Modèles, protocoles et architectures pour transactions mobiles adaptables

Patricia Serrano-Alvarado, Claudia Roncancio, Michel Adiba, Cyril Labbé

Modèles, protocoles et architectures pour transactions mobiles adaptables

P. Serrano-Alvarado[1 2] — C.L. Roncancio[1] — M. Adiba[1] — C. Labbé[1]

[1] *Laboratoire Logiciels, Systèmes, Réseaux (LSR)*
BP 72, 38402 St Martin d'Hères, France
émail : Prénom.Nom@imag.fr
[2] *Actuellement INRIA Futurs - Projet Jacquard*
Laboratoire d'Informatique Fondamentale de Lille (LIFL)
UPRESA 8022 CNRS - UFR IEEA - Bâtiment M3
émail : Prénom.Nom@lifl.fr

RÉSUMÉ. *Cet article décrit nos propositions en matière de transactions mobiles. Nous considérons des environnements mobiles composés de diverses unités et gérant des bases de données. Pour nous une transaction mobile est une transaction où l'une des composantes au moins s'exécute sur une unité mobile. Nous décrivons les caractéristiques des environnements mobiles et nous proposons (1) un modèle de transactions mobiles adaptable au contexte d'exécution, le modèle AMT, (2) une étude analytique du modèle AMT qui permet de choisir la meilleure stratégie d'exécution d'une transaction mobile et (3) un intergiciel (TransMobi) comme support du modèle AMT. Nous avons également proposé un protocole spécifique de validation (CO2PC) qui combine une approche optimiste avec un protocole de validation à deux phases. Il autorise une validation optimiste ou non-optimiste des transactions composantes permettant ainsi plus de souplesse dans la validation.*

ABSTRACT. *This article describes our proposal related to Mobile Transactions. We consider Mobile Environments where fixed and mobile units are involved in a given global transaction. A mobile transaction is a transaction where at least one mobile unit takes part in its execution. We describe the main characteristics of these environments and we propose: (1) an adaptable mobile transaction model, called AMT. This model provides context awareness to mobile transactions; (2) an analytical study in order to improve performance and to choose the best execution strategy for mobile transactions, and (3) a middleware called TransMobi which interfaces application code and existing DBMS in order to define and to execute AMT mobile transactions. This middleware implements also a specific validation protocol, CO2PC that combines optimistic commit and the traditional 2PC.*

MOTS-CLÉS : *Transactions mobiles, environnements mobiles, adaptation contextuelle, protocoles de validation*

KEYWORDS: *Mobile transactions, mobile environments, context awareness, commit protocols.*

[e] soumission à *Ingénierie des Systèmes d'Information (ISI)*, le 20 juin 2005.

1. Introduction

Les avancées de la technologie des télécommunications sans fil et l'évolution des caractéristiques des unités mobiles comme les ordinateurs portables, les PDA (*Personal Digital Assistants*), les téléphones mobiles ou les cartes à puce, contribuent à la popularité croissante et à la diversité des applications mobiles. Un des objectifs de l'informatique mobile, en ce qui concerne la gestion de données, est de permettre aux utilisateurs mobiles d'accéder et de manipuler des données à partir de n'importe quel emplacement, à n'importe quel moment et à partir de n'importe quel type de poste (n'importe où, n'importe quand). L'accès se fait sur des bases de données installées sur des unités fixes ou mobiles [BER 03, BER 04, RON 04].

Un environnement mobile se compose d'Unités Mobiles (UM) et d'Unités Fixes (UF). Les UM sont généralement limitées en mémoire, en capacité de calcul, d'affichage et principalement en autonomie de batterie. Par ailleurs, la capacité de communication entre les UM et les UF est asymétrique. En effet, les UF n'ont pas de contraintes d'énergie et bénéficient généralement de canaux de communication à haut débit, capables de disséminer d'information à un large nombre d'UM.

Afin d'offrir aux utilisateurs une connexion continue (depuis le bureau, à l'aéroport, en centre ville, en train, en pleine mer, etc.), il est indispensable de faire cohabiter les technologies de communication. Cette cohabitation contribue à augmenter la variabilité des capacités de communication. Ainsi, passer d'un WLAN au GSM fait baisser de façon importante la bande passante. La situation est similaire pour le prix des communications. Malgré la cohabitation des réseaux, il existe encore des zones où la communication n'est pas possible. Entrer dans ces zones entraîne la coupure de la communication (déconnexion). Dans un contexte mobile, une déconnexion n'est pas considérée nécessairement comme une défaillance mais peut être vue comme un état normal du système.

Dans ce travail nous nous intéressons à la gestion des transactions mobiles. De manière très générale, nous considérons qu'une transaction mobile est une transaction où au mois une unité mobile participe à l'exécution.

Les applications mobiles doivent prendre en compte les particularités de l'environnement mobile. Ainsi, l'un des principaux défis de l'informatique mobile, en général, est de gérer au mieux les variations du contexte sans perdre pour autant le contrôle sur la qualité des services. Dans un environnement traditionnel, les concepteurs d'applications n'ont pas besoin de prendre en compte ni les capacités des participants ni les caractéristiques du réseau. Dans un environnement mobile, nous pensons qu'il est indispensable de donner les moyens aux concepteurs d'adapter l'exécution d'une transaction à l'environnement courant. Cela permet de surmonter les contraintes de la mobilité et d'adapter au mieux les performances de l'exécution des transactions.

Afin de gérer les transactions dans les environnements mobiles, plusieurs travaux de recherche et quelques produits commerciaux ont vu le jour (voir l'état de l'art présenté dans [SER 04b]). Cependant, ces propositions n'abordent pas de manière satisfaisante l'adaptabilité au contexte mobile. La plupart s'adaptent uniquement aux

déconnexions. De même les propositions actuelles sont limitées à des exécutions ne faisant intervenir qu'une seule UM.

Nous proposons un modèle de transactions adaptables AMT (*Adaptable Mobile Transaction*) ainsi que l'intergiciel TransMobi permettant sa mise en œuvre [SER 04a]. L'idée générale du modèle AMT est d'associer plusieurs plans d'exécution à une même transaction. Ceci permet de choisir le plan d'exécution le plus adapté à un contexte mobile particulier. L'étude analytique réalisée montre que l'utilisation du modèle AMT doit permettre d'améliorer la probabilité de validation d'une transaction tout en maîtrisant le coût d'exécution. L'intergiciel TransMobi garantit les propriétés transactionnelles en s'appuyant, entre autre, sur le protocole de validation original CO2PC que nous proposons. Ce protocole, en combinant un protocole de validation à deux phases avec une approche optimiste, autorise aussi bien les validations optimistes que non-optimistes.

La section 2 définit la notion d'adaptation contextuelle dans les environnements mobiles. La section 3 décrit le modèle AMT et l'étude analytique, effectuée dans la section 4, souligne les bénéfices attendus. La section 5 présente une vue globale de l'intergiciel TransMobi. Les techniques utilisées pour garantir les propriétés transactionnels et le protocole CO2PC sont décrits dans la section 6. Un rapide panorama des travaux sur les transactions mobiles est donné dans la section 7. Les conclusions et les perspectives de notre travail sont présentées dans la section 8.

2. Environnements mobiles et adaptation contextuelle

Les caractéristiques des environnements mobiles nous amènent à introduire une adaptation contextuelle en se basant sur la perception de l'environnement. Nous mettons en évidence les dimensions susceptibles de varier dans l'environnement et d'avoir un impact sur l'exécution des transactions. Ces dimensions peuvent être divisées en trois groupes : celles portant sur des caractéristiques des réseaux sans fil, celles concernant l'unité mobile et des dimensions sémantiques.

– Les caractéristiques des réseaux sans fil qui nous intéressent sont l'état de la connexion, le débit de la bande passante et le coût monétaire de la communication. En effet, selon le réseau sans fil utilisé, le débit de la bande passante peut être très variable, les déconnexions fréquentes et la facturation significative (facturation au volume de données transféré ou au temps de connexion).

– Concernant l'unité mobile, nous nous intéressons à son autonomie en batterie, sa capacité de calcul et sa disponibilité en mémoire (cache ou persistante).

– Les caractéristiques sémantiques portent sur des aspects propres à l'application concernée ou au profil utilisateur. Nous incluons ici la localité (endroit où se trouve l'unité mobile) et le temps de connexion estimé avant la prochaine déconnexion. Les déconnexions pouvant être volontaires ou non, cette dernière caractéristique pourrait aussi être considérée dans les groupes précédents.

La définition des dimensions est flexible afin de permettre l'introduction de dimensions « sur mesure » . Par exemple, la qualité des données requises (fraîcheur) par une application peut devenir une dimension.

Pour chaque dimension, nous considérons un ensemble d'états possibles. Par exemple l'état de la connexion peut être *connecté* ou *déconnecté* ; le débit, le prix de communication, la batterie, le cache et la mémoire disponibles reflètent un certain niveau de qualité, *bon*, *moyen* ou *mauvais*. En pratique, ces états sont traduits par des intervalles dans les unités de mesure correspondantes. Pour les unités mobiles, la dimension localité concerne sa position géographique.

Comme nous le verrons dans la suite, les définitions des transactions incluront des descripteurs d'environnement spécifiant le(s) état(s) requis pour une certaine exécution. Si plusieurs sites participent à l'exécution d'une transaction, le descripteur peut décrire ou non l'état requis pour chaque site.

3. Modèle de transactions mobiles adaptables

Le modèle AMT, *Adaptable Mobile Transactions*, que nous proposons permet aux programmeurs de définir des alternatives transactionnelles pour une tâche applicative. Nous cherchons à offrir un cadre transactionnel souple qui associe des informations contextuelles et des alternatives d'exécution pour une action. Nous considérons des tâches ou actions impliquant des transactions sur une ou plusieurs bases de données ou sites. Le développeur d'applications peut envisager divers choix pour accomplir une même action, chacun d'eux correspondant à un certain critère de performance adapté à un état du contexte d'exécution. Une transaction de type AMT, notée T_{AMT}, permet d'introduire une ou plusieurs *alternatives d'exécution* associées chacune à un *descripteur d'environnement*. Un tel descripteur exprime l'état du contexte d'exécution requis pour le lancement de l'alternative correspondante (par exemple connexion et localité).

Les alternatives peuvent être sémantiquement équivalentes et l'exécution réussie de l'une d'entre elles représente l'exécution correcte d'une T_{AMT}. Ainsi, lorsqu'une T_{AMT} est lancée, l'état courant de l'environnement est examiné et l'alternative d'exécution appropriée démarre. Si l'état de l'environnement mobile ne permet l'exécution d'aucune alternative, l'exécution de la T_{AMT} est reportée. Dans ce cas, une alternative sera démarrée aussitôt qu'un état acceptable survient.

Le programmeur a un moyen de maîtriser certains aspects des performances de l'exécution des transactions. Ces aspects peuvent être le taux de réussite des transactions, les messages échangés ou encore le coût financier induit par l'exécution et les communications.

3.1. *Définition d'une transaction* T_{AMT}

Les transactions T_{AMT} ont une structure arborescente. Une T_{AMT} joue un rôle de coordinateur d'un ensemble d'alternatives d'exécution. Chaque alternative coordonne un ensemble de *transactions composantes*. L'accès aux données se fait seulement par les transactions composantes, qui sont considérées comme les feuilles de l'arbre. Le niveau T_{AMT} et les alternatives sont seulement des unités de contrôle. Si on fait une analogie avec les multi-transactions, les transactions composantes sont des sous-transactions introduites par les alternatives d'une T_{AMT}.

Dans la suite, nous présentons une définition intuitive du modèle AMT. Pour sa définition formelle voir [SER 04a] et [SER 04c].

Les états de l'environnement sont représentés par des *descripteurs d'environnement* qui indiquent les dimensions pertinentes et leurs valeurs.

Définition 1 *Le Descripteur de l'environnement (DE) contient un ensemble de dimensions et le(s) état(s) correspondants, DE = {Dimension = état(s)}*

Définition 2 *Une transaction mobile adaptable est définie par* $T_{AMT} = \langle AE_k \rangle$ *où :*

– $\langle AE_k \rangle$, $k > 0$, est une liste d'alternatives d'exécution. AE_k a priorité[1] sur AE_{k+1}.

– $AE_k = (DE_k, PE_k)$. Une alternative d'exécution contient un plan d'exécution PE_k qui sera lancé seulement si l'environnement mobile courant offre les caractéristiques spécifiées par le descripteur d'environnement DE_k.

– $DE_k = \{Dimension = etat(s)\}$. Un descripteur d'environnement contient un ensemble de dimensions avec leur(s) état(s).

– $PE_k = \{(t_{ki}, tc_{ki}, SiteId)\}$ est un ensemble dont chaque élément introduit une transaction composante t_{ki}, sa transaction de compensation tc_{ki} (si possible) et le site $SiteId$ où l'exécution doit avoir lieu. $SiteId$ indique la base de données et l'unité (mobile ou fixe) qui l'accueille.

Il existe une *relation de dépendance* $\mathcal{RD}$ entre les éléments de PE_k. Cette relation indique le parallélisme $(||)$ ou la séquentialité $(<)$ de l'exécution des transactions composantes du plan. Un plan peut donner lieu à une exécution sur plusieurs sites. Un site exécute au plus une transaction composante par alternative.

Les transactions composantes sont exécutées par les SGBD sous-jacents, sous leur responsabilité.

Une transaction composante peut avoir une transaction de compensation dont le rôle est de défaire, d'un point de vue sémantique, le travail fait par la transaction qu'elle compense. Les transactions de compensation ne sont pas obligatoires : une transaction composante peut ne pas être compensable mais aussi dans certains cas, une compensation n'est pas nécessaire (par exemple, dans le cas des transactions de lecture). Cependant, comme nous le verrons par la suite, leur présence augmente la souplesse de gestion des transactions et l'autonomie des sites.

3.2. *Exemple d'une transaction* T_{AMT}

Nous considérons un utilisateur disposant d'une UM (avec des capacités de stockage) et un magasin électronique composé de deux sites serveurs, installés sur le réseau fixe – SCatalogue et SAchat. Le premier site permet d'interroger le catalogue

1. Les priorités peuvent être déterminées selon : le coût d'exécution, la qualité des réponses, la probabilité de déclenchement, etc. des AE_k.

AE_k	DE_k	PE_k
k=1	{état-catalogue= *àjour*}	{(SélectionArticles, UM) < (CommandePaiement, SAchat)}
k=2	{état-connection=*connecté*, bande-passante = *forte, moyenne*, prix-communication=*modéré* état-catalogue=*present, absent*},	{(RécupereCatalogue, SCatalogue) < (SélectionArticles, UM) < (CommandePaiement, SAchat)}
k=3	{état-connection=*connecté*, bande-passante=*faible*, état-catalogue=*absent*}	{(RécupereCatalogue, SCatalogue) < (Sélection-PaiementLocal, UM) < (Commande, SAchat)}

Tableau 1. *Exemple* $T_{AMTachat}$.

du magasin tandis que le deuxième prend des commandes et gère les paiements. Nous définissons une $T_{AMTachat}$ contenant les transactions composantes suivantes permettant au client de :

– RécupereCatalogue récupérer le catalogue du magasin.

– SélectionArticles sélectionner des articles à partir d'une copie du catalogue.

– CommandePaiement d'envoyer une commande et de payer sur le serveur.

– PaiementLocal de payer sur l'UM, avec de l'argent électronique, sans communiquer avec les serveurs du magasin.

– Commande d'envoyer une commande (sans inclure le paiement).

– Sélection-PaiementLocal = SélectionArticles + PaiementLocal

Le tableau 1 introduit trois alternatives d'exécution qui utilisent les transactions composantes que nous venons de décrire. Une alternative est déclenchée si l'état de l'environnement mobile satisfait le descripteur d'environnement correspondant. Dans cet exemple, les caractéristiques du réseau mobile qui déterminent le choix d'une alternative sont : la disponibilité de la connexion, la bande passante et le prix de communication. La présence du catalogue sur l'UM est une caractéristique qui a été définie spécifiquement pour cette application. Cette caractéristique prend les valeurs *absent* (si le catalogue n'est pas sur l'UM), *présent* (si une version, probablement non à jour ou incomplète, existe dans l'UM) et *àjour* (si la dernière version est sur l'UM). Les caractéristiques qui n'ont pas d'importance pour cette application, n'apparaissent pas dans les descripteurs d'environnement.

Dans cet exemple, les alternatives d'exécution sont organisées selon leur coût d'exécution. L'exécution de AE_1 est moins cher que celle de AE_2. Dans AE_1, une version à jour du catalogue existe sur l'UM, ceci permet d'économiser des messages. Il est possible de déclencher AE_1 en mode déconnecté et la transaction CommandePaiement peut être différée à une prochaine reconnexion. L'alternative AE_2 est déclenchée si la qualité de communication est acceptable (connexion avec une bande passante au moins *moyenne* et à un prix *modéré*). AE_3 s'exécute lorsque la qualité de communication est mauvaise. L'avantage de cette dernière alternative est que Sélection-PaiementLocale peut être faite en mode déconnecté car le paiement est sur l'UM. La transaction Commande est déclenchée lors d'une prochaine réconnexion.

La relation de dépendance $\mathcal{RD}$ entre les transactions composantes de chaque alternative est de séquentialité (<). Les transactions de compensation pour cet exemple peuvent être composées principalement d'opérations pour annuler et rembourser des commandes.

3.3. *Propriétés des transactions* T_{AMT}

Pour présenter les propriétés du modèle AMT il est nécessaire de distinguer les trois niveaux : les transactions AMT, ses alternatives d'exécution et ses transactions composantes. Chaque niveau, effectue les opérations *begin, abort, commit*. Rappelons que les transactions composantes sont exécutées par les SGBD qui doivent assurer les propriétés ACID. Dans la suite nous nous concentrons sur les propriétés des alternatives et des T_{AMT} en considérant principalement les aspects atomicité, isolation et correction des transactions. L'aspect durabilité repose sur les SGBD sous-jacents après la validation des transactions composantes. Le choix des propriétés tient compte des caractéristiques des contextes mobiles et cherche à augmenter l'autonomie des unités mobiles. Les propriétés peuvent être résumées de la manière suivante : les T_{AMT} sont semi-atomiques et reposent sur l'atomicité sémantique des alternatives. Les alternatives incluant des transactions composantes compensables relâchent l'isolation. La sérialisabilité locale à un SGBD est de sa responsabilité alors que le niveau global est à la charge du support des T_{AMT}.

3.3.1. *Atomicité sémantique des alternatives*

Comme nous l'avons indiqué, les transactions composantes d'une alternative peuvent être exécutées sur une unité mobile ou sur une unité fixe. Nous cherchons à favoriser le travail autonome des unités mobiles, même en mode déconnecté, tout en limitant l'impact sur les autres participants du système. Tenant compte des limitations des ressources et des déconnexions courantes, le modèle AMT relâche l'atomicité des alternatives d'exécution. L'objectif est de limiter le blocage des ressources utilisées par les transactions composantes lorsque c'est possible. Ainsi, on distingue les transactions composantes compensables des transactions composantes non compensables. Les compensables sont autorisées à valider localement sans attendre la validation de l'alternative. Cette validation optimiste permet de libérer les ressources au plus tôt. Cependant, si l'alternative annule, les transactions de compensation correspondantes doivent être exécutées. [2]

Les transactions composantes non compensables ne peuvent pas valider de manière optimiste. Lorsqu'une transaction de ce type termine, les ressources utilisées seront retenues jusqu'à ce qu'une décision globale (de validation ou d'annulation) soit prise pour l'alternative. Dans ce cas, une annulation de l'alternative va simplement impliquer un *abort* local de la transaction composante.

---

2. Les transactions de compensation des transactions composantes exécutées séquentiellement sont exécutées dans l'ordre inverse de validation. Celles correspondant aux transactions composantes exécutées en parallèle sont exécutées de façon concurrente.

L'utilisation de transactions de compensation comme un moyen de récupération sémantique conduit à une atomicité sémantique [GAR 83] au niveau des alternatives. Dans le cas où une alternative contiendrait exclusivement des transactions composantes non compensables (aucune transaction de compensation n'est nécessaire), on obtient la *failure atomicity*. Ainsi, une AE_k contenant un ensemble de t_{ki}, est atomique sémantiquement si : (1) soit toutes les t_{ki} définies dans AE_k valident ; (2) soit toutes les t_{ki} définies dans AE_k sont compensées ou annulées.

3.3.2. *Semi-atomicité des transactions T_{AMT}*

La semi-atomicité est assurée si (1) la validation de la T_{AMT} implique celle d'une seule AE_k et l'annulation ou la compensation des sous-transactions d'autres AE_l (si elles ont été démarrées) et (2) l'annulation de la T_{AMT} implique l'annulation ou la compensation de toutes les sous-transactions de l'AE_k active.

3.3.3. *Ordonnancement global*

Les transactions composantes d'une T_{AMT} s'exécutent sur des SGBD différents. Malgré le fait que les sources des différents SGBD soient disjointes, les dépendances de précédence entre les transactions composantes d'une même alternative obligent à préserver un ordre d'exécution aussi bien à l'intérieur de l'AE_k qu'entre les AE_k concurrentes – appartenant à des T_{AMT} différentes.

Le critère utilisé pour contrôler la correction d'exécution des alternatives concurrentes est la *sérialisabilité globale*. La sérialisabilité globale des alternatives veut que chaque SGBD réalise une exécution localement sérialisable et qu'au niveau global, les transactions composantes soient perçues dans le même ordre relatif. Rappelons que sur chaque site, il existe une seule transaction composante par alternative et que des transactions locales au site (sauf les T_{AMT}) peuvent y être exécutées.

La sérialisabilité des T_{AMT} est fournie à travers celle des alternatives d'exécution. Par ailleurs, nous considérons qu'il n'y a pas de contraintes d'intégrité globales (inter bases), ainsi la cohérence sémantique est préservée.

4. Etude Analytique du Modèle AMT

Traditionnellement, le système exécute une transaction quel que soit l'état de l'environnement. Si l'environnement ne permet pas une exécution correcte d'une transaction, celle-ci est annulée même si une autre façon de l'exécuter aurait pu réussir. De façon similaire, les différents états de l'environnement génèrent des coûts différents. Permettre au système de choisir le type d'exécution en fonction de l'état de l'environnement, est une façon de borner le coût d'exécution.

Cette section a pour objectif de souligner les bénéfices obtenus grâce à l'adaptabilité offerte par le modèle AMT et grâce à la perception de l'environnement mobile. Nous verrons qu'il améliore les performances et permet de choisir la façon dont les transactions s'exécutent en fonction des coûts d'exécution.

Tout au long de cette section, nous utilisons l'exemple $T_{AMTachat}$ introduit dans la section 3.2.

	$j=1$ Bon	$j=2$ Moyen	$j=3$ Mauvais
état-connexion $i=1$	*connecté*		*déconnecté*
bande-passante $i=2$	*forte*	*moyenne*	*faible*
prix-communication $i=3$		*modéré*	*élevé*
état-catalogue $i=4$	*àjour*	*présent*	*absent*

$$P = \begin{bmatrix} 0.8 & 0 & 0.2 \\ 0.7 & 0.2 & 0.1 \\ 0 & 0.4 & 0.6 \\ 0.2 & 0.3 & 0.5 \end{bmatrix}$$

Figure 1. *Environnement mobile considéré pour $T_{AMTachat}$.*

4.1. *Modélisation de l'environnement*

Dans cette section, nous introduisons la matrice qui décrit les caractéristiques de l'environnement mobile.

En effet, un tel environnement peut être défini par un ensemble de dimensions avec ses états correspondants. Nous pouvons définir la probabilité que l'état courant de l'environnement coïncide avec les différents états des dimensions considérées.

Définition 3 *p_{ij} est la probabilité de la dimension i d'être dans l'état j.*

Dans la matrice qui en résulte $P = (p_{ij})$, $\forall i,\ \sum_j p_{ij} = 1$. P est dépend du réseau mobile, des unités mobiles utilisés ou encore des habitudes des utilisateurs.

Exemple 1 *La figure 1 montre l'environnement considéré pour $T_{AMTachat}$ et une matrice pouvant être associée à cet environnement. Les lignes correspondent aux dimensions (caractéristiques) de l'environnement. Les colonnes aux différents états que chaque dimension peut avoir à un instant donné. D'autres niveaux de qualité peuvent être introduits, les niveaux utilisés dans la figure 1 le sont à titre d'exemple. Les cases vides indiquent que la caractéristique correspondante de l'environnement n'a pas d'état pour ce niveau de qualité.*

4.2. *Matrice de déclenchement d'une alternative d'exécution*

Nous montrons ici comment spécifier l'environnement nécessaire pour l'exécution d'une alternative d'exécution (AE_k).

Définition 4 *Pour chaque AE_k=(DE_k, PE_k) nous dénotons par Δ^k la matrice booleenne où :*

$$\delta_{ij}^k = \begin{cases} 1 & si\ l'état\ j\ de\ la\ dimension\ i\ est\ acceptable\ pour\ AE_k \\ 0 & autrement \end{cases}$$

La perception de l'environnement mobile permet au système de choisir une AE_k si l'état de l'environnement correspond au DE_k associé. Puisque AE_k a une priorité plus grande que AE_{k+1} nous pouvons considérer, sans perte de généralité que :

Propriété 1 *Si l'état de l'environnement est adéquat pour une AE_k il ne l'est pas pour une $AE_{k'}$ de la même T_{AMT}. C'est-à-dire : $\forall (k,k'), k \neq k', \exists i$ tel que $\forall j, \delta_{ij}^{k'} \neq \delta_{ij}^k$*

Exemple 2 *Les matrices Δ^k qui correspondent aux trois alternatives de $T_{AMTachat}$ (cf. Tableau 1) sont :*

$$\Delta^1 = \begin{bmatrix} 1 & 0 & 1 \\ 1 & 1 & 1 \\ 0 & 1 & 1 \\ 1 & 0 & 0 \end{bmatrix} \Delta^2 = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 1 \end{bmatrix} \Delta^3 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{bmatrix}$$

4.3. *Probabilité de déclenchement d'une AE_k et de l'AMT*

Définition 5 *Soit q^k la probabilité pour que AE_k soit sélectionnée pour être exécutée. q^k sera appelée la probabilité de déclenchement d'AE_k. Comme la probabilité que la dimension i ait un état acceptable pour AE_k est donnée par $\sum_j \delta_{ij}^k p_{ij}$, nous avons :*

$$q^k = \prod_i (\sum_j \delta_{ij}^k p_{ij})$$

Une AE_k a la possibilité d'être initiée ($q^k > 0$) si pour chaque dimension i il existe un état j ($\forall i, \exists j$ tel que $\delta_{ij}^k = 1$).

Définition 6 *Soit q_{AMT} la probabilité de déclenchement de T_{AMT}. Grâce à la propriété 1, nous avons : $q_{AMT} = \sum_k q^k$.*

4.4. *Coût d'exécution*

Afin de calculer le coût d'exécution d'une alternative nous définissons ici une matrice de coût. Cette matrice est conçue en associant un coût aux dimensions et états considérés dans la figure 1.

Définition 7 *C^k est une matrice où c_{ij}^k est le coût d'exécution d'AE_k dans l'état j de la dimension i.*

c_{ij}^k doit être défini par le programmeur de l'application en fonction des besoins d'optimisation des coûts.

Une alternative d'exécution est lancée par le système lorsque l'environnement mobile est dans l'état j de la dimension i avec la probabilité par dimension : $\frac{\delta_{ij}^k p_{ij}}{\sum_j \delta_{ij}^k p_{ij}}$

Le coût moyen associé à la dimension i de l'exécution de AE_k est donné par :

$$c_i^k = \frac{\sum_j \delta_{ij}^k c_{ij}^k p_{ij}}{\sum_j \delta_{ij}^k p_{ij}}$$

Définition 8 *Notons c_i le coût moyen, associé à la dimension i, de l'exécution de T_{AMT}. En considérant que l'environnement est stable pendant l'exécution, AE_k est*

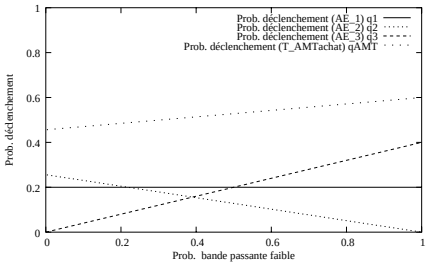

Figure 2. *Probabilité de déclenchement vs bande passante (T_{AMT} et AE_k).*

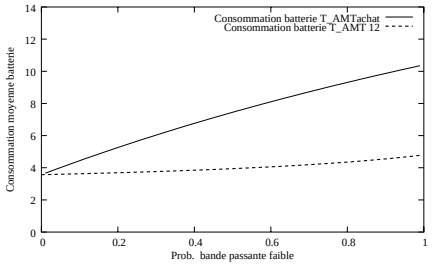

Figure 3. *Consommation de batterie vs bande passante (deux T_{AMT}).*

initiée avec une probabilité q^k, donc le coût de la T_{AMT} par rapport à une dimension est c_i^k, avec une probabilité q^k, d'où :

$$c_i = \frac{\sum_k c_i^k q^k}{\sum_k q^k}$$

4.5. *Exemple*

Afin d'étudier les AE_k dans de différents types d'environnement, nous nous proposons d'analyser les indices de performance en fonction de la probabilité que bande-passante = *faible* (p_{23}) :

$$P = \begin{bmatrix} 0.8 & 0 & 0.2 \\ (1-p_{23})/2 & (1-p_{23})/2 & p_{23} \\ 0 & 0.4 & 0.6 \\ 0.2 & 0.3 & 0.5 \end{bmatrix}$$

La figure 2 montre la probabilité de déclenchement des AE_k. Nous pouvons voir que si la bande passante est souvent *faible* (par exemple p_{23} = 0.8), l'alternative EA_3 a la probabilité la plus forte d'être initiée. q^1 est constante car elle ne dépend pas de la probabilité de bande-passante = *faible* mais uniquement de état-catalogue = à*jour*. Nous pouvons voir que $T_{AMTachat}$ a une meilleure probabilité de déclenchement que les différentes alternatives prises séparément.

Pour la transaction $T_{AMTachat}$ nous pourrions identifier l'utilisation de la mémoire comme un coût associé à la dimension état-catalogue ou la consommation du CPU comme un coût associé à la dimension état-connection, en considérant qu'en mode déconnecté plus d'opérations sont faites sur l'unité mobile. On choisit ici de ne considérer que des coûts liés au prix de communication et à l'utilisation de la batterie en les associant respectivement aux dimensions prix-communication et bande-passante. Une bande passante réduite accroît la consommation de batterie car le temps d'exécution augmente. Le réseau sans fil utilisé est UMTS. C'est un réseau à commutation par paquets où la bande passante peut varier théoriquement de 144 Kbps (avec une mobilité véhiculaire) , 384 Kbps (mobilité à pied) à 2 Mbps à l'intérieur d'un bâtiment. Le prix de communication dépend du volume des données transmises (paquets). Par exemple, l'exécution d'AE_1 a besoin de trois messages transmis par le

réseau sans fil. Tout d'abord, il y a un message d'enregistrement (*login*), ensuite, un autre pour la commande d'achat (transaction composante CommandePaiement), finalement, un message est reçu par l'UM pour confirmer la commande (acquittement). Les exécutions d'AE_2 et d'AE_3 ont besoin de 2 messages supplémentaires. L'un pour demander le catalogue et l'autre pour son transfert (transaction composante RécupereCatalogue).

Nous identifions trois types de messages : petits (login, acquittement et demande de catalogue), moyens (pour CommandePaiement et Commande) et grands messages (pour RécupereCatalogue). Nous considérons que les messages petits, moyens et grands sont respectivement composés par 1, 10 et 20 paquets. Si le plan d'exécution d'AE_k comprend n_s petits, n_m moyens et n_l grands messages, alors le nombre total de paquets envoyés ou reçus[3] par l'UM pendant l'exécution est $n_p = n_s + 10n_m + 20n_l$.

Afin d'estimer le coût de communication (prix de communication et consommation de batterie) nous considérons qu'envoyer un seul paquet dans l'état *modéré* (respective *élevé*) coûte une unité de prix (respective deux unités). De plus, si la bande passante est forte (respective *moyenne*, *faible*) envoyer/recevoir un paquet utilise 0.1% (respective 0.2%, 0.4%) de la capacité de la batterie. Dans ce cas, le coût associé à la dimension bande-passante est la consommation de la batterie.

Dans $T_{AMTachat}$, $n_p = 2 + 10 + 0 = 12$ pour AE_1 et $n_p = 3 + 10 + 20 = 33$ pour AE_2 et AE_3, ainsi les coûts d'exécution des trois alternatives peut être calculé :

$$C^1 = \begin{bmatrix} 0 & 0 & 0 \\ 1.2 & 2.4 & 4.8 \\ 0 & 12 & 24 \\ 0 & 0 & 0 \end{bmatrix} \quad C^2 = C^3 = \begin{bmatrix} 0 & 0 & 0 \\ 3.3 & 6.6 & 13.2 \\ 0 & 33 & 66 \\ 0 & 0 & 0 \end{bmatrix}$$

La figure 3 montre la consommation moyenne de batterie pour $T_{AMTachat}$ ainsi que pour une variante sans AE_3. Cette nouvelle T_{AMT} est appelée T_{AMT12}. Nous pouvons voir que $T_{AMTachat}$ a une meilleure probabilité de déclenchement. Ceci est dû au fait qu'elle dispose d'une AE_k supplémentaire (AE_3). La consommation de batterie moyenne de $T_{AMTachat}$ augmente avec la probabilité que la bande passante soit *faible*. Ceci est dû au fait que la probabilité de déclenchement d'AE_3 est plus grande que celle d'AE_2 (cf. figure 2). Par conte T_{AMT12} a une meilleure consommation moyenne de batterie car elle ne comporte pas AE_3 qui a un coût très élevé.

4.6. *Bilan sur l'analyse des performances*

Le type d'analyse présenté ici permet de faire des estimations liées aux variations d'état de l'environnement mobile et à l'exécution des transactions type AMT (T_{AMT}). D'une part, la modélisation de l'environnement mobile permet de calculer la probabilité de déclenchement des T_{AMT} et des alternatives d'exécution (AE_k). D'autre part, la modélisation des coûts d'exécution de plusieurs alternatives dans les différents états de l'environnement mobile, permet d'établir un compromis sur les critères de qualité.

3. Éventuellement, il pourrait être intéressant de distinguer l'envoie et la réception des messages. Pour l'UM il est moins coûteux de recevoir des messages que de les envoyer.

En jouant sur le nombre d'alternatives d'exécution disponibles, le concepteur d'applications peut choisir entre de nombreuses combinaisons pour une même T_{AMT}. Il faut souligner que la conception de la T_{AMT} qui donne les meilleures probabilités de déclenchement ou les meilleurs coûts d'exécution ne dépend pas du nombre d'alternatives définies mais de la capacité de s'adapter aux variations d'état de l'environnement.

En conclusion, l'adaptabilité aux variations d'état de l'environnement mobile offre l'opportunité d'améliorer les performances et les coûts d'exécution. Cependant, la conception de transactions adaptables peut s'avérer très compliquée. Pour faire face à ce degré de complexité, des outils d'aide à la conception basés sur le type d'analyse faite ici peuvent être définis.

5. L'Intergiciel TransMobi

5.1. *Vue générale*

TransMobi est un intergiciel situé entre le code applicatif et des SGBD existants. Les SGBD sous-jacents et les applications peuvent être localisés sur des unités mobiles et/ou fixes. TransMobi suit une architecture de type *client-agent-serveur* répartie entre des unités mobiles et fixes.

La capacité de gestion des SGBD considérés est très variable. Ils peuvent être très puissants (généralement localisés sur des unités fixes) ou très légers (généralement localisés sur des unités mobiles). L'existence de plusieurs « SGBD légers » commerciaux – comme FastObjects [Fas 02], PointBase [Poi 02] ou Oracle9i Lite [Ora 02] – permet de considérer des unités mobiles capables de gérer des données localement. TransMobi, dans son rôle d'intégrateur, tient compte de l'hétérogénéité des SGBD intégrant le système mobile – aucun type de SGBD particulier n'est ciblé. Afin d'être compatible avec le plus grand nombre de SGBD, les interfaces requises par TransMobi sont uniquement celles de *begin, commit, abort*. TransMobi rend transparent les variations de l'environnement mobile aux SGBD sous-jacents. Les SGBD ne s'occupent pas de l'environnement mobile, ils n'ont pas connaissance du caractère mobile des clients ou des sources.

D'une manière très générale, le fonctionnement global se déroule de la manière suivante. L'application demande à TransMobi l'exécution d'une transaction T_{AMT}. Selon l'état courant de l'environnement mobile, TransMobi choisit, parmi les alternatives d'exécution de la T_{AMT}, la plus adéquate. Ensuite, il distribue les transactions composantes aux SGBD correspondants. TransMobi doit assurer un ordre correct d'exécution globale ainsi que la validation de la T_{AMT}.

Cette section introduit les trois tiers de l'architecture de TransMobi (section 5.2) ainsi que la manière de percevoir l'environnement mobile (section 5.3).

5.2. *Architecture à trois tiers de TransMobi*

Dans TransMobi le tiers client est nommé *TMClient*, l'agent *TMAgent* et le serveur *TMServeur* (cf. figure 4). Seul le TMAgent est contraint d'être localisé sur une unité fixe. Les autres tiers peuvent être situés aussi bien sur des unités mobiles que fixes.

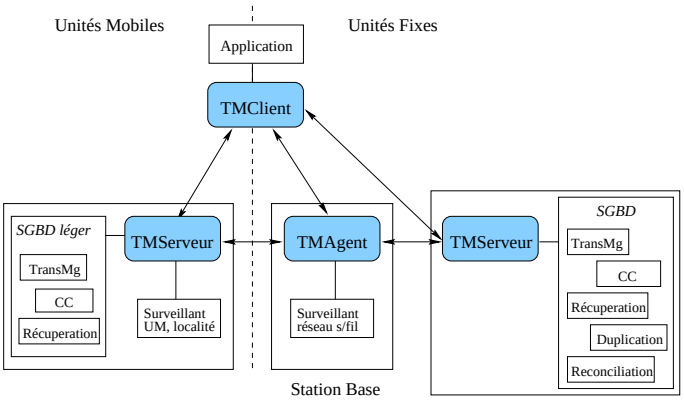

Figure 4. *Architecture client-agent-serveur de TransMobi*

Les tiers installés sur des unités mobiles doivent être attachés à un TMAgent. Toute communication entre un tiers mobile et un tiers fixe doit passer par un TMAgent.

TMClient

Un *TMClient* interagit directement avec l'application mobile. Il est appelé *TMClient-fixe* ou *TMClient-mobile* selon qu'il s'exécute sur des unités fixes ou mobiles. Il existe un TMClient par site qui peut interagir avec plusieurs applications mobiles et avec les TMServeurs des différents SGBD sous-jacents. Un TMClient-mobile doit être attaché à un TMAgent pour pouvoir communiquer avec des TMServeurs-fixes. Sur un même site, il peut y avoir une application et un SGBD. Dans ce cas, le TMClient et le TMServeur peuvent interagir directement qu'ils soient fixes ou mobiles.

Le TMClient reçoit, de l'application, la demande d'exécution d'une T_{AMT}. Il analyse les descripteurs d'environnement des alternatives de la T_{AMT} et selon l'état courant de l'environnement, il décide quelle alternative peut démarrer. Il est possible qu'au moment du déclenchement de la T_{AMT}, aucune alternative d'exécution ne puisse démarrer. Dans ce cas, l'exécution de la T_{AMT} peut être reportée jusqu'à ce que l'état de l'environnement soit adéquat ou alors elle est annulée.

TMServeur

Le *TMServeur* interagit directement avec le SGBD sous-jacent. Il est appelé *TMServeur-fixe* ou *TMServeur-mobile* selon qu'il s'exécute sur des unités fixes ou mobiles. Un TMServeur interagit avec les SGBD serveurs 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*). Il participe au processus de validation globale (cf. section 6.1). Le TMServeur peut coordonner le processus de validation globale et gérer le *graphe de sérialisabilité globale* (cf. section 6.2) lors-

qu'il est un TMServeur-fixe. Lorsqu'il est mobile, le *TMServeur* interagit avec un surveillant de l'état des ressources locales (cf. section 5.3). Un TMServeur-mobile fournit l'information sur les ressources de l'unité mobile où il s'exécute et éventuellement sur les services offerts.

Dans le processus global, le TMServeur reçoit les transactions composantes et coordonne leur exécution sur le SGBD sous-jacent.

TMAgent

Les *TMAgents* sont installés sur des stations base ou sur des unités fixes capables de communiquer avec des unités mobiles. Le TMAgent est un représentant des tiers mobiles qui stocke les messages destinés aux unités mobiles déconnectées. Il peut aider les TMClients pendant la durée des déconnexions. Par exemple, un TMClient-mobile peut soumettre ses demandes au TMAgent, se déconnecter, et récupérer les résultats une fois la connexion rétablie. Il peut être utilisé pour économiser, par exemple, la vie de la batterie. Un TMClient-mobile peut soumettre ses demandes à son TMAgent, entrer en mode veille et attendre que le TMAgent l'avertisse lorsque la réponse est prête. Le TMAgent suit le déplacement des unités mobiles. Par exemple, lors d'un hand-off, le dernier TMAgent envoie l'information concernant l'unité mobile au nouveau TMAgent. Il stocke l'information de coordination transactionnelle des TMServeurs-mobiles (l'état des transactions en cours, le processus de validation, etc.). Il peut également coordonner le processus de validation globale à défaut d'un TMServeur-fixe participant à l'exécution d'une T_{AMT} (cf. section 6.1). Le TMAgent interagit avec un surveillant de l'environnement (service d'événements) qui perçoit l'état du réseau sans fil. Le TMAgent envoi l'information sur le réseau sans fil aux TMClients mobiles ou fixes.

Les trois tiers de l'architecture TransMobi peuvent être composés de différentes façons. La composition dépend principalement du caractère mobile des TMClients et des TMServeurs. Les configuration possibles dépendent des besoins en distribution des transactions mobiles. Celles-ci peuvent être réparties sur des unités fixes et mobiles selon des modèles d'exécution différents.

5.3. *Perception de l'environnement mobile*

Afin de rendre possible l'adaptabilité, il est nécessaire de « percevoir » l'état de l'environnement mobile à un moment donné. Pour cela, TransMobi utilise un *service d'événements*. Le tableau 2 introduit les types d'événements identifiés. La définition des événements est très flexible, d'autres événements peuvent être introduits.

Un TMServeur-mobile peut connaître l'état de ses ressources et un TMAgent l'état du réseau sans fil. Pour les TMClients, l'état de l'environnement est fourni par des TMAgents et par les TMServeurs-mobiles.

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 change pas beaucoup, l'événement

	Type d'événement	Information attachée	Mode Not.
RSF	*e-connexion*	Id de l'UM	*Push/Pull* synchrone/ asynchrone
	e-déconnexion	Id de l'UM	
	e-hand-off	Id de la nouvelle SB	
	e-bande-passante-change	Largeur de la bande passante	
	e-prix-communication-change	Prix de communication	
UM	*e-batterie-disponible*	Batterie disponible	
	e-cache-disponible	Cache disponible	
	e-mémoire-persistant-disponible	Mémoire persistante disponible	
	e-capacité-calcul-disponible	Capacité calcul disponible	
	e-localité-changes	Localité	

Tableau 2. *Type d'événements pour la perception de l'environnement mobile.*

e-bande-passante-change n'est pas nécessaire. Ainsi, l'implantation du service d'événements peut varier selon le contexte d'exécution de chaque application mobile.

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.[4] 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. Par exemple, dans la surveillance des ressources des unités mobiles, l'information concernant la consommation de la batterie (*e-batterie-disponible*) peut être obtenue avec PowerScope [FLI 99] – un outil pour surveiller la consommation d'énergie – qui utilise de l'échantillonnage statistique pour profiler l'utilisation d'énergie d'un système informatique (logiciel, matériel).

6. Techniques pour garantir les propriétés des T_{AMT}

TransMobi doit garantir les propriétés correspondant aux niveaux des alternatives d'exécution (AE_k) et des T_{AMT} (niveau racine). Plus particulièrement, l'atomicité sémantique, la cohérence sémantique, la sérialisabilité globale et la semi-atomicité.

La semi-atomicité est garantie par chaque TMClient, du fait que celui-ci doit assurer l'exécution d'une seule alternative. La cohérence sémantique est obtenue par l'utilisation de transactions de compensation. Dans la section 6.1, nous proposons un protocole pour la gestion de l'atomicité sémantique des alternatives d'exécution, nommé CO2PC. Ensuite, dans la section 6.2, nous proposons une adaptation d'une technique utilisée dans les systèmes multibases afin d'assurer la sérialisabilité globale des alternatives d'exécution. Enfin, la section 6.3 montre la gestion des déconnexions.

6.1. *Le protocole de validation CO2PC*

CO2PC est une Combinaison d'une approche optimiste avec le protocole de validation à deux phases 2PC [GRA 78]. CO2PC permet une validation optimiste (unilatérale/anticipée) ou non-optimiste des transactions composantes.

4. Un bon exemple est le système de fichiers *proc* qui offre des informations sur l'état courant du noyau Linux.

L'approche optimiste permet la validation locale des transactions composantes avant la validation de l'alternative d'exécution. En cas d'annulation de l'alternative, la validation anticipée rend nécessaire l'exécution de transactions de compensation. En effet, pour que les transactions composantes puissent valider unilatéralement, elles doivent être compensables.

L'approche non-optimiste est utilisée pour les transactions composantes non-compensables ou pour celles qui ne permettent pas la validation anticipée. Le choix de l'utilisation du protocole 2PC est motivé par son interface *préparation*. En effet, l'état *préparation* rend possible la rétention des ressources des transactions composantes non-compensables jusqu'à la validation ou l'annulation de l'alternative correspondante. Dans 2PC les composantes qui terminent leurs opérations entrent dans l'état de *préparation*. Dans cet état, une transaction ne peut être ni annulée ni validée de manière autonome par le SGBD sous-jacent. Seule une opération de validation/annulation émise par un gestionnaire global peut faire sortir de cet état les transactions composantes.

Dans le protocole CO2PC, il existe un coordinateur qui assure la validation globale d'un ensemble de participants. Chaque participant a un *proxy* associé.

Un **participant** est représenté par le *proxy* qui intéragit avec le SGBD sous-jacent exécutant une transaction composante. Le *proxy* est un intermédiaire entre l'application et le SGBD sous-jacent auquel il cache la validation globale. Il rend possible la participation, dans une validation globale, du SGBD même s'il n'est pas conçu pour le faire. Il permet également de réduire le nombre de messages nécessaires pour l'exécution de CO2PC. Il est installé sur le même site que son SGBD sous-jacent et il peut valider les transactions composantes de manière optimiste ou non-optimiste.

Le rôle de **coordinateur** est joué par l'un des participants fixes. Un participant mobile ne doit pas être coordinateur car il est sujet à des déconnexions fréquentes. Le coordinateur est désigné au démarrage de l'alternative d'exécution. Chaque validation globale peut avoir un coordinateur différent.

D'une façon générale, les transactions composantes (t_{ki}) d'une alternative d'exécution (AE_k) sont réparties. Chacune des t_{ki} est exécutée par un SGBD 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 *validation*, la décision prise est la validation globale, dans le cas contraire, l'alternative est annulée globalement.

La figure 5 présente le diagramme de séquences de CO2PC. Le cas de configuration de la figure comporte deux participants, un mobile et un fixe. Le participant fixe exécute une validation non-optimiste et le mobile une validation optimiste. Le rôle du coordinateur est joué par le participant fixe.

Validation optimiste. Un participant faisant une validation optimiste (appelé *participant-opt*), demande l'exécution de la t_{ki} au SGBD sous-jacent correspondant. Une fois l'exécution terminée, t_{ki} est validée ou annulée de manière unilatérale. Ensuite,

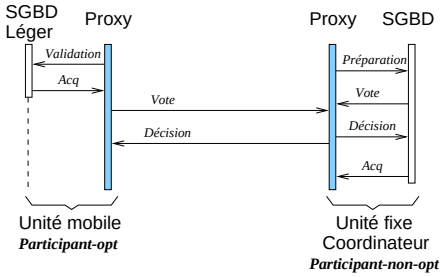

Figure 5. *Le protocole de validation CO2PC.*

Site	Opération à journaliser
Participants optimistes	*Demande de Validation*
	Acquitement
Participants non-optimistes	*Début2PC*
	Préparation
Tous les participants	*Vote*
	Décision
Coordinateur	*DébutCO2PC*
	Chaque *Vote*
	Décision

Figure 6. *Journalisation pendant l'exécution du protocole CO2PC.*

le participant-opt envoie son *vote* au coordinateur. Celui-ci prend la décision globale et la propage. Si la décision est de valider, les participants-opt ont fini. Si la décision est d'annuler, les participants-opt doivent démarrer une transaction de compensation.

Validation non-optimiste. Un participant faisant une validation non-optimiste (appelé *participant-non-opt*) demande l'exécution de la t_{ki} au SGBD sous-jacent correspondant. Une fois l'exécution terminée, le participant-non-opt exécute le protocole 2PC localement.[5] Afin d'exécuter localement le protocole 2PC, le *proxy* joue le rôle de coordinateur du SGBD sous-jacent qui joue le rôle de participant. Il initialise le protocole 2PC avec son SGBD sous-jacent (message de *préparation*). Le SGBD lui envoie le *vote* qui est redirigé au coordinateur (global). Si le vote est une *annulation*, le participant annule t_{ki} de façon unilatérale, sinon, t_{ki} entre dans l'état de *préparation*. Le coordinateur prend une décision globale et la propage aux participants. Dès que la décision du coordinateur arrive au participant, la *validation/annulation* correspondante est demandée au SGBD. Il faut remarquer que la partie 2PC du protocole CO2PC se fait en local et n'implique pas d'échange de messages entre sites différents.

Situations de blocage. Les participants-opt ne peuvent pas être bloqués grâce à l'approche optimiste, mais ils peuvent entraîner le blocage des participants-non-opt. Un participant-non-opt peut être bloqué s'il vote pour une validation et : (1) l'un des participants n'envoi pas son vote (à cause d'une panne ou d'une déconnexion indéfinie) ou se déconnecte indéfiniment ; (2) le coordinateur tombe en panne. Afin de limiter la durée des blocages, des délais (*timeouts*) sont utilisés. Chaque AE_k, ainsi que chacune de ses t_{ki}, a un délai assigné. Le délai de l'AE_k doit être plus grand que celui des t_{ki}. Chaque participant doit envoyer le vote avant l'expiration du délai de la t_{ki} qu'il exécute. Les participants peuvent renégocier les délais, par exemple lorsqu'ils vont se déconnecter et qu'ils connaissent le moment de leur prochaine reconnexion.

Annulation d'une AE_k. L'annulation d'une AE_k peut avoir plusieurs raisons : une t_{ki} vote pour une annulation globale ; un cycle dans l'ordre d'exécution se produit

5. Une transaction non-compensable ne peut pas être exécutée sur un SGBD sous-jacent qui ne fournit pas l'interface 2PC.

et un problème de sérialisabilité survient (cf. section 6.2) ; le délai de l'AE_k expire sans que le coordinateur ait reçu tous les votes.

Reprise après panne

Journalisation Pendant l'exécution de CO2PC, le déroulement des différentes étapes est enregistré pour la reprise après panne. Ceci se fait aussi bien dans le journal du coordinateur que dans celui des participants (cf. tableau 6). Puisque les défaillances et les déconnexions peuvent arriver à n'importe quel moment, l'information journalisée doit être écrite sur un support persistant avant d'envoyer les messages.

Compensation. Lorsqu'une alternative d'exécution est annulée, les participants gérant des transactions non-compensables demandent l'annulation de t_{ki} et ceux gérant des transactions compensables demandent l'exécution de transactions de compensation si la t_{ki} correspondante a été validée.Une fois que les transactions de compensation ont démarrée, elles doivent être validées. Cette caractéristique est appelé *persistance de compensation* [GAR 87]. Elle est assurée par la resoumission des transactions de compensation jusqu'à leur validation. Les transactions de compensation des transactions composantes qui ont validé séquentiellement sont exécutées dans l'ordre inverse de validation.

Durabilité. Afin de respecter l'autonomie et de limiter le besoin de communication, CO2PC délègue la responsabilité d'assurer la durabilité des transactions aux SGBD sous-jacents. A la différence d'autres protocoles, CO2PC ne force pas l'écriture des journaux des modifications sur des supports stables (unités fixes) avant la validation des transactions composantes. En effet, nous avons fait un compromis pour concevoir un protocole convenable pour l'environnement mobile. CO2PC offre aux SGBD sous-jacents la possibilité de décider de la manière d'assurer la durabilité des modifications faites par des transactions validées. En effet, un SGBD localisé sur une unité mobile peut décider de faire le transfert des modifications sur un support stable : après la validation de chacune des transactions, après la validation d'un nombre défini de transactions, lorsqu'il bénéficie d'une bonne connexion, etc.

Les avantages de CO2PC

CO2PC est un protocole flexible qui offre plusieurs avantages, il permet :

- la déconnexion des participants ;
- la validation unilatérale des transactions composantes (validation optimiste) ;
- la validation coordonnée avec l'alternative d'exécution (validation non-optimiste) ;
- de limiter le nombre de messages nécessaires pour son exécution (2 par participant) : un message de *vote* et un autre de *décision* ;
- de délèguer aux SGBD la décision d'assurer la durabilité de la manière la plus convenable.

CO2PC est conçu pour relâcher l'atomicité. Cependant, si toutes les transactions composantes d'une alternative d'exécution font une validation non-optimiste, l'ato-

micité n'est pas relâchée. Dans [BOB 04] nous proposons une analyse qualitative et quantitative des protocoles CO2PC, 2PC, UCM et TCOT.

CO2PC et TransMobi

Dans l'implantation de CO2PC dans TransMobi, les TMServeurs assurent le rôle des *proxies* et la communication entre les unités mobiles et les fixes se fait par l'intermédiaire d'un TMAgent. Comme le rôle du coordinateur doit être sur une unité fixe, à défaut de TMServeur participant dans une alternative d'exécution, un TMAgent peut jouer ce rôle. La figure 7 montre qu'aussi bien les unités mobiles que fixes peuvent valider d'une manière optimiste ou non-optimiste.

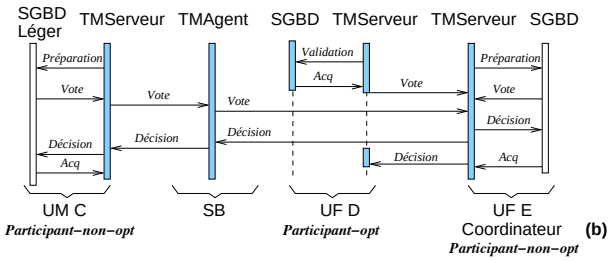

Figure 7. *Le protocole CO2PC et TransMobi.*

En ce qui concerne la journalisation, Les TMServeurs-mobiles doivent journaliser l'information liée au CO2PC dans leur journal mais aussi dans celui du TMAgent correspondant. Ceci est dû au fait que le stockage physique des unités mobiles n'est pas considéré comme stable car elles peuvent subir des pertes, dommages et déconnexions. Si l'unité mobile se déconnecte, la journalisation sur le TMAgent va attendre une prochaine reconnexion.

6.2. *Sérialisabilité globale*

Pour fournir la sérialisabilité globale, deux conditions sont nécessaires : (1) les SGBD sous-jacents doivent produire des ordonnancements sérialisables et recouvrables et (2) chaque SGBD maintient l'ordre relatif des sous-transactions perçu au niveau global.

Dans notre travail, nous supposons que les SGBD sous-jacents préservent les propriétés ACID des transactions composantes, et de ce fait la première condition est assurée. Pour garantir la deuxième condition, la méthode utilisée doit assurer un ordre d'exécution globalement sérialisable. Nous considérons en plus, qu'afin d'être applicable aux systèmes mobiles, la méthode utilisée doit préserver l'autonomie des SGBD sous-jacents et[6] permettre l'hétérogénéité des mécanismes de contrôle de concurrence

6. A cause des caractéristiques de l'environnement mobile, nous considérons qu'il est important de préserver l'autonomie des sites et de permettre leur hétérogénéité.

utilisés dans les SGBD sous-jacents. Il faut également utiliser des approches optimistes plutôt que pessimistes afin d'éviter le blocage du système.

Les techniques utilisées dans les systèmes multibases peuvent être adaptées à l'environnement mobile. Parmi elles, nous proposons d'adapter la méthode OTM (*Optimistic Ticket Method*) [GEO 94] qui est traditionnellement utilisée avec le protocole de validation 2PC. Ici, nous proposons une combinaison de OTM avec CO2PC. L'idée principale dans OTM est de convertir les conflits indirects[7] en conflits directs. Sur chaque SGBD, il est nécessaire de stocker une donnée spéciale, appelée *ticket*. Chaque sous-transaction globale lit et incrémente le ticket de façon atomique dans le site où elle s'exécute – il existe un ticket par site. Le ticket doit être géré comme n'importe quelle autre donnée. L'accès au ticket entraîne un ordre d'exécution local qui peut être perçu au niveau global.

Au niveau global, un graphe de sérialisabilité globale (GSG) est maintenu. Les nœuds du graphe correspondent aux AE_k en cours de validation ainsi qu'à celles récemment validées. Les arcs représentent l'ordre d'accès aux tickets. Lorsqu'un cycle se produit l'une des AE_k doit être annulée. Dans TransMobi, le GSG est maintenu par un gestionnaire global de sérialisabilité (GG) localisé sur un TMServeur-fixe. Pour l'instant, nous considérons que cette gestion est centralisée sur un seul TMServeur. La construction du GSG ne génère pas de nouveaux messages sur le réseau sans fil, car les participants attachent au message de vote de CO2PC, la nouvelle valeur du ticket. En effet, les nouveaux messages sont transmis sur le réseau fixe entre le coordinateur de CO2PC et le gestionnaire du graphe. Il y a un message par participant (envoie de la valeur du ticket) et un pour la notification sérialisable/cycle.

L'adaptation de la méthode OTM offre plusieurs avantages : c'est une méthode simple ; elle génère un ordre d'exécution sérialisable des alternatives d'exécution et par conséquent des transactions T_{AMT} ; elle préserve l'autonomie des SGBD sous-jacents ; elle autorise l'hétérogénéité des protocoles de contrôle de concurrence ; elle ne génère pas de nouveaux messages sur le réseau sans fil ; elle permet la déconnexion des sites mobiles (cf. section 6.3) ; grâce à l'utilisation de CO2PC les transactions composantes ont la possibilité de faire une validation unilatérale et les ressources peuvent être libérées de manière anticipée.

6.3. *Gestion des déconnexions*

Une propriété importante d'un système mobile est de considérer la déconnexion comme un état normal. TransMobi gère la déconnexion au niveau transactionnel. Car, pendant l'exécution d'une T_{AMT}, les unités mobiles peuvent subir des déconnexions. Il existe deux types de déconnexions : désirable et indésirable. La première est prévue et souhaitée par l'utilisateur. La deuxième est une conséquence d'une variation d'état de l'environnement mobile. Par exemple, l'unité mobile est sortie de l'espace géogra-

7. Dans les systèmes multibases, l'exécution de transactions locales, non visibles au niveau global, peut entraîner des conflits indirects.

phique couvert par le réseau sans fil ou la batterie de l'unité mobile est épuisée. Les protocoles de TransMobi tolèrent les deux types de déconnexions.

Dans CO2PC, les participants peuvent se déconnecter avant et après avoir voté. Le délai affecté aux transactions composantes permet aux unités mobiles de se déconnecter avant d'avoir voté, pourvu que le *vote* soit envoyé au coordinateur avant l'expiration du délai. Une unité mobile peut se déconnecter temporairement après que le *vote* soit envoyé. Dans ce cas, la décision du coordinateur est stockée dans le TMAgent et elle est redirigée vers l'unité mobile lors de la prochaine reconnexion. Les participants optimistes, ou non-optimistes, peuvent se déconnecter (de manière prévue ou imprévue). La différence est que pour les non-optimistes, les ressources restent bloquées jusqu'à la prochaine reconnexion. Ceci est dû au fait que la partie 2PC de CO2PC se termine lorsque la décision globale arrive au participant.

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. En effet, l'incrémentation du ticket est une opération complètement indépendante de l'état de connexion des UM. L'unique contrainte est l'envoi de la valeur du ticket au coordinateur de CO2PC. Néanmoins, la durée de la déconnexion a une conséquence importante. Le GSG est construit dans l'ordre dans lequel les T_{AMT} valident. Ainsi, s'il existe un conflit, l'alternative qui essaie d'être sérialisée sera annulée. Cela veut dire que la probabilité d'annulation/validation est directement liée au temps qu'une transaction prend pour être sérialisée.

Nous avons développé un prototype de l'intergiciel TransMobi qui implante quelques uns des composants décrits dans cette section. Le réseau sans fil utilisé est un *Wireless Local Area Network* (WLAN) IEEE 802.11b. Le développement a été fait en PersonalJava [Per 00].[8] Comme système gestionnaire de bases de données nous avons utilisée une version de test de PointBase [Poi 02]. Les unités mobiles utilisées sont un ordinateur portable Compaq Armada 1700 et un PDA de type IPAQ H3850 de Compaq.

7. Autres Travaux

Le modèle AMT a été inspiré par DOM [BUC 92] et Flex [ELM 90] où l'idée générale est de définir des transactions *équivalentes* à exécuter en cas de disfonctionnement. Le modèle de transaction DOM permet des transactions *fermées* et *ouvertes, imbriquées*. Des transactions de compensation ainsi que des transactions contingentes peuvent être spécifiées pour qu'elles soient exécutées en cas de défaillance de la transaction de départ. Dans le modèle Flex, les transactions contingentes sont définies comme des transactions *fonctionnellement équivalentes*. Quand l'exécution d'une transaction dépend de l'annulation d'une autre, un ordre d'exécution est défini. Contrairement à AMT, les transactions de DOM et Flex ne sont pas définies pour des environnements mobiles et la notion de prise en compte du contexte n'est pas considérée.

8. PersonalJava est maintenant remplacé par la famille Java 2 Micro Edition (J2ME).

Le panorama des travaux sur les transactions mobiles est large. Une analyse détaillée a été faite dans [SER 04b]. En général, l'adaptabilité est très limitée, par exemple la plupart des systèmes étudiés limité celle-ci à la prise en compte des déconnexions. C'est le cas de Clustering [PIT 99], Two-tier replication [GRA 96], HiCoMo [LEE 02], IOT [SAT 90], Pro-motion [WAL 99], Prewrite[MAD 01], MDSTPM[YEO 94] et Pre-serialization [DIR 00]. Seul Moflex [KU 00] s'intéresse à l'adaptabilité en cours d'exécution si un hand-off se produit. Au contraire, notre proposition AMT permet de s'adapter à différents types d'environnements.

Concernant les types d'exécution, plusieurs travaux considèrent les transactions lancées par une UM et exécutées en totalité sur une UF. Par exemple, KT [DUN 97], MDSTPM, Moflex and Pre-serialization. Les autres propositions se concentrent sur l'exécution de la transaction au niveau de l'UM : HiCoMo, IOT et Pro-motion. Seuls clustering et Two-tier replication considèrent deux types d'exécution, (1) sur une UM (lors des déconnexions) et (2) répartie sur unités mobiles et fixes (pendant les connexions). Aucune des propositions étudiées ne traite des exécutions réparties entre plusieurs UM ou entre des unités fixes et mobiles. TCOT [KUM 02] et UCM [BOB 00] considèrent ces types d'exécutions même si ces travaux se placent plutôt au niveau du processus de validation de la transaction. Avec le modèle AMT, il est possible de définir des transactions dans les cinq types d'exécution mentionnés plus haut. AMT s'adapte ainsi facilement à la grande variété des environnements mobiles.

Bien que les aspects de duplication des données ne soient pas nécessairement liés au transactionnel, ils sont au cœur de plusieurs travaux sur les transactions mobiles. Des propositions comme Clustering, Two-tier replication et IOT considèrent que les UM contiennent des données dupliquées. Pendant les déconnexions, ces copies sont modifiées par des transactions de secondes classes. Au moment de la reconnexion, différents processus de conciliation sont proposés (re-exécution par des transactions de première classe, synchronisation des copies, etc.). Dans notre travail nous n'avons pas traité ces aspects, en séparant duplication et transaction. Nous considérons un système multibases où chaque site (fixe ou mobile) est indépendant des autres. Une approche similaire est prise dans KT [DUN 97], Pre-serialization, MDSTPM et Moflex. Cependant, dans ces travaux, l'environnement multibase n'est compose que de SGBD s'exécutant sur des unités fixes. Les unités mobiles ne font que lancer des transactions.

Finalement, la principale différence du modèle AMT avec les autres travaux existants est la prise en compte de la description de l'environnement d'exécution, lors de la définition de la transaction.

8. Conclusions et Travaux Futurs

Cet article a traité les transactions mobiles en apportant plusieurs contributions : (1) Nous avons proposé le modèle AMT qui s'inspire des travaux existants sur les modèles de transactions étendues. Il a pour originalité la prise en compte des caractéristiques de l'environnement selon plusieurs dimensions comme le débit de transmission, la connectivité, les ressources de l'unité mobile, etc. Notre modèle concerne des transactions qui impliquent plusieurs SGBD hétérogènes qui tournent sur des unités fixes

ou mobiles. Une spécification formelle de notre modèle de façon à pouvoir le comparer avec d'autres modèles de transactions existants est présentée dans [SER 04a] et [SER 04c]. (2) Nous avons effectué une étude analytique qui peut être vue comme un outil semi-automatique pour optimiser la conception et l'exécution des transactions mobiles. Cette étude est réalisée par une analyse a priori du coût d'exécution de plusieurs alternatives d'exécution. (3) Nous proposons enfin l'intergiciel TransMobi comme support efficace du modèle AMT. La propriété d'atomicité sémantique est notamment assurée par le protocole de validation CO2PC que nous proposons. Ce protocole original autorise les validations optimistes et non-optimistes en combinant une approche optimiste avec un protocole de validation à deux phases.

Les perspectives que nous avons sont les suivantes : (1) L'implantation d'un utilitaire de développement d'application de façon à faciliter la conception des transactions AMT. Les paramètres de l'environnement permettent de mieux prendre en compte tel ou tel type d'environnement et aussi de tenir compte des habitudes de l'utilisateur. Une fois l'environnement défini, le calcul analytique comme celui présenté ici permet d'optimiser dynamiquement T_{AMT} par rapport à la qualité de service désirée. (2) L'analyse de la pertinence d'adapter les transactions non seulement au début (comme c'est le cas actuellement dans notre proposition) mais aussi pendant l'exécution ; il importe de considérer ici les travaux déjà effectués sur les transactions adaptables. (3) Appliquer le modèle AMT à des environnements pair à pair ou à des réseaux ad hoc qui comme le environnements mobiles ont des caractéristiques très variables.

9. Bibliographie

[BER 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 and 4, 2003.

[BER 04] 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., « Mobile Databases : a Selection of Open Issues and Research Directions) », *ACM SIGMOD Record*, vol. 33, n° 2, 2004.

[BOB 00] BOBINEAU C., PUCHERAL P., ABDALLAH M., « A Unilateral Commit Protocol for Mobile and Disconnected Computing », *Int. Conf. Parallel and Distributed Computing Systems (PDCS)*, Las Vegas, USA, Août 2000.

[BOB 04] BOBINEAU C., LABBÉ C., RONCANCIO C., , SERRANO-ALVARADO P., « Performances de Protocoles Transactionnels en Environnement Mobile », *Journées de Bases de Données Avancées (BDA)*, Montpellier, France, Octobre 2004.

[BUC 92] BUCHMANN A., ÖZSU M. T., 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.

[DIR 00] DIRCKZE R. A., GRUENWALD L., « A Pre-Serialization Transaction Management Technique for Mobile Multidatabases », *Mobile Networks and Applications (MONET)*, vol. 5, n° 4, 2000.

[DUN 97] DUNHAM M. H., HELAL A., BALAKRISHNAN S., « A Mobile Transaction Model that Captures Both the Data and the Movement Behavior », *ACM/Baltzer Journal on special topics in mobile networks and applications*, vol. 2, n° 2, 1997.

[ELM 90] ELMAGARMID A. K., LEU Y., RUSINKIEWICS M., « A Multidatabase Transaction Model for INTERBASE », *Int. Conf. on Very Large Databases (VLDB)*, Brisbane, Australia, Août 1990.

[Fas 02] FASTOBJECTS BY POET, « FastObjects j2 http ://www.fastobjects.com/ », 2002.

[FLI 99] FLINN J., SATYANARAYANAN M., « PowerScope : A Tool for Profiling the Energy Usage of Mobile Applications », *IEEE Workshop on Mobile Computing Systems and Applications (WMCSA)*, New Orleans, USA, Février 1999.

[GAR 83] GARCIA-MOLINA H., « Using Semantic Knowledge for Transaction Processing in a Distributed Database », *ACM Transactions on Database Systems (TODS)*, vol. 8, n° 2, 1983.

[GAR 87] GARCIA-MOLINA H., SALEM K., « Sagas », *ACM SIGMOD Conference*, San Francisco, USA, Mai 1987.

[GEO 94] GEORGAKOPOULOS D., RUSINKIEWICZ M., SHETH A., « Using Tickets to Enforce the Serializability of Multidatabase Transactions », *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, vol. 6, n° 1, 1994.

[GRA 78] GRAY J., « Notes on Database Operating Systems », *Advanced Course : Operating Systems*, n° 60 LNCS, 1978.

[GRA 96] GRAY J., HELLAND P., P.O'NEIL, SHASHA D., « The Dangers of Replication and a Solution », *ACM SIGMOD Conference*, Montreal, Canada, Juin 1996.

[KU 00] KU K., KIM Y., « Moflex Transaction Model for Mobile Heterogeneous Multidatabase Systems », *IEEE Workshop on Research Issues in Data Engineering*, San Diego, USA, Février 2000.

[KUM 02] KUMAR V., PRABHU N., DUNHAM M. H., SEYDIM A. Y., « TCOT- A Timeout-Based Mobile Transaction Commitment Protocol », *IEEE Transactions on Computers*, vol. 51, n° 10, 2002.

[LEE 02] LEE M., HELAL S., « HiCoMo : High Commit Mobile Transactions », *Kluwer Academic Publishers Distributed and Parallel Databases (DAPD)*, vol. 11, n° 1, 2002.

[MAD 01] 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.

[Ora 02] ORACLE CORPORATION, « Oracle9i Lite : The Internet Platform For Mobile Computing http ://otn.oracle.com/products/lite/ », 2002.

[Per 00] PERSONALJAVA APPLICATION ENVIRONMENT SPECIFICATION V1.2 (FINAL), « http ://java.sun.com/products/personaljava/ », Novembre 2000.

[PIT 99] PITOURA E., BHARGAVA B., « Data Consistency in Intermittently Connected Distributed Systems », *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, vol. 11, n° 6, 1999.

[Poi 02] POINTBASE JAVA DATABASES, « PointBase http ://www.pointbase.com », 2002.

[RON 04] RONCANCIO C. L., « Intergiciels et services pour la gestion de données réparties », rapport n° Habilitation à Diriger des Recherches, Juin 2004, Institut National Polytechnique de Grenoble, Grenoble, France.

[SAT 90] SATYNARAYANAN M., KISTLER J., KUMAR P., OKASAKI E., SIEGEL H., STEERE C., « Coda : A Highly Available File System for Distributed Workstation Environment », *IEEE Transactions on Computers*, vol. 39, n° 4, 1990.

[SER 04a] SERRANO-ALVARADO P., « Transactions Adaptables pour les Environnements Mobiles », PhD thesis, Université Joseph Fourier, Grenoble, France, 2004.

[SER 04b] SERRANO-ALVARADO P., RONCANCIO C., ADIBA M., « A Survey of Mobile Transactions », *Kluwer Academic Publishers Distributed and Parallel Databases (DAPD)*, vol. 16, n° 2, 2004.

[SER 04c] SERRANO-ALVARADO P., RONCANCIO C., ADIBA M., LABBÉ C., « An Adaptable Mobile Transaction Model for Mobile Environments », *Int. Journal of Computer Systems Science and Engineering (IJCSSE), Special Issue on Mobile Databases*, vol. 20, n° 2, 2004.

[WAL 99] WALBORN G. D., CHRYSANTHIS P. K., « Transaction Processing in PROMOTION », *ACM Symp. on Applied Computing*, San Antonio, USA, Février 1999.

[YEO 94] YEO L. H., ZASLAVSKY A., « Submission of Transactions from Mobile Workstations in a Cooperative Multidatabase Processing Environment », *Int. Conf. on Distributed Computing Systems (ICDCS)*, Poznan, Poland, Juin 1994.