de validation non-bloquant qui a été proposé pour des systèmes synchrones est le protocole à trois phases, appelé aussi 3PC (Three-phase Commit) [Ske81]. 3PC est obtenu par une extension directe du 2PC en ajoutant à ce dernier une phase supplémentaire, appelée *phase de pré-validation*. Pour remédier à la latence élevée du 3PC, Babaoglu et Toueg ont proposé un nouveau protocole de validation non-bloquant appelé ACP-UTRB (ACP-Uniform Timed Reliable Broadcast) [BT93]. ACP-UTRB a la même structure que le 2PC (i.e., deux phases pour terminer une transaction) tout en étant non-bloquant grâce à l'utilisation d'une primitive de diffusion, appelée Diffusion Fiable ou Atomique*,* qui garantit que soit tous les participants délivrent le message de décision, soit aucun ne le fait.

Dans les systèmes asynchrones, la validation atomique répartie s'avère difficile à résoudre. En effet, le protocole 3PC suppose que la détection des défaillances est fiable. Cette hypothèse est irréaliste dans un environnement où l'asynchronisme peut conduire à des suspicions erronées des sites corrects. Or, Fisher, Lynch et Paterson ont démontré qu'il est *impossible* de résoudre d'une façon déterministe un problème d'accord tolérant aux pannes dans un système asynchrone du fait de la non fiabilité des détections de pannes [FLP85]. Ce résultat est communément appelé *résultat d'impossibilité FLP*. Chandra et Toueg ont alors proposé de contourner ce problème par l'utilisation de mécanismes de détection de pannes qui peuvent "faire des erreurs" en introduisant le concept de *détecteurs de pannes non fiables* [CT96]. Plus précisément, ils ont montré que le problème de *consensus*, une forme abstraite du problème d'accord, peut être résolu dans un système asynchrone augmenté d'un détecteur de pannes non fiable. Un tel détecteur peut être associé à chaque processus (site) pour lui indiquer la liste des autres processus qu'il suspecte d'être défaillants. De tels détecteurs peuvent être mis en œuvre à l'aide de délais de garde (timeouts) mais leurs suspicions peuvent être erronées.

3.1.4.1 Validation atomique non bloquante et consensus

Guerraoui a montré que, contrairement au consensus, le problème de la validation non-bloquante ou NB-AC (Non Blocking Atomic Commitment) ne peut pas être résolu dans un système asynchrone avec des détecteurs de pannes non-fiables à cause de la propriété de non-trivialité. En effet, cette propriété nécessite une connaissance précise, donc fiable, des pannes des participants [Gue95]. En affaiblissant la propriété de non trivialité, on obtient un nouveau problème, appelée *validation atomique non-bloquante faible*, qui peut être résolu dans un système asynchrone augmenté de détecteurs de pannes non fiables [GS95]. La validation atomique non-bloquante faible est définie par les mêmes propriétés d'accord, de validité et de terminaison que la validation atomique non-bloquante initiale, et par la propriété de non trivialité suivante:

- **Non trivialité faible**: si tous les participants votent oui et aucun participant n'est soupçonné, alors tous les participants décident de valider.

Le protocole DNB-AC [GS95] exploite le *protocole de consensus* proposé dans [CT96] pour une *validation atomique non-bloquante faible*. DNB-AC a le même comportement qu'un protocole 3PC décentralisé mais il tolère les cas de pannes dans un système asynchrone grâce au protocole de consensus non bloquant utilisé pour la terminaison en cas de panne d'un ou de plusieurs participants. DNB-AC est formé de trois séquences de messages. Pendant la première étape, un coordinateur envoie une demande de vote à tous les participants. Le protocole devient ensuite décentralisé (i.e., il n'y a plus de coordinateur). Dans la deuxième étape, un participant qui vote *oui*, envoie son vote à tous les autres participants. Dans la troisième et dernière étape, un participant qui reçoit un vote oui de la part de tous les participants, envoie un message pré-validation à ces derniers. Finalement, lorsqu'un participant reçoit tous les messages pré-validation, il décide de valider la transaction.

Le problème du consensus est traditionnellement défini pour des communications fiables et des défaillances de type arrêt immédiat pour les processus (Lynch, 1996). Chaque processus propose une valeur initiale choisie librement. Tout processus non défaillant doit choisir de façon irrévocable l'une des valeurs proposées comme valeur de décision (finale). On a consensus si tous les processus qui décident choisissent la même valeur.

Un algorithme résout le consensus pour f défaillances, $0 < f < n$ (n étant le nombre de processus), si lors de chaque exécution de cet algorithme en présence de f défaillances au plus, on a les propriétés suivantes :

- **Accord** : tous les processus décident identiquement.
- **Validité** : la valeur de décision est l'une des valeurs proposées.
- **Terminaison** : tout processus correct finit par décider.

Le consensus est posé par des besoins applicatifs comme, par exemple, élection de leader, changement de mode opérationnel, reconfiguration (par exemple, exclusion ou non exclusion d'un véhicule), démarrage coordonné d'une opération applicative distribuée.

Le problème NB-AC ressemble au problème de consensus du fait que tout deux nécessite que les processus atteignent une décision commune parmi deux possibles même en présence de pannes de processus. Cependant, il a été observé que [Gue02]:

- Dans NB-AC, un processus peut décider de valider si tous les processus participants à la transaction votent *yes*. Si l'on revoie les propriétés, ceci indique que le consensus n'est pas un cas particulier du NB-AC.

- Dans NB-AC, les processus peuvent tous voter *yes* et ensuite décider *abort*. Dans le consensus, les processus devraient décider *commit* dans ce cas-ci. Ceci signifie que NB-AC n'est pas un cas particulier de consensus.

Guerraoui a aussi montré qu'en fait il existe des différences entre les deux problèmes et qu'ils sont *incomparables* (si au moins deux processus peuvent tomber en panne mais que la majorité des processus est correcte) [Gue02]. Il montre qu'il existe un détecteur de panne qui résout NB-AC mais qui ne résout pas le consensus et un détecteur de panne qui résout le consensus mais pas le NB-AC. Il propose d'ajuster les détecteurs de pannes dans les ACPs qui sont ramenés, en pratique, à l'utilisation de délais de garde. Il est suggéré d'employer deux timeouts pour chacune des phases. L'un ayant une valeur suffisamment élevée pour éviter une fausse détection de pannes, durant la phase de préparation, qui conduirait à des annulations de transactions pouvant en réalité être évitées. L'autre timeout pourrait avoir une valeur relativement faible pour permettre une réaction rapide au pannes durant la phase d'accord

(exécution d'un algorithme de consensus comme sous protocole); les fausses suspicions pourraient retarder le délais de la décision mais n'a pas d'impact sur le résultat.

Gray et Lamport ont de leur côté proposé un protocole de validation non-bloquant basé sur le consensus appelé Paxos Commit Protocol [GL06]. Ils utilisent *2f+1* coordinateurs en tolérant que *f* d'entre eux tombent en panne. Pour n participants, *2fN* messages de plus que 2PC sont produits si aucune panne ne survient. Ce coût est considéré comme étant supportable dans le cas des réseaux locaux filaires. Le 2PC est obtenu comme le cas particulier où *f*=0.

Les protocoles de validation utilisant le consensus, ont généralement employé l'algorithme du consensus comme un protocole de terminaison en cas de panne (ou suspicion) [GS95, Gue02] alors que Paxos Commit implémente l'algorithme de validation avec l'algorithme du consensus Paxos ce qui contredit à première vu les résultats de [Gue95]. Mais en fait, dans [Gue95] la définition de la validation est plus forte car il est exigée qu'une transaction soit validée si tous les participants ne tombent pas en panne et choisissent de valider tout en tolérant des délais de transmission indéterminés. Alors que [GL06] exige que le réseau entier ne tombe pas en panne, ce qui signifie que tous les messages envoyés entre les nœuds sont délivrés en un temps limite connu. Cette hypothèse est plus faible.

L'abondance de la littérature sur le protocole de validation à deux phases et ses variantes jusqu'à récemment montre l'importance d'un tel paradigme pour les systèmes distribués. Ce paradigme qui a été aussi revu dans le cadre des systèmes multibases de données, connaît actuellement un regain d'intérêt dans le cadre des systèmes mobiles. Le reste de ce chapitre est consacré à la validation de transactions en environnement mobile.

3.2 Impact de la mobilité sur la validation des transactions

Nous commençons cette section par monter l'impact de la mobilité sur les protocoles de validation atomiques traditionnels. Pour cela nous nous basons sur le protocole 2PC qui comme nous l'avons vu est le plus répandu et que les autres protocoles ne sont souvent que ses variantes.

Comme c'est le cas dans d'autres protocoles distribués, le protocole 2PC est basé sur l'hypothèse que toutes les parties participant dans l'exécution du protocole s'exécutent sur des sites fixes, équipés par des ressources de calcul et d'alimentation suffisantes, et que les échanges de messages se font sur des réseaux filaires dotés d'une bande passante disponible en permanence [CN03]. Ces hypothèses ne sont plus vérifiées dans le nouvel environnement où les sites peuvent être des terminaux portable plus ou moins faiblement équipées en ressources (CPU, mémoire, et énergie) et communiquent via des liens sans fil. La communication non filaire induit une bande passante beaucoup plus faible, une latence, un taux d'erreurs et un coût plus élevés. Dans cette section nous allons montrer l'impact des caractéristiques de l'environnement sans fils et mobile sur l'exécution du protocole de validation atomique 2PC.

Deux types d'entités coopèrent dans l'exécution du protocole: le coordinateur et les participants. Le coordinateur réside en général sur le site où la transaction est initiée. Les participants sont des processus s'exécutant sur les sites où sont accédées les ressources. Pour le recouvrement, des journaux *(Logs)* doivent être sauvegardés sur un espace de stockage fiable et permanent au niveau de chaque site y compris celui du coordinateur. Nous supposons qu'une transaction mobile T_m est initiée par un client mobile s'exécutant sur une UM.

i- Les communications sans fils. Selon le scénario 2PC, le coordinateur devrait s'exécuter (même si ceci n'est pas une exigence stricte) sur l'UM initiatrice. Le protocole 2PC nécessite deux rondes de messages et *4n* messages, *n* étant le nombre de participants. Ceci ne peut être

toléré du fait du coût élevé, la bande passante faible et variable et la latence élevée des communications sans fil entre le client mobile et le serveur. Afin de minimiser l'utilisation des communications sans fil, il est nécessaire que le rôle de l'UM soit aussi minimal que possible. Ceci signifie principalement que le coordinateur ne doit pas être exécuté sur l'UM. Dans la littérature, la SB (chapitre 2) est choisie pour héberger le coordinateur. Ainsi, le nombre de messages transitant sur le réseau sans fil, en particulier en partance de l'UM est significativement réduit. D'autre part, vu que les messages transitent par la SB, ils pourraient augmenter la latence du protocole. Lorsque le coordinateur est sur la partie fixe du réseau, les communications entre les participants et le coordinateur ne transitent pas par la SB.

ii- Les limites en capacité de calcul et des sources d'énergie des UMs. La responsabilité du coordinateur nécessite des capacités de traitement et de stockage, par exemple, pour les journaux. Malheureusement, les UMs sont généralement légères et ne possèdent pas des ressources suffisantes. Le rôle du coordinateur nécessite aussi des capacités de traitement transactionnel qui pourraient ne pas être fournies sur certaines UMs. Malgré d'énormes progrès en termes de capacité de traitement, la source d'énergie restera limitée dans le futur. Ainsi, la conservation d'énergie elle aussi plaide en faveur de la minimisation du rôle de l'UM, de ses échanges de messages et du placement de la charge sur la partie fixe du réseau. De plus, les journaux seront plus saufs sur le réseau fixe (chapitre 2).

iii- La mobilité des UMs les rend sujettes à des migrations imprévisibles. Les UMs peuvent se connecter sur n'importe quelle SB, à n'importe quel moment et peuvent aussi se déplacer tout en restant connectées. Lorsqu'une UM se déplace d'une cellule à l'autre, il est probable qu'elle ne puisse pas communiquer avec le coordinateur. Ceci a un impact surtout sur les situations de recouvrement où le coordinateur et le client ont besoin d'être en contact afin de terminer la transaction. Deux strategies sont alors possibles:

-Une coordination statique dans laquelle le coordinateur réside à la même place durant toute l'exécution du protocole. Il est nécessaire de faire en sorte que le coordinateur connaisse la nouvelle localisation de l'UM à chaque fois qu'il se déplace. La distance entre le coordinateur et l'UM peut augmenter la latence du protocole et sa fenêtre de vulnérabilité[14].

-Une coordination dynamique dans laquelle le coordinateur se déplace en même temps que l'UM, c.-à-d; le coordinateur résidera toujours dans la même cellule que celle de l'UM. L'inconvénient d'une telle stratégie est le coût qui peut être introduit par des migrations éventuellement fréquentes. Le handoff ou migration du coordinateur consiste à transférer les informations d'état de l'ancien coordinateur vers le nouveau. Ce dernier doit aussi informer tous les participants de ce changement. Ce processus introduit des messages supplémentaires et peut augmenter la latence du protocole.

iv- Une UM peut se **déconnecter** volontairement pour plusieurs raisons, par exemple, lorsque l'utilisateur se déconnecte pour ne pas être dérangé dans un meeting ou pour réduire sa batterie. Les déconnexions peuvent aussi apparaître involontairement et imprévisiblement, par exemple, lorsque l'UM traverse une zone non couverte (dans un tunnel), à cause d'une panne de batterie ou du terminal. Si le protocole 2PC est exécuté en environnent mobile, les déconnexions donneront lieu à une augmentation du taux d'annulation des transactions car les déconnexions seront traitées comme des pannes. La situation peut s'empirer car les déconnexions sont de longue durée. Ceci est inacceptable en environnement mobile car les déconnexions fréquentes ne sont pas des situations exceptionnelles mais représentent plutôt un mode normal d'opération, et ne peuvent de ce fait être traitées comme des pannes.

[14] La fenêtre de vulnérabilité est la période s'étalant entre la phase de vote et la phase de décision.

Contrairement au 2PC traditionnel, un protocole ne doit pas compter sur une disponibilité continue des UMs pour participer à la validation d'une transaction.

Ajoutons aux inconvénients précédents, la difficulté à compter sur les UMs pour la sauvegarde des journaux. En effet, les UMs ne sont pas fiables car elles sont sujettes à la perte, destruction ou vol. Ainsi, l'UM n'est pas un bon choix pour l'hébergement d'un coordinateur.

Dans le cas où la transaction mobile implique des serveurs mobiles la même analyse peut être faite et les mêmes problèmes se retrouvent concernant les serveurs mobiles.

L'analyse de l'impact des contraintes de l'environnement mobile sur le 2PC standard peut être trouvée dans [NDD04].

Dans ce qui suit nous présentons les travaux consacrés à la validation des transactions mobiles (TM). Les travaux résumés dans la section 3.3.1 concernent les modèles de transaction du chapitre 2, qui sont basés sur une exécution sur l'UM ou sur les UFs (mode d'exécution 1 et 2). La section 3.3.2 regroupe les travaux qui se sont concentrés sur la proposition d'un protocole de validation pour des transactions mobiles dont l'exécution est répartie entre UMs et UFs.

3.2.1 Les solutions proposées dans le cadre de modèles de transactions mobiles

Dans la majorité des cas les modèles proposés font la validation des transactions en deux étapes. La première se réalise sur les UMs et la seconde sur le serveur de base de données. IOT, Clustering, Two-tier replication, Twin transactions, Pro-motion, HiCoMo, et Prewrite exécutent une validation locale avec chacune des caractéristiques spécifiques. Clustering et Two-tier replication font une validation locale uniquement en mode déconnecté en utilisant des transactions particulières. En mode connecté, un protocole de validation est utilisé. Dans ce cas, plusieurs sites peuvent participer à la validation. HiCoMo, Pro-motion et Prewrite ne font pas la différence entre le mode connecté et le mode déconnecté. IOT fait aussi une validation locale (en mode connecté et déconnecté) mais la récupération en cas de panne n'est pas garantie à cause du stockage limité sur les UMs. En fait ces modèles sont basés sur le modèle d'exécution 2 qui permet d'exécuter une transaction mobile sur l'UM. Dans la seconde étape de validation, les transactions validées localement consolident les modifications permanentes sur le serveur de la base de données. La validation de la transaction peut comprendre des mécanismes de réconciliation ou la ré- exécution de transactions.

La réconciliation dans Clustering se fait d'une manière syntaxique; les transactions faibles sont annulées ou défaites si leurs écritures faibles sont en conflit avec les transactions strictes. Dans Two-tier replication, les transactions tentatives sont ré- exécutées dans leur ordre de validation. Si cette ré- exécution échoue, alors la transaction tentative est annulée. Afin d'augmenter les chances de réussite, on peut considérer des transactions ayant des opérations commutatives. Dans HiCoMo on augmente les chances de validation en ayant cette propriété de commutativité mais aussi en acceptant des marges d'erreurs tolérées et la ré- exécution des transactions bases.

IOT fournit quatre options pour réconcilier les transactions en attente: (1) ré- exécuter les transactions en utilisant les fichiers à jour du serveur (ceci est l'option par défaut), (2) invoquer l'Application Specific Resolver (ASR) des transactions, ici, le programmeur des transactions peut attacher à une transaction un ASR qui sera automatiquement invoqué par le système, (3) annulation de la transaction et (4) demander aux utilisateurs de résoudre les conflits manuellement. Dans Pro-motion, les compacts impliqués dans les transactions validées localement sont vérifiés. Si un compact n'est plus valable alors les transactions

mobiles sont annulées et des -procédures de contingence- (attachées à chaque validation locale) sont exécutées afin d'obtenir une atomicité sémantique. Prewrite répartie l'exécution entre l'UM et l'UF et utilise le verrouillage afin d'assure la validation des données pré- écrites sur la partie fixe. Dans Semantics-based, les transactions s'exécutent localement sur des fragments d'objets chargés sur l'UM. Ces fragments sont des copies exclusives qu'une transaction peut manipuler localement. A la terminaison de la transaction, l'UM retourne les fragments au serveur qui les regroupe de nouveau par une opération de fusion.

Les modèles Reporting, MDSTPM, Kangaroo, Pre-serialisation, Terminaux alternatifs et Moflex utilisent le mode d'exécution 1 qui consiste à exécuter les transactions mobiles entièrement sur la partie fixe. Dans ce cas la validation est aussi exécutée sur la partie fixe avec aussi des spécificités pour chaque cas. Souvent c'est l'atomicité sémantique qui est offerte dans ces systèmes qui utilisent des transactions de compensation afin d'offrir une atomicité relâchée.

L'idée proposée dans le modèle Batch commit est d'exécuter la transaction entière localement sur l'unité mobile moyennant des copies caches des données, et pour sa validation, la transaction entière est envoyée à un serveur de bases de données situé sur un site fixe pour le reste des opérations. Twin transactions est conçu pour assurer la synchronisation de copies multiples de données et utilise des règles de résolution de conflits. L'annulation et la compensation sont admises.

Reporting et AMT admettent le mode d'exécution 5 qui répartie l'exécution des transactions mobiles sur un ensemble d'UMs et d'UFs. Tous deux offrent l'atomicité sémantique avec la particularité pour AMT de supporter les 5 modèles d'exécution et d'offrir un protocole spécialement conçu pour la validation répartie impliquant tous les sites participant à l'exécution de la transaction mobile. Le modèle Team transactions supporte le modèle d'exécution 6; c'est-à-dire, réparti entre plusieurs UMs mais en mode de communication sans infrastructure. Les nœuds enfants rapportent leurs résultats périodiquement et délèguent la validation au coordinateur qui prend la décision finale de valider ou non la transaction.

Mis à part Reporting, AMT et Team transactions, la validation dans les modèles cités procède en deux étapes, l'une sur l'UM et l'autre sur la parie fixe avec en général la participation des stations bases et des serveurs fixes. Dans la deuxième étape interviennent des techniques de synchronisation ou de réconciliation et parfois même de compensation. Il y a aussi dans certains modèles une distinction entre le mode connecté et le mode déconnecté. Parfois le protocole 2PC ou une de ses variantes est utilisé en mode connecté ou dans les modèles multibases comme MDSTP, Pre-serialisation ou Moflex.

3.2.2 Protocoles répartis de validation pour l'environnement mobile

Dans la section qui suit nous étendons notre analyse et considérons les travaux consacrés plus spécifiquement aux protocoles répartis de validation de transactions mobiles. L'objectif de ces propositions n'est pas de proposer un nouveau modèle de transaction mais d'assurer la propriété d'atomicité. Les participants peuvent être des sites mobiles ou fixes (modèles d'exécution 3, 4, 5). En général, la motivation derrière ces protocoles est de fournir un processus de validation mobile qui prend en compte (1) les caractéristiques limitées des réseaux sans fil en réduisant le nombre de messages et en palliant aux déconnexions et (2) la nature mobile et portable des UMs en considérant les SBs dans le processus de validation [NDD06b].

3.2.2.1 Le protocole TCOT

Le protocole TCOT *(Timeout-Based Mobile Transaction Commitment)* valide les transactions avec un nombre minimum de messages ascendants (dans le sens client vers serveur) [KDDS00, KPDS02]. Le système envisagé offre un mode de connectivité appelé "connectivité mobile" (*mobile connectivity)* qui permet aux clients de rester connectés tout le temps au réseau via le canal sans fil indépendamment de leurs états (mobile, statique, en veille, etc.) et localisations en opposition au mode de "connectivité intermittente" *(intermittent connectivity)* où un client décide volontairement quand se connecter/déconnecter (figure 3.2).

Dans TCOT une transaction T_m est divisée en fragments qui sont exécutés sur différents sites (le premier fragment est exécuté au niveau de l'UM initiatrice). TCOT est une approche optimiste qui suppose qu'il est possible de définir un timeout qui permettrait à des fragments de transaction de se terminer et valider avant son expiration au niveau de chaque site avec un risque minimum d'annulations. Le timeout est une fonction de plusieurs variables système et de communication (charge, taux d'E/S, taux de succès d'accès au cache, etc.). La SB coordinatrice envoie les fragments aux systèmes de bases de données correspondants pour la mise à jour de la copie primaire. Le site participant prend une décision locale de valider/annuler son fragment et en informe le coordinateur. L'UM envoie son journal des mises à jour au coordinateur qui à son tour les transmet au système de bases de données une fois que la décision finale de validation/annulation globale est prise. Soit E_t la borne supérieure du délai d'exécution, c.-à-d; suffisamment long pour permettre à un fragment de finir avec succès son exécution et S_t la borne supérieure du délai de transfert des données de l'UM vers le système de bases de données. La validation globale est décidée par le coordinateur s'il recoit le journal des mises à jour de l'UM avant l'expiration de S_t et si les messages de validation émanant de tous les participants sont reçus. Autrementt, si le timeout (Max (E_t)+S_t) expire, ou si un message d'annulation est reçu alors l'annulation globale est annoncée à tous les participants. Si un participant reçoit le message d'abandon après une décision locale de validation, la *compensation* est alors nécessaire.

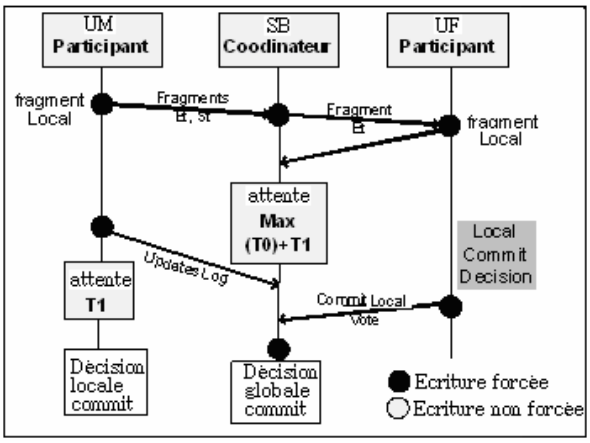


Figure 3.2-Le protocole TCOT

Prise en charge des caractéristiques mobiles

La gestion des déconnexions dans TCOT se concrétise dans le fait de disposer d'une copie de données dans le cache du mobile afin de permettre une exécution autonome de la branche transactionnelle. Le mobile doit ajuster le délais E_t avant son expiration et avant par exemple de se mettre en veille. Le handoff du site mobile affecte principalement le transfert des mises à jour vers le coordinateur adéquat si la localisation de ce dernier change. En cas de handoff TCOT propose une localisation statique ou dynamique pour le coordinateur. Dans le cas de la

localisation statique, le site mobile enregistre l'identité du coordinateur localisé sur la station base où la transaction est initialement enregistrée. Dans le cas dynamique, lors d'un handoff, le mobile doit accepter la nouvelle station base comme coordinateur. Le site mobile doit aussi transférer l'identité de l'ancien coordinateur au nouveau coordinateur durant la phase d'enregistrement dans la nouvelle cellule.

Les terminaux mobiles considérés dans TCOT doivent disposer de capacités de gestion de cache, de la construction et du transfert du journal de mise à jour. Le terminal doit aussi être capable d'extraire sa branche transactionnelle, d'estimer le temps nécessaire au traitement et au transfert des opérations de sa branche et de la compenser en cas d'abandon.

L'analyse de correction et de performance de TCOT montre que dans le cas de fonctionnement normal seuls 2 messages sont nécessaires. Cependant, dans le cas de déplacement rapide et de déconnexions fréquentes, TCOT augmente la probabilité d'annulation et par conséquent le nombre de messages sans fil croît exponentiellement. TCOT demande le transfert du journal des mises à jour vers un support stable avant la validation unilatérale sur le mobile (après le Timeout), ce qui consomme de la bande passante du réseau sans fil.

La formule de calcul de E_t a été obtenue à partir d'une étude analytique, cependant une analyse empirique est nécessaire afin d'effectuer une vérification plus réaliste. De plus, le choix de S_t et son impact sur les performances du protocole n'a pas bénéficié d'une grande attention. En effet, ce facteur est une fonction de paramètres du réseau sans fil qui sont extrêmement variables et fluctuants. L'hypothèse que l'UM soit connectée en continu ne parait pas très réaliste pour les réseaux actuels. La non disponibilité et non fiabilité du réseaux peut très vite accroître le taux d'annulation et la compensation ne satisfait pas toutes les applications. Des serveurs mobiles n'ont pas été considérés.

3.2.2.2 Le protocole UCM

Le but du protocole UCM *(Unilateral Commit for Mobile and disconnected computing)* [BPA00] est de supporter le fonctionnement en mode déconnecté, des clients et serveurs légers et mobiles (figure 3.3). Une transaction s'exécutant en mode déconnecté peut valider aussitôt que son journal a été transféré sur la SB sans attendre l'accusé de réception des serveurs fixes car toutes les vérifications prennent place avant le point de validation. Durant l'exécution d'UCM, certains serveurs peuvent se déconnecter, les systèmes qui n'exportent pas d'interface standard 2PC sont acceptés et seulement une ronde de messages est nécessaire. Le protocole UCM est basé sur l'idée de la validation à une phase *(One-phase Commit)*. 1PC élimine la phase de vote du 2PC, durant laquelle le coordinateur vérifie si oui ou non les participants peuvent garantir les propriétés ACI localement. En effet, dans UCM ces propriétés sont garanties par chaque participant avant d'arriver au point de validation, c.-à-d ; durant la phase d'exécution de la transaction [BPA00].

L'infrastructure réseau supposée est celle de la figure 1.1 avec la possibilité d'inclure des cartes à puce (par exemple, un téléphone équipé avec une carte SIM). Cinq types de composants interagissent durant l'exécution d'UCM, à savoir l'*Application*, le *LogAgent* qui journalise chaque opération avant son exécution, les *Participants* qui exécutent ces opérations, le *Coordinator* qui pilote la terminaison du protocole et le *PAgents* qui représentent les participants durant la terminaison du protocole et qui sont utilisés durant le recouvrement. Le *Coordinator* est toujours localisé sur le réseau fixe tandis que les autres composants peuvent être hébergés par une UM. Durant la phase d'exécution, le *LogAgent* enregistre chaque opération avant de l'envoyer au participant par une écriture non forcée. Un accusé de recéption est reçu pour chaque opération. Au moment de la terminaison,

l'application émet une requête commit après reception de tous les accusés de réception. A ce point, les propriétés ACI sont localement vérifiées par les participants. La propriété de Durabilité est assurée par le *Coordinator* qui reçoit le journal du *LogAgent* et le sauvegarde par une écriture forcée avant de diffuser la décision aux participants.

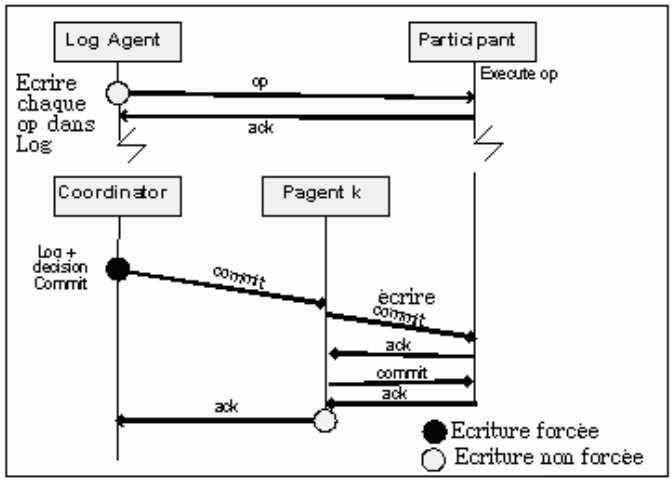


Figure 3.3-Le protocole UCM

Prise en charge des caractéristiques mobiles

Les déconnexions sont traitées par l'adoption de l'approche 1PC qui exige un contrôle de concurrence pessimiste et un contrôle d'intégrité continu pour assurer les propriétés ACI avant l'émission de la requête de validation. Ceci convient bien au serveurs légers qui de par leur construction n'admette q'un utilisateur à la fois. Cependant, cette contrainte n'est pas toujours possible pour les autres types de participants. Si une transaction mobile est générée par un site mobile et fait participer exclusivement des sites fixes, le mobile délègue la station base même s'il est connecté au réseau pour se préserver des déconnexions inattendues durant la validation. Si une transaction mobile générée par un site fixe ou mobile fait participer des sites mobiles, les participants peuvent être indisponibles lors du processus de validation et le risque d'annulation de transactions augmente avec la longévité de la transaction. La procédure de recouvrement compte sur le journal du coordinateur; d'où la limitation de l'autonomie des participants et l'importance cruciale de ce journal. En outre, le transfert du journal d'une transaction sur le réseau sans fil induit un coût.

La réduction du nombre de messages durant la phase de validation se traduit par un nombre élevé de messages échangés (Application – LogAgent – Participant) et d'écritures lors de l'exécution d'une transaction UCM. En d'autres termes, la réduction de la latence du protocole de validation UCM augmente le temps d'exécution globale de la transaction mobile ainsi que sa fenêtre de vulnérabilité. Le problème de handoff n'est pas évoqué de façon explicite.

3.2.2.3 Le protocole CO2PC

Le protocole CO2PC *(Combination of an Optimistic approach and 2PC)* est une combinaison de l'approche optimiste avec 2PC qui assure l'atomicité sémantique en permettant aux participants d'effectuer soit une validation locale optimiste (les résultats sont partagés localement) ou une validation locale non optimiste (figure 3.4) [Ser04, SRAL05]. Les auteurs tentent d'augmenter la flexibilité des participants (particulièrement des UMs), ainsi les transactions compensables peuvent être validées localement d'une manière optimiste, tandis que les transactions non compensables doivent attendre la décision globale. Le coordinateur doit être un participant fixe qui est choisi au moment de l'initialisation. Si aucun

hôte fixe n'est disponible alors un Agent localisé sur la station base jouera ce rôle. Un participant validant de manière optimiste (appelé *opt-participant*) exécute sa branche de transaction et la valide/l'annule unilatéralement. Puis, il envoie son *vote* au coordinateur. Si la décision globale est de valider, les *opt-participants* ont terminés; sinon ils doivent exécuter des transactions de compensation. Un participant non optimiste exécute le 2PC localement avec le SGBD local. Chaque SGBD envoie son *vote* au Client/Serveur qui le fait suivre au coordinateur de CO2PC. Si le *vote* est négatif, le participant abandonne la branche de transaction; sinon la branche entre dans l'état *preparé*. Une fois que la décision du coordinateur arrive aux participants, la décision est exécutée par le SGBD sous-jacent.

Il faut noter que le 2PC est exécuté en local sur le même site et ne génère donc pas de messages sans fil. Seuls les messages *vote et décision sont* transmis sur les liens sans fil (2 messages par participant). 2PC est utilisé par le protocole CO2PC car – pour les transactions non-compensables – les ressources doivent être retenues jusqu'à la validation/annulation globale. Le protocole CO2PC est proposé pour prendre en charge plusieurs types de transactions (non-compensables et compensables) et différents types d'UMs (avec des ressources limitées ou non limitées).

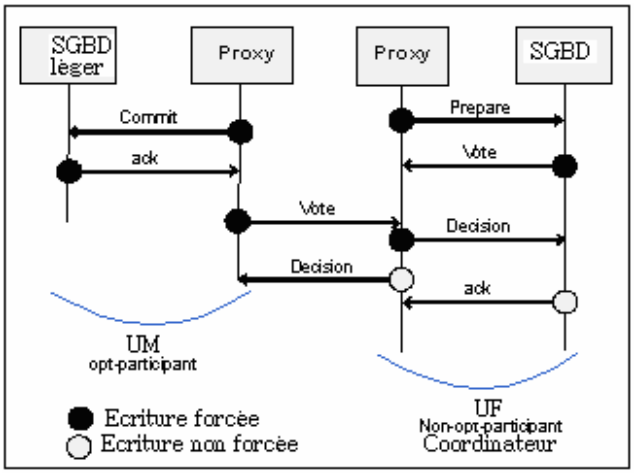


Figure 3.4-Le protocole CO2PC

Prise en charge des caractéristiques mobiles

Un participant non-optimiste peut se bloquer si, après avoir voté pour une validation, un participant est déconnecté avant l'envoie du vote ou si le coordinateur tombe en panne ou se déconnecte indéfiniment. Pour limiter le blocage indéfini et briser d'éventuels interblocages, CO2PC utilise des délais de garde (timeouts). Le participant délègue l'attente de la décision à son agent avant de se déconnecter. Dans ce cas, la décision du coordinateur est journalisée par un Agent et est envoyée à l'UM après sa reconnexion. Les participant optimistes et non optimistes peuvent se déconnecter, sauf que seules les ressources non optimistes restent bloquées jusqu'à la reconnexion.

Le recouvrement est basé sur les journaux du coordinateur et des participants. Cependant ; afin de se protéger des déconnexions, les journaux devraient être sauvegardés avant l'envoi des messages. Pour les UMs le journal doit aussi être sauvegardé par l'Agent sur la SB. Afin de préserver l'hétérogénéité des SGBDs sous-jacents, aucune hypothèse n'est émise sur leur politique de recouvrement. Pour préserver l'autonomie et réduire le coût des communications sans fil, CO2PC délègue la responsabilité d'assurer la durabilités aux SGBDs locaux et donc le transfert des journaux des mises à jour n'est pas exigé avant la validation des transaction. Ces dispositions permettent le relâchement de l'Atomicité et de la Durabilité.

3.2.2.4 Le protocole O2PC-MT

En combinant OCC [ACL87] et 2PC, le protocole O2PC-MT *(Optimistic Two-Phase Commit Protocol for Mobile Transactions)* offre un support pour les transactions longues et permet aux MUs de soumettre des transactions mobiles en plusieurs messages tout en se déplaçant de façon arbitraire [DMW01]. Dans cette architecture, les serveurs de base de données résidant sur les UFs et sur les SBs constituent le système de bases de données réparti. Chaque serveur est autonome et supporte des transactions locales via le mécanisme de contrôle de concurrence Two-Phase Locking. Il supporte des transactions mobiles globales via le schéma O2PC-MT. Chaque SB peut aussi opérer en tant que coordinateur de transaction mobile appelé MTMC *(Mobile Transaction Manager)*. Le MTMC reçoit les opérations des transactions initiées par les UMs et coordonne leurs exécutions et leur validation globale (figure 3.5).

A la réception des premières opérations d'une nouvelle transaction mobile, une SB devient son MTMC et en informe les autres SBs. Les opérations sont envoyées aux participants appelés MTMPs *(Mobile Transaction Manager Participants)* pour l'exécution en une seule étape. Si la prochaine opération ou les données qu'elles désirent accéder ne sont pas disponibles, le MTMC suspendra la transaction mobile jusqu'à ce que la contrainte soit levée. Si le MTMC atteint l'instruction de validation et que toutes les opérations la précédant ont fini leur exécution alors la transaction mobile entamera la validation 2PC.

L'algorithme Two-Phase Commit execute par O2PC-MT est composé de deux phases: la phase Validation et la phase Validation-Globale. Dans la phase Validation, le MTMC envoie le message Prepare à tous les MTMPs. Selon le cas un message Ready ou Abort est retourné au MTMC. En fait, durant cette phase, le MTMP P_i exécutera une transaction base $\mathbf{BT_i}$ qui contient exactement les mêmes opérations que la sous-transaction $\mathbf{T_i}$. En exécutant $\mathbf{BT_i}$, MTMP P_i utilise le mécanisme *Two-Phase Locking* (2PL) et doit détecter deux types de conflits, les conflits de lecture *(Read-Set conflict)* et ceux d'exécution *(During-Execution conflict)*.

Les conflits de lecture signifient que soit les données lues par Ti et BTi sont différentes ou bien que Ti et BTi ont lu des valeurs différentes pour les mêmes éléments de données. Un MTMP P_i détecte les conflits d'exécution lorsqu'une opération de $\mathbf{BT_i}$ échoue. Plusieurs facteurs peuvent favoriser les conflits d'exécution; par exemple, la violation des contraintes d'intégrité ou la tentative de mise à jour d'éléments supprimés. Il est montré dans [DMW01] que pour une sous-transaction $\mathbf{T_i}$ $(1 \le i \le m)$, si aucun conflit n'est détecté durant l'exécution de $\mathbf{BT_i}$, alors $\mathbf{BT_i}$ peut être validée au niveau du MTMP P_i sans violer la serialisabilité ; sinon $\mathbf{BT_i}$ ne peut être validée.

Dans la phase Validation-Globale, le MTMC doit recevoir les messages READY de tous les MTMPs avant l'expiration d'un délai de garde afin de pouvoir diffuser les messages GLOBAL-COMMIT qui permettront aux MTMP P_i $(1 \le i \le m)$ de valider $\mathbf{BT_i}$ et de répondre par ACK au MTMC. Pour terminer, le MTMC transmetta les résultats finaux à l'UM et diffisera un messages aux SBs pour la mise à jour des tables de routage qui ont servi lors du handoff.

Prise en charge des caractéristiques mobiles

L'UM peut se déconnecter après l'envoi des opérations de sa transaction au coordinateur sur la SB, les participants fixes se chargent de traiter ces opérations. La SB coordinatrice informe les autres SBs du réseau par un message sur son rôle de coordinatrice pour la transaction en question. Ce message est sauvegardé dans une table de routage qui permet à ces SBs de transférer les opérations de la transaction mobile à sa SB coordinatrice dans le cas de

handoff. L'exécution se passe sur les UFs et donc ce schéma n'exige pas de grosse ressources pour l'UM. Un seul message sans fil est utilisé durant la phase de validation pour transmettre les résultats finaux vers l'UM (des messages sont nécessaires pour le transfert des opérations de l'UM vers le coordinateur).

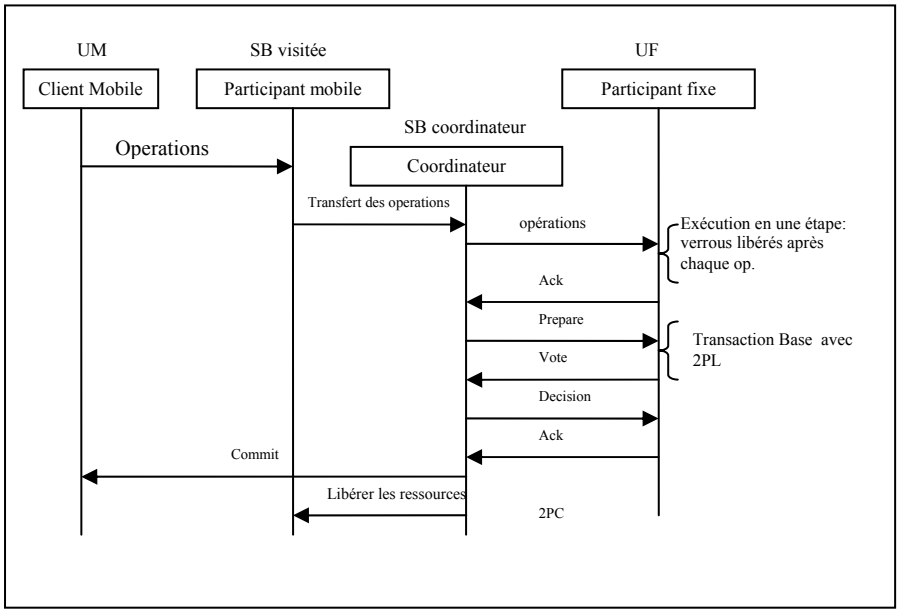


Figure 3.5-Le protocole O2PC-MT

3.2.2.5 Le protocole ICPM

Böse et al. proposent le protocole *(ICPM: Integrated Commit Protocol for Mobile Network Databases)* de validation atomique non bloquante pour les MANETS en combinant les protocoles 2PC, 3PC et le consensus [BBG+05]. Une transaction initiée par un site *initiateur* la divise en sous-transactions non compensables s'exécutant sur plusieurs bases de données. Il est supposé l'existence dans le réseau mobile d'un groupe composé de *f* sites ou nœuds qui sont relativement stables et proches, connectés à une distance d'un saut les uns des autres. Ce groupe est choisi pour jouer le rôle de coordinateurs dans le protocole proposé. Il est également supposé qu'à tout moment, au moins un coordinateur parmi le groupe, appelé aussi *cluster* de coordinateurs, survive. Le cluster agit en tant que coordinateur pour les bases de données jouant le rôle de participants dans le déroulement du 2PC. L'initiateur est le site d'exécution de l'application et il est responsable de désigner les coordinateurs qui sont différents des sites participants. Chaque coordinateur du cluster est un représentant pour une ou plusieurs bases de données. Un *coordinateur principal* prend la décision globale pour le compte de tous les autres coordinateurs grâce à l'exécution du protocole 3PC. Si le coordinateur tombe en panne ou un partitionnement du réseau le rend injoignable, alors un protocole de consensus est exécuté pour terminer la transaction (figure 3.6). Il est montré dans [BBG+05] que contrairement au protocole Paxos commit [GL06] basé sur le consensus qui nécessite au moins (½ *f* +1) coordinateurs opérationnels, le protocole ICPM peut se suffire d'un seul coordinateur pour être non-bloquant dans le cas où il est possible soit d'exclure soit de détecter de façon fiable le partitionnement réseau. En plus, Paxos commit génère *nf* messages de plus que 2PC alors que le protocole intégré en génère *4f* de plus seulement.

L'utilisation du protocole 3PC dans le cluster de coordinateurs permet de pallier au problème de blocage du protocole 2PC qui peuvent être causés par la panne du coordinateur ou le partitionnement du réseau ou la perte de messages. Le blocage n'a lieu que dans le cas où moins de la moitié du nombre de coordinateurs sont opérationnels ou dans le cas d'un état inconnu dans une partition réseau. Le consensus n'est utilisé qu'en cas de panne pour terminer

la transaction. D'autre part, l'emploi de 3PC dans un Manet est très coûteux en nombre de messages, le cluster permet atténuer ce problème vu qu'il tire avantage des communications sans fil à un saut entre les coordinateurs. L'utilisation du 2PC au niveau des bases de données permet d'implémenter le protocole proposé comme une extension aux systèmes existants facilitant ainsi le transfert des applications classiques vers l'environnement mobile.

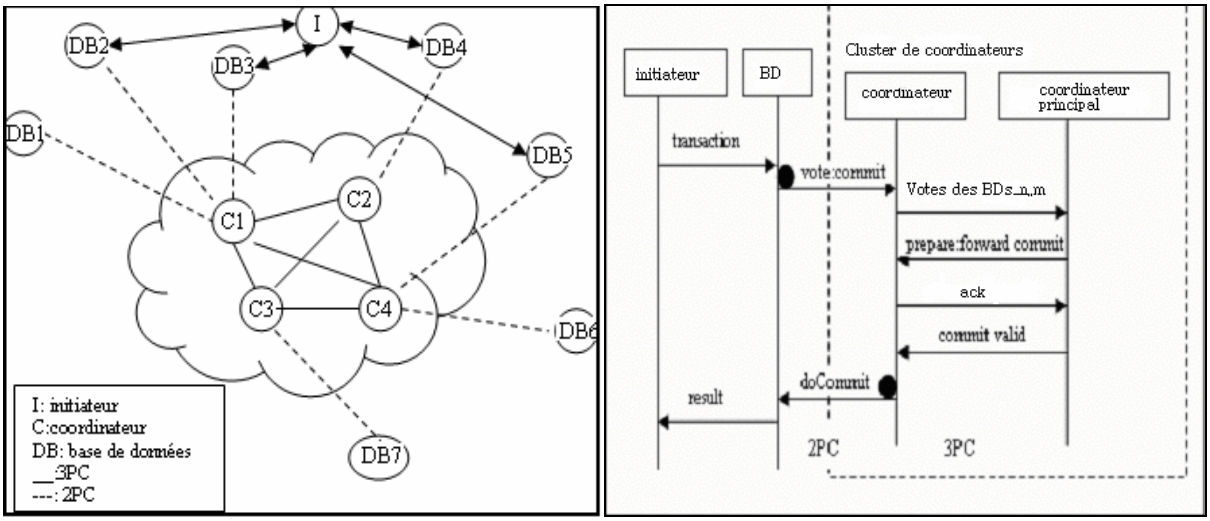


Figure 3.6-Architecture du système et séquence des messages dans le cas sans pannes

Prise en charge des caractéristiques mobiles

Les protocoles 3PC et consensus ont été proposés comme solutions à la validation atomique non bloquante. En environnement mobile, les risques de blocage sont plus importants que dans l'environnement distribué, aussi ces solutions paraissent encore plus indiquées. Cependant, leur coût en nombre de messages qui engendre une consommation de bande passante et de batterie reste non négligeable. Il faut également noter que les UMs doivent posséder des capacités de traitement suffisantes pour participer au traitement transactionnel. D'autre part, seule une évaluation du taux de succès des transactions pourraient indiquer si une telle approche est efficace dans les réseaux MANET ou non, d'autant plus que même en environnement distribué, l'implémentation de ce type d'approche n'est pas encore répandue.

3.2.3 Discussion

Les propositions qui considèrent que les transactions sont exécutées sur des systèmes multibases, tels que MDSTP, Pre-serialisation et Moflex sont confrontés au problème classique de garantir l'atomicité globale qui dépend du niveau d'autonomie des SGBDs du système. Le problème est posé lorsque des SGBDs ne peuvent pas participer dans une validation atomique globale [BGMS92].

Pour les systèmes qui effectuent la validation en deux étapes, sur l'UM puis sur les SBs/serveurs, cela implique le relâchement de l'atomicité et demande l'exécution d'un processus extra. Mais il est intéressant de permettre une exécution locale pour les UMs afin que le travail soit possible en mode déconnecté. Les annulations en cascade qui peuvent apparaître comme dans Clustering, Two-tier replication et Pro-motion, touchent les transactions locales. De plus, ces annulations concernent seulement des transactions faibles et tentatives car les résultats locaux sont disponibles uniquement pour ce type de transactions. IOT prévient l'incohérence en cascade par la notification aux utilisateurs des objets accédés par des transactions non validées. Dans la réconciliation, où une transaction non validée est en

train d'être résolue, l'état d'incohérence local et global de l'objet doit être exposé, de cette façon, le processus de résolution peut choisir entre les deux.

La comparaison entre les protocoles décrits dans la section 3.3.2 est résumée dans la table 3.1. Les critères choisis reflètent le comportement de ces protocoles par rapport aux caractéristiques mobiles. L'objectif des protocoles de validation est d'assurer la propriété d'atomicité. En fait, dans les systèmes traditionnels les protocoles de validation avec le processus de journalisation et recouvrement permettent d'assurer les deux propriétés, atomicité et durabilité. Rappelons que la durabilité assure la persistance des mises à jour après la validation. Dans l'environnement mobile, assurer ces deux propriétés est très contraignant. UCM et TCOT assurent la durabilité mais au prix d'un transfert du journal des mises à jour vers la partie fixe du réseau avant la validation. Ce transfert induit un coût en termes d'utilisation de la bande passante sans fil et de violation de l'autonomie des sites qui sont obligés d'extérioriser les informations du journal et de ne pas pouvoir recouvrir de façon indépendante. En revanche, C02PC opte pour un coût réduit en contre partie des propriétés d'atomicité et de durabilité relâchées. TCOT relâche seulement la propriété d'atomicité.

Les protocoles utilisant des techniques pessimistes de contrôle de concurrence peuvent causer des blocages à cause des déconnexions et réduire ainsi la disponibilité des données. Les techniques optimistes permettent la validation anticipée des transactions et la libération des ressources. Pour cela TCOT et CO2PC comptent sur les transactions de compensation mais TCOT ne supporte pas les déconnexions.

UCM est plus souple que 2PC du fait qu'il ne requiert pas d'interface spécifique, il réduit le nombre d'étapes et de messages et tolère les déconnexions. La particularité de O2PC-MT réside dans le fait qu'il préserve le fonctionnement du 2PC qu'il effectue entièrement sur la partie fixe, son apport concerne l'aspect contrôle de concurrence optimiste qui réduit la durée du maintien des verrous. Le support des déconnexions et de la mobilité est prévu vu que cela est pris en charge par les SBs comme cela est le cas pour UCM et CO2PC.

Le protocole ICPM ne peut pas vraiment être comparé aux autres protocoles car il est conçu pour les environnements ad hoc. Il combine l'utilisation du 2PC, 3PC et un algorithme de consensus afin de réduire au maximum le blocage dû aux défaillances d'UMs ou au partitionnement du réseau causé par les déconnexions fréquentes.

Le protocole 2PC est évidemment le protocole le plus coûteux en termes d'utilisation de la bande passante vu son nombre de messages élevé. Les protocoles à une phase réduisent aussi la latence. Rappelons que l'hétérogénéité est ici définie par rapport à la diversité des UMs et des systèmes de gestion de données. En résumé, les protocoles diffèrent par le fait qu'ils font des compromis différents entre les contraintes imposées et les propriétés offertes qui satisferont des applications et pas d'autres.

Table 3.1-Comparaison des protocoles de validation mobile

	Atomicité	Durabilité	Blocage	Ressources sur l'UM	Nb. Messages sans fil	Latence (Nb. étapes)	Contrôle de concurrence
2PC	stricte	garantie	possible	variable	4n	2	Variable
O2PC-MT	stricte	garantie	possible	faibles	1	2	Optimiste
TCOT	sémantique	garantie	évitable	importantes	2	1	2PL stricte
UCM	stricte	garantie	possible	variables	2	1	2PL stricte
CO2PC	sémantique	Non garantie	possible	importantes	2	1	Optimiste
ICPM	Stricte	garantie	évitable	importantes	4n+nf f: nb coordinateurs	4	-

Déconnexions	Mobilité	Autonomie	Hétérogénéité
Non	Non	Non	Possible
Oui	Oui		_
Non	Oui	Non	non
Oui	-	Non	oui
Oui	-	Oui	oui
Oui	-	-	-

3.3 Conclusion

Dans le futur, les systèmes mobiles seront assez divers avec probablement différents modèles de transactions et modes de traitement. Certains suivront le schéma ACID classique et d'autres non. Les protocoles basés sur la sémantique éliminent la période d'incertitude de terminaison de transaction et le blocage. De tels protocoles sont proposés comme des alternatives à 2PC. Le Principe de ces protocoles alternatifs est de permettre à un participant de valider une transaction de façon unilatérale et de libérer les ressources qu'elle détient. Si la décision finale est une annulation globale, la compensation est utilisée pour défaire de façon sémantique les effets de la transaction annulée. Ces protocoles n'offrent pas l'atomicité stricte comme elle a été définie dans les transactions ACID traditionnelles mais offrent l'atomicité sémantique. Le problème qui apparaît est que la compensation a une applicabilité limitée et que pour de nombreuses applications il est nécessaire d'assurer l'atomicité stricte. Les besoins des applications futures varieront certainement et les deux types de validation stricte ou relâchée doivent être envisagés.

En réalisant l'étude des problèmes rencontrés par les techniques de traitement de transactions mobiles et des protocoles de validation nous avons pu dégager ce qui suit :

-le protocole de validation 2PC a suscité, et continue à le faire, tant d'intérêt dans les systèmes répartis et a gagné en popularité qu'il nous a paru intéressant de le reconsidérer de plus près dans le cas des systèmes mobiles;

-le besoin d'évaluation quantitative et par des méthodes de simulation et/ou d'expérimentation des techniques et protocoles transactionnels mobiles est ressenti. En effet, les propositions rencontrées se sont contentées d'une analyse des cas extrêmes et plus rarement d'évaluations analytiques;

-vu la diversité des besoins transactionnels, nous avons pensé à considérer l'approche de l'adaptation. En effet, comme nous l'avons vu dans les chapitres précédents, les caractéristiques de l'environnement mobile sont variables et dynamiques et leur conjugaison à la diversité des applications suggère une approche qui offre la flexibilité qui permet de s'adapter aux besoins.

A partir de ces points découle l'organisation de la suite de cette thèse. Dans le chapitre 4 qui suit nous présentons nos propres propositions, qui sont le fruit d'une approche plutôt intuitive, concernant la validation des transactions mobiles. Notre première proposition consiste en une adaptation de l'exécution du protocole 2PC prenant en compte les contraintes induites par l'environnement mobile à infrastructure. Notre deuxième proposition est une adaptation de la version répartie du protocole 2PC à l'environnement ad hoc. Le chapitre 5 présente les résultats de simulation et d'évaluation de performances de certains des protocoles présentés dans le présent chapitre ainsi que de notre proposition. L'objectif de cette simulation a été de réaliser une comparaison quantitative des performances des protocoles étudiés, de mesurer plus concrètement l'impact des contraintes des communications sans fil et de la mobilité sur les protocoles de validation de transaction et de dégager des indices d'efficacité quant aux différentes approches proposées. Puis, viennent les chapitres 6 et 7 consacrés à l'adaptabilité des techniques transactionnelles en particulier le modèle de transaction et la validation.

Chapitre IV

Proposition de protocoles de validation de transactions mobiles

If we knew what it was we were doing,
it would not be called research, would it?
-- Albert Einstein

Les protocoles de validation atomique distribués sont conçus d'une manière qui leur permet de fonctionner sur une infrastructure TCP ou UDP. L'objectif de ces protocoles est d'assurer la terminaison correcte et coordonnée entre les différents sites participant à la transaction tout en étant résistants aux pannes des sites et des communications. Cette résistance aux pannes est garantie grâce à un échange de messages entre les sites. Dans les systèmes distribués, les pannes qui peuvent entraver le fonctionnement d'un protocole de validation sont de trois types : la perte de messages, la duplication des messages et la panne d'un site coordinateur ou participant. Intuitivement, il paraît possible de compter sur la prise en charge des problèmes liés aux messages par le système de communication sous-jacent (TCP/IP peut offrir ses garanties, on peut même compter sur l'implémentation d'un service le multicast atomique). Or, même avec un système de communication parfaitement fiable, il n'est pas possible de simplifier la complexité en messages du protocole de validation tel que le 2PC car ses échanges restent nécessaires à la résistance aux pannes des sites. De ce fait, le protocole 2PC ne fait aucune hypothèse sur le sous-système de communication sous-jacent et fonctionne correctement aussi bien sur UDP[15] que sur TCP (ou entre toutes paires d'adresses IP et de numéros de port sur Internet ou un LAN).

Si l'on récapitule les problèmes du 2PC dans un environnement mobile on peut citer la fréquence des pannes dues aux communications sans fil ; les déconnexions ne doivent pas être traitées comme cela est fait avec les pannes citées ci-dessus; autrement le nombre de transactions annulées peut faire chuter gravement le rendement ou le taux de succès *(Througput)*[16] du système. De plus, les capacités de traitement, de stockage et de communication exigées pour le coordinateur font qu'il est préférable d'éviter de désigner une unité mobile comme coordinateur. Enfin, la mobilité du client ou du serveur participant dans le protocole ne doit pas entraver le déroulement des échanges entre les partenaires du protocole qui doivent rester joignables pour éviter de prolonger le temps d'exécution ou

[15] User Datagram Protocol est un protocol non orienté connexion de la couche Transport du modèle TCP/IP; il ne fournit pas de contôle d'erreur.
[16] Le nombre de transactions validées par unité de temps.

encore d'augmenter le nombre des annulations des transactions à cause de l'expiration des délais.

Partant de ce constat, dans ce chapitre nous proposons des approches de validation basées sur l'adaptation du protocole Two-phase commit de manière à pallier le mieux possible aux problèmes ci-dessus. Dans la section 4.1 nous montrons comment on pourrait adapter l'exécution du protocole 2PC dans un environnement mobile à infrastructure et en section 4.2 nous proposons d'utiliser la version distribuée du 2PC dans le cas d'un réseau sans infrastructure. Ce dernier environnement présente des spécificités liées principalement au caractère dynamique de la configuration de l'infrastructure de communication.

4.1 Mobile Two-phase Commit Protocol (M2PC)

L'objectif de notre protocole est de valider une transaction mobile T_m qui s'exécute sur plusieurs sites dont certains sont mobiles. L'architecture matérielle est représentée dans la figure 1.1. Nous supposons que T_m est initiée par une UM que nous avons appelée *Home-MH* et que la SB de rattachement de cette UM est *Home-BS*. Durant son déplacement d'une cellule à l'autre l'UM s'attache à une nouvelle SB que nous appelons *Current-BS*. Au moment de la validation, une demande *commit-request* est émise à partir de *Home-MH* dont la SB courante (il peut s'agir de la Home-BS si l'UM n'a pas migré) devient la SB de validation ou *Commit-BS.*

4.1.1 Le principe

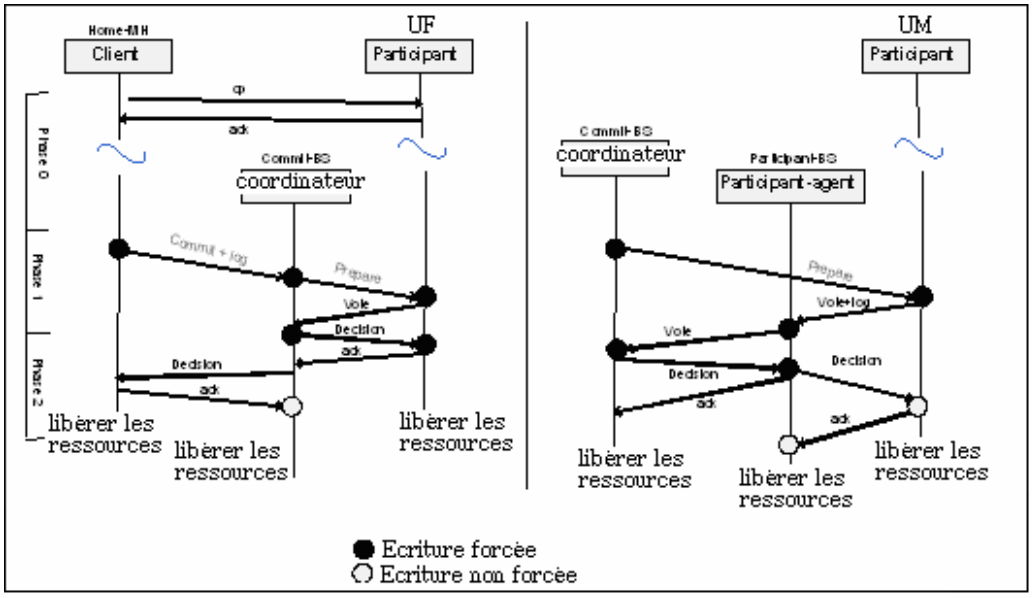
L'architecture matérielle reflète les différents rôles que chaque entité participant dans l'exécution du protocole M2PC doit jouer pour s'adapter d'une manière flexible à la configuration réseau sous-jacente (figure 4.1). De façon similaire au protocole 2PC, il y a trois rôles importants à représenter: l'*initiateur* de transaction qui est l'UM ayant lancé T_m, les *participants* qui sont les entités exécutant les opérations et le *coordinateur* qui coordonne la terminaison cohérente de la transaction. Sur la figure 4.1, l'initiateur est appelé *client***,** les serveurs sont des *participants* et la *commit-BS* est le *Coordinateur*. Plusieurs scénarios peuvent être considérés. Le scénario de la figure 4.1.a prend en compte des clients mobiles et des serveurs fixes alors que le scénario de la figure 4.1.b prend en compte des clients et des serveurs mobiles.

4.1.1.1 Cas d'un client mobile et de serveurs fixes

La stratégie choisie est de diviser les fonctionnalités en deux tâches: la première maintient le même schéma sur la partie fixe du réseau que celui du 2PC traditionnel; la seconde ajuste le schéma pour gérer la partie sans fil. En d'autres termes la coordination des UFs reste fidèle au protocole 2PC avec un *coordinateur* devant résider sur la partie fixe du réseau de façon à être directement en contact avec les participants fixes. Le coordinateur doit aussi être en contact avec le client résident sur l'UM, donc le meilleur choix semble être le placement du coordinateur sur la SB courante *current-BS*[17]. Le protocole M2PC peut se terminer soit dans la même cellule, soit dans une nouvelle cellule couverte par une nouvelle SB. Le *coordinateur* s'exécute sur la première SB qui reçoit la demande de validation; c'est-à-dire, la *commit-BS*. Ceci signifie que le coordinateur peut avec une grande probabilité se trouver aussi proche que possible du client. Le *coordinateur s'*exécute sur la même *commit-BS* même en cas de handoff de l'UM durant le processus de validation. En effet, comme nous considérons la mobilité uniquement pendant le temps d'exécution du protocole de validation, faire migrer le contrôle aussi fréquemment que l'UM se déplace paraît être sans intérêt et cela pour deux raisons. Soit

[17] Si la SB n'a pas les capacités d'hébergement et d'exécution requises pour le coordinateur, ce dernier peut être hébérgé par une autre UF du réseau.

l'UM se déplace lentement et donc la probabilité que le protocole se termine dans la même cellule est élevée. Soit elle se déplace à grande vitesse et donc la migration fréquente du contrôle peut accroître la latence du protocole et par conséquent sa vulnérabilité. Lorsqu'il change de cellule, le client mobile informe le coordinateur de sa nouvelle localisation en lui envoyant un message 'I-m-here'. En fait, contrairement aux protocoles décrits dans le chapitre 3, M2PC suppose que la pile protocolaire du réseau n'offre pas de support de gestion de mobilité. M2PC intégre lui-même le mécanisme de gestion de mobilité contournant ainsi le coût de modification des infrastructure réseau existentes ou des systèmes opérants.



a- Client mobile et serveurs fixes **b-** Client mobile et serveurs mobiles

Figure 4.1-Le protocole M2PC

4.1.1.2 Cas de client et serveurs mobiles

La figure 4.1.b décrit un scénario où au moins un serveur ou *participant* (autre que l'unité initiatrice) est mobile. Dans ce cas, T_m accède à des données localisées sur une UM. Soit une application où un chercheur rencontre des collègues dans une conférence et a besoin de se fixer un rendez-vous avec eux. Dans cette application, les agendas respectifs des chercheurs se trouvant sur leurs portables doivent être synchronisés; les participants sont tous mobiles.

Le protocole M2PC se comporte de manière similaire au cas avec des serveurs fixes. L'idée est d'avoir du côté d'un *participant mobile* un schéma similaire à celui que l'on a du côté du client mobile. Un agent, que nous appelons *participant-agent*, travaillera pour le compte du serveur mobile qui est libre de se déconnecter à partir de l'instant où il lui délègue la fonction de validation comme le fait le client mobile en déléguant le coordinateur après lui avoir envoyé son journal avec la demande de validation. Le *participant-agent* est alors responsable de la transmission des résultats au participant après la reconnexion de ce dernier. Il est aussi responsable de garder les journaux et du recouvrement en cas de panne. Comme dans le cas du client mobile, le *participant* mobile est libre de se déplacer vers une autre cellule durant l'exécution du protocole. Lorsqu'il change d'adresse IP, le *participant mobile* informe son *participant-agent* de sa nouvelle localisation. La charge de travail est déplacée vers la partie fixe du réseau préservant ainsi la capacité de traitement et les ressources de communication des UMs. Le trafic sur les liens sans fil est également minimisé.

4.1.1.3 Prise en charge des caractéristiques mobiles

Pour pallier aux déconnexions, nous faisons en sorte que le client/participant mobile délègue sa part de la validation au coordinateur/participant-agent qui est toujours disponible. Le client/participant envoie la demande de validation/vote au coordinateur/participant-agent en même temps que son journal, après cela, il peut se déconnecter.

Durant l'exécution de M2PC, le client/participant mobile peut changer de cellule et s'enregistrer dans une nouvelle cellule. Contrairement aux solutions suggérant de faire migrer le contrôle, M2PC ne le fait pas puisque le coordinateur/participant-agent reste dans la cellule où il se trouve au moment de la réception de la demande de validation *(commit-request)/vote*. Le protocole M2PC requiert seulement que le coordinateur/participant-agent soit informé de la localisation courante du client/participant (ce qui est effectué par le message 'I-m-here').

Le client/participant doit sauvegarder l'identité et la localisation du coordinateur/participant-agent pour les utiliser lorsqu'il s'enregistre à une nouvelle SB. Cependant, si aucune précaution n'est prise ces informations peuvent être perdues en cas de déconnexion ou panne soudaine. Pour évider cela, le client/participant doit utiliser une écriture forcée pour sauvegarder ces informations juste avant d'envoyer la demande de validation *(commit-request)/vote*. Ainsi, si la *commit-request/vote* est reçu correctement par la SB (coordinateur/participant-agent), on peut être sûr que l'écriture forcée a eu lieu. Comme le coordinateur/participant-agent est statique, une seule écriture forcée est nécessaire.

L'exécution effective du processus de validation a lieu sur la partie fixe du réseau, ce qui réduit le coût des communications sans fil et la latence du protocole.

4.1.2 Correction du protocole M2PC

Le protocole M2PC utilise le même schéma d'exécution que le 2PC dont la correction a été largement prouvée [BHG87, WV02, Gra93]. La correction en présence des déconnexions ou de la mobilité est directe.

Si ni déconnexion ni handoff n'ont lieu, le protocole s'exécute normalement dans la même cellule. Le *coordinateur* doit seulement transférer la décision au *client mobile* et attendre l'acquittement. Dans le cas de déconnexion ou de panne soudaine, nous supposons que l'UM va recouvrir en un délai fini. Durant cette déconnexion, le coordinateur/participant-agent garde les résultats pour le compte de l'UM jusqu'à sa reconnexion, puis, il les lui fait suivre et attend l'acquittement. La déconnexion de l'UM n'affecte en rien l'exécution du protocole. Dans le cas du handoff sans déconnexion, aussitôt que l'UM se reconnecte et s'enregistre à la nouvelle SB, elle contacte le *coordinateur/participant-agent* pour l'informer de sa nouvelle localisation. L'UM est donc libre de se déplacer durant l'exécution du protocole. Si l'UM effectue un handoff alors qu'elle est déconnectée. Au moment de sa reconnexion, elle s'enregistre sur une nouvelle SB qui devient alors sa *current-BS* puis contacte le *coordinator/participant-agent*.

Il est important de noter ici que le changement de cellule n'implique pas forcément le changement d'adresse IP. En effet, quand une unité mobile se déplace, elle peut soit (1) exécuter un handoff vers une cellule du même domaine ; mobilité intra-domaine ou micro-mobilité (figure 4.2.a), ou (2) aller dans une cellule appartenant à un nouveau domaine ; mobilité inter-domaine ou macro-mobilité (figure 4.2.b). Le premier type de handoff est de couche liaison (le handoff L2) et le deuxième type est un handoff de couche réseau (handoff L3). Seul le handoff L3 mène à changer l'adresse IP. L'attribution d'une nouvelle adresse IP se fait seulement dans le cas où le mobile se déplace vers une cellule de domaine différent (mobilité inter-domaines). Dans le cas contraire (mobilité intra-domaine), l'adresse IP du

mobile est toujours valide. Dans ce cas l'envoi du message de notification 'I-m-here' de l'UM vers le coordinateur/participant-agent est inutile. Ainsi, le protocole M2PC doit activer son mécanisme de mobilité seulement dans le cas du handoff L3.

Afin de limitter le blocage après l'émission du message de *décision* vers l'UM et en attendant la récéption du message *Ack* en retour, des délais de garde calculés en fonction du délai moyen de transmission d'un message, de la durée moyenne d'une déconnexion, des durées du handoff L2 et du handoff L3 sont utilisés afin de permettre au coordinateur/participant-agent de distinguer entre ces différents événements et de réagir en conséquence, soit par la retransmission immédiate de la décision, soit par sa retransmission après réception du message de localisation ou à la prochaine reconnexion. On peut trouver des détails sur la mise ne œuvre de M2PC dans [Che06].

Le client/serveur mobile ne participe pas directement dans l'exécution du protocole M2PC, ce sont le coordinateur et les agents participants qui exécutent le principe du 2PC sur le réseau fixe. Il faut aussi souligner que les journaux sont sauvegardés dans une mémoire stable grâce à leur transfert des UMs vers les UFs afin de garantir la tolérance aux pannes.

L'avantage d'avoir préservé le principe du 2PC rend possible de travailler facilement sur des infrastructures hybrides avec des unités mobiles ou fixes [BST04] sans se soucier des mécanismes de contrôle de concurrence ou de recouvrement utilisés par les différents systèmes. La seule condition est que toutes les entités participant à la transaction respectent le 2PC même s'ils ne désirent pas implémenter la mobilité. De manière générale, M2PC hérite des avantages et des inconvénients du protocole 2PC.

Dans la section suivante, nous analysons le protocole M2PC et nous évaluons son mécanisme de gestion de mobilité en le comparant à 2PC s'exécutant avec le support de Mobile-IP, le standard de gestion de mobilité de l'IETF (Internet Engineering Task Force) (voir section 4.1.4).

4.1.3 Evaluation du protocole M2PC

Durant la dernière décennie divers modèles et techniques ont été proposés pour le traitement transactionnel mobile. En général, la gestion de la mobilité est supposée être assurée par un mécanisme situé au dessous de la couche application. Beaucoup de propositions et normes ont été conçues pour la gestion de mobilité, néanmoins quelques difficultés ont retardé leur déploiement [KFCW04, LFH05]. Le protocole M2PC a adopté une approche qui a contourné le problème en proposant un déploiement du protocole 2PC sur une infrastructure réseau n'offrant aucun mécanisme de gestion de mobilité [NDD05a, NDD05b, NDD05c, NDD06a]. En effet, les protocoles de validation qui ont pris en charge les contraintes de la mobilité tels que O2PC-MT et TCOT ont considéré le handoff du point de vu de la migration ou non du contrôle (coordinateur) tandis que le protocole M2PC considère le problème du maintien de la communication entre les entités échangeant des messages en prenant en charge le handoff (changement d'adresse IP) au niveau de la couche application [CN05].

Pendant l'exécution de M2PC, le client mobile/participant peut changer de cellule et s'enregistrer à une nouvelle SB. Le coordinnateur/participant-agent de M2PC est lancé dynamiquement sur la *commit-BS* (la première SB recevant la demande de validation) et y reste pendant toute l'exécution du protocole. Le protocole M2PC exige que le coordonnateur/participant-agent soit en permanence au courant de la localisation du client/participant mobile afin de lui envoyer les résultats. Le client/participant contacte le coordonnateur/participant-agent pour l'informer de sa nouvelle adresse. Cette solution offre

une manière de traiter la mobilité au niveau de la couche application et inclut le mécanisme de mobilité dans le protocole lui-même.

Dans cette section, nous considérons d'abord le comportement de M2PC face à plusieurs paramètres intrinsèques à son mode de fonctionnement ou à l'environnement mobile. Nous étudions donc l'impact de la taille du journal sur le processus ACP; il faut se rappeler que pour assurer la durabilité, dans M2PC, les unités mobiles transfèrent leurs journaux vers la SB après avoir terminé leurs branches de transaction. Puis, nous étudions l'impact du délai induit par le processus d'obtention d'une nouvelle adresse IP dans le cas de la macro-mobilité (handoff L3). Nous analysons également l'impact de la charge du réseau sans fil en termes de nombre de participants fixes et mobiles. Puis, nous comparons le protocole 2PC standard s'exécutant au desssu de Mobile-IP et M2PC utilisant son propre mécanisme. Notre objectif est de voir quelle approche est la meilleure en termes de latence. Pour ces évaluations nous avons basé nos expériences sur NS-2 *(Network Simulator)* [NS03] qui représente la norme de fait pour la simulation des réseaux [HBG06].

Dans les simulations, nous avons utilisé la topologie du réseau sans fil proposée dans [Gre03] (Figure 4.2). La figure 4.2.a représente l'architecture d'un réseau mobile à infrastructure comprenant seulement une station base (une cellule). La figure 4.2.b et 4.2.c représente l'architecture comprenant plusieurs stations base (plusieurs cellules). La technologie de transmission du réseau sans fil utilisée est la norme IEEE 802.11 (WiFi).

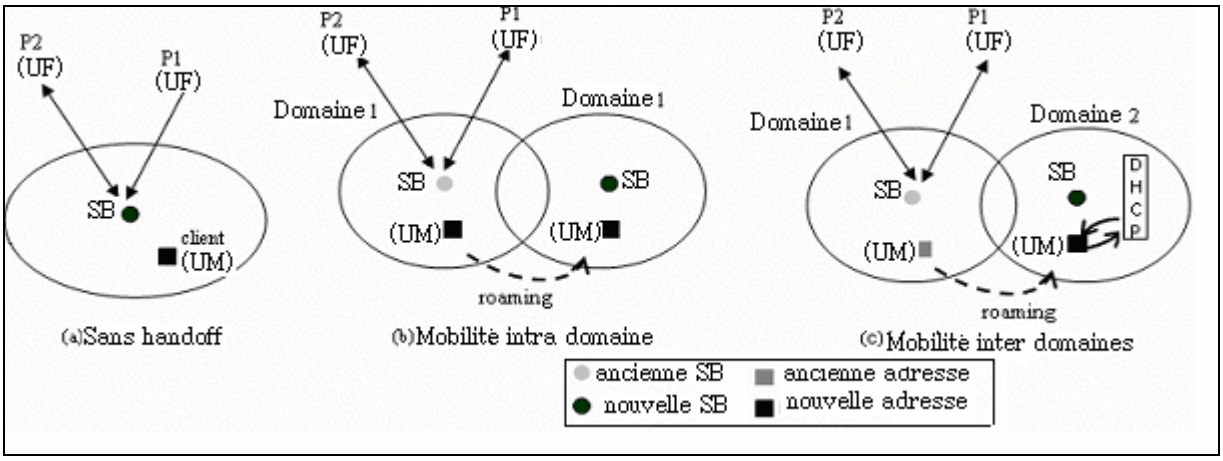


Figure 4.2-Architecture des réseaux sans fil simulés

Taille du journal (fichier log)

Nous avons une base de données mobile qui occupe un Méga d'espace mémoire [Vid01]. Dans une unité mobile, une donnée (un objet ou un item) est supposée occuper une page mémoire de 1KO [DCKM+94]. Une transaction mobile peut modifier entre 2 et 16 articles [KDDS 00]. Donc, la taille d'un fichier log varie entre 2 et 16 KO.

Temps des opérations disques (écritures forcées).

Le temps d'une opération disque d'un nœud mobile est entre 10 ms et 40 ms [CBML03] et celui d'un nœud fixe est entre 8 ms et 16 ms [NSB97].

La latence du handoff de niveau 2

La durée du handoff L2 varie entre une dizaine et environs 200 ms [KKLY+ 04] selon la technologie sans fil adoptée. La latence du handoff dans les réseaux WiFi est dans les environs de 158 ms [MN02].

Temps d'obtention d'une nouvelle adresse

Le temps d'obtention d'une adresse IP à partir d'un serveur DHCP *(Dynamic Host Configuration Protocol)* est soit pris aléatoirement entre 800 ms et 1670 ms [Kum03, KKLY+04] soit fixé à 1670 selon les besoins des simulations.

La table 4.1 suivante indique les valeurs des paramètres liés au réseau.

Table 4.1-Les paramètres du réseau de communication sans fil

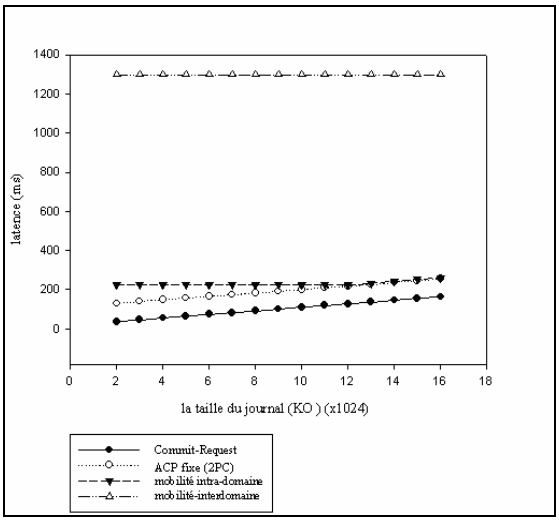
paramètre	**valeur**
Capacité des liens filaires	5Mb - 10 Mb
Type de liens filaires	Duplex
Topographie de la zone terrestre simulée	670 x 670 m^2
Vitesse de déplacement	10 – 140 m / s

4.1.3.1 Impact de la taille du journal

Pour une mailleure tolérence aux pannes, nous avons vu que les journaux sont transférés vers la partie fixe du réseau. Or, or ce transfert se fait sur les liens sans fil, ce qui peut à priori affecter la lattence du protocole. Afin d'étudier l'effet de ce paramètre (taille du fichier log) sur la latence du protocole M2PC, nous avons fixé le paramètre vitesse à 50 m/s, la position lors de l'initiation du processus de validation à 248 m et le rayon de la cellule à 250 m. Nous faisons varier le paramètre taille du fichier log entre 2KO et 16 KO.

Sur la figure 4.3, il y a une différence importante entre les latences de M2PC pour les deux types de mobilité. Bien que le délai d'exécution du processus de validation sur le réseau fixe augmente en fonction de la taille du journal, la latence de M2PC est stable dans le cas de la mobilité inter-domaines. Ceci signifie que la taille du journal influence le délai du message *commit-request*, mais n'a pas un impact significatif sur la latence globale du protocole. On peut expliquer ceci, par le fait que la latence de la partie fixe de l'ACP est beaucoup plus petite (maximum 225 ms) comparée au délai d'obtention d'une adresse IP même lorsque la taille atteint 16 KO. Si un handoff L3 se produit pendant l'exécution du protocole sur la partie fixe du réseau, le message de décision doit attendre au niveau du coordinateur jusqu'à ce que le message de localisation est envoyé après la fin du handoff L2 et l'obtension d'une nouvelle adresse IP. Ceci prend plus que 1200 ms, ce qui est plus grand que la latence de l'ACP pour la taille maximum essayée. Dans le cas de la mobilité intra-domaine, la latence augmente légèrement à partir d'une taille de journal de 12 KO. Au-dessous de cette taille, l'impact n'est visible que sur la partie fixe de la latence du protocole. En fait, lorsque la taille n'est pas très importante, le handoff L2 est plus long que le processus de validation. A partir de 12 ko, le processus de validation englobe le délai du handoff L2.

En résumé, une taille importante du journal peut allonger la latence de M2PC dans le cas d'une mobilité intra-domaine. En cas de mobilité inter-domaine la taille du journal n'a pas d'impact sur la latence de M2PC.



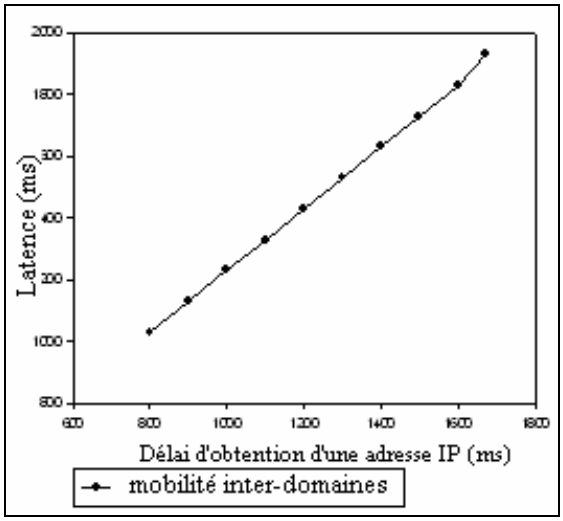


Figure 4.3-Impact de la taille du journal **Figure 4.4-**Impact de l'acquisition de l'adresse IP

4.1.3.2 Impact du délai d'acquisition d'une adresse IP

Dans cette expérience, la taille du journal a été fixée à 9 KO et le délai d'acquisition d'une nouvelle adresse IP a été varié entre 800 et 1670 ms. La vitesse du mobile est fixée à 50 m/s, la position intiale à 248 m et la portée à 250 m.

La figure 4.4 prouve que la latence est proportionnelle au délai d'acquisition d'une adresse IP; ceci est tout à fait prévisible. L'identification d'un réseau étranger et ensuite l'acquisition d'une adresse de DHCP constituent une composante importante du délai de handoff. Afin d'améliorer la qualité de service, des travaux de recherche ont proposé d'anticiper ce processus en commençant l'étape d'acquisition d'adresse, d'une manière proactive, en parallèle avec le handoff L2 [KKL+04, YPT+01, CNZ05]. Cette technique peut permettre une réduction considérable de la latence du handoff L3.

4.1.3.3 Impact du nombre de participants fixes

Dans ce scénario, nous avons fixés les paramètres réseau comme pour le scénario précédent et nous avons augmenté au fur et à mesure le nombre de participants fixes en commençant par deux. A chaque fois nous ajoutons un participant fixe dont le temps des opérations disque est pris aléatoirement entre 8ms et 10ms, la capacité de son lien filaire est soit 5Mb/s soit 10Mb/s, le délai de transmission des messages sur les liens filaires varie entre 5ms et 10ms.

Sur la figure 4.5, nous pouvons voir que le nombre de participants fixes n'est pas significatif dans les deux types de mobilité. C'est dû au fait que la prise de décision de valider ou non une transaction est réalisée en parallèle et de manière coopérative sur le réseau fixe et dépend principalement du participant le plus faible en termes de capacités de calcul, largeur de bande et distance du coordonnateur. Ces paramètres ont une influence négligeable sur la partie fixe du réseau.

4.1.3.4 Impact du nombre des participants mobiles

Des configurations sans handoff (figure 4.2.a), avec le handoff L2 ou le handoff L3 (figures 4.2.b et 4.2.c) ont été considérées. Nous avons 2 participants fixes et nous ajoutons progressivement les participants mobiles. Chacun des nœuds mobiles ajoutés exécute un handoff avec une probabilité p_i:

$$\begin{cases} P_i = 1 & \text{si le handoff se produit} \\ P_i = 0 & \text{sinon.} \end{cases}$$

La probabilité qu'un handoff soit de type L2 est P_{iL2} ou de type L3 est $P_{iL3} = 1- P_{iL2}$ (voir table 4.2).

Table 4.2-Probabilités de Handoff

participant	1	2	3	4	5	6	7	8	9	10
P_i	0	0	1	1	1	0	1	1	0	1
P_{iL2}	-	-	0	1	0	-	1	0	-	0
P_{iL3}	-	-	1	0	1	-	0	1	-	1

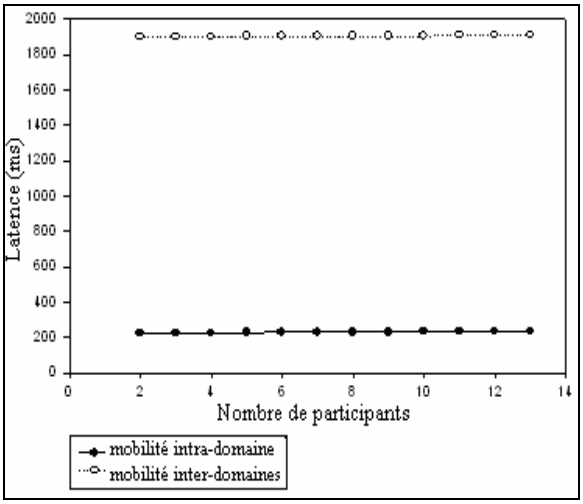


Figure 4.5-Impact des participants fixes

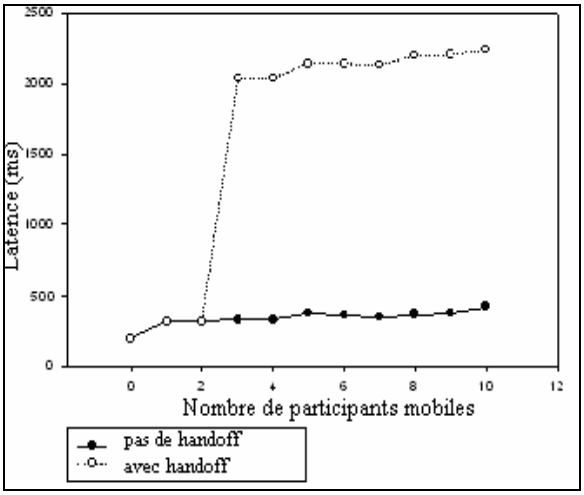


Figure 4.6-Impact des participants mobiles

Les résultats présentés dans la figure 4.6 peuvent être résumés comme suit:

- Dans la configuration sans handoff, en augmentant le nombre de participants mobiles (dans des cellules différentes), la latence augmente d'une façon plus ou moins significative entre de 224 ms (sans aucun mobile) et 425ms. Cette augmentation peut s'expliquer par le fait que les participants mobiles doivent envoyer leurs journaux en même temps que leurs votes.

- Dans la deuxième configuration, on observe une latence importante quand le troisième participant est introduit; c'est dû au handoff L3 qu'il effectue. La latence augmente légèrement avec l'augmentation du nombre de noeuds mobiles; cependant, la réalisation de plusieurs handoffs n'a pas une grande influence sur la latence du protocole même s'ils sont du niveau L3. En fait, les handoffs s'exécutent en parallèle pendant la période de prise de décision et donc la latence de M2PC dépend du dernier handoff L3 qui a lieu.

4.1.3.5 Impact de la charge du réseau sans fil

Dans cette expérience nous étudions l'impact du nombre de noeuds mobiles quand ils sont tous mobiles dans la même cellule. Nous augmentons le nombre d'unités mobiles actives dans la cellule *(Commit-BS)*. La figure 4.7 récapitule les résultats qui indiquent que quand les noeuds se déplacent dans la même cellule, la latence varie dans l'intervalle (183ms, 184ms) avec 2 noeuds dans la cellule. Elle augmente jusqu'à 374ms pour 6 noeuds mobiles bien qu'aucun handoff ne se produise. L'augmentation des risques de collision est responsable de cette latence. Lorsqu'il s'agit des handoffs L2 ou L3 l'effet sur la latence est très significatif (2124 ms à 12479 ms dans le cas de mobilité intra-domaine et 3231ms à 13649 ms dans le cas inter-domaines). Le surcoût est dû au fait que la même SB et le même canal servent à la fois au protocole et au contrôle du trafic. La partie contrôle augmente d'une manière importante quand les handoffs sont plus nombreux.

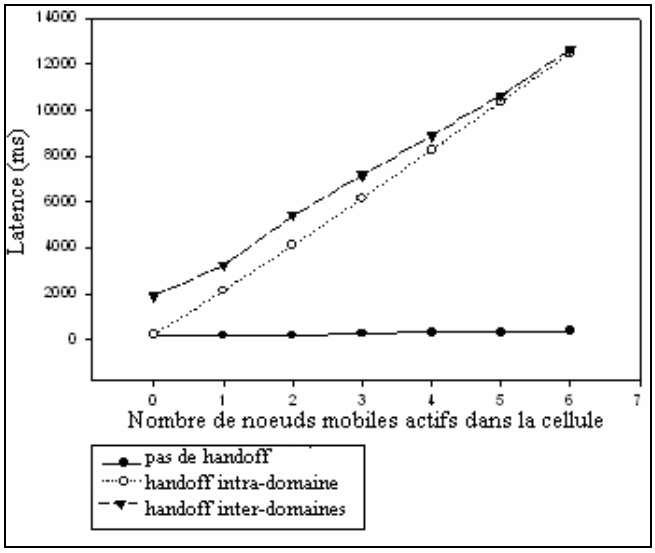


Figure 4.7-Impact de la charge du réseau sans fil

4.1.3.6 Résumé de l'étude du protocole M2PC

En résumé (table 4.3), le délai d'acquittions d'une adresse IP et la congestion du réseau sans fil ont un grand impact sur la latence de M2PC. La congestion est exprimée dans nos expériences en termes de nombre de noeuds actifs dans la même cellule. Les participants mobiles l'affectent quand ils sont dans des cellules différentes, mais leur influence est significative quand ils sont en activité dans la même cellule (figure 4.7). Le nombre de participants fixes n'affecte pas la latence. Contrairement à nos attentes, la taille du journal n'a d'impact significatif que dans le cas de la mobilité intra-domaine et seulement pour des valeurs élevées. La latence du handoff L3 et par conséquent du protocole M2PC peut être améliorée par une technique anticipant l'acquisition d'une nouvelle adresse IP, en lancant ce processus en parallèle avec le handoff L2.

Table 4.3-Performances du protocole M2PC

Paramètre	Taille du journal	Délai d'obtension d'adresse IP	Participants fixes	Participants mobiles
Degré d'influence sur la latence	En intra-domaine pour une taille relativement élevée	Proportionel	Non	Oui Important en intra-cellule

4.1.4 Evaluation du mécanisme de M2PC pour le support de la mobilité

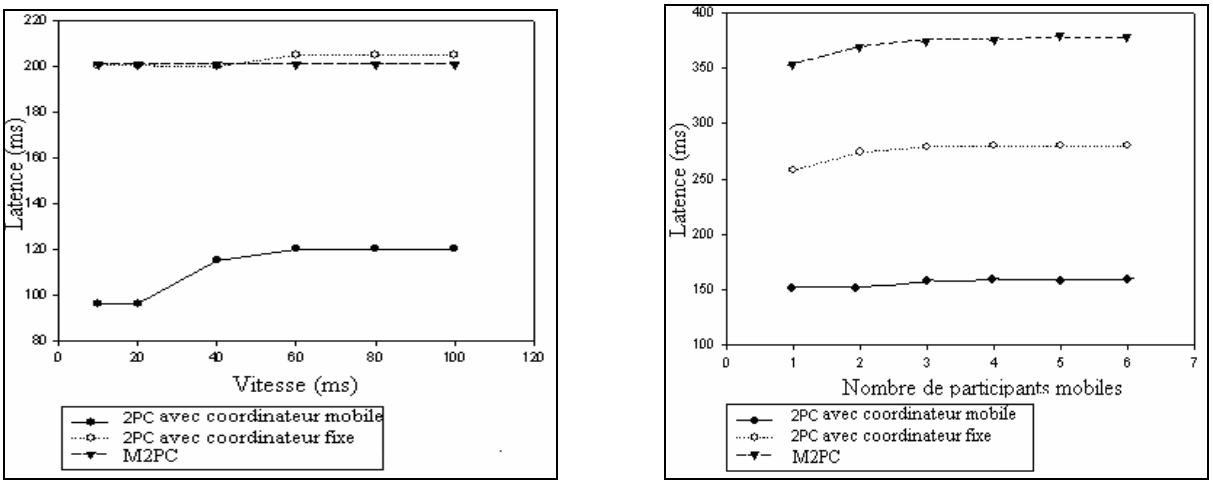
Dans cette section, nous comparons le protocole 2PC standard s'exécutant avec le support de Mobile-IP[18] et le protocole M2PC s'exécutant avec le support de son propre mécanisme de mobilité. En effet, M2PC intégre sa propore gestion de la mobilité, alors que les autres protocoles comptent sur le support de techniques implémentées au niveau des couches protocolaires sous-jacentes. Mobile-IP est l'une des techniques qui permet de rendre joignable à tout instant une unité mobile. Il s'agit d'un standard de l'IETF[19] qui permet de masquer la mobilité d'un équipement à ses correspondants sur le reste de l'Internet. Dans le cas du 2PC, par exemple, il permet de masquer la mobilité du client au coordinateur.

[18] http://www.ietf.org/html.charters/mobileip-charter.html

[19] http://www.ietf.org

Dans cette expérience, nous employons une topologie de réseau sans fil comprenant deux participants fixes, deux cellules appartenant à deux domaines différents et un client mobile. Nous définissons trois scénarios de mobilité. Dans le premier, nous exécutons le protocole M2PC. Dans le second nous exécutons 2PC avec le coordinateur situé sur le client mobile et dans le troisième scénario le coordinateur est situé au niveau de la SB. Nous faisons varier dans chaque scénario la vitesse du mouvement du client mobile puis le nombre de nœuds mobiles. Les résultats de la simulation concernent la latence et sont présentés dans la figure 4.8:

- La vitesse n'a pas d'impact visible sur la latence de M2PC et a peu d'impact sur la latence de 2PC avec un coordinateur fixe. Cependant, elle augmente la latence de 2PC quand le coordinateur est mobile. La latence augmente quand il y a handoff, à cause du routage triangulaire[20] utilisé par le Mobile-IP pour communiquer avec les participants. Plus la vitesse est élevée, plus la probabilité des handoffs augmente. M2PC préserve une latence stable en présence de la mobilité du client. Cela confirme que la meilleure localisation du coordinateur est sur la partie fixe du réseau.



(a) Effet de la vitesse (b) Effet du nombre de noeuds mobiles

Figure 4.8-Comparaison des protocoles 2PC et M2PC

- L'utilisation du mécanisme de mobilité de M2PC est légèrement mieux que l'utilisation de Mobile-IP dans le cas d'un coordinateur fixe qui est basé sur le routage triangulaire. Cependant, 2PC avec un coordinateur mobile donne de meilleures performances parce que dans le cas d'un coordinateur fixe (pour M2PC et 2PC): les protocoles ajoutent une étape de communication pour informer le client mobile au sujet de l'accomplissement de la transaction (décision envoyée vers le mobile et attente de l'accusé de réception). Cette étape n'est pas pénalisante car elle ne bloque pas les autres participants.

- Quand le nombre de participants mobiles augmente (figure 4.8.b) M2PC est beaucoup moins performant que 2PC à coordinateur fixe. Ceci signifie que la gestion de la mobilité des nœuds par M2PC est plus pénalisante car elle retarde l'arrivée des journaux et des votes et retarde par conséquent le processus décisionnel du coordinateur.

[20] Il s'agit du routage employé par Mobile-IP dont le principe permet à un noeud mobile d'envoyer directement ses paquets vers son correspondant, mais le correspondant doit forcément envoyer la réponse à un agent, se trouvant dans le réseau d'origine, qui fait suivre les paquets jusqu'au nœud mobile.

En résumé:

- le mécanisme de M2PC pour la mobilité est réalisable et pourrait être plus intéressant que le routage triangulaire de Mobile-IP en particulier en termes de déploiement.

- le routage triangulaire peut, dans certains scénarios, être plus pénalisant que le transfert des journaux ou le mécanisme de mobilité de M2PC (figure 4.8.a). Cependant, tout échange de messages entre une unité mobile et une unité fixe a un impact qui est plutôt lié à la qualité des communications sans fil (figure 4.8.b). En fait, nous avons montré à partir d'une évaluation analytique, basée sur la modélisation avec les réseaux de files d'attente, que le délai de handoff induit par le mécanisme de M2PC croît avec la croissance du taux d'erreurs du réseau sans fil. Ceci est dû au recouvrement d'erreur par TCP. De plus, le délai de handoff croît exponentiellement avec l'accroissement du taux d'arrivée des messages, c'est.-à-dire; quand le taux de traitement au niveau des différents composants approche le taux d'arrivée des messages (qui se traduit par la congestion). Cependant, la composante du délai de handoff dû au traitement des messages est négligeable comparée à celle induite par la transmission sans fil des messages. Autrement dit, c'est le délai de transmission sans fil qui constitue la plus grosse contribution au délai total de handoff (plus de 90%) [NDD05]. Ceci indique que le facteur principal influant sur la latence est induit par la non fiabilité des communications sans fil lors du transfert du message de localisation (des autres messages aussi) et surtout lorsque le nombre de nœuds mobiles augmente comme c'est le cas dans la figre 4.8.b.

- le 2PC standard (avec un coordonnateur mobile) produit un grand nombre de messages sans fil et ceci peut être désavantageux en termes de coût de communication et de consommation d'énergie.

4.2 Un protocole décentralisé pour la validation dans les réseaux ad hoc

Dans les environnements ad hoc une unité de calcul n'a aucune possibilité de support par des serveurs fixes; tous les noeuds sont des pairs. De plus, ces noeuds peuvent s'organiser spontanément en topologies arbitraires et fréquemment changeantes [YCG+03, CCL03]. Dans ces configurations, rien ne garantie que tous les nœuds qui souhaitent collaborer dans l'exécution d'un protocole de validation puissent être disponibles au même moment ou que les nœuds qui se déconnectent puissent se rencontrer à nouveau [BHPS05, PJYF03]. Par conséquent, ces noeuds ne peuvent compter que sur leur support les uns les autres et des pairs se trouvant dans leur environnement pour conclure une transaction. Un protocole tolérant aux défaillances offrant efficacement l'atomicité transactionnelle dans ces environnements instables doit pallier aux nouveaux défis.

Le protocole que nous proposons est basé sur l'adaptation de la variante distribuée du protocole 2PC à savoir le **Distributed Two-phase Commit *(D2PC)*** [Gra78]. L'idée principale derrière cette adaptation est de renforcer le mécanisme de recouvrement pour pallier aux déconnexions fréquentes et l'instabilité de l'environnement. Deux facteurs significatifs doivent être pris en considération. D'abord, il n'y a aucune possibilité d'avoir un support fixe pour lequel une UM peut déléguer des tâches ou la journalisation des données de recouvrement. Ensuite, étant donnée la nature très distribuée de l'environnement, il est nécessaire d'éviter un point de contrôle centralisé et d'avoir une coordination.

La réplication est souvent la solution adoptée pour la disponibilité des données dans les systèmes répartis et plus récemment dans les environnements mobiles. La réplication d'informations de recouvrement sur une station fixe est très courante dans les transactions mobiles. Dans l'environnement ad hoc considéré, nous avons également pensé à la réplication.

L'idée est de faire en sorte qu'une UM donnée puisse compter sur la réplication de ses données sur ses paires en employant la mémoire de ces derniers pour sauvegarder les informations nécessaires au bon déroulement et terminaison cohérente de la transaction à laquelle elle participe. En d'autres termes, les journaux employés habituellement par les systèmes de transaction doivent être disponibles malgré les déconnexions et les changements fréquents de configuration du réseau.

4.2.1 Le protocole D2PC

Le 2PC décentralisé (D2PC) préserve la même structure que le 2PC traditionnel, c'est-à-dire validation à deux phases, mais il réduit le nombre d'étapes de communication. Dans le 2PC qualifié de centralisé, le coordonnateur diffuse la demande de vote à tous les participants et centralise les réponses avant de prendre sa décision. Dans le 2PC décentralisé, le coordonnateur diffuse la demande de votes à tous les participants et chaque participant diffuse son vote tous les autres participants. En l'absence de défaillances, chaque participant peut prendre ainsi une décision d'une manière décentralisée. Ainsi, si un participant reçoit des votes positifs de tous les autres participants, il valide la transaction. Mais s'il reçoit au moins un vote négatif, il abandonne la transaction. Une étape de communication est ainsi gagnée par rapport au 2PC centralisé. Cependant, le nombre de messages transmis pendant la phase de diffusion des votes devient quadratique.

La technique de journalisation est identique à celle du 2PC. Chaque participant sauvegarde, par une écriture forcée dans un journal, le changement d'état du protocole durant son exécution. Un tel journal est utilisé quand un participant doit procéder au recouvrement pour terminer de manière consistante la transaction. Par exemple, un enregistrement *prepare* (avec les mises à jour) est écrit quand un participant envoie un message de vote *Yes*. Un participant écrit un enregistrement commit/abort après avoir reçu les votes requis. Une transaction peut être abandonnée durant l'exécution de ses opérations ou durant sa validation. La première situation peut se présenter quand l'exécution d'une opération échoue (pour un problème de contrôle de concurrence ou le crash d'un site). La dernière situation peut apparaître, durant l'exécution du protocole, à cause d'une panne de site, un timeout ou un vote *No*.

4.2.2 Le protocole D2PC pour un environnement ad hoc (A-D2PC)

Le protocole D2PC n'a pas d'intérêt pratique dans les systèmes distribués traditionels car malgré la réduction du nombre de phases de communication, il augmente le nombre de messages de vote. Vu la proprité de diffusion de l'environnement ad hoc, nous avons pensé à reconsidérer le protocole D2PC afin d'exploiter cette propriété dans l'échange d'informations entre les participants. De plus, l'aptitude des UMs à prendre la decision d'une manière décentralisée est une approche attractive dans cet environnement où il n'est pas du tout inéressant d'avoir un seul point de contrôle. En, effet, si l'une des UM est le coordinateur, il n'est pas evident qu'elle soit joignable durant la phase de validation ou de recouvrement éventuel.

Le recouvrement est habituellement fait en analysant les données du journal local qui sont stockées en mémoire stable et en communiquant avec le coordonnateur et d'autres participants pour se renseigner sur leurs états. Mais dans le modèle de système considéré, l'instabilité des noeuds est importante et de ce fait interroger le coordonnateur, les participants ou d'autres noeuds sur des événements après une déconnexion n'est pas un bon choix. En effet, tous ces noeuds ont pu probablement avoir déjà quitté le MANET. Ainsi l'idée principale de l'approche proposée n'est pas de compter sur une seule UM pour assurer la disponibilité des informations

du journal mais plutôt de faire une certaine réplication sur plusieurs UMs du MANET [NDD06c].

Dans notre proposition, nous préservons le principe de fonctionnement du protocole mais nous modifions le processus de journalisation. Chaque UM participant à l'exécution de la transaction y compris l'initiateur produira son journal localement. Chaque UM diffuse son journal (log) à ses voisins immédiats. A chaque insertion d'une action de validation (demande de vote, vote, décision), les voisins reçoivent un message contenant la mise à jour faite sur le journal. Périodiquement les copies des journaux sont synchronisées entre elles. En fait, la diffusion des données du journal aux voisins fait que chaque UM a une vue (éventuellement la plus récente) de l'état de la transaction même si un des participants est déconnecté pendant l'exécution du protocole ; ceci peut se produire très fréquemment. Les écritures forcées et la dissémination des informations augmentent le taux de validation des transactions. Sans cette précaution, un participant pourrait décider à tort d'annuler une transaction parcequ'il n'aurait pas reçu tous les votes à cause des déconnexions bien que tous les participants aient tous voté positivement. Grâce à la diffusion des journaux, il y a plus de chances que la validation soit décidée localement lorsque soit les votes demandées sont reçues, soit la décision est trouvée dans le journal. La figure 4.9 présente l'exécution du protocole avec un seul participant.

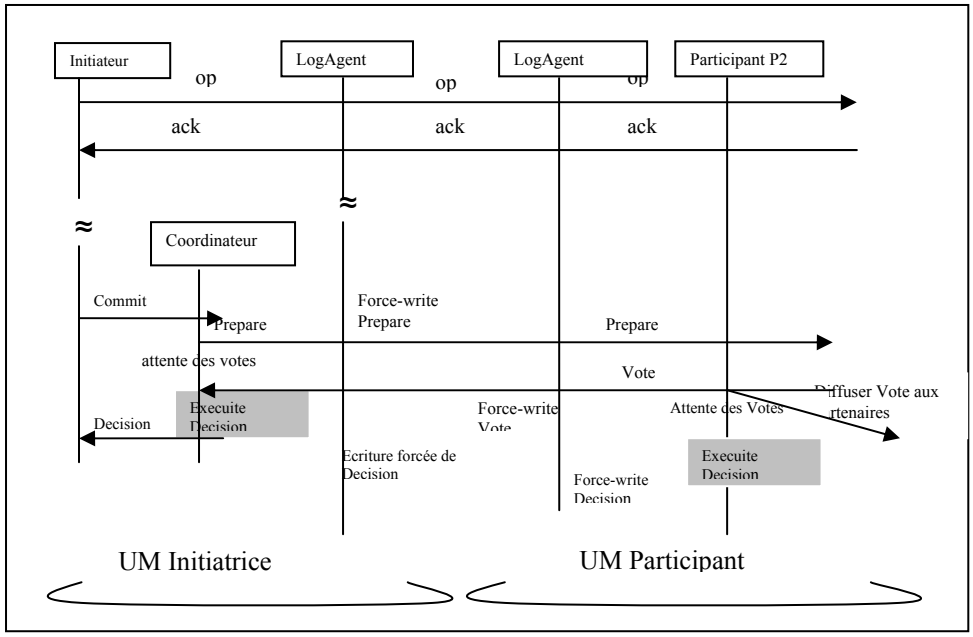


Figure 4.9-Le protocole A-D2PC pour l'environnement ad hoc

4.2.3 La dissémination des informations du journal

Au niveau de chaque UM, un composant logiciel appelé LogAgent est responsable de mettre à jour le journal local. L'identificateur d'une entrée du journal fait référence à une transaction ; son ID contient l'identificateur de transaction. Ce dernier doit être unique ainsi il se compose de l'identité du noeud initiateur et d'une estampille locale. Les entrées du journal correspondant à la même transaction ont le même identificateur de transaction.

Les mises à jour doivent être disséminées afin de permettre à chaque nœud participant d'avoir accès aux informations de recouvrement au besoin. Pour cela, un nœud doit exécuter les actions suivantes:

- quand il se connecte pour la première fois ou après une reconnexion, il doit découvrir des voisins dans sa zone de couverture et ensuite synchroniser sa copie du journal avec celles de ses voisins.

- chaque élément de donnée écrit dans le journal est disséminé vers les autres noeuds,
- un LogAgent synchronise périodiquement sa copie avec les autres copies de son environnement.

La technique de dissémination est supposée être conforme au protocole adaptatif de type pull *(Adaptive Pull Protocol)* de [BBHS05] pour lequel les résultats de simulation prouvent qu'il améliore la diffusion des données tout en limitant la charge du réseau et en réalisant le même taux de fraîcheur que d'autres protocoles plus coûteux. La figure 4.9 montre à quel moment les écritures forcées ont lieu avec leur diffusion immédiate vers les voisins situés à une distance d'un saut. Les noeuds recevant le message diffusé mettent à jour leur copie et en tiennent compte dans les synchronisations suivantes.

La synchronisation est initialisée quand un noeud joint le MANET pour la première fois ou quand il se reconnecte. Le noeud qui initie la synchronisation diffuse les IDs des journaux qu'il possède et réclame ceux de ses voisins. Ces derniers comparent la liste reçue à leur propre liste et déduisent la différence qu'ils envoient au nœud demandeur par la diffusion. Ce processus synchronise non seulement le noeud demandeur mais diffuse également l'information à tous les nœuds de la zone. Les noeuds impliqués ainsi que leurs voisins reçoivent l'information [BBHS05].

Chaque noeud doit maintenir une liste de ses voisins. Un simple composant logiciel de découverte basé sur la diffusion d'un message de présence peut être employé [BHPS05]. Chaque noeud diffuse ce court message pour informer les voisins immédiats au sujet de sa présence. Les noeuds extraient de ce message l'identité et l'adresse pour maintenir la liste de présence. Nous proposons d'inclure dans le message de présence la réserve d'énergie disponible au niveau du noeud. Cette information peut servir pour ne choisir pour participer dans le processus de diffusion que les voisins ayant la réserve d'énergie la plus élevée.

4.2.4 Traitement des déconnexions

Un participant peut perdre sa connexion avec l'un de ses partenaires ou peut complètement se déconnecter du MANET pendant l'exécution du protocole. Notre proposition permet de terminer la transaction de manière cohérente. Quand l'UM se reconnecte, elle doit se renseigner sur l'état de la transaction interrompue. Ceci est normalement réalisé grâce à l'utilisation des informations du journal.

- Dans le cas où le coordonnateur (ou l'initiateur) tombe en panne ou se déconnecte après avoir émis la demande de préparation et avant de recevoir tous les votes prévus: quand il se reconnecte, il doit se synchroniser avec ses voisins pour découvrir si des mises à jour ont eut lieu sur le journal. Si aucune décision n'a été enregistrée et que des votes manquent alors il rediffuse sa demande de préparation et attend un timeout avant de décider et de mettre à jour le journal.

- Si un participant reçoit une deuxième demande de préparation pour laquelle il a déjà répondu, il l'ignore, sinon le protocole est poursuivi normalement.

- Dans le cas où un participant se déconnecte après réception de la demande de préparation et avant d'envoyer son vote: quand il se reconnecte, il doit se synchroniser avec ses voisins pour découvrir si des votes ont été écrits dans le journal pendant la déconnexion. Il peut trouver une décision d'abandon ou au maximum n-1 votes de validation (n étant le nombre de participants), il diffuse son vote et met à jour son journal.

-Dans le cas où un participant se déconnecte après la réception de la demande de préparation et l'envoi de son vote mais avant de recevoir tous les votes: quand il se reconnecte, il doit se synchroniser avec ses voisins pour découvrir si plus de votes ont existent dans le journal. Il

peut trouver soit une décision ou les votes attendus. Si aucune décision n'est trouvée et des votes sont toujours manquant, alors il rediffuse son vote, met à jour le journal et attend pour un délai de garde avant de décider et terminer la transaction.

- Si un participant reçoit le vote du même participant deux fois, il ignore simplement le message.

Grâce à la diffusion des données du journal, une UM peut terminer une transaction même si certains ou tous ses partenaires ne sont pas accessibles. Cette technique permet au LogAgent de suivre l'évolution du protocole aux différents noeuds. L'étape de décision du protocole est distribuée entre les UMs, ceci leur permet de terminer de façon autonome la transaction, par conséquent réduire la fenêtre de vulnérabilité du protocole et la probabilité de son blocage; ce qui est très important dans un environnement instable.

4.2.5 Performance de A-D2PC

Le coût en termes de messages inclut n messages "prepare" et n*n messages de vote. La diffusion des journaux s'ajoute au coût à chaque fois qu'une mise à jour se produit dans les étapes de protocole. En cas de panne ou de déconnexion les participants doivent se synchroniser les uns avec les autres. Tous les noeuds participant à la dissémination même s'ils ne participent pas dans la transaction ajoutent un coût. Le coût relatif à la dissémination dépend de la technique utilisée et n'est pas exactement le même d'une technique à l'autre [BBHS05]. Cependant, ce coût est le prix à payer pour améliorer le throughput du système.

4.2.6 Travaux similaires

L'utilisation des protocoles épidémiques ou des techniques semblables de dissémination est une idée partagée par la proposition [HAE00] qui s'intéresse aux déconnexions prévisibles. La même idée est utilisée dans le Shared Log Space (SLS) [BHPS05] ainsi que la technique Neighborhood gestion de transaction [PJYF03] dans les réseaux ad hoc. Ces mécanismes visent à atténuer la nature dynamique du réseau, en particulier les déconnexions. L'idée du SLS a été utilisée pour établir un espace virtuel de partage d'informations même en présence des déconnexions. Le SLS sert les mêmes objectifs que notre proposition qui consiste à disposer des copies des logs sur plusieurs noeuds du MANET.

La technique du SLS est basée sur une architecture middleware qui a comme ambition un large champ d'application potentiel dans les environnements ad hoc et ne se limite pas au problème des transactions. En effet, elle offre des moyens d'avoir une vue plus ou moins précise de l'état d'un système très dynamique où un simple échange de messages ne suffit pas à la réalisation d'une tâche commune. Cependant, elle a été employée dans un contexte plutôt précis et limité à un couple de noeuds partenaires d'une transaction de e-commerce même si [BHPS05] n'exclut pas la perspective d'une extension à une implémentation plus sophistiquée.

La solution que nous proposons est plus simple et peut être appliquée au 2PC traditionnel ou n'importe quelle autre alternative ou optimisation. La différence se situera principalement en coût qui sera induit par le protocole. Par exemple, le 2PC traditionnel induirait un coût additionnel en termes de cycles de dissémination[21]. La technique Neighbourhood diffère de notre solution par le fait qu'elle est limitée pour assurer la cohérence transactionnelle de proximité et pas globale. La solution de [HAE00] traite seulement les déconnexions prévisibles.

[21] Le nombre d'écritures sur le journal a une incidence directe sur la fréquence des cycles de dissémination.

4.3 Conclusion

Dans ce chapitre nous avons décrit deux propositions de protocoles. Le protocole M2PC est destiné à un environnement mobile avec infrastructure et présente la particularité d'être exécutable sur un réseau n'ayant pas de support de mobilité dans les couches basses de la pile protocolaire TCP/IP. Une évaluation du protocole M2PC et plus spécifiquement de sa gestion de mobilité a été présentée. En résumé, le mécanisme de M2PC pour la mobilité est réalisable et pourrait être plus intéressant que le routage triangulaire de Mobile-IP en particulier en termes de déploiement mais pas forcément en en termes de latence. Cependant, afin de pallier aux déconnexions et réduire le coût en communication et consommation d'énergie, le 2PC standard doit être adapté par le transfert du traitement sur la partie fixe du réseau.

L'idée principale du protocole A-D2PC pour l'environnement ad hoc est de disposer d'informations de validation et de terminaison de transactions moyennant un protocole de dissémination. Ceci permet de terminer une transaction de façon cohérente même si les participants ne sont pas tous simultanément connectés durant toutes les étapes du protocole. L'évaluation de A-D2PC est complexe car elle fait appel aux techniques de dissimination dans les réseaux ad hoc qui constituent à elles seuls une problématique importante et nécessaite une investigation plus approfondie qui est en dehors du cadre de cette thèse.

Le chapitre 5 est consacré à l'étude comparative de protocoles de validations mobiles dont le protocole M2PC et certains protocoles présentés dans le chapitre précédent.

Chapitre V

Etude comparative des protocoles de validation de transactions mobiles

Measure what is measurable, and make measurable what is not so.
--Galileo Galilei

En réalisant notre état de l'art, nous avons constaté qu'il y avait une demande réelle de la recherche en termes d'évaluation des solutions proposées dans le domaine des techniques transactionnelles dont les protocoles de validation. En effet, ces derniers sont souvent évalués sur la base de l'estimation des performances dans le meilleur et le pire cas. Malgré leur intérêt à avoir rapidement une idée sur les performances des protocoles, ceci n'est pas toujours suffisant pour déterminer la meilleure approche ACP pour un système particulier avec des valeurs de paramètres spécifiques. Dans ce chapitre, nous présentons une étude comparative de quelques protocoles de validation de transactions mobiles en nous basant sur la simulation. En effet, en ce qui concerne les environnements mobiles, la simulation est favorisée par rapport à l'expérimentation dans un environnement réel car cette dernière est plus couteûse et moins flexible.

5.1 Modèle de simulation

Inspirés par l'étude des ACPs traditionnels et les techniques de transactions mobiles [CSA98, LAE98, CL99, KU99], nous avons divisé le système en trois sous modèles: modèle de base de données, modèle de transactions et modèle de réseau mobile [BN05]. Les modèles de base de données sont souvent semblables au modèle présenté dans [ACL87]. Une base de données est une collection d'objets (données élémentaires) qui sont distribués à travers un certain nombre d'emplacements (avec ou sans réplication de données). Un objet de données est uniquement par la relation (site, objet). Les nombres de sites et d'objets sont spécifiés comme paramètres au système. Nous tenons compte de la propriété d'hétérogénéité des systèmes mobiles qui différencie les proportions de distribution de données entre les noeuds. Ainsi, les tailles des parties de la base de données sur les noeuds fixes sont plus grandes que celles sur les noeuds mobiles. Une transaction (longue, courte) est modélisée comme une séquence d'opérations de lecture ou d'écritures terminées par une opération de validation ou d'annulation. Comme il s'agit de transactions mobiles, quelques parties de leurs calculs sont exécutées sur des noeuds mobiles et quelques parties sur des noeuds fixes. Le modèle de communication diffère du réseau filaire traditionnel. En effet, les noeuds fixes sont

interconnectés par un réseau de communication filaire à large bande fortement disponible tandis que les noeuds mobiles sont connectés au réseau fixe[22] par liaisons sans fil de faible largeur de bande passante.

NS-2 *(Network Simulator)* [NS03], OPNET [Opn04] et GloMoSim *(Global Mobile Information Systems Simulation Library)* [Glo01] sont les plateformes de simulation pour les réseaux mobiles les plus avancées et les plus référencées par la communauté des chercheurs [HBG06]. NS et GLoMoSim sont du domaine public. GloMoSim permet la simulation ad hoc de réseaux; il ne supporte pas les réseaux sans fil à infrastructure tels que celui représenté sur la figure 1.1 et sur laquelle sont basés les protocoles simulés. NS-2 est la norme de fait pour la simulation des réseaux [HBG06]. Il offre l'avantage de supporter les réseaux à infrastructure et les réseaux ad hoc (figure 1.2). Cependant, la simulation des systèmes de transaction sur NS n'est pas commode parce que ce dernier ne fournit pas des outils de base de données ou de traitement transactionnel. De part sa complexité l'ajout de composants/protocoles ou la modification de ceux qui existent ne sont pas évidents. Or, notre recherche nous a menés à DBsim (simulateur de base de données) qui est une plateforme de simulation pour des transactions distribuées [NSB97, SN95][23]. Cette plateforme, pour laquelle nous avons obtenu le code source, a semblé être un candidat intéressant à intégrer au package NS [NBD07].

Le simulateur de base de données répartie (DBsim) est un système à événements discrets conçu initialement pour l'évaluation des performances des protocoles de contrôle de concurrence dans un environnement réparti.

DBsim est, comme NS, écrit de façon modulaire et extensible en C++. Nous avons donc remplacé le module responsable de la gestion du réseau du simulateur DBsim par NS. Outre l'aspect mobilité, NS permet une estimation plus, réaliste pour l'environnement considéré, du délai de transit des paquets en fonction de la capacité des liens, la topologie du réseau, le routage, etc, alors que le gestionnaire de réseau original de DBsim modélise un délai de transmission fixe sur un type de réseau filaire (Cluster, LAN, WAN). En effet, dans un système de base de données répartie, le coût de communication constitue une part importante du coût global d'accès aux données et ceci est encore plus vrai dans le cas de l'environnement sans fil et mobile.

NS-DBsim est donc né de l'intégration de DBSim au dessus de la couche NS qui se charge de l'aspect communication sur réseaux mobiles. Les extensions que nous avons apportées ont permis de synchroniser les temporisateurs (timers) des deux systèmes, l'unité de temps utilisée par les deux systèmes qui était exprimée en millisecondes dans DBSim et en secondes dans NS, le paramétrage d'une simulation DBsim via l'interface de l'interpréteur TCL alors qu'initialement la recompilation du code C++ était nécessaire à chaque modification de paramètres, l'extension des stations bases pour qu'elles puissent héberger des applications, etc. La plateforme NS-DBsim (figure 5.1), qui est un facteur essentiel pour la modélisation des performances [Kou04], offre un environnement mobile d'exécution pour les transactions. Les principaux modules de cette plateforme sont le gestionnaire de transactions *(Transaction Manager: TM)*, le gestionnaire de données *(Data Manager: DM)*, le controleur de concurrence *(Concurrency Controller: CC)*, la couche communication *(Communication Layer: NS)*, le gestionnaire d'événements *(Event Manager: EM)* et l'éditeur de rapport *(Report Editor: RE)*.

[22] Un réseau ad hoc network n'inclue aucun support fixe telles que les stations base.
[23] Créé à NIT « Norweigian Institute of Technology ». http://www.ntnu.no/

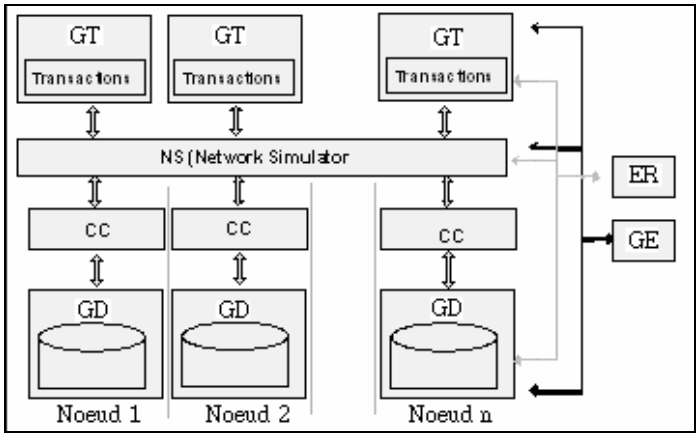


Figure 5.1-Architecture du simulateur DBsim

Le gestionnaire d'événementS (GE). Le GE est constitué de la boucle de simulation et d'une file d'attente FIFO des événements. La boucle de simulation est paramétrée par le nombre fixé de transactions générées pendant la simulation. Un événement est défini par son type (TM, DM, NET) et le moment où l'événement sera exécuté. Si l'événement est de type "TM", une transaction demande une nouvelle opération. Si l'événement est de type "DM", le DM associé à l'événement est appelé à exécuter une opération Lecture/Ecriture. Si l'événement est de type "NET ", le gestionnaire de réseau est appelé pour faire transiter un paquet de données. Pendant une itération de la boucle de simulation, un événement est extrait de la file d'attente des événements et est exécuté.

Le gestionnaire de transaction (GT). Le TM, est responsable de la création et de l'exécution des transactions. Lors de la création d'une transaction, l'ensemble des opérations associé à la transaction est dispatché entre les contrôleurs de concurrence. Le nombre des opérations dépend du type de la transaction (longue, courte). Le nombre de transactions concurrentes désigne la charge du système. Dans chaque simulation, le nombre de transactions concurrentes actives est fixé. Ce paramètre est désigné par MPL *(Multi Programming Level).*

Le gestionnaire de réseau (NS). DBsim simulait le transfert de données en supposant un délai de communication fixe selon le type de réseau. Mais avec l'intégration de NS ce délai est variable et est fonction des spécificités du réseau de communication sans fil et mobile considéré.

Le gestionnaire de données (GD). Chaque nœud contient un gestionnaire de données qui opère indépendamment des autres gestionnaires de données. Pour simuler l'exécution d'une opération Lecture/Ecriture, le DM calcule le moment où l'opération sera terminée et insert cet événement dans la file d'attente des événements. Le simulateur DBsim simule l'existence d'un disque et d'un cache sur chaque nœud du réseau. Une opération d'écriture sur le disque consomme plus de temps qu'une opération de lecture. La politique FIFO est employée pour la gestion du disque.

Le contrôleur de concurrence (CC). Un contrôleur de concurrence (appelé aussi scheduler), dans un système de gestion de bases de données, contrôle les exécutions concurrentes des transactions afin de satisfaire la propriété de sérialisation. Le contrôleur de concurrence contrôle l'ordre dans lequel le DM exécute les opérations Lecture, Ecriture, Validation et Annulation de transactions. Dans DBsim, deux techniques de contrôle de concurrence ont été implémentées : le 2PL (Two Phase Locking) et TO (Timestamp Ordering). Quand le contrôleur de concurrence reçoit une opération Lecture/Ecriture du TM, il envoie l'opération au DM pour exécution selon sa technique. Lorsque l'opération est terminée, le contrôleur de concurrence est notifié par le DM.

L'éditeur de rapport (ER). L'éditeur de rapport est employé pour collecter les statistiques lors d'une simulation. Il est responsable de calculer le nombre de transactions validées, celles annnulées selon leur type (courtes, longues), le temps d'exécution moyen, etc.

5.2 Les paramètres

Notre objectif est de mettre en évidence les indices de performance et les avantages de chaque approche. La comparaison des protocoles a été définie selon les critères de performance suivants [CSA98]: la *complexité en nombre de messages* représentant le nombre de messages échangés entre les sites participants, le *throughput* qui est le nombre de transactions validées par seconde, le taux d'annulation en fonction de la charge du système et la *complexité en temps* correspond à la latence du protocole. La réduction de cette latence permet de réduire la période d'exécution globale d'une transaction et de libérer rapidement les ressources [ACL87]. Nous avons également étudié l'impact du handoff et des déconnexions sur les performances des protocoles.

Le paramètre essentiel que nous avons employé pour modéliser les performances est la charge de travail qui représente le trafic d'entrée du système [Kou04]. La charge de travail est indiquée par le paramètre du niveau de multiprogrammation *(multiprogramming level ou MPL)* qui définit le nombre de transactions actives dans le système [NSB97, HRG97, CSA98, KDDS00]. Plusieurs autres paramètres sont définis tels que la longeur d'une transaction (courte vs. longue), la probabilité pour une transaction d'être en lecture (vs. écrire), délai d'une opération disque, etc. Nous avons utilisé également la probabilité de handoff un comme paramètre d'entrée [KU99] pour analyser l'effet de mobilité sur les ACPs.

Les paramètres et les initialisations que nous avons utilisés dans les simulations ont été inspirés par les travaux similaires sur les systèmes distribués [CSA98, LAA98, CL99, KU99, NSB97, HRG97] et sur les travaux moins nombreux proposés sur les environnements mobiles [CBML03, KU99]. En effet, il n'y a pas de recule dans le domaine de simulation des techniques de transactions mobiles, aussi est-il difficile d'établir un modèle de simulation. Notre seul guide a été de s'inspirer des travaux réalisés dans le contexte distribué et de tenir compte des spécificités technologiques du contexte mobile. Les tables 5.1 et 5.2 résument les paramètres liés aux aspects base de données et transactions et les tables 5.3 et 5.4 à ceux du réseau (en particulier sans fil).

La distribution des données et les transactions satisfont la propriété d'hétérogénéité qui n'est pas envisagée dans les systèmes traditionnels qui traditionnellement suppose que tous les sites et leurs charges de travail sont identiques et que toutes les interactions avec la base de données sont symétriques pour tous les sites. Dans les sytèmes mobiles, l'hétérogénéité est liée aux sites ou noeuds participant aux transactions en particulier en termes de capacité de stockage de données et de traitement transactionnel qui diffèrent d'un noeud à l'autre, sans oublier que deux types de réseaux, filaire et sans fil sont utilisés. Un noeud mobile est supposé contenir approximativement 1000 données qui représente 5% de toute la taille de la base de données (20000) [DCK+94, Vid01].

Excepté de *délai de transmission acceptable* pour le protocole TCOT, les paramètres de la table 5.3 et de la table 5.4 représentent les aspects réseau et communication qui sont intrinsèques à NS et permettre une description précise du modèle fondamental de communication ou un scénario de mobilité utilisé pendant une simulation. Un scénario de mobilité décrit l'architecture de réseau sans fil spécifique (le nombre de nœuds mobile/fixes, le nombre de stations base, l'interconnexion entre les divers noeuds, etc.) et l'information concernant le déplacement dans une cellule ou entre les cellules. La technologie utilisée est

IEEE 802.11 et la mobilité est gérée par le protocole Mobile-IP 4.0[24]. Pour les paramètres de la mobilité, nous imaginons un marchand itinérant se déplaçant entre ses clients et s'arrêtant parfois pendant un moment pour présenter et discuter de son produit avec ses clients potentiels (table 5.4).

Etant donné que NS est dédié nativement à la simulation des protocoles réseau, il permet un paramétrage riche des fonctionnalités des réseaux. Nous avons défini seulement les paramètres les plus pertinents pour nos évaluations.

Table 5.1-Les paramètres des fonctions des systèmes de base de données et des transactions

Paramètre	Définition
ACP	Le protocole de validation de transactions mobiles simulé.
Contrôleur de concurrence	Le contrôleur de concurrence utilisé pour assurer la sérialisation. Par exemple : 2PL.
MPL	Le nombre de transactions concurrentes dans le système à tout moment.
Nombre de nœud	Nombre des nœuds (mobiles et fixes) simulés dans le système, ils sont interconnectés via NS
Nombre d'opérations par transaction	Le nombre d'opérations demandées par une transaction. Selon ce nombre, une transaction est définie comme courte ou longue.
Taille de la base	Le nombre de données dans la base de données.
La distribution de données	La distribution des données sur les différents nœuds du système; le pourcentage des données localisé sur un nœud particulier. Nous devons respecter l'hypothèse d'hétérogénéité de notre modèle de simulation en considérant une distribution adéquate sur les nœuds mobiles.
La distribution des transactions	La probabilité qu'une nouvelle transaction soit générée par un nœud particulier. Nous devons respecter l'hypothèse d'hétérogénéité de notre modèle de simulation en considérant une distribution adéquate sur les nœuds mobiles.
Localité des données	La probabilité qu'une transaction accède à une donnée localisée sur le même nœud qu'elle.
Accès distant aux données	Lorsqu'une transaction accède à une donnée non localisée sur le nœud où elle est créée, c'est la probabilité que cette donnée soit localisée sur un site particulier.
Probabilité de transaction courte (Vs. longue)	La probabilité qu'une transaction soit courte (vs. longue).
Probabilité d'opération Lecture (Vs. Ecriture)	La probabilité que l'opération d'accès à une donnée soit une opération de lecture (vs. écriture)
Probabilité d'annulation	La probabilité qu'une transaction demande l'annulation avant la validation, cette probabilité est la même pour les transactions courtes et longues
Temps d'opération disque (fixe/mobile)	Temps nécessaire pour accéder à une donnée sur le disque. Nous devons respecter l'hypothèse d'hétérogénéité de notre modèle de simulation en attribuant une valeur adéquate à ce paramètre selon le type d'un nœud (mobiles, fixes).
Temps du message interne	Temps de traitement d'un message autres que la demande d'opération disque/cache.

Table 5.2-Les valeurs des paramètres

Paramètre	Min	Max
Nbr. d'opérations dans une transaction courte	2	8
Nbr. d'opérations dans une transaction longue	14	100
Temps entre transactions	10 ms	100 ms
Temps entre les demandes d'opération	1 ms	10 ms
Temps d'opérations disque d'un nœud fixe	8 ms	16 ms
Temps d'opérations disque d'un nœud mobile	10 ms	40 ms
Temps de traitement d'un message	1 ms	16 ms

Paramètre	Valeur
Nombre de transactions (max)	20 000
Nombre de données	20 000
Probabilité de transaction courte	80 %
Probabilité d'annulation	0.1 %
Probabilité de lecture	80 %

[24]http://www.ietf.org/html.charters/mobileip-charter.html

Table 5.3-Les paramètres des fonctions du réseau de communication sans fil

Paramètre	Définition
Nombre de nœuds mobiles	Nombre des nœuds mobiles simulés dans le système.
Nombre stations base	Nombre des stations bases simulées dans le système, il définit le nombre de cellule couverte dans le réseau sans fil.
Nombre de nœuds fixes	Nombre des nœuds fixes simulés dans le système
Taille d'un message	Taille des messages envoyés sur NS par des agents TcpApp.
Capacité des liens filaires	Capacité de transmission des messages en Mégaoctet sur les liens filaires
Type de liens filaires	Type de transmission sur le lien filaire
Topographie de la zone de couverture	La dimension (x, y) de la zone couverte par une station base
Vitesse de déplacement	La vitesse de déplacement des unités mobiles (en m/s)
Durée de pause	Durée d'immobilité d'une unité mobile dans une localisation
Délai de transmission acceptable pour le protocole TCOT	Un intervalle de temps acceptable [MinWirelessDelay, MaxWirelessDelay] utilisé seulement dans le protocole TCOT afin estimer le temps de transfert des mises à jour St .
Durée de déplacement global des unités mobiles	Les scénarios de mouvement dans NS sont définis par le modèle RWM (Random Waypoint Mobility) [TMUY 03, PLV 05]. Un scénario de mouvement est une alternance entre une phase de déplacement et une phase de pause (immobilité) des nœuds mobiles pendant une durée qu'on appelle durée de déplacement globale des unités mobiles.

Table 5.4-Les valeurs des paramètres

Paramètre	Valeur
Capacité des liens filaires	10 Mo
Capacité des liens sans fil	2 Mo
Type de liens filaires	Duplex
Topographie de la zone de couverture	670 x 670 m²
Vitesse de déplacement	1 m/s
Durée de pause	100 s
Durée de déplacement global des unités mobiles	20000 *s* (~ 6h)

Paramètre	Min.	Max.
Taille d'un message	500 octets	1000 octets
Délai de transmission acceptable pour le protocole TCOT	50 ms	500 ms

5.3 Résultats des simulations

Le résultats présenté dans cette section son basés sur les topologies de réseau sans fil de la figure 4.2. Notons que chaque simulation s'est déroulée pour atteindre le traitement de 20000 transactions.

5.3.1 Throughput vs. MPL et latence vs. MPL

Comme nous pouvons voir sur les figures 5.3 (sans mobilité) et 5.4 (mobilité dans la cellule), les protocoles CO2PC, UCM et M2PC ont des throughputs presque semblables pour des MPLs variant de 5 à 200. Le protocole TCOT donne de très bonnes performances quand le MPL est bas. Avec une augmentation du PML sa performance chute de manière significative, il devient le pire des protocoles. L'expiration fréquente de son timeout et par conséquent l'augmentation du taux d'annulation est responsable de ce comportement. Avec un MPL croissant le throughput du protocole 2PC devient plus proche de celui de CO2PC, de UCM et de M2PC et se comporte mieux que TCOT. Ceci s'explique par ce qui suit.

Avec un nombre élevé de noeuds mobiles, les courbes tendent à se rencontrer pour toutes les valeurs de MPL en raison de la congestion du réseau sous-jacent IEEE 802.11. En effet, quand le nombre de noeuds augmente dans une cellule, la performance du canal de transmission diminue dès que le débit maximal supportable est atteint [YV02, Bian00, Pha05].

Pour comprendre ce phénomène, nous avons cherché dans les fonctionnalités du réseau sans fil, c'est-à-dire ; la technique IEEE 802.11 de la gestion du canal. En effet, dans [YV02, Bia00, Pha05], la performance de la méthode d'accès DCF *(Distributed Coordination Function)* a été étudiée. Dans ce travail, la métrique "Throughput of saturation" des canaux de transmission est définie comme étant la limite que peut atteindre le système quand la charge augmente (le nombre de nœuds augmente). Selon les indications de la figure 5.2 [YV02], quand le nombre de noeuds augmente dans le réseau sans fil, la performance de IEEE 802 11 diminue sensiblement à cause du degré élevé de collision sur de support de communication

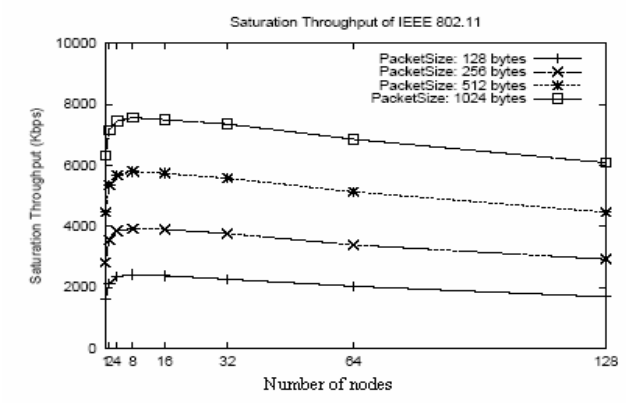


Figure 5.2-Throughput de saturation dans IEEE 802.11

De plus, quand le MPL augmente, le taux de conflit entre les transactions augmente aussi conduisant à la diminution des throughputs.

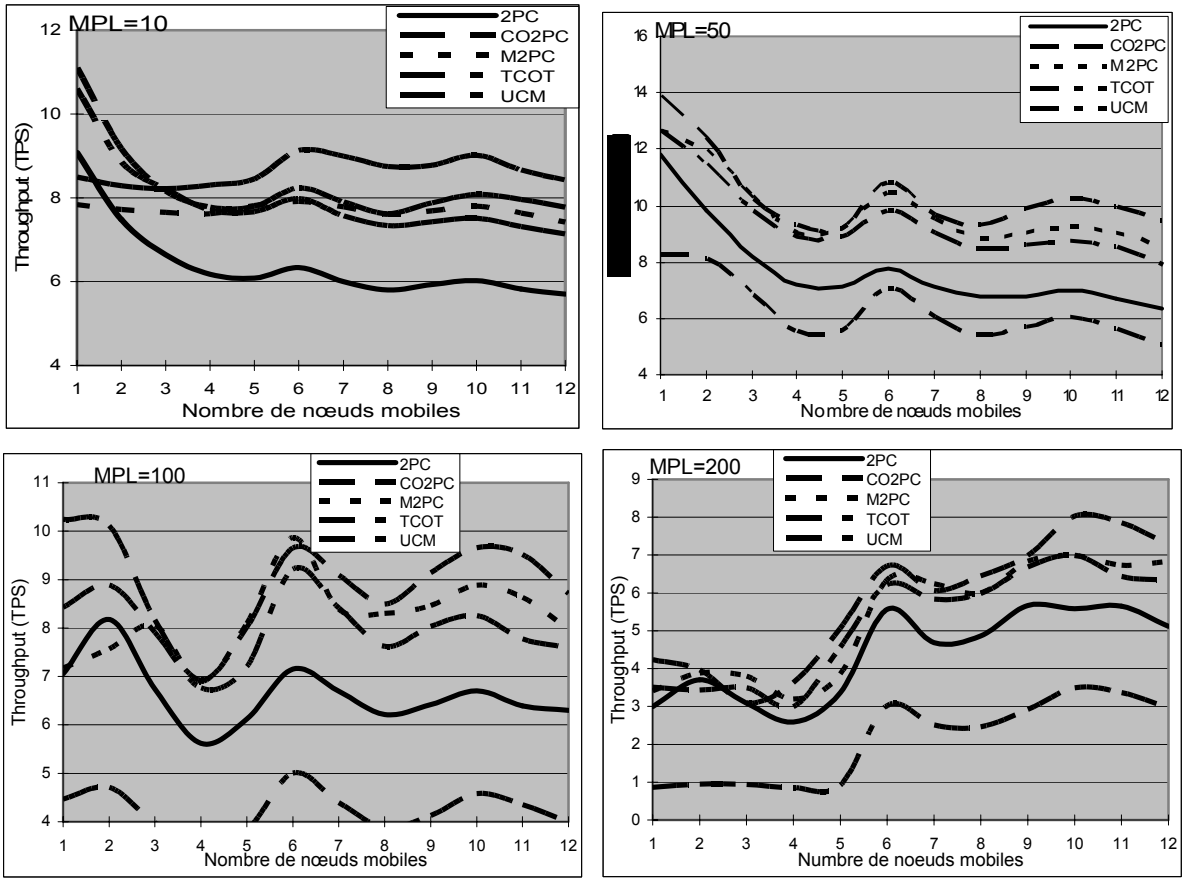


Figure 5.3-Throughputs, sans mobilité

Sur la figure 5.5, l'effet de la mobilité est significatif sur le throughput des transactions et les fluctuations sont très importantes. Il varie entre 1 et 14 transactions par seconde sur la figure 5.3, contre 0.2 et 3 sur la figure 5.4. Cependant, le nombre de noeuds n'est pas le seul facteur influent. En effet, la localisation ou les positions exactes des stations base, la vitesse, la direction, la durée de des pauses et la portée du signal ont également leur impact [TMUY03]. En outre, pendant que les noeuds se déplacent ils peuvent quitter la cellule pendant un moment et être hors de la portée du signal. Ces déconnexions provisoires et leurs durées influence également les performances; la longueur d'une déconnexion peut augmenter le temps de réponse et/ou la probabilité d'annulation. Ces paramètres sont liés à la nature même de l'application. Les fluctuations observées sont dûes au fait que la localisation (position dans la cellule) et la direction du nœud mobile sont choisis de façon aléatoire par NS.

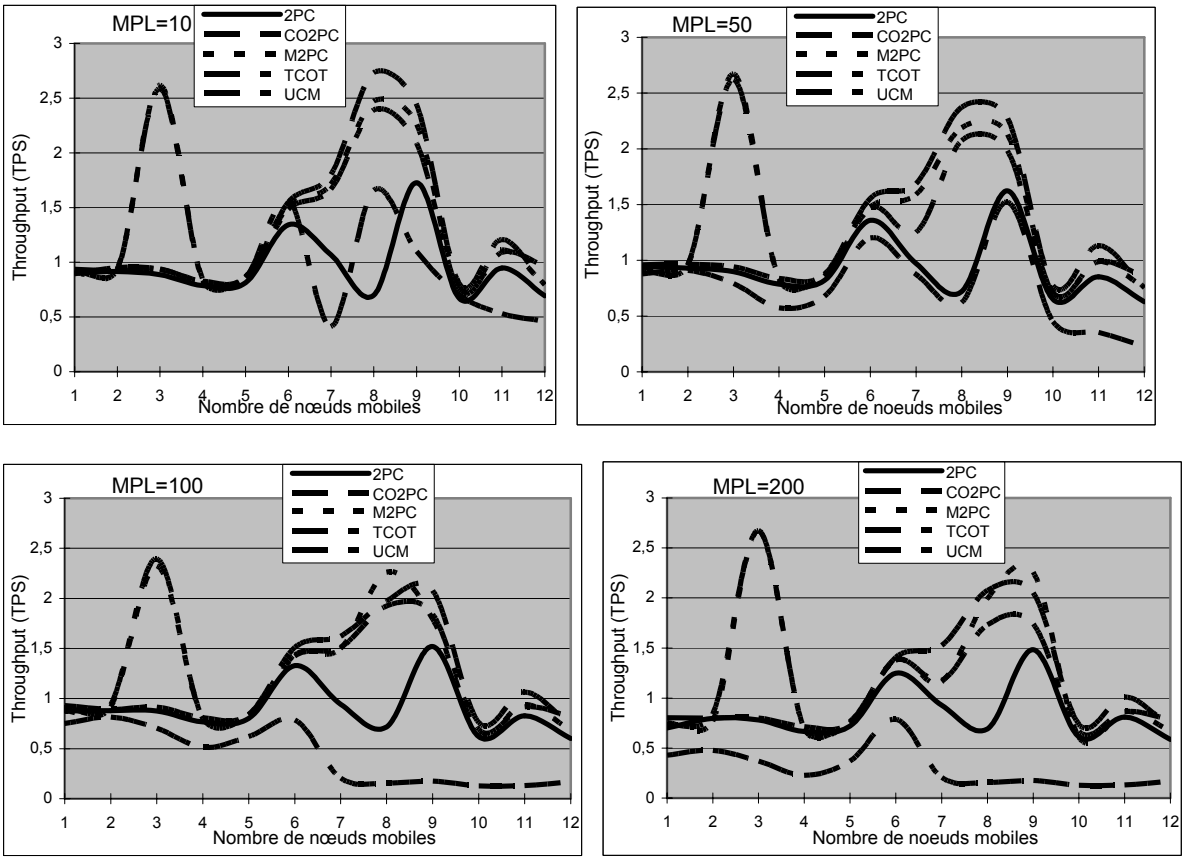


Figure 5.4-Throughputs, mobilité intra-cellule (pas de handoff)

TCOT offre la meilleure latence dans les deux cas, avec ou sans déplacement des noeuds mobiles, parce que le timeout limite la durée d'exécution d'une transaction dans tous les cas. CO2PC est le plus puissant dans le cas où les noeuds ne se déplacent pas ; mais UCM a presque la même latence que CO2PC. La courbe de M2PC est près de celle de UCM. Pour les MPLs faible à moyen CO2PC, UCM, M2PC et 2PC ont des courbes qui s'imbriquent dans le cas d'un petit nombre de noeuds mobiles en déplacement. Quand le nombre de noeuds est plus important, 2PC est le moins performant. Cependant pour des valeurs élevées de MPL, 2PC tend à approcher UCM et Co2PC et à être meilleur que M2PC (figure 5.5, figure 5.6). La latence est très élevée quand les noeuds participants se déplacent. En effet, la latence varie entre 19 ms et 2300 ms sur la figure 5.5, contre 19 ms et 23000 sur la figure 5.6. La différence est significative et inacceptable pour une application.

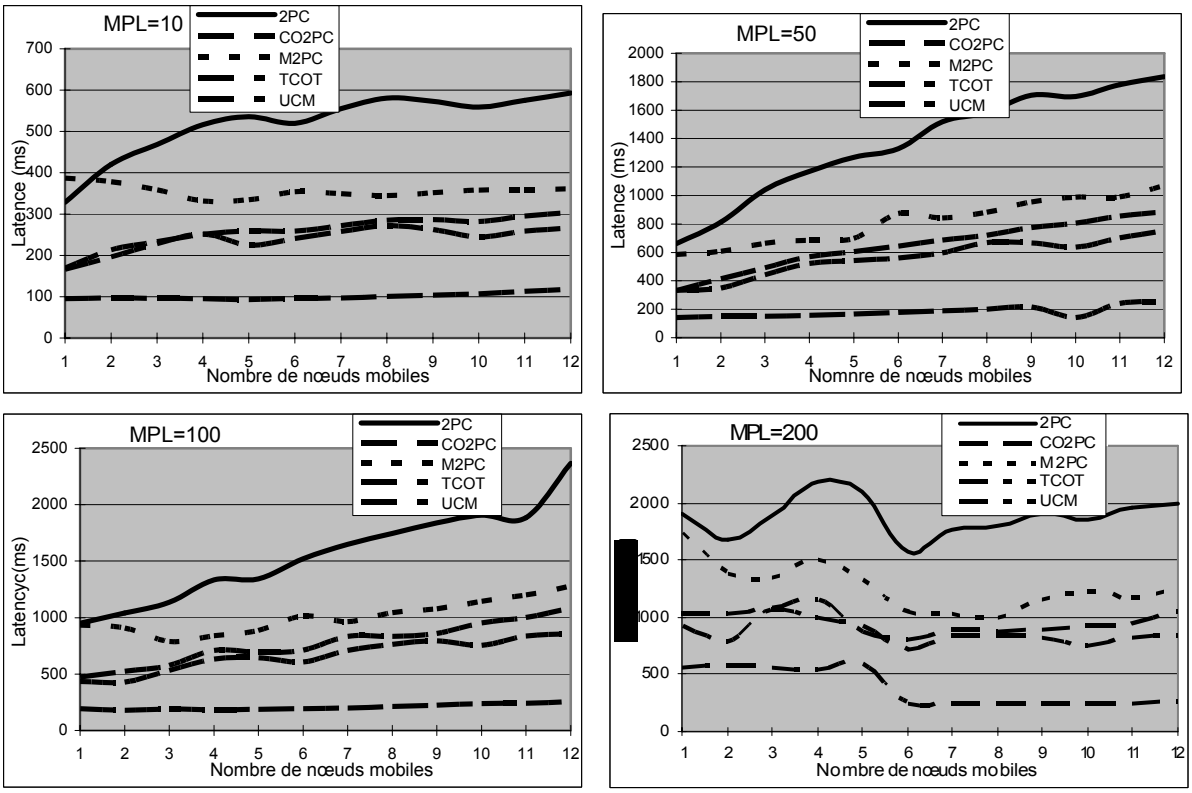


Figure 5.5-Latences, sans mobilité

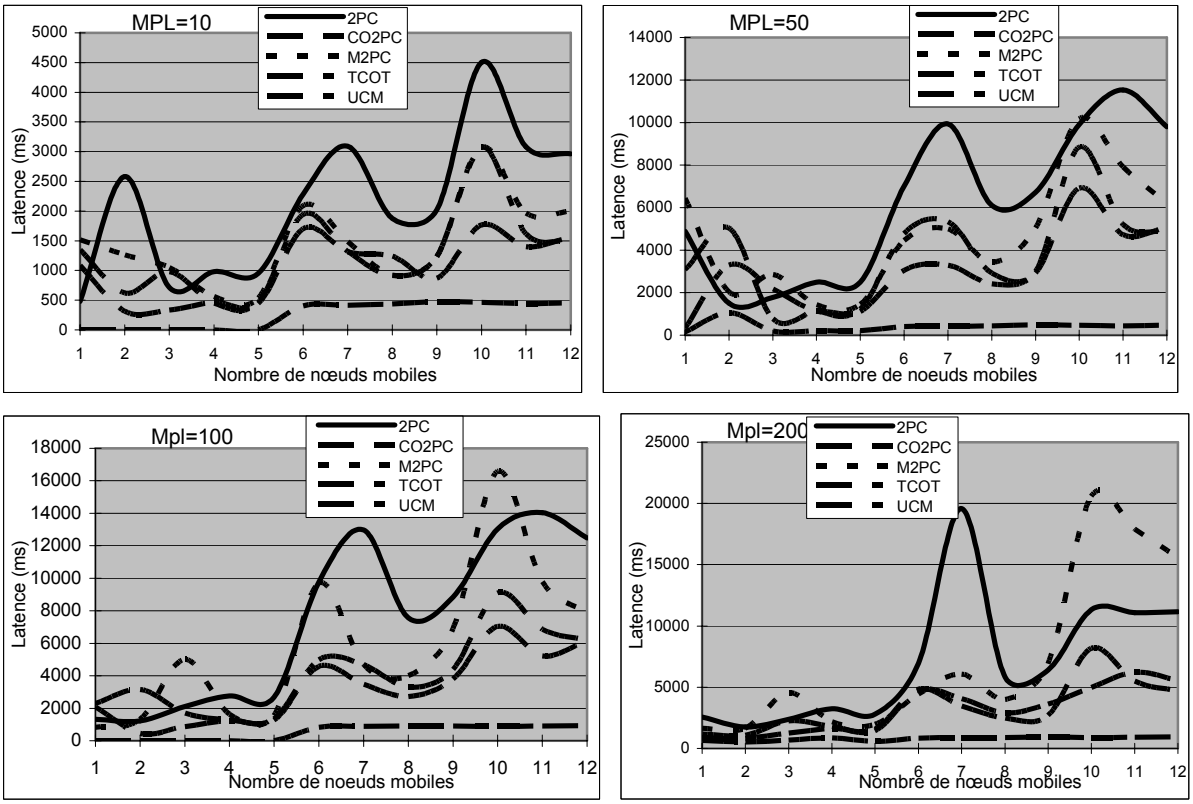


Figure 5.6-Latences, mobilité intra-cellule (pas de handoff)

5.3.2 Impact du handoff

Afin d'analyser l'effet du handoff sur les protocoles, nous avons utilisé le modèle suggéré dans [PLV05] pour créer des scénarios de mobilité. Pour analyser l'effet du handoff sur les protocoles, nous avons utilisé le modèle suggéré dans [PLV 05] pour la création des scénarios de mouvement. Ces scénarios de mouvement sont paramétrés par :

1. Le nombre de visites qu'un mobile exécute dans une cellule avant d'effectuer un handoff. Une visite désigne le déplacement d'une position vers une autre position dans la même cellule.
2. La probabilité d'effectuer un handoff d'une cellule à une autre après un nombre donné de visites.
3. La vitesse et le temps de pause avec lesquels l'utilisateur se déplace.

Les tests ont montré que lorsque les nœuds mobiles ne restent pas suffisamment lontemps dans la même cellule, le temps de réponse des protocoles s'allonge. Sur les figures 5.7 et 5.8, nous avons présenté le cas où le nombre de visites est fixé à 5 pour tous les protocoles. Pour tous ces protocoles, l'impact sur les performances ne commence à être visibles qu'à partir de la probabilité de 0.8. TCOT est le plus mauvais et le plus sensible aux handoff. UCM et CO2PC sont les meilleurs loin devant TCOT. Remarquons comme même une certaine stabilité de ces protocoles en dessous de la probabilité 0.8. Le protocole 2PC n'est pas aussi mauvais qu'on pouvait s'y attendre comparé à UCM et CO2PC particulièrement en termes de latence. Cepenant, son throuput commence à chuter à partir de la probabilité 08. Ce résultat, conjugué à celui du chapitre 4 précédent, suggère que Mobile-IP fournit un bon support pour un fonctionnement correct du système quand la fréquence du handoff est raisonnable (nous supposons la macro-mobilité).

Notons enfin que d'autres simulations sont nécessaires pour étudier les stratégies relatives au handoff du contrôle; c'est-à-dire, pour vérifier quelle stratégie est la meilleure: celle où le contôle (coordinateur/participant) suit l'unité mobile d'une cellule à l'autre ou celle ou le contrôle est statique.

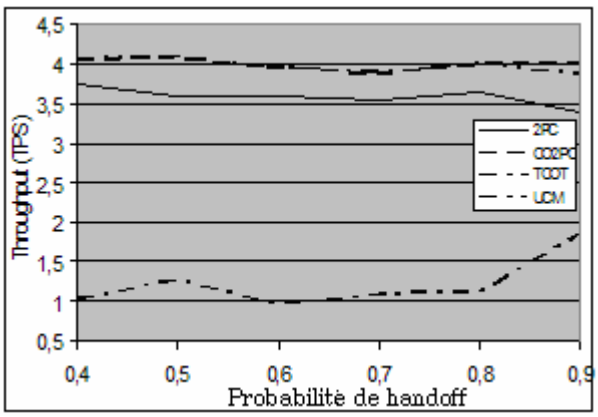


Figure 5.7-Impact du handoff sur le throughput

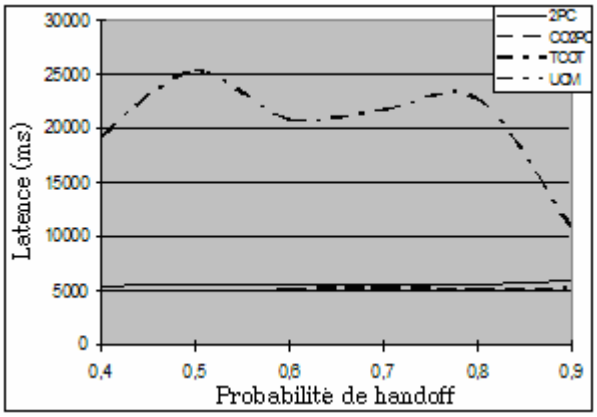


Figure 5.8-Impact du handoff sur la latence

5.3.3 Impact des déconnexions

Dans cette expérience, M2PC est exécuté avec le support de Mobile-IP, et le nombre de nœuds mobiles est fixé à 6.

La figure 5.9 montre qu'en absence de déconnexions, le temps de réponse est meilleur pour tous les ACPs avec la tendance de M2PC[25] et de 2PC d'avoir un meilleur temps de réponse pour MPL>100. Le throughput d'UCM et de CO2PC est le meilleur, suivi de M2PC pour MPL<100. Toutefois 2PC tend clairement à montrer une meilleure performance à partir du MPL=100, suivi de M2PC avec une tendance à la hausse et une chute de performances pour UCM, CO2PC et TCOT ; ce dernier étant le plus mauvais. On note également un pic positif pour MPL=50 pour tous les ACPs excepté 2PC qui est très bas pour ce point. En l'absence des déconnexions, la valeur du MPL=50 semble donner de bons résultats pour tous les protocoles excepté 2PC qui est meilleur pour MPL>100. Quand la zone est bien couverte et le support de mobilité est fourni par Mobile-IP, 2PC n'est pas si mauvais avec des valeurs élevées de MPL. Les déconnexions augmentent le temps d'exécution et diminue le throughput des transactions pour tous les ACPs, ceci est complètement prévisible. Ce qui est inattendu c'est que tous les ACPs sont plutôt semblables entre eux mis à part TCOT qui affiche clairement le plus mauvais temps de réponse. Le 2PC fusionne avec les autres protocoles en particulier pour le throughput.

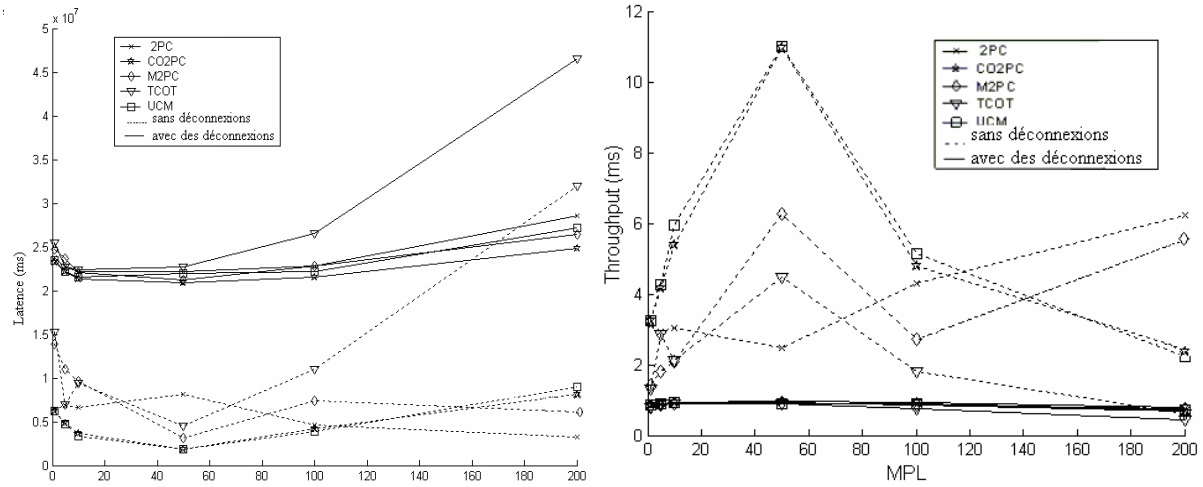


Figure 5.9-Impact des déconnexions

5.3.4 Nombre de messages vs. MPL

Avec ou sans le mouvement des noeuds mobiles, les résultats sont identiques puisque les messages sont de niveau application et dépendent seulement du nombre de transactions effectuées et des protocoles ACPs utilisés. Les résultats pour 10 noeuds mobiles sont

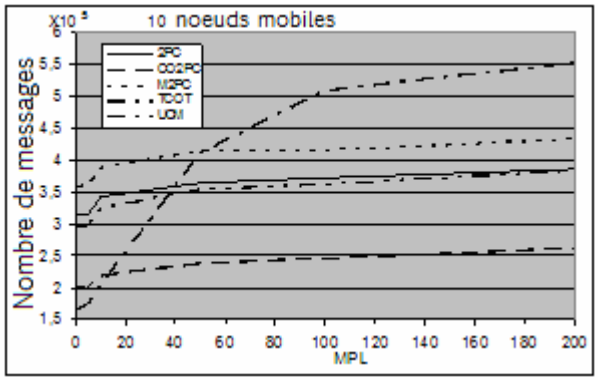


Figure 5.10-Comparaison de la complexité en messages

[25] Version adaptée par la prise en charge des déconnexions seulement (sans le mécanisme de mobilité).

présentés sur le schéma 5.10 ; nous avons obtenu des complexités presque semblables pour différentes configurations. Le nombre de messages présentés inclut les phases d'exécution et de validation. Dans UCM, la phase de vote a été enlevée mais des messages de journalisation sont ajoutés pendant la phase de l'exécution de transaction; c'est pourquoi presque la même complexité est obtenue pour 2PC et UCM bien que ce dernier soit un protocole à une phase.

Contrairement à UCM, CO2PC qui est également un protocole à une phase, effectue une validation locale avant une validation globale, il ne nécessite pas des messages additionnels. TCOT présente de bonnes performances quand le MPL est bas mais ce n'est pas le cas quand le MPL augmente. Comme prévu, M2PC requière plus de messages que 2PC en raison de ceux utilisés pour informer le client et les participants mobiles sur le résultat.

5.3.5 Taux d'annulation vs. MPL

La figure 5.11 présente les taux d'annulation des ACPs. Ces taux sont proches de ceux du 2PC pour tous protocoles excepté TCOT. En ce qui concerne le protocole TCOT, l'expiration du timeout est responsable de l'annulation des transactions. Les transactions ne restent pas dans le système suffisamment longtemps pour être la cause des conflits; la transaction est annulée à l'expiration du timeout. Nous précisons que la probabilité pour qu'une transaction soit annulée par l'utilisateur est basse (0.1).

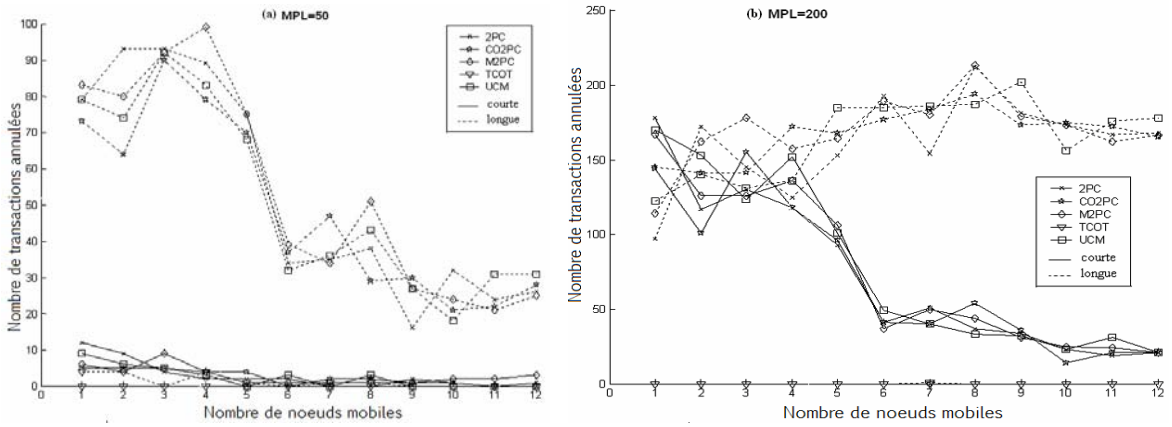


Figure 5.11-Comparaison des taux d'annulation

L'utilisation de timeouts dans un ACP mobile produit potentiellement un taux élevé d'annulations s'ils ne sont pas correctement choisis. Il est très difficile de faire ce choix parce qu'il dépend de beaucoup de facteurs tels que la technologie sans fil sous-jacente, le modèle de mobilité, etc.

5.3.6 L'effet de la localisation du coordinateur sur une station base

Les protocoles donnent de meilleures exécutions lorsque le coordonnateur est situé sur la station base; cependant le bénéfice n'est pas très significatif. Les figures 5.12 et 5.13 illustrent cette observation sur les protocoles M2PC et CO2PC respectivement. Ce constat est le même avec ou sans mobilité des noeuds. En effet, quand le coordonnateur est sur une station base, ses messages sont directement transmis aux noeuds mobiles tandis que quand il est sur un noeud fixe, les messages qu'il achemine aux noeuds mobiles sont d'abord conduits à la station base avant que cette dernière ne les fasse suivre aux noeuds mobiles. Ainsi, deux délais sont nécessaires dans le dernier cas. Il faut noter que dans IEEE 802.11 les stations base ne sont pas conçues pour supporter d'autres fonctions que celles liées au réseau, mais nous les avons étendues pour l'exécution du coordonnateur pour rester conformes à la conception des protocoles étudiés.

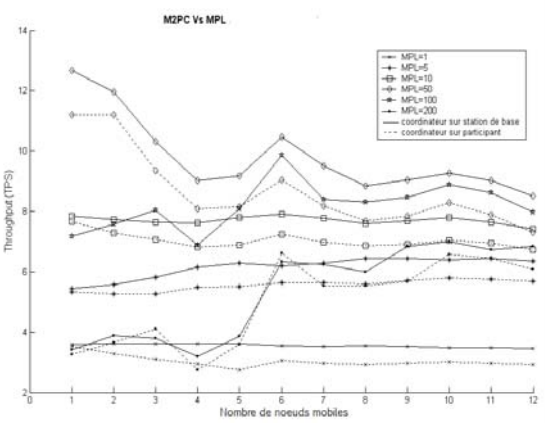


Figure 5.12-Influence de la station base sur M2PC

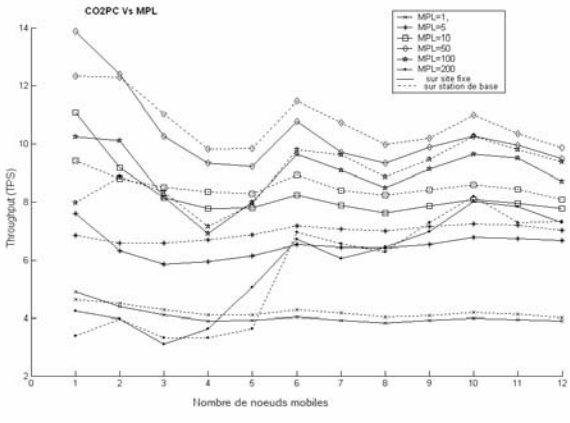


Figure 5.13-Influence de la station base sur C02PC

5.3.7 Taille du journal (log size) vs. throughput et latence

Dans cette expérience, nous désirons analyser l'impact de l'envoi du transfert des journaus sur les performances de M2PC mais avec le support de Mobile-IP. L'étude de l'impact de la taille des journaux sur le throughput et la latence du protocle M2PC a été réalisée de la façon suivante:

-pour une valeur de MPL donnée, une probabilité de handoff fixée et un nombre de nœuds mobiles précis, on fait varier la taille du journal et on mesure la latence puis le throughput.

- de nombreuses évaluations ont été effectuées pour plusieurs valeurs de MPL en fixant le nombre de nœuds et variant la probabilité de handoff et vice versa. Un échantillon de ces évaluations est représenté dans la figure 5.14. Suite à l'analyse des différents diagrammes réalisés, nous avons conclu que la taille du log n'a pas une influence significative sur le troughput ou la latence sauf si le nombre de nœuds mobiles est élevé et ce à partir d'une taille supérieure à 1000 octets. Une forte mobilité exprimée par une probabilité de handoff élevée ne détériore pas forcément les performances à cause du transfert des journaux. Lorsque la probabilité de handoff est élevée, il y a moins de concentration des nœuds au niveau de la même cellule, c'est là où les performances sont meilleures. Autrement dit, c'est la charge du réseau sans fil en nombre de nœuds mobiles qui désavantage significativement les performances des protocoles lorsque la taille des journaux devient importante. Ce résultat confirme d'ailleurs le résultat de l'analyse du protocole M2PC (chapitre 4).

5.3.8 Discussion et résumé de la comparison

La comparaison entre les protocoles est récapitulée dans la table 5.5; le nombre 1 indique le plus puissant et 5 le moins puissant. Cette classification tient compte globalement de toutes les évaluations réalisées dans les conditions spécifiées plus haut.

L'avantage de NS-DBsim est de fournir la possibilité de tenir compte et de tester différentes valeurs à la fois des paramètres des réseaux mobiles et des paramètres des techniques de transaction. Ceci nous a permis de montrer à quel point l'impact des paramètres de ce type de réseaux est significatif et imprévisible. Notons aussi que l'évaluation réalisée dans ce chapitre concernant les protocoles de validations des transactions en environnement mobile est, à notre connaissance, pionnière dans son genre. En effet, contrairement aux protocoles de validation en environnement distribués qui ont connus une littérature assez abondante en termes d'étude et d'évaluation, nous n'avons pas rencontré de littérature similaire concernant l'environnement mobile. Seul l'étude présentée dans [BRLS04] est consacrée

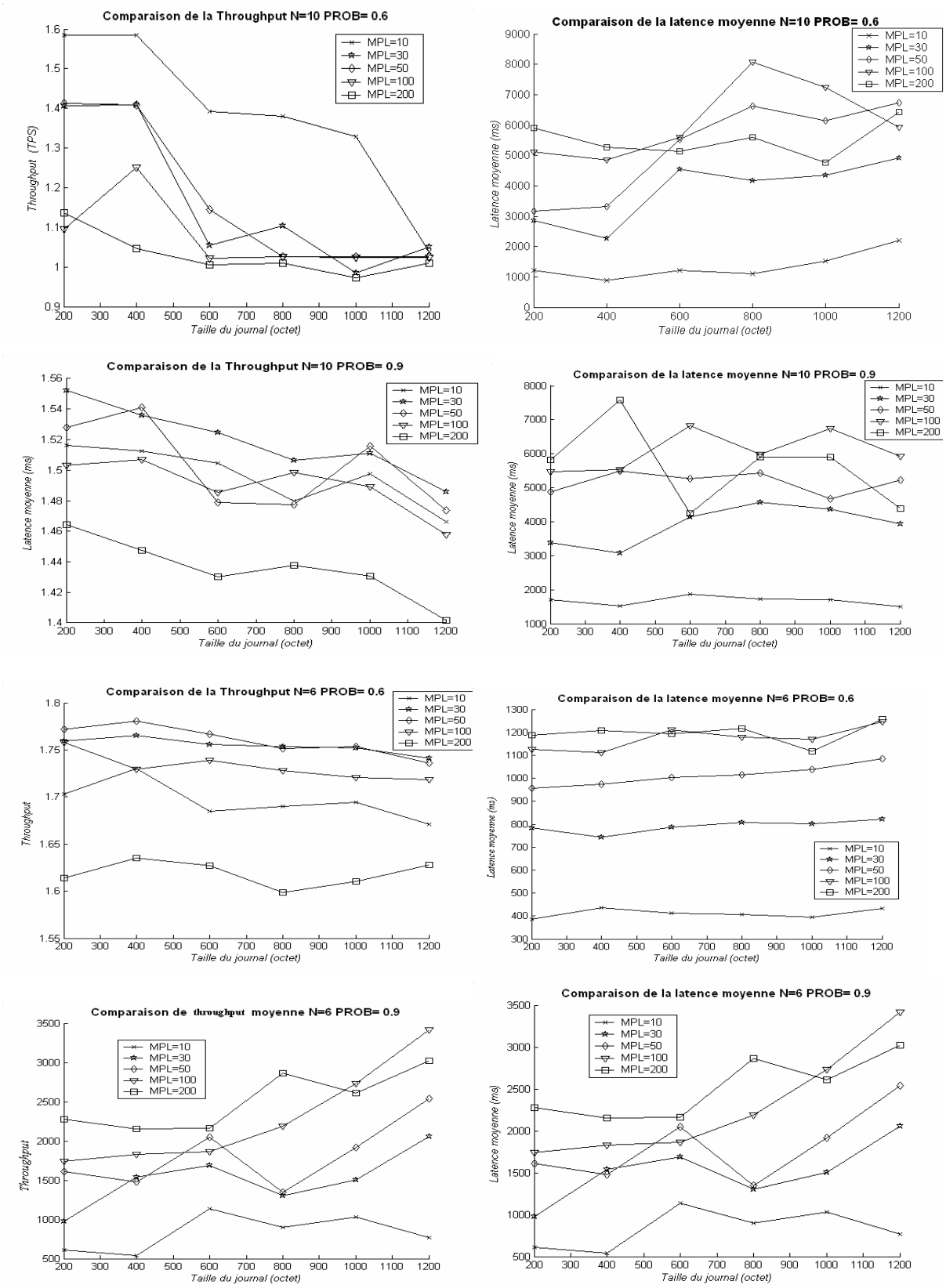


Figure 5.14-Impact de la taille du journal

à la comparaison des protocoles 2PC, CO2PC et UCM dans un environnement de simulation qui n'est pas destiné à modéliser spécifiquement un environnement mobile. L'évaluation réalisée dans [BRLS04] a été limitée à l'étude des effets des déconnexions sur ces trois

protocoles en termes de probabilité de blocage, de temps de validation et du taux d'annulation à tort. Il a été conclu que ces trois protcocoles avaient un comportement correct dans des conditions plutôt pessimistes en termes de connectivité et que UCM offrait les meilleures performances. Cette évaluation a été présentée par ces auteurs comme une première étape d'une étude comparative qu'ils ont initiée sur les protocoles de validation de transactions mobiles.

Le résumé des tables 5.5 et 5.6 et l'analyse des résultats de simulation nous ont permis de conclure ce qui suit :

- L'approche optimiste combinée avec 2PC est mieux que l'approche de TCOT basée sur des timeouts. CO2PC est le meilleur protocole. Cependant, ce protocole doit être exécuté sur une unité mobile avec un minimum de ressources pour pouvoir valider localement les sous-transactions. L'exécution de TCOT dépend des valeurs du timeout: une valeur élevée donne un bon throughput mais augmente la latence et inversement.

Table 5.5-Comparison globale des protocoles

ACP	Throughput			Latence		
	Sans mobilité	***Mobilité dans La cellule***	***Handoff***	*Sans mobilité*	***Mobilité dans la cellule***	***Handoff***
2PC	4	2	3	5	5	3
M2PC	2	1	2	4	4	2
CO2PC	1	1	1	3	2	1
UCM	3	1	1	1	3	1
TCOT	5	5	4	2	1	4

- L'approche à une phase sur laquelle est basé le protocole UCM est interessante car elle intègre des participants légers et n'exige pas d'interface "prepare", toutefois en termes de messages elle est plus coûteuse que CO2PC. De plus, UCM fait des hypothèses sur le contrôle de concurrence.

- Pour l'approche à deux phases du standard 2PC et du protocole M2PC. De façon globale, M2PC est meilleur particulièrement en termes de throughput. Cependant, si on analyse les résultats de plus près, le comportement de 2PC n'est pas exactement conforme à nos attentes et celles de la littérature. En effet, 2PC est un mauvais protocole en termes de latence, mais, il fusionne presque avec les autres protocoles dans le cas d'un nombre faible

Table 5.6-Impact des approches sur les performances

ACP	Approches	Impact sur le throughput	Impact sur la latence	Impact sur le nombre de messages
2PC	Two-phase standard Atomicité Durabilité garantie	Pas aussi Mauvais que prévu	Mauvais	Moyen (similair à UCM)
M2PC	Two-phases Support de mobilité Transfert des journaux Atomicité stricte Durabilité garantie	Bon	Moyen	Moyen (Pas loin de 2PC)
CO2PC	Validation optimiste Atomicité sémantique Durabilité non garantie	Meilleur	Bon	Meilleur
UCM	Validation à une phase Transfert des journaux Atomicity stricte Durability garanteed	Bon	Meilleur	Moyen (comme 2PC)
TCOT	Timeout Transfert des journaux Atomicité sémantique Durabilité garantie	Pire	Bon	Pire (à cause du taux d'abandon abortion)

de noeuds mobiles dans la cellule. En termes de throughput, il approche les autres protocoles pour des MPLs élevés. Mais comme nous l'avons expliqué, ceci peut s'expliquer par la technique de gestion du canal de IEEE 802.11 qui après un certain seuil de saturation, fait converger tous les protocoles. M2PC est une variante de 2PC tenant compte de la communication sans fil et des déconnexions. Cependant, son intérêt doit être considéré quand aucun soutien de mobilité n'est fourni (voir chapitre 4).

5.4 Conclusion

Les protocoles à deux phases ne font aucune hypothèse au sujet de la gestion des données et des techniques de contrôle de concurrence et assurent l'atomicité stricte. Leurs inconvénients se situent dans le coût en messages et la consommation d'énergie; un compromis entre ce coût et le désir de garantir une atomicité stricte s'impose. L'optimisme

Table 5.7-Analyse globale des approches

	Approche	**Facteurs négatifs**	**Facteurs et indices de performance**	**Contexte applicatif**
2PC	Deux phases Standard Atomicité stricte Durabilité garantie	Nombre de messages sans fil Déconnexion non prise en charge Blocage potentiel	Très répandue Propriétés fortes	Ressources de calcul et de communication importantes
M2PC	Deux phases Support de mobilité Transfert du journal Atomicité stricte Durabilité garantie	Transfert du journal Latence pas très bonne Trafic et journalisation additionnels	Déconnexion prise en charge Mobilité facile à déployer Propriétés fortes Performances acceptables	Besoins de propriétés strictes
CO2PC	Validation optimiste Atomicité sémantique Durabilité non garantie	Durabilité relâchée Ressources non négligeables Mobilité pas abordée	Blocage réduit Bonnes performances	Exige un environnement fiable ou un recouvrement relâché
UCM	One phase commit Transfert du journal Atomicité stricte Durabilité garantie	Transfert de journal Contrôle de concurrence stricte Trafic additionnel	Latence réduite Propriétés fortes Bonnes performances	Contrôle de concurrence stricte désirable ou disponible
TCOT	Timeout Transfert du journal Atomicité sémantique Durabilité garantie	Timeout Transfert du journal Optimisme pas toujours applicable Throughput faible	Latence et blocage réduits	Connectivité continue Propriétés relâchées

avec la compensation est intéressant quand des propriétés moins strictes sont préférées à un coût élevé d'exécution et de communication. La performance de l'approche à une phase peut être adoptée quand il est possible d'assurer un contrôle d'accès pessimiste tel que le verrouillage à deux phases strict *(strict two-phase locking)*. L'utilisation de timeout nécessite une évaluation précise. Mais cette évaluation n'est pas simple dans les environnements mobiles car elle est fonction de facteurs variables et instables tels que le délai de transmission des paquets, la fréquence de déconnexion et des paramètres de mobilité (vitesse, temps de pause, distance de la SB, etc.). Il peut être nécessaire d'utiliser des timeouts comme mécanisme additionnel pour empêcher ou limiter les situations de blocage comme dans les protocoles 2PC et CO2PC mais pas comme mécanisme de base (comme dans TCOT).

Chacune des approches a ses propres mérites et inconvénients et est appropriée pour une classe de contexte d'exécution ou besoin d'applications en termes de propriétés transactionnelles, de qualité de service et de coût (table 5.7). Comparé aux autres protocoles étudiés, le protocole 2PC ne réagit pas aussi mal qu'on pouvait s'y attendre. Cependant, il reste coûteux en termes de messages sans fil et par conséquent, de consommation d'énergie. Il est

important de noter, que les approches étudiées présentent la caractéristique d'être complémentaires. Ceci nous a permis de concevoir une solution pour assurer *l'atomicité adaptable* à la variabilité des besoins et des caractéristiques de l'environnement. Nous avons donc combiné les avantages de ces différents protocoles et pris en compte les conclusions tirées des résultats des simulations dans un protocole de validation adaptable. Dans le reste de la thèse nous nous consacrons à cette approche.

Chapitre VI

Modèle et protocole transactionnels adaptables

They always say that time changes things, but you actually have to change them yourself.
---Andy Warhol

Nothing is as practical as a good theory.
--Albert Einstein

Afin de surmonter les problèmes de l'environnement de calcul mobile, de nombreux modèles de transaction et techniques pour les environnements mobiles ont été suggérés. Cependant, ces propositions ont des limitations. Une des limitations principales est le support restreint du changement dynamique de ces environnements. En effet, en dépit de leur capacité à définir des modèles ou des protocoles de validation de transaction pour des caractéristiques spécifiques telles que les déconnexions, ils manquent généralement de capacité de faire des ajustements et des modifications durant l'exécution. De plus, ces modèles ciblent en général des applications spécifiques exigeant des propriétés transactionnelles particulières. Or, ces propriétés peuvent varier d'une application à une autre et peuvent aussi varier durant la vie d'une même application ou en fonction de l'environnement d'exécution.

Dans ce chapitre, nous proposons une approche flexible pour le traitement transactionnel à travers un modèle de transaction et un protocole de validation adaptables. Ces deux propositions ont pour objectif d'offrir un support pour une adaptation dynamique de l'exécution de transaction tenant compte des variations des caractéristiques de l'environnement mobile et des variations des besoins en termes de propriétés transactionnelles des applications.

Après avoir exposé les motivations pour le besoin d'adaptation des modèles et techniques transactionnelles en section 6.1, les sections 6.2 et 6.3 présentent le modèle de transaction Adapt *(Adaptable Transaction Model)*, ses propriétés et sa formalisation et la section 6.4 présente le protocole de validation adaptable aTCP *(Adaptable Transaction Commit Protocol)* et ses propriétés.

6.1 Motivation pour l'adaptation dynamique

Dans cette section, nous donons plus de détails sur les principales motivations à la conception de techniques transactionnelles adaptables à savoir la variabilité des contextes d'exécution et la variabilité des besoins applicatifs. Ces deux formes de variabilité et de dynamisme rendent difficile la prévision statique dès la conception de tous les scénarios

pouvant se présenter durant l'exécution d'une application transactionnelle ; d'où le besoin de techniques dynamiquement adaptables.

Peu de modèles de transactions mobiles ont été mis en application dans les systèmes disponibles dans le commerce. Une des raisons peut être la variation des exigences en termes de propriétés pour des transactions mobiles comme illustrée dans les exemples ci-dessus. Notre proposition pour un système transactionnel flexible tient compte d'un usage dynamique des différentes notions de propriétés par l'adaptation durant l'exécution. Une telle approche permettra aux programmeurs d'applications d'utiliser les avantages du traitement transactionnel tolérant aux pannes en fonction des exigences spécifiques à l'application et des propriétés environnementales.

6.1.1 Besoin de context-awareness et d'adaptation aux changements

A la différence des systèmes répartis fixes, les systèmes mobiles s'exécutent dans un contexte extrêmement dynamique. Nous définissons le *contexte* en tant que tout ce qui peut influencer le comportement d'une application, des ressources matérielles, telles que la mémoire, la puissance de batterie, la taille d'écran, et la capacité de traitement, aux ressources externes, tel que le lieu, le service distant, la largeur de bande passante disponible et la latence du réseau. Cependant, le contexte peut inclure des informations d'état sur l'environnement d'exécution de l'application et aussi de l'utilisateur [BD07] (voire aussi chapitre 2, section 2.4).

L'architecture des réseaux mobiles changent très considérablement des réseaux à infrastructure dans lesquels les noeuds mobiles utilisent les services offerts par un réseau backbone filaire, aux réseaux ad hoc mobiles qui sont habituellement complètement sans fil et non structurés avec les services qui peuvent seulement être offerts par d'autres noeuds mobiles accessibles agissant également comme des routeurs pour les messages destinés aux autres noeuds. Il y a d'autres combinaisons hétérogènes possibles, des systèmes hybrides, où la communication est basée sur des technologies ad hoc et à infrastructure.

L'environnement d'exécution d'une unité portable est extrêmement dynamique et sujet à des changements rapides et imprévisibles: la localisation n'est plus fixée et influence les services et les nœuds accessible ainsi que la qualité de connexion réseau. La performance des réseaux sans fil change considérablement selon les protocoles et les technologies employés ; WaveLan, GSM ou d'autres technologies. La largeur de bande peut chuter soudainement de bonne ou raisonnable à zéro et la connexion peut être perdue. Les déconnexions ne peuvent plus être considérés comme une exception, mais deviennent plutôt un comportement normal des communications sans fil.

En raison de la nature statique de leur contexte, les systèmes traditionnels ont adopté l'approche *transparente* qui cache les rares changements qui se produisent de sorte que les développeurs n'aient pas à les traiter explicitement. Cependant, l'approche transparente ne peut pas être adoptée en présence des niveaux élevés de l'hétérogénéité et de dynamisme intrinsèques des environnements mobiles. En outre, les applications peuvent disposer d'informations utiles qui pourraient permettre une exécution plus efficace. Par exemple, en raison des déconnexions fréquentes, une copie de données ne peut pas toujours être maintenue synchronisée ; si l'accès à une copie est soudainement perdu, la connaissance de l'application devrait être exploitée pour décider laquelle des copies disponibles utiliser (par exemple, une copie synchronisée peut ne pas être préférée à une copie non à jour, si elle est accessible seulement par un lien de réseau de qualité inférieure). Une application ou l'utilisateur peut préférer écarter certaines mises à jour faites sur des objets spécifiques afin d'accomplir une transaction plutôt que l'annuler. Les applications mobiles doivent donc se rendre compte des

changements se produisant dans leur contexte d'exécution, et s'adapter à ces changements, pour continuer à fournir un haut degré de qualité de service à leurs utilisateurs. Ceci a donné naissance à l'approche context-aware.

6.1.2 Besoin d'adaptation pour satisfaire les exigences variables des applications

Les caractéristiques des applications avancées changent beaucoup et le concept de transaction doit s'adapter aux exigences transactionnelles variables comprenant des contraintes de temps, de flexibilité d'exécution et l'exécution de transactions mobiles. Les propriétés ACID traditionnelles sont trop restrictives et peuvent ne pas répondre à ces exigences. Ces dernières peuvent changer entre les transactions de différentes applications ou de la même application. Elles peuvent également changer pendant la vie d'une application, et par conséquent ne seront pas entièrement connues au moment de sa conception. Nous donnons ici quelques exemples pour illustrer différents domaines d'application dont le workflow, le e-commerce, les systèmes d'information médicaux, le travail coopératif.

Le scénario d'arrangement de voyage est un exemple bien connu d'une transaction longue qui exige des propriétés extra ACID. L'utilisateur demande une reservation de chambre d'hôtel (T1), un vol (T2), une voiture (T3), une table de restaurant (T4), et un billet de théâtre (T5). Il peut également vouloir indiquer les hôtels et les restaurants potentiels. Les réservations ne sont pas d'égale importance, et une table de restaurant peut par exemple être omise de la transaction. Les résultats intermédiaires de quelques tâches (sous-transactions T1-T5) peuvent être validés et rendus visibles dès leur validation. Valider des résultats partiels exige l'exécution de transactions compensatrices en cas de l'annulation du voyage.

Dans les systèmes d'information médicaux, les informations médicales de différents types sont stockées sur un certain nombre de sites. Les différentes classes d'utilisateurs peuvent exécuter des transactions de différentes complexités et avoir des exigences variables. Les transactions simples peuvent seulement lire ou mettre à jour un journal patient unique, des transactions plus complexes peuvent employer des données de différentes sources ou des calculs longs où on a, par exemple, besoin de sources de données statistiques distribuées. En cas d'urgence des transactions temps réelles pourraient se contenter de données incomplètes ou temporairement incohérentes pour satisfaire l'utilisateur. L'utilisateur peut avoir besoin d'information au sujet des hôpitaux les plus proches disposant d'assez de personnel de santé disponible pour traiter un patient blessé. Ici l'information doit donner seulement une indication correcte des faits. Les transactions mobiles peuvent être nécessaires si une infirmière effectuant des visites à domicile veut accéder à l'information du patient à partir de son unité portable tout en se déplaçant vers son prochain patient. Les transactions peuvent également exiger l'accès en temps réel à des informations multimédia telles que des radiographies ou des rapports vocaux; ceci implique une garantie de la qualité de service.

Le partage d'informations dans une conférence universitaire est un autre exemple d'exécution de transaction dans les environnements mobiles. Les participants et les organisateurs de la conférence rencontrent et partagent l'information, traditionnellement par des présentations et des discussions informelles. Un système d'information mobile facilitera d'autres possibilités pour le partage d'informations, et l'utilisation des services transactionnels permettra à des programmeurs et à des utilisateurs d'applications de fixer un niveau approprié de correction. Les participants peuvent utiliser leur PDAs et ordinateurs portables pour rechercher l'information des organisateurs de conférence et des autres participants et peuvent partager également leurs propres informations (par exemple, des informations sur eux-mêmes, leurs projets de recherche, employeurs et une sélection de leurs articles). Les organisateurs de conférence fourniront le chargement des proceedings de la conférence, des présentations de

diapositives, des tableaux d'affichage, d'autres informations sur les produits des expositions et la vente de billets pour des événements de la conférence.

Dans un tel système d'information, nous trouverons des transactions longues et courtes, certaines avec les propriétés ACID strictes et certaines avec des propriétés moins strictes. Un exemple d'une longue transaction avec les propriétés relâchées est quand un utilisateur télécharge toute l'information disponible sur un thème spécifique proposée soit par les organisateurs, soit par les participants. Cette transaction pourrait durer assez longtemps pour que l'utilisateur se déplace au delà de la zone de couverture du réseau sans fil et perdre la connexion. Quand l'utilisateur est déconnecté, la transaction continue d'une manière transparente. Ce genre de transaction n'a pas besoin d'exiger l'atomicité stricte, mais l'utilisateur doit être notifié si la transaction s'est exécutée avec succès ou pas.

Dans d'autres circonstances un participant de la conférence peut exiger que les transactions chargées de récolter des informations sur lui et ses projets soient atomiques. De cette façon l'utilisateur peut s'assurer que toutes les informations qui le concernent sont complètes. Les transactions impliquant des paiements comme, par exemple, l'achat des billets pour un concert, exigeront les propriétés ACID complètes.

6.2 Un modèle de transaction flexible

Les nombreux modèles de transaction existants dans la littérature pour l'environnement mobile n'ont pas pris en charge les modes d'exécution de transactions distribuées (modes 3, 4, 5 et 6); mis à part une ou deux propositions. Or ces modes d'exécution nécessitent la révision des protocoles distribués qui permettent d'assurer les propriétés ACID; les protocoles de validation et les protocoles de contrôle de concurrence. Ceci nous a conduit à nous fixer comme objectif d'étudier les problèmes de validation atomique pour l'environnement mobile (chapitre 3, 4 et 5). Les problèmes de contrôle de concurrence n'ont été considérés que succintement dans cette thèse.

D'un autre côté, l'étude des modèles de transactions avancés ou étendus et leurs propriétés ainsi que les modèles mobiles nous a permis de constater que:

- La flexibilité nécessite la décomposition d'une transaction en un ensemble de sous-transactions composantes. Cette décomposition permet de relâcher les propriétés en ayant la possibilité d'associer des propriétés différentes à chaque sous transaction et à la transaction globale.
- L'adaptabilité à la variabilité du contexte d'exécution et des besoins transactionnels ne peut être obtenue sans passer par l'adaptabilité des propriétés ACID et des techniques qui leur sont associées.
- Différents types de sous-transactions ont été proposés dans la littérature (Compensable, non-compensable, contingente ou alternative, vitale ou non-vitale, réexecutable ou non) dans le but d'offrir plus de flexibilité et de pouvoir relâcher les propriétés ACID. Dans la section 6.2.1 qui suit, nous montrons comment cela se concrétise pour la propriété d'atomicité.

Alors nous avons poursuivi notre recherche en essayant de proposer un protocole (section 6.4) de validation atomique adaptable. Cette recherche nous a conduit à proposer un protocole de validation et un modèle de transactions adaptables qui sont basés sur un assemblage de différentes notions existantes dans la littérature augmenté de mécanismes capables de prendre en charge l'environnement mobile.

Le modèle adaptable que nous proposons a pour objectifs de permettre :

- l'adaptabilité aux variations de l'environnement mobile afin d'assurer le meilleur coût d'exécution possible,

- l'adaptabilité en acceptant les différents modèles d'exécution des transactions mobiles et en particulier les modèles d'exécution distribués (modèles 1-6, voir chapitre 1),
- la satisfaction des besoins applicatifs variés en offrant un support pour les propriétés ACID ainsi qu'à toute une gamme de propriétés avec différents "degrés de relâchement" de ces propriétés,
- de limiter les effets des déconnexions,
- de limiter le blocage,
- de prendre en compte la mobilité.

6.2.1 Notions d'atomicité flexible

Afin de satisfaire des exigences transactionnelles variables, la nécessité de relâcher les propriétés ACID a été proposée dans de nombreux travaux de recherche depuis le début des années 90s. Il y a eu un grand effort sur les modèles de transaction avancés et étendus [Elm92]. Cet effort a été poursuivi plus récemment dans le contexte du calcul mobile [SRA04] pour satisfaire des contraintes environnementales. Les recherches ont conduit à différentes notions d'atomicité (atomicité stricte et atomicité relâchée ou sémantique), de cohérence (stricte ou faible), d'isolation (stricte ou relâchée en permettant un entrelacement flexible entre transactions et un partage contrôlé des résultats intermédiaires), etc.

L'a*tomicité* standard, parfois appelée "atomicité stricte" *(strict atomicity)*, "atomicité complète" *(full atomicity)* ou *(failure atomicity)* fait référence au fait qu'une transaction doit être exécutée entièrement avec succès ou doit apparaître comme si qu'elle ne s'est jamais exécutée – c'est-à-dire, "*tout ou rien"*. Comme nous l'avons vu dans les chapitres précédents, le mécanisme communément utilisé pour assurer l'atomicité stricte dans les systèmes distribués est le protocole de validation à deux phases (*two-phase commit* (2PC) *protocol*). L'atomicité relâchée a été introduite par les modèles de transaction imbriqués ouverts *(open-nested model)* proposés généralement dans le contexte des systèmes multibases. L'atomicité sémantique peut être assurée grâce au concept de *compensation*. Les sous-transactions compensables peuvent être validées de façon unilatérale indépendamment de la transaction globale. L'ACIDité avec l'atomicité sémantique peut être préservée grâce à l'approche optimiste *Optimistic 2PC* (O2PC) [LKS91]. Le principe du protocole O2PC est le même que celui du 2PC (chapitre 3) jusqu'à l'envoi du message *prepare* à tous les participants. A la différence de 2PC, lorsque le message est reçu par les participants, ces derniers tentent de valider leurs sous-transactions de façon optimiste. Le résultat est reporté au coordinateur. Si toutes les sous-transactions sont validées, alors le coordinateur décide la validation globale, sinon, il décide l'annulation. En cas d'annulation, des transactions de compensation sont exécutées pour toutes les sous-transactions qui ont été validées.

On peut obtenir plus de flexibilité et de résistance aux défaillances, telle que la connectivité intermittente de l'environnement mobile, par l'introduction de transactions *alternatives* ou *de contingence*. Une transaction alternative peut être invoquée si la transaction pour laquelle elle est associée n'est pas validée. La "semi-atomicité" a été proposée comme critère de correction des transactions flexibles qui permet à une transaction globale d'être validée si ses sous-transactions *primaires* ou ses sous-transactions *alternatives* sont validées [Elm92, ZNBB94]. Une autre notion d'atomicité relâchée permet de valider une transaction même si certaines de ses sous-transactions ne sont pas validées. Ces sous-transactions sont dites *non vitales* [Chr93]. Les sous-transactions primaires ou alternatives peuvent être vitales ou non vitales. Toutes les composantes vitales doivent être validées pour que la transaction racine ou globale puisse être validée. Par exemple, l'utilisateur/application peut décider quand et quelles parties de la transaction devraient être validées (les parties vitales) même si d'autres parties (non vitales) ne peuvent être validées du fait, par exemple, que leurs sites d'exécution

sont déconnectés. Ces notions peuvent être étendue en permettant à une sous-transaction d'être tentée un certain nombre de fois en cas d'échec de la première exécution; c'est la notion de transaction ré-exécutable *(retriable)* [BGS92]. Une transaction est ré-exécutable si elle peut éventuellement être validée après avoir été tentée un certains nombre de fois et ce sur un état quelconque de la base de données. Par exemple, une transaction de débit d'un compte bancaire n'est pas ré-executable, alors qu'une transaction de crédit l'est.

6.2.2 Structure des transactions adaptables (AdapT)

En prenant en considération les notions présentées ci-dessus, nous définissons une transaction AdapT comme une transaction globale structurée en un arbre de transactions composantes ou sous-transactions suivant le modèle imbriqué ouvert où l'accès aux données se fait uniquement par les feuilles. Les sous-transactions peuvent avoir chacune des transactions *alternatives* ou *contingentes*; dans ce cas elles sont dites *remplaçables.* Une transaction alternative n'est exécutée que si la transaction principale à laquelle elle est associée n'est pas validée. Les sous-transactions ainsi que leurs alternatives peuvent ou non avoir des transactions de compensation. Elles sont alors soit *compensables*, soit *non compensables*. Les sous-transactions compensables peuvent être validées indépendamment et avant la transaction racine ou globale. Leurs ressources sont alors libérées plutôt et leurs résultats sont rendus visibles à d'autres transactions.

Dans l'exemple de la figure 6.1, T est la transaction globale. Elle est composée des sous-transactions *principales* {T1, T2, T3, T4, T5, T6, T7, T8}, des sous-transactions *secondaires* suivantes: alT1 qui est l'alternative de T1; alT1 étant compensable par calT1 et T1 par cT1, cT4 est la transaction de compensation de T4, alT6 est l'alternative de T6, cT3 est la transaction de compensation de T3 et enfin alT8 qui est l'alternative de T8.

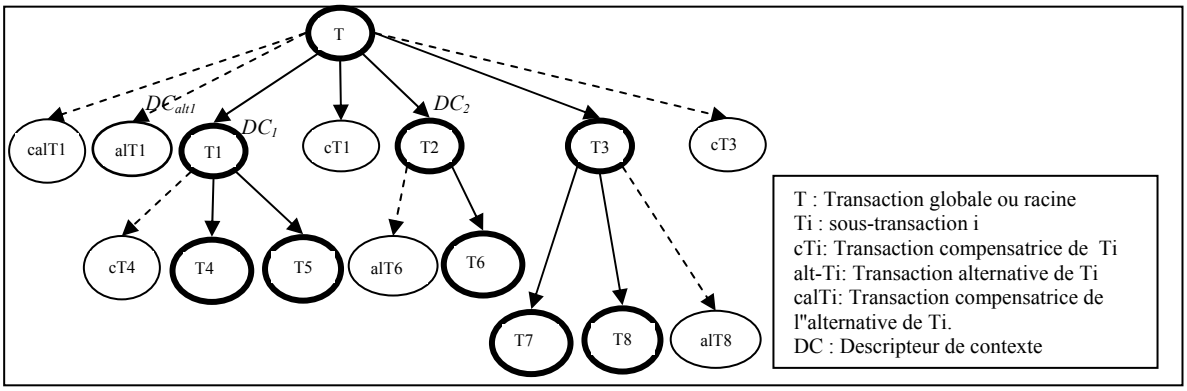


Figure 6.1-Structure d'arbre d'une transaction globale

Toutes les sous-transactions principales et leurs alternatives peuvent être *vitales* ou *non vitales* [Chr93]. Les transactions de compensation (ou compensatrices) qui sont aussi des transactions secondaires doivent être toujours vitales car la compensation n'est initiée qu'en cas de non succès de la transaction globale. La transaction globale ne peut être validée que si toutes ses sous-transactions vitales sont validées. Il suffit qu'une seule transaction vitale ne soit pas validée pour que la transasction globale ne soit pas validée.

Toutes les sous-transactions principales ou secondaires qui sont re-exécutables en cas de panne peuvent être déclenchées un certain nombre de fois, après un certain délais d'attente entre les tentatives ou lorsqu'une certaine condition ou événement lié à un changement du contexte (connexion d'un nœud, augmentation de la bande passante, etc.) se produit.

Une application d'un tel modèle de transaction peut être illustrée par le scénario de planification d'un voyage pour assister à une conférence. Il s'agit de faire une réservation

d'une place et l'achat d'un billet d'avion avec une préférence sur des conditions spécifiques (compagnie de préférence, l'horaire du vol, tarif), de réserver une chambre d'hôtel (distance par rapport au lieu de la conférence, tarif) et éventuellement une place de restaurant (proche de l'hôtel si possible). On peut structurer une telle application en sous-transactions dont l'une sera associée à l'organisation du vol et l'autre à l'hôtel, l'autre au restaurant. Chacune des sous-transactions peut faire appel à des alternatives; par exemple, le choix entre plusieurs compagnies et tarifs, de même pour les hôtels et le restaurant. Le vol et la chambre sont des réservations vitales. La validation de la sous-transaction d'achat de billet peut se faire avant la conclusion des autres réservations; grâce à la compensation il sera possible d'annuler le vol si en fin de compte la transaction globale n'est pas validée. On peut aussi donner la possibilité à l'utilisateur de changer d'avis sur l'importance de réserver une chambre d'hôtel à l'avance et de se contenter de seulement acheter un billet et de régler le reste des tâches sur place, etc.

On voit bien que seul un modèle possédant les propriétés indiquées ci-dessus est capable d'offrir une telle flexibilité. Dans les sections qui suivent nous définissons plus précisément la structure des transactions AdapT en mettant l'accent sur les aspects liés au contexte d'exécution mobile. Nous montrons dans la section 6.2.3 comment le modèle proposé est capable de prendre en charge les caractéristiques de l'environnement de calcul mobile et sa variabilité, avant de préciser les propriétés du modèle AdapT en 6.2.4 et les critères de corrections en 6.2.5. La spécification formellement est donnée en 6.3. Evidemment un tel modèle doit être supporté par des techniques dont la validation atomique flexible que nous aborderons dans la section 6.4.

6.2.3 Adaptabilité en environnement sans fil et mobile

L'aspect adaptation au contexte d'exécution mobile est obtenu en associant à chaque transaction principale et alternative un *descripteur de contexte (DC)*. Le descripteur DC_k décrit le contexte particulier ou les conditions nécessaires à l'exécution d'une sous-transaction T_k qu'elle soit principale ou alternative. Les alternatives peuvent être sémantiquement équivalentes et seule l'exécution de l'une d'entre elles doit être validée. Cependant, chaque alternative possède son propre DC_k décrivant par exemple l'état connecté/déconnecté, le débit, la batterie disponible, etc. Nous verrons par la suite que, de manière générale, le descripteur de contexte peut contenir un large éventail d'information selon le type de contexte par lequel on est intéressé (section 6.2.3.2).

A chaque sous-transaction T_k est associé également le site où elle doit s'exécuter *(SE_x)*; ce site pouvant être une UM ou une UF. Le site d'exécution d'une sous-transaction est le même que celui de sa transaction de compensation.

Grâce à ce modèle nous pouvons avoir plusieurs formes d'adaptation:

- concernant les modèles d'exécution (1 à 6) décrits dans les chapitres précédents, c'est-à-dire, la façon de distribuer l'exécution d'une transaction mobile entre UMs et UFs, on peut prévoir en fonction des conditions de l'environnement décrites dans le DC et le site d'exécution, plusieurs modèles grâce à la notion d'alternatives. Par exemple, une alternative s'exécutant sur l'UM dans le cas où la bande passante est faible et une copie des données est disponible sur le cache, l'autre alternative sémantiquement équivalente s'exécutant sur l'UF si une bonne connexion est offerte et à un prix intéressant.
- On peut également s'adapter aux conditions courantes par la possibilité de re-exécuter des sous-transactions lorsque certaines conditions sont vérifiées. Par exemple, après un échec d'une sous-transaction dû à l'indisponibilité d'un participant qui s'est déconnecté, on peut la re-executer dès que l'on détecte que se dernier s'est reconnecté au réseau.

- On peut aussi adapter les propriétés ACID, des plus strictes aux plus relâchées, comme on le verra plus en détails dans ce qui suit. Par exemple, concernant l'atomicité, il est possible d'assurer une atomicité stricte ou une atomicité sémantique ou relâchée. En effet, une application peut demander un degré d'atomicité stricte si le coût d'exécution ne lui revient pas cher et se contenter d'une atomicité sémantique autrement. Deux applications différentes peuvent exiger des degrés d'atomicité différents, etc. Ceci répond au besoin d'utiliser le modèle pour une variété de propriétés transactionnelles pour une même application ou plusieurs types d'applications mobiles.

6.2.3.1 La flexibilité du modèle AdapT

Nous proposons la définition d'une transaction adaptable suivante:

Définition 6.1. *Une transaction AdapT est une transaction* $T_{AdapT} = \{(T_i , Role_i, comp\text{-}T_i , alt\text{-}T_i , DC_i , i\geq 0, Siteid_x)\}$ *où:*

- T_i est une sous-transaction.
- $Role_i=\{(r_1, r_2, r_3, r_4)\}$ spécifie le rôle de la sous-transaction T_i avec*:*

$r_1 \in \{Compensable, non\text{-}compensable\}$, $r_2 \in \{remplaçable, non\text{-}remplaçable\}$,

$r_3 \in \{réexécutable, non\text{-}réexécutable\}$, $r_4 \in \{vitale, non\text{-}vitale\}$ et $comp\text{-}T_i$ *est la transaction de compensation si* T_i *est compensable.*

- $Alt\text{-}T_i = \{(alt\text{-}T_{ik} , comp\text{-}alt\text{-}lT_{ik}, DC_i , Siteid_x , k\geq 0\}$ représente la liste des transactions alternatives qui peuvent substituer la sous-transaction principale T_i avec:

$Alt\text{-}T_{ik}$ *, la kième alternative de* T_i *,* $comp\text{-}alt\text{-}T_{ik}$ *la transaction de compensation de* $alt\text{-}T_{ik}$ *si cette dernière est compensable, DCi le descripteur de contexte et* $Sideid_x$ *le site d'exécution de l'alternative et de sa transaction de compensation.*

- DC_i est un descripteur de contexte qui permet à la sous-transaction de s'exécuter d'une manière adaptée à l'état courant de l'environnement. Sur la figure 6.1, on ne représente que quelques exemples de DCs afin de ne pas surcharger le schéma.
- $Siteid_x$ est l'identité du site sur lequel la sous-transaction et sa transaction de compensation doivent être exécutées.
- Il existe une relation de dépendance $\mathcal{RD}$ entre les sous-transactions d'une même transaction globale.

$\forall (T_i , R_i, cT_i , AT_i , DC_i ,SE_x), (T_j , R_j , cT_j , AT_j , DC_j ,SE_y) \in T_{Adapt}$
$(T_i , R_i, cT_i , AT_i , DC_i ,SE_x)\ \mathcal{RD}\ (T_j , R_j , cT_j , AT_j , DC_j ,SEz)$

$\mathcal{RD}$ est une relation de dépendance de parallélisme ($\|$) ou de séquentialité ($<$).

La compensation permet aux UMs de disposer d'une certaine autonomie par le relâchement de l'atomicité grâce à la validation anticipée des sous-transactions compensables.

Lorsqu'une transaction T_{AdapT} est initiée et sa décomposition et l'identification des sous-transactions effectuée, l'état de l'environnement est alors examiné afin de décider du choix des composantes à exécuter en comparant l'état courant avec l'état spécifié dans les descripteurs de contexte. Une alternative (la plus prioritaire possible) dont le DC est satisfait doit être déclenchée pour les sous-transactions remplaçables. Si le contexte courant ne permet pas l'exécution d'une sous-transation alors la suite de l'exécution sera décidée selon le rôle de cette dernière:
-elle est vitale, reporter l'exécution de la T_{AdapT}, jusqu'à ce que les conditions le permettent.

-elle n'est pas vitale, lancer l'exécution de la T_{AdapT} et relancer la sous-transaction bloquée dès que les conditions le permettent.

En cours d'exécution, si le contexte change causant l'annulation d'une sous-transaction en cours alors selon son rôle:
-elle est vitale, non remplaçable et non re-exécutable, la T_{AdapT} doit être abandonnée.
-elle est vitale et remplaçable, réévaluer la possibilité d'exécuter une alternative.
-elle est vitale et re-exécutable, évaluer les paramètres de re-exécution et la re-exécuter.
-elle est non vitale, il n' y a pas d'incidence sur l'exécution du reste de la T_{AdapT}, la sous-transaction défaillante est juste ignorée.

Afin d'offrir l'adaptation dynamique du degrè d'atomicité, nous incluons la possibilité de modifier le rôle des sous-transactions. En effet, en cours d'exécution, si une sous-transaction doit être annulée, on peut permettre à l'utilisateur de modifier son rôle. Avant de décider, selon l'algorithme ci-dessus, de la façon de continuer T_{AdapT} suite à l'échec d'une sous-transaction, il y a différentes possibilités de changement de rôle:

-elle est remplaçable et des modifications peuvent porter sur la liste de ses alternatives (ajout, retrait ou modification).
-elle est vitale et elle devient non vitale (si une sous-transaction vitale et ses alternatives ne peuvent être validées alors elle peut être ignorée).

-elle est re-exécutable et elle ne l'est plus ou vice versa, les paramètres de réactivation sont modifiés (le délai d'attente avant la réactivation, la condition déclenchante ou le nombre de tentatives possibles). On peut décider de retenter une transaction de réservation d'un taxi même si au départ elle n'était pas programmée pour être tentée plusieurs fois.

Nous donnerons plus de détails sur la façon d'offrir cette flexibilité dans la section 6.4 consacrée au protocole de validation atomique, et aussi dans le chapitre 7 consacré à la mise en œuvre.

6.2.3.2 Le descripteur de contexte

Un *descripteur de contexte* fait référence à l'état, aux ressources, et aux conditions de l'environnement d'exécution de l'application et de l'utilisateur. Cette définition de contexte est large afin d'inclure toute caractéristique de l'utilisateur, de l'application, et de l'environnement qui peut être mesurée et interrogée. De manière générale, le descripteur de contexte est une représentation des caractéristiques variables de l'environnement qui peuvent affecter l'exécution d'une transaction.

Les caractéristiques de l'environnement mobile dépendent du réseau sans fil utilisé, des unités mobiles et fixes et de la mobilité de l'utilisateur:

- débit de communication,
- prix de communication,
- connexion/déconnexion,
- temps de connexion estimé,
- taille de la mémoire et du cache,
- capacité de calcul,
- capacité de la batterie
- localité
- etc.

Chacune de ses caractéristiques peut être définie par un état exprimé en niveaux de qualité. Par exemple, (forte, moyen, faible) pour la bande passante, (élevé, modéré, gratuit)

pour le prix de communication, (plaine, moitié, faible) pour la batterie, etc. Ces états peuvent être aussi traduits par des intervalles (le prix de communication est modéré s'il se situe dans [1,2] DA la minute ou d'autre forme comme par exemple (le taux de déconnexion est de 30%).

Définition 6.2. *Un Descripteur de Contexte (DC) est un ensemble de caractéristiques avec leurs états à un instant donné.*

DC= {caractéristique= {état(s)}}

Une transaction nécessitant une bonne connexion à un coût raisonnable peut avoir un

DC= {état-connexion=connecté, bande-passante=forte, coût-communication= {gratuit, modéré}}

Parmi les informations du contexte utilisateur, la localité peut être considéré par exemple pour des applications dépendantes de la localisation. Le contexte de l'application peut inclure la qualité des données requises (leur fraîcheur ou degré de cohérence (exiger l'accès à la copie principale ou la copie du cache)).

6.2.4 Propriétés des transactions AdapT

Chaque sous-transaction composante d'une transaction AdapT s'exécute entièrement sur un système de gestion de données sous-jacent. Elle possède les propriétés AID; la cohérence est préservée en respectant les contraintes d'intégrité qui sont dépendantes de l'application. En effet, la cohérence signifie que la base de données satisfait toutes ses contraintes d'intégrité. Cependant, contrairement aux propriétés AID, la cohérence est de la responsabilité partagée entre le système de gestion de transactions et les programmes d'application. Le système assure la cohérence en garantissant les propriétés AID, mais c'est au développeur de l'application de veiller à ce que cette dernière la préserve.

Avant de présenter les propriétés pour une transaction globale T_{AdapT} nous définissons différentes notions ou degré d'atomicité en nous basant sur l'analyse réalisée dans les sections et chapitres précédents.

- **Atomicité Stricte** exige que toutes les sous-transactions votent pour la validation avant la validation de la transaction globale. Cette dernière ne doit consister qu'en des sous-transactions vitales et non compensables. C'est la propriété d'atomicité classique qui nécessite que toutes les transactions composantes soient validées ou aucune ne le soit.
- **Atomicité Sémantique** exige qu'une transaction globale consiste en sous-transactions vitales dont certaines peuvent être compensables. Les sous-transactions non compensables doivent voter pour la validation avant la validation de la transaction globale. Les sous-transactions compensables peuvent être validées avant l'accomplissement de la transaction globale. Dans le cas où cette dernière est abandonnée, les sous-tranasctions de compensation doivent être exécutées pour annuler sémantiquement les effets des transactions ayant été validées unilatéralement. Donc, toutes les sous-transactions sont soit validées, soit compensées.
- **Atomicité Relâchée** est obtenue lorsque la transaction globale consiste en une combinaison quelconque de sous-transactions vitales ou non vitales, compensables ou non. Si une ou plusieurs sous-transactions non vitales sont annulées, la transaction globale peut comme même être validée. Dans cette définition d'atomicité nous avons inclus une gamme complète d'atomicité sémantique relâchée dans le sens où les sous-transactions non vitales peuvent être annulées sans pour autant empêcher la validation globale.

Afin d'offrir le plus de flexibilité et d'autonomie aux sites participant dans l'exécution des transactions tout en prenant en compte le plus grands nombre d'applications possibles, la

validation d'une transaction globale T_{AdapT} peut avoir les trois types d'atomicité selon la sémantique de l'application et ses exigences en termes de propriétés transactionnelles. Le degré d'atomicité peut être adapté de façon dynamique pendant l'exécution de la transaction grâce à un protocole de validation adaptable qui est présenté dans la section 6.4.

La possibilité de valider d'une façon *optimiste* permet une certaine autonomie grâce à la libération prématurée des ressources. Dans le contexte mobile cette option est très intéressante pour les UMs qui vont pouvoir travailler en mode déconnecté en bloquant le moins possible les ressources. L'utilisation de transactions de compensation comme un moyen de récupération sémantique, conduit au relâchement de l'atomicité pour une atomicité sémantique. Une transaction de compensation doit obligatoirement être validée. Cette contrainte est appelée *persistance de compensation* [Gar83, GS87, KLS90] et permet d'éviter d'utiliser un protocole de validation pour assurer l'atomicité des transactions de compensation. Les transactions de compensation des sous-transactions composantes exécutées séquentiellement doivent s'exécuter dans l'ordre inverse de validation afin qu'elles puissent observer un état consistant avec l'état observé par leurs transactions associées.

Les transactions non compensables ou pour lesquelles la libération des ressources ne peut être faite de façon prématurée doivent subir l'approche *pessimiste* ne libérant les ressources qu'après la validation ou l'abandon de la transaction globale. Dans notre proposition, il est tout à fait possible de combiner les deux approches en acceptant des sous-transactions compensables et d'autres non compensables sous la même transaction globale.

En plus de ces deux options et leur combinaison, la notion de vitalité ou non vitalité qualifiant les sous-transactions, qu'elles soient compensables ou non, offre plus de flexibilité. En effet, le fait de pouvoir ignorer l'échec de certaines sous-transactions, qualifiées de non vitales, permet de réduire le nombre d'annulation de transactions globales en se contentant des sous-transactions vitales qui présentent une importance pour l'application ou l'utilisateur. Si l'on reste dans le cas de l'environnement mobile, cette approche permet par exemple, d'ignorer des sous-transactions non vitales s'exécutant sur des UMs qui se sont déconnectées ou ayant échouée à cause d'autres paramètres de l'environnement. Ceci permet de réduire les coûts d'exécution car on peut au moins valider la partie du travail réalisé par les sous-transactions vitales et qui sont de la plus haute importance pour l'utilisateur ou l'application en particulier lorsque la transaction est de longue durée de vie.

Afin d'assurer les propriétés de Cohérence et d'Isolation, des techniques de contrôle de concurrence sont employées. Dans le cas des transactions AdapT, le critère de correction est fourni à travers la *sérialisation globale* [BRS92, BHG87, Pap86]. La sérialisation globale nécessite que :

- les sous-transactions soient sérialisables localement, et
- les sous-tranasctions en conflit appartenant à deux transactions globales soient exécutées dans le même ordre au niveau de tous les sites.

Dans la section 6.2.5 nous donnons plus d'explications sur la sérialisation globale.

Concernant la Durabilité d'une transaction AdapT, elle est préservée du moment qu'une fois validée elle n'est plus annulée ou défaite. Par contre au niveau des sous-transactions compensables la durabilité n'est pas préservée dans le cas ou la transaction globale n'est pas validée. La durabilité des sous-transactions de compensation doit être préservée. La table 6.1 récapitule les propriétés de notre modèle.

Table 6.1-Propriétés du modèle AdapT

Propriétés	Sous-transaction	Transaction globale
Atomicité	√	Adaptable (stricte, sémantique, relâchée)
Cohérence	√ conditionnée	Adaptable (stricte, sémantique)
Isolation	√	Adaptable (stricte, relâchée)
Durabilité	√ conditionnée	√
Critère de correction	Sérialisabilité	Sérialisabilité globale

6.2.5 Critères de correction des transactions AdapT

La sérialisabilité est la théorie de correction de référence pour l'exécution concurrente de transactions dans les bases de données [BHG87]. Dans les systèmes répartis, on parle de sérialisabilité globale. Ainsi, un ordonnancement est globalement sérialisable si l'ordre d'exécution des sous-transactions suit le même ordre sérialisable dans chaque SGBD local ainsi que globalement. Nous avons deux grandes classes de SGBDs répartis ; des systèmes homogènes ou des systèmes hétérogènes. La distinction entre ces deux classes peut être ramenée à deux différences principales [WV02]:

- Conceptuellement, l'autonomie des systèmes hétérogènes est une forte exigence car tous les protocoles de contrôle de concurrence et de recouvrement doivent être conçus sans compter sur la collaboration avec les systèmes sous-jacents. Dans le cas homogène, il suffit d'adapter les techniques déjà existantes au cas distribué. La *sérialisabilité globale* revient à appliquer par exemple, une technique de verrouillage répartie et le protocole 2PC pour la validation. Dans le cas hétérogène la *sérialisabilité globale* s'avère plus compliquée. L'un des problèmes rencontrés est le non partage de l'information concernant la gestion locale et donc les ordonnancements locaux sont hors du contrôle global. La sérialisabilité et la validation de ce type de sytèmes ont été traitées dans de nombreux travaux [BGS92, WV02]. Si l'on ajoute à cela la dimension mobile et sans fil, comme on l'a vu à travers cette thèse pour la validation, les choses se compliquent.

-Techniquement, cela revient à prendre en compte la présence de deux types de transactions; les transactions locales et les transactions globales. Dans le cas homogène, il n' y a que des transactions globales (une transaction classique s'exécutant de façon répartie), ce qui évite les interférences (en termes d'ordonnancement) avec les transactions locales pouvant être à l'origine de *conflits indirects*[26] entre des transactions globales [BGS92, WV02].

Exemple: Considérons un système multibases localisé sur deux sites : *s1* avec les articles de données *a* et *b, s2* avec les articles de données *c* et *d.* T_1 et T_2 sont des transactions globales de lecture.

$$T_1: r_1(a)\ r_1(c)$$

$$T_2 : r_2(b)\ r_2(d)$$

De plus, soient T3 et T4 deux transactions locales aux sites s1 et s2, respectivement, définies comme suit :

$$T_3 : w_3(a)\ w_3(b)$$

$$T_4 : w_4(c)\ w_4(d)$$

[26] *Les conflits indirects* sont causés par des ordonnancements locaux qui incluent des transactions locales et des sous-transactions composantes de transactions globales. Ces conflits ne peuvent être observés au niveau global.

Supposons que la transaction T_1 soit exécutée et validée au niveau des deux sites et après que T_2 soit exécutée et validée au niveau des deux sites. Une telle exécution peut résulter en les ordonnancements locaux s_1 et s_2 générés sur le site s_1 et s_2 respectivement :

$$S_1: r_1(a)\ c_1\ w_3(a)\ w_3(b)\ c_3\ r_2(b)\ c_2$$

$$S_2: w_4(c)\ r_1(c)\ c_1\ r_2(d)\ c_2\ w_4(d)\ c_4$$

Comme résultat, T1 est sérialisée avant T2 sur le site s1 et après T2 sur le site s2 ; de ce fait, la sérialisabilité globale n'est pas maintenue.

Dans cet exemple, le problème apparaît parce que les transactions locales crèent des conflits indirects entre les transactions globales. Le gestionnaire globale de transactions n'a connaissance ni des transactions locales, ni de ces conflits indirectes.

Les conflits indirects peuvent introduire des cycles dans la relation d'ordre des transactions globales, ce qui affecte la sérialisabilité globale.

Ici, nous considérons le cas des sytèmes hétérogènes car comme nous l'avons vu dans le premier chapitre de cette thèse, les systèmes mobiles sont des systèmes dont le besoin d'autonomie est intrinsèque. En effet, d'une part, les UMs requièrent cette propriété afin de pouvoir travailler même pendant les périodes de déconnexion, d'autre part, le reste du système doit pouvoir poursuivre son exécution indépendamment des UMs qui peuvent être souvent injoignables.

La sérialisation d'une transaction AdapT est fournie à travers celle de ses sous-transactions, cependant, même si la sérialisation globale est préservée, le relâchement de l'atomicité et de l'isolation donne une cohérence sémantique.

Afin de ne pas compromettre la cohérence sémantique lors de l'utilisation des transactions de compensation, nous adoptons les mesures suivantes [BGS92]:

- La validation partielle d'une transaction globale (valider sur un site et annuler sur un autre) peut entraîner la violation des contraintes d'intégrité globales. La cohérence doit être rétablie par les transactions de compensation en préservant les contraintes d'intégrité globales. Mais afin d'éviter que l'incohérence soit perçue par les transactions s'exécutant entre la sous-transaction et sa compensation, les contraintes d'intégrité globales sont interdites. Ceci ne pose pas de problème si nous supposons que les sources de données des systèmes sous jacents sont disjointes.
- Pour que l'exécution des transactions de compensation concerne seulement le site où la transaction compensable est exécutée, les dépendances de valeur avec les sous-transactions compensables d'une même transaction AdapT sont interdites. Sinon, cela voudrait dire qu'une sous-transaction peut avoir une influence sur plusieurs sites et donc que sa compensation doit aussi les toucher. L'interdiction des dépendances de valeur évite les annulations en cascade.
- Les dépendances entre les sous-transactions de la même transaction globale avec les sous-transactions ré-exécutables soient interdites afin qu'aucun résultat concernant la première tentative ne se soit divulguée à d'autres transactions. C'est-à-dire, les valeurs lues par la transaction ré-exécutable ne doivent pas être divulguées.

Dans cette thèse nous avons utilisé la *sérialisabilité globale* comme critère de correction pour laquelle nous avons adopté une approche présentée dans la section 6.5. Cependant, dans cette thèse nous avons concentré nos efforts sur la validation atomique et nous sommes conscients qu'une étude plus approfondie est nécessaire pour traiter les problèmes de contrôle de concurrence en environnement mobile [MA05].

6.2.6 Exemple

L'adaptabilité du modèle AdapT peut être montrée dans l'exemple suivant: soit une application de e-commerce dans laquelle un utilisateur mobile désire acheter des marchandises à partir d'un magasin électronique qui a deux serveurs: un serveur catalogue (Catalogue server, CatS) et un serveur d'achat (Purchase server, PurS). CatS permet à l'utilisateur de sélectionner la marchandise et PurS gère les commandes et le paiement.

Une transaction globale T peut être composée par exemple des sous-transactions suivantes:

-T1: permet à un client d'avoir une copie du catalogue à partir de CatS.
-T2: permet de choisir les articles à partir d'une copie du catalogue.
-T3: permet à un client d'envoyer une commande et de payer sur le serveur PurS.
-T4: permet d'exécuter le processus de paiement sur l'UM en utilisant la maonaie électronique, sans contacter le serveur PurS.
-T5: permet d'envoyer la commande sans inclure le paiement.

Selon le contexte (la connectivité des noeuds, la qualité de la bande passante, le coût des communications, la disponibilité ou non d'une copie du catalogue, etc.), l'exécution de T peut inclure des combinaisons variables de Ti, i=1…5. Par exemple, {T2, T3} peuvent être exécutées si un catalogue à jour existe sur l'UM et une bonne connexion existe entre elle et le serveur PurS. {T1, T2, T3} peuvent être exécutées si la connexion et la bande passante sont bonnes et le coût de communication est modéré. {T1, T2, T4, T5} peuvent être exécutées si la connexion est bonne, la bande passante est faible et une copie du catalogue n'est pas disponible sur l'UM. Mais, si une déconnexion apparaît avant l'exécution de T5, cette dernière peut être re-exécutée plus tard. Comme on peut le voir T2 et T3 sont vitales car T ne doit pas être validée si aucune sélection ou paiement ne sont fait. T4 est une alternative pour T3 et est vitale aussi.

6.3 Spécification formelle du modèle AdapT

Dans cette section, nous proposons une formalisation du modèle AdapT en utilisant le formalisme ACTA[27]. ACTA [CR90, CR94] est un framework développé pour caractériser les interactions qui existent dans les modèles de transactions étendus. ACTA permet de spécifier (1) les effets des transactions sur d'autres transactions et (2) les effets des transactions sur les objets. Dans l'annexe B, nous présentons ACTA afin de faciliter la compréhension de cette section.

6.3.1 Definition axiomatique du modèle

Le modèle de transaction AdapT se compose de transactions imbriquées ouvertes avec la compensation. Une transaction peut être décomposée en n'importe quel niveau d'emboîtement, qui donne une transaction racine et un ensemble de sous-transactions (ou transactions composantes). Ces transactions s'appellent transactions principales. Des transactions secondaires, compensatrices et alternatives, peuvent être associées aux sous-transactions composantes principales. Les transactions de compensation ou compensatrices peuvent également être associées à des transactions alternatives. Les transactions peuvent avoir plusieurs rôles ; elles peuvent être vitales/non-vitales, re-exécutables/non re-exécutables, compensables/non-compensables, et remplaçables/non-remplaçables.

[27] ACTA signifie actions en Latin, il a été choisi par ses concepteurs Chrysanthis et Ramamitham étant donné que ce framework est approprié pour l'expression des propriétés des actions utilisées pour composer un calcul.

Dans ce qui suit nous décrivons les dépendances entre les sous-transactions et la racine (transactions principales) et entre les sous-transactions principales et les sous-transactions secondaires.

Les notations utilisées sont les suivantes:

- T est une transaction mobile adaptable du modèle AdapT, $T \neq T'$
- $T = \{T_1 (r_1, r_2, r_3, r_4), \ldots, T_n (r_1, r_2, r_3, r_4)\}$, $n > 0$
- *Comp-Trs* est l'ensemble des sous-transactions compensables, ¬comp-Trs est l'ensemble des sous-transactions non compensables, $Comp\text{-}Trs \cap \neg comp\text{-}Trs = \varnothing$
- *comp-*T_i est la sous-transaction compensatrice de T_i
- *alt-*$T_i = \{aT_{i1} (r_1, r_2, r_3, r_4), \ldots, aT_{is} (r_1, r_2, r_3, r_4)\}$, $s \geq 0$ est l'ensemble des sous-transactions alternatives d'une T_i
- t dénote T_i ou *comp-*T_i *ou alt-*T_{is}
- $T_i (r_1, r_2, r_3, r_4)$ dénote une sous-transaction principale i avec son rôle (voir définition 6.1)
- T_p dénote une transaction AdapT parent {racine ou autre}
- *Su* dénote l'ensemble des transactions qui s'exécutent sur le site u. On a supposé que T a le droit d'exécuter une seule sous-transaction par site.

Définition 6.2 : définition du modèle AdapT ;

i. $ES_T = ES_t$ = {begin, commit, abort}

ii. EI_T, EI_t = {begin}

iii. ET_T, ET_t = {commit, abort}

iv. t satisfait les axiomes fondamentaux I à IV (voir annexe B)

v. $Visibilité_t = H^{(Su)}$

vi. $ConflitSet_t = \{p_t [ob]\ t_i \neq t;\ t',\ t \in S_u\ ;\ Inprogress\ (p_{t'} [ob])\}$

vii. $\forall ob\ \exists p\ (p_t [ob] \in H) \Rightarrow (ob$ est atomique*)*
ob est correct et sérialisable.

viii. $(commit_t \in H)) \Rightarrow \neg\ (t\ C^*\ t)$
t peut valider *localement* si elle ne génère pas de cycles.

ix. $(begin_{ti} \in H) \Rightarrow ConditionContexte \wedge (t\ \mathcal{BD}\ T) \in \mathrm{DepSet_{ct}}$.

Le début d'une sous-transaction t est conditionné à la satisfaction d'une condition d'environnement et le démarrage de la transaction racine T. La condition d'environnement devient vraie lorsque l'état actuel du contexte (caractéristiques de l'environnement) coïncide avec les valeurs exigées dans le *descripteur de contexte*.

x. $\exists ob\ \exists p\ (commit_t\ [p_{ti}\ [ob]] \in H) \Rightarrow (commit_t \in H)$
La validation de p_t [ob] implique la validation de t.

xi. $(commit_t \in H) \Rightarrow \forall ob\ \forall p\ ((p_{ti}\ [ob] \in H) \Rightarrow (commit_{Ti}\ [p_t\ [ob]] \in H))$
La validation de t implique la validation de toutes ses opérations.

xii. $\exists ob\ \exists p\ (abort_t\ [p_{ti}\ [ob]]\ \in H) \Rightarrow (abort_t\ \in H)$
L'annulation de $p_{ti}\ [ob]$ implique l'annulation de *t.*

xiii. $(abort_t \in H) \Rightarrow \forall ob\ \forall p\ ((p_{ti}\ [ob] \in H) \Rightarrow (abort_t\ [p_{ti}\ [ob]]\ \in H))$
L'annulation de *t* implique l'annulation de toutes ses opérations.

xiv. $(commit_T \in H) \Rightarrow \neg (T\ \mathcal{R}^*\ T)$
T ne peut valider globalement que si elle ne génère pas de cycles.

xv. $T_i\ (r_1, r_2, r_3, r_4)\ \mathcal{BCD}\ T_j\ (r_1, r_2, r_3, r_4)$
$(begin_{Ti} \in H) \Rightarrow (commit_{Tj} \rightarrow begin_{Ti})$

$T_i\ (r_1, r_2, r_3, r_4)\ \mathcal{BCD}\ alt\text{-}T_{is}\ (r_1, r_2, r_3, r_4)$
$(begin_{Ti} \in H) \Rightarrow (commit_{\ alt\text{-}Tjs} \rightarrow begin_{Ti})$

Cet axiome établie un ordre partiel entre les sous-transactions inter dépendantes y compris leurs transactions alternatives.

xvi. $T_P\ \mathcal{CD}\ T_i\ (r_1, r_2, r_3, r_4)$
$(commit_{Tp} \in H) \Rightarrow ((commit_{Ti} \in H) \Rightarrow (commit_{Ti} \rightarrow commit_{Tp})$

La validation des transactions parents est dépendante de celles des enfants. Si la transaction parent et la transaction enfant valident alors la validation de l'enfant précède celle du parent. Cependant, les sous-transactions peuvent valider indépendamment.

xvii. $T_i(compensable, r_2, r_3, r_4)\ \mathcal{WD}\ T_P$
$(abort_{Tp} \in H) \Rightarrow (\neg (commit_{Ti} \rightarrow abort_{Tp}) \Rightarrow (abort_{Ti} \in H))$

Cet axiome garanti l'annulation d'une sous-transaction enfant compensable qui n'a pas validé en cas d'annulation du parent.

xviii. $T_i(\neg compensable, r_2, r_3, r_4)\ \mathcal{CD}\ T_P$
$(commit_{\ Ti} \in H) \Rightarrow ((commit_{\ Tp}) \Rightarrow (commit_{\ Tp} \rightarrow commit_{\ Ti})$
$T_i(\neg compensable, r_2, r_3, r_4)\ \mathcal{AD}\ T_P$
$(abort_{Tp} \in H) \Rightarrow (abort_{Ti} \in H)$

Cet axiome oblige les sous-transactions non compensables à ne jamais valider avant leurs parents. Si le parent annule alors les enfants non compensables annulent.

xix. $comp\text{-}T_i(r_1, r_2, r_3, r_4)\ \mathcal{BCD}\ T_i\ (compensable, r_2, r_3, r_4)\ \&\ cT_i(r_1, r_2, r_3, r_4)\ \mathcal{BAD}\ T_P$
$(begin_{comp\text{-}Ti} \in H) \Rightarrow (commit_{Tj} \rightarrow begin_{comp\text{-}Ti})\ \&\ (begin_{comp\text{-}Ti} \in H) \Rightarrow (abort_{Tp} \rightarrow begin_{comp\text{-}Ti})$

Les transactions compensatrices d'une transaction ne peuvent commencer leur exécution avant la validation de cette dernière. De plus, la transaction compensatrice ne peut commencer avant l'annulation de la transaction parent.

$Comp\text{-}T_i(r_1, r_2, r_3, r_4)\ \mathcal{CMD}\ T_P$

$(abort_{Tp} \in H) \Rightarrow (commit_{cTi} \in H)$

Pour les sous-transactions compensables qui ont validées, si leurs parents annulent alors les sous-transactions compensatrices doivent valider.

xx. $alt\text{-}T_{is}$ = s^{ième} alternative de T_i
$alt\text{-}T_{is}(r_1, remplaçable, r_3, r_4)\ \mathcal{BAD}\ T_i(r_1, remplaçeable, r_3, r_4)$
$(begin_{alt\text{-}Tis} \in H) \Rightarrow (abort_{Tj} \rightarrow begin_{alt\text{-}Tis})$

Une transaction alternative ne peut commencer avant l'annulation de la transaction à laquelle elle est associée.

$Alt\text{-}T_{is}(r_1, \neg\ remplaçable, r_3, r_4) \wedge (ConditionContexte)\ \mathcal{BAD}\ T_{i\text{-}1}(r_1, \neg\ remplaçeable, r_3, r_4)$.
$(begin_{alt\text{-}Tis} \in H) \Rightarrow (ConditionContexte) \wedge (abort_{alt\text{-}Tjs\text{-}1} \rightarrow begin_{alt\text{-}Tis})$

Le début d'une transaction alternative est conditionné à la satisfaction d'une condition d'environnement et une transaction alternative ne peut commencer avant l'annulation d'une transaction alternative précédente. La condition d'environnement devient vraie lorsque l'état actuel du contexte (caractéristiques de l'environnement) coïncide avec les valeurs exigées dans le *descripteur de contexte* spécifié pour la transaction.

xxi. $T_P\ \mathcal{AD}\ T_{ki}(r_1, r_2, r_3, vital)$
$(abort_{Ti} \in H) \Rightarrow (abort_{Tp} \in H)$

Si une sous-transaction vitale annule, alors la transaction parent doit annuler.

xxii. $T_i(r_1, r_2, re\text{-}exécutable, r_4)\ \mathcal{BAD}\ T_i(r_1, r_2, re\text{-}exécutable, r_4)$
$(begin_{Ti} \in H) \Rightarrow (abort_{Ti} \rightarrow begin_{Ti})$
Une sous-transaction ne peut être re-exécutée que si la tentative précédente a été annulée.

Les dépendances utilisées sont :

Dépendance de validation (Commit Dependency) ($t_j\ \mathcal{CD}\ t_i$)**:** si les deux transactions t_i et t_j valident, la validation de t_i doit précéder celle de t_j :
$(commit_{tj} \in H) \Rightarrow ((commit_{ti} \in H) \Rightarrow (commit_{ti} \rightarrow commit_{tj}))$.

Dépendance d'annulation (Abort Dependency) ($t_j\ \mathcal{AD}\ t_i$) : si t_i annule, alors t_j aussi:
$(abort_{ti} \in H)) \Rightarrow (abort_{tj} \in H)$.

Dépendance d'annulation faible (Weak-Abort Dependency) ($t_j\ \mathcal{WD}\ t_i$): si t_i annule et t_j n'a pas encore validé, t_j doit annuler aussi. En d'autres termes, si t_j valide et t_i annule, la validation de t_j doit précéder l'annulation de t_i dans l'historique:
$(abort_{ti} \in H) \Rightarrow (\neg (commit_{tj} \rightarrow abort_{ti}) \Rightarrow (abort_{tj} \in H))$.

Dépendance de compensation (Force-Commit-on-Abort Dependency) ($t_j\ \mathcal{CMD}\ t_i$):
si t_i annule, t_j doit valider:
$(abort_{ti} \in H) \Rightarrow (commit_{tj} \in H)$.

Dépendance de début (Begin Dependency) (t_j $\mathcal{BD}$ t_i): t_j ne peut pas démarrer si t_i n'a pas démarré:

$(begin_{tj} \in H) \Rightarrow (begin_{ti} \rightarrow begin_{tj})$.

Dépendance début sur validation (Begin-on-Commit Dependency) (t_j $\mathcal{BCD}$ t_i): t_j ne peut pas démarrer jusqu'à ce que t_i valide:

$(begin_{tj} \in H)) \Rightarrow (commit_{ti} \rightarrow begin_{tj})$.

Dépendance début sur annulation (Begin-on-Abort Dependency) (t_j $\mathcal{BAD}$ t_i): t_j ne peut pas démarrer jusqu'à ce que t_i annule:

$(begin\ t_j \in H) \Rightarrow (abort\ t_i \rightarrow begin\ t_j)$.

Les axiomes 1-3 de la définition 6.2 indiquent les événements significatifs du modèle AdapT. L'axiome 4 dit que les sous-transactions ainsi que leurs transactions de compensation et alternatives doivent respecter les axiomes fondamentaux. L'axiome 5 montre que la visibilité de T_i est limitée à la projection de l'historique H sur S_u (l'ensemble de transactions qui s'exécutent sur le site *u*). De ce fait, dans l'axiome 6, l'ensemble des conflits de T_i est composé potentiellement de toutes les opérations en cours faites par d'autres transactions dans S_u. L'axiome 7 indique que tous les objets accédés par une transaction atomique sont atomiques (voir annexe B). L'axiome 8 indique que les sous-transactions doivent être sérialisables.

L'axiome 9 introduit la prise en compte de l'état de l'environnement dans l'exécution des transactions. Les axiomes de 16 à 22 expriment les différentes dépendances entre les transactions. Ces dépendances spécifient les liens qui découlent directement de la structure. Par exemple, qu'un ordre partiel existe entre les sous-transactions qui sont liées; une transaction parent ne peut être validée que si toutes ses sous-tranasctions vitales sont validées (axiome 21), etc.

6.3.2 La propriété d'atomicité

Dans cette section, nous montrons que les transactions composantes ainsi que de compensation sont atomiques (Lemme 6.2) donc avec les propriétés AID.

Lemme 6.1: Si $T_i \in T_{AdapT}$; T_i est failure atomic.

Preuve du Lemme 6.1: T_i est failure atomic si elle satisfait les deux conditions de la définition B.5 (voir annexe B).
1. La condition 1 (la clause du tout) est dérivée des axiomes 10 et 11.
2. La condition 2 (la clause du rien) est dérivée des axiomes 12 et 13.

Lemme 6.2 : Si $T_i \in T_{AdapT}$; T_i est une transaction atomique.

Preuve du Lemme 6.2: T_i est une transaction atomique si elle satisfait les deux conditions du Théorème B.1 (voir annexe B).

1. La condition 1 (failure atomic) est dérivée du Lemme 6.1.
2. La condition 2 (sérialisable) est dérivée des axiomes 7 et 8.

Puisque les transactions atomiques satisfont les propriétés AID (cf. Théorème B.1), alors les transactions composantes ainsi que de compensation sont des transactions AID.

Les transactions atomiques, réparties et emboîtées

Les transactions composantes peuvent être des transactions plates (atomiques) mais aussi des transactions réparties et emboîtées puisqu'elles satisfont les propriétés AID.

D'après le Théorème B.2, une transaction répartie est atomique (setwise failure atomicity) et sérialisable.

D'après le Théorème B.3, une transaction emboîtée est sérialisable. D'après la spécification introduite dans [Chr91], la racine d'une transaction emboîtée est failure atomic (Axiomes 11-14 et Lemme 4.4 dans [Chr91]).

Ainsi :

- Les transactions atomiques, les transactions réparties, ainsi que la racine des transactions emboîtées assurent la propriété d'atomicité[28].
- Les transactions atomiques, les transactions réparties et emboîtées sont sérialisables.

Donc,

Une AdapT T_k peut être composée par des transactions composantes atomiques, réparties ou emboîtées.

Atomicité relâchée d'une transaction AdapT

Avant de montrer que les transactions AdapT ont la propriété d'atomicité relâchée, nous commençons par définir la validation d'une transaction T (Lemme 6.3) ainsi que son annulation (Lemme 6.4). Ensuite, nous obtenons l'atomicité relâchée d'une transaction T (Lemme 6.5).

Lemme 6.3: La validation d'une transaction T
Soit H l'historique d'une transaction T avec n transactions composantes.
$((commit_T \in H) \Rightarrow \ \forall i;\ 1 \leq i \leq n\ (commit_{Ti\ (r1,r2,\ r3,\ vitale)} \in H))$

La validation d'une transaction globale T, implique la validation de toutes les transactions composantes vitales T_i associées.

Preuve du Lemme 6.3: Si T valide, son ensemble de transactions composantes vitales doit valider grâce à la dépendance d'annulation de T sur T_i vitale de l'axiome 21 et à l'axiome fondamental III:
$\forall i;1 \leq i \leq n\ ((abort_{Ti\ (r1,r2,\ r3,\ vitale)} \in H)) \Rightarrow (abort_T \in H)) <=> ((commit_T \in\ H) \Rightarrow (commit_{Ti\ (r1,r2,\ r3,\ vitale)} \in\ H))$

Lemme 6.4: L'annulation d'une transaction T
Soit H l'historique d'une transaction T avec n transactions composantes.
$(abort_T \in\ H) \Rightarrow \forall i;1 \leq i \leq n;\ \forall j;1 \leq j \leq n;\ i \neq j;\ ((\ abort_{Ti} \in\ H) \wedge (commit_{Tj} \rightarrow commit_{comp\text{-}Tj}))$

L'annulation de T implique l'annulation ou la compensation des transactions composantes associées (vitales ou non).

Preuve du Lemme 6.4:
Cas 1. Si T annule lorsque T_i est en cours, T_i annule grâce aux dépendances $\mathcal{WD}$ ou $\mathcal{AD}$ de T_i sur T (Axiomes 17). Grâce à l'axiome fondamental II, il n'est pas nécessaire de spécifier que

[28] Failure atomicity et setwise failure atomicity assurent l'atomicité d'une ou de plusieurs transactions respectivement.

seules les transactions initiées sont annulées. Egalement, grâce à l'axiome fondamental III, il n'est pas nécessaire de spécifier que seule les transactions non validées sont annulées. Grâce à la dépendance $\mathcal{BCD}$ de *comp-*T_j sur T_i (axiome 19) *comp-*T_j ne démarre pas dans ce cas :
$(abort_T \in H) \Rightarrow \forall i; 1 \leq i \leq n; (abort_{Ti} \in H)$

Seules les $T_i \in$ `Comp-Trs` peuvent valider grâce à la dépendance $\mathcal{WD}$. Par contre, les $T_i \in \neg$`Comp-Trs` valident une fois que T valide grâce à la dépendance d'annulation de T_i sur T. Ainsi, toutes les $T_i \in \neg$`Comp-Trs` en cours seront annulées lors de l'annulation de T.

Cas 2.

1. Si T annule après que T_i valide et avant que une T_j démarre, *comp-*T_i doit valider grâce à la dépendance $\mathcal{CMD}$ de l'axiome 19:
$(abort_T \in H) \Rightarrow \forall i; 1 \leq i \leq n; (commit_{comp\text{-}Ti} \in H)$

2. Étant donnée l'existence de la dépendance $\mathcal{BCD}$ de *comp-Ti* sur T_i dans l'axiome 19, si *comp-*T_i valide alors T_i a du valider :
$(begin_{cTj} \in H) \Rightarrow (commit_{Ti} \rightarrow begin_{comp\text{-}Ti})$

Par l'Axiome fondamental II:
$(commit_{comp\text{-}Ti} \in H) \Rightarrow (begin_{comp\text{-}Ti} \rightarrow commit_{comp\text{-}Ti})$

Ainsi, par la sémantique de la relation de dépendances, si *comp-*T_i valide, elle le fait après T_i
$(commit_{comp\text{-}Ti} \in H) \Rightarrow (commit_{Ti} \rightarrow commit_{comp\text{-}Ti})$

3. A partir de 1 et 2 :
$((abort_T \in H) \Rightarrow \forall i; 1 \leq i \leq n; (commit_{comp\text{-}Ti})) \wedge ((commit_{comp\text{-}Ti} \in H) \Rightarrow (commit_{Ti} \rightarrow commit_{comp\text{-}Ti}))$

Plus simplement:
$(abort_T \in H) \Rightarrow \forall i; 1 \leq i \leq m; (commit_{Ti} \rightarrow commit_{comp\text{-}Ti})$

Cas 3. Si une T est annulée lorsqu'une T_i est en cours (Cas 1) et après qu'une T_j est validée (Cas 2):
$(abort_T \in H) \Rightarrow \forall i; 1 \leq i \leq n; \forall j; 1 \leq j \leq n; i \neq j ((abort_{Ti} \in H) \wedge (commit_{Tj} \rightarrow commit_{comp\text{-}Tj}))$

Avant de montrer que les T ont la propriété d'atomicité relâchée, nous définissons l'atomicité relâchée des alternatives d'exécution comme suit:

Définition 6.4 : Atomicité sémantique d'une transaction T.

Chaque transaction T assure l'atomicité relâchée:
1. si T est validée, toutes les transactions T_i *vitales doivent aussi être validées, et*
2. si T est annulée, toutes les transactions T_i *vitales doivent soit être annulées soit être compensées.*

Autrement dit:
1. $(commit_T \in H) \Rightarrow \forall i; 1 \leq i \leq n; (commit_{Ti\ (r1,r2,\ r3,\ vitale)} \in H)$
2. $(abort_T \in H) \Rightarrow \forall i;\ 1 \leq i \leq n; (\neg (commit_{Ti\ (r1,r2,\ r3,\ vitale)} \rightarrow abort_T) \Rightarrow (abort_{Ti} \in H)) \wedge$
$((commit_{Tic(r1,r2,\ r3,\ vitale)} \rightarrow abort_T) \Rightarrow (commit_{Tic(r1,r2,\ r3,\ vitale)} \rightarrow commit_{comp\text{-}Ti}))$
où comp-T_i *est une transaction de compensation pour* T_i.
Plus simplement :
$(abort_T \in H) \Rightarrow \forall i;\ 1 \leq i \leq n;\ \forall j;\ 1 \leq j \leq n;\ i \neq j ((abort_{Ti} \in H) \wedge (commit_{Tj} \rightarrow commit_{comp\text{-}Tj}))$.

Lemme 6.5: Chaque transaction *T* assure l'atomicité relâchée.
T assure l'atomicité relâchée en satisfaisant les deux conditions de la définition 6.3.

Preuve du Lemme 6.5 :
1. La condition 1 de la définition 6.3 (si *T* valide, toutes les transactions T_i doivent aussi valider) est satisfaite par le Lemme 6.3.

2. La condition 2 de la définition 6.3 (si *T* annule, toutes les transactions T_i doivent soit annuler, soit compenser) est satisfaite par le Lemme 6.4.

6.3.3 La propriété de sérialisabilité globale des transactions AdapT

Les transactions AdapT sont sérialisables globalement. En effet, à partir de certains axiomes de la définition 6.2, nous déduisons la sérialisabilité globale (Lemme 6.6).

Nous commençons par définir la sérialisabilité globale d'un ensemble de transactions adaptables *T*.
Soit *t* une transaction locale t_i ou composante T_i.
Soit *G* l'ensemble des transactions adaptables.
Soit S_u l'ensemble des transactions *t* qui s'exécutent sur le site *u*.
Soit Su_{commit} le sous-ensemble de S_u qui contient les transactions *t* validées.
Soit $\mathcal{R}$ une relation binaire sur *G*.

Définition 6.4 : Sérialisabilité globale des transactions adaptables.
Un ensemble G assure la sérialisabilité globale:
1. si l'ordre d'exécution des transactions composantes assure un ordre sérialisable dans chaque site, et
2. si les transactions *T* assurent également un ordre sérialisable sur tous les sites.
Autrement dit :
$\forall T ; T' \in G; T \neq T'$
$(T \mathcal{R} T')$ *si*
$\exists S_u ; \exists T_i \in Su_{commit} ;$ T_i *composante de* $T ; \exists T_i' \in Su_{commit} ;$ T_i' *composante de* $T' ; \exists t \in Su_{commit} ; \exists ob; ob_1 \exists p; q$
$(conflit (p_{Ti} [ob]; q_{Ti'} [ob]) \wedge (p_{Ti} [ob] \rightarrow q_{Ti'} [ob])) \vee$
$(conflit (p_{Ti} [ob]; q_t [ob]) \wedge (p_{Ti} [ob] \rightarrow q_t [ob])) \wedge$
$(conflit (p_{Ti} [ob_1]; q_{Ti'} [ob_1]) \wedge (p_{Ti} [ob_1] \rightarrow q_{Ti'} [ob_1]))$

Un ensemble G est globalement sérialisable si:
1. $\forall u \ \forall t \in Su_{commit} (commit_t \in H) \Rightarrow \neg (t\ C^*t)$
2. $\forall T \in G (commit_T \in H) \Rightarrow \neg (T\ \mathcal{R}^*\ T)$

Lemme 6.6 *: Les T assurent la sérialisabilité globale.*

Preuve du Lemme 6.6 *:* Pour assurer la sérialisabilité globale les deux conditions de la définition 6.4 doivent être satisfaites :
1. La condition 1 est dérivée de l'axiome 8.
2. La condition 2 est dérivée de l'axiome 14.

6.4 Le protocole de validation adaptable *a*TCP

Un ACP supportant un modèle de transaction flexible comme celui décrit ci-dessus offre une large gamme de propriétés ACID et non ACID. Le programmeur d'application peut spécifier les niveaux d'ACIDité en termes de types ou rôles des sous-transactions. Dans cette section nous présentons le protocole *a*TCP (Adaptable Transaction Commit Protocol).

Du fait de la mobilité, le protocole proposé doit permettre aux systèmes locaux une liberté d'exécution et limiter les communications sans fil et le blocage des ressources. Il permet une validation anticipée optimiste pour les sous-transactions compensables et une validation pessimiste coordonnée avec la transaction globale pour celles qui ne sont pas compensables. Pour ces dernières, une fois que leurs opérations sont terminées et qu'elles se mettent dans l'état de *préparation*, elles ne peuvent ni être annulées ni validées de manière autonome.

Le protocole *a*TCP comporte un **coordinateur** et un ensemble de **participants** (figure 6.2). Chaque participant est en fait composé du participant qui représente de façon classique le système de gestion des données locales et un *proxy* associé.

● Le ***proxy*** interagit avec le participant local exécutant une sous-transaction. Le proxy permet la participation, à la validation globale, du système local même s'il n'est pas conçu pour le faire. Il permet notamment de cacher l'hétérogénéité des techniques utilisées localement. Grâce au proxy le nombre de messages est réduit lors de la validation globale. Le proxy se trouve sur le même site, UM ou UF, que le système sous-jacent.

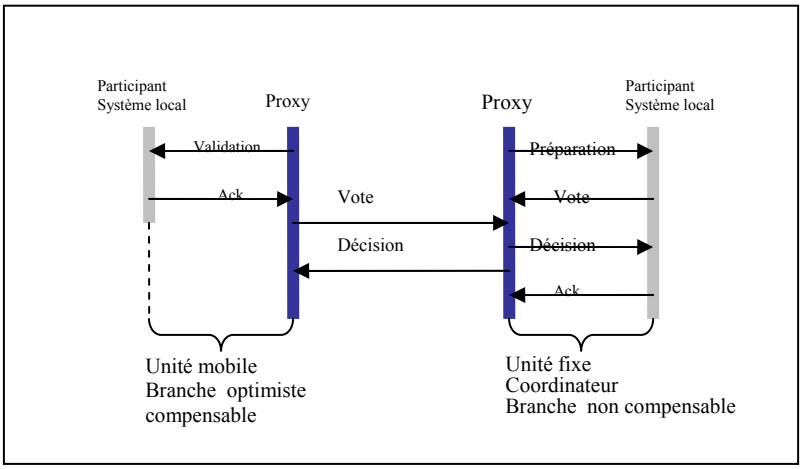


Figure 6.2-Le protocole de validation globale

A la fin d'exécution des opérations d'une sous-transaction compensable, le proxy décide de valider de façon optimiste et envoie un message de validation (vote-commit dans le diagramme figure 6.2) au coordinnateur. Si la transaction globale est annulée par la suite, le proxy doit exécuter une sous-transaction de compensation. A la fin d'exécution d'une sous-transaction non compensable, le proxy exécute avec le participant local un ACP pessimiste, envoie le vote au coordinnateur et attend la décision globale avant de finaliser la décision locale. Si la sous-transaction est annulée, le proxy fait suivre le message de vote d'annulation (vote-abort) au coordinnateur qui doit décider selon le rôle de la sous-transaction, le contexte et l'application/utilisateur quelles actions exécuter avant de se mettre dans l'état *"abort"* (figure 6.3).

● Le rôle du **coordinateur** doit être assuré par un des participants fixes. Comme cela a été expliqué dans le chapitre 3, un participant mobile n'est pas indiqué pour assurer ce rôle vu

qu'il est sujet à de fréquentes déconnexions et ne dispose pas toujours de ressources suffisantes pour la journalisation et la transmission des messages.

D'une façon générale, les sous-transactions (T_i) d'une transaction globale T_{AdapT} sont réparties. Chacune des T_i est exécutée sur un gestionnaire de données sous-jacent. Afin de valider globalement, tous les participants (proxies) doivent envoyer un message de vote (validation ou annulation) au coordinateur qui prend une décision globale, et la propage aux participants. Si tous les votes sont pour la validation, la décision prise est la validation globale. Dans le cas contraire, plusieurs situations peuvent se présenter.

i) Si le degré d'atomicité désiré est "atomité stricte" alors T_{AdapT} sera abandonnée.

ii) Si le degré d'atomicité désiré est "atomicité sémantique" alors T_{AapT} sera abandonnée et si des sous-transactions compensables ont été validées, il faudrait démarrer des transactions de compensation sur les participants optimistes.

iii) Si le degré d'atomicité est "atomicité relâchée" alors T_{AapT} n'est pas forcément abandonnée. En effet, la décision dépend du rôle de la sous-tranaction qui n'a pas été validée.

-Elle est non vitale, il n' y a pas d'incidence sur l'exécution du reste de la T_{AdapT}, la sous-transaction défaillante est juste ignorée.
-Elle est vitale, à ce niveau il est possible de changer le rôle de la sous-transaction en non vitale et la sous-transaction défaillante est juste ignorée.
-Elle est vitale et ne change pas mais elle n'est ni remplaçable et ni re-exécutable, la T_{AdapT} doit être abandonnée.
-Elle est vitale et remplaçable, réévaluer la possibilité d'exécuter une alternative.
-Elle est vitale et re-exécutable, réévaluer les paramètres de re-exécution et la re-exécuter (les paramètres de re-exécution).

● Le **participant** représente ici le système sous-jacent responsable de l'exécution effective des sous-transactions sur les données. Son protocole est standard, car nous avons tenu à préserver l'autonomie des sites existants et leur hétérogénéité.

Le diagramme de transitions d'état du protocole *a*TCP *(Adaptable Transaction Commitment Protocol)* est donné dans la figure 6.3. L'étiquette associée à l'arc représente, au dessus de la ligne, la raison de la transition, c'est-à-dire, le message reçu ou une action effectuée et au dessous de la ligne, le message émis ou une action effectuée.

Le protocole proposé combine les approches 1PC, 2PC, hybride entre 1PC et 2PC, l'optimisme et le pessimisme pour une meilleure adaptabilité. Il utilise les timeouts pour limiter les effets du blocage et la notion de transactions non vitales pour réduire le taux d'annulation. Il offre plus de flexibilité grâce à la combinaison de différents types de sous-transactions. Il faut aussi remarquer que le protocole offre la possibilité d'une validation répartie classique avec des sites offrant l'interface 2PC, ou même 1PC[29].

6.4.1 Situations de blockage

Les participants exécutant une approche optimiste (sous-transactions compensables) ne peuvent pas être bloqués car ils prennent leur décision de façon autonome et ne sont pas obligés de bloquer leurs ressources jusqu'à la décision globale. Les participants exécutant une approche pessimiste (sous-transactions non compensables) peuvent être bloqués dans l'état "*wait"*, lorsque après avoir émis leurs votes pour une validation:

[29] Dans ce cas le coordinateur initie le protocole, le proxy ne fera que faire suivre les messages entre le coordinateur et les participants.

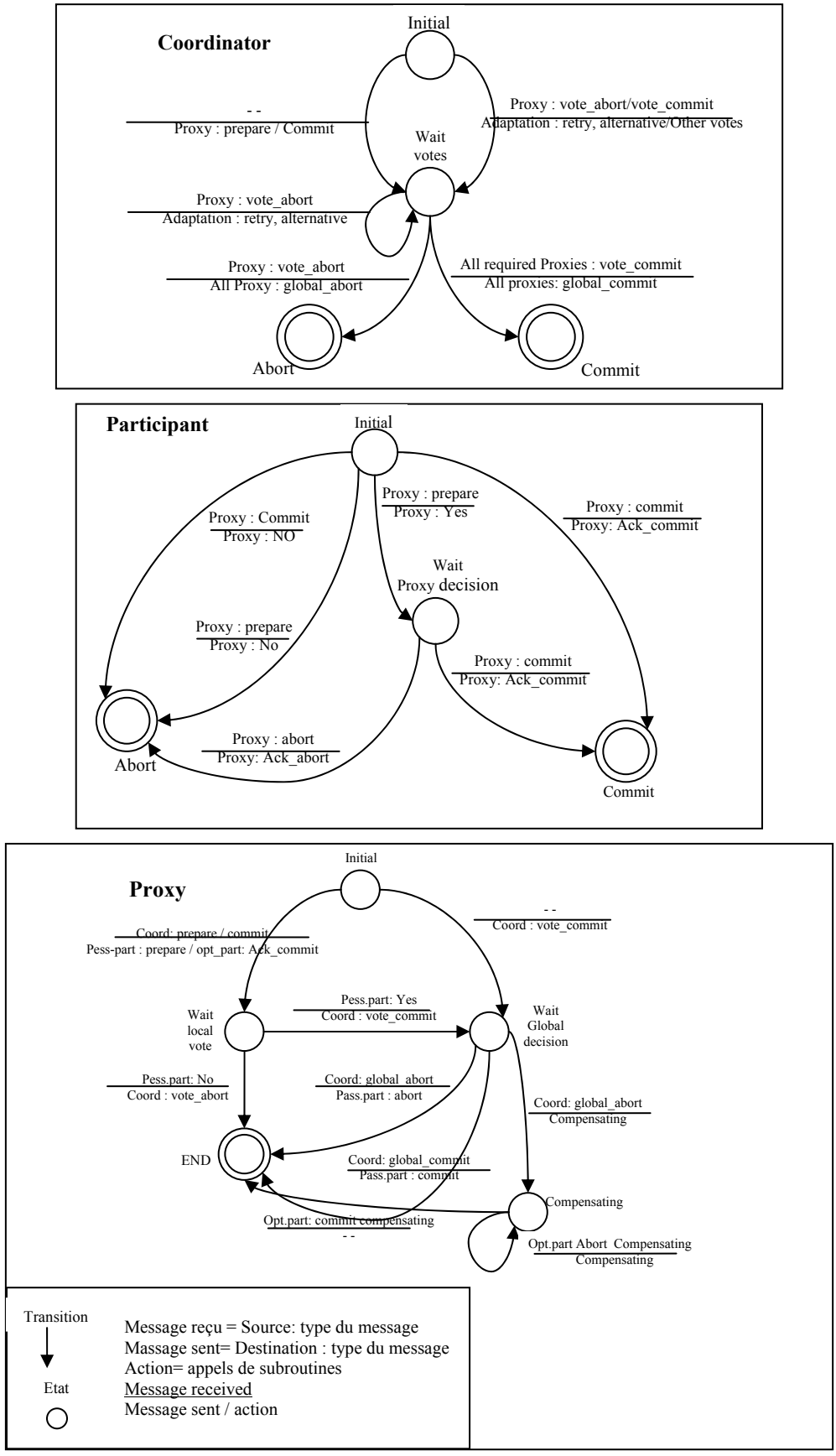


Figure 6.3-Diagramme de transition d'état du protocole adaptable *a*TCP

- l'un des autres participants se déconnecte ou tombe en panne avant l'envoie de son vote,
- le coordinateur tombe en panne, ils se déconnectent indéfiniment.

Afin de limiter la durée des blocages, des délais de garde (timeouts) sont utilisés de la manière suivante:

- Chaque nœud participant doit envoyer son vote avant l'expiration du timeout assigné à sa sous-transaction. Le parcticipant peut renégocier ce délai lorsqu'il prévoit de se déconnecter pour une durée déterminée, et en informer le coordinateur.

- Le coordinateur doit recevoir tous les votes avant un délai global associé à la T_{AdapT} , ce délai étant plus grand que ceux des sous-transactions. Sinon il prend la décision d'annuler.

- Les participants de leur côté doivent recevoir la décision globale avant l'expiration du délai global.

-Si un nœud participant se déconnecte, l'attente de la décision globale est déléguée à un Agent-représentant qui se charge de lui transmettre le résultat lors de la prochaine reconnexion. Suite à cela le participant terminera la transaction selon la décision du coordinateur: validation, annulation ou compensation.

6.4.2 Reprise après panne

Pendant le déroulement des étapes du protocole, des informations sont enregistrées dans un journal pour permettre le recouvrement en cas de pannes. La journalisation se fait au niveau du coordinateur et des participants. Vu que les pannes et les déconnexions peuvent arriver à tout moment, les journaux doivent être écrits sur un support persistant.

- Le participant optimiste: enregistre la demande de validation, l'acquittement, le vote et la décision.
- Le participant pessimiste: enregistre le début du protocole 2PC, la préparation, le vote et la décision.
- Le coordinateur: enregistre le début du protocole *a*TCP, chaque vote et la décision.

Etant donnée la non fiabilité des unités mobiles, les informations des journaux des modifications des données doivent être enregistrées par des écritures forcées sur des supports stables sur des sites fixes. En effet, afin d'assurer la durabilité, il faudrait que ces écritures soient réalisées avant la validation des sous-transactions. Or, cela induirait un coût supplémentaire dû au transfert des journaux (sur le réseau sans fil) et limiterait aussi l'autonomie (chapitres 3 et 5). Mais si cette sauvegarde ne se fait pas, la propriété de durabilité risque d'être compromise. A ce niveau on propose de laisser au développeur d'application de choisir entre deux stratégies [Ser04]:

- forcer le transfert des journaux des modifications pour une durabilité stricte et garantie;

- déléguer la responsabilité d'assurer la durabilité aux systèmes sous-jacents en les laissant choisir de faire le transfert des modifications soit après la validation de chaque transaction, d'un groupe de transactions ou lorsqu'une bonne connexion est disponible, etc. Ceci relâche la propriété de durabilité.

6.4.3 Caractéristiques de *a*TCP

Le protocole *a*TCP possède les propriétés suivantes:

- permet la validation optimiste des sous-transactions compensables; ce qui répond:
    - au besoin d'autonomie des UMs;
    - à la réduction des situations de blocage.

- Permet la validation globale pessimiste lorsque les propriétés strictes sont désirées.
- Réduit le nombre de messages sans fil (un message de vote et un message de décision) dans le cas où on préfère que le coût d'exécution soit minimisé. Cependant, il permet d'utiliser le 2PC ou 1PC traditionnel lorsque le contexte le permet et l'application le désire.
- Permet la déconnexion des participants.

*a*TCP peut assurer une atomicité stricte, sémantique ou relâchée, ceci dépend de la sémantique de l'application et/ou du contexte; d'où la possibilité de faire un compromis entre les propriétés et le coût d'exécution.

Au Niveau global, le schéma d'exécution du protocole *a*TCP est similaire à celui de CO2PC qui a montré de bonnes performances par rapport aux autres protocoles. Cependant, dans le cas de l'atomicité relâchée où des adaptations sont effectuées, ces performances vont être différentes dans le sens où elles devraient s'adapter à la politique choisie par l'utilisateur.

6.5 La sérialisabilité globale

Comme cela a été montré dans la section 6.2.5, assurer la sérialisabilité globale peut s'avérer très compliqué en particulier lorsqu'il s'agit du cas des systèmes hétérogènes qui, en général, ne partagent pas l'information concernant leur gestion locale. De ce fait, les ordonnancements locaux sont hors de la portée du contrôle global. Seule la demande d'exécution des sous-transactions est sous contrôle global. L'une des conséquences de ces caractéristiques est la probabilité d'avoir des conflits indirects.

Afin de faire face à ce problème, de nouveaux critères de correction et techniques de gestion ont été proposés dans les systèmes multibases [BGMS92] dont OTM *(Optimistic Ticket Method)* [GRS94, GRS91] que nous proposons d'utiliser dans cette thèse pour sa simplicité.

OTM introduit des conflits directs entre les transactions globales: chacun des sites accédés contient une donnée qui doit être mise à jour (incrémentée) par toutes les sous-transactions. Ainsi, l'accès au ticket dans chaque site indique l'ordre relatif entre les sous-transactions. Cela permet au gestionnaire global d'assurer un ordonnancement globalement sérialisable. Le gestionnaire global gère un graphe qui représente la sérialisabilité globale. Les transactions globales sont les noeuds du graphe. Les arcs représentent l'ordre d'accès au ticket sur chaque site. Lorsqu'il existe un cycle dans le graphe, le gestionnaire demande l'annulation d'une transaction globale faisant partie du cycle.

La figure 6.4 montre deux transactions AdapT (T et $T^{'}$), chacune avec deux sous-transactions qui s'exécutent sur les sites A et B. Sur chaque site il existe un ticket (ticket A et B) dont l'accès entraîne un ordre entre les transactions T et $T^{'}$: $T_1 < T_1^{'}$ et $T_2^{'} < T_2$. Au niveau global, un graphe de sérialisabilité globale (GSG) est maintenu (figure 6.4). Les noeuds du graphe correspondent aux transactions T et T'. Les arcs représentent l'ordre d'accès aux tickets. Si T et $T^{'}$ s'exécutent sur un même site et $T_i^{'}$ obtient un ticket plus grand que celui de T_i, alors un arc orienté de T vers $T^{'}$est inséré. Lorsqu'un cycle se produit l'une des transactions doit être annulée. La décision d'annulation d'une des transactions générant un cycle, peut être basée sur différents critères (la plus récemment déclenchée, celle qui a le plus de transactions composantes non-validées,…). Un noeud peut être enlevé du graphe de sérialisation globale lorsque la transaction correspondante est validée et que toutes celles qui ont un arc (qui arrive ou qui part) relié à ce noeud ont également été validées.

Dans le chapitre 7, nous montrons l'utilisation d'OTM en combinaison avec le protocole *a*TCP.

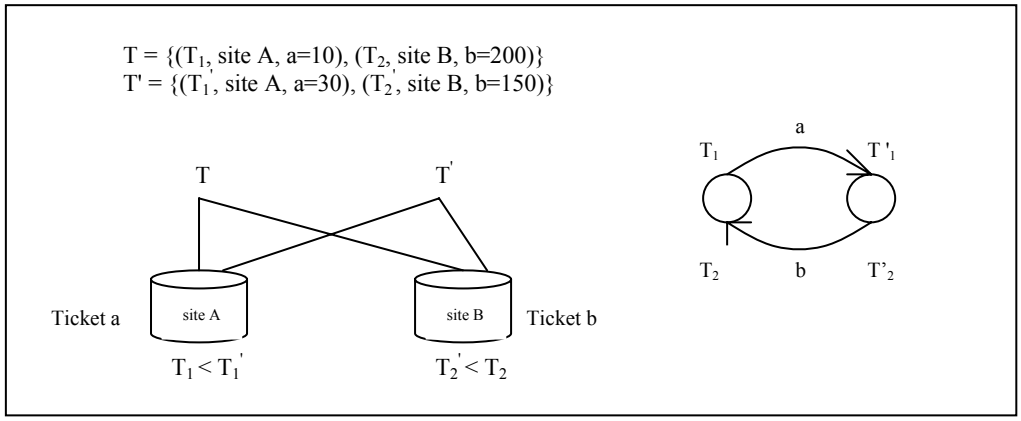


Figure 6.4-Exemple de l'accès aux tickets dans OTM.

6.6 Conclusion

Le besoin d'adaptation des systèmes transactionnels n'est plus contestable et cela est démontré à travers les travaux de recherche qui au départ ce sont intéressés à l'adaptation en termes de prise en charge de plusieurs modèles et qui ont été élargis à l'environnement mobile [TR96, WW04, Kar03, JK04, AK05, Rak98, SVF04, EK06, RSM06]. En effet, la recherche bibliographique et l'analyse des travaux du domaine du transactionnel et des problèmes posés par le calcul mobile permettent de se rendre compte que les modèles et techniques transactionnels ont été depuis longtemps confrontés à la diversité des besoins applicatifs en termes de propriétés transactionnelles. Avec l'avènement du calcul mobile, la variabilité et le caractère dynamique du contexte d'exécution des applications, cette diversité ne fait que s'affirmer allant jusqu'à induire une variabilité des besoins transactionnels durant la vie d'une application ou d'une transaction. A travers ce chapitre nous avons fait des propositions de solutions à un certain nombre problèmes liés à la flexibilité et adaptabilité du traitement transactionnel. Les principales contributions se situent dans le modèle AdapT et le protocole *a*TCP. Le modèle AdapT ne cible pas d'application particulière. Il est basé sur une structure emboitée ouverte associant des sous-transactions de différents types et la description du contexte d'exécution. Ce modèle permet d'offrir plusieurs degrés d'atomicité ajustable dynamiquement grâce au support du protocole de validation adaptable *a*TCP. En tenant compte du contexte, il est possible de contrôler le coût d'exécution des transactions. Notre modèle permet également d'intégrer plusieurs modes d'exécution des transactions mobiles.

Une certaine complexité est ajoutée à la définition des transactions car il faut spécifier les rôles ou types des sous-transactions et décrire le contexet d'exécution. Dans le chapitre 7, nous montrons qu'il est possible d'aider le développeur d'applications dans cette définition par l'utilisation d'une description XML permettant de spécifier la structure transactionnelle et le contexte d'exécution. Cette description définit en fait les stratégies ou politique d'adaptation car elle indique au système les choix en termes de propriétés et de coût d'exécution acceptables.

Le chapitre 7 présente, donc, une architecture qui a pour objectif de supporter la mise en œuvre de techniques transactionnelles adaptables conformes aux propositions décrites dans ce présent chapitre.

Chapitre VII

Architecture middleware pour l'adaptation

A journey of a thousand miles must begin whith a single step.
--Lao-tsu

Dans ce chapitre, nous introduisons un middleware *(que nous désignons par MATrans pour Middleware for Mobile Adapatable Transactions)* conçu pour mettre en oeuvre le modèle AdapT et le protocole *a*TCP présentés dans le chapitre 6. L'objectif de notre approche est de faciliter l'accès transactionnel aux données localisées sur des unités mobiles et fixes en offrant une adaptabilité aux variations du contexte et aux variations des besoins transactionnels. Dans ce chapitre nous commençons par donner l'architecture middleware en 7.1. Ensuite, nous présentons en 7.2 notre proposition sur l'approche context-aware pour l'implémentation de la flexibilité et de l'adaptabilité en nous basant sur l'utilisation de *politiques d'adaptation*.

7.1 Architecture middleware

L'architecture que nous proposons est un middleware situé entre le code applicatif et des serveurs de données existants (figure 7.1). Les bases de données sous-jacentes et les applications peuvent être localisées sur des unités mobiles et/ou fixes. L'architecture est basée sur le modèle *client-agent-serveur* réparti entre des unités mobiles et fixes. Les systèmes de traitement transactionnel locaux sont hétérogènes et ont des capacités de gestion variant des plus légères (sur les UMs) aux plus puissantes (sur les UFs).

Les applications sont de différents types et de différentes complexités. Elles peuvent demander l'exécution de transactions selon les 5 modèles[30] présentés dans le chapitre 1 (exécution sur l'UM, l'UF ou répartie entre plusieurs unités mobiles et/ou fixes). De manière générale, l'application demande à MATrans d'exécuter une transaction T_{AdapT}. Selon l'état courant du contexte mobile et les exigences de l'application et de l'utilisateur en termes de propriétés, MATrans décide de la manière la plus adéquate d'exécuter la transaction notamment en choisissant les alternatives pour les sous-transactions et en distribuant les sous-transactions aux sites correspondants. MATrans assure la sérialisabilité et la validation globales.

[30] Le modèle 6 relatif à une infrastructure ad hoc n'est pas considéré dans cette architecture qui compte sur le support de la partie fixe du réseau.

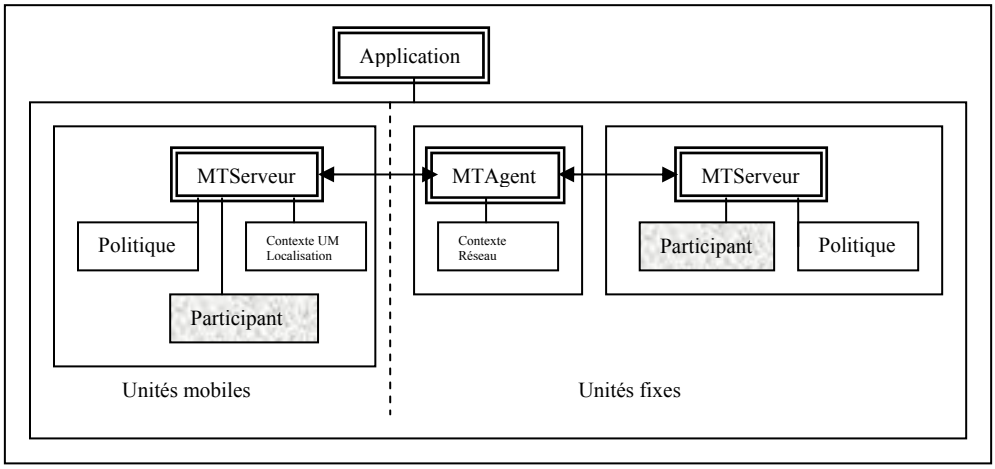


Figure 7.1-Architecture de MATrans

Dans MATrans le tiers client est l'Application, le tiers agent est le MTAgent *(Mobile Transaction Agent)* et le tiers serveur est le MTServeur *(Mobile Transaction Server)* (figure 7.1). Seul le tiers MTAgent est contraint d'être localisé sur une unité fixe. Les autres tiers peuvent être situés aussi bien sur des unités mobiles que fixes. Les tiers installés sur des unités mobiles doivent être attachés à un MTAgent. Toute communication entre un tiers mobile et un tiers fixe doit passer par un MTAgent.

7.1.1 Le MTServeur

Le MTServeur reçoit, de l'application, la demande d'exécution d'une T_{AadapT}. Il analyse les descripteurs de contexte, les sous-transactions et selon l'état courant de l'environnement et les propriétés demandées, il décide quelles sous-transactions peuvent être exécutées. Il est possible qu'au moment du déclenchement de la T_{AdapT}, aucune sous-transaction ne puisse démarrer. Dans ce cas, l'exécution de la T_{AdapT} peut être reportée jusqu'à ce que l'état de l'environnement soit adéquat ou elle soit annulée.

Le MTServeur interagit directement avec le serveur de données sous-jacent (le participant). Il est appelé MTServeur-fixe lorsqu'il s'exécute sur des unités fixes et MTServeur-mobile lorsqu'il s'exécute sur des unités mobiles. Sur un même site, il peut y avoir une application et un serveur de données. Dans ce cas, l'application et le MTServeur peuvent interagir directement qu'ils soient fixes ou mobiles. Un MTServeur :

- interagit avec les participants en demandant l'exécution de transactions composantes pour le compte des applications mobiles. Il utilise des interfaces classiques pour communiquer avec eux (begin, commit, abort).
- participe au processus de validation globale ;
- peut coordonner le processus de validation globale lorsqu'il est un MTServeur-fixe ;
- peut gérer le graphe de sérialisabilité globale lorsqu'il est un MTServeur-fixe ;
- lorsqu'il est mobile, il interagit avec un surveillant de l'état des ressources locales. Un MTServeur-mobile fournit l'information sur les ressources de l'unité mobile où il s'exécute et éventuellement sur les services offerts.

7.1.2 Le MTAgent

Les MTAgents jouent un rôle particulièrement important. Ils sont installés sur des stations base ou sur des unités fixes capables de communiquer avec des unités mobiles. La fonction principale d'un MTAgent est de soutenir les tiers mobiles.

L'utilisation des agents est très favorable aux unités mobiles qui peuvent avoir des déconnexions et qui généralement ont des ressources limitées. Un MTAgent réduit les besoins de communication sans fil car il assure la présence de l'unité mobile. Ceci limite la communication directe avec l'unité mobile. De même, il permet d'économiser des ressources sur les unités mobiles.

Lorsqu'une application mobile (s'exécutant sur une UM) a besoin de répartir des transactions composantes, le MTServeur-mobile les transmet à son correspondant MTAgent qui se charge de la répartition. Lorsqu'il s'agit d'une application fixe le MTServeur-fixe correspondant fait lui-même la répartition des transactions composantes. Pour communiquer avec un MTServeur-mobile, le MTserveur-fixe doit passer par le MTAgent correspondant. D'une façon générale, le rôle du MTAgent vis-à-vis des tiers mobiles est le suivant:

- répartit l'interaction entre la partie fixe et la partie mobile: (1) les unités mobiles avec le MTAgent et (2) le MTAgent avec les unités fixes. Ceci permet d'utiliser des protocoles différents pour chaque partie et l'interaction peut être faite de façon indépendante ;
- est un représentant des tiers mobiles qui stocke les messages destinés aux unités mobiles déconnectées ;
- peut aider les MTServeurs pendant la durée des déconnexions. Par exemple, un MTServeur-mobile peut soumettre ses demandes au MTAgent, se déconnecter, et récupérer les résultats une fois la connexion rétablie ;
- peut être utilisé pour économiser, par exemple, la vie de la batterie. Un MTServeur-mobile peut soumettre ses demandes à son MTAgent, entrer en mode veille et attendre que le MTAgent l'avertisse lorsque la réponse est prête ;
- suit le déplacement des unités mobiles (en coopération avec d'autres MTAgents). Par exemple, lors d'un handoff, le dernier MTAgent envoie l'information concernant l'unité mobile au nouveau MTAgent ;
- peut aider à libérer de la place sur le support persistant des MTServeurs-mobiles en sauvegardant des données temporairement ;
- stocke l'information de coordination transactionnelle des MTServeurs-mobiles (l'état des transactions en cours, le processus de validation, etc.) ;
- peut coordonner le processus de validation globale à défaut d'un MTServeur-fixe participant à l'exécution d'une T_{AdapT} ;
- interagit avec un surveillant de l'environnement (service d'événements) qui perçoit l'état du réseau sans fil. Le MTAgent envoie l'information sur le réseau sans fil aux MTServeurs-mobiles ou fixes.

7.2 Approche context-aware pour l'implémentation du protocole *a*TCP

Comme nous l'avons expliqué précédemment, nous sommes partis de l'idée que la flexibilité et l'adaptabilité recherchée doivent être reconsidérée du point de vue de propriétés adaptables en fonction du contexte, de la sémantique des applications et des objets manipulés. La qualité de service visée sera celle perçue par l'utilisateur et elle dépendra de la perception que l'on a du *contexte*. La figure 7.4 décrit les différents modules ou composants du MTServeur qui prend en charge la majeure partie de la fonction d'adaptation en fonction du contexte[31].

Les systèmes mobiles *contexte-aware* ou adaptatifs sont généralement associés à l'utilisation de *règles* ou *politiques* qui définissent le comportement que doit adopter un système en réponse à un état du contexte. En effet, un système adaptable qui a sa logique d'adaptation encodée directement dans l'application ne peut fonctionner d'une manière flexible

[31] Actuellement, le *context-aware* est l'un des thèmes de recherche importants du *mobile computing* [BD07].

ou s'adapter à des changements imprévus. La spécification de politiques permet de séparer clairement le contrôle de l'adaptation des mécanismes d'adaptation proprement dits. L'architecture du MTServeur que nous proposons (figure 7.4) permet d'offrir un protocole de validation dynamiquement adaptable capable de livrer différents degrés d'atomicité en fonction du contexte et utilisant des politiques *(policy-based approach)*. Avant d'expliquer le fonctionnement du MTServeur, nous définissons d'abord les concepts utilisés et que nous avons introduits dans le chapitre 2, section 2.4.

7.2.1 Adaptation basée sur les politiques

L'adaptation est un processus, qui modifie continuellement l'application pendant son exécution pour qu'elle soit à tout moment la mieux adaptée aux circonstances présentes et aux exigences de l'application et de l'utilisateur. D'une façon générale, on peut décomposer ce processus en trois étapes successives [Dav05]:

1. Observation de l'environnement (contexte d'exécution) de l'application et/ou de l'application elle même, et détection de l'occurrence de certaines circonstances nécessitant une adaptation. Cette étape peut être implémentée grâce à la sensibilité au contexte (context-awareness) d'une application adaptative.

2. Prise de décision : étant donnés d'une part l'état actuel de l'application et d'autre part les nouvelles circonstances détectées dans la première phase, l'application doit décider des opérations de reconfiguration à effectuer pour s'adapter aux nouvelles circonstances. Cette étape correspond à la *stratégie d'adaptation (policy decision point)* de l'application.

3. Action : une fois que les opérations de reconfiguration ont été déterminées, il reste à les appliquer concrètement à l'application, tout en évitant de perturber son bon fonctionnement normal. Cette étape correspond aux *mécanismes de reconfiguration dynamique (policy enforcement point)* de l'application adaptative.

Dans le cas du calcul mobile, *l'approche basée sur les politiques* est adoptée dans des travaux récents comme [STM00] pour la téléphonie contexte-aware sur WAP, [EFDC02] qui propose un système et [KC03, Kee04, CEM03] qui proposent des frameworks pour l'adaptation. En ce qui concerne les systèmes transactionnels, nous avons rencontré [SVF06] qui décrit un certain nombre de politiques pour la gestion des transactions mobiles en fonction de l'état de l'environnement, par exemple la disponibilité des nœuds participants, et la proposition, par [RSM06], d'un service de validation qui adapte la variante du protocole 2PC en fonction des taux d'annulation et de succès des transactions en cours.

7.2.2 Contexte et perception de l'environnement Mobile

Par contexte nous faisons référence à l'état, aux ressources et aux conditions ou exigences de l'environnement d'exécution, de l'application et de l'utilisateur. Des adaptations dynamiques sont la plupart du temps nécessaires à cause des changements de ce contexte. Dans ce chapitre, étant donné que nous sommes intéressés par l'environnement de calcul mobile, nous prenons en compte les principales caractéristiques des unités mobiles et du réseau sans fil. Ces informations sont fournies par un gestionnaire de contexte.

La perception de l'environnement utilise un service d'événements. En général, les événements sont notifiés avec des mécanismes de pull ou push. Dans le premier cas, le consommateur (MATrans) récupère les événements auprès du mécanisme de notification (Event service). Par contre, dans la technique push, le mécanisme de notification signale les événements au consommateur. Les deux types de notification push/pull peuvent être exécutés de manière immédiate (synchrone) ou différée (asynchrone). Dans le mode immédiat, la

notification est faite lorsque l'événement est généré. Par contre, en mode différé les notifications sont faites à des instants précis (périodiquement) ou lorsque le consommateur a besoin des événements. Voici une liste, que l'on peut étendre, de quelques types d'événements. Les événements sont produits à partir de l'information fournie par différents surveillants de l'environnement:

e-connexion, identité de l'UM ;
e-déconnexion, identité de l'UM ;
e-handoff identité de la nouvelle SB ;
e-bande-passante-change, largeur de la bande passante ;
e-prix-communication-chang, prix de communication ;
e-batterie-disponible, batterie disponible ;
e-cache-disponible, cache disponible ;
e-mémoire-persistant-disponible, mémoire persistante disponible ;
e-capacité-calcul-disponible, capacité de calcul disponible ;
e-localité-changes, localité (maison, travail, centre ville); etc.

Le besoin de détection et de notification des événements dépend de l'environnement considéré ainsi que des contraintes sémantiques des applications et des clients. Par exemple, si la bande passante du réseau ne souffre pas de grands changements ou si le prix de communication ne varie pas alors les événements correspondants n'ont pas lieu d'être. Si le prix de communication n'a pas d'importance pour l'utilisateur, il n'est pas intéressant à surveiller. Il faut aussi noter que seules les variations importantes ayant un impact sur l'exécution des transactions doivent être signalées; par exemple, les fortes variations de bande passante (forte à moyenne) et pas les petites variations.

La surveillance de l'environnement mobile peut être faite par le système d'exploitation, la couche de communication, etc. Fréquemment les systèmes d'exploitation prévoient des fonctions pour la surveillance des ressources. Ces fonctions peuvent être directement utilisées ou modifiées afin d'obtenir la surveillance adéquate. Certaines caractéristiques de l'environnement sont faciles à surveiller, d'autres ont besoin d'un matériel particulier. De nombreux travaux ont été consacrés à la collecte, la description et la structuration des informations de contexte, on peut se référer à [BDR06] pour plus de détails.

7.2.3 L'adaptation dans MATrans

Dans cette section, nous présentons les différentes fonctionnalités liées à l'adaptation dans MATrans. Cette adaptation est située au niveau du MTServeur.

7.2.3.1 La notification d'événements dans MATrans

MATrans a besoin de connaître l'état du contexte: (1) lorsqu'une demande d'exécution de transaction arrive, (2) lorsqu'un un état particulier est attendu pour déclencher une des sous-transactions (par exemple, ré-exécuter une transaction lorsqu'un changement d'état de connexion se produit), (3) lorsque le coordinateur reçoit un message d'échec concernant une sous-transaction et qu'une adaptation s'impose et (4) lorsqu'un changement est opéré au niveau de la politique.

1. Connaître l'état de l'environnement au moment du lancement d'une transaction, demande des notifications de type pull. Si l'état correspond au descripteur d'environnement (DC_i) d'une des sous-transactions, celle-ci est déclenchée.

2. Connaître l'état de l'environnement dès qu'un certain état se produit, demande des notifications de type push immédiat. Si l'état ne convient à aucun descripteur d'environnement

des alternatives, le déclenchement peut être retardé jusqu'à ce que l'environnement soit adéquat.

3. Connaître l'état de l'environnement lorsqu'une adaptation est initiée par le coordinateur, demande des notifications de type pull.

4. Connaître l'état de l'environnement lorsqu'un changement est opéré au niveau de la politique, demande des notifications de type pull.

La notification des différents événements peut être faite de la manière la plus appropriée aux caractéristiques de l'environnement mais aussi aux besoins des applications.

7.2.3.2 La spécification de politiques

Les politiques peuvent être utilisées pour la représentation de tous les types d'activité d'adaptation et de monitoring. Le terme "politique" est utilisé de différentes manières dans la littérature. Une définition générale est qu'une "politique" est *une description déclarative d'objectifs à atteindre et d'actions à entreprendre dans différentes situations* [Kee04].

Pour la spécification de la politique, nous avons opté pour une description XML [XML00]. Un fichier de ce type peut être lu, compris et généré par un utilisateur, un programmeur ou une application.

Chaque document XML possède une structure logique et une structure physique [W3C01]. Physiquement, le document est composé d'unités appelées entités. Une entité peut référencer d'autres entités afin de les inclure dans le document. Un document commence dans une entité "root" ou une entité document. Logiquement, le document est composé de déclarations, d'éléments, de commentaires, de références et d'instructions de traitement. Un markage explicite permet d'indiquer ces différentes rubriques. Les éléments peuvent aussi contenir des attributs. La structure d'un document XML est exprimée via un schéma XML, qui est aussi un document XML définissant la synthaxe valide d'une instance de document XML. Le terme instance de document dénote un document conforme au schéma associé.

```
<Transactions>
 <CompensablesTrans>
  <trans-id> T1 <trans-id/>
   <CompensatingTrans> comp-T1 </CompensatingTrans>
       <ContextDescriptor> CD1
         <BatteryLevel> 56% </BatteryLevel>
         <DisconnectionRate> 40% </DisconnectionRate>
          <..> .. </..>
       </ContextDescriptor/>
      <..> .. </..>
   </CompensablesTrans>
<NonCompensablesTrans>
  <trans-id> T2 <trans-id/>
   <ContextDescriptor> CD2
     <DisconnectionRate> 60% </DisconnectionRate>
       <..> .. </..>
     </ContextDescriptor/>
       <..> .. </..>
 </NonCompensablesTrans>
<Transactions>
```

Figure 7.2-Instance de document XML

Les figures 7.2 et 7.3 illustrent une isntance de document XML et son schéma. La structure des étiquettes *(tags)* XML dans l'instance du document est régie par la définition du schéma. Par example, la deuxième ligne du schéma déclare "Transactions" comme l'élément racine du document. L'élément "CompensableTrans" est ansuite ajouté à la racine comme un élément fils. La hiérarchie est étendue de manière similaire pour incorporer tous les éléments désirés. Le choix des étiquettes dépend de leur pertinance par rapport à l'élément réel dans une application particulière. L'extensibilité de ce mécanisme permet de créer des documents spécifiques à l'application.

```
<xs : schema >
  <xs : element name ="Transactions">
   <xs : complexType>
    <xs : element name = "CompensablesTrans">
      <xs : complexType>
       <element name = "trans-id" type ="xs : string"/>
         <xs : element name = " CompensatingTrans" type="xs :  string"/>
          <xs : element name = " ContextDescriptor">
            <xs : complexType>
              <xs : element name =" BatteryLevel" type= "xs : string"/>
              <xs : element name ="DisconnectionRat" type ="xs :  string"/>
                 …….
            <xs : /xs : complexType>
         <xs : /element>
        <xs : /complexType>
      <xs : /element>
       <xs : element name = "NonCompensablesTrans">
         <xs : complexType>
          <element name = "trans-id" type ="xs : string"/>
            <xs : element name = " ContextDescriptor">
              <xs : complexType>
                 <xs : element name =" BatteryLevel" type= "xs : string"/>
                 <xs : element name ="DisconnectionRat" type ="xs :   string"/>
                       …….
              <xs : / complexType>
            <xs : /element>
           <xs : /complexType>
         <xs : /element>
        ……
       <xs : /complexTpe>
   <xs : /element>
<xs : /schema>
```

Figure 7.3-Schéma de document XML

Notre spécification de politique décrit:

-les besoins en termes de propriétés transactionnelles en spécifiant le degré d'atomicité désiré;
- les sous-transactions d'une transaction T_{AdapT} et leurs rôles, leurs transactions compensatrices et alternatives, les conditions de leurs exécution (qui correspondent aux descripteurs de contexte), etc.

Voici, la structure d'une spécification de politique[32]:

```
<Requirements>
  <Requirement>
    <Name>NameOfProperty</Name>
    <Degree>DegreeOfproperty</Degree>
    <TansactionList>
      <Transaction>
        <TransactionId>IdentifierOfTransaction</TransactionId>
        <PrincipleContextDescriptor>
            <disconnexionRate> Rate </disconnexionRate>
            <bandwidth> bandwidth </bandwidth>
            ……
        </PrincipleContextDescriptor>

        <TransactionParmeters>
            <Vitality>ParameterValue </Vitality>
            <Compensation> Comp-TransactionId </Compensation>
            <Re-execution> ConditionsOfRéexecution </Re-execution>
            <Replacable>
              <AlternativesList>
                <Alternative>
                  <AlternativeTransaction> Alt-TransactionId </AlternativeTransaction>
                  <AlternativeDescriptor>
                      <disconnexionRate> Rate </disconnexionRate>
                      <bandwidth> bandwidth </bandwidth>
                      ……
                  </PrincipleContextDescriptor>
                  <Vitality>ParameterValue </Vitality>
                  <Compensation> Comp-TransactionId </Compensation>
                  <Re-execution> ConditionsOfRéexecution </Re-execution>
                </Alternative>
              </AlternativesList>
            </Replacable>
        </TransactionParmeters>
      </Transaction>
    </TransactionList>
  </Requirement>
</Requirements>
```

Voici un exemple de spécification pour une transaction demandant **l'atomicité stricte** quelque soit le contexte:

```
<Requirements>
  <Requirement>
    <Name>Atomicity</Name>
    <Degree>Strict</Degree>
  </Requirement>
</Requirements>
```

Voici un exemple de spécification pour une transaction demandant **l'atomicité sémantique** quelque soit le contexte:

```
<Requirements>
  <Requirement>
    <Name>Atomicity</Name>
    <Degree>Semantic </Degree>
    <TansactionList>
      <Transaction>
        <TransactionId>T1 </TransactionId>
        <PrincipleContextDescriptor>
            <disconnexionRate> >30% </disconnexionRate>
            <bandwidth> moderate </bandwidth>
            ……
        </PrincipleContextDescriptor>

        <TransactionParmeters>
            <Vitality>Vital </Vitality>
            <Compensation> CT1 </Compensation>
        </TransactionParmeters>
```

[32] Cette syntaxe nous a été inspirée par [AK05] qui l'a utilisée dans sa proposition d'un framework capable de supporter plusieurs modèles de transactions. Mais nous l'avons étendue de manière à inclure les aspects ECA et contexte-awarness.

```
            </Transaction>
          <Transaction>
              <TransactionId>T2 </TransactionId>
              <PrincipleContextDescriptor>
                    <disconnexionRate> <10% </disconnexionRate>
                    <bandwidth> High  </bandwidth>
                    ……
              </PrincipleContextDescriptor>

              <TransactionParmeters>
                    <Vitality>Vital </Vitality>
                    <Compensation> CT2 </Compensation>
              </TransactionParmeters>
            </Transaction>
          </TransactionList>
      </Requirement>
  </Requirements>
```

Voici un exemple de spécification pour une transaction demandant **l'atomicité relâchée** s'adaptant au contexte:

```
  <Requirements>
    <Requirement>
      <Name>Atomicity </Name>
      <Degree>Relaxed </Degree>
      <TansactionList>
        <Transaction>
          <TransactionId>T1 </TransactionId>
          <PrincipleContextDescriptor>
                <disconnexionRate> > 30%</disconnexionRate>
                <bandwidth> moderate  </bandwidth>
                ……
          </PrincipleContextDescriptor>
          <TransactionParmeters>
                <Vitality>Vital </Vitality>
                 <Re-execution> 3 tentatives </Re-execution>
                <Replacable>
                   <AlternativesList>
                          <Alternative>
                               <AlternativeTransaction> alT11</AlternativeTransaction>
                               <AlternativeDescriptor>
                                      <disconnexionRate> < 30% </disconnexionRate>
                                      <bandwidth> low  </bandwidth>
                                      ……
                                </PrincipleContextDescriptor>
                                <Vitality>Vital </Vitality>
                                <Compensation> calT11 </Compensation>
                          </Alternative>
                          <Alternative>
                               <AlternativeTransaction> alT12</AlternativeTransaction>
                        </Alternative>
                    </AlternativesList>
                </Replacable>
          </TransactionParmeters>
        </Transaction>
      </TransactionList>
    </Requirement>
  </Requirements>
```

Comme on le voit dans les exemples ci-dessus, la formulation des règles d'adaptation correspond à indiquer les conditions sur l'état des caractéristiques (par exemple, Disconnection rate>30%) et l'action indique les valeurs associées aux paramètres (par exemple, vital) qui se traduit au final par le degré d'atomicité.

La politique peut être mise à jour à tout moment pour prendre en compte le désire de l'utilisateur et de l'application, mais aussi pour éventuellement élargir le contexte.

7.2.3.3 Les mécanismes d'adaptation

La figure 7.4 présente les différentes entités participant dans l'exécution d'une transaction mobile adaptable. Dans ce schéma, on retrouve bien sûr les fonctionnalités classiques nécessaires à la gestion de transactions mais aussi les fonctionnalités propres à un

environnement context-aware permettant l'adaptation. L'architecture modulaire du gestionnaire d'adaptation (Adaptation Manager) lui permet d'être flexible et extensible avec chaque sous-gestionnaire assurant un nombre précis de responsabilités.

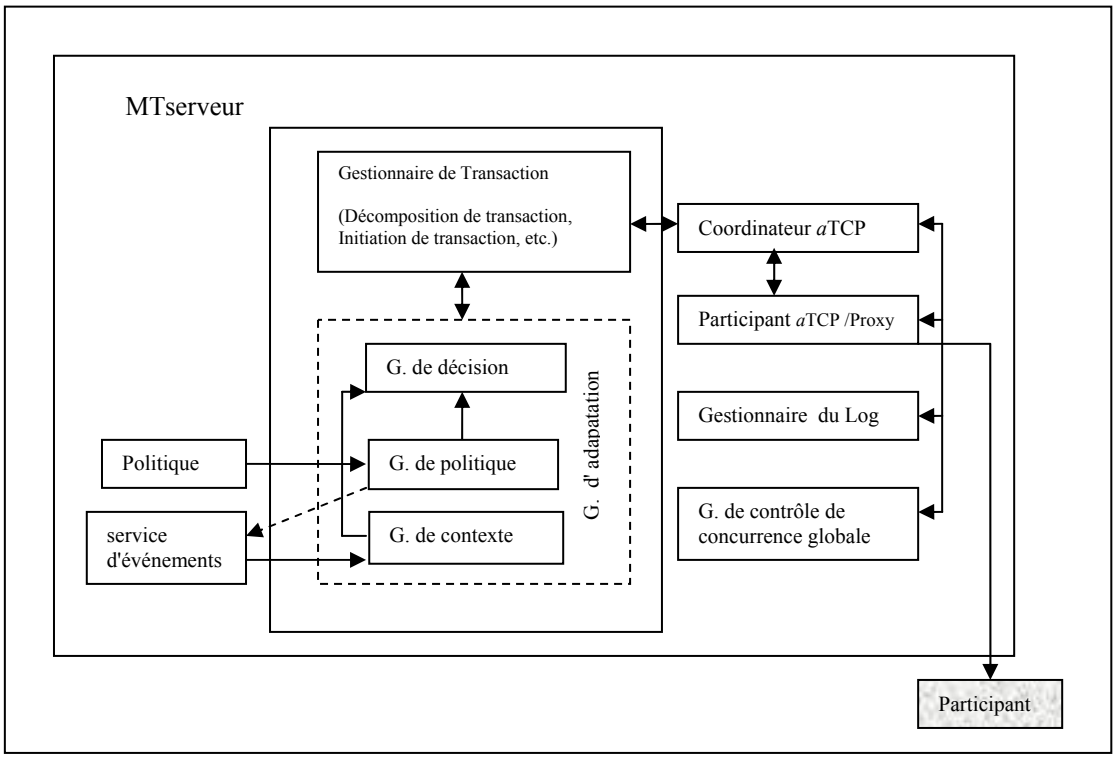


Figure 7.4-Architecture du MTServeur adaptable

Le gestionnaire de contexte ou *(Context Manager)*. Il assure deux rôles:

- Fournir un mécanisme d'identification et d'interrogation des valeurs des variables du contexte.

- Surveiller les changements de ses variables en fonction des alertes du service d'événements *(Event Service)*. Ce dernier interagit avec les différents mécanismes de surveillance.

Les informations sur les changements de contexte sont transmises au gestionnaire de décision ou (*Decision Manager)*.

Le gestionnaire de contexte est responsable de la définition et enregistrement dynamique de nouveaux événements, et la spécification du moment (valeur de tolérance) de déclenchement.

Le service d'événements *(Event Service)*. Offre les mécanismes permettant aux événements d'être manipulés et déclenchés. Au moment de l'interprétation de la politique par le gestionnaire de politique, ce dernier fait appel au service d'événements pour vérifier que tous les événements désirés sont enregistrés.

Le gestionnaire de décision ou (*Decision Manager)*. Est responsable d'évaluer et d'interpréter la politique en se basant sur les informations d'état du contexte fournies par le service d'événements et le résultat de l'analyse du gestionnaire de politique ou *(Policy Manager)* pour déclencher l'application de l'adaptation appropriée. Le traitement de la politique s'effectue au moment de son chargement, au moment de lancement d'une transaction ou à son échec (pour soit la ré-exécuter ou exécuter une alternative) et lorsqu'un changement dans l'état du contexte survient.

Le gestionnaire de décision agit comme un dépôt de politiques et comme un évaluateur de politiques. Il se base sur l'état du contexte et les données extraites de la politique et décide de la stratégie à déclencher (par exemple identifier la sous-transaction alternative à exécuter).

Le résultat du traitement du gestionnaire de décision est traduit en un appel au gestionnaire de transaction qui est chargé d'opérer l'adaptation proprement dite:

- exécuter les sous-trasnactions alternatives,
- changement de rôle,
- modification des paramètres de re-exécution.

Le gestionnaire de politiques ou *(policy Manager).* Est responsable d'analyser la politique (document XML) et de convertir les règles en un format de données qui peut être utilisé par le gestionnaire de décision. Afin de supporter des spécifications dynamiques de politiques, on peut charger et interpréter la spécification à tout moment durant l'exécution. Le gestionnaire de politique se chargera de leur propagation aux autres modules.

7.2.3.4 Fonctionnement de MATrans

A/ Le Transaction Manager:

● décompose T_{AdapT} ,
● définie les sous-transactions de compensation,
● définie les sous-transactions alternatives,
● constitue le contexte de transaction avec toutes les informations et paramètres nécessaires à la gestion et au contrôle.

Le contexte reflète le modèle et il contient :

- identité de transaction
- valeur du timeout,
- paramètres (re-exécutable ou non, vitale ou non, remplaçable ou non, compensable ou non),
- identités des parents,
- référence vers la sous-transaction de compensation,
- référence vers les sous-transactions alternatives,
- identité du site d'exécution,
- référence vers la spécification de politique.

● Maintien deux listes appartenant à la transaction : une liste des transactions compensables et une liste des transactions non compensables. Ces deux listes sont utilisées par le coordinateur durant l'exécution du protocole *a*TCP.

B/ Une fois la décomposition et la constitution du descripteur de contexte terminée, vient la phase d'initiation des sous-transactions sur les sites.

Dans cette phase, le contexte courant est évalué par le gestionnaire de contexte, et le résultat est comparé aux descripteurs de contexte des sous-transactions fournis par le gestionnaire de politique. Le gestionnaire de décision est chargé de cette comparaison et de la désignation des sous-transactions à exécuter. Celles-ci sont alors distribuées par le gestionnaire de transactions vers leurs sites d'exécution. Si le MTServeur est mobile, il charge son MTAgent de cette tâche.

C/ Durant l'exécution des sous-transactions, s'il y en a une qui n'est pas validée, le coordinateur initie le processus d'adaptation qui débute par la réévaluation du contexte courant et l'analyse de la politique (au cas où celle-ci ait été modifiée, par exemple l'utilisateur décide

de se passer de la sous-transaction qui vient d'échouer en la rendant non-vitale) et le processus se poursuit par :

- la reexécution (paramètres éventuellement modifiés),
- le choix d'une alternative,
- le changement de rôle,
- etc.

Les informations du contexte et les listes des participants et des transactions compensables sont mises à jour.

Puis le coordinateur décidera du sort de la transaction en poursuivant l'exécution du protocole *a*TCP.

7.2.3.5 Implémentation du protocole *a*TCP dans MATrans

Les MTServeurs assurent le rôle des *proxies* se trouvant sur chacun des sites participant à l'exécution des sous-transactions (figures 6.3 et 7.4). Le rôle du coordinateur est assuré par un des MTServeurs se trouvant sur l'unité fixe. Si tous les MTServeurs sont mobiles (transaction distribuée sur des unités mobiles; modèle d'exécution 4) alors un MTAgent jouera le rôle de coordinateur (figure 7.5).

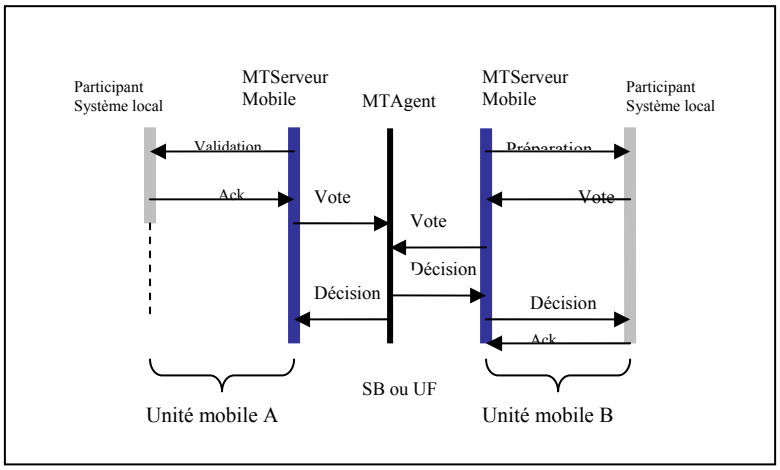


Figure 7.5-Le protocole de validation globale

Les MTServeurs mobiles doivent journaliser en local les informations de contrôle liées à l'exécution de la validation mais aussi sur le MTAgent correspondant afin de palier à l'instabilité et non fiabilité des unités mobiles. Pendant, les déconnexions, la journalisation sur le MTAgent sera interrompue puis reprise à la prochaine reconnexion.

7.2.3.6 Propriétés du protocole *a*TCP

Par rapport aux protocoles étudiés dans les premiers chapitres de cette thèse, *a*TCP possède les propriétés de flexibilité et d'adaptabilité qui permettent:

- d'assurer un degré d'atomicité variable dynamiquement. CO2PC permet l'atomicité stricte et l'atomicité sémantique, mais le choix est fixé dès le début de l'exécution et ne peut pas être modifié par la suite,
- de supporter la déconnexion des participants,
- de préserver l'autonomie des sites en offrant la possibilité de choisir entre la garantie d'une Durabilité par le transfert des journaux des mises à jour des données et le non transfert en

délégant aux participants locaux la tâche de choisir le moment de transférer ces mises à jour (par exemple en regroupant les résultats d'un groupe de transaction, etc.).
- de ne pas imposer une technique de contrôle de concurrence aux sites.

De manière générale, il peut s'adapter à des besoins applicatifs variés en termes de propriétés transactionnelles en tenant compte des caractéristiques de l'environnement mobile. Les coûts d'exécution sont maîtrisés grâce à la spécification de politiques.

7.2.3.7 Le protocole *a*TCP et le contrôle de sérialisation

MATrans compte sur un gestionnaire de contrôle de concurrence globale *(Global Concurrency Controler: GCC)* qui permet d'assurer la sérialisabilité entre les transactions AdapT. Le GCC basé sur la méthode OTM [GRS91, GRS94] expliquée dans le chapitre 6 (section 6.5), est localisé sur un MTServeur fixe comme le coordinateur. Donc les échanges entre le coordinateur et le GCC se font sur le réseau fixe. Les étapes de déroulement du protocole GCC sont imbriquées dans celles du protocole *a*TCP:

■ chaque sous-transaction T_i lit et incrémente la valeur du ticket ;

■ lorsque T_i termine son exécution, le TMServeur correspondant envoie le vote au coordinateur du protocole *a*TCP avec la valeur du ticket attachée ;

■ le coordinateur envoie la valeur du ticket avec les informations sur T_i au GCC qui est connu par tous les coordinateurs ;

■ le GCC crée un noeud pour la transaction adaptable T dans le graphe, s'il s'agit de la première sous-transaction pour T et insère un arc entre T et les autres transactions adaptables (figure 7.5 où seulement deux transactions AdapT; T et T' sont considérées) si leurs sous-transactions accèdent au ticket sur le même site, la direction de l'arc doit suivre l'ordre d'accès du ticket ;

■ aussitôt qu'un cycle est détecté, le GCC avertit le coordinateur. Celui-ci se comporte de la façon suivante:

-- si T_i est vitale alors il demande l'annulation de la transaction globale T.
-- sinon il annule seulement T_i

■ si aucun cycle n'est détecté après l'insertion de tous les arcs, le coordinateur validera T.

La figure 7.6 montre le déroulement de la gestion du graphe de sérialisabilité globale. Un cycle est généré à cause de l'ordre d'accès contradictoire des sous-transactions dans les deux sites aux bases de données A et B.

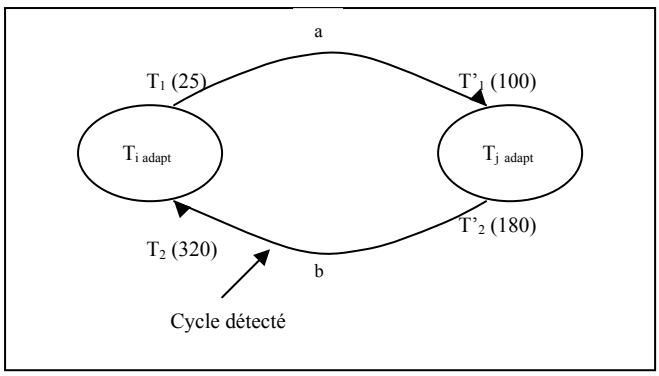


Figure 7.6-Graphe de sérialisation globale

La construction du Graphe de Sérialisation Globale (GSG) ne génère pas de nouveaux messages sur le réseau sans fil, grâce au fait que les participants attachent au message de vote, la valeur du ticket et que les nouveaux messages (envoie du ticket au GCC et notification du GCC au coordinateur) transitent sur le réseau fixe (figure 7.5).

La méthode OTM est avantageuse de par sa simplicité de mise en oeuvre et de par le fait qu'elle n'émet pas d'hypothèse sur les méthodes locales de contrôle de concurrence. Son désavantage réside dans le fait que dans certains cas, elle génère des conflits là où possiblement il n' y en a pas et peut alors entraîner des annulations de transactions à cause des conflits générés par l'accès au ticket. Cependant lorsque seulement l'**atomicité relâchée** est demandée l'annulation va être limitée car les sous-transactions non vitales vont être sacrifiées avant d'en arriver aux transactions globales.

7.2.3.8 Support des déconnexions

Une propriété clé d'un système mobile est la considération de la déconnexion comme un état normal du système. Pendant l'exécution d'une T_{AdapT} , les unités mobiles peuvent subir des déconnexions. Si un participant mobile se déconnecte avant d'avoir envoyé son vote, aucun problème n'est posé pourvu que le vote soit envoyé au coordinateur avant l'expiration du délai de garde assigné à la sous-transaction concernée. Si un participant mobile se déconnecte temporairement après que le vote soit envoyé, la décision du coordinateur est stockée dans le MTAgent et elle est redirigée vers l'unité mobile lors de la prochaine reconnexion. La déconnexion des sites appliquant l'approche pessimiste ou stricte bloque les ressources jusqu'à la prochaine reconnexion.

En ce qui concerne le protocole OTM, les déconnexions ne sont pas un problème car l'ordre de sérialisabilité est défini par l'accès aux tickets. Cependant l'envoi de la valeur du ticket au coordinateur subit les mêmes contraintes que l'envoi du vote puisqu'il y est attaché. Néanmoins, la durée de la déconnexion a une conséquence importante. En effet, la probabilité d'annulation/validation est directement liée au temps qu'une transaction prend pour être sérialisée et qui dépend du délai d'arrivée de la valeur du ticket au GCC.

Une déconnexion prévue peut être gérée entre le tiers mobile et son MTAgent. Il s'agit pour le tiers mobile de transmettre à son MTAgent le journal des transactions actives. Ce dernier se charge de stocker tous les messages concernant ces transactions. Les délais peuvent être renégociés si la date de la prochaine reconnexion est connue. Lors de la reconnexion d'un tiers mobile, le MTAgent lui transmet tous ses messages en attente ce qui lui permet de connaître par exemple les décisions de validation ou non.

7.3 Conclusion

Dans ce chapitre nous avons présenté une architecture pour la mise en œuvre des solutions de flexibilité et d'adaptabilité du traitement transactionnel présentées dans le chapitre 6. L'originalité réside dans le fait d'avoir employé l'approche context-aware *basée sur les politiques* dans le domaine de l'adaptation transactionnelle.

Les principales fonctionnalités de l'architecture proposée sont:

- L'adaptation des transactions selon le contexte et selon les exigences applicatives en termes de propriétés.
- L'adaptation de la propriété d'atomicité.
- L'adaptation à plusieurs modèles d'exécution et notamment les modèles d'exécution répartis.

Enfin, nous pouvons dire que de nombreux outils d'implémentation existent actuellement et que l'architecture proposée est tout à fait réalisable.

Conclusion

If you don't make mistakes, you're not working on hard enough problems. And that's a big mistake.
-- Frank Wilcezek

Dans ce chapitre nous récapitulons nos principales contributions et introduisons quelques perspectives de recherche.

Résumé des contributions

Les domaines d'application et les environnements d'exécution d'aujourd'hui ont des besoins transactionnels variables et qui dépassent les propriétés ACID traditionnelles. Les applications sont également sous une évolution constante, qui signifient que de nouveaux besoins peuvent surgir dans le futur. Cette variabilité exige un environnement flexible d'exécution de transaction. Ces besoins ne sont pas offerts par les solutions transactionnelles courantes où les transactions sont simplement ACID. De plus, les caractéristiques de l'environnement mobile sont contraignantes et très variables et ne peuvent pas être supportées par les solutions traditionnelles qui sont généralement dédiées à un type d'application ou à la prise en charge d'une seule (ou sous-ensemble de) caractéristiques (déconnexion, mobilité ou faiblesse de la bande passante, etc.). Des solutions prenant en charge la variabilité et s'adaptant dynamiquement au contexte sont des sujets de recherche en pleine expansion. Dans cette thèse nous avons proposé d'offrir des solutions aux problèmes de flexibilité des systèmes transactionnels de façon à permettre l'adaptation aux exigences variées et variables des applications en termes de propriétés et aux caractéristiques contraignantes et variables des environnements mobiles.

Après une synthèse bibliographique incluant un nombre important des travaux du domaine nous avons concentré nos travaux principalement sur les problèmes de la validation des transactions mobiles. Ceci nous a amené à :

- la proposition du protocole de validation M2PC *(Mobile Two-phase Commit Protocol)* pour les réseaux mobiles à infrastructure et le protocole de validation A-D2PC *(Ad hoc Decentralized 2PC)* pour les architectures ad hoc. L'étude des performances de M2PC a permis de conclure que le mécanisme de M2PC pour la gestion de la mobilité est réalisable et pourrait être plus intéressant que le routage triangulaire de Mobile-IP en particulier en termes de déploiement mais pas en termes de latence.

- l'étude et l'évaluation des protocoles de validations de transactions en environnement mobile. Cette évaluation, réalisée sur une plateforme de simulation permettant de modéliser les aspects transactionnels et les aspects communication des réseaux mobiles, a permis de

mettre en évidence les indices de performance des différentes approches. De plus, cette étude, nous a confortés dans notre idée de concevoir un protocole flexible capable d'offrir une atomicité adaptable pouvant s'accommoder aux variations du contexte mobile et aux besoins des applications.

- l'étude des problèmes d'adaptation dynamique des modèles et protocoles de validation de transactions mobiles aux variations de contexte et à la variabilité des besoins applicatifs en termes de propriétés transactionnelles qui a aboutit à :

- un modèle de transaction flexible et adaptable *AdapT (Adaptable Transaction Model)* supportant des propriétés adaptables, avec sa formalisation ACTA. Grâce à ce modèle, nous pouvons avoir plusieurs formes d'adaptation:

  i. une adaptation à différents modèles d'exécutions distribuées d'une transaction mobile entre UMs et UFs, en fonction des conditions de l'environnement décrites dans un descripteur de contexte et la sémantique de l'application.

  ii. une adaptation aux conditions courantes par la possibilité de re-exécuter des sous-transactions lorsque certaines conditions sont vérifiées ou par le choix d'alternatives.

  iii. une adaptation des propriétés des plus strictes aux plus relâchées. Concernant l'atomicité, il est possible d'assurer une atomicité stricte, une atomicité sémantique ou une atomicité relâchée.

- un protocole de validation *a*TCP *(Adaptable Transaction Commit Protocol)* qui permet une adaptation aux exigences des applications et du contexte mobile en termes de propriétés transactionnelles et de coût d'exécution. Ce protocole combine certains des aspects positifs des différentes approches de validation étudiées tout en offrant les différents degrés d'atomicité.

- une architecture pour une implémentation context-aware avec l'utilisation des politiques et l'adaptation dynamique en cours d'exécution. Les deux notions, "context-awareness" et "politiques" ouvrent de nouvelles perspectives pour le traitement transactionnel dont le besoin d'adaptation s'est fait sentir depuis les travaux sur les modèles de transactions avancées, mais qui jusqu'ici n'a pas connu un grand succès en pratique.

Nous avons aussi évoqué la sérialisabilité globale et proposé d'adapter la méthode OTM pour sa simplicité de mise en œuvre et du fait qu'elle préserve l'hétérogénéité et l'autonomie des sites participant dans l'exécution d'une transaction mobile.

Perspectives

Pour nos travaux de recherche futurs, nous nous proposons un certain nombre de points dont les principaux sont cités dans cette section.

- **Généralisation de notre approche à d'autres contextes**

Nos contributions sont basées principalement (mis à part le protocole de validation A-D2PC), sur une architecture de type cellulaire. Cependant, nous envisageons de réétudier nos propositions dans les contextes, également variables, des réseaux ad hoc ou pair à pair. Nous considérons que le modèle de transaction AdapT est tout à fait adéquat pour ces environnements grâce à la flexibilité qu'il offre. Par contre, il est nécessaire de modifier le middleware MATrans qui compte sur le support d'unités fixes ; en particulier, il faut repenser la manière de supporter ou déplacer les fonctions du tiers Agent qui supporte les

déconnexions et la mobilité. La coordination globale et la sérialisabilité globale doivent d'être elles aussi adaptées.

- **La sérialisabilité globale et l'adaptation des autres propriétés**

Comme nous l'avons indiqué auparavant, nous avons concentré cette thèse sur la validation atomique dont le relâchement cause automatiquement le relâchement de la sérialisabilité. Les autres propriétés n'ont pas vraiment été traitées, en particulier l'Isolation et la Cohérence (qui doit prendre en compte la réplication des données), mais aussi la Durabilité notamment les problèmes liés aux journaux (transfert, taille, persistance, autonomie et recouvrement). En perspective, il serait intéressant de revenir sur ces questions de contrôle de concurrence et de sérialisabilité [MA05], de cohérence et de recouvrement.

- **Mise en œuvre**

Pour la prise en charge du modèle AdapT et du protocole *a*TCP, nous avons proposé une architecture context-aware. Nous envisageons de valider notre architecture à travers une implémentation. Pour cela, et vu la complexité et le nombre des problèmes traités, nous avons déjà une idée sur un certain nombre de mécanismes et d'approches à explorer dont l'approche *réflexive* [Smi82, Mae87] et le paradigme des *composants* [Szy97, CBCP01, RM07] qui sont des outils qui permettent de mettre ne œuvre des middlewares flexibles [BCP98]. Nous reviendrons aussi sur un certain nombre de points comme la gestion du contexte et la spécification des politiques.

- **Gestion des transactions et mobilité**

A propos de la gestion de mobilité, [DK99] analyse l'impact des stratégies de gestion de mobilité sur la gestion des transactions mobiles. La gestion de transactions inclus les fonctions suivantes: le maintien des informations d'état d'une transaction (dans des tables), la journalisation (fichier log) des informations nécessaire au recouvrement, l'exécution du recouvrement quand cela s'impose, et la validation. Trois options de gestion des transactions mobiles ont été examinées à savoir : la coordination fixée sur une station base d'attachement ou de domiciliation de l'UM (Home BS), fixée au niveau de la station base où se trouve l'UM au moment de l'initiation de la transaction (Anchor BS), ou se déplaçant d'une station base à l'autre en même temps que l'UM (Move). Cette étude a conclue que, d'une part, la performance de la stratégie de gestion de transaction mobile employée dépend de la nature du mouvement de la transaction et est meilleure si elle s'adapte à l'environnement d'exécution. Autrement dit, une transaction simple accompagnée de peu de déplacements devrait s'accommoder à la stratégie (Home BS), si l'UM est souvent en dehors de son réseau mère, la stratégie Home BS est plus adaptée. Une transaction complexe s'accommoderait mieux avec la stratégie Move. D'autre part, la stratégie de gestion de mobilité (en général dans la couche réseau) a aussi son impact sur cette performance. Il est suggéré dans cette étude de prendre en considération les fonctions réseaux (en particulier la gestion de mobilité), lors de l'évaluation des techniques de gestion des transactions mobiles; chose qui n'a pas été faite dans les travaux examinées. Dans le chapitre 4, lors de l'évaluation du protocole M2PC, nous avons testé deux approches de gestions de mobilité (au niveau réseau et au niveau application). Cependant, nous devons élargir cette étude à d'autres techniques de support de mobilité, de types de réseaux sans fil et aussi d'autres protocoles. Concernant le premier aspect, nous pensons qu'il faut pouvoir adapter la stratégie de gestion de transaction mobile au modèle de mobilité des unités mobiles. A notre avis, ces aspects constituent un sujet de recherche intéressant.

- **Retour sur la validation atomique non bloquante et le consensus**

L'utilisation du consensus pour résoudre le problème de validation a fait l'objet comme nous l'avons constaté dans le chapitre 3 d'une riche littérature. On est alors tenté de se demander si la même démarche pourrait être poursuivi dans le contexte mobile d'autant plus que les effets du blocage sont encore plus importants dans ce nouveau contexte. Notons que:

-Les solutions proposées pour l'environnement distribué sont évaluées et comparées presque exclusivement selon des mesures de complexité en messages et temps en moyenne, en pire cas et en meilleur cas. Mais le critère fondamental pour la validation non bloquante est le taux de succès: lorsque les conditions permettent de décider «commit», quelle est la proportion de terminaisons avec «commit» (par rapport aux terminaisons avec «abort») qui est garantie par un algorithme? Nous ne pouvons donc rien dire quand à l'efficacité de ces propositions par rapport aux solutions bloquantes moins coûteuses et qui ont plus de succès en termes d'implémentation notamment (comme le 2PC). Les solutions proposées pour les systèmes distribués supposent des conditions d'exécution non applicables dans l'environnement mobile. Par exemple, la solution de [GL06] n'admet aucune perte de messages et au moins f nœuds fiables ($2f$+1 coordinateurs sont utilisés parmi n participants). Or, ces hypothèses sont difficiles à garantir dans un environnement mobile sujet à de fréquentes déconnexions, pannes de nœuds et partitionnement de réseaux.

-Dans la pratique, l'implémentation des détecteurs de pannes est approchée par l'utilisation de délais de garde *(timeouts)*. Ces délais n'étant déjà pas simples à évaluer dans le cas des systèmes distribués, il est évident qu'avec autant de paramètres variables et de facteurs fluctuants du mobile cette tâche devient encore plus complexe.

-Comme la validation atomique, le problème du consensus nécessite aussi d'être revisité dans le contexte du calcul mobile. Des travaux ont déjà été proposés, [BHM99], mais, nous n'avons pas exploré plus que cela ce paradigme, et nous ne pouvons conclure sur ce sujet.

A notre connaissance, le problème de la validation non bloquante n'est pas encore clos dans le contexte distribué [GL06] et logiquement, les travaux futurs le concernant devraient logiquement prendre en compte les caractéristiques du contexte mobile.

Bibliographie

[AAFZ95] Acharya S., Alonso R., Franklin M., Zdonik S., Broadcast disks: data management for symmetric communication environments, In *Proceedings of the ACM SIGMOD Conference*, 1995, pp. 199–210.

[AAFZ96] Acharya S., Alonso R., Franklin M., Zdonik S., Prefetching from a Broadcast Disk, Dans *Int. Conf. on Data Engineering (ICDE)*, New Orleans, USA, 1996.

[ABC+79] Astrahan M. M., Blasgen M. W., Chamberlin D. D., Gray J. N., Griffiths P. P., King W. F., Lorie R. A., McJones P. R., Mehl J. W., Putzolu G. R., Slutz D. R., Strong H. R., Tiberio P., Traiger I. L., Yost R. A., An overview of System R: A Relational Database System, *IEEE Computer*, vol. 13, n° 4, 1979, pp. 43-55.

[ABG+06] Akdere M., Bilgin C., Gerdaneri O., Korpeoglu I., Ulusoy Ö., Cetintemel U., A Comparison of Epidemic Algorithms in Wireless Sensor Networks, to appear in *Computer Communications*, http://www.cs.bilkent.edu.tr/~oulusoy/compcomm.pdf

[ABP03] Anciaux N., Bouganim L., Pucheral P., Memory Requirements for Query Execution in Highly Constrained Devices, Dans *Int. Conf. on Very Large Databases (VLDB)*, Berlin, Germany, 2003.

[AC04] Al-Houmaily Y., Chrysanthis P., 1-2PC: The One-Two Phase atomic commit protocol, in *the Proc. of the ACM Symp. On Applied computing*, 2004, pp. 684-691.

[AC95] Al-Houmaily Y., Chrysanthis P., Two-Phase Commit in gigabit-networked distributed database, in *Proc. of the 8th int. conf. on parallel and distributed computing systems (PDCS)*, 1995, pp. 554-560.

[ACL87] Agrawal R., Carey M.J., Livny M., Concurrency Control Performance Modeling: Alternatives and Implication, *ACM Transactions on Database Systems*, vol. 12, n° 4, 1987, pp. 609-654.

[ACPJ03] Avancha S., Chakraborty D., Perich F., Joshi A., Data and Services for Mobile Computing, Handbook of Internet Computing, accepted for publication in *"Handbook of Internet Computing", CRC Press Publisher*, 2003, ebiquity.umbc.edu/paper/html/id/35/Data-and-Services-for-Mobile-Computing.

[AK05] Arntsen A., Karlsen R., ReflecTS: a flexible transaction service framework, Proceedings of the 4th workshop on Reflective and adaptive middleware systems, Grenoble, France, November 28-December 02, 2005.

[AK93] Alonso R., Korth H., Database systems issues in nomadic computing, In *Proceedings of the ACM SIGMOD Conference*, Washington, DC, 1993, pp. 388–392.

[AKJ+02] Avancha S., Korolev V., Joshi A., Timothy Finin T., Yesha Y., On Experiments with a Transport Protocol for Pervasive Computing Environments, *Computer Networks*, vol. 40, n°4, 2002, pp. 515-535.

[AP98] Abdallah M., Pucheral P., Validation atomique: état de l'art et perspectives, *Revue Ingénierie des sytèmes d'information (ISI)*, vol. 5, n°6, 1998.

[AS99] André F., Segara M.T., On building a file system for mobile environments using generic services, *Proc. of the12th International Conference on Parallel and Distributed Computing Systems*, Fort Lauderdale, Florida, USA, August 1999.

[Bad98] Badache N., Ordre causal et tolérence aux défaillances en environnement mobile, *Thèse de Doctorat d'Etat en Informatique*, USTHB, 1998.

[Bag98] Baggio A., Cadmium: Resource-aware System Support for Mobile Environments, *Middleware'98 conference*, The Lake District, England, 1998, pp. 15-18.

[Bag99] Baggio A., Objets Distribués Adaptables pour Environnements Mobiles, *Thèse do Doctorat,* Université Pierre & Marie Curie _ Paris VI, Paris VI, 1999.

[BAI93] Badrinath B.R., Acharya A., Imielinski T., Impact of Mobility on Distributed Computations, *Operating Systems Review*, vol. 27, n° 2, 1993, pp. 15-20.

[Bar97] Barbará D., Certification Reports: Supporting Transactions in Wireless Systems, *Proc. 17th Int'l Conf. Distributed Computing Systems*, Baltimore, 1997.

[Bar97] Barbara D., Certification reports: supporting transactions in wireless systems, In *Proceedings of 17th International Conference on Distributed Computing Systems*, USA, 1997, pp. 466–473.

[Bar99] Barga R., A Reflective Framework for Implementing Extended Transactions, *Ph.d. dissertation*, Oregon Graduate Institute of Science and Technology, 1999.

[BB95] Barke A. V., Badrinath B. R., I-TCP: Indirect TCP for Mobile Hosts, In *15th International Conference on Distributed Computing Systems*, Vancouver, British Columbia, Canada, *IEEE Computer Society Press*, 1995, pp. 136-143.

[BBG+05] Bose J.H., Bottcher S., Gruenwald L., Schweppe H., Steenweg T., An Integrated Commit Protocol for Mobile Network Databases, *Proceedings of the 9th International Database Engineering & Application Symposium (IDEAS'05)*, 2005, pp. 244-250.

[BBHS05] Böse J., Bregulla F., Hahn K., Scholz M., Adaptive data dissemination in mobile ad-hoc networks, *Proc. Of INFORMATIK 2005 -Informatik LIVE!*, Band 2, Beiträge der 35. Jahrestagung der Gesellschaft für Informatik e.V. (GI), LNI P-68, 2005, pp. 528-532.

[BBK91] Balter R., Banâtre J.P., Krakowiak S., Construction des Systèmes D'exploitation Répartis, *Collection didactique éditée par INRIA*, France, 1991.

[BBO+03] Bernard G., Ben-Othman J., Bouganim L., Canals G., Defude B., Ferrié J., Gançarski S., Guerraoui R., Molli P., Pucheral P., Roncancio C., Serrano-Alvarado P., Valduriez P., Mobilité et Bases de Données : Etat de l'Art et Perspectives (Partie I et II), Chronique *Technique et science informatiques (TSI)*, vol. 22, n° 3-4, 2003.

[BCF+97] Besancenot J., Cart M., Ferrié J., Guerraoui R., Pucheral P., Traverson B., Les systèmes transactionnels: concepts, normes et produits, livre, *Editions Hermes*, 1997.

[BCRP98] Blair G., Coulson G., Robin P., Papathomas M., An Architecture for Next Generation Middleware, *Proc. of the IFIP International Conference on Distributed Systems Platforms and Open Distributed Processing: Middleware'98*, The Lake District, U.K., 1998.

[BD07] Baldauf M., Dustdar S., A Survey on Context-aware systems**,** *International Journal of Ad Hoc and Ubiquitous Computing*, Vol. 2, No. 4, 2007.

[BDR06] Baldauf M., Dustar S., Rosenberg F., A Survey on Context-Aware Systems, In Int. Journal of Ad Hoc and Ubiquitous Computing, 2006.

[BG81] Bernsein P., Goodman N., Concurrency control in distributed database system, *ACM Computing Survey*, vol. 13, n° 2, 1981.

[BGS92] Breitbart Y., Garcia-Molina H., Silberschatz A., Overview of Multidatabase TransactionManagement, *The VLDB Journal*, vol. 1, n° 2, 1992, pp. 181–240.

[BHG87] Bernstein P.A., Hadzilacos V., Goodman N., Concurrency control and recovery in database system, *Addison Wesley Publishers*, 1987.

[BHM99] Badache N., Hurfin M., Macedo R., A solution for the consensus problem in a mobile environment, In *The 18th IEEE International Performance Computing and Communications*, 1999, pp. 29-35.

[BHPS05] Böse J. H., Hahn K., Pelz L. C., Scholz M., Optimistic Fair Transaction Processing in Mobile Ad-Hoc Networks, Institut für Informatik, Freie Universität Berlin, Report B-05-22, 2005.

[Bia00] Bianchi G., Performance analysis of IEEE 802.11 Distributed Coordination Function, *IEEE journal on selected areas in communications*, vol. 18, n° 3, 2000.

[BLN86] Batini C., Lenzerini M., Navathe S. B., A Comparative Analysis of Methodologies for Database Schema Integration, *ACM Computing Surveys*, vol. 18, n° 4, 1986, pp. 323–364.

[BLRS04] Bobineau C., Labbé C., Roncancio C. L., Serrano-Alvarado P., Comparing transaction commit protocols for mobile environments**.** *Proceedings. 15th International Workshop on Database and Expert Systems Applications*, 30 Aug.-3 Sept. 2004, pp. 673 - 677.

[BN05] For performance evaluation of atomic commit protocols in mobile environment, Boukantar L., Nouali N., *Proc. Of the int. conf. on modelling and simulation, general applications and models in Science, ICMS 2005*, 22-24 November, Marrakech, Morocco, 2005.

[BN97] Bernstein P.A., Newcomer E., Principles of transaction processing, Morgan KaufmannPublishers, Inc. San Francisco, California, 1997.

[BNB05] Bouabache F., Nouali N., Badache N., Architecture d'Adaptation d'une Application de Video on Demand, *SETIT'2005*, 27 - 31 mars, Tunisie, 2005.

[BOH+ 92] Buchmann A., Tamer Özsu M., Hornick M., Georgakopoulos D., Manola F. A., Database Transaction Models for Advanced Applications, chapitre 5, *A Transaction Model for Active Distributed Object Systems*, Morgan Kaufmann Publisher, 1992.

[BOH+92] Buchmann A., Özsu M.T., Hornick M., Georgakopoulos D., Manola F. A., A Transaction Model for Active Distributed Object Systems, chapitre 5, *Database Transaction Models for Advanced Applications*, Morgan Kaufmann Publisher, pp.123-158, 1992.

[BP96] Barga R., Pu C., Reflection on a legacy transaction processing monitor, In *Proceedings Reflection '96*, San Francisco, CA, USA, 1996.

[BP97] Barga R., Pu, C., A reflective framework for implementing extended transactions, In Jajodia S. and Kerschberg L., editors, Kluwer Academic Publisher , Chapter 3, 1997, pp. 63–90.

[BP98] Balasubramaniam S., Pierce B. C., What is a file synchronizer?, Dans *Int. Conf. on Mobile Computing and Networking (MobiCom)*, Dallas, USA, 1998.

[BPA00] Bobineau C., Pucheral P., Abdallah M., A unilateral commit protocol for mobile and disconnected computing, In *Proc. of the 12th int. conf. on parallel and distributed computing Systems (PDCS)*, Las Vegas, USA, 2000.

[BRS92] Breitbart Y., Garcia-Molina H., Silberschatz A., Overview of Multidatabase Transaction Management, *VLDB Journal*, vol. 1, n° 2, 1992, pp. 181-239.

[BS97] Brown K., Singh S., M-TCP: TCP for Mobile Cellular Networks, *ACM Computer Communications Review*, vol. 27, n° 5, 1997, pp. 19-43.

[BST04] Bolchini C., Schreiber F. A., Tanca L., A Context-Aware Methodology for Very Small Data Base Design, *SIGMOD Record*, Vol. 33, No. 1, 2004.

[BT93] Babaoglu O.,Toueg S., Non-Blocking Atomic Commitment, Distributed Systems, Sape Mullender (Ed.), *ACM Press*, 1993.

[BZSH00] Buchholz S., Ziegert T., Schill A., Held A., Transaction Processing in a Mobile Computing Environment with Alternating Client Hosts, In *RIDE 2000, Proc. of the 10th Int. Workshop on Research Issues in Data Engineering: Middleware for Mobile Business Applications and E-Commerce*, 2000, San Diego, CA, USA, pp. 1–8, *IEEE Computer Society Press*, 2000.

[CBCP01] Clarke M., Blair G.S., Coulson G., Parlavantzas N., An efficient component model for the construction of adaptive middleware, In Middleware, Heidelberg, Germany, 2001.

[CBML03] Chung Y., Bhargava B., Mahoui M., Lilien, L., Autonomous Transaction Processing Using Data Dependency in Mobile, In *proc. of The Ninth IEEE Workshop on Future Trends of Distributed Computing Systems (FTDCS'03)*, 2003, pp. 138.

[CCL03] Chlamtac I., Conti M., Liu J. N., Mobile ad hoc networking: imperatives and challenges, *Ad hoc network journal*, vol. 1, n° 1, 2003, pp. 13-64.

[CEM01] Capra L., Emmerich W., Mascolo C., Reflective Middleware Solutions for Context-Aware Applications, Dept. of Computer Science, University College London, 2001.

[CEM03] Capra L., Emmerich W., Mascolo C., CARISMA: Context-Aware Reflective Middleware System for Mobile Applications, *IEEE Transactions on Software Engineering*, vol. 29, n° 10, 2003.

[CFZ01a] Cherniack M., Franklin M. J., Zdonik S., Expressing User Profiles for Data Recharging, *IEEE Personal Communications: Special Issue on Pervasive Computing*, 2001.

[CFZ01b] Cherniack M., Franklin M. J., Zdonik S., Expressing User Profiles for Data Recharging, In *IEEE Personal Communications: Special Issue on Pervasive Computing*, 2001, pp.6–13.

[CGFZ03] Cherniack M., Galvez E.F., Franklin M.J., Zdonik S., Profile-Driven Cache Management, *International Conference on Data Engineering (ICDE)*, Bangalore, India, 2003.

[Che06] Chehbour F., Evaluation de performances de protocols de validation atomique pour les environnements mobiles et sans fil. Thèse de Magister en informatique de l'université des science et technologies Houari Boumédiène (USTHB), 2006.

[CHR01] Cerqueira R., Hess C.K., Romn M., Campbell R.H., Gaia: A Development Infrastructure for Active Spaces, In *Workshop on Application Models and Programming Tools for Ubiquitous Computing (held in conjunction with the UBICOMP 2001)*, 2001.

[Chr91] Chrysanthis P. K., ACTA: A Framework for Modeling and Reasoning about Extended Transactions, *PhD thesis*, University of Massachusetts, Amherst,USA, 1991.

[Chr93] Chrysanthis P. K., Transaction Processing in Mobile Computing Environment, *Proc. Of the IEEE Workshop on advances in parallel and distributed systems*, Princeton, New Jersy, 1993, pp. 77-83.

[CL99] Chen G.C., Lee S.Y., An analytic model for performance analysis of concurrency control strategies in mobile environments, *The computer journal*, vol. 42, n° 6, 1999, pp.511-521.

[CN03] Chahbour F., Nouali N., Disconnection handling in mobile Internet, *Proc. Of the 10th ISPE int. conf. on concurrent engineering: research and applications,* ISBN: 90 5809 625 4, Carlton Madeira Hotel, Madeira Island –Portugal, 26-31 July, 2003.

[CN05] Chahbour F., Nouali N., Gestion de la mobilité au niveau de la couche application: état de l'art, *SETIT'2005*, Sousse, Tunisie, 27 - 31 mars 2005.

[CNZ05] Chahbour F., Nouali N., Zeraoulia K., Fast Handoff for Hierarchical Mobile SIP Networks, *Proc. Of the WEC'05, the third World Enformatika Conference*, vol. 5, Istanbul, Turquie, 27-29 Avril, 2005.

[CR90] Chrysanthis P.K., Ramamritham K., ACTA: A Framework for Specifying and Reasoning about Transaction Structure and Behavior, *ACM SIGMOD*, 1990.

[CR94] Chrysanthis P. K., Ramamritham K., Synthesis of extended transaction models using ACTA, *ACM Transactions on Database Systems (TODS)*, vol. 19, Issue 3, September 1994, pp. 450 – 491.

[CSA98] Chrysanthis P., Samaras G., Al-Houmaily Y., Recovery and Performance of Atomic Commit Processing in Distributed Database Systems, Chapter 13, In *Recovery Mechanisms in Database Systems*, Kumar V. and Hsu M. eds., Prentice-Hall, 1998.

[CT96] Chandra T., Toueg S., Unreliable Failure Detectors for Reliable Distributed Systems, *Journal of the ACM*, vol. 34, n° 1, 1996. Une version préliminaire est parue dans *ACM International Symposium on Principles of Distributed Computing (PODC)*, 1991.

[Dav05] David P. C., Développement de composants Fractal adaptatifs : un langage dédié à l'aspect d'adaptation, *Thèse de Doctorat*, Spécialité Informatique, Université de Nantes UFR Sciences et Techniques, N° ED 366-204, 2005.

[DCK+94] Douglis F., Caceres R., Kaashoek F., Li K., Marsh B., Tauber J.A., Storage alternatives for mobile computers, in *The 1st Symp. on Operating Systems Design and Implementation (OSDI)*, Monterey, CA, 1994, pp.25-37.

[Dey01] Dey A. K., Understanding and Using Context Future Computing Environments Group, College of Computing & GVU CenterGeorgia Institute of Technology, 2001.

[DFBC96] Davies N., Friday A., Blair G. S., Cheverst K. W. J., Distributed Systems Support for Adaptive Mobile Applications, *ACM-Baltzer Mobile Networks and Nomadic Applications*, vol. 1, n°4, 1996, pp. 399-408.

[DG00] Dirckze R. A., Gruenwald L., A Pre-Serialization Transaction Management Technique for Mobile Multidatabases, *Mobile Networks and Applications (MO-NET)*, vol. 5, n°4, 2000.

[DG98] Dirckze R. A., Gruenwald L., A Toggle transaction management technique for mobile multidabases, In Gardarin G., French J., Pissinou N., Makki K., Bougamin, editors, *Proc. of the 7th ACM CIKM Int. Conf. on Information and Knowledge management*, 1998, Bethesda, Maryland, USA, ACM press, New York, 1998, pp. 371-377.

[DGH+87] Demers A., Greene D., Hauser C., Irish W., Larson J., Shenker S., Sturgis H., Swinehart D., Terry D., Epidemic Algorithms for Replicated Database Maintenance, In *Proceedings of the Sixth ACM Symposium on Prinples of Distributed Computing*, 1987, pp.1-12.

[DH95] Dunham M. H., Helal A. S., Mobile Computing and Databases: Anything New?, In *ACM SIGMOD Record*, ACM Press, 1995, pp. 5-9.

[DHB97] Dunham M.H., Helal A., Balakrishanan S., A mobile Transaction Model That Captures Both The Data And Movement Behaviour, *ACM/Baltzer Journal on Special Topics in Mobile Networks and Applications,* vol. 2, 1997, pp. 149–162.

[DK98] Dunham M. H., Kumar V., Location Dependent Data and its Management in Mobile Databases, Dans *Int. DEXA Workshop on Mobility in Databases and Distributed Systems,* Vienna, Austria, 1998.

[DK99] Dunham M. H., Kumar V., Impact of Mobility on Transaction Management, Dans *Int. Workshop on Data Engineering for Wireless and Mobile Access (Mo-biDE)*, Seattle, USA, 1999.

[DLB+02] Das S. K., Lee E., Basu K., Kakani N., Sen S., Performance Optimization of VoIP Calls over Wireless Links using H.323 *Protocol, Proc. of the 2002 INFOCOM*, 2002, pp. 1386-1394.

[DMW 01] Ding Z., Meng X., Wang S., O2PC-MT: A Novel Optimistic Two-Phase Commit Protocol for Mobile Transactions, In *Proc. of the 12th intl. conf., DEXA 2001*, Munich, Germany, LNCS, vol. 2113/2001, Springer-Verlag Heidelberg, 2001, p. 846.

[DVCK99] Datta A., VanderMeer D. E., Celik A., Kumar V., Broadcast Protocols to Support Efficient Retrieval from Databases by Mobile Users, *ACM Transactions on Database Systems (TODS)*, vol. 24, n° 1, 1999.

[ECDF01] Efstratiou C., Cheverst K., Davies N., Friday A., An Architecture for the Effective Support of Adaptive Context-Aware Applications, In *Proceedings of the Second International Conference on Mobile Data Management (MDM 2001)*, 2001.

[EFDC02] Efstratiou C., Friday A., Davis N., Cheverst K., Utilising the Event Calculus for Policy Driven Adaptation on Mobile Systems, In *3rd International Workshop on Policies for Dis-tributed Systems and Networks (POLICY'02)*, Monterey, California, USA. 2002.

[EK06] Eidsvik A.K., Karlsen R., Mobile middleware and transaction services, NIK-06, *The Norwegian Information Technology Conference.*, Arrangør, Norway NIK'06, November 2006.

[EKB06] Eidsvik A.K., Karlsen R., Blair G., Grace P., Transaction Service Discovery in Mobile Environments, *Proceedings of the 20th International Conference on Advanced Information Networking and Applications, (AINA'06)*, vol. 2, 2006, pp. 224 –228.

[Elm92] Elmagarmid A.K., Database Transaction Models for Advanced Applications, *Morgan Kauffman publisher*, 1992.

[ELR90] Elmagarmid A. K., Leu Y., Rusinkiewics M., A Multidatabase Transaction Model for INTERBASE, Dans *Int. Conf. on Very Large Databases*, 1990.

[ELR90] Elmagarmid A.K. Leu Y., Rusinkiewics M., A multidatabase transaction model for INTERBASE, in *16th Int. Conf. on Very Large Databases (VLDB)*, Brisbane, Australia, 1990, pp. 507-518.

[ES00] Eyers T., Schulzrinne H., Predicting Internet Telephony Call Setup Delay, *Proc. of 1st IP-Telephony Wksp.*, Berlin, Germany, 2000.

[Fas02] FastObjects by Poet. FastObjects j2 http://www.fastobjects.com/, 2002.

[FLP85] Fisher M., Lynch N., Paterson M., Impossibility of Distributed Consensus with One Faulty Process, *Journal of the ACM*, vol. 32, n° 2, 1985.

[Fra01] Franklin M., Challenges in Ubiquitous Data Management, *Proc. Informatics*, 2001.

[Fra96] Franklin M. J., Special Issue on Data Dissemination, *editor. IEEE Technical Committee on Data Engineering*, 1996.

[FZ94] Forman G. H., Zahorjan J., The challenges of mobile computing, *Computer*, vol. 27, n° 4, 1994, pp. 38 -47.

[FZ96] Franklin M., Zdonik S., Dissemination-Based Information Systems, *IEEE Data Engineering Bulletin*, vol. 19, n° 3, 1996.

[Gar02] Garg K., Wireless network evolution 2G to 3G, *Edition Prentice Hall PTR*, 2002.

[Gar83] Garcia-Molina H., Using Semantic Knowledge for Transaction Processing in a Distributed Database, *ACM Transactions on Database Systems (TODS)*, vol. 8, n°2, 1983, pp.186-213.

[GB01] Gruenwald L., Banik S.M., A Power Aware Technique to Manage Real-Time Database Transactions in Mobile Ad-hoc Networks, in *Proc. Of the 4TH Int. Workshop on Mobility in Data bases and Distributed Systems*, 2001, pp. 570-574.

[GBE+ 00] Güting R.H., Böhlen M.H., Erwig M., Jensen C.S., Lorentzos N.A., Schneider M., Vazirgiannis M., A Foundation for Representing and Querying Moving Objects, *ACM Transactions on Database Systems (TODS)*, vol. 25, n° 1, 2000.

[GGG01] Gupta A., Gore M.M., Gupta N., Team Transaction: A New Transaction Processing Model for Mobile Ad Hoc Networks, *Technical Report*, Dept of CSE, IIT-Kanpur, Nov. 2001.

[GK87] Garcia-Molina H., Kenneth S., SAGA, *Proc. of ACM SIGMOD 1987, Int. Conf. on Management of Data,* 1987, pp. 249-259.

[GKW+02] Ganesan D., Krishnamachari B., Woo A., Culler D., Estrin D., Wicker S., An Empirical Study of Epidemic Algorithms in Large Scale Multihop Wireless Networks, University of California Los Angeles, Computer Science Department, *Technical Report* UCLA/CSD-TR-02-0013, March, 2002. http://www.intel-research.net/Publications/Berkeley/120520021022_19.pdf

[GL06] Gray J., Lamport L., Consensus on transaction commit, *ACM Transactions on Database Systems (TODS)*,vol. 31 , n°1, 2006, pp. 133–160.

[Glo01] GLOMOSIM, Glomosim : Global Mobile Information Systems Simulation Library, UCLA Parallel Computing Loboratory, 2001, http://pcl.cs.ucla.edu/projects/glomosim/.

[GMPG00] Goff T., Moronski J., Phatak D. S., Gupta V., Freeze-TCP: A True End-to-End TCP Enhancement Mechanism for Mobile Environments, In *INFOCOM*, vol. 3, pp. 1537-1545, Tel Aviv, Israel, March 2000.

[GR93] Gray J., Reuter A., Transaction processing: Concepts and Techniques, USA: Morgan Kaufman, 1993.

[GR98] Gray J., Notes on Database Operating Systems. Operating Systems: An Advanced Course, *Lecture Notes in Computer Sciences*, vol. 60, Springer Verlag, 1998.

[Gra81] Gray J., The Transaction Concept: Virtues and Limitations, *In proc. Of the 7th VLDB*, Cannes, 1981, pp. 144-154.

[Gre03] Greis M., Tutorial for NS simulator, VINT group, 11 March 2003, http://www.isi.edu/nsnam/ns/tutorial/

[GRS91] Georgakopoulos D., Rusinkiewicz M., Sheth, A., On serializability of multidatabase transactions through forced local conflicts, In Cercone N., Tsuchiya M., editors, *Proc. Of the 7th IEEE int. Conf. On Data Engineering, ICDE'91*, Kobe, Japan, 1991, pp. 314-323, *IEEE Computer Society Press*, Los Alamitos, CA, 1991.

[GS87] Garcia-Molina H., Salem K., Sagas, in *ACM SIGMOD Int. Conf. on Management of Data*, San Francisco, USA, 1987, pp. 249-259.

[GS95] Guerraoui R., Schiper A., The Decentralized Non-Blocking Atomic Commitment Protocol, *IEEE International Symposium on Parallel and Distributed Processing (SPDP)*, 1995.

[GSC96] Garcia-Solaco M., Saltor F., Castellanos M., Semantic Heterogeneity in Multidatabase Systems, In *Bukhres O. A., Elmagarmid A. K., editors, Object-Oriented Multidatabase Systems, A Solution for Advanced Applications*, chapter 5, Prentice Hall, Eaglewoods Cliffs, NJ, 1996, pp. 129–202.

[Gue02] Guerraoui R., Non-blocking atomic commit in asynchronous distributed systems with failure detectors, *Distributed Computing*, vol. 15, n° 1, Springer Berlin, Heidelberg publisher, 2002, pp:17-25.

[Gue95] Guerraoui R., Revisiting the Relationship between Non-Blocking Atomic Commitment and Consensus Problems, *Proceedings of the International Workshop on Distributed Algorithms (WDAG), LNCS*, Springer Verlag, 1995.

[GW82] Garcia-Molina H., Wiederhold G., Read-only transactions in a distributed database, *ACM Transactions on Database Systems*, vol. 7, n° 2, 1982, pp. 209–234.

[HAE00] Holliday J., Agrawal D., El Abbadiu A., Exploiting Planned Disconnections in Mobile Enrironments. In RIDE 2000, *Proc. of the 10th Int. Workshop on Research in Data Engineering: Middleware for Mobile Business Applications and E-Commerce*, San Diego, CA, USA, February 27-28, 2000, pp. 25-30, *IEEE Computer Society Press*, 2000.

[Hae84] Haerder T., Observations on optimistic concurrency control schemes, *Information Systems*, vol. 9, n° 2, 1984, pp. lll-120.

[HBG05] Hogie L., Bouvry P., Guinand F., An overview of Manets Simulation, *Electronic Notes in Theoritical Computer Science*, vol. 150, n°1, 2006, pp. 81-101, *Proceedings of the First International Workshop on Methods and Tools for Coordinating Concurrent, Distributed and Mobile Systems (MTCoord 2005)*.

[HC 00] Al-Houmaily Y. J., Chrysanthis P. K., An Atomic Commit Protocol for Gigabit-Networked Distributed Database Systems, Journal of Systems Architecture, *The EuroMicro Journal*, vol. 46, n° 9, pp. 809-833, June 2000.

[HH94] Huston L.B., Honeyman, P., Peephole log Optimization, in *IEEE Workshop on Mobile Computing Systems and Applications*, Santa Cruz, USA, 1994.

[HRG97] Haritsa J., Ramamritham K., Gupta R., Characterization and optimisation of commit processing performance in distributed database systems, In *ACM SIGMOD Int. Conf. on Management of Data*, Tucson, Arizona, 1997.

[HS93] Hjelsvold R., Sandsta O., Two Phase Locking and Timestamp Ordering, A Comparison Study, *Technical report*, Norwegian Institute of Technology, 1993.

[HSAA03] Holliday J., Steinke R., Agrawal D., El Abbadi A., Epidemic Algorithms for Replicated Databases, *IEEE Transactions on Knowledge and Data Engineering*, vol. 15, no. 5, 2003, pp. 1218-1238.

[HSW94] Huang Y., Sistla P., Wolfson O., Data replication for mobile computers, In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, 1994, pp. 13–24.

[IB93] Imielinski T., Badrinath B.R., Data Management for Mobile Computing, SIGMOD RECORD, vol. 22, n° 1, March 1993.

[IB94] Imielinski T., Badrinath B.R., Mobile wireless computing: challenges in data management, *Communications of the ACM*, vol. 37, n°10, 1994, pp. 18–28.

[IBM02a] IBM Software Products, DB2 Everyplace, 2002, http://www-3.ibm.com/software/data/db2/everyplace/

[IBM02b] IBM Software Products, WebSphere Everyplace Access, 2002, http://www-3.ibm.-com/software/pervasive/products/mobile_apps/ws_everyplace_access.shtml

[Inf99] Informix Software, Inc. Informix Cloudscape Application Synchronization, November 1999.

[ISO92] ISO, Open System Interconnection, Distributed Transaction Processing (OSI-TP) Model, ISO IS 10026, 1992.

[Jan05] Jang S., Implementation of Context-Aware Application Model in Smart Environments, Accepted in *parial fulfilment of the requirements for the degree of Doctor of Philosophy*, Department of Information and Communications Gwangju Institute of Science and Technology, Republic of Korea, 2005, 131p.

[JBE95] Jing J., Bukhres O., Elmagarmid A.K., Distributed lock management for mobile transactions, in *15th Int. Conf. on Distributed Computing Systems (ICDCS)*, Vancouver, Canada, 1995, pp.118-126.

[JHE99] Jing J., Helal A.S., Elmagarmid A. K., Client-Server Computing in Mobile Environments, *ACM Computing Surveys*, vol. 31, n° 2, 1999.

[JK04] Jakobsen A. A., Karlsen R., ReflecTS, A Reflective Transaction Service Framework for Open Applications, *Technical Report*, TR 2004-51, Dep. Of CS. Univ. of Tromsø, Norvege, 2004.

[Kar03] Karlsen R., An adaptive transactional system - framework and service synchronization, In *International Symposium on Distributed Objects and Applications (DOA)*, Catania, Sicily, November 2003, LNCS, vol. 2888/2003, pp:1208-1225.

[KC03] Keeney J., Cahill V., Chisel: A Policy-Driven, Context-Aware, Dynamic Adaptation Framework, In *Proceedings of the Fourth IEEE International Workshop on Policies for Dis-tributed Systems and Networks (POLICY 2003)*, 2003, pp. 3-14.

[KCGS95] Kim W., Choi I., Gala S., Scheevel M., On Resolving Schematic Heterogeneity in Multidatabase Systems, In *Kim W., editor, Modern Database Systems*, chapter 26, pp. 521–550, *ACM Press*, New York, NJ, 1995.

[KDDS00] Kumar V., Dash K., Dunham M.H., Seydim A.Y., A timeout-based mobile transaction commitment protocol, In *ADBIS-DASEAA 2000, Advances in DB and Information Systems in cooperation with ACM SIGMOD*, Prague, Czech republic, 2000.

[Kee04] Keeney J., Completely Unanticipated Dynamic Adaptation of Software, *Ph.D. Thesis*, Department of Computer Science, Trinity College Dublin, Dublin, Ireland, 2004.

[KFCW04] Kawamura S., Fu H., Choi M., Wu S., End-to-End Mobility Management: A Two-Phase Deployment Scheme for Personal Use, *International Conference on Wireless Networks (ICWN-04)*, Las Vegas, Nevada, USA, 2004.

[KJ03] Karlsen R., Jakobsen A. B. A., Transaction service management an approach towards a reflective transaction service, *In 2nd InternationalWorkshop on Reflective and Adaptive Middleware*, Rio de Janeiro, Brazil, June 2003.

[KK00] Ku K. I., Kim Y. S., Moflex Transaction Model for Mobile Heterogeneous Multidatabase Systems, In *RIDE 2000, Proc. of the 10th Int. Workshop on Research Issues in Data Engineering: Middleware for Mobile Business Applications and E-Commerce*, February 27–28, 2000, San Diego, CA, USA, pp. 39–46, *IEEE Computer Society Press*, 2000.

[KKL+04] Kim W., Kim M., Lee K., Yu C., Lee B., Link Layer Assisted Mobility Support Using SIP for Real-time Multimedia Communications, *International Workshop on Mobility Management and Wireless Access (MobiWac 04) in conjunction with Mobicom 2004*, Philadephia, USA, 2004.

[KKS98] Kuo T. W., Kao Y. T., Shu L., A two-version approach for real time concurrency control and recovery, In *Proceeding of the Third IEEE International High Assurance Systems Engineering Symposium*, Washington, DC, November, 1998.

[KLS90] Korth H.F., Levy E., Silberschatz A., A Formal Approach to Recovery by Compensating Transactions, in *proc. of Int. Conf. on Very Large Databases (VLDB)*, Brisbane, Australia, August 1990.

[KMS05] Korpipää P., Malm E. J., Salminen I., Rantakokko T., Kyllönen V., Känsälä I., Context management for end user development of context-aware applications. International Conference On Mobile Data Management, *Proceedings of the 6th int. conf. on Mobile data management*, Ayia Napa, Cyprus, 2005, pp. 304 – 308.

[KOU04] Koubaa A., Introduction à L'Evaluation De Performance Des Systèmes Informatiques et de Communication, ENSEM-INPL, 2004, http://www.loria.fr/~akoubaa/ENSEM/.

[KPDS02] Kumar V., Prabhu N., Dunham M. H., Seydim A. Y., TCOT-A Timeout-Based Mobile Transaction Commitment Protocol, *IEEE Transactions on Computer*, vol. 51, n° 10, 2002, pp. 1212-1218.

[KR81] Kung H.T., Robinson J.T., On optimistic methods for concurrency control, *ACM Transactions on Database Systems*, vol. 6, n° 2, 1981, pp. 213–226.

[Kra03] Krakowiak S., Patrons et canevas pour l'intergiciel, Quatrième École d'été sur les Intergiciels et sur la Construction d'Applications Réparties, Autrans, France, August 2003.

[KRB91] Kiczales G., Rivieres J., Bobrow D., The Art of the Metaobject Protocol, *MIT Press*, 1991.

[KS92] Kistler J., Satyanarayanan M., Disconnected operation in CODA file system, *ACM Transactions on Computer Systems*, vol. 10, No. 1, february 1992, pp. 3-25.

[KU99] Kayan K., Ulusoy O., An Evaluation of Real-Time Transaction Management Issues in Mobile Database Systems, *The Computer Journal Special Issue on Mobile Computing*, vol. 42, n° 6, 1999, pp. 501-510.

[KYL04] Kim S., Yang S. 0., Lee S., Maintaining mobile transactional consistency in hybrid broadcast environments, *Acta Informatica Publisher*, Springer-Verlag, vol. 41, n° 2-3, December 2004, pp. 65 - 81.

[LAE94] Liu L., Agrawal D., El Abbadi A., The performance of Two-Phase Commit Protocols in the presence of site failures, *Technical Report* TRCS94-09, Depart. of C.S., Univ. of California, Santa Barbara, 1994.

[LAE98] Liu M.L., Agrawal D., El Abbadi A., The performance of two phase commit protocols in the presence of site failures, *Distributed and Parallel Databases*, vol. 6, 1998, pp. 157-182.

[Lee04] Lee S. K., Energy Efficient Transaction Processing in Mobile Broadcast Environments, *ADBIS*, 2004, pp. 409-422.

[LeM03] Le Mouël F., Environnement adaptatif d'exécution distribuée d'applications dans un contexte mobile, *Thèse de doctorat* de l'université de Rennes 1, France, Mention informatique, soutenue en décembre 2003.

[LFH05] Le D., Fu X., Hogrefe D., A Review of Mobility Support Paradigms for the Internet, *Tech. Rep.*, n° IFI-TB-2005-01, C.S. Institute, Georg-August University, January 2005.

[LH02] Lee M., Helal S., HiCoMo: High Commit Mobile Transactions, *Kluwer Academic Publishers Distributed and Parallel Databases (DAPD)*, vol. 11, n° 1, 2002.

[LHK03] Lee S. K., Hwang C. S., Kitsuregawa M., Using Predeclaration for Efficient Read-Only Transaction Processing in Wireless Data Broadcast, *IEEE Trans. Knowl. Data Eng.*, vol. 15, n° 6, 2003, pp. 1579-1583.

[LHLH98] Lam K. W., Hyuk Son S., Lee V. C. S., Hung S. L., Using Separate Algorithms to Process Read-Only Transactions in Real-Time Systems, *IEEE Real-Time Systems Symposium*, 1998, pp. 50-59.

[LHY99] Lee S. K., Hwang C. S., Yu H. C., Supporting Transactional Cache Consistency in Mobile Database Systems, *MobiDE*, 1999, pp. 6-13.

[LK04a] Lee S. K., Kim S. S., Efficient Transaction Processing in Mobile Data Broadcast Environments, *DASFAA*, 2004, pp. 750-761.

[LK04b] Lee S. K., Kim S. S., Performance Evaluation of a Predeclaration-Based Transaction Processing in a Hybrid Data Delivery, *Mobile Data Management*, 2004, pp. 266-273.

[LKS91] Levy E., Korth H. F., Silberschatz A., An optimistic commit protocol for distributed transaction management, In *Proc. OF ACM SiGMOD Int. Conf. on Management of Data*, Denver, Colorado, U.S., 1991, pp. 88-97.

[LLK04] Lee V. C. S., Lam K. W., Kuo T. W., Efficient validation of mobile transactions in wireless environments, *Journal of Systems and Software*, vol. 69, n° 1-2, 2004, pp. 183-193.

[LLK99] Lee V.C.S., Lam K.Y., Kao B., Priority scheduling of transactions in distributed real-time databases, *Real-time Systems*, vol. 16, n° 1, 1999, pp. 31–62.

[LLSC02] Lee V. C. S., Lam K. W., Son S. H., Chan E. Y. M., On Transaction Processing with Partial Validation and Timestamp Ordering in Mobile Broadcast Environments, *IEEE Trans. Computers*, vol. 51, n° 10, 2002, pp. 1196-1211.

[LS94] Lu Q., Satynarayanan M., Isolation-Only Transactions for Mobile Computing, *ACM Operating Systems Review*, vol. 28, n° 2, 1994.

[LS95] Lu Q., Satyanarayanan M., Improving Data Consistency in Mobile Computing Using Isolation Only Transaction, *Proceedings of The 5th Workshop in Operating Systems,* Orcas Island WA, May 4-5 1995.

[LSLH98] Lam K.W., Son S.H., Lee V.C.S., Hung S.L., Using separate algorithms to process read-only transactions in real-time systems, In *Proceedings of IEEE Real-Time Systems* Symposium, 1998.

[LYL96] Lam K. W., Yau W. C., Lee V. C. S., Applying Similarity in Concurrency Control for Real-Time Database Application, *DEXA*, 1996, 143-152.

[MA 05] Martinez J., Alvarado M., Concurrency control for nested transactions based on autonomous agents, *Intelligencia Artificial*, vol. 9, 2005.

[MA01] Mouël F. L., André-AeDEn F., un cadre générale pour une distribution adaptative des applications en environnements mobiles, *Revue électronique sur les réseaux et l'informatique répartie*, n° 11, mars 2001.

[Mad98] Madria S.K., Transaction models for mobile computing, in *Proceedings of 6th IEEE Singapore International Conference on Network, World Scientific*, Singapore, 1998.

[Mae87] Maes P., Concepts and experiments in computational reflection, Technical Report 272, Cambridge, 1982, In *Proceedings of theConference of Object-Oriented Programming Systems,Languages and Applications (OOPSLA)*, oct 1987.

[Mat88] Mattern F., Virtual Time and Global States in Distributed Systems, In Cosnard, Quinton, Raynal, and Roberts, editors, *Proc. of the International Conference on Parallel and Distributed Computing, North-Holland*, 1988, pp. 215-226.

[MB01] Madria S. K., Bhargava B., A Transaction Model for Improving Data Availability in Mobile Computing, *Kluwer Academic Publishers Distributed and Parallel Databases (DAPD)*, vol. 10, n° 2, 2001.

[Med95] Medjahed B., Transactions imbriquées et contrôle de concurrence, Thèse de Magister en Informatique, USTHB, Alger N° 03/95-M/IN, 1995.

[MHL+89] Mohan C., Haderle D., Lindsat B., Pirahesh H., Schwarz P., ARIES: a transaction Recovery Method Supporting Fine-Granularity Locking and Partial Rollbacks Using Write-Ahead Logging, *ACM Transactions on Database Systems*, Jan. 1989.

[MHL+92] Mohan C., Haderle D.J., Lindsay B.G., Pirahesh H., Schwarz P.M., ARIES : a transaction recovery method supporting _ne-granularity locking and partial rollbacks using write-ahead logging, *ACM Transactions on Database Systems (TODS)*, vol. 17, 1992, pp. 94-162.

[Mic01] Microsoft Corp., http://msdn.microsoft.com/library/, 2001.

[MLO86] Mohan C., Lindsay B., Obermarck R., Transaction management in the R*distributed database management system, *ACM transactions on database systems*, vol. 11, n° 4, 1986, pp. 378-396.

[MV00] Momin K.A., Vidyasankar K, Flexible integration of optimistic and pessimistic concurrency control in mobile environments, in *ADBIS-DASFAA Symp. on Advances in Databases and Information Systems*, vol. 1884, LNCS, Prague, Czech Republic, 2000, pp. 346-353.

[Nar 94] Narasayya V. R., Distributed Transactions in a Mobile Computing System, *IEEE Workshop on Mobile Computing Systems and Applications*, 1994.

[NB01] Gestion des transactions en environnement mobile, Nouali-Taboudjemat N., Badache N., publié dans la revue *RIST*, Vol. 11 N°21, ISSN 1111-0015, 2001.

[NB03] Interaction avec une base de données via la technologie WAP 2.0, Nouali N., Boukantar L., *Proceedings of the IEEE-(internationale) setit'2003*, Sousse, Tunisie, mars 2003.

[NBD07] Nouali-Taboudjemat N., Boukantar L., Drias H., Performance evaluation of Atomic Commit Protocols for mobile transactions, *Inderscience, Int. J. Intelligent Information and Database Systems*, Vol.1, No.2, 2007.

[NDD04] Nouali N., Drias H., Doucet A., Executing the Two-phase Commit Protocol in Mobile Wireless Environment, *Special session on wireless computing, International Conference on Advances in Computer Science and Technology (ACST 2004)*, St. Thomas, Virgin Islands, USA, 2004, pp. 315-320.

[NDD05a] Nouali N., Doucet A., Drias H., A Two-Phase Commit Protocol for Mobile Wireless Environment, In *Proc. Sixteenth Australasian Database Conference (ADC2005)*, Newcastle, Australia. CRPIT, vol. 39, Williams, H. E. and Dobbie, G., Eds., ACS, 2005, pp.135-144.

[NDD05b] Nouali N., Drias H., Doucet A., Revisiting distributed protocols for mobility at the application layer, *International conference on parallel and distributed systems*, In *proceedings of the 3rd World Enformatika Conference, WEC'05*, in Istanbul, Turkey, April 27-29, 2005.

[NDD05c] Nouali N., Drias H., Doucet A., Mobility Management Support for Transactional Applications, *5th Workshop on **A**pplications and **S**ervices in **W**ireless **N**etworks June 29th - July 1st, ASWN 2005*, Paris, France, 2005.

[NDD06a] Nouali N. Drias H., Doucet A., A Mobility-aware Two-Phase Commit Protocol, *The international Arab Journal of Information Technology*, vol. 3, n° 1, 2006, pp.1-8.

[NDD06b] Nouali N., Drias H., Doucet A., Protocols for Committing Mobile Transactions, *The international Arab Journal of Information Technology*, vol.3, n°2, 2006, pp.134-143.

[NDD06c] Nouali N., Drias H., Doucet A., A Two-phase Protocol for Transaction Commitment in Mobile Ad-hoc Environment, *1st International Conference on Multidisciplinary Information Sciences and Technologies, InSciT2006*, Merida, Spain, October 25-28, 2006.

[NP94] Norvag K., Pedersen H., DBMS Scheduling Performance, *T.R*, Norwegian Institute of Technology, 1994.

[NS03] The Network Simulator - ns-2, The ns Manual, The VINT Project, UC Berkeley, LBL, USC/ISI, and Xerox PARC, Kevin Fall and Kannan Varadhan Editors, 1994.

[NSB97] Norvag K., Sandsta O., Bratbergengen K., Concurrency Control in Distributed Object-Oriented Database Systems, in *Advances in Databases and Information Systems ADBIS'97, edited in Electronic Workshops in Computing (eWiC)*, Springer-Verlag, 1997, pp.9-17.

[OMG94] Object Management Group, Object Transaction Service, *OMG Document 94.8.4. OMG editor*, 1994.

[Opn04] OPNET Technologies, OPNET Modeler accelating networks R&D, 2004, www.opnet.com.

[Ora02a] Oracle Corporation, Oracle9i Lite: The Internet Platform For Mobile Computing http://otn.oracle.com/products/lite/, 2002.

[Ora02b] Oracle Corporation, Oracle9iAS Wireless, http://otn.oracle.com/products/iaswe/, 2002.

[Osi92] Open System Interconnection, Distributed Transaction Processing (OSI-TP) Model. ISO IS 10026, 1992.

[OV99] Özsu T., Valduriez P., Principles of Distributed Database Systems, *Prentice Hall*, 2nd édition, 1999.

[PA00] Pardon G., Alonso G., CheeTah : a lightweight transaction server for plug-and-play Internet data management, in *Proc. of VLDB 2000*, Cayro, Egypt, Sept. 2000.

[PA02] Popovici A., Alonso G., Ad-hoc transactions for mobile services, in Lecture Notes in Computer Science, *Proc. of the Third International Workshop on Technologies for E-Services*, ISBN: 3-540-44110-7, 2002, pp. 118–130.

[PAC+02] Perich F., Avancha S., Chakraborty D., Joshi A., Yesha Y., Profile Driven Data Management for Pervasive Environments, *LNCS*, vol. 2453, *Book of the 13th International Conference on Database and Expert Systems Applications (DEXA 2002)*, Aix en Provence, France, September 01, 2002, pp. 361-370.

[PACJY02] Perich F., Avancha S., Chakraborty D., Joshi A., Yesha Y., Profile Driven Data Management for Pervasive Environments, In *DEXA*, 2002.

[Pap86] Papadimitriou C., The theory of database concurrency control, *Computer Science Press*, Inc. New York, NY, USA, 1986.

[PB03] Perron M., Bai B., Low cost commit protocol for mobile computing environments, December 1999. Http://www.cs.Ualberta.ca, Accessed Sept. 2003.

[PB93] Pitoura E., Bhargava B., Dealing with Mobility: Issues and Research Challenges, *Technical Report* CSD-TR-070, Depart. Of Computer Science, Purdue University, November 1993.

[PB94a] Pitoura E., Bhargava B., Revising Transaction Concepts for Mobile Computing, *Proc. of the IEEE Workshop on Mobile Systems and Applications*, Santa Cruz, Ca, Dec 94.

[PB94b] Pitoura E., Bhargava B., Building Inforamtion Systems for Mobile Environments, *Proceedings of the 3rd International Conference on Information and Knowledge Management*, Novembre 1994, (CIKM), Gaithersburg, USA, 1994, pp. 317-378.

[PB94c] Pitoura E., Bhargava B., Maintaining Consistency of Data in Mobile Distributed Environments, *Technical Report*, TR694-25, Department of Computer Science, Purdue University, 1994.

[PB99] Pitoura E., Bhargava B., Data Consistency in Intermittently Connected Distributed Systems, *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, vol. 11, n° 6, 1999.

[PBVB01] Pucheral P., Bouganim L., Valduriez P., Bobineau C., Picodbms: Scaling down database techniques for the smartcard, *Very Large Databases (VLDB) Journal*, vol. 10, n° 2-3, 2001.

[PC99a] Pitoura E., Chrysanthis P. K., Exploiting Versions for Handling Updates in Broadcast Disks, Dans *Int. Conf. on Very Large Databases (VLDB)*, Edinburgh, Scotland, UK, September 1999.

[PC99b] Pitoura E., Chrysanthis P.K., Scalable processing of readonly transactions in broadcast push, In *Proceedings of the 19th IEEE International Conference on Distributed Computing System*, 1999.

[Per97] Perkins C. E., Mobile IP Design Principles and Practices, Addison-Wesley Wireless Communication Series, Reading, MA, 1997.

[PGG02] Popovici A., Gross T., Gustavo A., Dynamic weaving for Aspect Oriented Programming, in *1st Int. Conf. on Aspect-Oriented Software Development*, Enschede, The Netherlands, Apr. 2002.

[Pha05] Pham P., Comprehensive Analysis of IEEE 802.11, *Mobile Networks and Applications*, vol. 10, 2005, pp. 691-703.

[PJFY04] Perich F., Joshi A., Finin T., Yesha Y., On Data Management in Pervasive Computing Environments, *IEEE Transactions on Knowledge and Data Engineering*, vol. 16, n° 5, 2004, pp. 621-634.

[PJYF03] Perich F., Joshi A., Yesha Y., Finin T., Neighborhood-Consistent Transaction Management for Pervasive Computing Environments, *14 th LNCS conf. Database and Expert Systems Applications (DEXA2003)*, vol. 2736/2003, Publisher: Springer-Verlag GmbH, 2003, pp. 276 – 286.

[PKH88] Pu C., Kaiser G., Hutchinson N., Split transactions for Open-ended Activities, In *proceedings of the 14th Conference on Very Large Databases*, 1988.

[PLV05] PalChaudhuri S., Le boudec J. Y., Vojnovié M., Perfect simulation for random trip mobility models, in *Proc. of The 38th annual Symposium on Simulation*, San Diego, California, 2005, pp.72-79.

[Poi02] PointBase Java Databases, PointBase, http://www.pointbase.com, 2002.

[PS98] Pitoura E., Samaras G., Data Management for Mobile Computing, *Kluwer Academic Publishers*, 1998.

[Puj 01] Pujolle A. V., Réseaux de mobile et réseaux sans fil, *Edition Eyrolles*, 2001.

[Rak98] Rakotonirainy A., Adaptable Transaction Consistency for Mobile Environments, *DEXA Workshop*, 1998, pp. 440-445.

[Ram01] Ramampiaro H., CAGISTrans: Adaptable Transactional Support for Cooperative Work, Norges teknisk- naturvitenkapelig universitet (NTNU), *Dr.Ing. Thesis,* NTNU 2001:94, IDI-nr. 6/2001.

[RBLS04] Roncancio C. L., Bobineau C., Labbé C., Serrano-Alvarado P., Comparing Transaction Commit Protocols for Mobile Environments, In *IEEE International Workshop on Mobility in Databases and Distributed Systems*, Saragoza, Espagne, Septembre 2004.

[RC01] Roman M., Campbell R. H., A model for ubiquitous applications, *Technical Report*, UIUCDCSR-2001-2223 UILU-ENG-2001-1730, University of Illinois at Urbana-Champaign, May 2001, Avaliable at http://devius.cs.uiuc.edu/gaia/papers/appmodel-tech01.pdf.

[RC96] Ramamritham K., Chrysanthis P. K., Advances in Concurrency Control and Transaction Processing, *IEEE Computer Society Press*, 1996.

[RM07] Rouvoy R., Merle P., Using Microcomponents and Design Patterns to Build Evolutionary Transaction Services, In *Electronic Notes in Theoretical Computer Science (ENCTS)*, January 2007, pp. 111–125.

[RN04] Ramampiaro H. Nygård M., CAGISTrans: Providing adaptable transactional support for cooperative work - An Extended Treatment, In *Information Technology & Management (ITM) Journal*, vol. 5, n° 1-2, 2004, pp. 23-64, Kluwer Academic Publisher. Hingham, MA, USA.

[RSM06] Rouvoy R. , Serrano-Alvarado P., Merle P., Towards Context-Aware Transaction Services, in *Proceedings of the 6th International Conference on Distributed Applications and Interoperable Systems (DAIS'06)*, Bologna, Italy, *Lecture Notes in Computer Science*, Springer-Verlag, vol. 4025, June 2006, pp. 272–288.

[RZ96] Rasheed A., Zaslavsky A., Ensuring Database Availability in Dynamically Changing Mobile Computing Environment, *7th Australasian Database Conference, Melbourne*, 1996, pp. 100-108.

[RZ98] Rasheed A., Zaslavsky A. Twin-Transactions - Delayed Transaction Synchronisation Model, W/S Mobility & replication, ECOOP'98, Brussels, Belgium, July, 1998, Also published in *Lecture Notes in Computer Science*, Demeyer S., Bosch J., (Eds) Object-Oriented Technology, *ECOOP'98 Workshop Reader*, Springer, pp. 311-312.

[SA00] Segara M.T., André-A F., Generic approach to satisfy adaptability needs in mobile environments, *Proc. Of the 33rd Annal Hawaii International Conference on System Sciences*, january 2000, Maui, Hawaii, USA.

[Sat96] Satyanarayanan M., Mobile information access, *IEEE Personal Communications*, vol. 3, n° 1, 1996, pp. 26-33.

[SAW94] Schilit B., Adams N., Want R., Context-aware computing applications, In *Proceedings of IEEE Workshop on Mobile Computing Systems and Applications*, Santa Cruz, California, 1994, pp. 85-90, *IEEE Computer Society Press*, 1994.

[SC90] Stamos J., Christian F., A low cost atomic commit protocol, in *Proc. of the 9th symposium on reliable distributed systems*, 1990.

[Ser04] Serrano-Alvarado P., Transactions Adaptables pour les Environnements Mobiles, *PhD thesis*, Joseph Fourier University, Grenoble, France, 2004.

[SKC 03] Samaras G., Kyrou G., Chrysanthis P. K., Two-Phase Commit Processing with Restructured Commit Tree, *Post Proc. of the 8th Panhellenic Conference on Informatics and Computer Science*, LNCS series, 2003.

[SKC 01] Samaras G., Kyrou G., Chrysanthis P. K., Structuring the Commit Tree for Better Performance of Two Phase Commit Processing, *Proc. of the 8th Panhellenic Conference on Informatics and Computer Science*, Cyprus, Nov. 2001.

[Ske81] Skeen D., Non-Blocking Commit Protocols, *Proceedings of ACM-SIGMOD International Conference on Management of Data (SIGMOD)*, 1981.

[SKK+90] Satyanarayanan M., Kistler J. J., Kumar P., Okasaki M. E., Siegel E. H., Steere D. C., Coda: A Highly Available File System for a Distributed Workstation Environment, *IEEE Transactions on Computers*, vol. 39, n° 4, 1990, pp. 447–459.

[SKMN89] Schumann R., Kroger R., Mock M., Nett E., Recovery Management in the Relax distributed transaction Layer, *Proc. of the 8th Symposium on reliable distributed systems, Seattle*, 1989.

[SL90] Sheth A. P., Larson J. A., Federated Database Systems for Managing Distributed, Heterogeneous, and Autonomous Databases, *ACM Computing Surveys*, vol. 22, n° 3, 1990, pp.183–236.

[SM03] Saha D., Mukherjee A., Pervasive Computing: A Paradigm for the 21st Century, *IEEE Computer, IEEE Computer Society Press*, 2003, pp. 25-31.

[Smi82] Smith B.C., Procedural Reflection in Programming Languages, *PhD thesis*, MIT, MIT Computer Science, 1982.

[Smi82] Smith B.C., Procedural Relection in Programming Languages, *PhD Thesis*, MIT, MIT Computer Science Technical Report 272, Cambridge, Mass., 1982.

[SN95] Sandsta O., Norvag K., Distributed Concurrency Control Performance: A comparison between Timestamp ordering and two phase locking schedulers, *Technical report*, Norwegian Institute of Technology, 1995.

[SNK+94] Satyanarayanan M., Noble B., Kumar P., Price M., Application-aware adaptation for mobile computing, In *Sixth ACM SIGOPS European Workshop*, Dagstuhl, Germany, 1994, http://www.cs.cmu.edu/afs/cs/project/coda/Web/docs-ody.html.

[SNP+97] Shanmugasundaram J., Nithrakasyap A., Padhye J., Sivasankaran R., Xiong M., Ramamritham K., Transaction processing in broadcast disk environments, In *Jajodia S., Kerschberg L. (Eds.), Advanced Transaction Models and Architectures*, Kluwer, Boston, 1997, pp. 321–338.

[SNT+97] Noble B. D., Satyanarayanan M., Narayanan D., Tilton J. E., Flinn J., Walker K. R., Agile application-aware adaptation for mobility, In *Sixteen ACM Symposium on Operating Systems Principles*, Saint Malo, France, 1997, pp.276-287, http://www.cs.cmu.edu/afs/cs/project/coda/Web/docdir/s16-reprint.ps.Z.

[SRA04] Serrano-Alvarado P., Roncancio C.L., Adiba M., A Survey of Mobile Transactions, *International Journal on Distributed and Parallel Databases (DAPD)*, vol. 16, n° 2, 2004, pp. 193-230.

[SRAL03] Serrano-Alvarado P., Roncancio C. L., Adiba M., Labbé C., Adaptable Mobile Transactions, In *19ièmes Journées Bases de Données Avancées*, Lyon, France, october 2003.

[SRAL05] Serrano-Alvarado P., Roncancio C. L., Adiba M., Labbé C., An Adaptable Mobile Transaction Model for Mobile Environments, In *Int. Journal of Computer Systems Science and Engineering (IJCSSE), Special Issue on Mobile Databases*, vol. 20, n° 3, 13 pages, ISSN 0267-6192, April 2005.

[SSW95] Schaad W., Schek H. J., Weikum G., Implementation and Performance of Multi-level Transaction Management in a Multibase Environnement, 0-8186-7056-8/95, *IEEE*, 1995, pp. 108-115.

[ST94] Schilit B., Theimer M., Disseminating Active Map Information to Mobile Hosts, *IEEE Network*, vol. 8, n° 5, 1994, pp. 22-32.

[STM00] Schmidt A., Takaluoma A. Mäntyjärvi J., Context-Aware Telephony Over WAP, In *journal Personal Ubiquitous Computing*, 2000, vol. 4, n°4, pp. 225-229.

[Str93] Stroud R., Transparency and reflection in Distributed Systems, *ACM Operating Systems Review*, vol. 22, n° 2, April 1993.

[SVF04] Santos N., Veiga L., Ferreira P., Transaction policies for mobile networks, In *5th IEEE International Workshop on Policies for Dist. Systems and Networks(Policy 2004)*, 2004.

[SWR98] Singh S., Woo M., Raghavendra C., Power Aware Routing in Mobile Ad Hoc Networks, in *Proc. 4th International Conf. on Mobile Computing and Networking (MOBICOM '98)*, 1998, pp 181-190.

[Syb00] Sybase Inc. Adaptive Server Anywhere User's Guide, November 2000.

[Syn00] SyncML Consortium. SyncML Sync Protocol, 1999–2000.

[Szy97] Szyperski C., Component Software, Beyond Object-Oriented Programming, *Addison-Wesley*, 1997.

[TMUY03] Tsumochi J., Masayama K., Uehara H., Yokoyama M., Impact of mobility metric on routing protocols for mobile Adhoc Networks, in *Proc. of the IEEE Pacific Rim Conference on Communications, Computers and Signal Processing (PACRIM03)*, Victoria, 2003, pp. 322-325.

[TR96] Theel O., Raynal M., RR-2999 – Static and Dynamic Adaptation of Transactional Consistency, *Rapport de recherche de l'INRIA*, Rennes, Equipe ADP, 30 pages, Octobre 1996.

[Tra 92] Traverson B., Le protocole de validation en deux phases - Stratégies d'optimisation et évaluation de performance, *Réseaux et Informatique Répartie*, vol. 2, n° 4, 1992.

[TZ03] Türker C., Zini G., A Survey of Academic and Commercial Approaches to Transaction Support in Mobile Computing Environments, *Technical Report*, n° 429, ETH-Zentrum,CH-8092 Zürich, Switzerland, 34 pages, 2003.

[Var02] Varshney S., Implementation of a collaborative transaction processing system on MANET, *Master thesis of technology*, Dept. of Computer Science & Engineering, Indian Institute of Technology, Kanpur, 2002.

[VCFS00] Vidot N., Cart M., Ferrié J., Suleiman M., Copies convergence in a distributed real-time collaborative environment, Dans *Int. Conf. on Computer Supported Cooperative Work (CSCW)*, Philadelphia, USA, December 2000.

[Vid01] Vidal B., Application mobiles avec oracle, *Edition Eyrolles*, 2001.

[Vie01] Viellard E., Oracle 9i Lite Business white paper, Oracle corporation, September 2001.

[Vin02] Vingralek R., GnatDb: A Small-Footprint, Secure Database System, Dans *Int. Conf. on Very Large Databases (VLDB)*, Hong Kong, China, August 2002.

[Vol01] Völter M., Server-Side Components - A Pattern Language, In *Sixth European Conference On Pattern Languages of Programs*, Irsee, Germany, 4-8 July 2001.

[W3C01] W3C XML Schema. www.w3.org/XML/Schema

[WC95] Walborn G. D., Chrysanthis P. K., Supporting Semantics-Based Transaction Processing In Mobile Database Applications, *Proc. Of The 14th Symposium on Reliable Distributed, Systems (SRDS)*, Bad Neuenahr, Germany, September 1995, pp. 31-40.

[WC97] Walborn G. D., Chrysanthis P. K., PRO-MOTION: Management of Mobile Transactions, Dans *ACM Symp. on Applied Computing*, San Jose, USA, March 1997.

[WC99] Walborn G. D., Chrysanthis P. K., Transaction Processing in PRO-MOTION, Dans *ACM Symp. on Applied Computing*, San Antonio, USA, February 1999.

[Wei91] Weiser M., The Computer for the Twenty-First Century, *Scientific American*, September 1991, pp. 94-10.

[Wei91] Weikum M., Principles and Realization Strategies of Multilevel Transactions Management, *ACM Transactions on Database Systems*, vol. 16, n° 1, March 1991, pp. 132–180.

[Wei93] Weisern M., The Computer for the Twenty-First Century, *Scientific American*, vol. 265, n° 3, 1993, pp. 94–104.

[WS92] Weikum G., Schek H. J., Concepts and Applications of Multilevel Transactions and Open Nested Transactions, Chapitre 13, Special supplement to Database Transaction Models FOr Advanced Applications By Ahmed K. Elmagarmid, Mogrgan Kaufman Publishers, Inc., 1992.

[WV02] Weikum G., Vossen G., Transactional information systems, Theory, algorithms, and the practice of concurrency control and recovery, USA, Morgan Kaufmann, 2002.

[WW04] Wang Y., Weihai Y., Adaptable Transaction Processing in The Web Services Domain, NIK-04, *The Norwegian Information Technology Conference,* Stavanger, Norway NIK'04, November 2004.

[WXCJ98] Wolfson O., Xu B., Chamberlain S., Jiang L., Moving Objects Databases: Issues and Solutions, Dans *Int. Conf on Statistical and Scientific Database Management (SSDBM)*, Capri, Italy, July 1998

[X/OP96] X/OP, CAE Specification, Distributed Transaction Processing: Reference Model, X/Open Guide, Version 3, G307, X/Open Company Limited, 1996.

[XML00] eXtensible Markup Language (XML) 1.0, W3C Recommendation 6 October 2000, http://www.w3.org/TR/REC-xml.

[YAB02] Yamin A., Augustin I., Barbosa J., Geyer C., ISAM, A pervasive view in distributed mobile computing, 2002.

[YCG+03] Yang L., Conner S., Guo X., Hazra M., Zhu J., Common wireless ad hoc network usage scenarios, Internet draft, *Work in progress*, 2003.

[Yok92] Yokote Y., The Apertos Reflective Operating System: The Concept and Its Implementation, in *Proc. of the Conference on Object-Oriented Programming Systems, Languages, and Applications (OOPSLA)*, October 1992.

[YPT+01] Yegin A., Perkins C., Tsirtsis G., El-Malki K., Khalil M., Fast Handovers for Mobile IPv6, IETF draft, draft-ietf-mobileip-fast-mipv6-03.txt. Dommety G. Editor, 2001.

[YV02] Yang X., Vaidya N. H., Pipelined packet scheduling in wireless LANs, *Research report*, University of Illinois at Urbana-Champaign, 2002.

[YZ94a] Yeo L. H., Zaslavsky A., Layered Approach to Transaction Management in Multidatabase Systems, In *Proc. of the 5th Int. Hong Kong Computer Society DatabaseWorkshop: Next Generation Database Systems*, 1994, pp. 179–189.

[YZ94b] Yeo L. H., Zaslavsky A., Submission of Transactions from Mobile Workstations in a Cooperative Multidatabase Processing Environment, In *Proc. of the 14th IEEE Int. Conf. on Distributed Computing Systems*, June 21–24, 1994 Poznan, Poland, pp. 372–379, *IEEE Computer Society Press*, 1994.

[ZAFA94] Zdonik S., Alonso R., Franklin M., Acharya S., Are disks in the air just pie in the sky, In *Proceedings of the Workshop of Mobile Computing Systems and Applications*, California, 1994.

[ZNBB94] Zhang A., Nodine M., Bhargava B., Bukhres O., Ensuring Relaxed Atomicity for Flexible Transactions in Multidatabase Systems, In *ACM SIGMOD Conference*, Minneapolis, USA, 1994, pp. 67-78.

[ZPL98] Zhou T., Pu C., Liu L., Dynamic Restructuring of Transactional Workflow Activities: A Practical Implementation Method, In *Proceedings of the 7th International Conference on Information and Knowledge Management*, Washington, D.C., November 1998.

Annexe A
Rappel de la logique d'ordre zéro

L'objectif de cette annexe est d'aider à la compréhension de la spécification formelle du modèle faite dans le chapitre 6. Nous présentons d'abord un petit rappel des lois de la logique (section A.1). Ensuite, nous rappelons quelques règles de déduction naturelle (section A.2).

A.1 Lois logiques

- Double négation : $\neg (\neg A) \Leftrightarrow A$

- Transitivité : $((A \Rightarrow B) \wedge (B \Rightarrow C)) \Rightarrow (A \Rightarrow C)$

- Commutativité : $(A \wedge B) \Leftrightarrow (B \wedge A)$
  $(A \vee B) \Leftrightarrow (B \vee A)$

- Associativité : $((A \wedge B) \wedge C) \Leftrightarrow (A \wedge (B \wedge C))$
  $((A \vee B) \vee C) \Leftrightarrow (A \vee (B \vee C))$

- Distributivité: $(A \wedge (B \vee C)) \Leftrightarrow ((A \wedge B) \vee (A \wedge C))$
  $(A \vee (B \wedge C)) \Leftrightarrow ((A \vee B) \wedge (A \vee C))$

- Lois de Morgan : $\neg (A \wedge B) \Leftrightarrow (\neg A \vee \neg B)$
  $\neg (A \vee B) \Leftrightarrow (\neg A \wedge \neg B)$

- Contraposition : $(A \Rightarrow B) \Leftrightarrow (\neg B \Rightarrow \neg A)$

- L'implication : $(A \Rightarrow B) \Leftrightarrow (\neg A \vee B)$

- La négation d'une implication : $\neg (A \Rightarrow B) \Leftrightarrow (A \wedge \neg B)$

- Double implication : $(A \Leftrightarrow B) \Leftrightarrow ((A \Rightarrow B) \wedge (B \Rightarrow A))$

A.2 Règles de déduction naturelle

Les règles de déduction naturelle peuvent être d'introduction ou de réduction.

- A B on en déduit $A \wedge B$

- $(A \Rightarrow B, A)$ on en déduit B (Modus ponens)

- $(A \Rightarrow B, \neg B)$ on en déduit $\neg A$ (Modus tollens)

- $(A \Rightarrow B)$ $(B \Rightarrow A)$ on en déduit $(A \Leftrightarrow B)$

Annexe B

Le formalisme ACTA

ACTA [CR90, CR94] est un cadre formel et extensible conçu pour la spécification de transactions. Il est basé sur la logique du premier ordre. ACTA permet de spécifier et de raisonner sur les interactions entre les transactions. Il considère différents types d'interaction entre les transactions ainsi que les effets des transactions sur les objets.

ACTA organise l'expression des effets des transactions (sur d'autres transactions et sur des objets) en cinq dimensions : l'historique, les dépendances entre transactions, la vue d'une transaction, l'ensemble des conflits d'une transactions et la délégation (voir figure B.1).

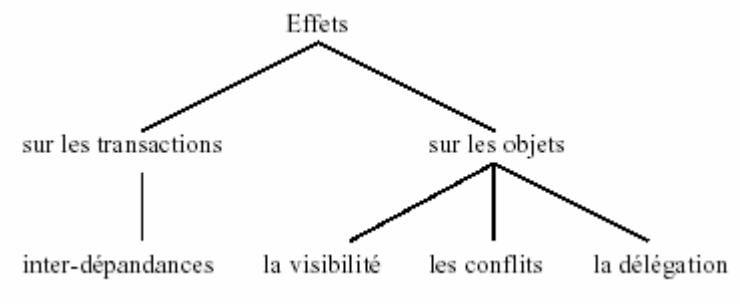


Figure B.1-Les dimensions du canevas ACTA

L'interdépendance des transactions permet de définir la structure du modèle transactionnel ainsi que les aspects liés à l'atomicité. La visibilité et les conflits sur les objets concernent les caractéristiques liées à l'isolation et à la correction des données manipulées (sérialisabilité). La propriété de cohérence ne peut pas être exprimée par le formalisme ACTA. Cette restriction n'a pas de grandes répercutions sur la spécification correcte des modèles transactionnels car la vérification des contraintes d'intégrité n'intervient pas dans la définition des modèles.

Cette section introduit le formalisme ACTA. Nous présentons les notions de base nécessaires à la définition du modèle AMT (cf. chapitre 6). Pour une présentation plus complète voir [CR90, CR94]. La section B.5 présente quelques spécifications indispensables comme la sérialisabilité ou l'atomicité.

B.1 Les objets, les événements

Les transactions accèdent aux objets et les manipulent en demandant des opérations (par exemple, lire, écrire, incrémenter). Ces opérations sont considérées comme des événements atomiques. $p_t[ob]$ dénote l'événement qui correspond à la demande de l'opération p sur l'objet *ob* par la transaction t. OE t dénote l'ensemble des événements qui peuvent être demandées par *t*, ainsi, $p_t[ob] \in O\,E_t$.

L'état d'un objet est représenté par son contenu. Les opérations produisent toujours un résultat. Le résultat d'une opération sur un objet dépend de l'état de l'objet. Pour un objet dans un état donné *s*,

return(*s; p*) exprime le résultat retourné par une opération *p*, et *état*(*s; p*) dénote l'état de l'objet après l'exécution de *p*.

Les effets des opérations sur les objets ne sont pas permanents pendant l'exécution. Les opérations doivent être validées (commit) ou annulées (abort) de manière explicite. Si une opération n'est pas validée ou annulée, alors elle est en cours d'exécution (in progress). Une opération est validée (ou annulée) lorsque la transaction qui l'a demandée valide (ou annule). Ainsi, les effets d'une opération *p* demandée par une transaction *t* sur un objet *ob* sont rendus permanents sur la base de données lorsque $p_t[ob]$ est validée.

En dehors de la demande des opérations sur les objets, les transactions demandent des primitives de gestion - début (begin), validation, annulation. La définition des primitives et leur sémantique dépendent du modèle de transactions. De ce fait, la demande d'une primitive de transaction est un événement significatif. ACTA ne propose pas un ensemble particulier d'événements significatifs mais il fournit un moyen pour le spécifier.

- ES_t dénote l'ensemble des Événements Significatifs pour la transaction *t*.

- EI_t dénote les Événements d'Initiation qui peuvent être demandés pour initier l'exécution de la transaction *t*. $EI_t \subset ES_t$.

- ET_t dénote les Événements de Terminaison qui peuvent être demandés pour terminer l'exécution de la transaction *t*. $ET_t \subset ES_t$.

- Les événements correspondant aux opérations sur les objets sont différents des événements de gestion des transactions $OE_t \neq ES_t$.

B.2 Les historiques et les conditions sur l'occurrence des événements

La notion d'historique est fondamentale dans ACTA. Un historique représente l'exécution concurrente d'un ensemble de transactions. ACTA capture les effets des transactions B.2 Les historiques et les conditions sur l'occurrence des événements sur d'autres transactions, ainsi que les effets sur les objets, au travers de contraintes sur les historiques. Ceci conduit à la définition de modèles de transactions en termes d'un ensemble d'axiomes. Ces axiomes sont des assertions invariables concernant les historiques ou des pré-conditions/post-conditions des opérations ou encore des primitives de gestion des transactions. La correction des différents modèles de transactions peut également être exprimée en termes de propriétés des historiques générés.

Définition B.1 : L'exécution d'une transaction

L'exécution d'une transaction t est un ordre partiel sur l'ensemble d'événements E_t avec une relation d'ordre $<_t$ où:

1. $E_t \subseteq (EO_t \cup ES_t)$; et

2. $<_t$ dénote l'ordre temporel dans lequel les événements demandés par t arrivent. Autrement dit, E t contient les événements sur les objets qui peuvent être demandés par t et les événements significatifs liés à *t*.

Définition B.2 : Historique H

Un historique H de l'exécution concurrente d'un ensemble de transactions T, contient tous les événements associés à T et indique l'ordre (partiel) de l'occurrence de ces événements. H_{ct} dénote l'historique des événements produits avant un instant donné (historique courant).

La place d'apparition d'un événement dans l'historique peut être affecté de trois manières différentes: (1) un événement *e* peut être contraint de se produire *après* un autre événement *e'* ; (2) un événement *e* peut apparaître seulement si une condition *c* est vraie ; (3) une condition *c* peut demander l'occurrence d'un événement *e*.

- $e \rightarrow e'$: le prédicat est vrai si l'événement *e* précède l'événement *e'* dans *H* .
- ($e \in H$) $\Rightarrow$ condition *H*: si *e* appartient à *H*, alors la condition sur *H* est vrai. Autrement dit, condition *H* est nécessaire pour que *e* soit dans *H*.

- condition $H \Rightarrow (e \in H)$: si la condition sur H est vrai, alors e appartient à H. Autrement dit, condition H est suffisante pour que e soit dans H.

Par ailleurs, la projection d'un historique H selon un critère donné x est exprimée par *projection*(H,x). Par exemple, *projection*(H, t) est la projection de l'historique H sur une transaction t. Cette projection notée H^t , donne l'ordre des événements liés à t. Nous pouvons également considérer *projection*(H,ob), qui est la projection de l'historique H sur un objet spécifique ob. Cette projection notée $H^{(ob)}$ donne l'historique de la demande des opérations sur l'objet ob.

B.3 Dépendances des transactions

ACTA utilise des *dépendances* comme un moyen de spécifier et de raisonner sur le comportement des transactions concurrentes. Les dépendances sont exprimées en termes d'événements significatifs associés aux transactions. Ainsi, l'ensemble de dépendances inter-transactions, qui se produisent pendant l'exécution concurrente des transactions est noté *DepSet*. Cet ensemble est relatif à l'historique H. $DepSet_{ct}$ dénote l'ensemble de dépendances jusqu'à un instant donné (ct), où $DepSet_{ct}$ concerne H_{ct} .

Types de dépendances

La liste de dépendances présentée ici n'est pas exhaustive. Grâce au caractère extensible d'ACTA, d'autres dépendances peuvent être définies.

Soit t_i et t_j deux transactions et H un historique fini qui contient tous les événements appartenant à t_i et t_j .

Dépendance de validation (Commit Dependency) (t_j CD t_i)**:** si les deux transactions t_i et t_j valident, la validation de t_i doit précéder celle de t_j :
$(commit_{tj} \in H) \Rightarrow ((commit_{ti} \in H) \Rightarrow (commit_{ti} \rightarrow commit_{tj}))$.

Dépendance de validation forte (Strong-Commit Dependency) (t_j SCD t_i) : si t_i valide, alors t_j valide aussi:

$(commit_{tj} \in H) \Rightarrow (commit_{ti} \in H)$.

Dépendance d'annulation (Abort Dependency) (t_j AD t_i) : si t_i annule, alors t_j aussi:
$(abort_{ti} \in H)) \Rightarrow (abort_{tj} \ H)$.

Dépendance d'annulation faible (Weak-Abort Dependency) (t_j WD t_i): si t_i annule et t_j n'a pas encore validé, t_j doit annuler aussi. En d'autres termes, si t_j valide et t_i annule, la validation de t_j doit précéder l'annulation de t_i dans l'historique:
$(abort_{ti} \in H) \Rightarrow (\neg (commit_{tj} \rightarrow abort_{ti}) \Rightarrow (abort_{tj} \in H))$.

Dépendance de terminaison (Termination Dependency) (t_j TD t_i): t_j ne peut pas valider ou annuler tant que t_i n'a pas validé ou annulé:

$(e' \in H) \Rightarrow (e \rightarrow e')$ où $e \in \{commit_{ti}, abort_{ti}\}$, et $e' \in \{commit_{tj} , abort_{tj}\}$.

Dépendance d'exclusion (Exclusion Dependency) (t_j ED t_i): si t_i valide et si t_j a démarré son exécution, alors t_j annule (tous deux, t_i et t_j , ne peuvent pas valider):

$(commit_{ti} \in H) \Rightarrow ((begin_{tj} \in H) \Rightarrow (abort_{tj} \in H))$.

Dépendance de compensation (Force-Commit-on-Abort Dependency) (t_j CMD t_i):
si t_i annule, t_j doit valider:
$(abort_{ti} \in H) \Rightarrow (commit_{tj} \in H)$.

Dépendance de début (Begin Dependency) (t_j BD t_i): t_j ne peut pas démarrer si t_i n'a pas démarré:
$(begin_{tj} \in H) \Rightarrow (begin_{ti} \rightarrow begin_{tj})$.

Dépendance séquentielle (Serial Dependency) (t_j SD t_i) : t_j ne peut pas démarrer tant que t_i n'a pas validé ou annulé :
$(begin_{tj} \in H) \Rightarrow (e \rightarrow begin_{tj})$ où $e \in \{commit_{ti} , abort_{ti}\}$.

Dépendance début sur validation (Begin-on-Commit Dependency) (t_j BCD t_i): t_j ne peut pas démarrer jusqu'à ce que t_i valide:

$(begin_{tj} \in H)) \Rightarrow (commit_{ti} \rightarrow begin_{tj})$.

Dépendance début sur annulation (Begin-on-Abort Dependency) (t_j BAD t_i): t_j ne peut pas démarrer jusqu'à ce que t_i annule:

$(begin\ t_j \in H) \Rightarrow (abort\ t_i \rightarrow begin\ t_j)$.

Dépendance début sur validation faible (Weak-begin-on-Commit Dependency)

(t_j WCD t_i) : si t_i valide, t_j peut démarrer après que t_i valide:

$(begin\ t_j \in H) \Rightarrow ((commit_{ti} \in H) \Rightarrow (commit_{ti} \rightarrow begin_{tj}))$.

B.3.1 Sources de dépendances

Les dépendances entre les transactions peuvent provenir soit directement des propriétés structurelles des transactions - *dépendances de structure* - soit indirectement par le fait que les transactions manipulent les mêmes objets - *dépendances de comportement*.

Dépendances de structure

La structure d'une transaction étendue définit ses transactions composantes et les relations entre elles. En effet, les dépendances peuvent spécifier des liens dans la structure. Par exemple, dans une Sagas, la relation avec ses composantes est établie au démarrage de la transaction composante. Ceci peut être exprimé par la dépendance d'annulation faible de la transaction composante t_i vis-à-vis de la Saga S, (t_i WD S), et par la dépendance d'annulation de la Saga vis-à-vis de la transaction composante (S AD t_i).

Comme la relation s'établie au démarrage de la transaction composante t_i , cette dépendance s'exprime par une post condition :

$post(begin\ t_i) \Rightarrow (((t_i\ \mathrm{WD}\ S) \in DepSet_{ct}) \wedge ((S\ \mathrm{AD}\ t_i) \in DepSet_{ct}))$

La dépendance d'annulation de la Saga sur ses transactions composantes assure l'annulation de la Saga lorsqu'une transaction composante annule. La dépendance d'annulation faible garantit l'annulation d'une transaction composante non-validée si la Saga annule. Il faut remarquer que cette dépendance n'interdit pas à la transaction composante de valider et de rendre ses effets visibles.

Dépendances de comportement

Les dépendances (D) formées par l'interaction de transactions sur un objet partagé sont déterminées par les propriétés de synchronisation sur cet objet. Deux opérations sont en con□it si l'ordre de leur exécution est important. Par exemple, une relation de conflit (D) peut apparaître de la manière suivante :

$(p_{t\,i}\,[ob] \rightarrow q_{t\,j}\,[ob]) \Rightarrow (t_j\ \mathrm{D}\ t_i))$

Ceci veut dire que si la transaction t_i demande une opération p et qu'ultérieurement une transaction t_j demande une opération q sur le même objet, alors t_j génère une dépendance de type D sur t_i .

B.4 Effets des transactions sur les objets

La correction de l'exécution concurrente de transactions dépend des dépendances entre elles et de la manière dont elles accèdent aux objets. Plus concrètement, la correction dépend de l'effet des événements significatifs associés aux transactions ainsi que des opérations demandées. La section B.3.1 a abordé le premier point. Cette section aborde le second. Tout d'abord, nous introduisons les effets des opérations sur les transactions et les inter-relations induites par ces effets. Ensuite, nous expliquons la visibilité des effets des opérations d'une transaction sur une autre transaction.

Conflits entre les opérations

D'une façon générale, deux opérations p et q sont en conflit si elles accèdent au même objet et au moins une d'entre elles est une écriture (*conflit* ($H^{(ob)}$, p, q)). C'est-à-dire, deux opérations sont en conflit si leurs effets sur l'état d'un objet ou leurs valeurs de retour ne sont pas indépendants de leur ordre d'exécution :

return - value - dependent($H^{(ob)}$, p, q) est vrai si (*conf lit*($H^{(ob)}$, p, q)) est vrai et

return ($H^{(ob)}$ o p, q) ≠ *return* ($H^{(ob)}$ q).

Dans ACTA le contrôle de la concurrence sur un objet est exprimé en termes de *relations de conflit* :

$(p_{t\,i}\,[ob] \rightarrow q_{tj}\,[ob]) \Rightarrow condition_H$

Où condition H est une dépendance de relation qui inclut les transactions t_i et t_j qui demandent les opérations conflictuelles p et q sur l'objet ob.

La $condition_H$ dans une relation de conflit peut inclure d'autres événements significatifs définis par les différents modèles transactionnels. En outre, la généralité qui porte la relation de conflit montrée ici, permet la spécification de différents types de contrôle de concurrence.

Visibilité d'une transaction

La visibilité d'une transaction est sa capacité de voir les effets d'autres transactions pendant son exécution. Dans ACTA le contrôle sur la visibilité des objets est défini par deux entités, *Visibilité* et *ConflictSet*. *Visibilité* est l'ensemble des objets visibles à un instant donné. Ainsi, lorsque la visibilité d'une transaction porte sur l'historique courant des objets de la base de données, sans aucune restriction, la définition est :

$Visibilité_t = H_{ct}$

La restriction de la visibilité peut être exprimée par la projection d'un prédicat sur l'historique courant :

$Visibilité_t = projection\ (H_{ct}\ , prédicat\ (t, H_{ct}\ , DepSet_{ct}\ ))$.

Le prédicat dépend de t, des événements dans H_{ct} et des dépendances inter-transactions $DepSet_{ct}$.

Ensemble des conflits d'une transaction

L'ensemble des conflits d'une transaction t, dénoté par $ConflictSet_t$, contient toutes les opérations en cours avec lesquelles des conflits doivent être déterminés. La composition d'un $Conf\ lictSet_t$ est aussi définie par des prédicats qui peuvent inclure des événements demandés par t, par d'autres transactions t_i , par des événements de H_{ct} , ainsi que par des dépendances dans $DepSet_t$:

$Conf\ lictSet_t = \{p_{ti}\,[ob] \mid predicat\ (t, t_i\ , H_{ct}\ , DepSet_{ct})\}$

Cela dit, l'ensemble des conflits d'une transaction t est composé par toutes les opérations en cours faites par d'autres transactions :

$Conf\ lictSet_t = \{p_{t'}\ [ob] \mid t' \neq t\ , Inprogress\ (p_{t'}\,[ob])\}$.

B.5 Quelques spécifications en utilisant ACTA

Dans cette section, nous montrons la définition de sérialisabilité, de failure atomicity, de setwise failure atomicity et des transactions atomiques.

B.5.1 Notions de correction

Sérialisabilité

On définit la sérialisabilité de la manière suivante grâce au formalisme ACTA:

Soit T un ensemble de transactions.
Soit C une relation binaire sur les transactions de T.

Soit T_{commit} le sous-ensemble de T qui contient les transactions qui ont validé.
Soit H_{commit} l'historique des événements liés aux transactions dans T_{commit}.

Définition B.3 : Sérialisabilité

$\forall t_i , t_j \in T_{commit} , t_i \neq t_j$

$(t_i \, C \, t_j)$ *si* $\exists \, ob \, \exists \, p, q \; (conf\ lit \; (p_{ti} \, [ob]; \, q_{tj} \, [ob]) \wedge (p_{ti} \, [ob] \rightarrow q_{tj} \, [ob]))$

Soit C^* *la fermeture transitive de C* :

$(t_i \, C^* \, t_k)$ *si* $[(t_i \, C \, t_k) \vee \exists \, t_j \, (t_i \, C \, t_j \wedge t_j \, C^* \, t_k)]$.

H_{commit} est *(conflit)* ***sérialisable*** *si* $\forall t \in T_{commit} \neg (t \; C^* \; t)$.

Autrement dit, deux transactions ont une relation d'ordre (C), lors qu'elles sont en conflit et lorsque la fermeture transitive est acyclique. Cette définition de sérialisabilité est en réalité celle de *"sérialisabilité conflictuelle"* où l'ordre d'exécution des transactions qui ne sont pas en conflit n'affecte pas la correction des objets.

Les objets atomiques

La sérialisabilité porte sur un ensemble de transactions validées. Cependant, il est aussi important d'assurer la sérialisabilité sur les transactions en cours d'exécution. En effet, il faut assurer que l'annulation d'une opération entraîne l'annulation des opérations pour lesquelles une dépendance de valeur existe. La définition qui suit montre que pour qu'un objet ait un comportement correct, il doit assurer que lorsqu'une opération (p_{ti}) annule, toutes les opérations (q_{tj}) dont leur valeurs dépendent de (p_{ti}), doivent aussi annuler. Un objet a aussi un comportement sérialisable si la relation des transactions validées qui lui ont accédé est acyclique où C_{ob} considère seulement les relations concernant ob ($C_{\text{ob}} \subseteq C$).

Définition B.4 : Correction d'un objet

Un objet ob a un comportement ***correct*** *si :*

$\forall t_i , t_j , t_i \neq t_j , \; \forall p, q$

$(return - value - dependent \; (p, q) \wedge (p_{ti} \, [ob] \rightarrow q_{tj} \, [ob]) \wedge$

$\neg ((commit \; [p_{ti} \, [ob]] \rightarrow q_{tj} \, [ob]) \vee (abort \; [p_{ti} \, [ob]] \rightarrow q_{tj} [ob])) \Rightarrow$

$((abort [p_{ti} \, [ob]] \in H^{(ob)}) \Rightarrow (abort \; [q_{tj} \, [ob]] \in H^{(ob)}))$.

$\forall t_i , t_j \in T_{commit} , t_i \neq t_j$

$(t_i \, C_{ob} \, t_j)$ *si* $\exists \, p, q \; (conf\ lit \; (p_{ti} \, [ob]; \, q_{tj} \, [ob]) \wedge (p_{ti} \, [ob] \rightarrow q_{tj} [ob]))$

Un objet ob a un comportement **sérialisable** *si* :

$\forall \; t \in T_{commit} \neg (t \; C^*_{\text{ob}} \; t)$.

Un objet ob est **atomique** *s'il a un comportement* **correct** *et* **sérialisable**.

B.5.2 Différents types d'atomicité

Failure atomicity

L'atomicité, ou plus particulièrement la *failure atomicity* comme indiqué dans [Chr91], implique la validation de toutes les opérations d'une transaction ou d'aucune d'entre elles ("tout ou rien"). Dans la

définition qui suit, le premier point exprime la clause "tout" tandis que le second exprime la clause "rien".

Définition B.5 : ***Failure atomicity***

Une transaction t est failure atomic si :

1. $\exists\ ob\ \exists\ p\ (commit\ [p_t\ [ob]] \in H) \Rightarrow$
$\forall ob'\ \forall q\ ((q_t\ [ob'\ ] \in H) \Rightarrow (commit\ [q_t\ [ob'\ ] \in H))$

2. $\exists\ ob\ \exists\ p\ (abort\ [\text{p t}\ [ob]] \in H) \Rightarrow$
$\forall ob'\ \forall q\ ((q_t\ [ob'\ ] \in H) \Rightarrow (abort\ [p_t\ [ob'\ ]] \in H))$

Voir Annexe A.

Setwise failure atomicity

Setwise failure atomicity est une généralisation de *failure atomicity* pour des transactions composées de plusieurs transactions. De façon similaire à *failure atomicity*, la définition qui suit indique dans sa première condition que si une opération demandée par une transaction t_i appartenant à T est validée, alors, toutes les opérations demandées par les transactions dans T sont validées. La seconde condition indique que si une opération demandée par t_i qui appartient à T est annulée, alors, toutes les opérations demandées par les transactions de T sont annulées.

Définition B.6 : ***Setwise failure atomicity***

Un ensemble de transactions T est setwise failure atomic si :

1. $\exists\ t_i \in T\ \exists$ ob $\exists$ p (commit$_{ti}$ [p$_{ti}$ [ob]] $\in$ H))
$\forall\ t_j \in T\ \forall$ ob' $\forall$q ((q$_{tj}$ [ob'] $\in$ H)) (commit$_{tj}$ [q$_{tj}$ [ob']] $\in$ H));

2. $\exists\ t_i \in T\ \exists$ ob $\exists$ p (abort$_{ti}$ [p$_{ti}$ [ob]] $\in$ H))
$\forall\ t_j \in T\ \forall$ ob' $\forall$q ((q$_{tj}$ [ob'] $\in$ H)) (abort$_{tj}$ [q$_{tj}$ [ob']] $\in$ H))

Failure atomicity ainsi que setwise failure atomicity assurent l'atomicité d'une ou de plusieurs transactions.

B.5.3 Transactions atomiques

Cette section introduit en premier lieu les axiomes fondamentaux des transactions (définition B.7), puis les axiomes des transactions atomiques (définition B.8).

Les axiomes fondamentaux des transactions

Chaque modèle de transaction définit un ensemble d'événements significatifs que les transactions de ce modèle peuvent demander en plus des opérations sur les objets. En effet, une transaction t est toujours associée à un ensemble d'événements d'initiation (EI_t) et à un ensemble d'événements de terminaison (TE$_t$). Ici, nous montrons un ensemble d'axiomes fondamentaux qui sont applicables à tout modèle de transactions. Ces axiomes spécifient la relation entre les événements significatifs de type identique ou différent, ainsi qu'entre les événements significatifs et les opérations sur les objets.

Définition B.7 : Les axiomes fondamentaux des transactions

I $\forall\alpha \in EI_t\ \ (\alpha \in H^t) \Rightarrow \neg\exists\beta \in EI_t\ (\alpha \rightarrow \beta)$

Une transaction ne peut pas être initiée par deux événements différents.

II $\forall\delta \in EI_t\ \ \exists\alpha \in EI_t\ (\delta \in H^t) \Rightarrow (\alpha \rightarrow \delta)$

Si une transaction est terminée, elle a dû auparavant être initiée.

III $\forall\gamma \in ET_t\ \ (\gamma \in H^t) \Rightarrow \neg\exists\delta \in ET_t\ (\gamma \rightarrow \delta)$

Une transaction ne peut pas être terminée par deux événements différents.

IV $\forall$ob $\forall$p (p$_t$[ob] $\in$ H) $\Rightarrow$ (($\exists\alpha \in$ EI$_t$ ($\alpha \rightarrow$ p$_t$[ob])) ^ ($\exists\gamma \in$ ET$_t$ (p$_t$[ob] $\rightarrow\gamma$)))

Seules les transactions en cours peuvent demander des opérations sur les objets. Cet axiome énonce simultanément qu'une transaction doit toujours être initiée et terminée.

Les transactions atomiques

Nous considérons les transactions atomiques comme des transactions plates avec les propriétés AID. Elles sont sérialisables et atomiques. Autrement dit, elles sont exécutées sans interférence, comme si elles respectaient un ordre séquentiel. De plus, soit toutes les opérations d'une transaction sont exécutées soit aucune d'entre elles. Dans la définition qui suit, les axiomes 8, 9 et 10, définissent la sémantique de l'événement commit en termes de l'opération commit définie sur les objets. Les axiomes 11 et 12 définissent la sémantique de l'événement abort en termes de l'opération abort définie sur les objets.

Définition B.8 : Définition axiomatique des transactions atomiques

1. ES_t = {begin, commit, abort}

2. EI_t = {begin}

3. ET_t = {commit, abort}

4. t satisfait les axiomes fondamentaux I à IV

5. V isibilité$_t$ = H_{ct}
Une transaction perçoit l'état courant de tous les objets dans la base de données.

6. Conf litSet$_t$ = $\{p_{t'} [ob] \mid t' \neq t, Inprogress(p_{t'} [ob])\}$
Les effets des conflits concernent toutes les opérations en cours réalisées par d'autres transactions.

7. $\forall ob \; \exists p \; (p_t [ob] \in H) \Rightarrow$ (ob est atomique)
Tous les objets sur lesquels une transaction demande une opération sont atomiques (ob est atomique s'il est correct et sérialisable).

8. $(commit_t \in H) \Rightarrow \neg(t \; C^* \; t)$
Une transaction atomique peut valider seulement si elle ne fait pas partie d'un cycle dans une relation C (cf. définition B.3).

9. $\exists ob \; \exists p \; (commit_t [p_t [ob]] \in H) \;) \; (commit_t \in H)$
Si une opération sur un objet valide, la transaction qui en fait la demande doit valider.

10. $(commit_t \in H) \Rightarrow \forall ob \; \forall p \; ((p_t [ob] \in H) \Rightarrow (commit_t [p_t [ob]] \in H))$
Si une transaction valide, toutes les opérations demandées par la transaction valident aussi.

11. $\exists ob \; \exists p \; (abort_t [p_t [ob]] \in H) \Rightarrow (abort_t \in H)$
Si une opération sur un objet annule, la transaction qui en fait la demande doit annuler.

12. $(abort_t \in H) \Rightarrow \forall ob \; \forall p \; ((p_t [ob] \in H) \;) \; (abort_t [p_t [ob]] \in H))$
Si une transaction annule, toutes les opérations demandées par la transaction sont aussi annulées.

Théorème B.1 : Propriétés des transactions atomiques :

1. Si t est une transaction atomique, t est failure atomic,

2. Un ensemble T de transactions atomiques validées est sérialisable.

Preuve La preuve est donnée par [Chr91]. Nous tenons juste à souligner que la condition 2 est dérivée de la relation C établie entre les transactions avec des opérations en conflit (Axiome 7) ainsi que du critère de sérialisabilité établie dans l'Axiome 8. C'est-à-dire, les transactions respectent un ordre sérialisable si les opérations en conflit sont sérialisées et s'il n'existe pas de cycle entre les transactions.

Nous considérons qu'une transaction atomique satisfait les propriétés AID. La propriété de durabilité est assurée lors de la validation des opérations sur les objets (cf. section B.1).

B.5.4 Transactions réparties et emboîtées

Le deux théorèmes qui suivent présentent les propriétés des transactions réparties et emboîtées fermées (cf. section 2.3.1), nous en ferons référence dans le chapitre 4.

Théorème B.2 : Propriétés des transactions réparties :

1. Si t_d est une transaction répartie alors t_d est setwise failure atomic,

2. Un ensemble de transactions réparties validées S est sérialisable.

Preuve La preuve est donnée par [Chr91].

Théorème B.3 : Propriétés des transactions emboîtées :

1. Un ensemble de transactions emboîtées T est sérialisable,

2. Les opérations sont validées seulement par la transaction racine :
$\forall ob\ \forall p\ \forall t\ ((p_t\ [ob] \in H) \wedge (commit_{tr}\ [p_t\ [ob]] \in H)) \Rightarrow (t_r$ est une transaction racine),

3. Si une transaction t_0 annule, toutes les opérations réalisées par t_0 et ses descendantes annulent :
$(abort_{t0} \in H) \Rightarrow$
$\forall t, (t = t_0 \vee t \in Descendants(t_0))\ \forall ob\ \forall p\ ((p_t\ [ob] \in H) \Rightarrow (abort[p_t\ [ob]] \in H))$

Preuve La preuve est donnée par [Chr91].

Annexe C

Concepts pour les middlewares

Réflexivité

Le concept de la réflexivité a été introduit par Smith dans sa thèse de doctorat en 1982 [Smi82]. Selon l'auteur, un programme peut posséder une représentation de lui-même, et peut s'en servir pour raisonner sur lui-même (introspection) et éventuellement pour modifier sa propre interprétation (adaptation). Dans l'architecture d'un système réflexif, la séparation des préoccupations fonctionnelles et extra fonctionnelles est assurée par l'utilisation de deux niveaux de programmation : le niveau de base qui définit le code fonctionnel d'un système et le méta-niveau qui correspond au code extra fonctionnel du système. Les opérations assurant le passage d'un niveau à un autre sont la réification et la réflexion. Ces deux opérations suivent un protocole d'interactions appelé protocole à méta-objets (en anglais, MOP pour *Meta-Object Protocol*) [KRB91]. La figure 1 tirée de [Kra03] illustre l'ensemble des concepts précédemment définis.

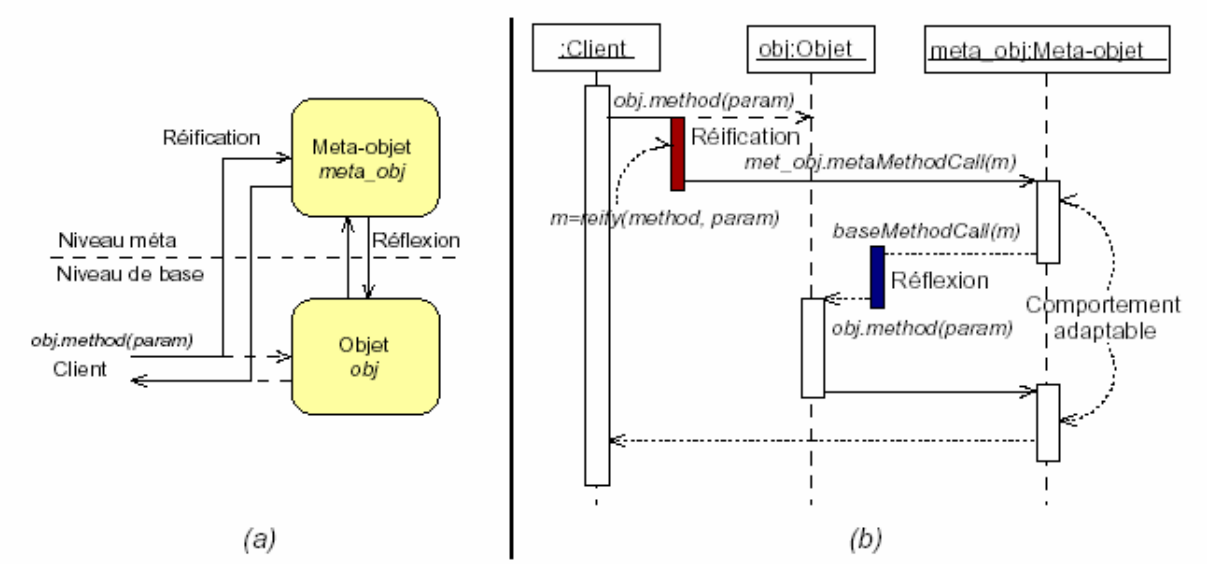


Figure 1 – Principe de fonctionnement des langages à méta-objets

Paradigme composant/conteneur

Il n'existe pas aujourd'hui une définition unique de composant logiciel. Toutefois, plusieurs définitions ou caractéristiques sont acceptées. Szyperski définit un composant comme « une unité de composition avec des interfaces contractuellement spécifiées et des dépendances explicites sur son contexte. Un composant peut être déployé indépendamment et il est sujet à des compositions par des parties tiers » [Szy97]. L'architecture logique d'un composant illustrée dans la figure 2 est caractérisée par deux parties [Vol01]. La première partie est fonctionnelle, elle représente le code métier du composant. Généralement, cette partie est appelée le contenu ou composant.

La deuxième partie est extra fonctionnelle, gérée par le conteneur qui s'occupe, par exemple, de la gestion du cycle de vie des composants et des connexions inter-composants. La communication entre

le conteneur et le composant se fait à travers des interfaces. Des interfaces internes offertes par le conteneur et utilisées par le développeur du composant aident à l'implantation du comportement du composant. Des interfaces de rappel offertes par le composant et utilisées par le conteneur sont disponibles pour la configuration des préoccupations extra fonctionnelles.

Le conteneur contrôle toutes les interactions du composant avec le monde extérieur. Ce contrôle est réalisé à travers des entités appelées objets de contrôle. Dans la plupart des modèles de composants, le conteneur offre au minimum deux objets de contrôle, un intercepteur pré-invocation et un intercepteur post-invocation. L'intercepteur pré-invocation est un objet de contrôle externe (exporté par le conteneur) ; il est visible de l'extérieur du composant et offre les mêmes interfaces que le composant. L'intercepteur post-invocation est un objet de contrôle interne (local) ; il n'est visible que de l'intérieur du conteneur. Le conteneur peut contenir d'autres objets de contrôle. Chaque objet de contrôle peut représenter un accès vers un service extra fonctionnel de l'intergiciel (middelware) ou bien réalise un service que le conteneur utilise dans la gestion des composants. Ces services peuvent être interrogés lorsque le composant reçoit ou émet des invocations.

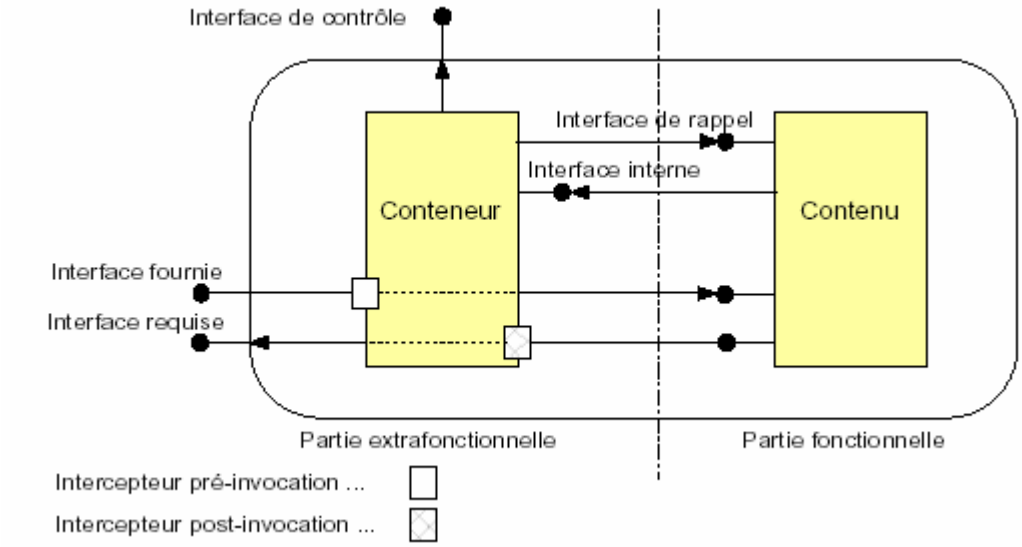


Figure 2 – Structure logique d'un composant

Notion de service

Dans le modèle de conception orienté composant, le composant utilise et fournit des fonctionnalités accessibles à travers des connexions entre les composants. L'application répartie est vue comme un assemblage de plusieurs composants qui interagissent entre eux pour accomplir les fonctionnalités de l'application. Chaque ensemble d'interactions est vu comme une composition logique de composants qui définit un service de l'application. Ainsi, l'application dans son ensemble peut être vue comme un ensemble de services.

Table des matières

Introduction **1**
Motivations et Objectifs 1
Contributions 3
Organisation du document 4

Chapitre I
Impact de la mobilité sur le traitement transactionnel **5**
1.1 Caractéristiques de l'environnement mobile 5
1.1.1 Architecture du système mobile 6
1.1.2 Les problèmes induits par un système mobile 7
1.2 Des systèmes répartis aux systèmes pour environnements mobiles 8
1.3 Transactions : concepts et impact de la mobilité 11
1.3.1 Concepts de base du traitement transactionnel 11
1.3.1.1 Les propriétés ACID 12
1.3.1.2 Contrôle de concurrence 12
1.3.1.3 Validation et reprise 14
1.3.2 Impact de l'environnement mobile sur les transactions 15
1.3.3 Modèle d'exécution de transaction mobile 16
1.4 Conclusion 18

Chapitre II
Gestion des transactions **19**
en environnement mobile **19**
2.1 Les modèles de transactions 19
2.1.1 Evolution du modèle de transaction 20
2.1.1.1 Le modèle linéaire 20
2.1.1.2 Les modèles avancés ou étendus 20
2.1.1.3 Le modèle emboîté 21
2.1.1.4 Le modèle emboîté fermé 21
2.1.1.5 Le modèle emboîté ouvert 22
2.1.1.6 Le modèle multi niveaux 22
2.1.1.7 Le modèle Saga 23
2.1.1.8 Le modèle flexible 23
2.1.1.9 DOM 24
2.1.2 Analyse et applicabilité à l'environnement mobile 24
2.2 Techniques de transaction pour environnements mobiles 26
2.2.1 Les propositions académiques 26
2.2.1.1 Reporting et Co-transactions 26
2.2.1.2 Le modèle de validation de transaction par Batch (par lot) 27
2.2.1.3 Isolation-Only Transactions (IOT) 27
2.2.1.4 Multi Database System Transaction Processing Manager (MDSTPM) 27
2.2.1.5 Clustering 28
2.2.1.6 Semantic-based 28
2.2.1.7 Two-tiers Replication 29
2.2.1.8 Twin-Transactions model 29
2.2.1.9 Kangaroo 30
2.2.1.10 Pro-Motion 30

2.2.1.11 Pre-serialisation ou Toggle 30
2.2.1.12 Terminaux Alternatifs 31
2.2.1.13 Moflex 32
2.2.1.14 Prewrite 32
2.2.1.15 Team Transaction Model 33
2.2.1.16 HiCoMo 33
2.2.1.17 Le modèle AMT 34
2.2.1.18 Discussion 35
2.2.2 Les approches commerciales 39
2.2.2.1 Informix Cloudscape 40
2.2.2.2 Sybase Anywhere 40
2.2.2.3 Microsoft SQL Server CE 41
2.2.2.4 WebSphere Everyplace et DB2 Everyplace 41
2.2.2.5 Oracle9iAS Wireless et Oracle9i Lite 42
2.2.2.6 PointBase 42
2.2.2.7 FastObjects j2 43
2.2.2.8 Discussion 43
2.3 Autres techniques de traitement transactionnel 44
2.3.1 Traitement transactionnel en environnement ad hoc 44
2.3.2 Motivation et challenges pour les techniques transactionnelles dans les MANETs 45
2.3.2.1 Effet de l'environnement sur les modèles de transaction 45
2.3.2.2 Effets des applications sur les modèles de transaction 46
2.3.2.3 Le modèle Team Transaction 46
2.3.2.4 Déconnexions planifiées 46
2.3.2.5 Gestion de transaction temps-réel en conservant l'énergie *(power-aware)* 47
2.3.2.6 Un système de transaction auto-organisant pour les réseaux ad hoc 48
2.3.3 Traitement transactionnel pour applications commerciales dans un MANET 48
2.3.4 Le modèle de Transaction Neighborhood pour l'informatique diffuse 49
2.3.5 Discussion 50
2.3.6 Les contributions dans les environnements de diffusion 51
2.3.6.1 L'approche de diffusion multiversion 52
2.3.6.2 L'approche des rapports de certification 53
2.3.6.3 Approche de validation partielle sur le client 53
2.3.6.4 L'approche OCC avec estampille de mise à jour et la pré-déclaration 54
2.3.6.5 Discussion 54
2.4 L'adaptation dans les systèmes mobiles 56
2.5 Conclusion 59

Chapitre III
La validation atomique 61
3.1 La validation de transaction dans les systèmes distribués 61
3.1.1 Définition de la validation atomique 62
3.1.2 Le protocole de validation à deux phases 62
3.1.3 Les optimisations et les variantes du protocole 2PC 63
3.1.4 Le blocage des protocoles de validation 64
3.1.4.1 Validation atomique non bloquante et consensus 65
3.2 Impact de la mobilité sur la validation des transactions 67

3.2.1 Les solutions proposées dans le cadre de modèles de transactions mobiles 69
3.2.2 Protocoles répartis de validation pour l'environnement mobile 70
3.2.2.1 Le protocole TCOT 71
3.2.2.2 Le protocole UCM 72
3.2.2.3 Le protocole CO2PC 73
3.2.2.4 Le protocole O2PC-MT 75
3.2.2.5 Le protocole ICPM 76
3.2.3 Discussion 77
3.3 Conclusion 79

Chapitre IV
Proposition de protocoles de validation de transactions mobiles **81**
4.1 Mobile Two-phase Commit Protocol (M2PC) 82
4.1.1 Le principe 82
4.1.1.1 Cas d'un client mobile et de serveurs fixes 82
4.1.1.2 Cas de client et serveurs mobiles 83
4.1.1.3 Prise en charge des caractéristiques mobiles 84
4.1.2 Correction du protocole M2PC 84
4.1.3 Evaluation du protocole M2PC 85
4.1.3.1 Impact de la taille du journal 87
4.1.3.2 Impact du délai d'acquisition d'une adresse IP 88
4.1.3.3 Impact du nombre de participants fixes 88
4.1.3.4 Impact du nombre des participants mobiles 88
4.1.3.5 Impact de la charge du réseau sans fil 89
4.1.3.6 Résumé de l'étude du protocole M2PC 90
4.1.4 Evaluation du mécanisme de M2PC pour le support de la mobilité 90
4.2 Un protocole décentralisé pour la validation dans les réseaux ad hoc 92
4.2.1 Le protocole D2PC 93
4.2.2 Le protocole D2PC pour un environnement ad hoc (A-D2PC) 93
4.2.3 La dissémination des informations du journal 94
4.2.4 Traitement des déconnexions 95
4.2.5 Performance de A-D2PC 96
4.2.6 Travaux similaires 96
4.3 Conclusion 97

Chapitre V
Etude comparative des protocoles de validation de transactions mobiles **98**
5.1 Modèle de simulation 98
5.2 Les paramètres 101
5.3 Résultats des simulations 103
5.3.1 Throughput vs. MPL et latence vs. MPL 103
5.3.2 Impact du handoff 107
5.3.3 Impact des déconnexions 107
5.3.4 Nombre de messages vs. MPL 108
5.3.5 Taux d'annulation vs. MPL 109
5.3.6 L'effet de la localisation du coordinateur sur une station base 109
5.3.7 Taille du journal (log size) vs. throughput et latence 110
5.3.8 Discussion et résumé de la comparison 110
5.4 Conclusion 113

Chapitre VI
Modèle et protocole transactionnels adaptables 115
6.1 Motivation pour l'adaptation dynamique 115
6.1.1 Besoin de context-awareness et d'adaptation aux changements 116
6.1.2 Besoin d'adaptation pour satisfaire les exigences variables des applications 117
6.2 Un modèle de transaction flexible 118
6.2.1 Notions d'atomicité flexible 119
6.2.2 Structure des transactions adaptables (AdapT) 120
6.2.3 Adaptabilité en environnement sans fil et mobile 121
6.2.3.1 La flexibilité du modèle AdapT 122
6.2.3.2 Le descripteur de contexte 123
6.2.4 Propriétés des transactions AdapT 124
6.2.5 Critères de correction des transactions AdapT 126
6.2.6 Exemple 128
6.3 Spécification formelle du modèle AdapT 128
6.3.1 Definition axiomatique du modèle 128
6.3.2 La propriété d'atomicité 132
6.3.3 La propriété de sérialisabilité globale des transactions AdapT 135
6.4 Le protocole de validation adaptable *a*TCP 136
6.4.1 Situations de blockage 137
6.4.2 Reprise après panne 139
6.4.3 Caractéristiques de *a*TCP 139
6.5 La sérialisabilité globale 140
6.6 Conclusion 141

Chapitre VII
Architecture middleware pour l'adaptation 142
7.1 Architecture middleware 142
7.1.1 Le MTServeur 143
7.2 Approche context-aware pour l'implémentation du protocole *a*TCP 144
7.2.1 Adaptation basée sur les politiques 145
7.2.2 Contexte et perception de l'environnement Mobile 145
7.2.3 L'adaptation dans MATrans 146
7.2.3.1 La notification d'événements dans MATrans 146
7.2.3.2 La spécification de politiques 147
7.2.3.3 Les mécanismes d'adaptation 150
7.2.3.4 Fonctionnement de MATrans 152
7.2.3.5 Implémentation du protocole *a*TCP dans MATrans 153
7.2.3.6 Propriétés du protocole *a*TCP 153
7.2.3.7 Le protocole *a*TCP et le contrôle de sérialisation 154
7.2.3.8 Support des déconnexions 155
7.3 Conclusion 155

Conclusion
Résumé des contributions 156
Perspectives 157

Bibliographie 160

Annexe A
Rappel de la logique d'ordre zéro 178

Annexe B
Le formalisme ACTA 179

Annexe C
Concepts pour les middlewares 188

Liste des figures

Figure 1.1-Architecture d'un système mobile avec infrastructure mobile 6
Figure 1.2-Architecture d'un système mobile sans infrastructure (ad hoc) 6
Figure 1.3-Classification des sytèmes de bases de données mobiles 9
Figure 1.4-Diagramme d'état d'une transaction. 14
Figure 1.5-Un environnement d'exécution de transaction mobile 16

Figure 2.1-Exemple d'une transaction multi niveaux 23

Figure 3.1-Le protocole 2PC 63
Figure 3.2-Le protocole TCOT 71
Figure 3.3-Le protocole UCM 73
Figure 3.4-Le protocole CO2PC 74
Figure 3.5-Le protocole O2PC-MT 76
Figure 3.6-Architecture du système et séquence des messages dans le cas sans pannes 77

Figure 4.1-The M2PC protocol 83
Figure 4.2-Architecture des réseaux sans fil simulés 86
Figure 4.3-Impact de la taille du journal 88
Figure 4.4-Impact de l'acquisition de l'adresse IP 88
Figure 4.5-Impact des participants fixes…. ……..89
Figure 4.6-Impact des participants mobiles ……..89
Figure 4.7-Impact de la charge du réseau sans fil 90
Figure 4.8-Comparaison des protocoles 2PC et M2PC 91
Figure 4.9-Le protocole A-D2PC pour l'environnement ad hoc 94

Figure 5.1-Architecture du simulateur DBsim 100
Figure 5.2-Throughput de saturation dans IEEE 802.11 104
Figure 5. 3-Throughputs, sans mobilité 104
Figure 5. 4-Throughputs, mobilité intra-cellule (pas de handoff) 105
Figure 5.5-Latences, sans mobilité 106
Figure 5.6-Latences, mobilité intra-cellule (pas de handoff) 106
Figure 5.7-Impact du handoff sur le throughput 107
Figure 5.8-Impact du handoff sur la latence 107
Figure 5.9-Impact des déconnexions 108
Figure 5.10-Comparaison de la complexité en messages 1088
Figure 5.11-Comparaison des taux d'annulation 1099
Figure 5.12-Influence de la station base sur M2PC……...110
Figure 5.13-Influence de la station base sur CO2PC……. 110
Figure 5.14-Impact de la taille du journal……. 111

Figure 6.1-Structure d'arbre d'une transaction globale 120
Figure 6.2-Le protocole de validation globale 136
Figure 6.3-Diagramme de transition d'état du protocole adaptable *a*TCP 138
Figure 6.4-Exemple de l'accès aux tickets dans OTM. 141

Figure 7.1-Architecture de MATrans 143
Figure 7.2-Instance de document XML 147
Figure 7.3-Schéma de document XML 148
Figure 7.4-Architecture du MTServeur adaptable 151
Figure 7.5-Le protocole de validation globale 153
Figure 7.6-Graphe de sérialisation globale 154

Liste des Tables

Table 1.1-Compatibilité des verrous 13

Table 2.1-Synthèse des modèles de transaction avancés 25
Table 2.2-Table récapitulative des modèles de transaction académiques 38
Table 2.3-Récapitulatif des approches commercialisées 44
Table 2.4-Characteristiques des solutions pour le cas ad hoc et pervasif 51

Table 3.1-Comparaison des protocoles de validation mobile 79

Table 4.1-Les paramètres du réseau de communication sans fil 87
Table 4.2-Probabilités de Handoff 89
Table 4.3-Performances du protocole M2PC 90

Table 5.1-Les paramètres des fonctions des systèmes de base de données 102
Table 5.2-Les valeurs des paramètres 102
Table 5.3-Les paramètres des fonctions du réseau de communication sans fil 103
Table 5.4-Les valeurs des paramètres 103
Table 5.5-Comparison globale des protocoles 112
Table 5.6-Impact des approaches sur les performances 112
Table 5.7-Analyse globale des approches 113

Table 6.1-Propriétés du modèle Adapt 126