Mobisnap: Um sistema de Base de Dados para Ambientes Móveis

Miguel Cunha[1], Nuno Preguiça[1], José Legatheaux Martins[1],
Henrique João Domingos[1], Sérgio Marco Duarte[1],
Francisco Moura[2] e Carlos Baquero[2]

[1] Departamento de Informática
Faculdade de Ciências e Tecnologia, Universidade Nova de Lisboa
{miguel.cunha, nmp, jalm, hj, smd}@ di.fct.unl.pt

[2] Departamento de Informática
Escola de Engenharia, Universidade de Minho
{fsm, cbm}@ di.uminho.pt

Palavras chave: base de dados móveis; desconexão; prevenção de conflitos; resolução de conflitos.

Resumo

O Mobisnap é um sistema de base de dados para ambientes móveis. O sistema é baseado numa arquitectura cliente/servidor utilizando replicação optimista. Os clientes mantêm uma cópia parcial dos dados e o sistema combina um mecanismo que evita os conflitos – reservas – com outro que permite a sua resolução – transacções móveis. Estes mecanismos permitem que os clientes mantenham a sua autonomia mesmo em situações de desconexão.

I. Introdução

Um ambiente de computação móvel apresenta características distintas em relação aos tradicionais sistemas distribuídos [15]. Embora a evolução tecnológica expectável permita melhorar as características dos dispositivos móveis, estes apresentarão sempre limitações, quando comparados com os computadores fixos. Uma das principais distinções é o facto de a conectividade disponível ser bastante variável e muitas vezes deficiente. A desconexão total por períodos de tempo variável é comum e pode resultar não só da inexistência de rede, mas também ser devida a motivos económicos (custo das comunicações) ou de consumo (a energia disponível é reduzida). Adicionalmente, os dispositivos móveis possuem recursos computacionais limitados. Estas características gerais devem ser levadas em conta no desenvolvimento de sistemas para estes ambientes.

O Mobisnap é um sistema de base de dados para ambientes de computação móvel. A sua arquitectura é baseada num modelo "extended client/server" [7]. O servidor encontra-se numa máquina fixa e mantém a cópia principal dos dados. Os clientes executam num dispositivo móvel e mantém um cópia parcial dos dados. Desta forma, limitam-se os recursos necessários nos clientes ao mesmo tempo que se limitam os prejuízos quando um cliente se perde (por exemplo, devido à danificação dum dispositivo móvel).

Em situações de desconexão, os clientes mantêm a sua autonomia e permitem aos utilizadores a continuação do seu trabalho. As aplicações que correm no dispositivo móvel podem aceder à cópia local dos dados e submeter alterações sobre a forma de transacções móveis [14]. Uma transacção móvel é um programa escrito numa linguagem semelhante ao PL/SQL [11] e que permite a especificação exacta das condições de execução da transacção, incluindo pré-condições, pós-condições e actualizações alternativas. Este mecanismo permite a especificação de regras de resolução de conflitos [16]. Em situações de desconexão, as transacções móveis são executadas no cliente devolvendo um resultado provisório. O resultado final duma transacção é

o obtido aquando da sua execução no servidor – este resultado pode ser diferente do obtido provisoriamente caso tenham existido modificações concorrentes conflituantes na base de dados central. Entre o cliente e o servidor foi implementado um sistema de comunicação com semântica "*exactly-once*".

Para permitir garantir o resultado duma transacção de forma autónoma foi implementado um mecanismo de reservas. Este mecanismo permite ao cliente reservar temporariamente a utilização de determinados valores da base de dados – por exemplo, um cliente pode reservar para si a utilização duma parte dum stock dum produto, permitindo garantir o sucesso das transacções que usem apenas o número de instância reservadas. Um sistema de permissões controla a utilização dos diferentes tipos de reservas pelos diferentes utilizadores. As reservas, ao permitirem determinar o resultado duma transacção de forma autónoma, permitem realizar operações com um elevado grau de segurança mesmo em desconexão e diminuir assim as repercussões das deficiências das comunicações com o servidor. O tempo necessário para obter o resultado duma transacção é também menor.

O modelo de transacções móveis do Mobisnap foi apresentado em [14]. Neste artigo apresentam-se algumas extensões ao modelo original, em particular a introdução de reservas "shared value-use" e dum sistema de permissões associado. Adicionalmente, discute-se a implementação do modelo proposto, assim como de outros componentes do sistema. O resto do artigo encontra-se organizado da seguinte forma. Na secção 2 apresenta-se a arquitectura do sistema, incluindo o mecanismo de transacções móveis e o sistema de reservas. Na secção 3 apresenta-se a implementação de alguns elementos chave do sistema. Na secção 4 exemplifica-se a utilização do sistema no âmbito duma aplicação de suporte a uma força de venda móvel. Na secção 5 compara-se o Mobisnap com sistemas semelhantes e a secção 6 termina o artigo com breves conclusões.

II. Funcionalidades e estrutura do sistema Mobisnap

O sistema Mobisnap tem uma estrutura cliente/servidor. O servidor central é constituído por um servidor de base de dados tradicional complementado por um ou mais *front-ends* que são os intermediários entre os clientes e o servidor central ("extended client/server architecture"). Os clientes são sistemas móveis contendo um sistema reduzido de gestão de base de dados, assim como as componentes cliente do sistema Mobisnap. Cada cliente gere uma réplica móvel de um subconjunto da base de dados central (*mobile snapshot)*. Os clientes podem operar autonomamente e também executar operações ou transmitir transacções móveis para o servidor central via os *front-ends* Mobisnap. Fisicamente, o cliente é tipicamente um computador ou um terminal pessoal portátil. Quanto ao servidor Mobisnap, este é tipicamente constituído por um ou mais servidores que mantêm a réplica principal e persistente dos dados. Este ou estes servidores devem possuir uma elevada disponibilidade.

Após esta breve descrição da arquitectura do sistema, apresentam-se detalhadamente os seus principais mecanismos: transacções móveis e reservas. Esta secção conclui-se com uma descrição dos componentes que constituem o sistema.

A. Transacções móveis

Uma transacção móvel é um programa escrito numa linguagem adaptada da linguagem PL/SQL que é interpretado tanto pelo cliente como pelo servidor e que permite aos clientes realizarem actualizações (eventualmente diferidas) dos dados da base de dados central. A transacção móvel começa por ser interpretada e executada no cliente de forma a que o utilizador do terminal móvel tenha imediatamente acesso a um resultado, eventualmente provisório, da transacção. No entanto, o resultado definitivo e persistente da mesma só é calculado quando a transacção é executada no servidor central. A transmissão da transacção móvel do cliente para o servidor pode ser síncrona ou assíncrona conforme as condições de conectividade e os meios de comunicação existentes. Para este efeito o sistema Mobisnap suporta meios de comunicação tradicionais (conexões TCP/IP quando as condições de conectividade o permitem), assim como

meios menos tradicionais para transmissão de transacções (correio electrónico ou disquetes por exemplo).

Cada cliente Mobisnap mantém duas versões da réplica do *snapshot* da base de dados central, as versões *commited* e *tentative*. A versão *commited* guarda os dados recebidos directamente do servidor, sem que entretanto tenham sido efectuadas quaisquer actualizações sobre os mesmos, podendo representar dados potencialmente desactualizados. A versão *tentative* é baseada nos dados da versão *commited*, mas reflecte também a execução no cliente de todas as transacções móveis entretanto submetidas pelo cliente. Através da utilização das duas cópias dos dados, é possível a uma aplicação pôr em evidência a possível fraca consistência dos dados no cliente e no servidor, mas ao mesmo tempo, utilizando a cópia *tentative,* apresentar uma versão da evolução esperada dos dados no servidor central.

Para aumentar a probabilidade de sucesso da transacção móvel no servidor esta permite ao programador exprimir pré-condições, pós-condições e actualizações alternativas. Por exemplo, ao invés de uma transacção só ter sucesso se o preço no servidor for o indicado ao cliente no momento da venda (naturalmente suportada no terminal móvel), a mesma poderá ter sucesso caso o preço no servidor seja igual ou inferior ao indicado ao cliente. A figura 1 exemplifica esta técnica.

```
-------------------------------- Encomenda ----------------------------------------
BEGIN
       SELECT price, stock INTO prod_price, prod_stock FROM produtos
              WHERE name ='Chocolate 200gr';
       IF prod_price <= 300.00 AND prod_stock >= 50 THEN
         -- UPDATE ENCOMENDAS, STOCK, …
         NOTIFY('SMTP', 'sales007@sales.corp.pt', 'Encomenda executada com sucesso.');
         COMMIT;
       ENDIF;
       ROLLBACK;
ON ROLLBACK NOTIFY( 'SMS', '9199873223','Não foi possivel executar a encomenda.');
END;
```

Fig. 1 - Exemplo de uma transacção móvel.

B. Reservas

Se bem que a forma de conceber as transacções permite por si só aumentar a taxa de sucesso das transacções móveis quando comparadas com outras técnicas tradicionais (*locks* ou estampilhas temporais associadas aos registos por exemplo [3]), é desejável encontrar formas de limitar a divergência entre os dados *tentative* do cliente e os dados *commited* do servidor, assim como aumentar ainda mais a taxa de sucesso das transacções móveis. Para este efeito o sistema Mobisnap introduz um mecanismo de reservas. As reservas assemelham-se a *locks* semanticamente ricos, que têm de ser validados periodicamente através de um sistema de *leases* [2] e que permitem aos clientes desconectados obter alguma forma de garantia sobre o resultado final da transacção quando a mesma for executada no servidor.

Existem diversos tipos de reservas que um cliente/utilizador pode colocar na base de dados. Cada um dos tipos tem uma semântica distinta.

- **Reservas escrow [9]** - são utilizadas sobre dados passíveis de serem divididos. Tipicamente destina-se a dados como valores de *stocks* de produtos, que podem ser repartidos por vários vendedores, permitindo que haja múltiplos vendedores a efectuarem vendas de um produto em simultâneo, sem haver conflitos.
- **Reservas slot** - são utilizadas para reservar o direito a um cliente Mobisnap de inserir um registo com valores pré-definidos. Por exemplo, um vendedor que pretenda reservar o direito de marcar uma reunião em determinado local e numa determinada data, evitando assim que outro o possa fazer.
- **Reservas value-change** - são utilizadas para reservar o direito de alteração de determinados valores existentes na base de dados, como por exemplo, a alteração da descrição de um produto.

- **Reservas value-use** - são utilizadas para reservar o direito de utilização de determinados valores existentes na base de dados, como por exemplo, um vendedor reservar o direito de vender um determinado produto a um determinado preço, salvaguardando-o de possíveis alterações futuras.
- **Reservas anti-lock** ou **shared value-use** - são utilizadas para impedir que algum utilizador reserve um dado valor como por exemplo o direito de incrementar o número total de vendas realizadas.

Cada uma das reservas atribuídas são limitadas no tempo, ou seja, uma reserva só confere as garantias indicadas durante um período limitado de tempo fixado pelo utilizador e limitado pelo gestor do sistema. Desta forma é possível salvaguardar o bom funcionamento do sistema mesmo em situações em que o cliente se encontra desconectado do servidor durante longos períodos. Por outro lado, o sistema Mobisnap tem um sistema de permissões suplementar ao das bases de dados tradicionais que define que utilizadores podem colocar que reservas e limitar o período de validade das mesmas.

As permissões sobre a colocação de reservas são verificadas quando um cliente / utilizador solicita reservas. De resto o sistema utiliza o sistema de permissões convencional da base de dados central que controla os direitos de acesso das transacções móveis em função dos direitos do utilizador que as submete. A verificação das permissões durante a execução no cliente não é necessária.

Cada reserva, uma vez obtida por um utilizador, é transmitida para o seu posto móvel e a sua definição fica armazenada nas bases de dados central e móvel. Quando uma transacção móvel é executada no cliente móvel, o interpretador Mobisnap do cliente verifica se as operações de manipulação do *snapshot* da base de dados têm reservas associadas. Em função das mesmas, é então possível definir o resultado da execução de uma transacção móvel no cliente (no servidor o resultado é o convencional: *commited* ou não). Os resultados possíveis da execução de uma transacção móvel no cliente são os seguintes:

- ***Reservation commit***- quando a transacção móvel terminou a sua execução com sucesso numa instrução de *commit* e todas as operações executadas na transacção móvel estão salvaguardadas por reservas possuídas pelo cliente Mobisnap. Este resultado garante o sucesso da execução da transacção móvel na réplica principal desde que esta seja transmitida para o servidor Mobisnap com todas as reservas utilizadas ainda dentro da sua validade. Se uma das reservas expirar até à submissão da transacção móvel, a garantia deixa de ser válida;
- ***Tentative commit***- quando a transacção móvel terminou a sua execução com sucesso numa instrução de *commit*, embora não existam reservas suficientes para garantir que todas as operações da transacção móvel sejam executadas com sucesso na réplica principal;
- ***Tentative abort***- quando a transacção terminou a sua execução numa instrução de *abort no* cliente;
- ***Unknown***- quando os dados existentes na réplica local do cliente não permitem que seja calculado o resultado da transacção (tipicamente por falta de dados na réplica local).

Assim, através da utilização do mecanismo de reservas passa a ser possível ter uma ideia mais aproximada do resultado de uma transacção mesmo que esta seja transmitida mais tarde para o servidor.

C. Composição do sistema

Nesta subsecção apresentam-se brevemente os principais componentes do cliente e servidor Mobisnap - fig. 2. Tanto o cliente como o servidor dispõem de sistemas de gestão de bases de dados adequados às respectivas funcionalidades. Também, ambos dispõem de interpretadores

das transacções móveis e dos comandos Mobisnap acessíveis ao utilizador. Os dados sobre as reservas, permissões, transacções a transmitir, transacções executadas, etc. estão também armazenados nas bases de dados do cliente e do servidor.

Os interpretadores de comandos estão preparados para a interpretação de código que tem como linguagem base a linguagem PL/SQL e possuem ainda a capacidade de interpretar pedidos de novas reservas e pedidos de criação ou actualização de réplicas de dados (*snapshots*). No cliente o interpretador de comandos tem como objectivo a interpretação de instruções recebidas directamente do utilizador de forma a transformá-las em pedidos para o sistema Mobisnap, podendo ser dirigidos aos dados locais ou dirigidos ao sistema Mobisnap no servidor, com o intuito, por exemplo, da geração de novas reservas, da criação de novas réplicas de dados ou mesmo da submissão de transacções móveis. No servidor o interpretador de comandos interpreta as transacções móveis recebidas dos clientes assim como executa os outros pedidos recebidos dos mesmos.

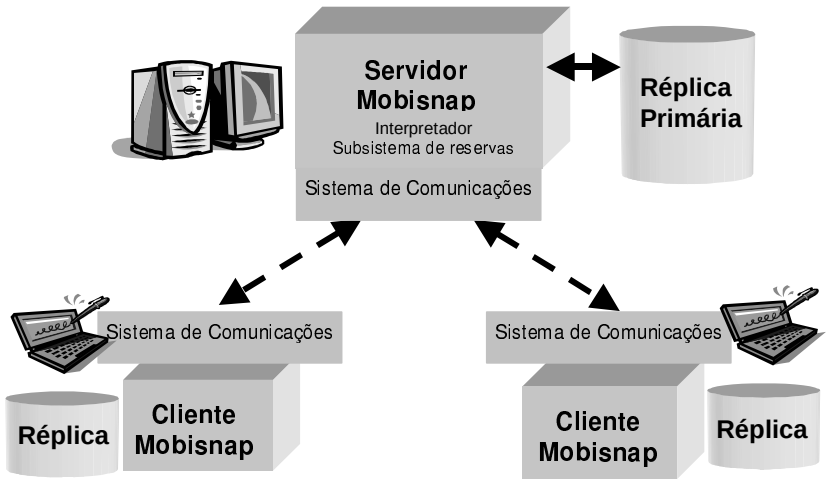

Fig. 2 - Componentes do sistema Mobisnap

Todas as reservas e transacções móveis possuem identificadores únicos e globais e estão registadas nas bases de dados. Desta forma foi relativamente fácil implementar um sistema de comunicações (síncronas ou assíncronas) com uma semântica *exactly once*. Da mesma forma é relativamente fácil um cliente saber mais tarde o resultado (*commited* ou não) da execução de uma transacção móvel submetida assincronamente.

A comunicação entre os clientes e o servidor podem ser síncronas ou assíncronas dependendo do estado da conectividade disponível. Em qualquer dos cenários, como veremos adiante, a semântica da comunicação e do resultado das operações é bem definida. As operações síncronas mais comuns consistem no carregamento de um *snapshot* móvel, na sua actualização, ou na obtenção de reservas (ver adiante). As operações assíncronas mais frequentes consistem na transmissão de uma transacção móvel do cliente para o servidor ou na transmissão de uma notificação do servidor para o cliente. No entanto, quase todas as operações podem ser síncronas ou assíncronas.

O subsistema de comunicações utiliza técnicas criptográficas para garantir autenticação dos utilizadores e a comunicação segura independentemente do carácter síncrono ou assíncrono da interacção. As estampilhas temporais usadas para controlar a validade das chaves e das reservas estão implementadas de tal forma que não exigem a sincronização dos relógios do servidor com os clientes e acomodam relógios de baixa qualidade nos clientes.

O servidor Mobisnap pode ser implementado num *front-end* colocado entre os clientes e um sistema de gestão de bases de dados e uma base de dados legados. Apenas se exige que esse sistema de gestão de bases de dados comporte também as tabelas do *front-end* Mobisnap (de permissões, reservas, estados, etc.) e suporte um sistema de *triggers* o qual se veio a revelar essencial para a implementação das reservas como a seguir será descrito.

III. Opções de implementação do sistema Mobisnap

Nesta secção apresentamos a forma como estão realizados os principais conceitos e funcionalidades do sistema Mobisnap. Os pontos principais que são tratados dizem respeito à implementação das reservas, das permissões e do subsistema de comunicações. O protótipo actual foi escrito em Java e utiliza as bases de dados Oracle e HypersonicSQL, respectivamente no servidor e no cliente.

A. Implementação das reservas

O processamento das reservas envolve os seguintes aspectos: a criação da reserva, o processo de detectar a intenção de utilização de uma reserva quando é submetida uma transacção móvel no cliente Mobisnap e o processamento das reservas associadas a uma transacção móvel quando esta é submetida no servidor.

1) Criação de reservas

Uma reserva é criada através de um comando especial expresso na linguagem do sistema Mobisnap. O comando especifica o tipo da reserva, os dados abrangidos e o tempo de validade da mesma. Se as condições de conectividade o permitirem, o comando é transmitido imediatamente para o servidor. Geralmente as reservas são criadas de forma síncrona, por exemplo, um vendedor efectua os seus pedidos de reservas no início do dia, ligando o seu dispositivo à rede local da empresa. O servidor, ao receber o pedido de criação da reserva, efectua todas as verificações necessárias para validar a criação da mesma e, caso todas as verificações sejam bem sucedidas, a reserva é criada e enviada para o cliente Mobisnap. O descritor de uma reserva compreende o seu tipo, dados envolvidos, período de validade e identificador único não reutilizado (UID). Qualquer reserva válida é também registada na base de dados central. O armazenamento das reservas na base de dados central garante a sua persistência e tolerância a falhas do servidor.

Se um pedido de reserva é aceite, é necessário realizar diversas operações na base de dados central de forma a garantir a semântica das mesmas. Para as reservas *escrow*, o servidor Mobisnap diminui ao registo existente na base de dados central as instâncias reservadas. Desta forma outros clientes do sistema não terão acesso às instâncias reservadas.

A implementação das reservas *value-change* e *slot* baseia-se na construção de um *trigger* que bloqueia qualquer tentativa de concretizar operações de actualização do registo ou registos reservados. Note-se que no caso das reservas *slot* se trata de um conjunto de registos que podem ainda não existir. Quanto às reservas *value-use*, não é necessário realizar nenhuma operação especial na base de dados central aquando da sua criação.

Finalmente, as reservas *shared-value-use* ou *anti-lock* têm também uma implementação simples na medida em que o seu papel é impedir a colocação de outras reservas sobre um registo. Portanto, antes da colocação de uma reserva sobre um registo é necessário verificar se o mesmo não está protegido por este tipo de reserva, caso em que o pedido de reserva deve falhar.

Nas situações em que uma reserva *escrow* expira sem ser totalmente consumida, o valor das instâncias reservadas é reposto no registo de forma a as mesmas poderem voltar a ser utilizadas. A expiração de reservas implementadas através de *triggers* implica também que estes são retirados da base de dados.

2) Detecção da utilização de reservas no cliente Mobisnap

O cliente Mobisnap mantém na sua base de dados a informação sobre as reservas (não expiradas) que possui. A detecção da utilização das reservas numa transacção móvel é efectuada de forma transparente quando a transacção é executada no cliente. Quando a transacção é transmitida do cliente para o servidor, ela leva associada a lista de identificadores das reservas utilizadas.

A utilização das reservas *escrow* é detectada da seguinte forma. Quando se lê um registo na base de dados usando uma expressão PL/SQL do tipo "*select Column into Var*", se existe uma

reserva sobre este registo a reserva fica associada à variável usada para armazenar o valor. Quando são efectuados testes (por exemplo, em expressões IF), verifica-se se estes testes podem ser garantidos pelas reservas existentes (usando eventuais associações entre reservas e variáveis) – caso não possam ser garantidos a transacção não pode devolver o valor *reservation commit* (em [14] detalha-se o modo como é obtido o resultado duma transacção no cliente). Quando se efectua uma instrução de *update* sobre um registo para o qual existe uma reserva, detecta-se o número de instâncias utilizadas.

A detecção da utilização da reserva *value-change* ou *anti-lock* no cliente Mobisnap é realizada verificando as instruções de *update* que existentes na transacção móvel.

A detecção da utilização da reserva *value-use* é semelhante à das reservas *escrow* e efectuada aquando da utilização de variáveis iniciadas em expressões "*select Column into Var*".

Para a detecção de reservas *slot*, verificam-se as expressões do tipo "*select count(*) into Var*" que permitem verificar a existência de registos com determinados valores – estes valores podem ser posteriormente usados em testes. A reserva é associada à transacção móvel quando existe uma operação de inserção ou actualização de dados sobre um registo que se encontra no *slot* definido pela reserva (e verificado aquando das instruções *insert* e *update*).

3) Processamento das reservas aquando da execução de transacções móveis no servidor Mobisnap

O processamento de uma transacção móvel pelo servidor Mobisnap inclui uma fase de pré processamento em que são tomadas em consideração as reservas associadas à transacção. Essas reservas são indicadas pelo cliente e detectadas como descrito anteriormente. Durante a fase de pré-processamento, além de se verificar a validade das reservas utilizadas, a base de dados central é colocada num estado que reflecte as reservas utilizadas pela transacção. De seguida, a transacção é executada normalmente sobre os dados existentes na réplica principal. Finalmente, a fase de pós-processamento serve para actualizar a informação do sistema, nomeadamente a respeitante às reservas.

A fase de pré-processamento das reservas *escrow* consiste em incrementar o registo associado tendo em conta as instâncias utilizadas na transacção – desta forma garante-se que a transacção não falha devido à existência dum valor inferior de instâncias. Um reserva *escrow* pode ser consumida em mais que uma transacção móvel, utilizando cada uma das transacções móveis uma fracção do valor total de instâncias reservadas. Na fase de pós-processamento são actualizados os dados da reserva, incluindo o número de instâncias realmente utilizadas.

As reservas cuja implementação se baseia na utilização de *triggers* têm como fase de pré-processamento a remoção dos mesmos. De seguida a transacção móvel é executada normalmente, sendo o *trigger* reactivado na fase de pós-processamento.

Para as reservas *value-use* o valor a usar pela reserva é actualizado durante o pré-processamento e o antigo valor é restaurado na fase de pós-processamento.

B. Sub-Sistema de permissões

Com o subsistema de permissões é possível controlar os direitos de criação de reservas quer por tipo de reserva, quer pelos dados sobre as quais estas incidem (desde a definição da tabela, até à definição de um conjunto de registos dentro de uma tabela).

As permissões podem ser descritas num ficheiro XML que contém os dados referentes às permissões de cada utilizador ou grupos de utilizadores do sistema. O servidor Mobisnap lê esta informação e guarda-a na base de dados. Para alterar as permissões pode-se alterar o ficheiro de permissões e pedir ao servidor que volte a interpretar o mesmo.

Este subsistema pode ser estendido no futuro, de forma a controlar a criação de réplicas pelo cliente e restringindo o acesso dos clientes à execução de operações no Mobisnap (a não existência desta funcionalidade foi uma das razões pela qual as permissões são especificadas num ficheiro, em vez de existir um função da API que as permitisse modificar). Actualmente

um cliente pode carregar um *snapshot* qualquer desde que tenha direitos de leitura sobre os dados.

C. Subsistema de comunicações

1) Comunicações seguras síncronas e assíncronas

Os principais objectivos deste subsistema são a transmissão fiável dos dados, mesmo em situações de desconexão. Desta forma, salvaguarda-se a entrega dos dados apesar das falhas do cliente e/ou do servidor e/ou do sistema de comunicações. Assim, implementou-se um sistema que permite estabelecer conexões síncronas quando possível e diferir as comunicações quando tal não é possível. Pode-se optar imediatamente por usar uma forma de comunicação assíncrona suportada por diferentes transportes.

Para manter a integridade do sistema Mobisnap e consequentemente dos seus dados, é necessário que as comunicações sejam efectuadas usando uma semântica *exactly-once*. Para tal, cada mensagem tem um identificador único e a informação referente a cada mensagem é guardada na base de dados até se garantir a sua transmissão. O sistema reemite sistematicamente as mensagens cuja recepção ainda não foi confirmada, detecta duplicados e ordena as mensagens transmitidas.

O subsistema de comunicações Mobisnap comporta também a noção de sessão (síncrona ou assíncrona). Uma sessão inicia-se por uma mensagem de pedido de autenticação junto do servidor central (todos os clientes conhecem a chave pública do servidor e o servidor conhece as senhas dos utilizadores). Uma vez o utilizador autenticado com sucesso, a sessão é criada com uma duração limitada. Associada a cada sessão existe uma chave secreta criptográfica que permite proteger o conjunto de mensagens trocadas durante a sessão.

Apesar de este esquema ser suficiente para proteger as interacções entre os clientes e os utilizadores do sistema, a verdade é que ele obriga a períodos, eventualmente longos, de validade das chaves de sessão, devido ao carácter potencialmente assíncrono de todas as sessões, pelo que o mesmo deverá continuar a ser melhor analisado. Outro aspecto que necessita de ser aprofundado é o do cálculo dos *time-outs* que devem ser usados em função dos diferentes tipos de transportes utilizados.

2) Transmissão de reservas entre clientes

Um outro aspecto intimamente relacionado com o subsistema de comunicações e a problemática da autenticação tem a ver com a possibilidade de transmissão de reservas entre clientes. Tal hipótese aumentaria muito a flexibilidade do sistema e a possibilidade de cooperação autónoma dos clientes visto que permitiria a um cliente ceder uma ou mais das suas reservas a outro, sem obrigatoriedade de contacto com o servidor.

Conceptualmente é simples imaginar que a transmissão de uma reserva se limita à transmissão do seu descritor de um cliente para outro. No entanto, isso levanta diversos problemas de segurança porque é necessário que quem recebe a reserva possa verificar que a mesma é autentica, e é também preciso garantir que quem a cede não a vai tentar utilizar posteriormente. De certa forma este problema é em parte semelhante ao de construir um sistema de dinheiro digital a menos de que neste caso o problema do anonimato da utilização das notas digitais não se coloca.

O sistema desenhado baseia-se na analogia do cheque. Uma reserva é um cheque digital emitido pelo servidor e cuja verificação de autenticidade se baseia em técnicas de assinaturas digitais baseadas em criptografia assimétrica (chave pública do servidor). Ceder a reserva a outro cliente é o mesmo que endossar um cheque em que a associação entre o endosso e a reserva, assim como a verificação da autenticidade do endosso, se baseiam também em técnicas de assinaturas digitais baseadas em criptografia assimétrica.

Assim, quando uma reserva chega ao servidor para ser utilizada por uma transacção móvel, este regista a eventual cadeia de endossos da mesma, para posterior detecção de potenciais fraudes pois é assim fácil detectar a reutilização de uma reserva e o responsável pela fraude.

No contexto de aplicação do sistema Mobisnap, julgamos que esta aproximação, que consiste em detectar a posteriori as fraudes, e identificar então os respectivos responsáveis, é suficiente.

IV. A aplicação de teste desenvolvida

Como forma de validar o desenvolvimento do sistema Mobisnap, foi desenvolvida uma pequena aplicação. A aplicação desenvolvida visa o suporte a uma força de vendas, onde existe um conjunto de vendedores que necessitam de grande mobilidade, e um servidor central que gere todo o sistema, possuindo a réplica principal dos dados. Para os vendedores considerou-se que estes trabalham por zonas, ou seja, cada vendedor lida com os clientes de uma determinada região da área de acção da empresa. Tal como os vendedores trabalham tendo em conta uma determinada região, também os clientes da empresa se encontram distribuídos e organizados em regiões. Assim, fazendo face às necessidades do sistema, considerou-se um esquema de base de dados simples, com tabelas de produtos, vendedores, regiões, fornecedores, vendas (registo de todas as vendas efectuadas), entre outras, sendo as apresentadas as mais relevantes. Também de especial interesse foi a introdução de uma "agenda", onde os vendedores podiam registar determinados eventos, como demonstrações de produtos ou reuniões com clientes. Para a agenda, considerou-se que esta podia ser manipulada quer pelo seu utilizador (o vendedor), quer pelo utilizador central, que poderia pretender efectuar uma marcação de reunião para o vendedor.

Sobre o cliente Mobisnap foi desenvolvida uma aplicação baseada em Java Server Pages [5] e Java Beans [4], podendo assim utilizar qualquer *browser* como interface da aplicação. A geração de páginas dinâmicas é efectuada através do pequeno servidor de *servlets* Jetty [6] (que ocupa menos de 200Kb). Esta configuração é perfeitamente compatível com os recursos disponíveis num grande número de dispositivos móveis, incluindo PDAs.

A aplicação cliente possui duas fases distintas. Na primeira fase, em que o cliente é iniciado, o utilizador selecciona os dados que pretende armazenar no seu cliente Mobisnap (tipicamente serão pequenos subconjuntos de toda a informação contida na réplica principal), e reserva os dados necessários. Em relação às reservas, utilizaram-se reservas *escrow* sobre os *stocks* dos produtos – cada utilizador pode reservar uma parcela do valor existente. Para os produtos reservados foram ainda usadas reservas do valor do seu preço, garantindo assim ao vendedor o preço a que este pode vender os produtos. Adicionalmente foram ainda utilizadas as reservas *slot* para a gestão da agenda dos vendedores, podendo estes reservar determinados períodos para as suas reuniões.

Em termos de operações suportadas pelo sistema, foi implementado um esquema de criação de transacções móveis para permitir a submissão de dados relativos a vendas, várias operações de actualização (utilizando reservas *value-change*), inserção e listagem de dados, procurando explorar todas as capacidades do sistema. Todas as funcionalidades da aplicação (incluindo as de iniciação) são desencadeadas pelo utilizador através dum interface gráfico (ver figura 3) – as funções da aplicação geram as correspondentes chamadas à API do Mobisnap. A API do Mobisnap (que inclui métodos para criação de reservas e réplicas de dados, listagens de dados e submissão de transacções móveis) revelou-se de fácil utilização e eficiente, diminuindo as dificuldades de desenvolvimento normalmente associadas a aplicações deste tipo.

Um aspecto interessante foi a possibilidade de contrastar listagens de dados considerados "estáveis", tendo em conta a versão *commited* dos dados e a versão *tentative* dos dados. Desta forma torna-se possível reflectir a divergência entre os dados existentes na réplica do cliente e a réplica principal. Foi também importante verificar o funcionamento das reservas, que cumpriram os seus objectivos, aumentando as garantias dadas ao utilizador do cliente Mobisnap aquando da execução de transacções móveis.

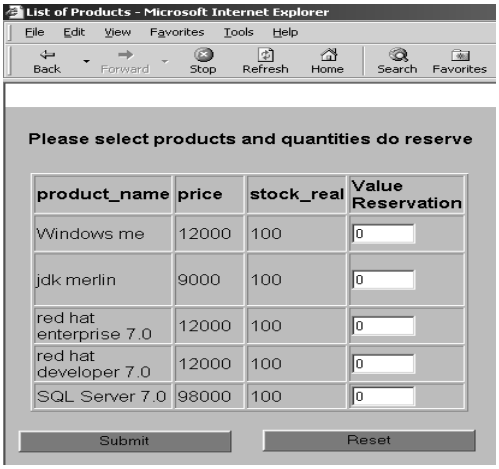

Fig. 3 – Interface para obtenção de reservas *escrow*

V. Trabalho Relacionado

Inúmeros trabalhos de investigação foram desenvolvidos no âmbito da replicação optimista em sistemas distribuídos ([1,8] são apenas dois exemplos). Com o advento da computação móvel, diversos sistemas baseados em modelos de replicação optimista foram criados tendo em conta as novas características. Alguns destes trabalhos são apresentados em [7,13].

O Oracle Lite [10] é uma base de dados comercial que suporta utilizadores móveis. Os clientes mantêm *snapshots* da base de dados central. As alterações efectuadas são propagadas para os servidores onde as transacções válidas são aplicadas. A validade das transacções é verificada através da detecção de conflitos usando os conjuntos de leitura e escrita. É possível definir regras de resolução de conflitos associadas a tabelas (e.g. "o cliente ganha", "o servidor ganha"). No entanto, esta aproximação, baseada na propagação do estado, não permite a utilização de informação semântica associada às transacções na resolução de conflitos. O modelo de transacções móveis proposto é por isso mais flexível.

O Bayou [16] é um sistema de bases de dados que suporta a partilha de dados entre os utilizadores móveis. A sua arquitectura é baseada num conjunto de servidores que mantêm cópias das bases de dados e propaga as escritas entre si. Cada base de dados mantém dois valores – *tentative* e *committed*. Um servidor é responsável por definir a ordem pela qual as escritas devem ser executadas na versão *committed*. As escritas contém informação que permite a detecção de conflitos genérica e regras de resolução de conflitos. O mecanismo de transacções móveis do Mobisnap permite a mesma funcionalidade.

Em [12], os autores propõem um sistema de gestão de transacções para bases de dados móveis. Para cada transacção, o cliente guarda os conjuntos de leitura e escrita. Adicionalmente, devem ser especificadas duas funções: uma de resolução de conflitos e outra de custo. A base de dados central tenta *serializar* as transacções executadas nos clientes com base nas funções de custo e de resolução de conflitos (que é executada sempre). Embora este modelo seja bastante genérico, pensamos que o algoritmo de resolução de conflitos é demasiado complexo para poder ser usado na prática. Adicionalmente, e como em todos os sistemas referidos anteriormente, não existe nenhum mecanismo que permita garantir o resultado de uma transacção de forma autónoma num cliente. Este mecanismo é muito importante porque permite garantir o resultados das transacções efectuadas pelos utilizadores desconectados.

Em [9], os autores propõem a utilização de técnicas de divisão (*escrow*) de dados para permitir garantir o resultado das transacções de forma autónoma em ambientes móveis. A ideia consiste em dividir o número de instâncias representadas num registo por um conjunto de clientes móveis. Cada cliente pode garantir os resultados que usam apenas as instâncias usadas. No nosso trabalho, estendemos esta ideia através do mecanismo de reservas, o que permite garantir o resultado de forma autónoma em novas situações. Além disso, aplicamo-lo a uma base de dados relacional e apresentamos a sua implementação como um sistema de *middleware*.

Em [17], os autores generalizam as técnicas de divisão através da utilização da semântica dos objectos envolvidos. A ideia consiste em dividir objectos complexos em fragmentos menores que podem ser manipulados autonomamente e reintegrados caso se respeitem certas restrições. Apesar de interessante, esta aproximação pode ser usada apenas num pequeno número de objectos (por exemplo, conjuntos) e é mais adequada a sistemas de bases de dados orientados para os objectos. O sistema de reservas proposto no nosso trabalho permite fornecer garantias sobre o resultado das operações efectuadas em desconexão num modelo de base de dados relacional.

VI. Conclusões

O Mobisnap é um sistema de bases de dados para ambientes de computação móvel. Este sistema lida com as características destes ambientes através dum conjunto de mecanismos que actuam de forma combinada. Devido aos recursos computacionais limitados dos dispositivos móveis, apenas um subconjunto dos dados é armazenada nos clientes. A base de dados central mantém a cópia "oficial" dos dados, o que permite limitar os prejuízos quando algum dispositivo móvel se perde.

As modificações da base de dados são efectuadas através de programas escritos numa linguagem semelhante ao PL/SQL. Estes programas são executados no cliente de forma provisória. O seu resultado definitivo é obtido aquando da sua execução no servidor. Este mecanismo de transacções móveis permite à aplicação especificar de forma precisa a semântica das operações a efectuar. Desta forma pode-se evitar ou diminuir a detecção de falsos conflitos. Adicionalmente, pode-se especificar métodos de resolução dos conflitos que possam surgir devido à execução concorrente de transacções conflituantes.

As transacções móveis são complementadas com um mecanismo de reservas que permite, em certas condições, determinar o resultado duma transacção autonomamente num cliente. A obtenção destas reservas é regulada por um sistema de permissões que estende os habituais sistemas de permissões das bases de dados. Desta forma, os clientes reduzem a necessidade de comunicar com os servidores o que permite, para certas transacções, operar em situações de desconexão, reduzir o consumo de energia e reduzir o tempo necessário para a obtenção do resultado. O Mobisnap combina este conjunto de mecanismos de forma inovadora num sistema de *middleware* de suporte à computação móvel.

VII. Referências

1. S. Davidson, H. Garcia-Molina, D. Skeen. Consistency in Partitioned Networks. *ACM Computing Surveys*, C-31, 1982.
2. C. Gray, D. Cheriton, Leases: an efficient fault-tolerant mechanism for distributed file cache consistency. Em *Proceedings of the 12th ACM Symposium on Operating Systems Principles*, 1989.
3. J. Gray, A. Reuter. Transaction Processing: Concepts and Techniques. Morgan Kaufmann Publishers, 1993.
4. http://java.sun.com/products/javabeans/
5. http://java.sun.com/products/jsp/
6. http://jetty.mortbay.org
7. J. Jing, A. Helal, A. Elmagarmid. Client-server computing in mobile environments. *ACM Computing Surveys*, 1999.
8. L. Kawell Jr., S. Beckhardt, T. Halvorsen, R. Ozzie, I. Greif. Replicated Document Management in a Group Communication System. Em *Proceedings of the 2nd ACM Conference on CSCW*, 1988.
9. N. Krishnakumar, R. Jain. Escrow techniques for mobile sales and inventory applications. *Wireless Networks*, 3, 1997.

10. Oracle. Oracle9i Lite whitepaper. Setembro de 2001.
11. Oracle. PL/SQL User's guide and reference - release 8.0. 1997.
12. S. Phatak, B. Badrinath. Transaction-centric reconciliation in disconnected databases. ACM Monet, Julho 2000.
13. E. Pitoura, G. Samaras. Data Management for Mobile Computing. *Kluwer Academic Publishers*, 1998.
14. N. Preguiça, C. Baquero, F. Moura, J.Legatheaux Martins, et al. Mobile transaction management in Mobisnap. Em *Proceedings of ADBIS-DASFAA 2000 (LNCS 1884)*, Setembro de 2000.
15. M. Satyanarayanan. Fundamental Challenges in Mobile Computing. Em Proceedings of the 15th ACM Symposia on Principles of Distributed Computing, 1996.
16. D. Terry, M. Theimer, K. Petersen, A. Demers, M. Spreitzer, C. Hauser. Managing Update Conflicts in Bayou, a Weakly Connected Replicated Storage System. Em *Proceedings of the 15th ACM Symposium on Operating Systems Principles,* 1995.
17. G. Walborn, P. Chrysanthis. Supporting semantics-based transaction processing in mobile database systems. Em *Proceedings of the 14th Symposium on Reliable Distributed Systems*, 1995.