Un middleware pour les applications transactionnelles hospitalières mobiles

Françoise André, Erwan Saint Pol

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Un middleware pour les applications transactionnelles hospitalières mobiles

Françoise André - Erwan Saint Pol

N° 3863

Janvier 2000

THEME 1

ISSN 0249-6399 ISRN INRIA/RR--3863--FR

Un middleware pour les applications transactionnelles hospitalières mobiles

Françoise André et Erwan Saint Pol

Thème 1: Réseaux et systèmes

Projet Solidor

Rapport de recherche n˚3863

Janvier 2000

16 pages

Résumé : L'arrivée des technologies nomades dans l'hôpital ouvre la voie vers de nouvelles applications. Dans ce rapport nous nous intéressons aux problèmes soulevés par l'utilisation d'un réseau local sans fil dans le cadre d'une application transactionnelle mobile. Nous prenons comme exemple le cas d'une application de prescription médicale hospitalière. L'objectif est de permettre au personnel soignant de saisir sa prescription directement au chevet de son patient au moyen d'un ordinateur portable connecté à un réseau local sans fil. La qualité de transmission d'un tel réseau est très variable. L'application doit s'adapter à ces variations de façon à offrir une qualité de service constante à l'utilisateur. Au lieu de modifier l'application nous proposons d'élaborer un middleware qui réalisera les tâches relatives à l'adaptation. Notre middleware adapte l'exécution des transactions en fonctions de la qualité de communication du réseau sans fil : lorsque la qualité est satisfaisante les transactions sont exécutées sur le serveur, lorsqu'elle est faible elles sont exécutées sur le poste mobile.

Mots-clé : middleware, systèmes transactionnels, informatique médicale, réseaux locaux sans fil, nomadisme.

(Abstract: pto)

Unité de recherche INRIA Rennes
IRISA, Campus universitaire de Beaulieu, 35042 Rennes Cedex (France)
Téléphone : 02 99 84 71 00 - international : +33 2 99 84 71 00
Télécopie : 02 99 84 71 71 - international : +33 2 99 84 71 71

A Middleware for Mobile Transactional Hospital Applications

Abstract: Today the arrival of mobile technologies in the hospital opens the way to new applications. In this report we focus on the problems raised by the use of a wireless local area network for a mobile transactional application. We consider the example of a medical prescription application. We want to enable the caregiver to register his/her prescription at the bed of his/her patient by using a portable computer connected by a wireless local area network. The quality of transmission of such a medium is very variable. The application must adapt to these variations in order to propose a constant quality of service to the caregiver. Instead of modifying the core of the application we propose to built a middleware which handles these variations for the application. Our middleware adapts the execution of transactions depending on the network quality of communication : when quality is high, transactions are executed on the server ; when quality is low, transactions are executed on the mobile host.

Key-words: middleware, transactional systems, health care, wireless LAN, mobility.

Les ordinateurs portables sont de plus en plus utilisés et tout porte à croire qu'ils occuperont une place importante dans le monde de demain. On en trouve de toutes tailles, adaptés à chaque situation, de la station de travail portative aux PC de poche. Les premiers ont des caractéristiques techniques proches des stations de travail : un écran confortable, un processeur haut de gamme, une quantité de stockage (RAM, disque dur) égale à celle des stations de travail fixes mais ils sont encombrants (> 2Kg) et souffrent d'une autonomie limité (< 4H). Les derniers ont des ressources très limitées : ils sont dotés d'un petit écran monochrome (17 x 6,7 cm), d'un processeur peu performant en comparaison des processeurs actuels des stations fixes, une capacité mémoire faible (mémoire flash de 8 à 16 M qui remplace la RAM et le disque dur) mais ils sont peu encombrants (< 500g) et bénéficient d'une grande autonomie (de 3 à 55 H). Les ordinateurs portables peuvent être reliés aussi bien à un réseau filaire ou à un réseau sans fil, ce qui permet à l'utilisateur de se déplacer tout en restant connecté à son réseau. Les réseaux sans fil se répartissent en deux grandes catégories : les réseaux à grande échelle (de type GSM [1]) et les réseaux à petite échelle (réseaux locaux de type Wavelan [1]). Les premiers possèdent une qualité de transmission très faible : une bande passante de 9600 bits/s, et sont sujets à de fréquentes déconnexions ; les derniers possèdent une qualité de transmission largement supérieure : une bande passante de 2Mbits/s mais très variable en fonction de l'architecture du réseau et de l'environnement ; les déconnexions sont rares et de courte durée.

La conception d'une application prenant en compte chacun de ces aspects se révèle être une tache difficile. Notre but est de faciliter la conception de telles applications en fournissant au développeur un certain nombre d'outils logiciels réalisant une tâche particulière. Notre objectif au sein du projet Solidor est de fournir un environnement logiciel nommé MolèNE [2][3] prenant en compte chacun des aspects de la mobilité. MolèNE est un environnement modulaire qui fournit au programmeur un ensemble de stratégies d'adaptation ainsi que la possibilité de créer les siennes. La construction de ce système nous amène à étudier des cas particuliers d'application. En ce sens nous avons étudié le cas du commerce électronique [4][5] et nous nous intéressons dans ce rapport à l'adaptation d'une application de prescription médicale hospitalière à une exécution sur un ordinateur portable (poste mobile) connecté par un réseau local sans fil au réseau de l'hôpital. Il s'agit d'une application transactionnelle client/serveur classique. Le poste mobile utilisé dispose de ressources confortables proches d'une station de travail portative. Le réseau local sans fil offre une bande passante de 2 Mbits/s. La qualité de transmission de ce réseau peut subir de grandes variations . Ces variations ont une influence directe sur les temps de réponse de l'application. Ces derniers peuvent devenir insupportables pour l'utilisateur. Notre solution est donc de fournir un environnement logiciel appelé Charcot permettant d'adapter l'exécution des transactions en fonction de la qualité de transmission du réseau sans fil. Lorsque la qualité est bonne les transactions sont exécutées sur le serveur, lorsqu'elle est faible elles sont exécutées sur le poste mobile. La réalisation de cette solution est effectuée par l'assemblage de différents composants que nous avons définis.

Nous décrivons dans la première section de ce rapport le processus de saisie d'une prescription tel qu'il se déroule à l'heure actuelle. Dans la seconde section nous présentons notre système. La troisième section est consacrée à l'étude des systèmes apparentés et nous concluons dans la quatrième section.

1 La saisie des prescriptions dans un service hospitalier

Actuellement dans la plupart des services hospitaliers chaque médecin au cours de sa visite note sur un cahier l'ordonnance qu'il prescrit aux différents patients qu'il examine. A la fin de sa visite, il remet son cahier à un/une secrétaire. Celui/celle-ci saisit sur un terminal l'ordonnance des patients au moyen d'un logiciel dédié. Ce logiciel effectue un certain nombre de contrôles sur la prescription du médecin. Il vérifie par exemple qu'il n'y a pas d'incompatibilité entre les médicaments à administrer au patient. Lorsque le logiciel refuse une prescription, le médecin doit intervenir pour la modifier. Lorsque la prescription est enfin validée elle est stockée dans le système d'information hospitalier où elle pourra ensuite être utilisée par les différents services de l'hôpital. Le service de gestion de stock des médicaments par exemple pourra l'utiliser pour savoir quels médicaments commander et en quelle quantité. Cette façon de procéder présente un certain nombre d'inconvénients. D'une part le/la secrétaire peut faire des erreurs lors de la recopie de l'ordonnance du médecin. D'autre part la prescription peut être rejetée au moment de la validation, ce qui oblige le médecin à la modifier après sa visite. Tous ces inconvénients retardent le processus de livraison du médicament au patient.

Le système nommé Charcot que nous développons doit résoudre ces problèmes. L'objectif est de permettre à un médecin de saisir directement l'ordonnance d'un patient au chevet de celui-ci. Pour cela on lui fournit un ordinateur portable connecté à un réseau local sans fil, ainsi qu'une application de prescription médicale adaptée. Les problèmes précédents n'ont plus lieu d'exister : il n'y a plus de risque d'erreurs dues à la recopie puisque le médecin saisit lui même la prescription, il n'y a plus à intervenir sur les ordonnances après la visite puisque les contrôles de validité de la prescription d'un patient sont effectués instantanément. De plus cela permet au médecin d'accéder à toutes les informations qui peuvent l'aider durant sa visite. Cette solution présente un gain de temps pour le médecin et de meilleures prestations pour le patient.

La nouvelle plate-forme de prescription médicale est composée, comme le montre la figure 1, d'une application de prescription médicale (APM) exécutée sur un poste mobile et d'un serveur de base de données du système d'information hospitalier (SIH) situé sur le réseau fixe de l'hôpital. Il s'agit d'une architecture client-serveur classique : l'APM transmet des requêtes SQL au Système de Gestion de Base de Données (SGBD) du SIH qui lui envoie les réponses. L'originalité provient de l'utilisation d'un réseau local sans fil à émission d'ondes radios.

Figure 1 : architecture de la plate-forme de prescription médicale

Cette plate-forme est difficile à réaliser en raison de l'utilisation d'un réseau local sans fil car la qualité de transmission d'un tel réseau est plus faible que celle d'un réseau local standard (la bande passante est environ dix fois inférieure à celle d'un réseau ETHERNET [6]), de plus elle est très variable. La qualité pourra être très bonne pendant une certaine période et très mauvaise pendant d'autres périodes en fonction de l'infrastructure du réseau sans fil (proximité des points d'accès) et de l'environnement (matériaux empêchant la propagation des ondes radios). Ces problèmes technologiques interviennent directement sur la qualité de service offerte à l'utilisateur final : une baisse de la qualité de transmission rendra pénible la saisie de la prescription. De plus dans un système transactionnel classique les utilisateurs mobiles ne sont pas les seuls à subir les conséquences de l'utilisation d'un réseau sans fil. En effet, les utilisateurs fixes seront bloqués par les verrous posés par les transactions initiées à partir des sites mobiles (transactions mobiles). Plus la qualité de transmission du réseau sans fil se dégrade, plus les transactions mobiles sont longues et donc plus la qualité de service offerte aux utilisateurs se dégrade.

2 Charcot

Notre solution est de proposer un logiciel (middleware) appelé Charcot situé entre le poste mobile et le SIH qui permette à l'application de s'affranchir de ces difficultés. Nous émettons les hypothèses suivantes : la liaison sans fil est disponible la plupart du temps, les déconnexions sont donc rares et courtes ; il y a peu de chances que plusieurs personnes modifient les données d'un même patient au même moment, il y a donc peu de conflits entre les transactions ; le poste mobile dispose de ressources importantes et possède un SGBD.

Charcot permet de s'adapter à la qualité de transmission du réseau sans fil : lorsqu'elle est bonne les transactions sont exécutées sur le serveur, lorsqu'elle est mauvaise les transactions sont exécutées sur le poste mobile. Exécuter les transactions sur le serveur présente l'avantage de pouvoir travailler sur des données à jour et de rendre les prescriptions immédiatement disponibles pour les différents services de l'hôpital. Exécuter les transactions sur le poste mobile présente l'avantage de rendre la saisie de la prescription plus rapide, cela permet également de ne pas gêner le travail des utilisateurs fixes. Cependant cela présente l'inconvénient de ne pas tou-

jours travailler sur des données à jour ainsi que de ne pas rendre les prescriptions aussitôt disponibles pour les services de l'hôpital. Pour ces raisons il ne faut exécuter les transactions sur le poste mobile que lorsque cela est rendu vraiment nécessaire par la mauvaise qualité de la transmission.

La Figure 2 montre l'architecture du système : Charcot est divisé en deux parties, une partie situé sur le poste mobile et une partie situé sur le réseau fixe de l'hôpital. L'application envoie des transactions à Charcot sur le poste mobile, celui-ci décide de les faire exécuter par le SGBD du mobile ou de les envoyer à son partenaire du réseau fixe. Charcot situé sur le serveur fait exécuter les transactions qu'il reçoit de son partenaire par le SGBD du S.I.H.

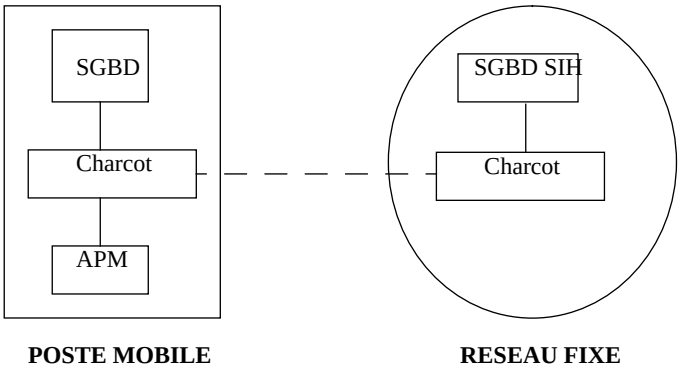

Figure 2 : architecture générale du système

Charcot est divisé en cinq services :

- un service de communication fiable permettant de stocker puis de retransmettre les messages ne pouvant être émis temporairement en raison d'une mauvaise liaison ;

- un service de préchargement sur le poste mobile des données susceptibles d'être utilisées par les transactions futures exécutées en local sur le poste mobile ;

- un service de réplication qui permet de créer des copies de données du serveur sur le poste mobile et de maintenir la cohérence entre les deux versions ;

- un service d'exécution qui permet de passer de façon transparente à l'application d'un mode d'exécution des transactions sur le serveur à un mode d'exécution des transactions sur le poste mobile et vice-versa ;

- un service de mesure de la qualité de transmission du réseau sans fil.

Dans un premier temps nous allons décrire chacun de ces services ensuite nous verrons comment nous les assemblons dans Charcot.

2.1 Service de communication fiable

Notre service de communication fiable offre une connexion durable même en cas de déconnexion. Il se place au dessus du service TCP/IP [6]. Le protocole TCP permet d'établir une connexion entre deux machines. Il assure que les messages envoyés arrivent toujours à leur destinataire et dans l'ordre d'envoi pendant une même connexion. En cas de coupure réseau très courte TCP ne termine pas la connexion et cette "micro coupure" est transparente à l'application. En cas de déconnexion trop longue il termine la connexion. Tous les messages qui ont été envoyés durant la déconnexion mais avant que TCP ne la détecte sont perdus. Le service de communication fiable de Charcot établit une connexion durable entre deux machines. Il assure qu'un message envoyé finit par être reçu par le partenaire et que les messages sont reçus dans le même ordre qu'ils ont été envoyés quel que soit le nombre de déconnexions et leur durée.

Pour cela on stocke chaque message envoyé dans un historique, lorsque l'on détecte que le réseau n'est plus disponible on ferme la connexion TCP et on stoppe l'envoi de messages. Lorsque l'on détecte que le réseau est de nouveau disponible on ouvre une nouvelle connexion TCP et on réémet tous les messages de l'historique. Les messages sont numérotés afin que le récepteur ignore les messages qu'il a déjà reçu. Un message est retiré de l'historique lorsque l'on est sûr qu'il a été reçu par le partenaire. On ne pratique pas d'acquittement systématique des messages reçus mais on l'inclut dans chaque message envoyé : chacun des deux partenaires lorsqu'il envoie un message indique quel est le numéro du dernier message qu'il a reçu. Lorsqu'un des deux partenaires n'émet aucun message il envoie un message d'acquittement, lorsqu'il a reçu un certain nombre de messages de son partenaire.

Ce service est utilisé lorsque la partie cliente de Charcot veut communiquer avec la partie serveur. Il est nécessaire car bien que les déconnexions longues sont supposées très rares, il peut y en avoir. Ce service permet de rendre Charcot résistant aux déconnexions.

2.2 Service de réplication

Notre service de réplication a pour objectif de maintenir à jour sur le poste mobile des copies de données du serveur (site maître). Dans le cadre des applications qui nous intéressent une donnée est une ligne dans une table, une ligne est identifiée par sa clé primaire. Nous appelons *snapshot* une copie partielle ou totale d'une table du site maître. Le service de réplication assure une mise à jour régulière ou sur de-

mande des *snapshot*s avec les données du site maître ainsi qu'une mise à jour du site maître avec les données des *snapshot*s.

2.2.1. Création d'un *snapshot*

Un *snapshot* peut être vide lors de sa création auquel cas seule la structure de la table du site maître est recopiée sur le poste mobile. Il peut également être initialisé avec un sous-ensemble des lignes de la table. Pour obtenir ce sous-ensemble on utilise une requête SQL SELECT fournie par l'utilisateur du service : le service de réplication exécute cette requête sur le site maître et transfère les lignes résultantes dans le *snapshot*.

2.2.2. Ajout et suppression de données dans un *snapshot*

Lorsqu'un *snapshot* a été créé l'utilisateur du service peut à tout moment demander au service d'y ajouter des données, il peut par exemple créer un *snapshot* en l'initialisant avec l'ordonnance du patient *x* puis par la suite ajouter l'ordonnance du patient *y*. Pour savoir quelles sont les lignes de la table à ajouter au *snapshot* l'utilisateur du service doit à nouveau fournir une requête SQL SELECT. Il est également possible de supprimer des données en procédant de la même façon.

2.2.3. Mise à jour des *snapshot*s

Les lignes des *snapshot*s sont mises à jour par le service de réplication à l'aide d'opérations (insertions, suppressions, modifications) qui permettent d'actualiser la valeur des lignes du *snapshot* avec celles du site maître. Les mises à jour ont lieu régulièrement ou sur demande de l'utilisateur du service. Nous expliquons ici comment nous déterminons les opérations qui permettent cette mise à jour.

Nous créons un journal par table du site maître dans lequel nous indiquons les mises à jour effectuées (lignes écrites) dans cette table. Chaque mise à jour est numérotée et contient le type de la mise à jour (insertion, suppression, modification). Pour une insertion on indique la valeur insérée, pour une suppression on indique la valeur avant suppression, pour une modification on indique la valeur avant et après modification. Grâce à ce journal il nous est facile de connaître les écritures ayant été opérées sur une table donnée. Parmi toutes ces écritures il y en a qui peuvent intéresser le *snapshot* et d'autres non. Nous utilisons la condition de la clause WHERE de la requête fournie lors de la création du *snapshot* ainsi que celles fournies lors de l'ajout ou de la suppression de données pour créer une requête de sélection des données qui filtre les données qui sont censées être sur le *snapshot* ou qui devraient y être. La requête de création du *snapshot* ainsi que chaque requête d'ajout de données permettent d'obtenir les conditions que les données doivent respecter pour être dans le *snapshot*. Les requêtes de suppression de données permettent d'obtenir les conditions que les données ne doivent pas satisfaire pour être dans le *snapshot*. Pour obtenir la condition du filtre on opère la différence entre l'union des premières et l'union des dernières. Après application sur le journal du filtre que l'on vient de calculer on obtient un ensemble de lignes qui doivent être transférées au *snapshot*. Pour chaque

ligne on obtient l'ensemble des opérations successives qu'elle a subit. On ne garde pour chaque ligne que sa valeur avant la première opération et sa valeur après la dernière opération. Si on n'obtient pas de valeur avant cela signifie que la ligne n'est pas dans le *snapshot*, l'opération à appliquer sur le *snapshot* est donc une insertion ; si on n'obtient pas de valeur après cela signifie que la ligne est dans le *snapshot* mais qu'elle n'a plus lieu d'y être, l'opération à appliquer est donc une suppression ; si on obtient les deux valeurs on détermine l'opération à appliquer en fonction de la nature de la première opération et de la dernière opération effectuée sur la ligne.

Grâce à cette méthode on assure que les lignes dont on a demandé la présence sont sur le *snapshot* et seulement celles-là et on assure qu'elles sont à jour. Cette méthode présente l'avantage de ne pas consommer trop de ressources sur le site maître. En effet il suffit d'avoir un journal pour chaque table répliquée, le nombre de journaux est donc indépendant du nombre d'utilisateurs mobiles ce qui permet d'étendre facilement le système.

2.2.4. Réintégration des transactions effectuées sur le poste mobile

Le service de réplication utilise un journal fournit par le SGBD du poste mobile dans lequel chaque opération d'écriture effectuée sur le poste mobile est notée. Pour chaque opération sont indiqués à quelle transaction elle correspond ainsi que son numéro d'ordre dans la transaction ; sont indiqués également la nature de l'opération (insertion, modification, suppression) ainsi que la valeur de la ligne avant et après l'opération. Le service de réplication consulte ce journal régulièrement ou à la demande de l'utilisateur du service afin de réintégrer les transactions. Pour une même ligne modifiée par n opérations d'une même transaction on ne garde que la valeur avant la première opération et la valeur après la dernière opération. Les transactions que l'on envoie au serveur afin d'être réintégrées sont donc constituées d'un ensemble de lignes écrites. Les transactions sont envoyées par le service de réplication côté mobile (SRM) au service de réplication côté serveur (SRS). Elles sont envoyées une par une et dans l'ordre de leur exécution sur le poste mobile. Lorsque le SRS reçoit une transaction il regarde si elle peut être réintégrée en opérant une détection des conflits éventuels entre cette transaction et celles ayant été exécutées sur le site maître : soit *T* la transaction à réintégrer, on doit déterminer si *T* est réintégrable pour cela on compare la valeur avant *T* de chaque ligne *l* écrite par *T* avec sa valeur actuelle sur le serveur. Si la valeur est différente on détecte un conflit. Une écriture peut être une insertion, une suppression ou une mise à jour. Soit *before(l)* la valeur sur le *snapshot* d'une ligne *l* avant l'exécution de *T*, soit *present(l)* la valeur actuelle d'une ligne *l* sur le site maître. *present(l)=ø* signifie que la ligne n'existe pas sur le site maî-

tre. Soit *valIns(l)* la valeur d'une ligne insérée par *T* dans le *snapshot*. On a le tableau suivant :

Tableau 1 : détection des conflits

	present(l)=ø	present(l)=x	present(l)=y
INSERT ValIns(l)=x	Pas conflit	Pas conflit	Conflit
DELETE Before(l)=x	Pas conflit	Pas conflit	Conflit
UPDATE Before(l)=x	Conflit	Pas conflit	Conflit

Si aucun conflit n'est détecté le SRS met à jour sur le site maître chaque ligne modifiée par la transaction. Si au moins un conflit est détecté le SRS indique au SRM que la transaction ne peut pas être réintégrée. Ce dernier calcule alors la liste des transactions validées localement qui sont dépendantes de la transaction *t* qui vient d'être refusée. Ensuite il va déterminer l'ensemble des lignes qui ont été modifiées par ces transactions (*t* comprise) et il va demander leur valeur actuelle au SRS afin de les actualiser, ce qui a pour effet d'annuler les opérations opérées par ces transactions et d'actualiser les lignes concernées par ces opérations.

2.3 Service de préchargement des données sur le poste mobile

Notre service de préchargement des données assure que les données nécessaires à l'application sont présentes sur le poste mobile avant ou au moment où celle-ci en a besoin, tout en chargeant le moins possible de données sur le poste mobile. Ce service se divise en deux parties. La première partie consiste à charger les données au coup par coup : les données ne sont chargées qu'au moment où l'application en a besoin. La seconde partie consiste à anticiper le chargement des données : les données dont l'application est susceptible d'avoir besoin dans le futur sont chargées à l'avance. Pour cela l'application indique à des moments précis de son exécution qu'elle va utiliser telle donnée. Le service est alors capable de la bloquer jusqu'à ce que la donnée soit disponible, il est également capable de calculer quelles sont les données dont l'application aura besoin dans l'avenir et quelles sont les données dont elle n'a plus besoin.

Le service de préchargement de données repose entièrement sur le service de réplication pour le chargement effectif des données physique ainsi que pour assurer leur mise à jour régulière.

2.4 Service de changement de mode d'exécution

Le rôle de ce service est d'effectuer le passage entre l'exécution des transactions sur le poste mobile et l'exécution des transactions sur le serveur.

Nous introduisons trois mode de connexion :

- le mode fortement connecté ;
- le mode connecté ;
- le mode faiblement connecté.

Dans le mode fortement connecté les transactions sont exécutées sur le serveur et on n'effectue pas de préchargement de données. Dans le mode connecté les transactions sont également exécutées sur le serveur mais on effectue le préchargement de données. Dans le mode faiblement connecté les transactions sont exécutées sur le poste mobile et on effectue le préchargement de données. Les deux passages qui nous intéressent particulièrement sont le passage du mode connecté au mode faiblement connecté ainsi que le passage du mode faiblement connecté au mode connecté ; le seul intérêt du mode fortement connecté étant de ne pas effectuer de préchargement si cela n'est pas nécessaire.

Passage du mode connecté au mode faiblement connecté : il faut attendre que la transaction en cours qui s'exécute sur le serveur soit terminée. Ensuite if faut s'assurer que les données préchargées sur le poste mobile sont à jour. Lorsque tout est prêt de nouvelles transactions sont autorisées à s'exécuter et à partir de maintenant elles seront exécutées sur le poste mobile.

Passage du mode faiblement connecté au mode connecté : il faut attendre que la transaction en cours d'exécution sur le poste mobile soit terminée. Ensuite if faut s'assurer que toutes les transactions qui ont été validées sur le poste mobile ont été réintégrées. Lorsque tout est prêt de nouvelles transactions sont autorisées à s'exécuter et à partir de maintenant elles seront exécutées sur le serveur. Il est important de noter qu'une transaction n'est jamais coupée en cours d'exécution. Nous n'avons pas besoin de le faire car il y a peu de déconnexions et lorsqu'il y en a elles sont de courte durée. Si il y a une déconnexion en cours de transaction, le service de communication fiable assure qu'elle pourra continuer normalement après la reconnexion.

2.5 Service de mesure de la qualité de communication

Ce service permet d'avertir les autres services lorsque la qualité de communication passe au dessus ou en dessous de certains seuils. Les services qui souhaitent être avertis de tels changements s'abonnent au service de mesure de la qualité. Lors de son abonnement le futur abonné indique les différents seuils qui l'intéressent.

Lorsque ces seuils sont atteints le service avertit l'abonné. La qualité est mesurée en fonction du rapport signal bruit du signal reçu par le poste mobile.

2.6 Coopération entre services dans Charcot

Chaque service est distribué en deux parties : une partie sur le poste mobile et une partie sur le réseau fixe. Ces parties communiquent en utilisant le service de communication fiable. La figure 3 montre les liens entre les différents services : le service d'exécution utilise le service de réplication afin de forcer la réintégration des transactions lors du passage du mode faiblement connecté au mode connecté. Le service d'exécution s'abonne au service de mesure de la qualité de communication afin d'être averti des variations de la qualité de communication. Le service d'exécution indique au service de préchargement d'activer ou de désactiver le préchargement des données ; le service de préchargement utilise le service de réplication pour réaliser le préchargement physique des données.

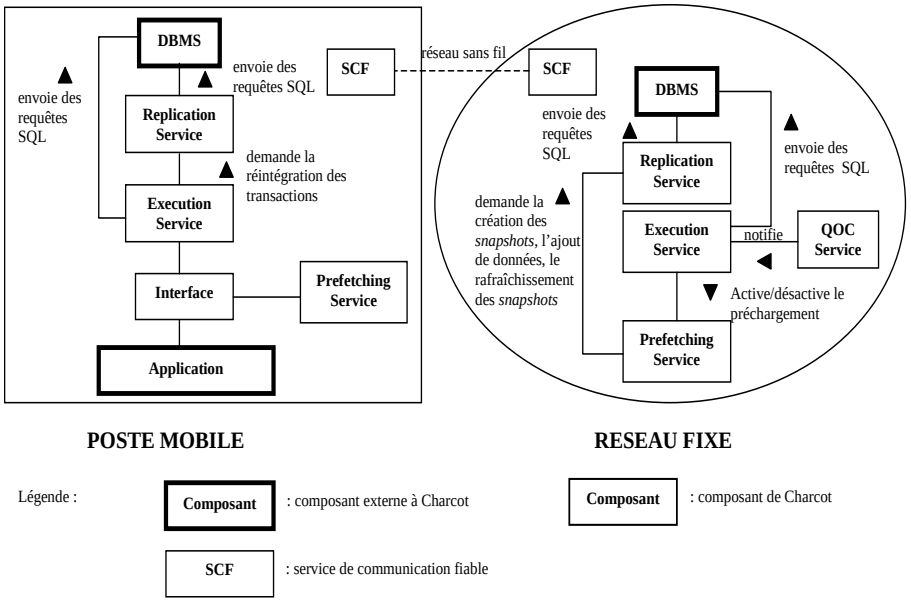

Figure 3 : architecture détaillée de Charcot

3 Travaux apparentés

De nombreux travaux ont été menés sur les problèmes liés à l'utilisation de réseaux sans fil. Ces travaux considèrent souvent le travail en mode déconnecté comme mode de travail normal sur un poste mobile. On trouve dans ce domaine des systèmes académiques comme CODA [7][8] qui permet de répliquer des fichiers sur un poste mobile. L'utilisateur de CODA travaille toujours sur des copies locales même lors-

qu'il est connecté ce qui nous l'avons vu n'est pas souhaitable. CODA se charge de réintégrer sur le serveur de fichiers les modifications effectuées sur le poste mobile en détectant les conflits. On trouve des solutions comparables dans des systèmes de gestion de bases de données transactionnelles du commerce :

Oracle Lite [9] est un SGBD allégé permettant de créer des copies de données du site maître sur le poste mobile. Ces copies sont appelées des *snapshot*s. Une fois que les *snapshot*s sont créés l'utilisateur est en mesure d'exécuter des transactions sur le poste mobile. Oracle Lite permet de rafraîchir les *snapshot*s avec les données du site maître et de réintégrer sur le site maître les transactions validées sur le poste mobile. Ces deux opérations sont réalisées en même temps au sein de la même opération appelée "rafraîchissement" et uniquement à la demande de l'utilisateur. Lorsque l'application demande un "rafraîchissement" elle est bloquée jusqu'à ce que celui-ci soit terminé. Oracle Lite possède un inconvénient majeur dans le cadre qui nous intéresse : il est impossible de faire de la gestion de cache. Il est par exemple impossible de créer un *snapshot* vide et d'y ajouter des données plus tard, il n'est pas possible non plus de supprimer des données dont on n'a plus besoin. Un autre inconvénient est qu'Oracle Lite ne détecte pas les conflits lecture/écriture.

"Sybase SQL anywhere" [10] est lui aussi un SGBD allégé permettant de répliquer des données d'un site maître sur un poste mobile. Il effectue indépendamment et en tâche de fond la réintégration sur le site maître des transactions validées ainsi que la mise à jour des données du poste mobile. Il est possible avec ce système de faire de la gestion de cache bien que ceci ne soit pas prévu dans le logiciel. Il est donc nécessaire de concevoir un certain nombre de scripts qui se révèlent complexes pour arriver au résultat escompté. Le service de réplication de Charcot possède l'avantage d'intégrer cette fonctionnalité. SQL anywhere ne détecte pas non plus les conflits lecture/écriture.

En ce qui concerne la gestion de connexions intermittentes nous sommes assez proches d'un système comme DIANA [11]. DIANA est une architecture permettant de séparer l'affichage et les communications des traitements de l'application, ce qui permet de rendre les applications indépendantes d'un système d'affichage ou de communication particulier. DIANA s'intéresse également à la gestion des connexions/déconnexions. Dans DIANA l'application est située sur le réseau fixe, l'interface utilisateur étant située sur le poste mobile. Les données échangées entre l'application et le poste mobile sont respectivement des formulaires et des réponses aux formulaires, une transaction étant un ensemble de formulaires. Lorsque l'utilisateur est connecté l'application lui envoie les formulaires et celui-ci envoie ses réponses en retour. Lorsque l'utilisateur est déconnecté un 'remplaçant' de l'application est démarré sur le poste mobile pour prendre en charge les requêtes de l'utilisateur. Lors de la reconnexion tous les formulaires remplis pendant la déconnexion sont transmis à l'application qui les traite. Si une erreur survient pendant le traitement le formulaire est renvoyé à l'utilisateur. Bien que Charcot ne considère pas actuellement les déconnexions de longue durée notre système exécute également les transactions tantôt sur

le réseau fixe et tantôt sur le poste mobile, mais la nature des transactions n'est pas la même que dans DIANA. Charcot et plus particulièrement son service de réplication est prévu pour des applications transactionnelles qui utilisent un gestionnaire de base de données externe et une transaction est une suite de requêtes sur cette base alors que DIANA est prévu pour des applications qui n'utilisent pas de gestionnaire de base de données et une transaction est une suite de formulaires.

4 Conclusion

Charcot permet à une application transactionnelle de s'adapter aux variations de la qualité de transmission d'un réseau local sans fil. Charcot exécute les transactions sur le réseau fixe lorsque la qualité de transmission est bonne et sur le poste mobile lorsque la qualité de transmission est faible. Nous avons réalisé une maquette composée de Charcot et d'une application de prescription médicale simplifiée. L'application mise en œuvre suit le même déroulement qu'une application réelle. Nous pensons prochainement intégrer une application de prescription médicale réelle avec Charcot. Cette intégration nous donnera l'occasion de perfectionner le service de préchargement de données. Dans ce travail nous avons pris un certain nombre d'hypothèses réalistes avec le domaine d'application envisagé : qualité de transmission souvent suffisante, déconnexions rares et courtes et peu de conflits. Nous envisageons de relâcher ces contraintes afin d'étudier d'autres cas d'application. Notre but est de proposer une architecture modulaire sous forme de composants. Dans cette optique nous étudierons la façon de réagir à l'annulation d'une transaction lors de la phase de réintégration du point de vue de l'application. En effet les transactions exécutées sur le poste mobile sont dans un premier temps validées sur le poste mobile, puis ensuite réintégrées en arrière plan en sachant que la réintégration peut échouer à cause d'un conflit. Il s'agit d'un problème complexe à gérer pour le programmeur de l'application car cela nécessite de revenir en arrière dans le déroulement de l'application. Le problème est de savoir où et comment il faut revenir. Nous pensons également utiliser un mécanisme de résolution de conflits afin d'éviter le plus souvent possible d'annuler des transactions. Nous étudierons également le cas de déconnexions fréquentes en ajoutant la possibilité d'interrompre une transaction qui s'exécute sur le serveur afin de la reprendre sur le poste mobile.

5 Références

[1] G. H. Forman and J. Zahorjan. The Challenges of Mobile Computing. IEEE Computer, 27(6):38-47, Avril 1994.

[2] F. André and M. T. Segarra. A generic Approach to Satisfy Adaptability Needs in Mobile Environments. In proceedings of the 33rd Hawaii International Conference on System Sciences (HICSS-33) Maui, Hawaii, January 4-7 2000.

[3] F. André and M. T. Segarra. A Generic Approach to Build Mobile Applications. Rapport de recherche INRIA N° 3723 Juin 1999 - Thème 1.

[4] F.André and E. Saint Pol. A middleware for transactional Internet applications on mobile networks. In proceedings of the 1998 International Conference on Parallel and Distributed Processing Techniques and Appllications (PDPTA 98, July 13 - 16, 1998, Las Vegas Hilton, USA).

[5] F.André et E. Saint Pol. METIS : un environnement mobile pour les systèmes transactionnels sur Internet. Dans les comptes rendus de la conférence Rencontres Francophones du Parallèlisme des Architectures et des Systèmes (RenPar '11, 8 - 11 Juin, 1999, Rennes).

[6] A. Tanenbaum. Operating Systems : design and implementation. Prentice-Hall ; Englewood Cliffs, USA January 1987

[7] Lily B. Mummert, Maria R. Ebling, M. Satyanaranyanan. Exploiting weak connectivity for mobile file access. In SIGOPS '95 12/95 CO, USA. ACM 1995.

[8] J. J. Kistler and M. Satyanaranyanan. Disconnected operation in the coda file system. In thirteenth ACM symposium on operating system principles, volume 25, pages 213-225, Asilomar conference center, Pacific Grove, USA, 1991.

[9] Oracle Lite. http://www.oracle.com/mobile/.

[10] Sybase SQL Anywhere. http://www.sybase.com/products/anywhere/index.html.

[11] A. M. Keller, O. Densmore, W. Huang and B. Razavi. Zippering: Managing intermittent connectivity in DIANA In Mobile Networks and Applications 2, pages 357-364, USA 1997.

Unité de recherche INRIA Lorraine, technopôle de Nancy-Brabois, 615 rue du jardin botanique, BP 101, 54600 VILLERS-LÈS-NANCY
Unité de recherche INRIA Rennes, IRISA, Campus universitaire de Beaulieu, 35042 RENNES Cedex
Unité de recherche INRIA Rhône-Alpes, 655, avenue de l'Europe, 38330 MONTBONNOT ST MARTIN
Unité de recherche INRIA Rocquencourt, domaine de Voluceau, Rocquencourt, BP 105, LE CHESNAY Cedex
Unité de recherche INRIA Sophia-Antipolis, 2004 route des Lucioles, 06902 SOPHIA-ANTIPOLIS Cedex

Éditeur

INRIA, Domaine de Voluceau, Rocquencourt, BP 105 LE CHESNAY Cedex (France)
http://www.irisa.fr

ISSN 0249-6399