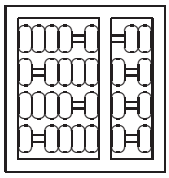

INSTITUTO DE COMPUTAÇÃO
UNIVERSIDADE ESTADUAL DE CAMPINAS

Estudo de Modelos de Transação para o Ambiente de Computação Móvel

Tarcisio da Rocha
Maria Beatriz Felgar de Toledo

Technical Report - IC-05-025 - Relatório Técnico

September - 2005 - Setembro

Estudo de Modelos de Transação para o Ambiente de Computação Móvel

Tarcisio da Rocha Maria Beatriz F. de Toledo*

Resumo

Transações se tornaram uma técnica largamente usada para garantir a consistência de dados, a confiabilidade e a recuperabilidade de aplicações distribuídas. Além dos modelos voltados para sistemas distribuídos tradicionais, a literatura tem apresentado vários modelos de transações para o ambiente de computação móvel. Porém, a maioria dos modelos propostos foram sugeridos e desenvolvidos para um domínio de aplicação específico. Isto torna estes modelos inflexíveis e inadequados para outros domínios de aplicação ou aplicações que necessitem de uma maior flexibilidade. Este relatório apresenta um levantamento de modelos de transação para a computação móvel destacando algumas das suas contribuições e limitações. Neste relatório também é apresentado um conjunto de requisitos que podem ser úteis no desenvolvimento de uma plataforma de apoio a transações no ambiente de computação móvel.

1 Introdução

Avanços nas tecnologias de telecomunicação e de dispositivos de computação portáteis possibilitaram o surgimento da Computação Móvel. Assim, dispositivos como laptops e PDA's equipados com interfaces de comunicação sem fio ganharam a capacidade de participar de computações distribuídas mesmo estando em movimento ou desconectados. Apesar de a computação móvel prover características de mobilidade bastante atraentes, a sua infraestrutura possui um alto grau de dinamismo que não era previsto pelos sistemas distribuídos tradicionais. As aplicações que executam no dispositivo móvel terão que enfrentar problemas como, largura de banda variável da comunicação sem fio, a possibilidade de desconexões freqüentes e inesperadas, conexões com redes heterogêneas, além de problemas inerentes ao dispositivo móvel, como energia e espaço em disco limitados.

Transações se tornaram uma técnica largamente usada para garantir a consistência de dados, a confiabilidade e a recuperabilidade de aplicações distribuídas. Além dos modelos voltados para sistemas distribuídos tradicionais, a literatura tem apresentado vários modelos de transações para o ambiente de computação móvel. Porém, a maioria dos modelos propostos foram sugeridos e desenvolvidos para um domínio de aplicação específico e não possuem flexibilidade suficiente para serem usados em outros domínios de aplicação.

*Ambos do Instituto de Computação, Universidade Estadual de Campinas, 13081-970 Campinas, SP.
Pesquisa desenvolvida com apoio financeiro da CAPES

Este relatório apresenta um levantamento de modelos de transação para a computação móvel destacando algumas das suas contribuições e limitações. É apresentado também um conjunto de requisitos que podem ser úteis no desenvolvimento de uma plataforma de apoio a transações para o ambiente de computação móvel.

2 O Ambiente de Computação Móvel

A computação móvel é um paradigma que permite que um computador portátil equipado com uma interface de comunicação sem fio possa participar da uma computação distribuída independentemente da sua localização física ou estando em movimento. Isto se tornou possível graças aos avanços nas tecnologias de telecomunicação, redes e dispositivos de computação portáteis.

A arquitetura de apoio à computação móvel normalmente apresentada pela literatura [7, 26, 12] é composta por unidades fixas e unidades móveis. Uma Unidade Móvel é um computador móvel capaz de se comunicar com a rede fixa através de um meio de comunicação sem fio. As unidades fixas da rede são máquinas fixas ou estações de apoio que se comunicam entre si através de uma rede fixa de alta velocidade.

Uma Máquina Fixa é um computador da rede fixa que normalmente não possui interface de comunicação sem fio. Uma Estação de Apoio pode ser definida como um computador que possui interface de comunicação sem fio e é, portanto, capaz de manter uma conexão sem fio com unidades móveis. As estações de apoio agem como interfaces entre as unidades móveis e os demais componentes da rede fixa.

Cada estação de apoio possui uma área de cobertura chamada Célula Sem Fio[1]. Esta é a área por onde uma unidade móvel pode se mover e manter a conexão com a estação de apoio correspondente. Ao se mover, uma unidade móvel poderá sair de uma célula e entrar em uma outra mantendo a comunicação com a rede fixa através de outra estação de apoio.

A computação móvel tem exigido várias mudanças no desenvolvimento de soluções no que diz respeito a circuitos integrados, processamento de sinais, projeto de rede e projeto de sistemas computacionais. Essas mudanças são necessárias porque a computação móvel introduziu uma série de novos obstáculos normalmente não existentes em ambientes tradicionais. Alguns destes obstáculos são citados a seguir [9, 16, 12, 32]:

- **Baixa largura de banda** - Redes sem fio possuem uma largura de banda bem inferior a das redes com fio. Técnicas como caching e compressão de dados têm sido usadas para reduzir os impactos da baixa largura de banda.

- **Custo da comunicação** - O custo da comunicação sem fio pode ser alto a depender da tecnologia que se esteja utilizando (por exemplo, via celular). Isto pode exigir o uso de técnicas que reduzam o tempo de alocação do canal de comunicação como a *operação desconectada*.

- **Desconexões freqüentes** - A susceptibilidade a desconexões freqüentes e inesperadas das redes sem fio requer dos dispositivos móveis um certo nível de autonomia,

[1] Wireless Cell

ou seja, requer o poder de continuar a operar enquanto desconectados da rede fixa. Para tanto, são usadas técnicas de operação assíncrona como *busca antecipada*[2] e *escrita com atraso*[3]. A *busca antecipada* é a ação de fazer uma cópia dos prováveis objetos que serão usados por um usuário em seu dispositivo móvel de forma que eles estejam disponíveis localmente quando a desconexão ocorrer. A *escrita com atraso* é uma técnica utilizada para atualizar os objetos remotos da rede fixa aproveitando os momentos em que haja largura de banda adequada para a transferência.

- **Escassez de recursos dos dispositivos móveis** - Recursos como energia, espaço em disco e capacidade de processamento podem ser escassos em dispositivos móveis. Em particular, o suprimento de energia dos dispositivos móveis normalmente é feito por baterias que possuem uma capacidade limitada de suprimento. Isto pode exigir técnicas para a redução do consumo de energia por parte dos elementos de hardware e software.

- **Segurança** - A segurança da comunicação sem fio é bem mais vulnerável do que na comunicação com fio, pois o sinal transmitido através da rede sem fio pode ser interceptado por outros dispositivos de captação intrusos. As soluções adotadas para lidar com este problema são a criptografia e a autenticação feitas por software ou hardware especializado.

- **Maior dinamismo dos dados** - Informações que são consideradas estáticas para um computador estacionário poderão passar a ser dinâmicas para um computador móvel. Um exemplo deste dinamismo é o endereço de rede do computador móvel que pode mudar cada vez que houver mudança de uma célula para outra.

3 Requisitos para Sistemas de Transações no Ambiente Móvel

Um sistema de transações voltado para a computação móvel deveria observar os seguintes aspectos:

1. **Características do ambiente móvel** - muitas das considerações feitas ao se desenvolver sistemas de transações para ambientes de rede tradicionalmente fixos deixam de ser válidas quando se trata do ambiente móvel. Este ambiente é caracterizado pelo seu dinamismo, escassez de recursos e instabilidade, fatores normalmente inexistentes nos ambientes de redes tradicionalmente fixos.

2. **Diversidade de requisitos a serem atendidos** - as aplicações transacionais podem ser de diferentes domínios exigindo assim, a garantia de requisitos específicos. Isto exige do sistema de transações uma certa flexibilidade de forma que os diversos requisitos de aplicações diferentes possam ser atendidos.

[2] Prefetching
[3] Delayed write-back

Diante dos aspectos citados acima, seguem alguns requisitos úteis que podem ser observados no processo de desenvolvimento de sistemas de transações para o ambiente de computação móvel:

- Atender os requisitos de aplicações de diferentes domínios que possam vir a utilizar o sistema de transações. Cada aplicação pode ter diferentes requisitos com relação à garantia de propriedades como:
  - **Isolamento** - algumas aplicações como, aplicações bancárias, exigem a garantia de um total isolamento entre as transações concorrentes. Porém, os mecanismos usados para se garantir um isolamento total entre transações podem gerar efeitos indesejáveis quando aplicados em outros domínios de aplicação. Isto pode exigir uma flexibilização da propriedade de isolamento. Uma forma de se flexibilizar a propriedade de isolamento é através do uso de *níveis de isolamento* [10].
  - **Atomicidade** - aplicar a propriedade tradicional da atomicidade pode causar grandes impactos no desempenho em determinados tipos de aplicações transacionais. Por exemplo, aplicações transacionais de longa duração como CADs poderiam sofrer grandes prejuízos se a cada falha que ocorresse todo o trabalho tivesse que ser desfeito. Isto exige a flexibilização desta propriedade permitindo que partes de uma transação possam ser desfeitas. Uma forma de se flexibilizar a propriedade da atomicidade é através do uso de técnicas de *transações aninhadas* [10].
  - **Durabilidade** - existem tipos de aplicação em que a durabilidade das transações tem que ser mantidas, ou seja, depois que uma transação é efetivada os seus efeitos não devem ser perdidos mesmo com a ocorrência de falhas. Já outros tipos de aplicação podem tolerar falhas que comprometam a durabilidade. Por exemplo, uma transação pode ser considerada efetivada mesmo se alguns de seus resultados não puderem ser efetivados. Desta forma, as regras de garantia da durabilidade poderiam ser personalizados a depender dos requisitos da aplicação transacional.
  - **Regras de consistência** - algumas aplicações exigem regras de consistência mais rígidas, outras mais flexíveis. Isto pode exigir um sistema de transações que permita a definição de regras de consistência personalizadas de acordo com os requisitos da aplicação transacional. Existem também aplicações que utilizam informações semânticas dos dados acessados como meio de flexibilizar regras de consistência. Em [35] é apresentado um modelo de transações baseado em informações semânticas dos dados acessados.
  - **Restrições de tempo** - existem aplicações que possuem restrições de tempo em seu processo de execução. Por exemplo, transações de tempo real exigem um tempo máximo para a sua execução. Em [18] é apresentado um modelo voltado para transações de tempo real.
- Devido a fatores como, o alto dinamismo do ambiente de computação móvel, os diferentes requisitos de aplicações e os diferentes cenários onde existe computação móvel,

um dos novos requisitos dos sistemas de transação é que eles sejam configuráveis antes da execução e reconfiguráveis em tempo de execução. Ser configurável antes da execução é uma característica que permite ao sistema se adequar ao ambiente do qual ele fará parte e se adequar às aplicações às quais ele apoiará. Ser reconfigurável em tempo de execução também é uma característica importante que permite ao sistema se adequar ao dinamismo do meio. As considerações usadas para configurar o sistema antes do período de execução podem deixar de ser apropriadas em tempo de execução devido ao dinamismo do ambiente móvel.

- O sistema de transações deve prover mecanismos de adaptação ao dinamismo do meio. Estes mecanismos devem respeitar as individualidades (requisitos) de cada aplicação de forma que uma ação de adaptação tomada pelo sistema para favorecer uma aplicação não seja prejudicial às demais.
- Ser adequado aos dispositivos de computação móvel em questões como memória disponível e capacidade de processamento.
- Ser extensível de forma que possa se adequar a requisitos de novas aplicações ou novos modelos de transação.

4 Modelos de Transações para a Computação Móvel

Nesta seção será apresentado um resumo dos principais modelos de transação para o ambiente móvel citados na literatura.

4.1 Coda-IOT

O Coda [33] é um dos trabalhos pioneiros e relevantes na área da computação móvel. Apesar de não ter sido este um dos seus objetivos iniciais, o Coda é um sistema que provê acesso a dados a partir de máquinas móveis. Para isto, esse sistema utiliza técnicas como operação desconectada e operação fracamente conectada.

Os primeiros enfoques do Coda foram voltados ao tratamento de falhas de partição que afetam os usuários de sistemas distribuídos. Para contornar estes problemas foram propostas basicamente duas técnicas: a *replicação de servidor* e a *operação desconectada*. A *replicação de servidor* é uma técnica fundamental para a garantia da disponibilidade de dados durante falhas que ocasionam partição da rede. O Coda utiliza um conjunto de servidores que contém réplicas dos dados úteis aos usuários da rede. Assim, quando ocorre a partição da rede, um usuário poderá acessar os dados de um outro servidor na mesma partição em que ele se encontra.

A *operação desconectada* é uma técnica usada principalmente para garantir a disponibilidade em circunstâncias de falhas em que a replicação de servidor não resolve. Um exemplo disso seria a perda de comunicação com todos os servidores da rede. A idéia central da operação desconectada é manter na cache da máquina do usuário uma cópia dos dados úteis ao mesmo. Assim, com a desconexão, o usuário poderia continuar as suas operações sobre a cópia em cache.

Estudos posteriores deram ao Coda a capacidade de operar em ambientes com comunicação sem fio, caracterizados pela largura de banda baixa e intermitente. Para isto, foi utilizada uma técnica chamada operação fracamente conectada [24] que foi apresentada como um meio de aliviar as limitações da operação desconectada. A operação desconectada apresentou problemas como: a falta de dados em cache, comprometendo o andamento das atividades do cliente; a não visibilidade das atualizações sobre dados em cache para outros clientes; conflitos de atualização dos dados nos servidores que se tornam mais evidentes quanto maior é a duração da operação desconectada. Para superar estes problemas, a operação fracamente conectada trouxe modificações ao Coda. Primeiro, o protocolo de manutenção dos dados em cache foi modificado para permitir uma rápida revalidação de grandes volumes de dados em cache depois de um período de desconexão. Além disso, foi desenvolvido um mecanismo chamado reintegração por partes que utiliza, de forma flexível, a largura de banda da rede na propagação da atualização dos dados em cache para os servidores.

Os modelos iniciais de replicação de dados no Coda só visavam a resolução de conflitos do tipo escrita/escrita. Os conflitos do tipo leitura/escrita eram ignorados visto que a possibilidade de sua ocorrência era tolerada pelos usuários de sistemas de arquivos compartilhados. Porém, com a operação desconectada a ocorrência destes conflitos aumentou de forma considerável. Para lidar com este problema, o Coda foi estendido com um mecanismo chamado Isolation-Only Transaction (IOT) [20]. Este mecanismo permite ao sistema admitir somente os conflitos leitura/escrita que satisfaçam certos critérios de serialização. Assim, o IOT busca dar uma melhor garantia de consistência a aplicações que executam no ambiente móvel.

No decorrer de anos de experiência com o Coda, observou-se que um mecanismo de cache automático totalmente transparente ao usuário não era suficiente para garantir a disponibilidade de dados na operação desconectada. A solução adotada foi o mecanismo chamado caching translúcido. Este mecanismo faz um balanço entre a total transparência e o total controle manual do processo de caching de dados. Com isso, o usuário passa a fazer parte do processo de seleção dos dados que serão carregados em cache enquanto que o sistema cuida dos detalhes de gerenciamento.

4.2 PRO-MOTION

O PRO-MOTION [36] é um sistema de processamento de transações projetado para lidar com problemas introduzidos pela desconexão e pelos recursos limitados do ambiente móvel. Seu principal bloco de formação é o *compact* que funciona como uma unidade básica de replicação de dados.

Um *compact* é representado como um objeto que encapsula: dados, operações para acesso aos dados, operações para o gerenciamento do compact, informação a respeito do estado corrente do compact, regras de consistência e obrigações. O compact representa um acordo entre o cliente móvel que requisitou o dado e o servidor de banco de dados. Neste acordo, o servidor de banco de dados delega à unidade móvel o controle de alguns dados que serão usados por transações locais. O cliente móvel, em contrapartida, concorda em honrar com as condições específicas do uso dos dados de maneira que sua consistência seja mantida

quando as atualizações forem propagadas de volta para o servidor de banco de dados. Cada compact é responsável pelo controle de concorrência e recuperação dos seus dados.

O PRO-MOTION sugere quatro fases de processamento de transações:

1. **Armazenamento**[4] - A unidade móvel está conectada à rede e os compacts estão sendo armazenados localmente na preparação para uma eventual desconexão.

2. **Processamento conectado** - A estação móvel está conectada à rede e as transações estão acessando dados diretamente do servidor.

3. **Processamento desconectado** - A estação móvel está desconectada da rede e as transações estão acessando dados locais contidos nos compacts.

4. **Ressincronização** - A estação móvel se reconectou à rede e as atualizações feitas durante a desconexão estão sendo reconciliadas com o banco de dados no servidor.

O PRO-MOTION permite efetivação[5] local de transações. Caso uma transação opte pela efetivação local, seus resultados se tornarão visíveis para outras transações da unidade móvel. Se a transação não iniciar com a opção `LOCAL`, ela só efetuará a efetivação local após a efetivação no servidor.

A consistência das operações sobre os dados no Pro-Motion pode ser caracterizada através de uma escala com dez níveis de isolamento baseados no padrão ANSI-SQL. O nível 9 garante uma total execução serial de transações e o nível 8 garante uma execução serializável[6]. Cada nível inferior representa um menor nível de isolamento. O nível 0 não garante nenhum tipo de consistência.

Cada método em um compact pode ser associado a um nível de isolamento e a um modo que indica se a operação é de leitura ou de escrita. Assim, um compact poderá conter métodos que possuem níveis de consistência diferentes. Uma transação também poderá especificar o seu nível mínimo de isolamento para operações de leitura e o seu nível mínimo para operações de escrita de forma que nenhuma das operações chamadas sobre os dados de um compact seja menor que estes níveis especificados.

4.3 Reporting e Co-Transactions

Reporting Transactions e *Co-Transactions* [4] é um modelo transacional aninhado aberto que relaxa as restrições impostas pelas transações atômicas. As transações atômicas tradicionais possuem restrições que impedem o particionamento da computação e o compartilhamento de resultados parciais. Porém, este trabalho considera que o particionamento da computação e o compartilhamento dos resultados parciais das transações no ambiente móvel são necessários devido à escassez de recursos nos dispositivos computação móveis e outros problemas como, desconexões e baixa largura de banda. Sendo assim, para reduzir os impactos destes problemas, este modelo propõe que as transações móveis sejam compostas por um conjunto de transações onde, parte executa na unidade móvel e parte na rede fixa.

[4] Hoarding
[5] Commit
[6] Serializable

Uma *Reporting Transaction* é uma transação que pode compartilhar alguns dos seus resultados parciais delegando-os a uma outra transação. Essa delegação pode ser feita em qualquer ponto de sua execução. Dessa forma, uma transação executando no computador móvel pode repassar resultados parciais para outra transação que executa na rede fixa.

Co-Transactions podem ser definidas como sendo *Reporting Transactions* que têm um comportamento semelhante a co-rotinas onde o controle é passado de uma transação para outra no momento do compartilhamento dos resultados parciais. As *Co-Transactions* diferem das *Reporting Transaction* pelo fato de que a execução das *Co-Transactions* são suspensas no momento da delegação e são reiniciadas a partir do ponto onde foram previamente suspendidas enquanto que as reporting transactions continuam a executar concorrentemente com a transação para a qual ela delegou seus resultados parciais.

4.4 Kangaroo

O modelo de transações Kangaroo [8] foi projetado como um modelo transacional que assimila características de mobilidade, ou seja, o local de origem de uma transação pode ser diferente do local de finalização da mesma. As transações Kangaroo são vistas como transações móveis que incorporam a propriedade de que transações, em um sistema de computação móvel, saltam de uma estação base para outra à medida que a unidade móvel se move entre células.

A execução de uma transação Kangaroo pode consistir na execução de um conjunto de transações chamadas Joey. Uma transação Kangaroo inicia a sua execução através de uma transação Joey em uma determinada estação base. À medida que a unidade móvel onde a transação Kangaroo se encontra se move de uma célula para outra, transação Joey da célula anterior é efetivada e é criada uma transação Joey na estação base da nova célula. Cada transação Joey representa uma parte da transação Kangaroo que foi executada em uma determinada célula.

Uma transação Kangaroo pode ser executada no modo *compensating* ou no modo *split*. Quando uma transação Kangaroo executa no modo compensating, uma falha em uma transação Joey força a transação Joey corrente e todas as demais transações Joey da transação Kangaroo a serem desfeitas. No modo split, quando ocorre uma falha, a decisão de efetivar ou de abortar as transações Joey passa a ser responsabilidade do DBMS.

Este modelo foi projetado com base nos conceitos de transações globais e transações split [28].

4.5 AMT

AMT [34] é um modelo que tem como objetivo dar a transações a capacidade de se adaptar à alta instabilidade do ambiente de computação móvel caracterizada por largura de banda variável, desconexões e diferentes custos de comunicação. Para isto, o modelo dá às transações a capacidade de estar cientes das modificações ocorridas no ambiente e de escolher a alternativa de execução mais adequada. De uma forma geral, uma transação AMT é composta por pelo menos uma alternativa de execução envolvendo um ou vários hosts fixos ou móveis. O sucesso de uma destas alternativas representa a execução correta

da transação.

Uma transação AMT pode conter descritores de ambiente que expressam os estados no qual o ambiente deve estar para que cada alternativa seja executada. Quando uma transação é iniciada, o estado do ambiente é verificado e a alternativa de execução apropriada é escolhida. Se não existir nenhuma alternativa para o estado do ambiente, a execução da transação pode ser adiada. Desta forma, uma alternativa de execução será executada assim que um estado aceitável do ambiente for detectado.

Um Descritor de Recursos `ED={dimension=value(s)}` define um conjunto de dimensões com os seus respectivos valores em um dado instante. A Tabela I introduz um conjunto de dimensões que podem influenciar na execução de transações.

Tabela I - Características do ambiente móvel

Dimension	**Values**
connection-state	connected, disconnected
bandwidth-rate	high, medium, low
communication-cost	free, cheap, expensive
available-battery	full, half, low
available-cache	full, half, low
available-persistent-memory	full, half, low
processing-capacity	high, medium, low

Uma Alternativa de Execução (EA) contém um conjunto de transações componentes que podem ser do tipo flat, distribuídas ou aninhadas fechadas. Transações *compensantes* podem ser associadas às transações componentes de forma que possam ser executadas em caso de falhas para recuperar o estado consistente da base de dados. As transações AMT e as EAs podem ser definidas como unidades de coordenação onde o acesso a dados é feito somente por transações componentes.

O modelo AMT utiliza um protocolo de efetivação chamado CO2PC que é uma combinação entre as abordagens otimista e pessimista (2PC[7]). Este protocolo permite que transações componentes compensáveis que executam em uma unidade móvel possam ser efetivadas localmente dentro da abordagem otimista enquanto que as transações não compensáveis só são efetivadas globalmente juntamente com a efetivação da EA.

4.6 TSenvironment

TSenvironment [13, 14, 15] é um ambiente de execução de transações voltado para ambientes dinâmicos e para dar apoio aos diversos tipos de requisitos exigidos por aplicações avançadas. Um dos ambientes para os quais o TSenvironment se destina é o ambiente de computação móvel. O TSenvironment é composto por diversos serviços de transações diferentes que podem ser usados concorrentemente por aplicações.

No TSenvironment, um serviço de transação é definido como um componente de software que pode ser independentemente desenvolvido e adicionado ao TSenvironment. Desta

[7]Two-Phase Commit

forma, serviços de transação também podem ser implementados ou modificados para atender requisitos específicos de aplicações. Um serviço de transação pode ser composto por um conjunto de componentes menores que podem ser agrupados para formar um serviço de transação completo e consistente. Cada um destes componentes menores que compõem um serviço de transação trata de uma tarefa bem definida como, por exemplo, processo de efetivação, recuperação, e gerenciamento de trancas[8].

Antes da execução de uma nova transação, um serviço de transação apropriado deve ser escolhido. Esta escolha pode depender de informações como, as propriedades da transação requeridas pelo usuário e os recursos disponíveis no ambiente. Baseado em informações como estas, será determinado automaticamente ou com a ajuda do usuário o serviço de transação adequado dentre os serviços de transações existentes no TSenvironment.

4.7 Moflex

Moflex [17] é um modelo que visa prover gerenciamento da mobilidade, heterogeneidade e flexibilidade na definição e execução de transações para o ambiente de computação móvel. Este modelo se propõe a permitir que transações possam ser submetidas mesmo enquanto o usuário se move entre diferentes células.

Uma transação Moflex é definida como uma coleção de subtransações que podem estar inter-relacionadas por um conjunto de dependências de execução. Podem ser definidos três tipos de dependências entre subtransações, dependências de sucesso (*ds*), dependências de falhas (*df*) e dependências externas (*de*). Uma transação `A` que possui uma *ds* em relação a uma transação `B` só pode ser executada se `B` tiver sido efetivada com sucesso. Se `A` possui uma *df* em relação a `B`, `A` só pode ser executada somente depois que `B` falhar. Quando uma transação possui uma *de*, ela só pode executar quando um predicado externo (tempo, custo ou localização) for satisfeito.

Uma transação Moflex pode conter regras de mudança de célula. Estas regras definem as políticas de execução de uma subtransação quando há mudança de célula em decorrência da movimentação da unidade móvel. Podem ser definidas as seguintes regras para a mudança de célula de uma transação: `continue(ti)` - a transação `ti` continua a sua execução na nova célula; `restart(ti)` - a transação `ti` é abortada na célula anterior e reiniciada na nova célula; `resume(ti)` - a transação `ti` é dividida em duas subtransações onde a primeira subtransação é efetivada na célula anterior e a segunda subtransação continua a execução na nova célula; `split_restart(ti)` - a transação `ti` é dividida em duas subtransações onde a primeira subtransação é efetivada na célula anterior e a segunda subtransação é reiniciada na nova célula.

As transações que executam com as regras `split-resume(ti)` ou `split_restart(ti)`, ou seja, que executam a operação de divisão da transação em duas subtransações, devem também possuir uma regra de junção que é usada para validar se a execução concatenada das subtransações resultantes é computacionalmente equivalente à execução de `ti`.

[8] Locks

4.8 HiCoMo

HiCoMo [19] é um modelo de transações voltado a aplicações de tomada de decisão que acessam dados do tipo estatístico ou agregados. O ambiente é composto por tabelas base na rede fixa e agregados de dados das tabelas base (*data warehouse*) nas unidades móveis. Este modelo se propõe a manter altas taxas de efetivação global de transações que acessam estes dados agregados durante os períodos de desconexão. Para isto, o modelo impõe restrições como: *(i)* os dados agregados a serem armazenados nas unidades móveis só devem conter valores como média, soma, mínimo e máximo; *(ii)* uma transação HiCoMo só pode realizar operações comutativas (adição e subtração) sobre os dados agregados; *(iii)* deve ser especificada uma margem de erro para a execução de uma transação HiCoMo.

O processamento de uma transação HiCoMo que executou sobre os dados agregados durante a desconexão é transferido para as tabelas base quando ocorre a reconexão. Para isto, transações base são criadas e executadas na rede fixa para criar o efeito sobre as tabelas base que foi realizado durante a desconexão. O modelo utiliza o seguinte algoritmo de transformação:

- **Passo 1.** Detecção de conflitos: verifica-se se há conflitos entre as transações HiCoMo e as transações base que executaram na rede fixa. Se algum conflito for detectado, a transação HiCoMo é abortada. Esta detecção de conflitos é implementada por um controle de concorrência otimista baseado em *timestamps*.

- **Passo 2.** Criação das transações base: se não existem conflitos, é criada uma transação base para cada tabela base que foi usada na formação dos dados agregados. Uma vez criadas, estas transações base são executadas como subtransações de uma transação aninhada estendida e aplicadas sobre as tabelas base.

- **Passo 3.** Cancelamento[9] de subtransações bases: quando submetidas à execução, algumas subtransações geradas podem abortar devido a regras de manutenção de integridade das tabelas base. Se a diferença que as subtransações abortadas criam estiver dentro da margem de erro, pode ser considerado que o processo de execução das subtransações obteve sucesso. Senão, a estratégia de geração de subtransações pode ser mudada e estas reexecutadas. Se a transformação não resultar em sucesso em algum ponto, a transação HiCoMo tem que ser abortada.

4.9 MobileTrans

MobileTrans [31] é um modelo que provê características de adaptabilidade e flexibilidade a transações que executam em ambientes de computação móvel. Com o MobileTrans, uma transação pode se adaptar em tempo de execução como um meio de lidar com cenários específicos do ambiente e as necessidades da aplicação. Para isso, o desenvolvedor deve especificar e prover a política de adaptação da transação. A política de adaptação consiste em associar um conjunto de atributos que irão determinar o comportamento da transação como, a especificação dos requisitos de consistência e atomicidade, se é para realizar caching

[9] Abort

de dados, de como os objetos serão copiados para a cache e como as falhas são tratadas. Os atributos que podem ser definidos são os seguintes:

Consistência - é possível especificar regras de consistência que permite o uso de versões de objetos desatualizadas se o nodo remoto correspondente não estiver disponível.

Busca [10] - a política descreve quais objetos devem ser copiados para a cache antes da execução da transação ou se os objetos devem ser copiados sob demanda no decorrer da execução da transação.

Delegação - é possível delegar a responsabilidade de efetivação de uma transação para outra transação.

Atomicidade - a política pode especificar se a transação pode ser efetivada mesmo se nem todos os nodos envolvidos estão acessíveis, ou seja, se algumas atualizações podem ser descartadas.

Caching - é possível especificar que os objetos copiados para a cache e os objetos modificados por transações locais serão armazenados de forma que possam ser acessados durante os períodos de desconexão.

Tratamento de falhas - a política é também responsável por determinar o que deve ser feito quando as condições especificadas de consistência, busca e atomicidade não puderem ser mantidas (devido ao estado da rede). A política também pode ser responsável por mudar a configuração da transação em tempo de execução como um meio de tratamento da falha ocorrida.

4.10 PTM

PTM [21, 22] é um modelo de transação para a computação móvel que visa aumentar a concorrência no acesso a dados e evitar a necessidade de desfazer ou compensar transações. Basicamente, o modelo permite que a transação execute mais dois tipos de operação além das tradicionais leitura e escrita: a pré-leitura e a pré-escrita. A operação de pré-escrita pode ser usada por uma transação quando ela desejar tornar visíveis a outras transações os valores dos objetos modificados pela mesma antes mesmo de ser efetivada. Estes valores serão visíveis a outras transações tanto na unidade móvel quanto na rede fixa assim que for executado um processo chamado pré-efetivação.

Enquanto a operação de leitura acessa valores de uma operação de escrita, a operação de pré-leitura acessa valores de uma operação de pré-escrita. Uma transação só pode ser pré-efetivada quando ela executar todas as operações de leitura, pré-leitura e pré-escrita desejadas. Depois que uma transação é pré-efetivada ela não pode mais ser abortada. Os valores escritos pelas operações de pré-escrita podem representar o valor exato ou uma versão abstrata do valor que o objeto modificado deverá conter depois da fase de efetivação final da transação.

[10] Fetching

4.11 Two-tier

Two-tier [11] é um modelo de replicação de dados para ambientes de computação móvel que possui como objetivos *(i)* prover uma alta disponibilidade de dados às aplicações que executam em unidades móveis, *(ii)* permitir a execução de transações em unidades móveis desconectadas, *(iii)* prover serialização na execução de transações e *(iv)* garantir que os resultados de transações sobre réplicas de um mesmo dado sejam convergidos em um único valor consistente.

O modelo de replicação Two-tier assume que o ambiente de execução é composto por estações base na rede fixa que se encontram sempre conectadas e que armazenam a maior parte dos objetos mestres da base de dados e por unidades móveis que podem se encontrar desconectadas na maior parte do tempo e que armazenam réplicas da base de dados. Cada objeto replicado em uma unidade móvel possui duas versões, a primeira versão é a chamada versão mestra e representa o valor mais recente recebido do objeto mestre, e a segunda é chamada versão de tentativa que será a versão sobre a qual as transações de tentativa executarão suas operações.

O modelo Two-tier considera que existem dois tipos de transações, as transações base e as transações de tentativa. Transações base executam suas operações somente sobre objetos mestres. Já as transações de tentativa executam suas operações somente sobre as versões de tentativa de uma unidade móvel. As transações de tentativa permitem a execução de transações em unidades móveis desconectadas. Na reconexão, as transações de tentativa produzem uma transação base que será responsável por executar operações correspondentes sobre as réplicas mestras nas estações bases.

Cada transação base gerada por uma transação de tentativa possui um critério de aceitação. Este critério possui o objetivo de verificar se as diferenças nos resultados de uma transação base em relação aos resultados da transação tentativa correspondente são aceitáveis ou não. Se a transação base falhar ou não atender o seu critério de aceitação, ela será abortada e é enviada uma mensagem para a unidade móvel esclarecendo o motivo. Se uma transação base é abortada, o usuário poderá revisá-la e submetê-la novamente.

4.12 Mobisnap

Mobisnap [29] é um modelo de gerenciamento de transações móveis que permite a execução de transações em unidades móveis desconectadas. O ambiente para o qual este modelo foi desenvolvido é composto por *(i)* servidores sempre conectados a uma rede fixa que armazena a cópia mestra de todos os itens de dados e *(ii)* unidades móveis que podem armazenar réplicas dos dados da rede fixa.

Assim como no modelo Two-tier, uma unidade móvel armazena duas cópias de cada item de dado que foi replicado: uma versão efetivada e uma versão de tentativa. A versão efetivada reflete a última versão que foi copiada da versão mestra da rede fixa. A versão de tentativa é baseada na versão efetivada, porém ela reflete as operações das transações móveis que executam na unidade móvel. Uma transação que executa na unidade móvel pode acessar ambas as versões sendo que a versão de tentativa representa a evolução esperada do item de dado correspondente da rede fixa.

Um aspecto relevante do modelo Mobisnap é que ele disponibiliza um mecanismo que permite a reserva prévia de itens de dados da rede fixa que serão usados por transações que executarão em uma unidade móvel desconectada. O modelo Mobisnap define quatro tipos de reserva:

1. **Escrow** - usado para dividir um recurso que seja divisível. Por exemplo, pode-se dividir as unidades de um lote de um determinado produto em várias partes sendo que cada parte seria reservada a um vendedor.

2. **Slot** - usado para reservar o direito de inserir um novo registro com valores pré-definidos. Por exemplo, pode-se reservar o direito de agendar um encontro em uma sala em um determinado horário.

3. **Value-change** - usado para reservar o direito de mudar o valor de itens de dados de uma base de dados. Por exemplo, reservar o direito de mudar a descrição de um produto.

4. **Value-use** - usado para reservar o direito de uso de itens de dados. Por exemplo, reserva-se o direito de vender um produto por um determinado preço mesmo que este seja atualizado.

Cada reserva de recurso feita na unidade móvel possui um prazo de expiração. Isto permite que o sistema use itens reservados por uma transação desconectada que não entrou em contato com o servidor para efetivar as operações. Enquanto o prazo de uma reserva não expira, nenhuma transação pode utilizar o item de dado reservado. Desta forma, se uma transação só acessa valores que foram previamente reservados, o seu resultado pode ser corretamente estabelecido na própria unidade móvel.

As transações que executaram com sucesso na unidade móvel são posteriormente propagadas para o servidor. Quando o servidor recebe uma transação móvel ele executa o seu programa transacional de forma a efetivar a execução da transação atualizando os dados do servidor. Nesta fase, uma transação móvel poderá ser abortada se houver casos como, detecção de conflito com outras transações sobre itens de dados que não foram previamente reservados ou que tiveram a reserva expirada.

4.13 Gold Rush

Gold Rush [3] é um middleware que apóia aplicações desenvolvidas em Java que residem em unidades móveis com conexão intermitente e acessam uma base de dados em um servidor centralizado. Gold Rush provê um mecanismo de cache que permite a transações acessar cópias dos dados remotos em cache durante o período de desconexão.

Enquanto a unidade móvel está conectada, os dados da base de dados centralizada necessários para a execução das transações são copiados para a cache da unidade móvel em preparação para uma subseqüente desconexão. Durante a desconexão, uma transação que reside na unidade móvel pode realizar suas operações sobre as cópias dos dados em cache. No modo desconectado, os dados da base de dados centralizada não recebem nenhum tipo de tranca (*lock*), ou seja, o controle de concorrência aplicado é do tipo otimista. Os

locks só são atribuídos às cópias locais de acordo com o tipo de operação da transação. Quando a transação entra na fase de efetivação na unidade móvel, todos os dados que foram modificados pela mesma são gravados, um registro com informações da transação é criado e armazenado em um arquivo de *log* para posterior reexecução da mesma no servidor.

Assim que a unidade móvel reconecta, o log da transação é reexecutado no servidor como uma tentativa de efetivar as operações da transação sobre a base de dados centralizada. Nesta fase, a transação é submetida a um processo de detecção de conflitos. Se a transação passar na detecção de conflitos ela é considerada efetivada. O mecanismo utilizado para a detecção de conflitos é o *timestamp*.

Para reduzir o volume de tráfego de dados entre a unidade móvel e o servidor, o Gold Rush tenta otimizar a transferência de dados evitando que dados da cache ainda considerados consistentes sejam retransmitidos do servidor.

4.14 Pre-serialization

Pre-serialization [5, 6] é uma técnica de gerenciamento de transações móveis sobre um sistema de bases de dados múltiplas (SBDM). Esta técnica permite que uma transação móvel estabeleça a sua ordem de serialização antes mesmo de chegar à fase de efetivação. Neste modelo, uma transação que executa na unidade móvel (chamada transação global) é composta por um conjunto de subtransações compensáveis na rede fixa. Cada subtransação engloba as operações da transação global realizadas sobre uma determinada base de dados do SBDM. Como todas as subtransações locais têm que ser compensáveis, elas são efetivadas antes mesmo que a transação global tome a decisão de entrar na fase de efetivação. Isto permite que os recursos alocados para a execução da transação global possam ser liberados paulatinamente de acordo com a efetivação da execução de suas subtransações.

Em cada estação de suporte do cenário para o qual a técnica é proposta existe um coordenador de transações globais (GTC) responsável por submeter as subtransações aos servidores da rede fixa, gerenciar a migração da unidade móvel e a desconexão e verificar as propriedades de isolamento e atomicidade.

Quando uma unidade móvel migra de uma estação de suporte para outra, o GTC da primeira estação se encarrega de enviar à segunda estação as informações necessárias para que a transação global possa continuar a sua execução.

Quando uma unidade móvel é desconectada, a transação global não é totalmente interrompida, ou seja, as subtransações que já tiverem sido submetidas continuam a executar na rede fixa. Neste período, todas respostas geradas pelas transações locais são inseridas em uma lista que é propagada para a transação global quando ocorre a reconexão. Para evitar que os recursos de uma transação global T_x que não voltou a se reconectar fiquem indefinidamente alocados, quando outra transação T_y deseja acessar tais recursos, estes são automaticamente liberados para o uso de T_y e T_x pode ser considerada abortada.

Para verificar a garantia das propriedades de atomicidade e isolamento da transação global, o GTC, em determinados períodos, executa um algoritmo de serialização global em grafos (PGSG). A atomicidade requer que todas subtransações sejam efetivadas ou que cada uma delas seja abortada ou compensada (atomicidade semântica). A propriedade de isolamento requer que duas transações globais conflitantes sejam serializadas na mesma

ordem em cada servidor onde suas subtransações executam.

4.15 Semantics-based

Esse trabalho [35] usa informação semântica de objetos para facilitar a execução de transações em unidades móveis sujeitas a desconexões. A informação semântica sobre objetos é usada para prover controle de concorrência e caching com uma granularidade mais fina permitindo uma manipulação assíncrona dos objetos em cache e efetivação unilateral de transações que executam na unidade móvel.

A idéia básica é dividir objetos maiores em fragmentos menores e do mesmo tipo que o objeto maior através do conhecimento sobre o objeto. Exemplos de objetos considerados fragmentáveis são: itens agregados, pilhas, filas e conjuntos. Desta forma, uma unidade móvel pode copiar para a sua cache um fragmento de objeto de acordo com as necessidades da transação que executará no modo desconectado minimizando o espaço em disco requerido na unidade móvel para o seu armazenamento. Cada fragmento de um objeto é considerado uma unidade de consistência.

Uma requisição de cache feita por uma unidade móvel deve incluir dois parâmetros: critério de seleção e condições de consistência. O critério de seleção especifica o objeto requisitado e o tamanho requerido para o fragmento do objeto. As condições de consistência especificam as regras que precisam ser satisfeitas para manter a consistência do objeto como um todo.

Quando um fragmento de um objeto é copiado para a cache de uma unidade móvel, esta ganha o direito exclusivo de operar sobre ele, ou seja, nenhuma outra transação poderá acessar este mesmo fragmento enquanto ele não for reintegrado pela unidade móvel que o requisitou. Entretanto os demais fragmentos do objeto podem ser usados por outras transações.

Ao se reconectar, a unidade móvel inicia um processo de reconciliação dos fragmentos de objetos em cache com os objetos do servidor da base de dados.

4.16 Clustering

Neste modelo [25, 27] dados geograficamente ou semanticamente relacionados podem ser agrupados. Cópias de dados em um mesmo agrupamento[11] têm que manter uma total consistência mútua enquanto que cópias de dados em agrupamentos diferentes toleram um certo grau de inconsistência. Agrupamentos podem ser configurados dinamicamente de acordo com as variações da conectividade ou dos demais recursos do ambiente: o grau de inconsistência entre agrupamentos pode ser mudado; agrupamentos podem ser unidos ou divididos.

Agrupamentos de dados podem ser usados para manter a autonomia de máquinas móveis e aumentar a disponibilidade de dados durante as desconexões. Por exemplo, os dados de uma máquina móvel podem ser considerados um agrupamento que, por sua vez, podem ser acessados durante o período de desconexão.

[11] Cluster

Neste modelo, a interface oferecida pelo gerenciador de dados provê, além das operações de leitura e escrita tradicionais (chamadas operações fortes), operações com garantia de consistência mais fraca: leitura fraca e escrita fraca. Estas operações fracas permitem acessar dados disponíveis localmente em um agrupamento. Especificamente, leituras fracas acessam dados dentro de um certo grau de inconsistência e escritas fracas fazem atualizações condicionais, ou seja, que podem se tornar definitivas ou não.

São providos dois tipos de transações, as fracas e as fortes. Uma transação forte é aquela que não inclui nenhuma operação fraca. Por incluir somente operações fortes, as transações fortes são consideradas atômicas, consistentes, isoladas e duráveis. Uma transação fraca inclui somente operações fracas, ou seja, acessa dados locais do agrupamento. Uma transação fraca pode ser efetivada localmente em um agrupamento. Depois da efetivação, as atualizações feitas neste cluster se tornam visíveis a outras transações fracas. Transações fracas que são efetivadas localmente estão sujeitas a um processo de reconciliação de forma que sejam efetivadas globalmente.

Para verificar se uma transação fraca pode ser efetivada globalmente e manter a consistência entre os agrupamentos envolvidos, o processo de reconciliação verifica se as escritas fracas executadas conflitam com transações fortes. Se o conflito existir, a transação fraca envolvida é desfeita. Como uma transação fraca pode acessar dados escritos por outras transações fracas localmente efetivadas, isto pode resultar em um cancelamento em cascata de transações.

5 Sumário e Discussão

5.1 Obstáculos do Ambiente

A Tabela II resume as estratégias utilizadas por cada trabalho revisado para superar obstáculos do ambiente de computação móvel. Os obstáculos observados foram: desconexões, variações da largura de banda, limitações dos suprimentos de energia, escassez de espaço em disco, mobilidade (mudança de célula).

Tabela II - Sumário das estratégias para superação de obstáculos

Modelo	DESCONEXÃO	L. DE BANDA	ENERGIA	DISCO	MOBILIDADE
Coda-IOT	Caching de dados; efetivação local de transações	Reintegração por partes; validação rápida de dados em cache	Não especificado	Não especificado	Não especificado
PRO-MOTION	Caching de dados; efetivação local de transações	Transmissão das partes dos dados em falta ou desatualizados	Não especificado	Transferência de partes de dados para a cache; otimização do log de recuperação	Não especificado
Reporting	A parte da computação da unidade móvel pode continuar a executar	Não especificado	Divisão da computação entre máquinas fixas e máquinas móveis	Divisão da computação entre máquinas fixas e máquinas móveis	Não especificado
Kangaroo	Transações interrompidas por uma desconexão podem continuam a executar depois da reconexão	Não especificado	Desligar a unidade móvel permitindo a continuação da execução das transações quando esta for religada	Não especificado	Divide a execução da transação para que esta possa continuar a executar na nova célula

continuação na próxima página

Tabela II - Sumário das estratégias para superação de obstáculos

Modelo	DESCONEXÃO	L. DE BANDA	ENERGIA	DISCO	MOBILIDADE
AMT	Permite que a transação decida qual alternativa deve ser executada	Permite que a transação decida qual alternativa deve ser executada	Permite que a transação decida qual alternativa deve ser executada	Permite que a transação decida qual alternativa deve ser executada	Não especificado
TSenvironment	Permite a implementação de serviços personalizados	Permite a implementação de serviços personalizados	Permite a implementação de serviços personalizados	Permite a implementação de serviços personalizados	Permite a implementação de serviços personalizados
Moflex	Permite criação de dependências entre transações	Permite criação de dependências entre transações	Permite criação de dependências entre transações	Permite criação de dependências entre transações	Permite criação de dependências entre transações
HiCoMo	Caching de dados; restrições no acesso a dados em cache	Não especificado	Não especificado	Não especificado	Não especificado
MobileTrans	Caching de dados	A transação pode decidir quais serão as medidas de adaptação	A transação pode decidir quais serão as medidas de adaptação	Não especificado	Não especificado
PTM	Caching de dados	Transfere para cache somente as partes dos dados a serem usados até a pré-efetivação	Parte da computação é transferida para a rede fixa	Transfere para cache somente a parte dos dados a serem usados até a pré-efetivação	Divide a execução da transação para que esta possa continuar a executar na nova célula
Two-tier	Caching de dados	Não especificado	Não especificado	Não especificado	Não especificado
Mobisnap	Caching de dados; reserva de acesso a dados com prazo de expiração	Não especificado	Não especificado	Não especificado	Transferência de informações de uma estação de apoio para outra necessárias para a continuação da transação
Gold Rush	Caching de dados	Cópia antecipada de dados para a cache e de reexecução de transações na rede fixa	Operar no modo desconectado para evitar gastos de energia com transmissão de dados	Não especificado	Não especificado
Pre-serialization	As operações são sempre realizadas na rede fixa; as subtransações que já tiverem sido submetidas na rede fixa continuam a executar	Não especificado	Não especificado	Não especificado	Não especificado
Semantics-based	Caching de dados	Transferir somente os fragmentos de objetos necessários	Operar no modo desconectado para evitar gastos de energia com transmissão de dados	Armazenar na cache somente fragmentos de objetos necessários durante a desconexão	Não especificado
Clustering	Caching de dados; agrupamento de dados	Definição dinâmica de níveis de consistência entre clusters	Não especificado	Não especificado	Criação e configuração dinâmica de agrupamentos de dados geograficamente relacionados.

5.1.1 Desconexões

Para lidar com as desconexões, a maioria dos trabalhos revisados utiliza técnicas de caching de dados, ou seja, transferência de dados para a cache da unidade móvel. Os dados em cache podem ser acessados por transações que executam na unidade móvel mesmo durante os períodos de desconexão.

Os modelos Kangaroo e Pre-serialization não utilizam caching de dados, pois, apesar de as transações serem submetidas a partir das unidades móveis, as suas operações são executadas diretamente na rede fixa. No modelo Kangaroo, quando ocorre uma desconexão, as transações são interrompidas, mas podem dar continuidade às suas execuções quando a conexão é refeita. Já no modelo Pre-serialization, as subtransações que já tiverem sido submetidas à rede fixa podem continuar a executar. As subtransações que ainda não tiverem

sido submetidas só poderão executar depois da reconexão.

5.1.2 Largura de banda baixa ou variável

Para a superar a largura de banda baixa ou variável do canal de comunicação entre as unidades móveis e a rede fixa, Coda-IOT e Gold Rush buscam aproveitar os períodos em que a largura de banda está alta para realizar os processos de transferência de dados. O Coda utiliza uma técnica chamada reintegração por partes que utiliza a largura de banda disponível de forma flexível para transferir os dados atualizados da unidade móvel para a rede fixa. O Gold Rush utiliza cópia antecipada de dados para a cache que serão necessários para a execução de transações. Estes dados podem ser copiados para a cache quando há uma maior disponibilidade de largura de banda.

Os modelos PRO-MOTION, PTM e Semantic-based buscam evitar a transferência desnecessária de dados e assim reduzir a quantidade de bytes a serem transferidos pelo canal de comunicação. O PRO-MOTION transfere para a cache somente as partes dos dados ou as partes dos métodos que estão em falta na cache da unidade móvel. O PTM transfere para a cache somente os dados que serão usados por transações até a fase da pré-efetivação. O modelo Semantic-based utiliza informações semânticas dos objetos para dividi-los em fragmentos menores. Somente os fragmentos que serão acessados por uma transação é que são transferidos para a cache.

5.1.3 Energia

Para lidar com as limitações dos suprimentos de energia das unidades móveis, os trabalhos revisados normalmente utilizam meios para reduzir o consumo de energia. Os modelos Reporting e PTM propõem a redução na carga de computação da unidade móvel permitindo a divisão da computação de transações entre a unidade móvel e a rede fixa. Outros trabalhos como, Gold Rush e Semantics-based, ao considerarem que o uso dos dispositivos de comunicação sem fio consome energia, propõem a execução de transações no modo desconectado como meio de redução deste consumo. Já o modelo Kangaroo propõe o desligamento da unidade móvel em períodos de ociosidade para economizar energia.

5.1.4 Espaço em disco

Para lidar com a escassez de espaço em disco das unidades móveis, parte trabalhos revisados procura reduzir a quantidade de dados trazidos para a cache destas unidades. O modelo PTM transfere para a cache da unidade móvel somente os dados que serão usados até a fase de pré-efetivação das transações. Já o modelo Semantics-based utiliza informações semânticas dos objetos para dividi-los em fragmentos menores e transfere para a cache somente aqueles fragmentos que serão necessários na execução de transações. O modelo PRO-MOTION, além de transferir para a cache somente a parte necessária dos dados, otimiza o log de recuperação de transações para economizar espaço em disco. O modelo Reporting propõe a divisão da computação de uma transação entre a unidade móvel e a rede fixa. Desta forma, os dados que fossem acessados somente pela parte da computação da rede fixa não precisariam ser trazidos para a cache.

5.1.5 Mobilidade

Para lidar com o processo de mudança de mudança de célula ocasionado pela mobilidade das unidades móveis, os trabalhos revisados adotaram diferentes estratégias. Quando ocorre mudança de célula, tanto o modelo Kangaroo quanto o PTM divide a transação em execução em duas transações. A primeira parte da transação é efetivada na célula anterior e a segunda parte continua a execução na nova célula.

No modelo Mobisnap, quando ocorre mudança de célula, a estação de apoio da célula anterior envia para a nova célula os dados necessários para dar continuidade às transações em execução.

No modelo Clustering é afirmado que a capacidade de configuração dinâmica dos agrupamentos de dados pode ser útil no processo de mudança de célula. Agrupamentos em diferentes células podem ser dinamicamente unidos em um único agrupamento para que os dados acessados por uma transação na célula anterior se tornem consistentes em relação aos dados da nova célula.

5.2 Flexibilidade das Propriedades ACID

A Tabela III apresenta um resumo dos trabalhos revisados em relação à flexibilidade das propriedades ACID. Para que um modelo de transações possa atender aos requisitos de aplicações avançadas, um dos seus requisitos seria prover flexibilidade destas propriedades.

Tabela III - Flexibilidade das propriedades ACID

Modelo	ATOMICIDADE	CONSISTÊNCIA	ISOLAMENTO	DURABILIDADE
Coda-IOT	Sem flexibilidade	Sem flexibilidade	Liberação de resultados parciais antes da efetivação global	Sem garantia de durabilidade
PRO-MOTION	Transações aninhadas	Regras de consistência para acesso a dados	Provê dez níveis de isolamento	Sem garantia de durabilidade
Reporting	Transações aninhadas	Não especificado	Permite liberação de resultados parciais durante a execução	Não especificado
Kangaroo	Divisão de transações no processo de mudança de célula	Não Especificado	Sem garantia de isolamento	Não especificado
AMT	Transações aninhadas	Não especificado	Liberação de resultados parciais antes da efetivação global	Não especificado
TSenvironment	Flexibilidade não especificada, porém há possibilidades desta ser implementada	Flexibilidade não especificada, porém há possibilidades desta ser implementada	Flexibilidade não especificada, porém há possibilidades desta ser implementada	Flexibilidade não especificada, porém há possibilidades desta ser implementada
Moflex	Divisão de transações no processo de mudança de célula	Regras de junção de subtransações que definem a corretude de uma transação	Sem garantia de isolamento	Não especificado
HiCoMo	Divisão de transações no processo de mudança de célula	Pode-se especificar uma margem de erro para execução de transações	Liberação de resultados parciais antes da efetivação global	Tolera o descarte de atualizações dentro de uma margem de erro
MobileTrans	Permite efetivação de transações mesmo se determinados objetos não puderem ser efetivados	Regras de consistência para acesso a dados	Permite a leitura de dados não atualizados	Tolera a efetivação de transações mesmo se todas as atualizações não puderem ser propagadas para a rede fixa
PTM	Transações aninhadas; divisão de transações no processo de mudança de célula	Não especificado	Liberação de resultados parciais antes da efetivação global	Sem garantia de durabilidade

continuação na próxima página

Tabela III - Flexibilidade das propriedades ACID

Modelo	ATOMICIDADE	CONSISTÊNCIA	ISOLAMENTO	DURABILIDADE
Two-tier	Sem flexibilidade	Pode-se definir um critério de aceitação para a execução de transações	Liberação de resultados parciais antes da efetivação global	Sem garantia de durabilidade
Mobisnap	Sem flexibilidade	Utiliza informações semânticas dos dados acessados para manter a consistência dos mesmos nos períodos de desconexão	Sem flexibilidade	Não especificado
Gold Rush	Sem flexibilidade	Sem flexibilidade	Permite a leitura de dados não atualizados	Sem flexibilidade
Pre-serialization	Sem flexibilidade	Sem flexibilidade	Liberação de resultados parciais antes da efetivação global	Sem flexibilidade
Semantics-based	Sem flexibilidade	Regras de consistência para acesso a dados; informações semânticas para manter consistência durante a desconexão	Liberação de resultados parciais antes da efetivação global	Sem garantia de durabilidade
Clustering	Sem flexibilidade	Uso de informação semântica para definição de níveis de inconsistência entre agrupamentos	Definição de transações fortes e fracas. Transações fracas liberam resultados parciais antes da efetivação global.	Sem garantia de durabilidade para transações fracas

5.2.1 Atomicidade

Uma das formas mais comuns de se flexibilizar a atomicidade é através do apoio a transações aninhadas. Transações aninhadas são compostas por subtransações que podem ser abortadas unilateralmente, ou seja, sem a necessidade de abortar a transação como um todo.

Dos trabalhos revisados, os que provêem apoio a transações aninhadas são: PRO-MOTION, Reporting, AMT e PTM. Outros trabalhos como Kangaroo, Moflex, HiCoMo e PTM, apesar de não proverem apoio a transações aninhadas, utilizam a divisão de transações em subtransações, a chamada operação **split**[28] . Esse processo de divisão de uma transação em subtransações também pode ser considerado uma forma de flexibilização da propriedade da atomicidade utilizado por estes modelos para lidar com o processo de mudança de célula.

Os trabalhos Coda-IOT, Two-tier, Mobisnap, Gold Rush, Pre-serialization, Semantics-based e Clustering não provêem flexibilização da atomicidade.

5.2.2 Consistência

São usadas diferentes formas de flexibilização dentre os trabalhos revisados que provêem flexibilização da consistência. PRO-MOTION, MobileTrans e Semantics-based flexibilizam a consistência permitindo que sejam definidas regras específicas de acesso a dados que definem a consistência da base de dados. Já trabalhos como Moflex, HiCoMo e Two-tier, flexibilizam a consistência permitindo a definição de regras que dizem se uma transação executou corretamente ou não. Os modelos Mobisnap e Clustering utilizam informações semânticas dos dados acessados para definir a consistência dos mesmos.

5.2.3 Isolamento

No ambiente de computação móvel, em linhas gerais, o processo de efetivação de uma transação possui dois níveis, a efetivação local e a efetivação global. A efetivação local

ocorre sobre os dados da cache da unidade móvel. A efetivação global ocorre depois da efetivação local e é responsável por propagar as atualizações realizadas localmente para os servidores da rede fixa. A forma mais comum de se flexibilizar a propriedade de isolamento usada pelos modelos revisados é a liberação os resultados da transação antes mesmo da efetivação global. Isto ajuda a melhorar a concorrência e a disponibilidade de dados a transações em unidades móveis desconectadas.

Os modelos Coda-IOT, Reporting, HiCoMo, PTM, Pre-serialization, Semantics-based e Clustering flexibilizam o isolamento permitindo a liberação de resultados parciais antes do processo de efetivação global. Os modelos MobileTrans e Gold Rush também flexibilizam o isolamento permitindo que transações acessem dados que não estão atualizados. Já o modelo PRO-MOTION, para flexibilizar o isolamento, provê dez níveis de isolamento. Cada nível de isolamento define o grau em que uma transação, no decorrer de usa execução, concorda em compartilhar resultados parciais com outras transações.

5.2.4 Durabilidade

Durabilidade é uma propriedade que visa garantir a persistência dos resultados de transações efetivadas. No ambiente de computação móvel isso implica em tornar durável nos servidores da rede fixa os resultados de transações que executaram em unidades móveis. Devido às características do ambiente móvel como, desconexões e outras falhas de comunicação, esta tarefa nem sempre é fácil de ser realizada. Alguns dos modelos revisados como HiCoMo e MobileTrans flexibiliza a durabilidade permitindo que transações possam ser efetivadas mesmo quando alguns dos seus resultados não puderem se tornar persistentes.

5.3 Aspectos relevantes e limitações

Nesta seção serão apresentados alguns aspectos relevantes dos modelos revisados bem como algumas de suas limitações.

5.3.1 Coda-IOT

Principais aspectos:

- Foi um dos projetos pioneiros ao utilizar o conceito de operação desconectada como um meio de aumentar a disponibilidade de dados durante os períodos de desconexão.

- Utiliza mecanismos que avaliam o estado do ambiente (por exemplo, largura de banda) e busca adequar o sistema a este estado. Um destes mecanismos é a chamada reintegração por partes que utiliza a largura de banda de forma flexível na propagação dos dados em cache para os servidores.

Limitações:

- Dentre as propriedade ACID, o modelo de transações IOT só provê apoio à propriedade de isolamento.

- O Coda provê somente mecanismos de adaptação transparente, ou seja, as medidas de adaptação tomadas pelo Coda são refletidas em todas as aplicações apoiadas pelo mesmo.

5.3.2 PRO-MOTION

Principais aspectos:

- Provê dez níveis de isolamento que podem ser usados como mecanismo de flexibilização da propriedade do isolamento. Isto permite a escolha dos níveis de isolamento adequados para os requisitos de cada aplicação.
- Flexibiliza a atomicidade através do apoio a transações aninhadas.

Limitações:

- Os mecanismos de adaptação do PROMOTION são centralizados nos compacts, ou seja, a reconfiguração de um compact pode afetar todas as transações que o utilizam.
- Não permite que uma transação possa escolher a estratégia de adaptação mais adequada de acordo com o estado do ambiente móvel.

5.3.3 Reporting

Principais aspectos:

- Flexibiliza o isolamento permitindo que resultados parciais de uma transação possam ser delegados a outras transações. Esta delegação pode ocorrer em qualquer ponto da execução de uma transação.
- Flexibiliza a atomicidade provendo transações aninhadas.
- Permite a passagem do controle de rotinas de uma transação para outra.

Limitações:

- Não trata aspectos relevantes do ambiente móvel como, desconexões e baixa largura de banda.

5.3.4 Kangaroo

Principais aspectos:

- Enfoca a questão da mobilidade de transações, permitindo que estas possam continuar a executar mesmo a partir de uma unidade móvel em movimento (durante os processos de mudança de célula).

Limitações:

- A estratégia utilizada pelo modelo para lidar com o processo de mudança de célula possui baixa flexibilidade e pode não ser adequado a alguns tipos de aplicações.
- Não prevê caching de dados, as operações de uma transação são sempre executadas na rede fixa. Desta forma, uma transação só pode dar continuidade a sua execução enquanto a unidade móvel estiver conectada.
- Não aborda a questão da baixa largura de banda e do possível alto custo da comunicação com a rede fixa.

5.3.5 AMT

Principais aspectos:

- Uma transação pode ser composta por um conjunto de alternativas dependentes do estado do ambiente. O sistema utiliza ciência das variações ocorridas nos recursos do ambiente para escolher a alternativa a ser executada.

Limitações:

- O conjunto de alternativas de uma transação é definida de forma estática, ou seja, as alternativas só são definidas no inicio da execução da transação. Não são providas medidas de adaptação para mudanças ocorridas durante a execução que não tenham sido previstas pelas alternativas.

5.3.6 TSenvironment

Principais aspectos:

- Permite a co-existência de diferentes serviços de transações. Antes da execução de uma transação, o serviço de transação mais adequado pode ser selecionado dentre os serviços existentes.
- Permite a implementação ou composição de novos serviços de transação de acordo com os requisitos de aplicações avançadas.

Limitações:

- Por enfocar aspectos que permitem a co-existência de diferentes serviços de transação, o modelo não apresenta mecanismos de adaptação às transações diante das mudanças que podem ocorrer no ambiente durante a execução das transações.

5.3.7 Moflex

Principais aspectos:

- Enfoca o aspecto da mobilidade de transações, permitindo definição de regras para mudança de célula.

- Permite a definição de dependências de execução entre transações.
- Uma transação pode ser configurada para executar somente quando um predicado externo for satisfeito (custo, tempo, localização).

Limitações:

- Não trata aspectos relevantes do ambiente móvel como, desconexões e baixa largura de banda.
- Não prevê caching de dados, as operações de uma transação são sempre executadas na rede fixa. Desta forma, uma transação só pode dar continuidade a sua execução enquanto a unidade móvel estiver conectada.

5.3.8 HiCoMo

Principais aspectos:

- Propicia altas taxas de sucesso na efetivação global de transações que acessam dados na cache de unidade móveis durante os períodos de desconexão.
- Permite a definição de uma margem de erro para a execução de transações.

Limitações:

- Restringe-se a domínios de aplicação muito restritos como, aplicações estatísticas, onde só há operações de soma e subtração.
- Não trata outros aspectos relevantes do ambiente móvel como, baixa largura de banda e escassez de recursos das unidades móveis.

5.3.9 MobileTrans

Principais aspectos:

- Pode ser aplicado ao ambiente ad-hoc permitindo interação direta entre dispositivos móveis.
- Permite a especificação de políticas de adaptação que definem o comportamento de transações diante do dinamismo do ambiente.

Limitações:

- As políticas de adaptação são definidas estaticamente e precisam descrever o comportamento de uma transação para cada tipo de variação que possa afetar a execução da mesma. Isto pode tornar definição de políticas uma tarefa complexa e estas também podem se tornar obsoletas quando os padrões de variação do ambiente sofrer alterações.
- A flexibilidade das propriedades ACID provida pode não ser suficiente para atender aos requisitos de aplicações avançadas que exigem o uso de, por exemplo, transações aninhadas e níveis de isolamento.

5.3.10 PTM

Principais aspectos:

- Permite a divulgação da versão final ou de uma versão abstrata de objetos antes da fase de efetivação final da transação. Esta é uma forma de aumentar a concorrência das transações e a disponibilidade dos dados diante de desconexões ocorridas depois da pré-efetivação.

Limitações:

- Voltado para aplicações que necessitam divulgar um esboço dos resultados antes da fase de efetivação final.
- Não provê mecanismos de adaptação com flexibilidade necessária para atender requisitos específicos de aplicações.

5.3.11 Two-tier

Principais aspectos:

- Permite a execução de transações sobre cópias em cache em unidades móveis desconectadas.
- Permite a definição de critérios de aceitação para as transações que executam sobre dados em cache.

Limitações:

- A flexibilidade das propriedades ACID provida pode não ser suficiente para atender aos requisitos de aplicações avançadas que exigem o uso de, por exemplo, transações aninhadas e níveis de isolamento.
- Não trata outros aspectos relevantes do ambiente móvel como, baixa largura de banda e escassez de recursos das unidades móveis.

5.3.12 Mobisnap

Principais aspectos:

- Provê um mecanismo de reserva de acesso a dados que dá uma maior garantia de efetivação das operações a serem executadas na unidade móvel desconectada.
- Permite a definição de prazo de expiração para os dados acessados em cache.

Limitações:

- Bloqueia temporariamente os itens de dados da rede fixa enquanto estão sendo acessados em unidade móveis reduzindo a concorrência.
- Não provê flexibilidade das propriedades ACID suficiente para atender requisitos de aplicações avançadas.

5.3.13 Gold Rush

Principais aspectos:

- Provê um mecanismo de caching que permite a execução de transações em unidade móveis desconectadas.

Limitações:

- Modelo simples, de baixa configurabilidade e de baixa flexibilidade das propriedades ACID.

5.3.14 Pre-serialization

Principais aspectos:

- Permite gerenciamento de transações móveis sobre sistemas de base de dados múltiplas.
- Permite que uma transação possa estabelecer a sua ordem de serialização antes de chegar à fase de efetivação.
- Permite a liberação de recursos alocados por uma transação paulatinamente.

Limitações:

- O modelo é voltado para sistemas de bases de dados múltiplas e não provê mecanismos de adaptação ao dinamismo do meio.
- Não prevê caching de dados. Com isso, as subtransações de uma transação global só executam no servidor.
- Possui baixa flexibilidade das propriedades ACID.

5.3.15 Semantics-based

Principais aspectos:

- Utiliza informação semântica de objetos para trazer para a cache da unidade móvel somente os fragmentos de objetos que são necessários para a execução de uma transação.
- Facilita reintegração dos dados após as reconexões.

Limitações:

- Sempre bloqueia na rede fixa os fragmentos que estão sendo usados nas unidade móveis. Isto pode reduzir a concorrência dos objetos da rede fixa.
- Não provê flexibilidade das propriedades ACID suficiente para atender requisitos de aplicações avançadas.

5.3.16 Clustering

Principais aspectos:

- Permite a configuração dinâmica de níveis de consistência entre agrupamento de dados a depender do estado da comunicação da rede.
- Transações podem ser configuradas para executar em um dos dois níveis de consistência: forte e fraco (permite a leitura de dados desatualizados).

Limitações:

- Não provê consistência forte para transações que executam localmente. Para se obter consistência forte tem que se estar conectado.
- Não provê flexibilidade das propriedades ACID suficiente para atender requisitos de aplicações avançadas.

6 Conclusão

Com o surgimento da computação móvel, novos obstáculos foram introduzidos em ambientes de sistemas distribuídos. Para permitir que aplicações de acesso a dados distribuídos possam executar neste ambiente de maneira satisfatória, vários modelos de transações para a computação móvel. Porém, estes modelos possuem limitações como: (i) alguns modelos são voltados para um domínio de aplicação específico; (ii) alguns modelos não possuem a flexibilidade das propriedades ACID necessária para atender requisitos mais específicos de aplicações avançadas; (iii) alguns modelos não tratam problemas relevantes do ambiente móvel.

Este relatório apresentou um estudo dos principais modelos de transação para a computação móvel destacando algumas de suas contribuições e limitações. Este estudo apresentou também uma série de requisitos que podem ser utilizados como base para a elaboração de um modelo de transações mais flexível, que atenda os requisitos específicos de cada aplicação e que forneça mecanismos de adaptação aos obstáculos do ambiente de computação móvel.

Observando os requisitos de sistemas de transações apresentados neste estudo, pode-se dizer que um Sistema que é capaz de prover flexibilidade e configurabilidade necessárias para atender a um domínio maior de requisitos de aplicações poderia ser complexo e de grande tamanho (em bytes). Porém, a complexidade poderia ser reduzida através do uso de componentes onde um serviço de transação poderia ser fruto de uma composição de um ou mais componentes implementados independentemente. Projetos como OpenORB [2] e DynamicTAO [30] apresentam middlewares baseados em componentes que podem ser personalizados para se adequar à disponibilidade de recursos do ambiente. Com o uso de componentes, o tamanho em bytes do sistema também poderia ser reduzido ao compor o sistema somente a partir dos componentes que fossem ser utilizados pelas aplicações da unidade móvel.

Referências

[1] Bernstein, P. A., Hadzilacos, V. and Goodman, N. *Concurrency Control and Recovery in Database Systems*, Addison Wesley, 1987.

[2] Blair, G. S., Coulson, G., Robin, P. and Papathomas, M. *An architecture for next generation middleware*, In Proceedings of the IFIP International Conference on Distributed Systems Platforms and Open Distributed Processing, Springer-Verlag, London, 1998.

[3] Butrico, M., Chang, H., Cocchi, A., Cohen, N., Shea, D. and Smith S. *Gold Rush: Mobile transaction middleware with java-object replication*, In Proc. of the Third USENIX Conference on Object-Oriented Technologies, pages 91-101, 1997.

[4] Chrysanthis, P. K. *Transaction Processing in a Mobile Computing Environment*, Workshop on Advances in Parallel and Distributed Systems, pages 77-82, 1993.

[5] Dirckze, R. A., Gruenwald, L. *A toggle transaction management technique for mobile multidatabases*, In Proceedings of the seventh international conference on Information and knowledge management table of contents, Bethesda, Maryland, pages 371-377, 1998.

[6] Dirckze, R. A., Gruenwald, L. *A pre-serialization transaction management technique for mobile multidatabases*, MONET - Mobile Networks and Applications, Vol. 5, pages 311-321, 2000.

[7] Dunham, M. H. and Helal, A. *Mobile Computing and Databases: Anything New?*, ACM SIGMOD Record, Vol. 24, No. 4, December, 1995.

[8] Dunham, M. H. and Helal, A. *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, pages 149-162, 1997.

[9] Forman, G. H. and Zahorjan, J. *The Challenges of Mobile Computing*, IEEE Computer, Vol. 27, April, 1994.

[10] Gray, J. e Reuter, A. *Transaction Processing Concepts and Techniques*, Morgan Kaufmann, 1992.

[11] Gray, J., Helland, P., O'Neil, P. and Shasha, D. *The dangers of replication and a solution*, ACM SIGMOOD Conference, Montreal, Canada, June, 1996.

[12] Imielinski, T. and Badrinath, B. R. *Mobile Wireless Computing: Challenges in Data Management*, Communications of ACM, Vol. 37, No. 10, October, 1994.

[13] Jakobsen, A.B., Karlsen, R. *Towards a Reflective Transactional Middleware Platform for Advanced Applications*, Proceedings of NIK, Norwegian Informatics Conference 2003, Oslo, November, 2003.

[14] Karlsen, R. *An Adaptive Transactional System - framework and service synchronization*, Proc. of the International Symposium on Distributed Objects and Applications (DOA), Catania, Sicily, November 2003.

[15] Karlsen, R., Jakobsen, A.B. *Transaction Service Management - An approach towards a reflective transactional service*, Proceedings of the 2nd Workshop on Reflective and Adaptive Middleware, Rio de Janeiro, Brazil, 2003.

[16] Katz, R. H. *Adaptation and Mobility in Wireless Information Systems*, IEEE Per-sonal Communications, Vol. 1, No. 1, 1995.

[17] Ku, K. and Kim, Y. *Moflex Transaction Model for Mobile Heterogeneous Multidatabase Systems*, In IEEE Workshop on Research Issues in Data Engineering, San Diego, USA, 2000.

[18] Lam, K., Li, G. and Kuo, T. *A multi-version data model for executing real-time transactions in mobile environment*, Proceedings of the 2nd ACM international workshop on Data engineering for wireless and mobile, pages 90-97, 2001.

[19] Lee, M. and Helal, S. *HiCoMo: High Commit Mobile Transactions*, Kluwer Academic publishers Distributed and Parallel Databases, 2002.

[20] Lu, Q. and Satyanaranyanan, M. *Isolation-only transactions for mobile computing*, In ACM Operating Systems Review, Vol. 28, No. 3, 1994.

[21] Madria, S. K., Bhargava, B. *A Transaction Model for Mobile Computing*, Proceedings of the 1998 International Symposium on Database Engineering & Applications table of contents, Page 92, 1998.

[22] Madria, S. K., Bhargava, B. *A Transaction Model to Improve Data Availability in Mobile Computing*, Kluwer Academic Publishers - Distributed and Parallel Databases, Vol. 10, No. 2, 2001.

[23] Mehrotra, S., Rastogi, R., Korth, H. F. and Silberschatz, A. *Non-serializable execution in heterogeneous distributed database systems*, In Proceedings of the 2nd International Conference on Parallel and Distributed Information Systems, pages 245–252, 1991.

[24] Mummert, L. B., Ebling, M. R. and Satyanarayanan, M. *Exploring Weak Connec-tivity for Mobile File Access*, In Proceedings of the 15th ACM Symposium on Opera-tion Systems Principles, Colorado, December, 1995.

[25] Pitoura, E. and Bhargava, B. *Maintaining consistency of data in mobile distributed environments*, In Proceedings of the 15th International Conference on Distributed Computing Systems (ICDCS'95), page 404, 1995.

[26] Pitoura, E. *Data Management for Mobile Computing*, Summer School on Mobile Computing, Jyvaskyla, 1998.

[27] Pitoura, E. and Bhargava, B. *Data Consistency in Intermittently Connected Distributed Systems*, In Transactions on Knowledge and Data Engineeging, Vol. 11, Nov. 1999.

[28] Pu, C., Kaiser, G., Hutchinson, N. *Split-transactions for open-ended activities*, Proceedings of the 14th VLDB Conference, 1988.

[29] Preguica, N., Baquero, C., Moura, F., Martins, J., Oliveira, R., Domingos, H., Pereira, J. and Duarte, S. *Mobile Transaction Management in Mobisnap*,In Proc. of the East-European Conference on Advances in Databases and Information Systems Held Jointly with International Conference on Database Systems for Advanced Applications: Current Issues in Databases and Information Systems, Prague, Czech Rep., pages 379-386. Springer, 2000.

[30] Roman, M., Kon, F. and Campbell, R. *Design and Implementation of Runtime Reflection in Communication Middleware: the dynamicTAO Case*, Proceedings of the ICDCS'99 Workshop on Middleware, Austin, Texas, May-June 1999.

[31] Santos, N., Veiga, L. and Ferreira, P. *Transaction Policies for Mobile Networks*, In Proceedings of the Fifth IEEE International Workshop on Policies for Distributed Systems and Networks, 2004.

[32] Satyanarayanan, M. *Fundamental Challenges in Mobile Computing*, Fifteenth ACM Symposium on Principles of Distributed Computing, Philadelphia, PA, May, 1996.

[33] Satyanarayanan, M. *The Evolution of Coda*, In ACM Transactions on Computer Systems, Vol. 20, No. 2, May, 2002

[34] Serrano-Alvarado, P., Roncancio, C., Adiba, M. and Labbé, C. *Adaptable Mobile Transactions*, In Proceedings of the 29th VLDB Conference, Berlin, Germany, 2003.

[35] Walborn, G. D. and Chrysanthis, P. K. *Supporting semantics-based transaction processing in mobile database applications*, In Proceedings of the 14TH Symposium on Reliable Distributed Systems, page 31, 1995.

[36] Walborn, G. D. and Chrysanthis, P. K. *Transaction Processing in PRO-MOTION*, In Proceedings of the 4th ACM Annual Symposium on Applied Computing, 1999.