Coerência de Cache e Percepção em Ambientes de Cooperação Móvel

Carla Diacui Medeiros Berkenbrock[1], **Celso Massaki Hirata**[1]

[1]Divisão de Ciência da Computação
Instituto Tecnológico de Aeronáutica - ITA
12228-900 - São José dos Campos - SP - Brasil

{diacui,hirata}@ita.br

Abstract. *The advance of mobile computing and the new ways of connectivity allow the integration of mobile devices in collaborative applications. However, there are many restrictions so that a successful integration can be accomplished. The restrictions include the screen size of mobile devices, limited battery power and maintenance related to cache coherence. Nowadays, a great deal of collaborative applications considers only strongly coupled environments where disconnection is not a problem. The goal of this paper is to present a mechanism to deal with cache coherence which occur in weakly coupled environments, and the awareness in collaborative applications.*

Resumo. *Os avanços da computação móvel em adição as novas formas de conectividade permitem a integração de dispositivos móveis em trabalhos colaborativos. Entretanto, existem muitas restrições para que essa integração seja realizada com sucesso. Entre eles, podem-se citar o tamanho da tela dos dispositivos móveis, limitado poder de bateria e coerência de cache. Atualmente, a maior parte das aplicações colaborativas considera ambientes fortemente acoplados, onde a desconexão dos dispositivos móveis em geral não é um problema. O objetivo deste artigo é apresentar um mecanismo para tratar a coerência de cache que ocorre em ambientes fracamente acoplados e a percepção de colaboração em aplicações colaborativas.*

1. Introdução

A computação móvel tem emergido como uma nova disciplina no campo da Ciência da Computação. Como dispositivos móveis, podem-se citar telefones celulares, *Personal Digital Assistants* (PDAs) e computadores portáteis. Esses dispositivos são providos de uma capacidade de processamento, armazenamento, e comunicação cada vez maior. Dessa forma, um passo importante é proporcionar maior interação entre os usuários móveis.

A área de Trabalho Cooperativo Apoiado por Computador, em inglês *Computer-Supported Cooperative Work* (CSCW), indica uma variedade de tecnologias as quais permitem que equipes de trabalho cooperem eletronicamente. Um exemplo dessas tecnologias são os sistemas de *groupware* [Luff and Heath 1998]. Um *groupware* é um sistema baseado em computador que suporta dois ou mais usuários engajados em uma tarefa comum e que forneça uma interface para ambiente compartilhado [Ellis and Gibbs 1989].

O avanço das tecnologias relacionadas às redes sem fio e a proliferação de dispositivos móveis têm possibilitado o desenvolvimento de sistemas de *groupware* móvel.

Entretanto, a maior parte das ferramentas para trabalho cooperativo não atendem as restrições impostas pelos computadores portáteis e redes sem fio [Buszko et al. 2001]. Dentre essas restrições pode-se destacar: gerência de energia, fraca conectividade e interface limitada.

A ocorrência de desconexões é comum em ambientes sem fio, o que torna importante a redução da necessidade de comunicação durante o processo de cooperação. Nesse sentido, a replicação de dados é uma técnica chave para fornecer uma maior disponibilidade e desempenho em ambientes móveis [Saito and Shapiro 2005]. As réplicas, armazenadas localmente nos dispositivos móveis, são conhecidas na literatura como cache [Barbara and Imieliski 1994, Chung and Cho 1998, Saito and Shapiro 2005].

A replicação torna-se uma importante forma de reduzir a quantidade de informações transferidas, assim como a latência de acesso aos dados e o tráfego da rede. No entanto, a replicação apenas possibilita essas reduções se o cliente encontrar as informações de seu interesse armazenadas na cache de seu dispositivo móvel. Por outro lado, é importante ressaltar que, por causa das desconexões torna-se difícil verificar a validade da cache.

Métodos eficientes de coerência de cache são críticos para garantir que as aplicações em ambientes sem fio tenham um desempenho razoável [Yuen et al. 2000]. Entretanto, a maioria dos métodos atuais para coerência de cache em ambientes móveis não foram projetados para trabalhar com sistemas cooperativos [Barbara and Imieliski 1994, Chung and Cho 1998, Wang et al. 2004]. Adicionalmente, a maior parte dessas estratégias não contempla a percepção *(awareness)*. A percepção é definida neste artigo como um sentimento de ilusão através do qual um usuário identifica a presença dos outros participantes e algumas de suas atividades no espaço de trabalho compartilhado.

Nesse contexto, o principal objetivo deste trabalho é propor uma técnica de manutenção da coerência de cache e provisão de percepção em um ambiente cooperativo computadorizado. Considera-se que esse ambiente suporte a mobilidade dos usuários através de uma interface de comunicação sem fio.

O artigo está organizado da seguinte forma. A seção 2 fornece uma breve visão sobre os trabalhos relacionados. A seção 3 contém a descrição da técnica proposta para prover a coerência de cache e percepção em aplicações de *groupware* móvel. A simulação da técnica e análise dos resultados obtidos está descrita na seção 4. Finalmente, na seção 5, apresentam-se as conclusões e trabalhos futuros.

2. Trabalhos Relacionados

No contexto da computação móvel, pode-se definir cache como uma forma de replicar os dados mais relevantes na memória dos dispositivos móveis. Dessa forma, a utilização da cache permite que esses dispositivos realizem o processamento de seus dados de forma mais rápida e consumindo uma quantidade menor de energia.

Neste trabalho, assume-se que os dados a serem manipulados no *groupware* possam ser replicados nos dispositivos móveis. Com isso, são geradas múltiplas cópias de uma determinada informação. Esse comportamento introduz o problema da consistência entre essas cópias, também conhecido como coerência de cache. Dessa forma, os be-

nefícios da replicação dos dados nas unidades móveis (UMs) apenas podem ser obtidos se a coerência de cache for mantida.

Para manter a coerência de cache é necessário informar as UMs quando itens de dados de suas caches forem alterados. Uma maneira de informar as UMs sobre essas alterações é através da notificação de invalidação (NI) [Barbara and Imieliski 1994]. A NI é uma abordagem utilizada para que a cache seja reutilizada após desconexões. Nessa abordagem, um servidor dissemina notificações contendo as alterações/invalidações a serem realizadas na cache das UMs. Esse servidor, responsável pelas notificações, pode ser do tipo *stateful* ou *stateless*. O servidor *stateful* tem conhecimento de quais UMs estão conectadas a ele e dos dados armazenados localmente nelas. Dessa forma, ao ocorrer uma alteração, ele envia notificações de invalidação (NIs) diretamente para as UMs afetadas. O servidor *stateless* não tem conhecimento sobre as UMs conectadas a ele, assim como, não possui conhecimento do conteúdo dos dados dessas UMs. Nesse caso, as mensagens de invalidação são enviadas para todas as UMs.

A maior parte dos trabalhos relacionados à coerência de cache em ambientes móveis utiliza servidores *stateless* [Barbara and Imieliski 1994, Berkenbrock and Dantas 2005]. Isso acontece porque as abordagens de invalidações imediatas, como a *stateful*, não são praticáveis em ambientes onde os clientes se conectam e desconectam freqüentemente da rede sem fio. Nesse caso, o servidor tem dificuldade em manter-se informado sobre todos os clientes que possuem uma cópia de um determinado registro que tenha sido modificado. Wang [Wang et al. 2004] propõe uma técnica de coerência de cache híbrida, onde são herdadas as características positivas das abordagens *stateless* e *stateful*. Entretanto, essas técnicas não foram projetadas para sistemas de *groupware*. Geralmente, os trabalhos relacionados à coerência de cache em ambientes móveis preocupam-se com a redução da quantidade de mensagens trocadas e eficiência da coerência, no entanto, eles não levam em consideração a percepção e o desempenho do trabalho cooperativo.

No trabalho de Neyem [Neyem et al. 2006], é proposta uma técnica para permitir o compartilhamento e manipulação de dados em um ambiente de *Mobile Ad-hoc NETworks* (MANETs). A técnica considera manipulação *off-line* dos dados, fornecendo funcionalidades de sincronização através de um processo de reconciliação. Para facilitar a detecção de conflitos e suportar a reconciliação dos dados, utiliza-se um sistema de versão que gerencia as diferentes versões dos documentos. A manutenção da coerência é baseada em árvores de diferenciação e técnicas de *merging*. Contudo, o trabalho não apresenta informações de percepção.

Kirsch-Pinheiro [Kirsch-Pinheiro et al. 2005] apresenta um *framework* para fornecer percepção com base no contexto do usuário. O ambiente pode ser acessado a partir de dispositivos móveis onde, devido às restrições como tamanho de tela reduzido, existe a necessidade de filtragem das informações de percepção. Nesse trabalho, o contexto atual do usuário é utilizado para filtrar as informações que sejam pertinentes à unidade móvel (UM).

Outro trabalho relacionado à percepção é o de Papadopoulos [Papadopoulos 2006]. Ele investiga oportunidades para melhorar a percepção em ambientes móveis. Papadopoulos considera em seu estudo, ambientes com restrições de

processamento e desconexões transitórias da rede. Para isso, ele elaborou um CSCW síncrono e desenvolveu um protocolo que propõe um balanceamento entre consumo de energia e tempo de notificação. O objetivo do autor é propor mecanismos que forneçam percepção para colaboradores móveis de maneira eficiente em termos de tempo e consumo de energia. Tanto o trabalho de Papadopoulos quanto o de Kirsch-Pinheiro são interessantes no que diz respeito à percepção, entretanto, eles não fornecem suporte a coerência de cache.

3. Uma Técnica para Coerência de Cache e Percepção em Ambientes de Cooperação Móvel

A forma de interação entre os participantes de um *groupware* pode ser síncrona ou assíncrona. Um *groupware* síncrono apresenta a idéia de simultaneidade no tempo, isto é, as notificações de eventos são imediatamente propagadas para todos os participantes. Em um *groupware* assíncrono os participantes interagem em momentos distintos, sendo necessário a existência de registros de mudanças de estados. Considerando um ambiente de cooperação móvel, com possibilidade de desconexão dos participantes, manter a sincronização dos dados geralmente é dispendioso e pode ser inapropriado. A abordagem proposta neste artigo utiliza um ambiente flexível. Os participantes conectados ao sistema trabalham de forma síncrona, enquanto os participantes desconectados podem continuar a cooperação de forma assíncrona. Dessa forma, o sistema possibilita diferentes estilos de trabalho cooperativo. Na seção 3.1 serão apresentadas as principais características do ambiente do *groupware* móvel proposto. A seção 3.2 apresenta a arquitetura do ambiente. A seção 3.3 descreve a técnica para tratar o problema da coerência de cache e percepção no ambiente de cooperação.

3.1. Ambiente Experimental

Em dispositivos móveis maiores, como *notebooks*, a mobilidade é limitada por fatores como peso e tamanho do equipamento. Para que o *groupware* suporte uma maior mobilidade dos usuários, no ambiente proposto, serão investigados problemas relacionados com pequenos dispositivos móveis, como PDAs.

No ambiente, cada dispositivo móvel possui capacidade de armazenar dados localmente. Conforme apresentado na seção 2, os dados das UMs podem ser atualizados através de servidores *stateful* ou de servidores *stateless*. Neste trabalho, optou-se pela utilização de um ambiente híbrido, mesclando características das abordagens *stateless* e *stateful*. Adicionalmente, o ambiente fornecerá suporte à percepção. Para fornecer dados de percepção, o ambiente permite que as UMs tenham conhecimento de informações relacionadas à conectividade e as atividades dos membros no grupo. A percepção das atividades realizadas segue o modelo de interface *What You See Is What I Did* (WYSIWID) [Marshall et al. 1991]. Nesse modelo, as interações entre os usuários ocorrem de forma assíncrona, pois a propagação das alterações para os demais usuários será adiada até que as condições, presentes no servidor do *groupware*, sejam estabelecidas.

O ambiente proposto representa as informações compartilhadas através de uma estrutura de dados do tipo grafo. Várias formas de representação que usam grafos podem ser adotadas, como por exemplo redes de Petri e diagramas de classe da *Unified Modeling Language* (UML). Adicionalmente, vários trabalhos utilizam grafos para de-

monstrar métodos de coerência em ambientes compartilhados [Correa and Marsic 2003, Neyem et al. 2006].

Apesar de o ambiente experimental considerar aplicações baseadas em representação de grafos, a técnica proposta neste artigo pode ser aplicada em outras aplicações existentes tais como controle de estoque e jogos. A técnica também pode ser considerada em classes de aplicações ainda não existentes ou aquelas que precisam satisfazer requisitos específicos. Uma classe de aplicação interessante é aquela que provê auxílio a reuniões presenciais: (i) para votação com requisito de anomicidade e confidencialidade, (ii) organizar grupos de interesse de maneira mais eficiente para realizar tarefas etc. A característica comum das aplicações é a percepção de ações, consistência de dados e mobilidade dos participantes.

O trabalho cooperativo é realizado a partir de um ambiente infra-estruturado, isto é, a rede sem fio envolve servidores com localização fixa (Estação de Suporte Móvel), que se comunicam com os dispositivos móveis (Unidades Móveis) via freqüências de radio. A Estação de Suporte Móvel (ESM) possui um ponto de acesso IEEE 802.11 (interface sem fio). Através desse ponto de acesso, a ESM fica responsável pela área geográfica chamada célula. Apenas as UMs que estiverem na área de cobertura dessa célula poderão se comunicar com o servidor. Com isso, a ESM atua como um intermediário entre as UMs e a Estação Fixa (EF), sendo responsável pela comunicação. A EF é o servidor de *groupware*. A Figura 1 representa o ambiente experimental.

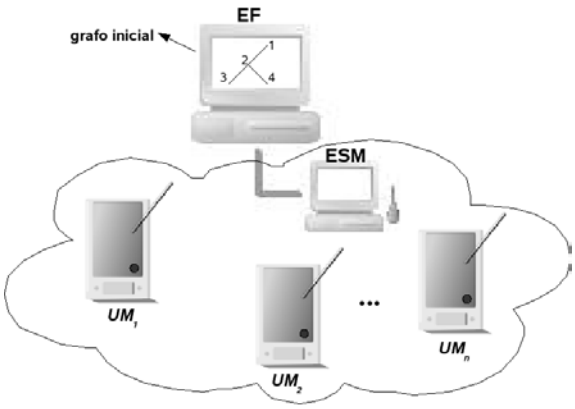

Figura 1. Ambiente experimental

3.2. Arquitetura do Ambiente

A arquitetura do ambiente proposto é apresentada na Figura 2. Os componentes principais dessa arquitetura são a EF e as UMs. Nesse ambiente, a EF se comunica com as UMs através do canal de *downlink*. Essa comunicação apenas será realizada se a UM possuir energia e estiver na área de cobertura da rede sem fio. As UMs podem obter réplicas dos grafos armazenados na EF. Adicionalmente, elas podem solicitar alteração dos grafos através do canal de *uplink*. No entanto, essa alteração apenas será realizada com a confirmação da EF.

A EF é composta pelos módulos descritos a seguir:

- Gerenciador de *groupware* (GG) - Responsável pelo gerenciamento das UMs no *groupware*. Através do GG, a EF recebe solicitações das UMs. O GG também é

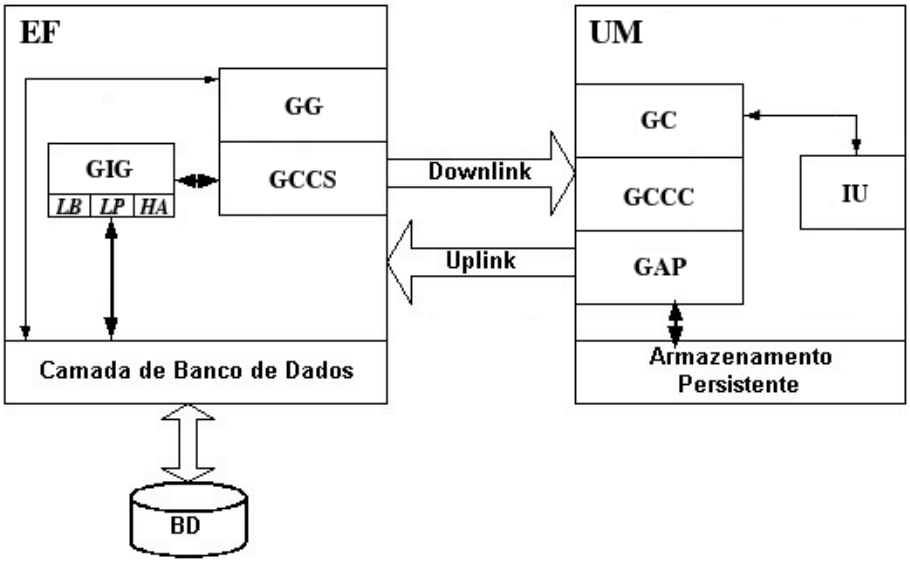

Figura 2. Arquitetura do ambiente

responsável por retornar as respostas a essas solicitações e enviar as notificações dos eventos ocorridos;

- Gerenciador para coerência de cache no servidor (GCCS) - Realiza o gerenciamento da coerência de cache e da percepção na EF. É responsável pela atualização do rótulo de tempo da EF (RT_{EF}), rótulo de tempo das UMs (RT_{UM}), janela de desconexão (*w*) e contador de tempo de desconexão (*ct*). Essas variáveis possuem papel importante na realização da coerência de cache e provisão da percepção. Os valores RT_{UM} e RT_{EF} ajudam a verificar se as UMs deixaram de receber alguma notificação de alteração (*NA*). A *NA* contém as informações necessárias para que as UMs atualizem suas caches, sempre que ocorrer alguma alteração no grafo. Assim, a *NA* auxilia no fornecimento da percepção a respeito das atividades realizadas pelos participantes do *groupware*. Os rótulos de tempo indicam quando o grafo foi atualizado pela última vez. A atualização dos valores RT_{UM} e RT_{EF} ocorre a cada alteração do grafo na EF. O valor de *w* indica quanto tempo as UMs podem permanecer desconectadas, sem que seus dados sejam invalidados. Os valores de *w* e *ct* são utilizados no fornecimento de um *time-out* para liberar bloqueios (travas) em itens de dados solicitadas por UMs desconectadas. Adicionalmente, o GCCS é responsável por ativar o gerenciador de informações no *groupware* (GIG) quando for necessário realizar alguma alteração no banco de dados; e
- Gerenciador de informações no *groupware* (GIG) - Responsável pela alteração da lista de bloqueios (*LB*), lista de pendências (*LP*) e histórico de alterações (*HA*). As informações de bloqueios de nodos do grafo, realizados pelo grupo, são armazenados na *LB*. A *LB* será utilizada para verificar se as caches das UMs, que se desconectaram da rede, precisarão ser invalidadas. A *LP* contém informações sobre as UMs que desejam trabalhar em um nodo que já foi bloqueado. O *HA* contém as alterações realizadas nos últimos *w* segundos. Essas variáveis são fundamentais tanto para o fornecimento da coerência de cache e percepção quanto para possibilitar a interação assíncrona. Por exemplo, com as informações conti-

das no *HA* e na *LP*, as UMs terão conhecimento das atividades realizadas pelos outros participantes do *groupware*. Adicionalmente, se uma UM ficar desconectada, ela poderá atualizar sua cache com o auxilio das informações do *HA* e da *LB*.

Cada UM é composta pelos seguintes módulos:

- Gerenciador de conexão (GC) - Gerencia a conexão da UM com a EF;
- Gerenciador para coerência de cache no cliente (GCCC) - Nele ficam registradas o valor do rótulo de tempo da sua UM (RT_{UM}), janela de desconexão (*w*) e contador de tempo de desconexão (*ct*). Os valores de RT_{UM} e *w* são atualizados pela EF. O valor de *ct* indica à quanto tempo a UM está desconectada. O GCCC também é responsável por ativar o gerenciador do armazenamento persistente (GAP) quando for necessário realizar alguma alteração na cache da UM;
- Gerenciador do armazenamento persistente (GAP) - Executa as alterações necessárias na cache do dispositivo móvel; e
- Interface com o usuário (IU) - Representa uma interface amigável para que as UMs possam acessar o ambiente. Através dessa interface a UM se mantêm informada sobre a atividade dos outros participantes do *groupware* e pode realizar consultas/alterações em suas réplicas locais, e confirmá-las na EF.

3.3. Descrição da Técnica

Os seguintes passos constituem uma descrição informal da técnica para coerência de cache e percepção em sistemas de *groupware* com suporte a mobilidade:

1. Após entrarem no *groupware*, as UMs solicitam o envio do grafo relacionado ao seu grupo cooperativo. Dessa forma, o grafo é armazenado na cache de cada UM. Adicionalmente, a EF envia um rótulo de tempo que será armazenado nas UMs para consistências futuras. Então, as UMs podem solicitar o bloqueio (trava) dos nodos de interesse;
2. Ao receber uma solicitação de bloqueio da UM, a EF pode aceitá-la ou recusá-la. Em caso de aceite, a EF notifica os demais membros do grupo sobre o bloqueio. Os bloqueios realizados pelo grupo são armazenados na *LB*. Se a EF recusar o bloqueio, ela envia uma mensagem para a UM solicitante com o motivo da recusa. Se a recusa ocorre porque o nodo solicitado já está bloqueado, a EF envia uma notificação para a UM que possui o bloqueio, avisando sobre o interesse do outro participante do grupo. A EF, então, adiciona a UM solicitante na *LP*. A UM apenas começa a trabalhar no grafo, após conseguir o bloqueio de algum nodo. Utiliza-se a abordagem centralizada de prevenção de *deadlock* para evitar o *deadlock* de recurso. As UMs devem solicitar o bloqueio de todos os nodos desejados de uma única vez. A EF tem o conhecimento do estado global da alocação dos nodos através da *LB*;
3. Cada UM portadora de um grafo possui o RT_{UM}, enviado pela EF, indicando quando o grafo foi atualizado pela última vez, conforme o exemplo apresentado na Figura 3. Após alterar o grafo, a UM pode confirmar as alterações com a EF. Antes de efetivar as alterações, a EF faz as seguintes verificações:
    - Se $RT_{UM} = RT_{EF}$, a EF registra as alterações da UM no *HA*. Em seguida, a EF atualiza RT_{EF} (o novo valor indicará o instante dessa última

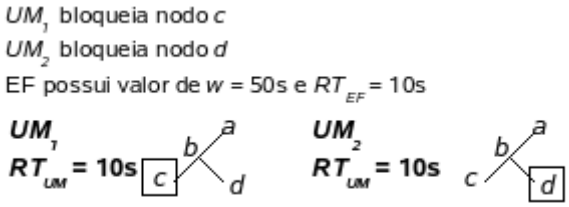

Figura 3. UMs solicitam bloqueio de parte do grafo

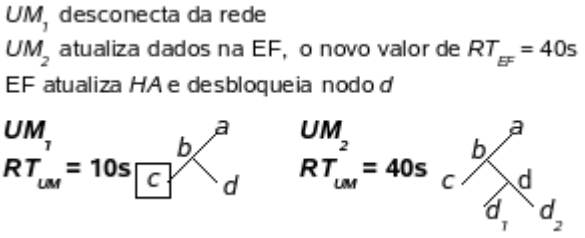

Figura 4. *UM*$_1$ desconecta da rede e *UM*$_2$ confirma alteração de dados

alteração) e envia uma *NA* para demais UMs, pertencentes ao grupo, que estão conectadas. A *NA* contém os novos dados do grafo e o novo rótulo de tempo (para que as UMs atualizem o valor RT_{UM} com o novo valor do RT_{EF}). Nesse momento, o bloqueio do nodo que foi atualizado é liberado. Veja o exemplo na Figura 4. A EF, então, verifica na *LP* se existe alguma UM interessada em obter bloqueio do nodo liberado. Caso positivo, a EF envia uma notificação para a UM interessada, questionando sobre o interesse em alterar o nodo liberado;

- Se $RT_{EF} - RT_{UM} \leq w$, a EF envia para a UM uma mensagem contendo todas as alterações realizadas após o tempo RT_{UM} . Com isso, a UM pode atualizar os seus dados e reiniciar o processo de confirmação da alteração;
- Se $RT_{EF} - RT_{UM} > w$, isso significa que ocorreram alterações no *groupware* que já não estão mais armazenadas no histórico. A EF, então verifica na *LB* se algum outro participante bloqueou o nodo utilizado pela UM após o tempo RT_{UM}. Caso positivo, a EF não tem capacidade de verificar a coerência de cache. Sendo assim, será necessário invalidar as informações da UM. Caso contrário, a EF envia novamente o grafo para que a UM possa unir as suas alterações com o grafo atual;

4. Quando uma UM se desconecta da célula de comunicação, se ela possuir algum nodo bloqueado, a EF avisa as demais UMs sobre a desconexão. Nesse caso, a partir do momento de desconexão, a EF e a UM desconectada iniciam o contador *ct*. É necessário que *ct* exista tanto na UM quanto na EF, pois é importante que a UM, ainda que desconectada, tenha percepção de quanto tempo ela poderá trabalhar. Adicionalmente, a EF precisa ter conhecimento do tempo de desconexão da UM. Enquanto *ct* for menor do que *w*, a EF não permite o bloqueio a nenhuma outra UM do nodo que está sendo atualizado pela UM desconectada. Caso contrário, a EF liberará o nodo bloqueado;
5. Quando a UM se reconecta a célula de comunicação, é verificado o valor de *ct*:
   - Se $ct \leq w$, a UM desconectada pode solicitar confirmação de alteração com a EF. Porém, antes disso, a EF verificará o RT_{UM} e enviará uma mensagem contendo todas as alterações que ocorreram no grupo enquanto a UM permaneceu desconectada. Veja o exemplo na Figura 5. Com isso, a UM atualiza os seus dados e reinicia o processo de confirmação da alteração, conforme o exemplo mostrado na Figura 6;
   - Se $ct > w$ será necessário verificar, na *LB*, se outro participante bloqueou o nodo utilizado pela UM, durante o seu período de desconexão. Caso

existam registros de bloqueio do nodo na *LB*, será necessário invalidar as informações da UM. Caso contrário, a EF envia novamente o grafo para que a UM realize a união das suas alterações ao grafo atual.

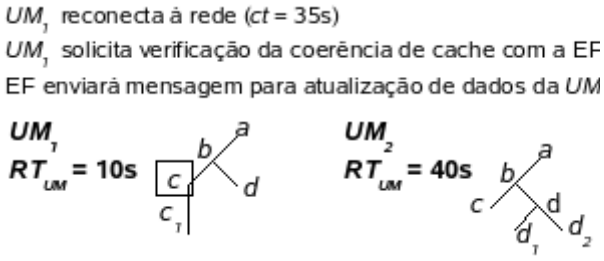

Figura 5. *UM*$_1$ reconecta à rede

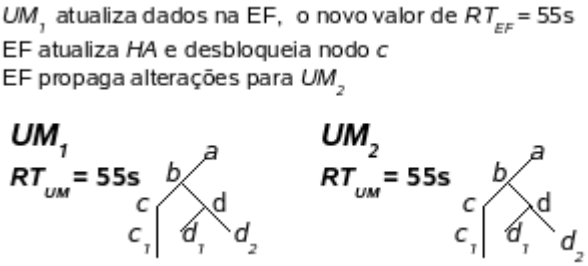

Figura 6. *UM*$_1$ confirma alteração de dados

4. Avaliação da Técnica

Durante o processo de cooperação, as UMs consomem energia para enviarem e receberem dados. Dessa forma, para que uma UM possa permanecer uma quantidade maior de tempo conectada, é importante reduzir a quantidade de mensagens trocadas durante o processo de cooperação [Papadopoulos 2006]. Os experimentos conduzidos neste artigo, consideram a quantidade de dados enviados e recebidos para verificar a eficiência da estratégia proposta.

Os experimentos foram conduzidos utilizando o simulador *Network Simulator* (NS-2) [Berkeley et al. 2006]. O NS-2 é um simulador para eventos discretos direcionado à pesquisa em redes de computadores. O NS-2 possibilita a visualização gráfica dos dados simulados através da ferramenta chamada *Network Animator* (NAM). A Figura 7 mostra animação do ambiente projetado. O nodo na forma de quadrado representa a ESM. O nodo conectado diretamente a ele corresponde a EF. Os demais nodos são as UMs.

O modelo desenvolvido suporta tanto redes cabeadas (para comunicação da EF com a ESM) quanto redes sem fio (para comunicação das UMs com a ESM). Ele foi escrito na linguagem de programação *Tool Command Language* (Tcl).

Para simular a técnica proposta para coerência de cache e percepção foi desenvolvida uma aplicação chamada *Mobile Groupware Cache Coherence* (MGCC). Essa aplicação foi escrita utilizando a linguagem de programação C++. O modelo, implementado em Tcl, em conjunto com a aplicação MGCC foram definidos de acordo com a arquitetura apresentada na seção 3.2. A aplicação MGCC é inserida em cada UM.

No ambiente é utilizado o protocolo de rede UDP e a camada MAC de acordo com o padrão 802.11. O link entre a EF e a ESM é bi-direcional e possui capacidade de 5 Mb/s para cada direção. O MGCC foi utilizado em um cenário com área de simulação de 400m×400m, contendo 10 nodos móveis. Nesse cenário ocorre a média de uma alteração por minuto.

O valor de *w*, considerado nos experimentos, é de 65 segundos. A configuração da máquina utilizada para simular o ambiente de cooperação é: processador AMD Sempron 2300, 512 MB de memória e sistema operacional Open Suse Linux 10.1.

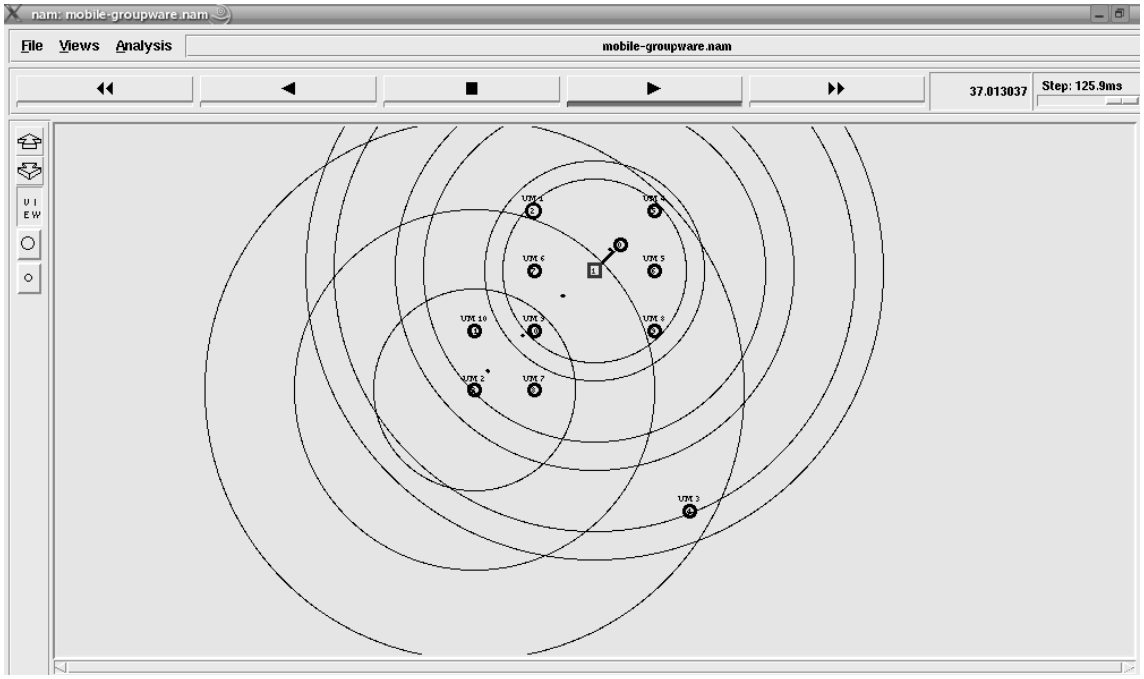

Figura 7. Animação do experimento

As Figuras 8, 9 e 10 apresentam os dados trocados entre as UMs e a EF, em um experimento contendo apenas uma desconexão. Nas Figuras 11, 12 e 13 são mostradas as mensagens trocadas no sistema, em um experimento com 3 desconexões. As desconexões apresentadas nas Figuras 8 e 11 possuem tempo inferior a *w*. Nesse caso, para garantir a coerência de cache e percepção, foi necessária a troca de um maior número de mensagens entre as UMs que ficaram desconectadas e a EF (média de 405 pacotes), se comparado com as UMs que permaneceram conectadas durante todo o experimento (média de 324 pacotes).

Quando o tempo de desconexão é maior do que *w*, poderão ocorrer duas situações. Na primeira situação, nenhum outro participante bloqueou o nodo utilizado pela UM durante a sua desconexão. Nesse caso, a EF envia novamente o grafo para a UM unir suas alterações ao grafo atual (Figuras 9 e 12). Na segunda situação, existe alguma outra UM na *LB*, bloqueando o nodo que a UM desconectada estava atualizando. Nesse caso, os dados da UM que acabou de reconectar serão invalidados (Figuras 10 e 13). Se a UM que teve os seus dados invalidados desejar continuar participando das atividades do grupo, ela necessitará reiniciar o processo de entrada no grupo cooperativo, o que resultará em um aumento no número de mensagens trocadas no sistema MGCC. Nessas duas situações, a média de pacotes trocados entre as UMs desconectadas e a EF é 486. Contudo, na segunda situação ocorre um aumento no número de mensagens trocadas não apenas entre as UMs que ficaram desconectadas, como também nas UMs que adquiriram os novos bloqueios.

A técnica proposta possui melhor desempenho em ambientes com um número reduzido de desconexões. Adicionalmente, o ambiente mostra-se eficiente em situações onde os intervalos de desconexão são menores do que *w*. Quando os intervalos de desconexão são maiores do que *w*, se outras UMs adquirem os bloqueios das UMs desconectadas, ocorrem invalidações dos dados armazenados nas UMs. Essas invalidações

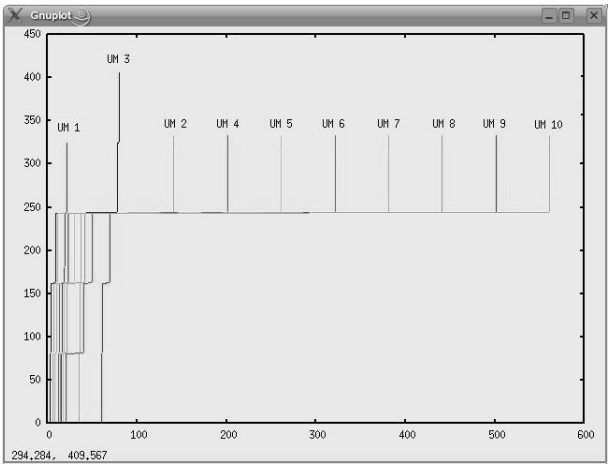

Figura 8. Desconexão de uma UM com *ct* inferior a *w*

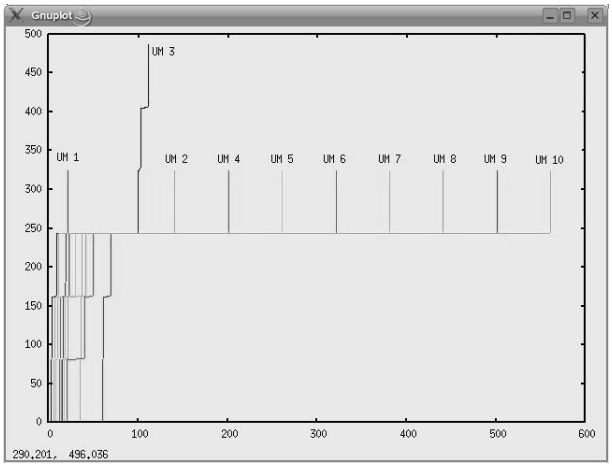

Figura 9. Desconexão de uma UM com *ct* superior a *w*, sem novos bloqueios no nodo em alteração

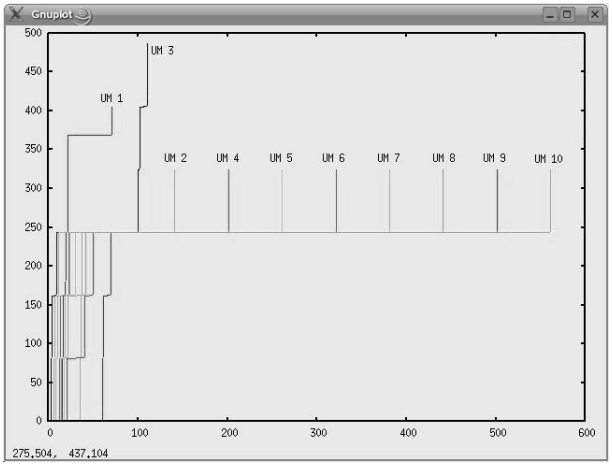

Figura 10. Desconexão de uma UM com *ct* superior a *w*, com novo bloqueio no nodo em alteração

são necessárias para assegurar a coerência de cache. Entretanto, um número elevado de invalidações comprometerá o desempenho do sistema, ocasionando retrabalhos e desmotivando os participantes do *groupware*. Em contrapartida, para evitar essas invalidações, a técnica usa a *LB*. Com a *LB*, o número de invalidações é reduzido, entretanto, a quantidade de mensagens trocadas entre a EF e a UM continua alta. Isso acontece porque, nesse caso, a EF já não possui no *HA* todas as alterações realizadas no grupo cooperativo durante o período de desconexão da UM. Então, para manter a cache, a EF necessita propagar todo o conteúdo do grafo para que a UM possa incluir no grafo as alterações que foram efetuadas em sua cache local.

Analisando a técnica proposta, pode-se provar informalmente que a coerênciade cache é garantida e livre de *deadlock*. Isso decorre pelas seguintes razões: (i) uso de bloqueios para realizar o controle de concorrência, (ii) uso de *time-out* para liberar bloqueios em itens de dados solicitados por UMs desconectadas, (iii) carga do grafo atualizado para as UMs desconectadas acima do período de *time-out*, e (iv) uso da abordagem centralizada

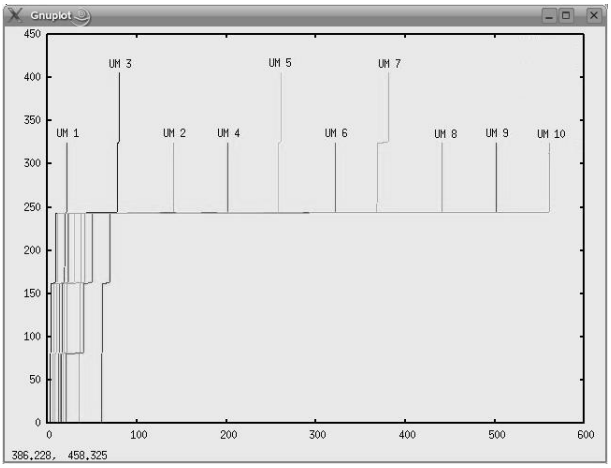

Figura 11. Desconexão de três UMs com *ct* inferior a *w*

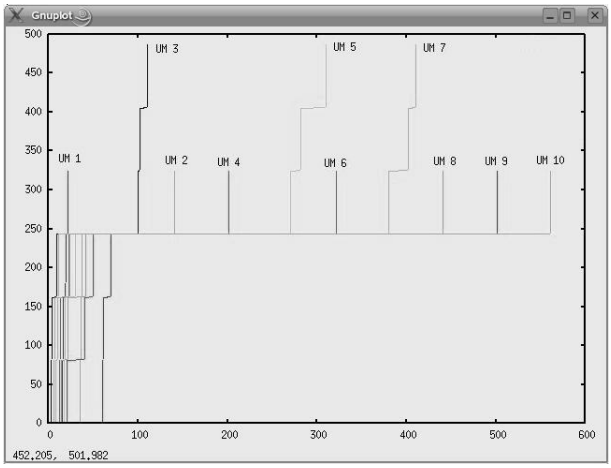

Figura 12. Desconexão de três UMs com *ct* superior a *w*, sem novos bloqueios nos nodos em alteração

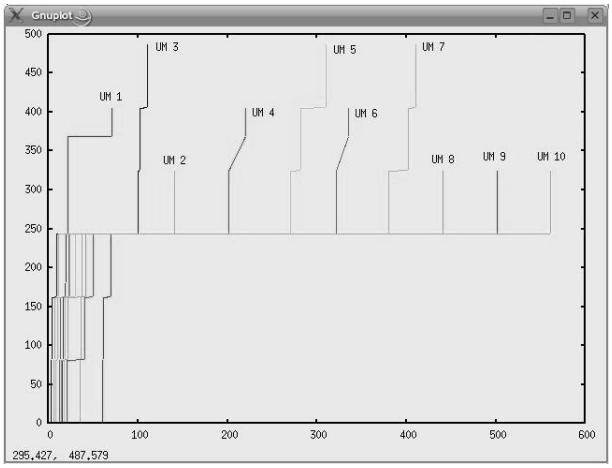

Figura 13. Desconexão de três UMs com *ct* superior a *w*, com novos bloqueios nos nodos em alteração

de prevenção de *deadlock*.

É importante ressaltar que a definição de um valor de *w* ideal, depende do tipo de aplicação. Ambientes com um valor de *w* pequeno podem resultar em elevado número de invalidações. No entanto, um valor alto para *w* pode resultar em retardos para obtenção de bloqueios e aumento da troca de mensagens no sistema. Dessa maneira, acredita-se que seja essencial obter um conhecimento prévio sobre comportamento dos usuários móveis dentro *groupware*, para que se possa determinar um valor eficiente para a janela de desconexão.

5. Conclusões e Trabalhos Futuros

Uma infra-estrutura completa para o desenvolvimento de sistemas de *groupware* móvel envolve uma série de fatores, como interface, acoplamento e interação. Cada um desses aspectos constitui áreas de pesquisas por si só. Neste artigo foi apresentado o projeto e simulação de uma estratégia para tratar os problemas de manutenção da coerência de

cache e falta de percepção em ambientes de *groupware* móveis.

Os experimentos foram conduzidos utilizando o simulador *Network Simulator*. Nele foi desenvolvido um modelo que representa a estrutura de um cenário de uso proposto. Adicionalmente, foi desenvolvida uma aplicação chamada *Mobile Groupware Cache Coherence* (MGCC).

Trabalhos futuros incluem o aperfeiçoamento do MGCC possibilitando a simulação de cenários de uso mais complexos. Pretende-se possibilitar que o sistema identifique o tamanho da janela de desconexão (w) de forma adaptativa. Assim, com base em informações relacionadas às desconexões e quantidade de bloqueios, o sistema poderá alterar o valor de w, reduzindo o tráfego na rede e a quantidade de invalidações.

Adicionalmente, pretende-se implementar o MGCC em um ambiente real. Para isso, será importante definir técnicas de visualização e filtragem de informações de percepção, considerando que o tamanho da tela dos PDAs dificulta a manipulação das informações.

6. Agradecimentos

Ao Conselho Nacional de Desenvolvimento Científico e Tecnológico - CNPq pelo apoio financeiro. Número do processo: 141781/2006-8. Agradecemos também aos revisores anônimos pela importante colaboração através dos comentários e sugestões.

Referências

Barbara, D. and Imieliski, T. (1994). Sleepers and workaholics: caching strategies in mobile environments. In *SIGMOD '94: Proceedings of the 1994 ACM SIGMOD international conference on Management of data*, pages 1–12, New York, NY, USA. ACM Press.

Berkeley, U., LBL, USC/ISI, and PARC., X. (2006). The ns manual.

Berkenbrock, C. D. M. and Dantas, M. A. R. (2005). Investigation of cache coherence strategies in a mobile client/server environment. *International Conference on Computational Science (ICCS 2005)*.

Buszko, D., Lee, W.-H. D., and Helal, A. S. (2001). Decentralized ad-hoc groupware api and framework for mobile collaboration. In *GROUP '01: Proceedings of the 2001 International ACM SIGGROUP Conference on Supporting Group Work*, pages 5–14, New York, NY, USA. ACM Press.

Chung, H. and Cho, H. (1998). Data caching with incremental update propagation in mobile computing environments. *Australian Computer Journal*, 30(2):77–86.

Correa, C. D. and Marsic, I. (2003). Software framework for managing heterogeneity in mobile collaborative systems. pages 125–134.

Ellis, C. A. and Gibbs, S. J. (1989). Concurrency control in groupware systems. In *SIGMOD '89: Proceedings of the 1989 ACM SIGMOD international conference on Management of data*, pages 399–407, New York, NY, USA. ACM Press.

Kirsch-Pinheiro, M., Villanova-Oliver, M., Gensel, J., and Martin, H. (2005). Bw-m: a framework for awareness support in web-based groupware systems. In *Proceedings of*

the Ninth International Conference Computer Supported Cooperative Work in Design, 2005, pages 240–246. IEEE Computer Society.

Luff, P. and Heath, C. (1998). Mobility in collaboration. In *CSCW '98: Proceedings of the 1998 ACM conference on Computer supported cooperative work*, pages 305–314, New York, NY, USA. ACM Press.

Marshall, C. C., Halasz, F. G., Rogers, R. A., and William C. Janssen, J. (1991). Aquanet: a hypertext tool to hold your knowledge in place. pages 261–275.

Neyem, A., Ochoa, S. F., and Pino, J. A. (2006). A strategy to share documents in manets using mobile devices. In *The 8th International Conference Advanced Communication Technology, ICACT 2006*, pages 1400–1404. IEEE Computer Society.

Papadopoulos, M.-C. (2006). Improving awareness in mobile cscw. *IEEE Transactions on Mobile Computing*, 5(10):1331–1346. Member-Constantinos Papadopoulos.

Saito, Y. and Shapiro, M. (2005). Optimistic replication. *ACM Comput. Surv.*, 37(1):42–81.

Wang, Z., Das, S. K., Che, H., and Kumar, M. (2004). A scalable asynchronous cache consistency scheme (saccs) for mobile environments. In *IEEE transactions on parallel and distributed systems*. IEEE Computer Society.

Yuen, J. C.-H., Chan, E., Lam, K.-Y., and Leung, H. W. (2000). Cache invalidation scheme for mobile computing systems with real-time data. *SIGMOD Rec.*, 29(4):34–39.