UNIVERSIDADE FEDERAL DE SANTA CATARINA
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

Carla Diacui Medeiros Berkenbrock

Investigação e Implementação de Estratégias de Notificação de Invalidação para Coerência de Cache em Ambientes de Computação Móvel sem Fio

Dissertação submetida à Universidade Federal de Santa Catarina como parte dos requisitos para a obtenção do grau de Mestre em Ciência da Computação.

Prof. Mario Antonio Ribeiro Dantas, Dr.
Orientador

Florianópolis, Fevereiro de 2005

Investigação e Implementação de Estratégias de Notificação de Invalidação para Coerência de Cache em Ambientes de Computação Móvel sem Fio

Carla Diacui Medeiros Berkenbrock

Esta Dissertação foi julgada adequada para a obtenção do título de Mestre em Ciência da Computação, área de concentração Sistemas de Computação e aprovada em sua forma final pelo Programa de Pós-Graduação em Ciência da Computação.

Prof. Raul Sidnei Wazlawick, Dr.
Coordenador do Curso

Banca Examinadora

Prof. Mario Antonio Ribeiro Dantas, Dr.
Orientador

Prof. Antônio Augusto Fröhlich, Dr.

Prof. Noemi Rodriguez, Dra.

Prof. Rômulo Silva de Oliveira, Dr.

Prof. Ronaldo dos Santos Mello, Dr.

Dedico esta dissertação aos meus pais, Sonia e Carlos,
por terem sido os grandes mestres de minha vida
e ao meu marido, Gian, por todo seu amor e por ter
embarcado comigo neste nosso sonho.

Agradecimentos

Agradeço ao meu orientador, Prof. Mario Dantas, pela oportunidade e pela experiência, dedicação e motivação transmitidos durante o mestrado.

Ao meu colega, amigo e amor Gian, pelas dicas sempre construtivas, conversas esclarecedores, carinho e companherismo.

Aos colegas do LABWEB pelos momentos de descontração, conselhos e amizade. Às meninas do LSD (Fabricia Lemos e Underléa Cabreira Corrêa) pelas conversas esclarecedoras. Ao colega Carlos Augusto Tibiriça pela amizade, pelas festas e pelas "pamonhas" trazidas de Goiás. À Vera Sodré (Verinha) pela simpatia e prestatividade oferecida.

Aos meus pais, Sonia e Carlos, pelo carinho, incentivo, confiança e pelos conselhos sempre valiosos.

À minha irmã, Camila, pelo apoio e por me mostrar que a distância não é capaz de enfraquecer certos sentimentos. À minha tia Célia, pela confiança e palavras de incentivo. À minha priminha Aninha, pelo incentivo e carinho.

Ao CNPq pelo apoio financeiro parcial.

A todos que direta ou indiretamente contribuíram para a realização deste trabalho.

E a Deus pelas oportunidades.

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Siglas

AMPS	*Advanced Mobile Phone Systems*
AT	*Amnesic Terminals*
CCS-IUP	*Cache Coherency Schema with Incremental Update Propagation*
CDMA	*Code Division Multiple Access*
EF	Estação Fixa
ESM	Estação de Suporte Móvel
ETSI	*European Telecommunications Standard Institute*
GAP	Gerenciador do Armazenamento Persistente
GC	Gerenciador de Conexão
GCCC	Gerenciador para Coerência de Cache no Cliente
GCCS	Gerenciador para Coerência de Cache no Servidor
GCORE	*Grouping with Cold Update-set Retention*
GNI	Gerenciador de Notificações de Invalidação
GSM	*Global System for Mobile Communications*
IR	*Invalidation Report*
IU	Interface com o Usuário
LS	Lista de Solicitações
NI	Notificação de Invalidação
NMT	*Nordic Mobile Telephone*
PDA	*Personal Digital Assistants*
RAM	*Randon Access Memory*
SO	Sistema Operacional
TACS	*Total Access Communication System*
TDMA	*Time Division Multiple Access*
TS	*Broadcasting Timestamp*
UCP	Unidade Central de Processamento
UM	Unidade Móvel
UMTS	*Universal Mobile Telecommunication System*
WLAN	*Wireless Local Area Network*

Resumo

A computação móvel está se tornando um ambiente de uso comum nos dias atuais. Desta forma é importante fornecer aos seus usuários serviços com uma qualidade similar à encontrada em ambientes cabeados.

Em ambientes de comunicação sem fio, computadores portáteis podem armazenar informações de forma local, visando reduzir a contenção na largura de banda e economizar energia. No entanto, a manutenção da consistência destas informações torna-se uma tarefa complexa, devido às desconexões e mobilidade dos clientes.

Nesta dissertação é apresentada a análise de desempenho de três estratégias para coerência de cache, utilizando um ambiente de configuração sem fio real. O trabalho de pesquisa realizado considera as estratégias *broadcasting timestamp*, *cache coherency schema with incremental update propagation* e *amnesic terminals*. Estas estratégias são baseadas em *broadcasts* periódicos de notificações de invalidação, sendo seus respectivos desempenhos analisados em um ambiente real.

Os testes experimentais observaram o comportamento das estratégias em relação ao impacto do tamanho da notificação de invalidação, intervalo de *broadcast* e quantidade de dados no cache nos dispositivos móveis. Os resultados demonstraram diferentes desempenhos para os parâmetros considerados. A estratégia *amnesic terminals* teve um desempenho interessante, no entanto, apresentou menos facilidades do que as outras abordagens analisadas.

Palavras Chave: Coerência de cache, Computação Móvel, Notificação de Invalidação, Redes sem fio.

Abstract

As mobile computing is getting a common environment on daily basis, it is important to provide services to users with a similar quality from wired networks.

A data cache mechanism can reduce the contention on a wireless network and also helps to save energy for mobile devices. However, maintenance of a cache consistency is a complex task because mobile users can frequently be disconnected from wireless networks.

In this work, we present a performance analysis of three different cache coherence strategies over a real wireless environment. Our research work considers the *broadcasting timestamp*, the *cache coherency schema with incremental update propagation* and the *amnesic terminals* strategies. These strategies are based on periodic broadcast of invalidation reports. The performance of these strategies is analyzed through an ordinary real environment.

The experimental results demonstrated advantages and constraints of the strategies based on effects from the invalidation report size, broadcast interval and mobile clients cache size. The results demonstrated different performances for the considered parameters. The amnesic terminals strategy had an interesting performance, but with fewer advantages compared to the other two approaches.

Keywords: Cache Coherence, Invalidation Report, Mobile Computing, Wireless Networks.

Capítulo 1

Introdução

O processamento distribuído é atualmente aplicado e desenvolvido nas mais diversas áreas sócio-econômicas. Esta forte aderência está relacionada a questões como o avanço tecnológico, economia e a necessidade do usuário obter um tempo de resposta cada vez menor. Em adição, características como compartilhamento de recursos, transparência e tolerância a falhas têm estimulado consideravelmente a utilização destes sistemas [COU 01].

Nos sistemas centralizados, o processamento pode não ser realizado onde os dados estão e sim onde o computador está. Desta forma, os sistemas distribuídos possuem diversas vantagens quando comparados aos sistemas centralizados. Neste contexto, podemos inserir a computação móvel, fazendo uso de características inerentes a este novo paradigma, como replicação e mobilidade, para trazer os dados até o usuário e permitir que este se movimente pela rede e, ainda assim, o processamento dos dados não seja interrompido.

A replicação dos dados nos dispositivos móveis torna-se uma importante forma de reduzir a quantidade de informações transferidas, assim como a latência de acesso aos dados e o tráfego na largura de banda [YUE 00, FIF 03, XU 03]. No entanto, este armazenamento local apenas reduz a latência das transações se o cliente encontrar as informações de seu interesse em seu cache local. Deste modo, é reduzido o tempo de acesso dos dispositivos móveis ao canal de *broadcast* [PIT 99]. Por outro lado, é im-

portante ressaltar que a ocorrência de desconexões é freqüente em configurações sem fio. Desta forma, torna-se difícil a verificação da validade dos dados armazenados localmente nos dispositivos móveis.

Métodos eficientes de coerência de cache são críticos para garantir que as aplicações em ambientes sem fio tenham um desempenho razoável [YUE 00]. Estes métodos, quando utilizados para suportar aplicações em redes sem fio (por exemplo, em banco de dados distribuídos) estão usualmente relacionados com a localização do usuário e com a requisição de elevadas larguras de banda. Contudo, os usuários móveis geralmente não possuem um comportamento padrão na rede. Em adição, a largura de banda disponível em redes de comunicação sem fio, algumas vezes é menor que nas redes cabeadas e possui elevado nível de interferências. Estes fatos tornam interessante a análise de métodos para coerência de cache e seus respectivos desempenhos em redes sem fio.

1.1 Motivação

A computação móvel introduz uma série de características inerentes que tornam o problema da coerência de cache uma desafiante área de pesquisa. Em adição, a pouca oferta de ambientes reais para análise de algoritmos de coerência de cache torna os trabalhos desenvolvidos nesta linha restritos a ambientes de simulação. Pelos motivos expostos, este trabalho visa desenvolver um ambiente de teste, onde possamos aplicar estratégias para coerência de cache. Este ambiente será utilizado para traçar uma análise comparativa entre três abordagens freqüentemente utilizadas em dispositivos móveis. Desta forma, neste ambiente serão confrontadas as estratégias *broadcasting timestamp* (TS), *cache coherence schema with incremental update propagation* (CCS-IUP) e *amnesic terminals* (AT).

1.2 Objetivos

1.2.1 Objetivo Geral

Nosso trabalho de pesquisa teve como meta traçar uma análise comparativa entre abordagens de coerência de cache em um ambiente móvel real.

1.2.2 Objetivos Específicos

Esta dissertação tem como objetivos específicos:

- Pesquisar estratégias de coerência de cache em dispositivos móveis;
- Selecionar as estratégias a serem implementadas;
- Identificar os testes padrão a serem realizados para análise das estratégias selecionadas;
- Implementar um ambiente de teste onde estas estratégias possam ser executadas;
- Obter resultados com base nos testes padrão pesquisados;
- Fazer uma análise comparativa entre os resultados obtidos;
- Sugerir a melhor técnica para um determinado contexto.

1.3 Estrutura do Documento

Esta dissertação está dividida em duas partes. A primeira parte apresenta uma revisão bibliográfica abordando os principais tópicos relacionados à coerência de cache em ambientes móveis. A segunda parte apresenta o ambiente desenvolvido e a avaliação do desempenho das estratégias selecionadas para implementação. Estas duas partes estão organizadas em 6 capítulos, conforme descrito a seguir.

- O capítulo 2 apresenta as principais características da computação móvel. Ainda são abordados os estados de operação de um elemento móvel, sua arquitetura e os principais modelos de comunicação utilizados em um ambiente móvel;

- O capítulo 3 aborda o problema da coerência de cache. Neste capítulo são apresentados os métodos para invalidação de cache, as formas do servidor gerenciar o cache nos clientes móveis, as formas de disseminação dos dados e os três algoritmos implementados;

- No capítulo 4 é apresentado o ambiente experimental desenvolvido. Neste capítulo são descritas as características principais deste ambiente; a descrição das opções de implementação estudadas, assim como as justificativas para a utilização de cada opção adotada; a arquitetura desenvolvida e o protótipo implementado;

- O capítulo 5 apresenta os resultados obtidos através da análise comparativa entre as estratégias de coerência de cache implementadas. Através desta análise é demonstrado o comportamento das estratégias em relação ao impacto do tamanho da notificação de invalidação, intervalo de *broadcast* e quantidade de dados no cache nos dispositivos móveis;

- Finalmente, no capítulo 6 são apresentadas as conclusões, problemas encontrados e propostas para trabalhos futuros.

Capítulo 2

Computação Móvel

A computação móvel tem se transformado em realidade devido aos recentes avanços na tecnologia da comunicação sem fio em conjunto com o maior poder de processamento dos computadores portáteis. Esta combinação tem tornado possível a utilização do paradigma *anytime-anywhere-computing* [YE 98, PER 01]. Neste paradigma, um dispositivo móvel pode receber um dado a qualquer hora, estando ele em qualquer lugar, desde que possa ser estabelecida uma conexão com a rede. Desta maneira, um usuário deste ambiente terá acesso a serviços independente de onde ele esteja localizado, sendo permitida a sua mudança de localização.

Conforme mencionado em SATYANARAYANAN [SAT 01a], muitos dos princípios básicos da computação distribuída continuam aplicáveis à computação móvel. No entanto, algumas restrições impostas pela mobilidade forçam o desenvolvimento de técnicas específicas para este novo paradigma. Dentre estas restrições, podemos destacar: a imprevisível variação na qualidade da rede sem fio, menor robustez e confiabilidade dos elementos móveis em relação a elementos estáticos, limitações dos recursos locais impostas pelo peso e tamanho dos dispositivos móveis, e o fato dos elementos móveis contarem com uma fonte de energia finita para seus equipamentos. No entanto, a computação móvel amplia o conceito tradicional de computação distribuída [MAT 98]. Isto é possível graças ao fato da comunicação sem fio eliminar a necessidade do usuário manter-se conectado a uma infra-estrutura de comunicação fixa e, na maior parte dos ca-

sos, estática. Desta forma, o paradigma da computação móvel tem afetado conceitos e modelos tradicionais em várias áreas da ciência da computação. Por exemplo, na área de redes de comunicação, as redes podem estar disponíveis em todos os lugares, garantindo, desta forma, a conectividade aos seus usuários independente de sua localização física. Redes de comunicação que possuem estas características são conhecidas como redes *ad hoc*. Outro exemplo pode ser observado na área da banco de dados, onde é introduzido o conceito dos dispositivos móveis acessarem os seus dados a partir de qualquer lugar (*anywhere*) e a qualquer momento (*anytime*).

Como previsto por LIU et al. [LIU 96], através da computação móvel já é possível realizar uma série de tarefas, como por exemplo, receber e-mails, faxes, fazer acesso a banco de dados, em adição ao tradicional serviço de comunicação de voz.

Desta forma, esforços para explorar novas tecnologias no que diz respeito à computação móvel e comunicação pessoal tem sido realizados em vários projetos. No entanto, a mobilidade dos computadores ainda introduz vários desafios, envolvendo o projeto e a implementação de sistemas.

Neste capítulo, abordaremos algumas das dificuldades impostas pela computação móvel, assim como os seus principais fundamentos.

2.1 Características

Embora a rede sem fio com clientes móveis seja essencialmente um sistema distribuído, existem algumas características que tornam este sistema uma produtiva área de pesquisa. Nesta seção serão apresentadas algumas destas características.

2.1.1 Portabilidade

Os computadores portáteis são capazes de substituir agendas eletrônicas, calculadoras, além de poderem ser integrados a telefones celulares. Com o avanço da tecnologia no desenvolvimento de computadores móveis, estes dispositivos tem se tornado cada vez menores e mais leves. No entanto, conforme mencionado por CUNHA

[dC 03], a portabilidade dos dispositivos móveis introduz algumas dificuldades de projeto no que diz respeito à energia, interface e capacidade de armazenamento.

As características e limitações dos dispositivos móveis quanto à portabilidade, de acordo com PITOURA & SAMARAS [PIT 98], são descritas a seguir:

- *Os elementos móveis são pobres em recursos quando comparados com os elementos estáticos*: Os dispositivos móveis muitas vezes são equipados com pouca memória do tipo RAM (*Randon Access Memory*) e seus processadores são mais lentos que o de computadores convencionais. Em adição, eles podem possuir pouca memória não volátil, a interface com o usuário é mais limitada, o monitor é pequeno ou tem pouca capacidade em seu *buffer*. Os dispositivos de entrada de dados em dispositivos móveis também são, na maioria das vezes, pequenos e limitados (por exemplo, teclados de aparelhos celulares);

- *Os elementos móveis contam com pouca capacidade em suas baterias*: Estes elementos dependem da energia fornecida por baterias. Normalmente as baterias possuem capacidades limitadas e em alguns lugares existe dificuldade para recarga da bateria;

- *Elementos móveis são poucos robustos*: Desta forma, podem serem danificados, perdidos ou roubados facilmente.

2.1.2 Mobilidade

A localização de um dispositivo móvel, assim como seu ponto de acesso a rede fixa, são alterados à medida que este se movimenta pela rede. Esta mobilidade provoca um estilo novo de computação que afeta tanto as redes de comunicação sem fio quanto as redes fixas, pois exige a criação de novas metodologias para gerenciamento dos dados. Como conseqüência da mobilidade, destaca-se a complexa e difícil tarefa de manter as bases de dados operacionais atualizadas. Desta forma, a mobilidade introduz restrições que eram inexistentes na computação tradicional formada por computadores

estáticos. Entre estas restrições, podemos citar preocupações com o tempo de vida da bateria, o risco de perda dos dados e a confiabilidade da conexão com a rede.

De acordo com PITOURA & SAMARAS [PIT 98], as principais conseqüências da mobilidade são:

- *A configuração dos sistemas que incluem elementos móveis não é estática*: Os algoritmos tradicionais para processamento distribuído precisam ser reprojetados devido à ausência de uma topologia fixa da rede com os elementos móveis. A distribuição da carga e a localidade dos elementos móveis também pode mudar dinamicamente;

- *Gerência da Localização*: O custo para localizar um elemento móvel é adicionado ao custo de cada comunicação envolvendo este elemento. Considerando que os dados de localização podem se alterar de forma muito rápida é necessário que estruturas de dados, algoritmos e planos de consultas eficientes sejam planejados e desenvolvidos para representar, gerenciar e consultar a localização dos elementos móveis;

- *Heterogeneidade*: O desempenho e a confiabilidade de um elemento móvel pode variar de acordo com a sua conectividade. O número de serviços disponíveis para um elemento móvel também pode variar de uma localização para outra, em função das características da célula e da estação de base na qual ele se conecta, assim como os recursos disponíveis nos elementos móveis variam. Como exemplo, pode-se citar a sua capacidade de memória, o tamanho de sua tela e o tempo de duração de sua bateria.

2.1.3 Adaptação

Os dispositivos móveis necessitam de mecanismos para gerenciamento das alterações do ambiente, para que possam reagir às mudanças e continuem oferecendo serviços aceitáveis. Apenas através de uma adaptação efetiva, aplicações nos computadores móveis podem superar desafios como imprevisíveis variações na qualidade da rede,

larga diferença na disponibilidade dos serviços, consumo de bateria remotos e limitações nos recursos locais impostas por restrições de peso e tamanho [SAT 01b].

2.1.4 Gerência de Energia

Devido as restrições do poder da bateria dos dispositivos móveis , estes podem estar freqüentemente desconectados. Desta maneira, atividades de leitura, consulta e escrita podem ser interrompidas durante o período de desconexão.

Progressos na tecnologia das baterias têm permitido intervalos maiores entre uma recarga e outra. No entanto, o alto custo de energia utilizado por um computador móvel para transmitir e receber dados diminui o tempo de vida da bateria.

Vários estudos estão sendo desenvolvidos, com o intuito de diminuir as restrições impostas pela gerência de energia. Dentre eles, destacamos FLINN & SATYANARAYANAN [FLI 99], que apresenta uma ferramenta para mapear o uso de energia pelas aplicações. Com esta ferramenta é possível determinar qual a fração de energia consumida por um processo em um período de tempo. Desta forma, pode-se identificar qual componente do sistema é responsável pelo maior consumo de energia. Em CHIASSERINI et al. [CHI 02] é realizado um estudo sobre o comportamento intrínseco das baterias, utilizando esse conhecimento no desenvolvimento de um modelo estocástico de bateria e em novos protocolos *energy-efficient*.

2.1.5 Fraca Conectividade

Os clientes móveis podem não estar conectados à rede continuamente. Eles também podem se desconectar de uma célula para se conectar à outra. Esta fraca conectividade causa várias limitações que não estão presentes quando a conectividade é forte.

O fato da comunicação entre uma unidade móvel e uma estação de suporte móvel (ESM) ser através de uma rede de comunicação sem fio, tem como principais conseqüências a conectividade fraca e intermitente. As redes sem fio são monetariamente mais caras e oferecem menores larguras de banda. Em adição, elas possuem uma latência

maior e são menos confiáveis do que as redes cabeadas.

A conectividade dos dispositivos móveis na rede sem fio é variável. Desta forma, a qualidade da conexão pode variar em função de interferências na comunicação, da distância da unidade móvel com a ESM na qual a unidade esta conectada e ao compartilhamento da ESM por várias unidades móveis.

As freqüentes desconexões e a mobilidade dos clientes introduzem o problema de incoerência do cache com relação aos dados armazenados no servidor [YUE 00].

2.1.6 Interface Limitada

O tamanho da tela em dispositivos móveis tem forçado a realização de pesquisas sobre novas interfaces para retorno de informações [BAR 99]. Esta característica pode representar limitações na memória e poder de bateria. Algumas unidades portáteis, como por exemplo, PDAs (*Personal Digital Assistants*), possuem telas muito pequenas.

2.1.7 Replicação de Dados

Em ambientes de computação móvel, dispositivos móveis com capacidade de conexão sem fio são carregados por usuários que querem se mover livremente por toda parte. Embora isto dê aos usuários uma comodidade importante em termos de mobilidade, a rede sem fio, em muitos casos, é custosa, possui largura de banda, às vezes, limitada e fornece conexões com menor qualidade e maior interferência do que as redes cabeadas. Conseqüentemente, desconexões, sejam elas intencionais ou não, ocorrem ocasionalmente. Desta forma, os dados necessários podem ser replicados nos clientes móveis para melhorar a disponibilidade durante as desconexões [LOH 00]. Devido a estes fatos, a habilidade de replicar dados em ambientes móveis é essencial para que unidades móveis possam continuar a processar os dados mesmo quando desconectadas [BAR 99].

2.1.8 Capacidade de Armazenamento

Outro importante recurso nos dispositivos móveis é a sua capacidade de armazenamento de informações [HUA 94, CHA 03] . Devido a limitação de armazenamento existente em alguns tipos de dispositivos móveis, torna-se difícil criar réplicas de banco de dados volumosos, particularmente em dispositivos como PDAs.

A capacidade de armazenamento é um importante parâmetro de projeto em aplicações móveis. No entanto, conforme comprovado em YE et al. [YE 98], o tamanho do cache não possui influência significativa no seu desempenho.

2.1.9 Assimetria na Comunicação

A comunicação entre estações móveis e os servidores fixos é de natureza não simétrica [MUR 98], ou seja, a largura de banda na direção *downlink* (servidor - para - cliente) é muito maior que na direção *uplink* (cliente - para - servidor) [BAR 99, HOU 01]. Em alguns casos, os clientes não possuem capacidade de enviar mensagens. Além disso, a transmissão de dados consome mais energia do que o recebimento dos dados [CAI 99].

Considerando estes fatos, torna-se mais custoso para os clientes enviar mensagens do que recebê-las. Desta forma, a comunicação assimétrica, juntamente com as restrições de energia que uma unidade móvel possui, faz com que o modelo de *broadcasting* de dados para os clientes seja uma proposta mais atrativa do que esperar que os clientes solicitem os itens de dados.

2.2 Estados de Operação de um Elemento Móvel

As unidades móveis podem assumir vários graus de operação. A figura 2.1, extraída de PITOURA & BHARGAVA [PIT 94] resume os diferentes estados de operação que podem ser assumidos por um elemento móvel.

As unidades móveis podem estar totalmente conectadas, desconectadas, parcialmente conectadas ou em modo de *cochilo* (*doze mode*) [IMI 94, BAR 95, GUP 01].

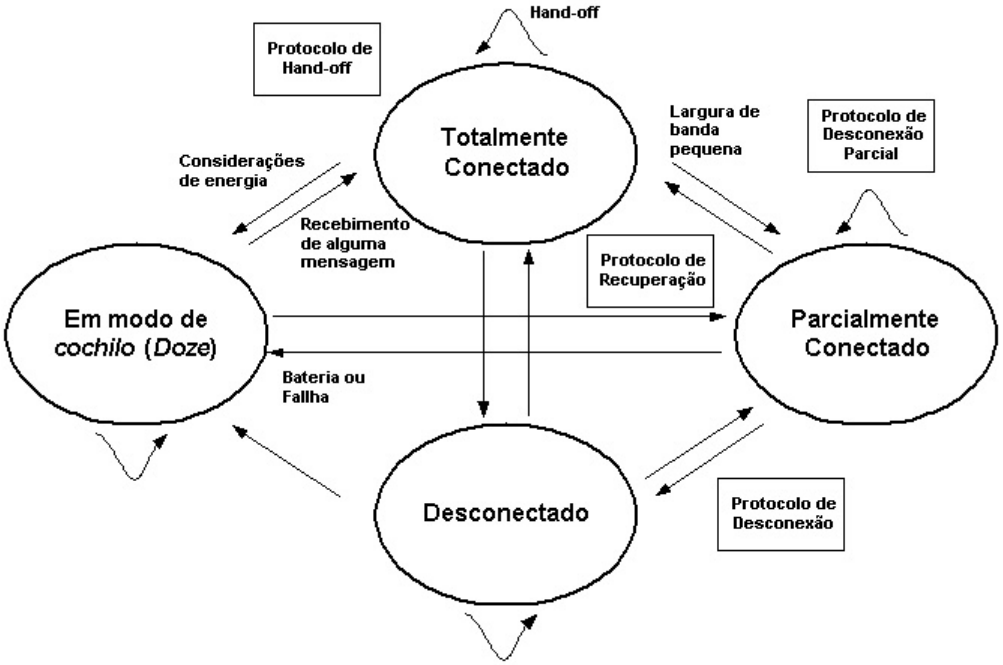

Figura 2.1: Estados de operação de um elemento móvel [PIT 94]

Um cliente móvel está conectado quando está em completo modo de operação. De forma contrária, um cliente está desconectado quando está inacessível ao resto da rede.

Em um ambiente móvel a maior parte das desconexões podem ser detectadas. Desta forma, um protocolo especial pode ser projetado para lidar com estas desconexões. O cliente móvel pode executar um protocolo de desconexão antes de se desligar fisicamente da rede. Este protocolo visa garantir a transferência suficiente de dados para que o mesmo continue a operar durante o estado de desconexão.

Um cliente móvel pode trocar seu estado de parcialmente conectado para executar no estado parcialmente desconectado. Enquanto permanecer neste estado, toda a comunicação com a rede deve ser limitada. Finalmente, enquanto um cliente móvel esta operando em um modo, ele pode entrar em uma nova célula. Neste caso, toda a informação pertinente ao estado do cliente móvel deve ser transferida à estação de base da nova célula, através da execução de um protocolo de hand-off [COR 03].

Para conservar a sua bateria, o cliente móvel pode operar em um estado

intermediário. Neste caso, este cliente está em modo de *cochilo*. No estado *doze*, a UCP está trabalhando em baixa proporção e a unidade móvel pode ser *acordada* por uma mensagem externa. Agindo desta maneira, o cliente consome menos energia.

De acordo com CHAN & RODDICK [CHA 03], os modos de operação desconectado e *cochilo* podem ser invocados propositadamente por um cliente para permitir a conservação de recursos limitados. Segundo IMIELINSKI et al. [IMI 94], é vantajoso que *palmtops* possam permanecer em modo de *cochilo* a maior parte do tempo e apenas entrarem no modo ativo (conectado), quando for aguardada a chegada de dados de interesse.

2.3 Tecnologia sem Fio

Em muitas aplicações, redes de comunicação sem fio podem eliminar o alto custo de instalação e manutenção de sistemas cabeados [HOU 01]. Outro importante papel das redes que suportam conexões sem fio é a capacidade de permitir a conexão de um elevado número de endereços fixos, e sem fio, utilizando a sua infra-estrutura.

2.3.1 Breve Histórico

As redes celulares tiveram seu inicio por volta de 1980 [AGR 99]. A tecnologia empregada na primeira geração (1G) iniciou o rápido crescimento da indústria de telefonia celular. Os sistemas predominantes nesta geração são o *Advanced Mobile Phone Systems* (AMPS), *Total Access Communication System* (TACS) e *Nordic Mobile Telephone* (NMT). Estas redes são voltadas para a comunicação de voz, e caracterizam-se por não possuírem fio, serem móveis e pessoais. No entanto, os sistemas analógicos não forneceram a qualidade de sinal desejada para um sistema de voz. Desta forma, foi necessário o desenvolvimento de novas tecnologias.

A segunda geração (2G) melhorou a capacidade, segurança e qualidade em relação às tecnologias utilizadas na 1G. A 2G utiliza os sistemas *Global System for Mobile Communications* (GSM), *Time Division Multiple Access* (TDMA) e *Code Divi-*

sion Multiple Access (CDMA). As redes de segunda geração tornaram-se grandemente difundidas. No entanto, elas não satisfizeram a alta velocidade de acesso às redes [AGR 99]. Uma evolução natural foram as redes de 2,5 G. Entretanto, as redes 2,5 G ainda não eram adequadas ao tráfego de serviços multimídia.

A terceira geração de sistemas móveis chamados *Universal Mobile Telecommunication System* (UMTS) [RAP 95, PAL 99] está sendo padronizada pelo *European Telecommunications Standard Institute* (ETSI). A UMTS representa uma evolução em termos de serviços e velocidade em relação às redes móveis utilizadas atualmente [UMT 04]. A UMTS visa fornecer aos usuários móveis novos serviços para *broadband* (bandas de larga freqüência) como multimídia sem fio, incluindo vídeo em tempo-real e alta velocidade de dados.

2.3.2 Redes Locais sem Fio

As redes locais sem fio (*Wireless LAN* ou WLAN) tornaram-se uma alternativa às redes convencionais com fio. Elas fornecem as mesmas funcionalidades das redes cabeadas, mas de forma flexível, de fácil configuração e com boa conectividade.

De acordo com DANTAS [DAN 02], uma rede local sem fio tem um grande conjunto de parâmetros de configuração que afetam o desempenho e a interoperabilidade da rede. Desta forma, existe o problema da qualidade da conexão sem fio. Esta qualidade pode variar em função do local e das falhas de comunicação entre os terminais remotos e seus clientes. Pode-se mencionar ainda, limitações quanto à segurança da comunicação entre os clientes móveis e os servidores fixos. No entanto, as WLANs são uma boa alternativa quando existe a necessidade de instalação rápida ou temporária de uma rede local.

2.4 Arquitetura da Computação Móvel

Um sistema de computação móvel pode ser definido, de uma forma genérica, como um conjunto de computadores móveis e computadores fixos. Os computa-

dores móveis se comunicam uns com os outros através de uma interface de comunicação sem fio, enquanto que os computadores fixos se comunicam com os outros através de rede cabeada.

De modo a suportar a comunicação de computadores móveis, a área geográfica abrangida pelo sistema é dividida em células. A comunicação sem fio em uma célula é suportada por uma ESM. A ESM é um computador fixo na rede fixa que possui interface sem fio. Ela provê uma cobertura de comunicação para uma célula. Um computador móvel pode se comunicar diretamente com a ESM, e vice versa, somente se estiver localizado dentro de uma célula. Na figura 2.2 é apresentado um ambiente móvel genérico, conforme BARBARÁ & IMIELÍNSKI [BAR 95].

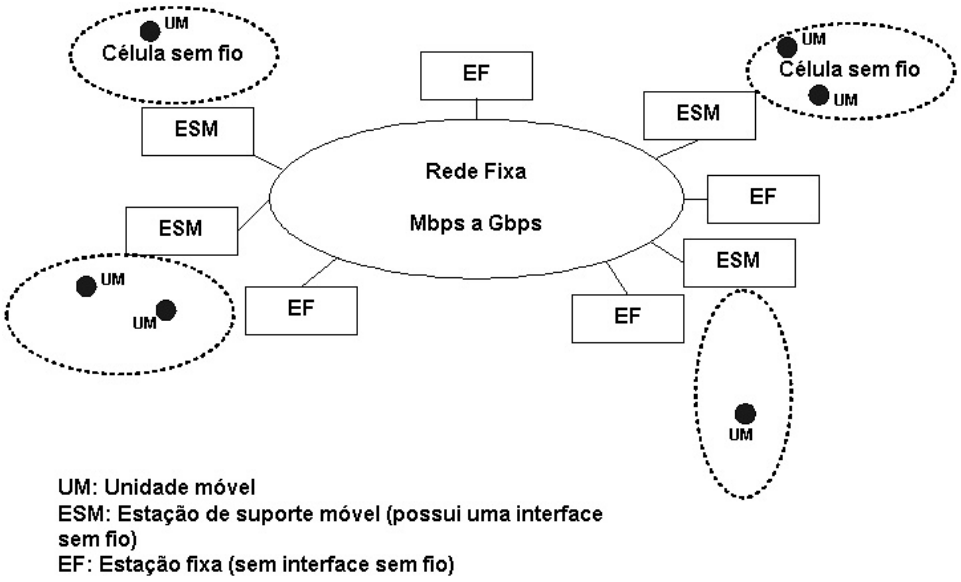

Figura 2.2: Um ambiente móvel [BAR 95]

De acordo com DUNHAM & HELAL [DUN 95], um sistema de computação móvel pode ser visto como um tipo dinâmico de sistema distribuído onde as ligações entre os nodos da rede são alterados dinamicamente. Estas ligações representam a conexão entre a unidade móvel e a estação de suporte móvel na qual as unidades estão conectadas.

2.5 Modelos de Comunicação na Computação Móvel

Vários modelos de comunicação podem ser aplicados em sistemas distribuídos tradicionais. No entanto, o ambiente de computação móvel é restrito em vários aspectos, dificultando a utilização de alguns destes modelos. Dentre os modelos utilizados em ambientes de computação móvel, destacamos os modelos cliente-servidor, fim-a-fim e agentes móveis.

2.5.1 Modelo Cliente-Servidor

No modelo (arquitetura) cliente-servidor, um cliente é geralmente um equipamento de usuário que possui capacidades de interface e processamento local. O servidor é um equipamento que pode fornecer serviços para as estações clientes.

O modelo cliente-servidor não permite interação direta entre os clientes [RAT 96]. Desta forma, os clientes se comunicam uns com os outros através do servidor. Em ambientes convencionais cliente-servidor, a largura de banda de transmissão é comparada à largura de banda do acesso ao disco [LEO 97]. No entanto, esta característica não pode ser obtida em ambientes móveis.

Na computação móvel, a unidade móvel atua como um cliente requisitando serviços dos servidores localizados na rede fixa. Nestes casos, os dados e as funcionalidades estão distribuídos através de vários servidores em diferentes estações fixas. Estas estações podem comunicar-se entre si para atender as solicitações dos clientes. Em muitos casos, o servidor é replicado em diferentes *sites* na rede fixa para aumentar a disponibilidade nos casos de falhas nos *sites* ou na rede de comunicação.

2.5.2 Modelo Fim-a-Fim

No modelo fim-a-fim não existe distinção entre estações que exercem o papel de cliente e estações que exercem o papel de servidor. Desta forma, cada estação tem a capacidade de se comportar tanto como um cliente quanto como um servidor. Conforme mencionado em RATNER et al. [RAT 96], soluções fim-a-fim permitem que

qualquer conjunto de réplicas se comunique diretamente com qualquer outro. Entretanto, segundo LAM et al. [LAM 01], em um ambiente móvel a performance do sistema pode ser seriamente afetada pela performance da rede, especialmente quando alguns dos itens de dados requisitados pela transação estiverem localizados nas estações móveis.

2.5.3 Modelo de Agentes Móveis

Agentes móveis são processos disparados de um computador origem que migram para outros computadores a fim de executar uma tarefa específica [CHE 95]. Desta forma, os agentes móveis podem vagar de computador a computador, procurando por um que contribua com a execução de uma tarefa, ou coletando informações.

Cada agente móvel possui instrução, dados e um estado de execução. O agente móvel executa de forma autônoma e independente da aplicação que o invocou. Ao chegar ao seu destino, o agente é autenticado, preparado para execução na entidade destino e executado. Objetivando executar sua tarefa, o agente móvel pode se transferir para outra estação, criar e disparar novos agentes ou interagir com outros agentes. Ao terminar sua tarefa, o agente móvel envia os resultados para a aplicação que o invocou ou para um servidor especificado inicialmente [MAT 98].

De acordo com CORTES & LIFSCHITZ [COR 03], dois fatores principais motivam a utilização dos agentes móveis na computação móvel. O primeiro é o fato dos agentes móveis oferecem um método assíncrono eficiente para procurar informações ou serviços, de forma rápida em uma rede. Neste modelo, clientes móveis são lançados na rede e procuram em toda parte pela informação ou serviço desejado. Em segundo, os agentes móveis suportam a conectividade intermitente, as redes lentas e os equipamentos com poucos recursos. Este segundo fator torna a utilização dos agentes móveis bastante atraente para a computação móvel.

Durante um período de desconexão o cliente móvel envia um agente móvel para a rede fixa e este agente age como um substituto da aplicação na rede fixa, permitindo interação durante a desconexão. De forma oposta, o agente móvel é carregado no cliente móvel, oriundo da rede fixa, quando acontece uma nova conexão. Quando

em fraca conectividade, todo o tráfego de comunicação através do *link* sem fio reduz a possibilidade de submissão de um grande número de mensagens pelos agentes fixos e na obtenção de seus resultados. Por este motivo, os agentes móveis deslocam a carga de computação dos clientes móveis, pobres em recursos, para a rede fixa.

Capítulo 3

Coerência de Cache

No contexto da computação móvel podemos definir cache como uma forma de armazenar os dados mais relevantes na memória, reduzindo a necessidade de acessar dispositivos de armazenamento e aumentando o desempenho das operações de leitura e escrita dos dados. Desta forma, a utilização do cache permite que os dispositivos móveis realizem o processamento de seus dados de forma mais rápida e consumindo uma quantidade menor de energia.

O armazenamento das informações relevantes no cache dos dispositivos móveis permite que múltiplos dispositivos consultem e alterem um mesmo conjunto de dados. Em adição, dispositivos móveis podem permanecer desconectados por longos períodos de tempo, se movimentar de uma área de cobertura para outra e acessar diferentes servidores de dados. Devido ao perfil dos dispositivos móveis, qualquer solução para o problema das desconexões deve ser baseada no cache. Enquanto estiver conectado, o dispositivo móvel deve ser carregado com todas as informações necessárias à continuação do processamento, para que este não seja interrompido em caso de desconexões.

O armazenamento em cache é uma importante técnica para reduzir a disputa no canal de comunicação entre o cliente e o servidor [BAR 95, CAI 99]. Além disso, este armazenamento pode ser uma forma efetiva de economizar energia, uma vez que não é necessário despender energia para a transmissão dos dados. O armazenamento dos dados nos dispositivos móveis pode ainda representar economia de custos, depen-

dendo do tipo de bilhetagem (tarifa) que esteja sendo aplicado.

No entanto, considerando que os dados sejam armazenados localmente nos dispositivos móveis, e que estes dispositivos estão sujeitos à desconexões e alterações de uma mesma informação de forma simultânea, faz-se necessária a utilização de mecanismos para prover a consistência destes dados. A utilização destes mecanismos é aplicada para que seja possível se certificar de que os dados que foram replicados para os clientes estejam consistentes com os dados armazenados no servidor. Contudo, são muitos os problemas relacionados com a coerência de cache. Neste capítulo serão abordados alguns destes problemas e algumas formas de estabelecer a coerência do cache.

3.1 Consistência dos Dados

Com a replicação dos dados nos dispositivos móveis são geradas múltiplas cópias de uma determinada informação. Este comportamento introduz o problema da consistência entre estas cópias.

O desempenho e a funcionalidade de sistemas distribuídos depende do nível de consistência fornecido [KOR 96]. Desta forma, os benefícios da replicação dos dados nas unidades móveis apenas podem ser obtidos se a consistência destes dados for mantida, ou seja, se os dados do cache forem atualizados quando os valores originais forem alterados.

3.1.1 Propriedades ACID

As propriedades ACID podem ser utilizadas para prover consistência e mecanismos confiáveis de computação [HAE 83, MUR 98, DIR 00]. Estas propriedades aplicadas em conjunto garantem a integridade e consistência das informações tratadas pelas transações.

- Atomicidade: Esta propriedade determina que ou todas as operações delimitadas na transação são refletidas corretamente ou nenhuma delas será. Desta forma, vê-se a transação como uma ação única. Se uma transação executa até o seu final, os

resultados obtidos serão efetivados; caso contrário, nenhum resultado obtido dentro da transação até o instante da ocorrência da falha será efetivado.

- Consistência: Esta propriedade determina que uma transação deve preservar a consistência do banco de dados. Ou seja, mesmo no caso do processamento concorrente de várias transações, o resultado será o mesmo que seria obtido caso estas várias transações fossem executadas de forma seqüencial.

- Isolamento: Esta propriedade é baseada na seriabilidade. Embora diversas transações possam ser executadas de forma concorrente, o sistema garante que, para todo par de transações T_i e T_j, T_i tem a sensação de que T_j já terminou sua execução antes de T_i começar, ou que T_j começou a sua execução após T_i terminar. Desta maneira, a execução concorrente deve equivaler a alguma execução serial de um conjunto de transações.

- Durabilidade: Esta propriedade garante que os resultados obtidos por uma transação que foi efetivada persistem, até mesmo se houver falhas no sistema.

3.1.2 Modelos para Consistência de Cache

- Consistência Forte: Um cliente possui consistência forte sobre um determinado item de dado quando este cliente requer notificação imediata sobre qualquer alteração que ocorra neste item. Desta forma, atualizações executadas no servidor são imediatamente propagadas para todos os clientes que requeiram consistência total. Neste modelo existe a necessidade dos dados estarem sempre consistentes.

- Consistência Fraca: é utilizada em casos onde os clientes móveis não necessitem serem notificados sobre as alterações no servidor à todo momento. As notificações sobre alterações podem ser atrasadas ou até mesmo omitidas se as alterações forem consideradas insignificantes ou irrelevantes. Neste caso, existe uma tolerância em relação à utilização de dados desatualizados ou apenas algumas alterações específicas são relevantes para a aplicação.

Em VORA et al. [VOR 02] estas duas classes de consistência são diferenciadas, buscando melhorar o desempenho do gerenciamento de cache em ambientes móveis.

A consistência dos dados pode ainda ser relaxada, ou seja, conforme mencionado em LAUZAC & CHRYSANTHIS [LAU 02], em ambientes móveis, o usuário pode querer introduzir retardos ou troca da exatidão dos dados, buscando reduzir os serviços de troca, ou minimizar os recursos requeridos. De acordo com ADVE & GHARACHORLOO [ADV 96], os modelos para relaxamento da consistência geralmente enfatizam a otimização no sistema que estes possibilitam.

Conforme KORDALE & AHAMAD [KOR 96], algumas aplicações podem preservar a correção dos dados utilizando fraca consistência, como forma de obter uma melhor performance. Em adição, a consistência forte não pode ser provida em sistemas onde os clientes ficam temporariamente desconectados, sendo que, esta desconexão pode ocorrer voluntariamente ou involuntariamente em ambientes móveis. Deste modo, a consistência forte não é possível, pelo fato de ser requerido a comunicação com os clientes móveis, após ser realizado um acesso e estes clientes podem estar fora da área de cobertura, ou desconectados. No entanto, é importante ressaltar que, em alguns casos, até um pequeno grau de inconsistência pode gerar efeitos prejudiciais, ou seja, a margem de erro aceitável pelo usuário é muito pequena [LAM 01].

3.1.3 Granularidade de Cache

Em ambientes de computação móvel geralmente não é praticável a replicação de conjuntos de dados inteiros em cada um dos clientes móveis. Isto acontece como resultado de uma reduzida capacidade de armazenamento local nestes dispositivos.

Visto que computadores móveis são designados para serem portáteis, geralmente eles são fisicamente pequenos e muitas vezes incapazes de guardar grandes quantias de dados. Desta forma, estes dados passam a ser usualmente manipulados em bancos de dados centralizados [CHA 03]. Em adicional, como muitas das redes de comunicação sem fio requeridas na computação móvel possuem largura de banda limitada,

existe a necessidade de que o gerenciamento da largura de banda utilizada seja realizado cuidadosamente.

3.2 Técnicas de Invalidação

Uma forma de assegurar a exatidão de transações é invalidar, isto é, abortar transações onde os valores dos dados lidos correspondem à diferentes estados do banco de dados [PIT 99]. Uma maneira de alcançar a consistência dos dados é através do *broadcast* de um *timestamp* ou número de versão junto com o valor de cada item de dado. Outra forma é através do envio de notificações de invalidação ou validação, conforme tratado a seguir.

3.2.1 Notificação de Invalidação

Esta é uma abordagem utilizada para que os dados armazenados localmente nas unidades móveis possam ser reutilizados após desconexões. Nesta abordagem, o servidor envia um *broadcast* periódico contendo uma notificação de invalidação. Nesta notificação são indicadas as alterações/invalidações que devem ser realizadas nos dados locais dos dispositivos móveis.

De acordo com CAO [CAO 02], soluções baseadas em notificação de invalidação são atrativas. Isto se deve ao fato destas soluções atingirem qualquer número de clientes que tenham recebido a notificação. No entanto, estas soluções possuem algumas desvantagens. Conforme mencionado em BARBARÁ [BAR 99], dependendo da técnica utilizada, notificações de invalidação podem causar "falsas negativas", fazendo com que o cliente elimine um item de dado de seu cache quando na realidade o item ainda era válido. De acordo com CAO [CAO 02], pode ocorrer uma longa latência para um cliente responder uma consulta, caso ele necessite receber uma próxima notificação para se certificar de que o seu cache está válido. Pode ocorrer também o fato do servidor incluir nas notificações, informações que não sejam utilizadas por nenhum cliente. Desta maneira, o servidor transmitirá dados sem necessidade, danificando a largura de banda.

E ainda, mesmo que várias unidades móveis possuam os mesmos itens de dados, todos deverão questionar e conseguir os dados do servidor separadamente.

O tamanho das notificações de invalidação cresce de forma proporcional ao número de itens de dados alterados. Este comportamento é ineficiente quando existe uma grande quantidade de itens alterados [LAI 03]. Notificações contendo informações detalhadas podem não ser desejáveis pelo fato destas poderem ser muito longas, consumindo muita largura de banda. Por outro lado, falsas invalidações podem ocorrer se informações detalhadas das alterações não forem disponibilizadas nas notificações [HOU 01].

3.2.2 Notificação de Validação

O conceito das notificações de validação visa tratar o problema da ineficiência que ocorre quando existe uma grande quantidade de itens alterados. Neste caso, o tamanho das notificações de invalidação torna-se muito grande [LAI 03].

Ao contrário das notificações de invalidação, nesta abordagem, são mantidas informações sobre os itens de dados que não foram alterados durante o último intervalo de *broadcast*. As notificações de validação podem ser utilizadas quando o tamanho das mensagens das notificações de invalidação se tornar muito extenso.

Nesta abordagem não é necessário janela de *broadcast*. Isto ocorre porque o *timestamp* de itens válidos pode ser usado para determinar alterações que tenham ocorrido anteriormente.

A notificação de validação é a melhor opção a ser utilizada em situações onde de n itens de dados, mais da metade destes itens tiverem sido alterados.

3.3 Abordagens para Invalidação de Cache

Um dispositivo de comunicação móvel que se comunica com um servidor de dados através de uma rede sem fio está sujeito a desconexões. De modo a suportar estas desconexões, precisam ser implementadas estratégias para invalidação de cache.

Estas estratégias são rigorosamente afetadas pela desconexão e mobilidade dos clientes, principalmente se o servidor não possuir conhecimento sobre quais clientes estão freqüentemente localizados em sua célula e se estes clientes estão conectados.

Conforme mencionado em LEONG & SI [LEO 97], mecanismos convencionais de coerência de cache são baseados em ambientes fortemente acoplados. Nestes ambientes, o servidor tem completo conhecimento do armazenamento local de cada cliente, sendo ele responsável por notificar todos os clientes relacionados sempre que uma alteração for efetuada. No entanto, em uma rede sem fio cada cliente móvel é fracamente acoplado com o servidor de banco de dados, conectando-se ou desconectando-se da rede livremente e freqüentemente. Desta maneira, não é interessante para o servidor manter-se informado de todas as cópias armazenadas em cada cliente móvel. Sendo assim, cada cliente deve ter uma responsabilidade maior em manter a coerência dos seus dados e determinar se um dado armazenado localmente deve ser invalidado.

Uma vez definido que uma unidade móvel pode armazenar dados, é possível considerar as formas de gerenciamento de cache descritas a seguir.

3.3.1 Abordagem *Stateful*

Nesta abordagem o servidor tem conhecimento de quais unidades móveis atualmente residem em sua célula, assim como possui conhecimento do estado do cache destas unidades. Desta forma, cada vez que ocorre uma alteração no banco de dados, o servidor envia mensagens de invalidação para as unidades afetadas.

Abordagens *stateful*, também conhecidas como *eager* [LEO 97], são escaláveis, mas implicam em um *overhead* (número de mensagens trocadas e o tempo em que um processador tem que esperar para prover a coerência do dado) significante devido ao gerenciamento dos dados.

A abordagem *stateful* possui grande complexidade [CAI 99]. Isto se deve a vários motivos, como por exemplo, o servidor precisa ter conhecimento de todas as unidades móveis que saíram de uma determinada célula ou estão desconectadas. Devido a esta complexidade envolvida na abordagem *stateful*, grande parte dos trabalhos realizados

na área de coerência de cache são baseadas na abordagem *stateless* [BAR 95, CAI 99].

3.3.2 Abordagem *Stateless*

Nesta abordagem o servidor não tem conhecimento de quais unidades móveis atualmente residem em sua célula, assim como não possui conhecimento do conteúdo e tempo do cache destas unidades. Neste caso, o servidor mantém um histórico das alterações realizadas e fornece estas informações para os dispositivos móveis através de *broadcasts* periódicos ou quando o dispositivo fizer a solicitação.

Devido à assimetria na comunicação e à fraca capacidade de transmissão das unidades móveis é interessante que os servidores tenham um papel mais ativo na manutenção da coerência de cache [HOU 01]. De acordo com LEONG & SI [LEO 97], abordagens de invalidações imediatas não parecem ser praticáveis em ambientes móveis onde clientes conectam e desconectam freqüentemente da rede sem fio. Neste caso, o servidor tem dificuldades em se manter informado sobre todos os clientes que possuem uma cópia do item modificado. Desta forma, abordagens envolvendo *broadcasts* de mensagens de invalidação para os dispositivos móveis tem sido bastante exploradas [BAR 95, CAI 99, BAR 99, HOU 01, LAI 03].

3.4 Disseminação de Dados

A disseminação de dados pode ser definida como a entrega de dados de um conjunto de produtores para um conjunto maior de clientes [BAR 99]. Em um ambiente de computação móvel, ela é amplamente utilizada devido à assimetria na comunicação entre clientes e servidores, visto que a largura de banda no sentido servidor-para-cliente (*downlink*) é muito maior que no sentido cliente-para-servidor (*uplink*). A disseminação de dados pode ser classificada quanto ao conteúdo, ao modo de *broadcast* e ao tipo, conforme descrito a seguir.

3.4.1 Conteúdo

De acordo com a taxonomia proposta por CAI et al. [CAI 97], o conteúdo das notificações de alteração disseminadas podem ser classificados em dois grupos:

- Orientados a registro: os itens de dados alterados são propagados para os clientes. As técnicas orientadas à registro tem sido referenciadas como *Propagação de Alteração*.

- Orientados a *id*: apenas os identificadores dos registros alterados são propagados para os clientes. As técnicas orientadas a *id* tem sido referenciadas como *Invalidação de Alteração*.

3.4.2 Modo de *Broadcast*

Os modos de *broadcast* das notificações de alteração disseminados podem ser classificados em:

- Assíncronos: o servidor envia um *broadcast* contendo uma notificação de invalidação para um item de dado assim que este item de dado é alterado. No entanto, podemos adicionar algumas informações extras às notificações enviadas de forma assíncrona. Por exemplo, é possível incluir informações sobre outros itens de dados e *timestamps* da alteração mais recente destes dados. Então, ao invés de ser enviada uma mensagem de invalidação, será enviado um relatório de invalidação. Neste caso, um cliente que tenha ficado desconectado por um período de tempo, pode esperar pelo próximo relatório de invalidação para verificar se os dados armazenados no seu cache foram ou não alterados. No entanto, no modo de *broadcast* assíncrono não existe um conhecimento sobre quando o próximo relatório será enviado. Desta forma, não é possível determinar qual será o tempo de espera para recebimento do próximo relatório. De acordo com HOU et al. [HOU 01], o maior problema na abordagem assíncrona é o imprevisível tempo de espera para as mensagens, que dependem das atividades de alteração dos dados.

- Síncronos: são baseados em *broadcasts* periódicos de notificações de invalidação. Neste caso, os servidores reúnem as alterações por um período de tempo e enviam um *broadcast* contendo estas alterações, em conjunto com informações sobre o tempo em que elas ocorreram. Alguma latência pode ser introduzida entre o tempo real de atualização e o tempo em que os clientes móveis serão notificados. Um cliente precisa verificar a notificação de invalidação para concluir se o seu cache continua válido. Se o seu cache não continua válido, o cliente necessita solicitar que o servidor atualize o seu cache. Desta forma, esta unidade lança uma consulta, solicitando que o servidor atualize o conteúdo do seu cache. O *broadcast* de notificações de invalidação divide o tempo em intervalos [HOU 01]. Uma unidade móvel, após se reconectar, precisa esperar até que o próximo relatório de invalidação chegue antes de responder a uma consulta. Assim, a unidade mantém uma lista dos itens solicitados durante um intervalo e responde as solicitações depois de receber a próxima notificação.

Conforme CHUNG & CHO [CHU 98], o modo síncrono é vantajoso quando um cliente é reconectado após algum período de desconexão. O modo assíncrono não provê qualquer garantia de latência para consultas dos clientes. Se o cliente consulta um item de dado após o período de desconexão, este também precisa esperar a próxima notificação ou submeter uma consulta para o servidor. No modo síncrono, no entanto, existe uma garantia de latência devido a natureza periódica desta abordagem.

3.4.3 Tipos de Disseminação de Dados

- *Pull Based*: Nesta abordagem, os clientes solicitam explicitamente as informações desejadas através do envio de mensagens para o servidor, o qual encaminha as informações solicitadas para os clientes. Desta forma, como mencionado em HARA [HAR 02], o servidor entregará os dados para cada cliente de forma separada, respondendo cada mensagem de solicitação recebida.

- *Push Based*: Nesta abordagem são enviadas informações para os clientes, sem esperar que os dados sejam solicitados. Sistemas *push-based* tem o problema de decidir

sobre a relevância dos dados que serão disseminados para os clientes [BAR 99]. O servidor também precisa decidir se os dados serão enviados periodicamente ou aperidioricamente. Envios periódicos tem a vantagem de permitir que os clientes sejam desconectados por certos períodos de tempo e ainda assim, caso estes recebam um *broadcast* depois de serem reconectados, não percam seus itens de dados. Por outro lado, disseminações aperiódicas possuem a vantagem de disponibilidade da largura de banda [BAR 99]. De acordo com HARA [HAR 02], uma das vantagens dos mecanismos *push-based* é o elevado *throughput* (quantidade de dados transmitidos em uma unidade de tempo) de dados em sistemas distribuídos com um grande número de clientes. Isto ocorre, visto que a ausência de contenção de comunicação entre os dados solicitados pelos clientes permite que estes compartilhem a largura de banda de forma efetiva.

- *Hybrid Mode*: Nesta abordagem são combinadas as estratégias *pull-based* e *push-based*. Conforme mencionado em BARBARÁ [BAR 99], esta combinação pode ser proveitosa tanto para o servidor quanto para os clientes.

3.5 Estratégias para Invalidação de Cache

Em ambientes de computação móvel, várias cópias de um determinado dado podem existir em um servidor de banco de dados e em computadores móveis ao mesmo tempo. Desta maneira, a garantia de que os dados armazenados em cache estão sendo mantidos de forma consistente pode ser fornecida através de estratégias para coerência de cache.

Muitos algoritmos têm sido propostos para arquiteturas cliente-servidor convencionais. Contudo, devido às características únicas de um ambiente móvel, algoritmos convencionais não são diretamente aplicáveis à computação móvel.

Nesta dissertação foram consideradas as técnicas *broadcasting timestamp*, *cache coherency schema with incremental update propagation* e *amnesic terminal*. Antes da escolha destas estratégias, foi realizado um estudo sobre as várias formas de

implementação de um ambiente de teste. Neste estudo, identificou-se que a utilização de técnicas de envio de notificação de invalidação para realizar a coerência de cache, demonstrou ser a opção mais atrativa. Outros pontos tiveram que ser definidos, como a abordagem a ser utilizada para invalidação do cache e o modo de *broadcast* a ser empregado. Nesta etapa, concluímos que a utilização de servidores *stateless* e modo de *broadcast* síncrono demonstraram ser mais apropriados para um ambiente de comunicação móvel.

Desta forma, foi realizada a seleção de estratégias que pudessem ser empregadas em um ambiente que apresentasse como característica comum a utilização de notificação de invalidação, servidores *stateless* e modo de *broadcast* síncrono. Em adição, os testes padrão adotados observam o tamanho das notificações de invalidação, o intervalo de *broadcast* entre o envio de uma notificação e outra e o tamanho do cache dos dispositivos móveis. Considerando estes testes, foi necessário que as estratégias adotadas apresentassem variações nos parâmetros de análise, para que fosse possível confrontar o resultado observado. Através do levantamento de técnicas que apresentam estas características, foram selecionadas, para implementação e análise, as estratégias para coerência de cache descritas a seguir.

3.5.1 *Broadcasting Timestamp* (TS)

Na estratégia *Broadcasting Timestamp* [BAR 95] o servidor envia um *broadcast* contendo uma notificação de invalidação para os clientes, informando os itens que foram alterados nos últimos *w* segundos. Esta notificação contém o *timestamp* da notificação atual (T_i) e uma lista de tuplas (*j, tj*), onde *j* especifica o item de dado que foi alterado e *tj* indica quando a alteração do item foi realizada.

A unidade móvel mantém uma variável T_l indicando quando a última notificação foi recebida. Se a diferença entre T_l e T_i for maior do que *w*, então todo o cache da unidade móvel é apagado. Caso o item contido na notificação de invalidação tenha sido alterado em um tempo maior que o *timestamp* armazenado no cache, a unidade apaga este item do seu cache. Desta forma, se o cliente necessitar utilizar o dado que foi excluído do cache, este deverá buscá-lo novamente no servidor. A unidade móvel também mantém

uma lista Q_i com informações consultadas no intervalo $[T_{i-1}, T_i]$. Por fim, o *timestamp* do cache do cliente móvel (T_l) será alterado de acordo com o *timestamp* atual (T_i) informado. Mais detalhadamente, o dispositivo móvel executa o algoritmo especificado na figura 3.1.

```
if (T_i − T_l > w) { drop the entire cache }
else {
  for every item j in the MU cache {
      if there is a pair [j, t_j] in U_i {
        if t_j^c < t_j {
          throw j out of the cache }
        else { t_j^c = T_i }
      }
  }
}
for every item j ∈ Q_i {
    if j is in the cache {
      use the cache's value to answer the query }
    else { go uplink with the query }}
T_l := T_i
}
```

Figura 3.1: Broadcasting Timestamp [BAR 95]

3.5.2 *Cache Coherence Schema with Incremental Update Propagation* (CCS-IUP)

A estratégia CCS-IUP [CHU 98] tenta evitar a invalidação dos dados do cache, reduzindo desta forma o número de perdas de informações. Além disso, esta estratégia propaga apenas as partes relevantes das modificações que afetam o dado armazenado localmente na unidade móvel. Isto reduz significativamente a quantidade de dados transmitidos. A figura 3.2 apresenta a técnica CCS-IUP.

Nesta estratégia, o banco de dados é representado por um conjunto de relações R_i. Cada relação possui um número de registros. Para cada registro na relação R_i existe um identificador único (*RID*). O servidor deve enviar um *broadcast* periódico no tempo T_k com uma notificação de invalidação contendo informações sobre os registros alterados nos últimos $\triangle$ segundos.

```
if (T_k - T_l > Δ) { drop entire cache }    /* reconnected after long periods of disconnection */
else {
    for every element (R_i, RID, VAL, TS) in Φ {
            if (∃V_c such that the client caches V_c and R_i ∈ PR_c) {
                    if (TS < TS(R_i)) { skip Φ }    /* The log has been reflected already */
                    else {
                      if (VAL is null) D_c = D_c ∪ {(RID, TS)}
                      else I_c = I_c ∪ {(VAL, TS)}
                      TS(R_i) = TS    /* increments the timestamp of base relation */
    }       }       }
    T_l = T_k    /* Reset the timestamp of the client to the current time */
}
for every query Q_i {
    if (the result of Q_i is not cached)    /* cache miss */
            { request Q_i to the server }
    else {
            Let V_i be the cached result of Q_i
            if (∃PR_i such that D_i ≠ ∅ or I_i ≠ ∅) {    /* Base relation has been updated */
                    call view reconstruction
                    TS(V_i) = MAX_{R∈PR_i}(TS(R))    /* view's TS is set to maximum TS of relations */
            }
            Use V_i to answer Q_i
}   }
```

Figura 3.2: CCS-IUP [CHU 98]

A notificação de invalidação é definida formalmente como uma lista $\oint$ cujo elemento é uma 4-tupla (R_i, *RID*, *VAL*, *TS*), sendo que R_i representa o identificador da relação ao qual o registro *RID* pertence; *RID* representa o identificador o registro que foi alterado no intervalo [T_k - $\triangle$, T_k]; *VAL* representa o novo valor do registro e *TS* representa o *timestamp* indicando quando o registro foi alterado pela última vez.

3.5.3 *Amnesic Terminals* (AT)

Na estratégia *Amnesic Terminals* [BAR 95] o servidor fica responsável por informar os identificadores dos itens que foram alterados desde a última notificação de invalidação. O algoritmo para a unidade móvel é apresentado na figura 3.3.

```
if (T_i − T_l > L) { drop the entire cache }
else {
  for every item j in the MU cache {
      if j is in the report {
        throw j out of the cache }
  }
}
for every item j ∈ Q_i {
    if j is in the cache {
      use the cache's value to answer the query }
    else { go uplink with the query }}
T_l := T_i
}
```

Figura 3.3: Amnesic Terminals [BAR 95]

A notificação de invalidação contém uma lista com os identificadores dos dados alterados (*j*). Após receber a notificação, a unidade móvel compara o item no seu cache com os itens informados na notificação. Se um item de dado armazenado no dispositivo móvel for informado na notificação, o dispositivo apaga este item do seu cache. Caso contrário, a unidade móvel considera o item do seu cache válido. Nesta estratégia, a unidade móvel também mantém uma lista Q_i com informações consultadas no intervalo [T_{i-1}, T_i].

3.5.4 Considerações sobre as Estratégias Implementadas

Conforme observado nas seções 3.5.1, 3.5.2 e 3.5.3, em todas as três estratégias implementadas, para se certificar da consistência do cache, o servidor envia uma notificação de invalidação (NI) para os dispositivos móveis a cada L segundos. O conteúdo desta NI varia de acordo com a estratégia que esteja sendo utilizada. Desta forma, o tamanho da notificação, para um mesmo item alterado, será diferente em cada estratégia:

- No algoritmo TS, a NI contém o *timestamp* da notificação atual (T_i) e uma lista de tuplas (*j, tj*), onde *j* especifica o item de dado que foi alterado e *tj* indica quando a alteração do item foi realizada;

- No algoritmo CCS-IUP, esta NI é formada por seu *timestamp* (T_i) e uma lista (R_i, *j*, *VAL*, tj), sendo que R_i representa o identificador da relação ao qual o registro *j* pertence; *j* representa o identificador do registro que foi alterado no intervalo [T_i - w, T_i]; *VAL* representa o novo valor do registro e tj representa o *timestamp* indicando quando o registro foi alterado pela última vez;

- Na estratégia AT, a NI é composta por seu *timestamp* (T_i) e uma lista com os identificadores dos dados alterados (*j*).

3.6 Trabalhos Correlacionados

A seguir são apresentados alguns trabalhos que relacionam-se com o projeto desenvolvido nesta dissertação:

- Em BARBARÁ & IMIELÍNSKI [BAR 95] é proposta uma taxonomia de diferentes estratégias de invalidação de cache e um estudo do impacto do tempo das desconexões dos clientes em sua performance. Neste trabalho são estudadas formas para melhorar a eficiência das técnicas de invalidação descritas. Em adicional, são expostas maneiras de implementar as técnicas estudadas em diferentes ambientes de rede. A análise dos experimentos foi realizada através da utilização de um modelo matemático e não utiliza um modelo real.

- O trabalho de PALAZZO et al. [PAL 99] é endereçado à análise de banco de dados distribuídos replicados minimamente. Neste trabalho são discutidas diferentes estratégias para replicação dos dados do usuário. O trabalho verifica as conseqüências das falhas nos nodos em diferentes estruturas de banco de dados distribuídos. Para realizar a análise do experimento foi desenvolvida uma rede de Petri.

- Em HOLLIDAY et al. [HOL 02] foi desenvolvido um *framework* de banco de dados distribuídos para ambientes móveis. Neste trabalho é considerado um banco de dados distribuído que pode ser feito inteiramente por componentes móveis. O trabalho demonstra como o ambiente deve ser projetado para suportar as freqüentes

desconexões dos bancos de dados locais. Ao contrário do experimento realizado nesta dissertação, que considera o modo de *broadcast* síncrono para disseminação dos dados, o trabalho de HOLLIDAY [HOL 02] utiliza o modo assíncrono.

- Em WANG et al. [WAN 03] é proposta uma estratégia para coerência de cache, utilizando o modo de *broadcast* assíncrono e envio de notificações de invalidação. Em adição, o trabalho desenvolvido apresenta uma proposta híbrida de abordagem de invalidação de cache. Desta forma, são utilizadas características das abordagens *stateless* e *stateful* em conjunto. A análise dos experimentos é realizada através de simulação.

Capítulo 4

Ambiente Proposto para Análise Comparativa entre as Estratégias de Coerência de Cache

Este capítulo apresenta o ambiente experimental desenvolvido. O objetivo deste ambiente é possibilitar a análise comparativa entre as estratégias de coerência de cache selecionadas. Seu desenvolvimento iniciou-se com o estudo das possibilidades de implementação. Desta forma, foi necessário identificar as características principais que este deveria apresentar, para que então, fossem selecionadas as estratégias de coerência de cache que se adequassem a estas características. A descrição das opções de implementação estudadas, assim como as justificativas para a utilização de cada opção adotada, são discutidas na seção 4.1. A seção 4.2 apresenta a arquitetura desenvolvida. Na seção 4.3 é apresentado o protótipo implementado.

4.1 Descrição do Ambiente Experimental

Muitos algoritmos têm sido propostos para arquiteturas cliente-servidor convencionais [WAN 91, CAR 91, BJO 02]. Da mesma forma, existe uma tendência na utilização de arquiteturas cliente-servidor em ambientes móveis [ELM 95, JIN 97,

HOL 02]. Nesta dissertação foi utilizado o modelo de comunicação cliente-servidor. Esta opção foi adotada, considerando que, através da utilização deste modelo de comunicação, é possível fazer uso de grandes servidores fornecendo os serviços que exijam uma quantidade maior de processamento e, desta forma, as plataformas móveis passam a ser utilizadas apenas como dispositivos de armazenamento de informações.

Para desenvolver os clientes e o servidor foram utilizadas funções *socket* [STE 98]. A figura 4.1 apresenta as funções *socket* utilizadas para estabelecer a comunicação entre um cliente TCP e um sevidor. Através desta figura, pode ser observado que em um primeiro momento o servidor é iniciado, então posteriormente um cliente é iniciado e se conecta ao servidor. O cliente envia um pedido para o servidor, que por sua vez processa este pedido e retorna uma resposta para o cliente. Este procedimento continua, até que o cliente feche a conexão, enviando uma notificação de fim-de-arquivo para o servidor.

No ambiente desenvolvido, os dispositivos móveis possuem capacidade de armazenar registros em seu cache local. Desta forma, podemos invalidar o cache destes dispositivos através de servidores *stateful* ou de servidores *stateless*. Os servidores *stateful* requerem que a ESM mantenha-se informada sobre quais clientes móveis estão atualmente sob sua cobertura e qual o conteúdo do cache destes clientes. De forma contrária, os servidores *stateless* não possuem estas informações. Nesta dissertação, optou-se pela utilização de um servidor *stateless*. Esta abordagem foi adotada, considerando que as abordagens de invalidações imediatas, como a *stateful*, não são praticáveis em ambientes onde os clientes possam conectar e desconectar freqüentemente da rede sem fio. Neste caso, o servidor tem dificuldades em manter-se informado sobre todos os clientes que possuem uma cópia de um determinado registro que tenha sido modificado.

A coerência do cache foi realizada através da disseminação de notificações de invalidações. Esta disseminação possui o modo de *broadcast* assíncrono e o modo síncrono. No modo assíncrono, o servidor envia uma NI para os clientes móveis, imediatamente após ocorrerem alterações nos itens de dados. No modo síncrono, as NIs são enviadas periodicamente. Nesta dissertação, foi utilizado o modo de *broadcast* síncrono. Para a realização desta escolha, considerou-se a natureza periódica existente no

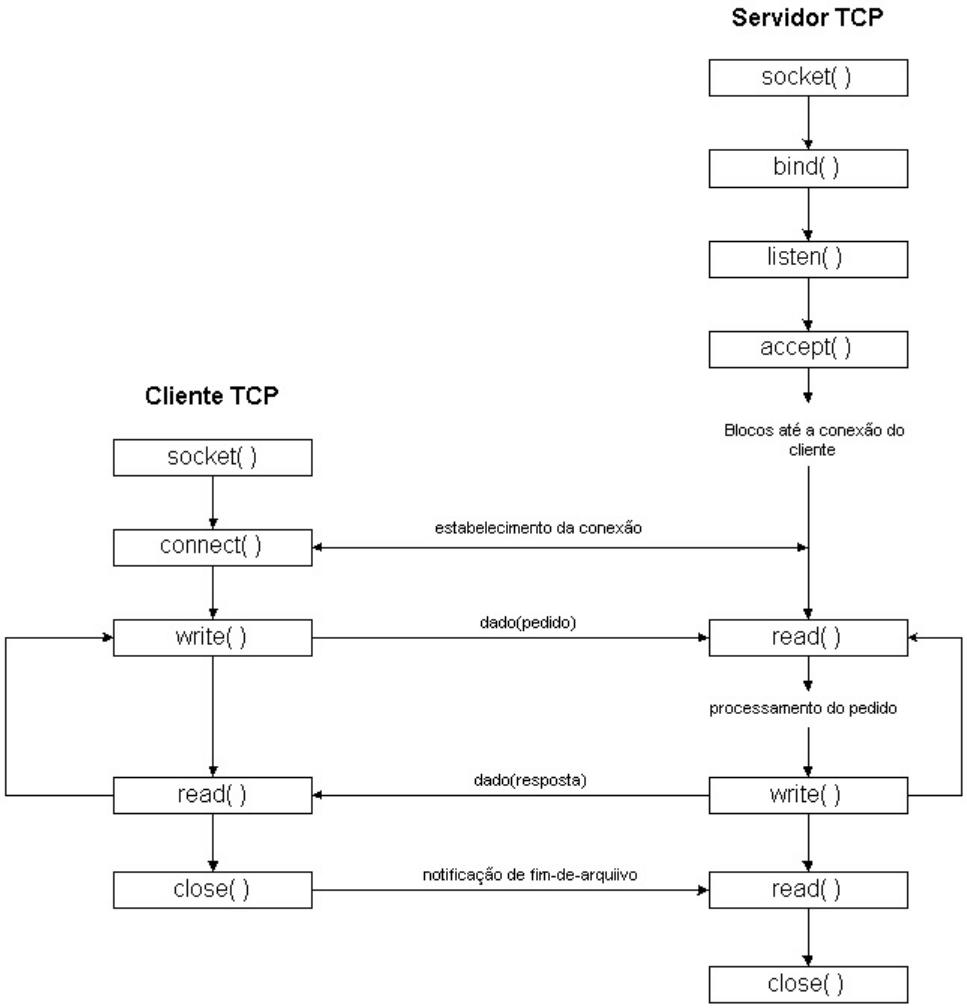

Figura 4.1: Funções *Socket*

modo síncrono. Desta forma, esta abordagem possui uma garantia de latência. Na abordagem assíncrona, as NIs serão apenas enviadas quando os dados forem alterados. Desta maneira, um cliente que tenha permanecido desconectado durante o envio da última NI apenas receberá uma nova notificação quando for efetivada alguma alteração no banco de dados.

O ambiente desenvolvido é composto por dois clientes móveis, uma EF e uma ESM, conforme apresentado na figura 4.2. A EF é o servidor de banco de dados e a ESM é a máquina responsável pela comunicação dos clientes móveis com o servidor de banco de dados. A ESM possui um ponto de acesso IEEE 802.11b (interface sem

fio). Através deste ponto de acesso, a ESM fica responsável pela área geográfica chamada célula. Apenas os dispositivos móveis que estiverem na área de cobertura desta célula poderão se comunicar com o servidor.

Figura 4.2: Ambiente experimental

O canal para comunicação sem fio é logicamente separado em dois sub-canais, um para *uplink* e outro para *downlink*, conforme apresentado na figura 4.3.

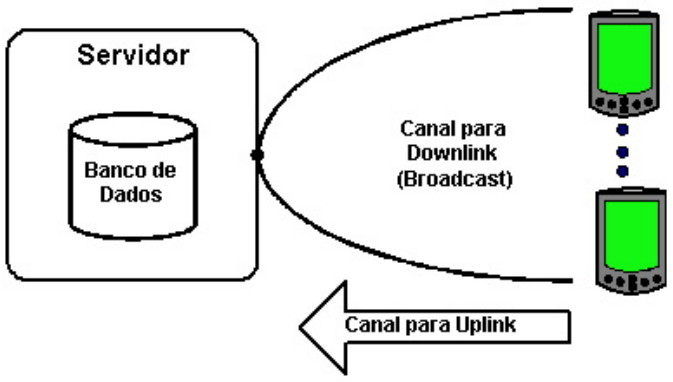

Figura 4.3: Especificação dos canais para comunicação sem fio

Através do canal de *uplink* os clientes submetem suas consultas para o

servidor via ESM. De forma contrária, através do canal de *downlink* o servidor utiliza a ESM para submeter suas respostas e *broadcasts* de NIs para os clientes.

A disseminação dos dados para os clientes pode ser do tipo *pull-based*, *push-based* ou híbrida. Nesta dissertação foi utilizada a abordagem híbrida. Desta forma, o cliente possui tanto características do tipo *pull-based*, podendo enviar solicitações para o servidor; quanto características do tipo *push-based*, permitindo que este receba mensagens do servidor, sem que seja necessário o envio de uma solicitação.

A tabela 4.1 ilustra as características de *hardware* e *software* do servidor e nodos clientes empregados nos testes.

Tabela 4.1: Características do ambiente

Nó	Modelo	Processador	Memória	SO
Clientes Móveis	Palm Tungsten C	400MHz	64MB	Palm OS 5.2.1
Servidor	AMD duron	1,2GHz	256MB	Windows XP

4.2 Arquitetura do Ambiente

A arquitetura do ambiente desenvolvido para realização dos experimentos é apresentada na figura 4.4. Esta arquitetura possui três componentes principais: um servidor, clientes e um sistema gerenciador de banco de dados. Em adição, foram implementadas as estratégias TS, CCS-IUP e AT.

O servidor é composto pelos módulos descritos a seguir.

- Gerenciador de conexão (GC) - Este elemento é responsável pelo gerenciamento das conexões dos clientes com o servidor. Através do GC são recebidas as solicitações, retornadas as respostas a estas solicitações e enviadas as notificações de invalidação para os dispositivos móveis ativos. Ao ser iniciado, o servidor apaga as informações contidas na notificação de invalidação. Por isso, sempre que o servidor for reiniciado, o conteúdo de suas NIs e o conteúdo do cache dos dispositivos móveis serão invalidados. O servidor verifica continuamente se algum cliente está

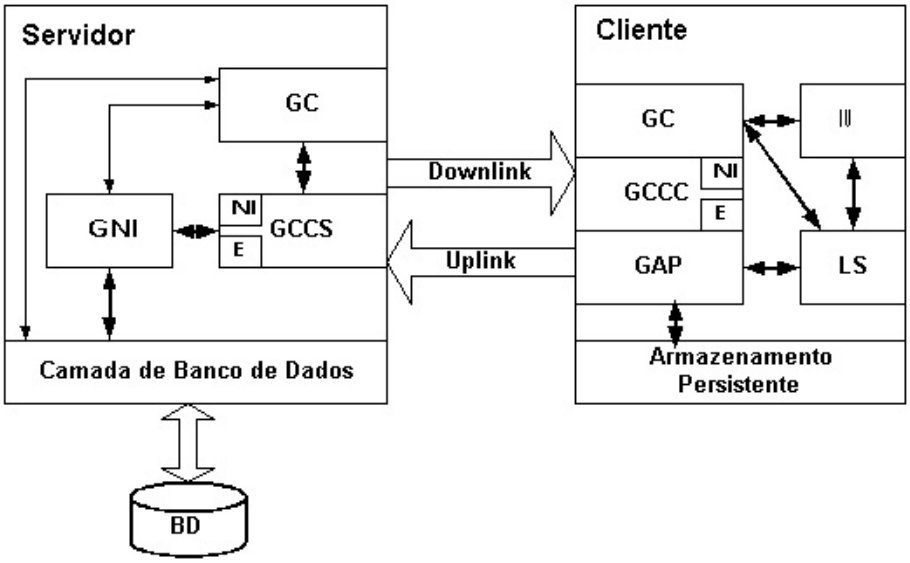

Figura 4.4: Arquitetura do ambiente

tentando estabelecer conexão. Em caso positivo, a conexão é estabelecida através das funções *socket*. Ao estabelecer uma conexão, o cliente pode solicitar uma operação de consulta ou uma alteração no banco de dados. Estas operações são realizadas pelos clientes de forma concorrente. Para que o recurso compartilhado fosse protegido, foram utilizados monitores. Em Java os monitores são executados através de métodos sincronizados. Desta forma, tanto a consulta quanto a alteração dos dados no servidor foram realizadas através destes métodos.

- Gerenciador para coerência de cache no servidor (GCCS) - O GCCS verifica qual estratégia (E) está sendo utilizada. Este componente é responsável por adicionar na notificação de invalidação os parâmetros necessários para a estratégia em vigor. Em adição, o intervalo entre o envio da NI e o tamanho da janela de *broadcast* são especificados em parâmetros controlados pelo GCCS. A rotina para exclusão dos dados da NI a cada janela de *broadcast* e intervalo entre o envio de uma NI e outra é especificado na figura 4.5. A rotina realizada pelo servidor para enviar o *broadcast* das NIs é apresentada na figura 4.6. Nesta rotina, foi utilizado o endereço de *broadcast* da rede onde se encontrava o ambiente de teste. A porta 4445 foi

selecionada para receber os pacotes (contendo a NI) enviados pelo servidor.

```
try {
        if (this.contJanela == 10)//tamanho do w(janela de broadcast)
        {
                this.bd.apagaNotifInvalid();//Limpa os dados da NI
                this.contJanela = 0;
                Date data = new Date();
                this.tsCache  = data.getTime(); //timestamp do servidor
        }
        Thread.sleep(20000);//Tempo entre o envio de um relatório e outro (20s)

     }
```

Figura 4.5: Rotina para enviar e excluir os valores de uma NI

```
try {
        byte[] buf = new byte[notificacao.length()];

        String dString = null;
        buf = notificacao.getBytes();

        InetAddress ip = InetAddress.getByName("150.162.60.255");
        DatagramPacket packet = new DatagramPacket(buf, 0, buf.length, ip, 4445);
        socket.send(packet);

    }
```

Figura 4.6: Rotina para o envio de *broadcasts* aos dispositivos móveis

- Gerenciador de notificações de invalidação (GNI) - Este elemento é responsável pela alteração das NIs quando os dados forem alterados no banco de dados. Se o item alterado já estiver contido na NI, apenas seu *timestamp* será modificado. Caso contrário, as informações referentes a este item serão incluídas na NI. Todas as informações pertinentes ao item alterado são armazenadas em uma tabela existente no servidor de banco de dados (tabela: notificaInval , informações: código do item, quantidade, *timestamp*). No entanto, na NI apenas são informados os parâmetros necessários para a estratégia que esteja sendo utilizada.

Os clientes são compostos pelos seguintes módulos.

- Gerenciador de conexão (GC) - O GC gerencia a conexão do cliente com o servidor. A figura 4.7 apresenta o algoritmo utilizado para estabelecer a conexão do cliente

móvel com o servidor. Após realizada a conexão, o cliente pode submeter consultas ou efetuar alterações ao banco de dados.

```
String hostname = "150.162.60.156";
int port = 3000;

try {
        sc = (SocketConnection)
        Connector.open("socket://" + hostname + ":" + port);

        disEntrada  = sc.openDataInputStream();
        dosSaida    = sc.openDataOutputStream();

        Date data = new Date();
        cm.tsCliente = data.getTime(); //timestamp do cache no dispositivo movel

        dosSaida.writeUTF(consulta);
        dosSaida.flush();

        cm.resultConsulta = disEntrada.readUTF();
}
```

Figura 4.7: Conexão do cliente com o servidor de banco de dados

- Gerenciador para coerência de cache no cliente (GCCC) - O gerenciamento do cache no cliente é realizado pelo GCCC. Este módulo é responsável por receber as NIs e aplicar as modificações necessárias ao conteúdo do seu cache, considerando a estratégia (E) que esteja sendo utilizada. A rotina realizada pelos dispositivos móveis para receber as NIs é apresentada na figura 4.8. Caso o conteúdo do cache necessite ser invalidado, a exclusão dos dados será realizada no próprio GCCC. Em adição, este elemento é responsável por ativar o gerenciador do armazenamento persistente (GAP) quando for necessário invalidar ou alterar algum item.

- Gerenciador do armazenamento persistente (GAP) - Este elemento executa as alterações necessárias no cache do dispositivo móvel. Desta forma, o GAP exclui os itens que foram invalidados (caso a estratégia utilizada seja TS ou AT) e atualiza o valor das informações (caso a estratégia em uso seja a CCS-IUP).

- Lista de solicitações (LS) - Na LS são armazenados todos os registros que estão sendo modificados. Ao fim de uma janela de *broadcast*, os itens que estiverem em cache, e não estiverem na LS dos dispositivos móveis, são invalidados. Desta

```
int port = 4445;
try {
        sc = (DatagramConnection)
        Connector.open("datagram://:" + port);

        while(true)
        {
        byte[] buf = new byte[512];
        is = sc.newDatagram(buf, buf.length);

        sc.receive(is);

        StringBuffer sb = new StringBuffer();
        for (int i = 0; i < buf.length; i++) {
                sb.append((char) buf[i]);
        }

     }
```

Figura 4.8: Rotina para receber a NI

forma, os dados que foram consultados pelo cliente são excluídos do seu cache. No entanto, os itens contidos na LS permanecem em cache, para que possam ser validados em uma próxima comunicação com o servidor.

- Interface com o usuário (IU) - O elemento IU representa uma interface amigável para que os usuários móveis possam acessar o ambiente. Através desta interface o cliente pode realizar consultas e alterações no servidor de banco de dados. Em adição, é possível enviar as estatísticas de utilização das estratégias para o servidor.

O sistema de banco de dados utilizado neste experimento foi o Mckoi SQL [Mck 04]. Este banco de dados foi escolhido por ser de código aberto e fornecer as funcionalidades necessárias para o ambiente desenvolvido (por exemplo, gatilhos e integridade referencial).

Os parâmetros considerados nos experimentos realizados são apresentados na tabela 4.2. O banco de dados possui 50 registros. Esta quantidade foi estabelecida, considerando que o ambiente possui apenas dois clientes. Desta forma, se o número de registros armazenados no banco fosse extenso, o problema do controle da concorrência poderia não ser facilmente identificado.

A janela de *broadcast* possui 10 intervalos e o intervalo entre cada *bro-*

adcast é de 20 segundos (da mesma forma que em [PAL 99, LAU 02, FIF 03]). Em adição, os testes experimentais realizados utilizando TS, CCS-IUP e AT consideram que os usuários móveis possuem em média 5, 15 e 25 porcento dos registros localizados no banco de dados.

Tabela 4.2: Parâmetros

Parâmetro	Valor
Número de registros no banco de dados (D)	50
Número de clientes	2
Intervalo de *broadcast* (L)	20s
Janela de *broadcast* (w)	10
Número de registros no cache (c)	5%, 15% e 25% de D

No ambiente proposto, o servidor se comunica com os dispositivos móveis através do canal de *downlink*. Esta comunicação apenas poderá ser realizada se o dispositivo móvel possuir energia e estiver na área de cobertura da rede sem fio. Os computadores móveis podem ler os dados e solicitar uma alteração para o servidor através do canal de *uplink*. No entanto, esta alteração apenas será realizada pelo servidor.

Considerando uma situação onde o mesmo número de registros tenha sido alterado pelos clientes móveis, utilizando as estratégias TS, CCS-IUP e AT, podemos observar que os algoritmos TS e CCS-IUP possuem o mesmo número de registros na notificação de invalidação. Neste caso, ambas as estratégias armazenam na sua notificação os registros alterados nos últimos w *broadcasts*. O algoritmo AT, no entanto, apenas armazena na notificação os registros que foram alterados nos últimos L segundos. Desta forma, existe uma tendência para que este algoritmo apresente na NI um número menor de registros do que os contidos nas notificações geradas pelos algoritmos TS e CCS-IUP.

4.3 Protótipo Desenvolvido

O ambiente de computação móvel pode disponibilizar aos usuários equipados com dispositivos móveis o acesso a banco de dados sobre redes sem fio. Por exemplo, passageiros podem acessar programações de vôos ou informações sobre o tempo enquanto estão na estrada; vendedores podem acessar informações de catálogo enquanto estão nos clientes; motoristas podem consultar informações sobre o trânsito; e outros inúmeros tipos de aplicações podem ser implementadas.

O problema da coerência de cache será tratado nesta dissertação através do uso de um sistema de controle de estoque. Este sistema possibilita que os usuários móveis compartilhem os dados fornecidos pelo servidor.

A figura 4.9 ilustra a tela inicial do sistema. Através desta tela o usuário pode especificar os parâmetros a serem consultados.

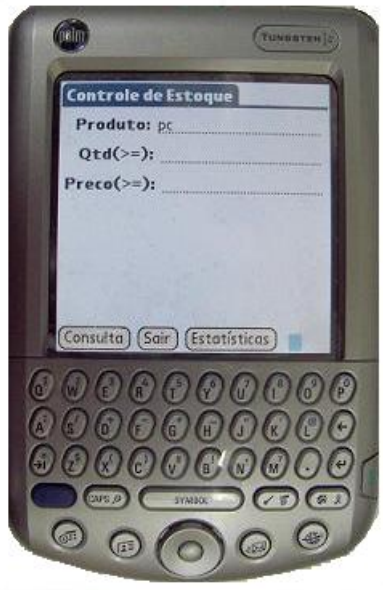

Figura 4.9: Tela para consulta

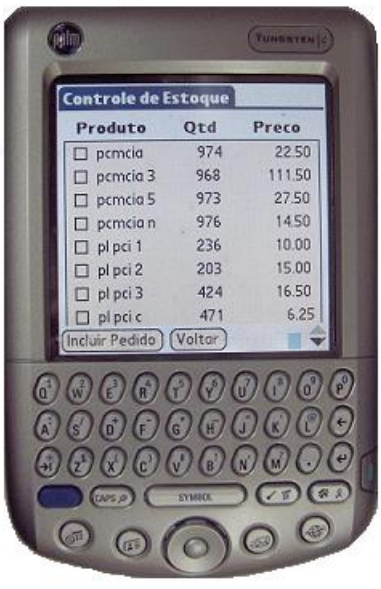

Figura 4.10: Resultado da consulta

Ao selecionar a opção consulta, a unidade móvel tentará estabelecer uma conexão com o servidor de banco de dados. Caso a conexão seja realizada com sucesso, será enviado ao servidor uma solicitação das informações que se encaixem nos

parâmetros selecionados pelo cliente. Caso o servidor encontre os dados solicitados, este encaminhará a resposta ao cliente. Caso contrário, o servidor enviará uma mensagem notificando que os dados solicitados não foram encontrados.

A figura 4.10 mostra a tela contendo os produtos solicitados pelo cliente. Nesta etapa, os dados consultados já estão armazenados no cache do dispositivo móvel. Sendo assim, o cliente já poderá receber NIs para prover a consistência dos seus dados em relação aos armazenados no servidor.

Os produtos que foram solicitados pelo cliente, e armazenados em seu cache, podem ser selecionados e incluídos no pedido, conforme apresentado na figura 4.11.

Na tela ilustrada através da figura 4.12, deve ser informada a quantidade requerida para cada produto incluído no pedido. O pedido pode ser então confirmado, cancelado ou o usuário pode voltar para a tela anterior. Caso o usuário realize a confirmação do pedido são realizadas as seguintes verificações:

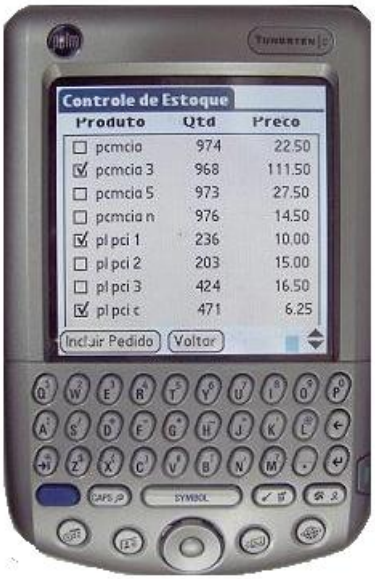

Figura 4.11: Itens selecionados

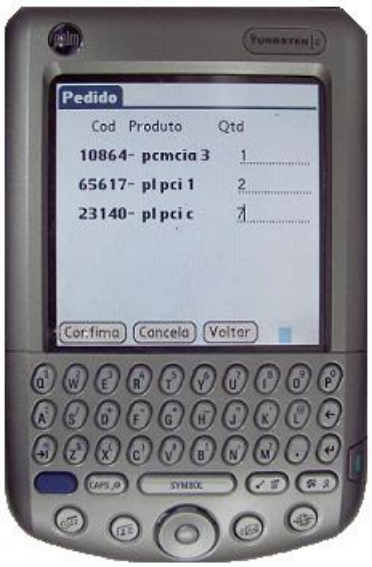

Figura 4.12: Itens incluídos no pedido

- A quantidade solicitada para os itens incluídos no pedido deve ser menor ou igual que a quantidade existente;

- Caso o cliente esteja conectado, os dados incluídos no pedido serão encaminhados para o servidor. Em seguida, o banco de dados e a NI são atualizados. Após realizar as atualizações necessárias, o servidor encaminhará uma mensagem para o cliente, informando o sucesso da operação;

- Caso o cliente esteja desconectado, os dados incluídos no pedido são mantidos na lista de solicitações (LS), esperando que uma nova conexão seja realizada, para que estes sejam encaminhados ao servidor. Quando a conexão ocorrer, o cache do dispositivo móvel será validado e, se as informações em cache continuarem válidas, os dados serão encaminhados para o servidor.

Caso o pedido seja confirmado, e as alterações tenham sido realizadas corretamente no servidor de banco de dados, será exibida a tela de confirmação apresentada na figura 4.13.

Para esta dissertação, foi definido que o banco de dados apenas pode ser alterado pelo servidor de banco de dados. Sendo assim, o servidor fica responsável por manter a integridade do banco. Desta forma, sempre que o cliente realizar a confirmação de um pedido, alterando os dados armazenados no banco de dados, o servidor será responsável pela alteração das NIs. Estas notificações serão encaminhadas para os clientes através de *broadcasts* periódicos. Os clientes devem então receber as NIs e executar as alterações necessárias no cache do dispositivo móvel.

Caso seja necessária alguma alteração no cache, será exibida a mensagem apresentada na tela representada pela figura 4.14 e o cliente móvel realizará os seguintes procedimentos:

- Caso a estratégia utilizada seja TS, os registros contidos na NI, que estiverem em cache, e possuam um *timestamp* maior do que o armazenado em cache, são excluídos do cache do dispositivo. Neste caso os dados informados na tela são reiniciados apenas com os valores que continuam consistentes;

- Caso a estratégia utilizada seja AT, os registros contidos na NI que estiverem em

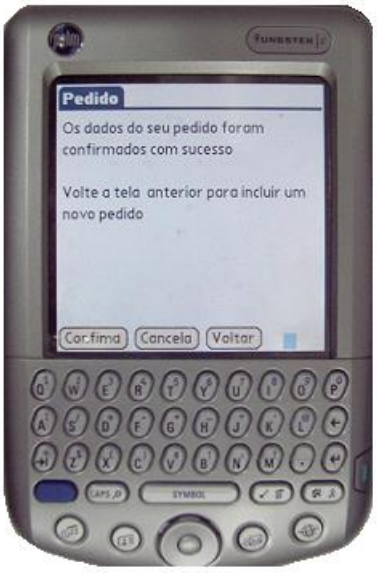

Figura 4.13: Confirmação do pedido

Figura 4.14: Atualização dos dados

cache são excluídos. A tela então será reiniciada apenas as informações que continuam válidas;

- Caso a estratégia em uso seja a CCS-IUP, os registros em cache com valores inconsistentes terão seus valores atualizados. Desta forma, o registro apenas é excluído do cache do dispositivo se ele tiver sido excluído do banco de dados. Em seguida a tela é reiniciada com os registros atuais.

A cada janela de *w* segundos (neste caso, *w* possui 10 intervalos de *broadcast*, sendo que cada intervalo é de 20 segundos) todos os dados no cache dos dispositivos móveis, que não estão na LS, são invalidados. Os registros contidos na LS continuam em cache, até que possam ser validados. Desta forma, a cada janela de *broadcast* é exibida a mensagem apresentada na figura 4.15 e são realizados os seguintes procedimentos:

- O servidor apaga todo o conteúdo da NI;

- O *timestamp* da NI é configurado com o valor atual;

Figura 4.15: Invalidação dos dados

- O cliente móvel apaga todo o conteúdo do seu cache. No entanto, as informações contidas na LS continuam em vigor.

Capítulo 5

Resultados Experimentais

Este capítulo apresenta os resultados obtidos através da análise comparativa entre as estratégias de coerência de cache implementadas. A finalidade deste estudo é demonstrar o comportamento de cada uma das estratégias, considerando os testes padrão empregados. Os testes experimentais observaram o comportamento das estratégias em relação ao impacto do tamanho da notificação de invalidação, intervalo de *broadcast* e quantidade de dados no cache dos dispositivos móveis. Os resultados demonstraram diferentes desempenhos para os parâmetros considerados [BER 04b, BER 04a].

5.1 Avaliação do Desempenho

Como explicado anteriormente, os testes experimentais realizados consideram um ambiente real de configuração sem fio. Esta configuração é formada por um nodo servidor e dois clientes móveis. Neste trabalho, foram estudadas as diferenças entre as estratégias TS, AT e CCS-IUP. Desta forma, foram usados os seguintes parâmetros:

- Tamanho das notificações de invalidação (NIs);
- Intervalo de *broadcast*;
- Tamanho do cache dos dispositivos móveis.

Para realização dos experimentos foi armazenado no cache nos dispositivos móveis em média 5, 15 e 25 % da quantidade de registros existente no banco de dados (localizado no servidor). O armazenamento foi realizado através de consultas efetuadas pelos usuários móveis. Conforme verificado na tabela 5.1, para análise do tempo utilizado para alteração do cache foram realizados experimentos com notificações de invalidação contendo o tamanho de 110, 170, 230 e 290 bytes. Foram realizadas 3 observações em cada combinação de estratégia, tamanho do cache e tamanho da notificação de invalidação.

Conforme explicado anteriormente, devido aos diferentes parâmetros considerados nas NIs para cada estratégia, existe uma variação do tamanho destas notificações. Desta forma, para que o tamanho das NIs, enviadas pelo servidor aos clientes móveis, atingissem 110, 170, 230 e 290 bytes, as estratégias TS e CCS-IUP deveriam conter respectivamente em sua NI informações referentes à 1, 2, 3 e 4 registros. A estratégia AT, por considerar um número reduzido de parâmetros em suas NIs, para os respectivos 110, 170, 230 e 290 bytes deveria conter informações referentes à 4, 8, 12 e 16 registros.

Após a confirmação de cada pedido, foi realizada a atualização dos registros armazenados no servidor de banco de dados e na NI. As confirmações dos pedidos ocorreram da seguinte forma:

- A cada 2 ou 3 intervalos de *broadcast*, quando utilizadas as estratégias TS e CCS-IUP, foi realizada a confirmação de pedido contendo 1 registro e, quando utilizada a estratégia AT, a confirmação do pedido possuía 4 registros. Esta diferença no número de registros alterados teve que existir para que o tamanho das NIs fossem equivalentes em todas as estratégias;

- Os testes foram realizados separadamente para cada estratégia. Após a realização dos experimentos, o banco de dados foi realimentado com os dados originais (sem alteração), para que todas as estratégias possuíssem as mesmas condições de funcionamento.

A tabela 5.2 apresenta as medidas descritivas do tamanho das NIs para os experimentos realizados.

Tabela 5.1: Medidas descritivas do tempo para alteração do cache em cada estratégia de coerência de cache

Estratégia	Tamanho do Cache (% de D)	Tamanho da NI (bytes)	Tempo médio de alteração (ms)	Desvio padrão
TS	5	110	46,67	4,71
		170	25,00	8,66
		230	30,00	0,00
		290	30,00	0,00
	15	110	80,00	10,00
		170	75,00	5,00
		230	70,00	0,00
		290	55,00	15,00
	25	110	166,67	4,71
		170	150,00	20,00
		230	160,00	10,00
		290	205,00	55,00
CCS-IUP	5	110	35,00	5,00
		170	30,00	0,00
		230	35,00	5,00
		290	40,00	0,00
	15	110	92,50	8,29
		170	90,00	10,00
		230	80,00	0,00
		290	110,00	0,00
	25	110	170,00	5,00
		170	190,00	10,00
		230	190,00	0,00
		290	190,00	0,00
AT	5	110	22,50	14,79
		170	45,00	15,00
		230	30,00	0,00
		290	40,00	0,00
	15	110	60,00	8,17
		170	50,00	20,00
		230	60,00	0,00
		290	80,00	0,00
	25	110	137,50	18,43
		170	126,67	5,00
		230	160,00	0,00
		290	130,00	20,00

Tabela 5.2: Medidas descritivas do tamanho da NI em cada estratégia de coerência de cache

Estratégia	Tamanho do Cache (% de D)	Intervalo de *broadcast* (ms)	Tamanho Médio da NI (bytes)	Desvio padrão
TS	5	20000	50,00	0,00
		80000	105,33	1,64
		140000	200,00	32,00
		200000	292,00	0,00
	15	20000	64,00	24,25
		80000	106,00	0,00
		140000	170,00	0,00
		200000	294,00	0,00
	25	20000	50,00	0,00
		80000	106,00	0,00
		140000	170,00	0,00
		200000	294,00	0,00
CCS-IUP	5	20000	54,00	0,00
		80000	118,00	0,00
		140000	242,00	33,94
		200000	342,00	0,00
	15	20000	54,00	0,00
		80000	118,00	0,00
		140000	146,67	64,11
		200000	342,00	0,00
	25	20000	54,00	0,00
		80000	120,00	0,00
		140000	198,00	0,00
		200000	348,00	0,00
AT	5	20000	48,00	0,00
		80000	56,67	19,38
		140000	84,67	24,51
		200000	50,00	0,00
	15	20000	48,00	0,00
		80000	48,00	0,00
		140000	76,00	26,00
		200000	102,00	0,00
	25	20000	48,00	0,00
		80000	48,00	0,00
		140000	76,00	26,00
		200000	151,00	7,00

O Intervalo de *broadcast* considerado foi de 20000, 80000, 140000 e 200000 milesegundos. Para cada combinação de cada estratégia, tamanho de cache e intervalo de *broadcast*, foram realizadas 3 observações.

Durante o desenvolvimento do trabalho, ocorreram problemas na sincronização do relógio do servidor com o relógio dos cliente móveis. As ferramentas para sincronização utilizadas apresentaram retardos na hora dos clientes em relação à hora do servidor. Desta maneira, alguns testes não puderam ser realizados, como por exemplo, o tempo para atualizar o cache dos dispositivos móveis, considera apenas o tempo que o cliente gasta para atualizar os seus dados. Desta forma, o tempo de envio dos dados do servidor até o cliente foi desconsiderado.

Os resultados apresentados na figura 5.1 indicam que as estratégias TS e CCS-IUP levam mais tempo para alterarem o conteúdo do seu cache do que a estratégia AT. Este comportamento acontece porque estas estratégias realizam um número de comparações maior para consistir o cache.

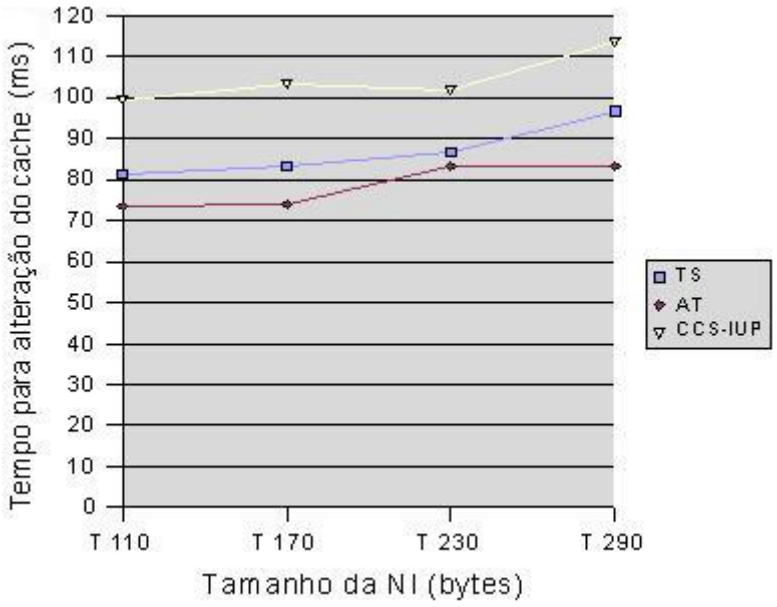

Figura 5.1: Tempo para alteração do cache vs. tamanho da notificação de invalidação

Ao receber a NI, o algoritmo TS verifica se o item *j* contido na NI pertence ao seu cache. Caso este item esteja em cache, o algoritmo verifica se *timestamp* (*tj*)

da última alteração de *j* é maior que o *timestamp* deste item no cache. De forma similar ao TS, o algoritmo CCS-IUP verifica se o item *j* pertence ao seu cache e qual o *timestamp* deste item. Caso o *timestamp* do item *j* (contido na NI) seja maior que o *timestamp* do item correspondente em cache, esta estratégia substitui o valor do item em cache pelo valor do parâmetro *VAL*. Desta forma, em adição às comparações, a estratégia CCS-IUP possui um mecanismo para substituição do cache. Este ponto explica o fato da estratégia TS executar uma alteração em menor tempo do que CCS-IUP. O algoritmo AT recebe a NI e apenas verifica se o item *j* pertence ao seu cache. Caso o item esteja localizado no cache, este elemento será eliminado do dispositivo móvel.

A figura 5.2 apresenta a média dos tamanhos das NIs em relação ao número de *broadcasts* enviados. O gráfico mostra que, nos algoritmos TS e CCS-IUP, o tamanho da NI tende a crescer com o tempo. Nestas estratégias, a NI contém os registros alterados nos últimos w *broadcasts*.

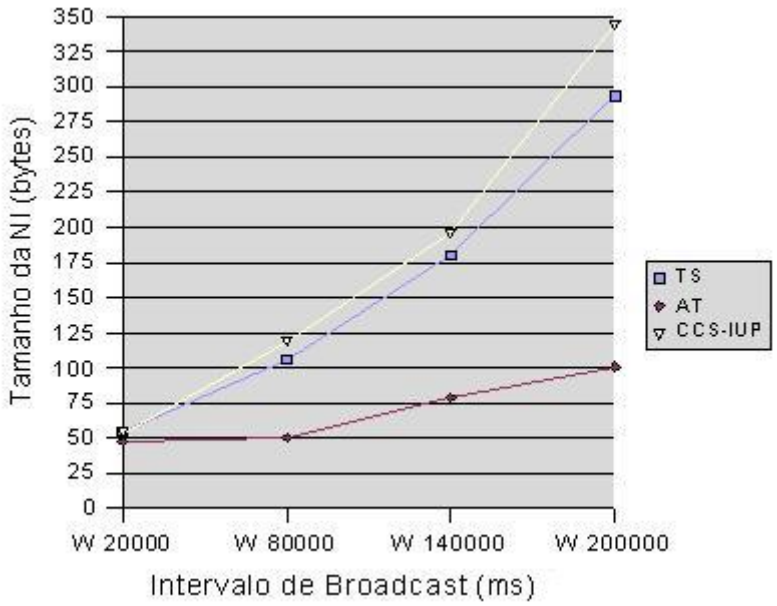

Figura 5.2: Tamanho da notificação de invalidação vs. intervalo de broadcast

Na estratégia CCS-IUP, a NI contém o seu *timestamp*, o identificador do registro alterado, o identificador da relação ao qual o registro alterado pertence, o novo valor do registro e o *timestamp* indicando quando o registro foi alterado pela última vez.

Desta forma, a quantidade de informações em cada NI de CCS-IUP é maior do que nas estratégias TS e AT. A NI de TS envia seu *timestamp*, o identificador do item alterado e o *timestamp* informando a última alteração deste item. A estratégia AT armazena na NI apenas os identificadores dos registros que foram alterados nos últimos L segundos (neste experimento, L= 20 segundos). Desta forma, a estratégia AT possui as menores NIs, seguida pelas estratégias TS e CCS-IUP.

O tamanho da notificação utilizando diferentes tamanhos de cache é apresentado na figura 5.3. O aumento do tamanho do cache resultou em um incremento no tamanho das NIs apenas para AT.

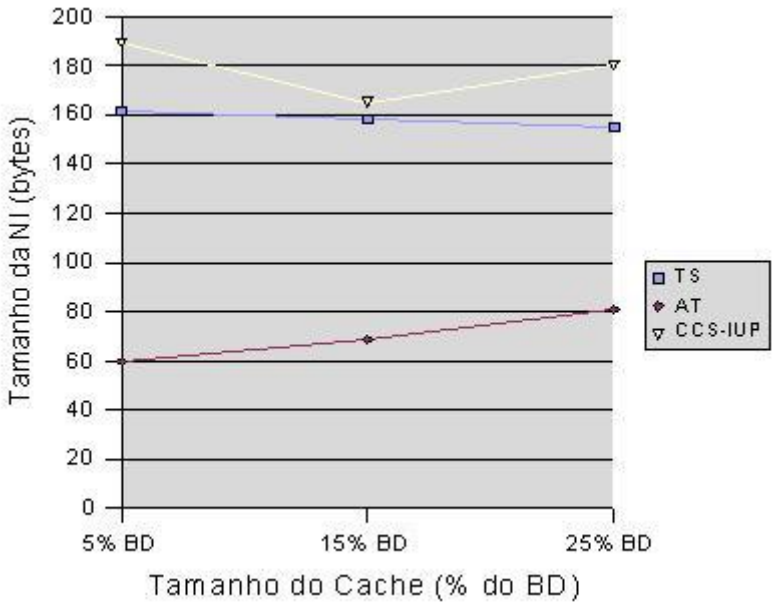

Figura 5.3: Tamanho da notificação de invalidação vs. tamanho do cache

Na estratégia TS obteve-se um pequeno decremento do tamanho da NI em relação ao aumento do tamanho do cache. Na estratégia CCS-IUP o comportamento não obteve um padrão. A possibilidade de uma quantidade maior de dados serem alterados cresce com o aumento do número de registros armazenados no cache. No entanto, como a quantidade de alterações foi realizada de forma uniforme, este comportamento não foi obtido. Quanto ao tamanho das notificações, o gráfico apresenta que a estratégia AT possui NIs menores que as estratégias TS e CCS-IUP.

Na figura 5.4, como esperado, é possível verificar que o incremento do tamanho do cache está diretamente relacionado com o aumento do tempo para a atualização dos dados armazenados nos clientes. Todas as estratégias analisadas apresentaram este comportamento. No entanto, é importante salientar que a estratégia AT necessita alterar o quádruplo do número de registros alterados pelas estratégias TS e CCS-IUP, para que possa obter uma NI de tamanho equivalente ao gerado por estas estratégias. Ainda assim, AT apresentou o menor tempo para alteração do cache nos dispositivos, seguida pela estratégia TS. Neste caso, TS e AT realizam uma quantidade menor de comparações para verificar a consistência do cache, do que a estratégia CCS-IUP. Além disso, as estratégias AT e TS não realizam substituições do cache.

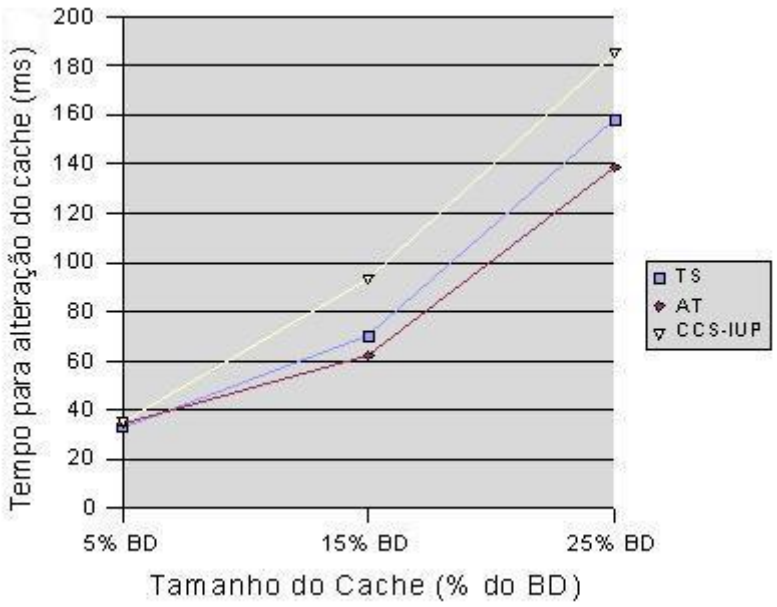

Figura 5.4: Tempo para alteração do cache vs. tamanho do cache

Capítulo 6

Conclusões e Trabalhos Futuros

Diferente das aplicações tradicionais cliente-servidor, a comunicação entre clientes móveis e servidores fixos é via canal de comunicação sem fio. Este meio de transmissão, na maior parte dos casos, possui menor largura de banda, menor confiabilidade e está sujeito a freqüentes desconexões. Sendo assim, armazenar os dados localmente nos clientes é uma alternativa interessante na busca de uma solução para estes problemas.

Nesta dissertação foi apresentada a análise de desempenho de três estratégias para coerência de cache utilizando redes sem fio. O trabalho de pesquisa teve seu início com a definição de algumas estratégias a serem empregadas em um ambiente real utilizando comunicação sem fio. O próximo passo caracterizou-se pela implementação destas estratégias para a realização dos testes experimentais. Em adição, foram desenvolvidos alguns componentes para controlar, estabelecer a comunicação e gerenciamento entre o servidor e os nodos clientes.

Os resultados experimentais demonstraram o comportamento das estratégias considerando a variação do tamanho das notificações de invalidação (NIs), intervalo de *broadcast* e tamanho do cache dos clientes móveis. Os resultados demonstraram diferentes desempenhos para os parâmetros considerados.

O algoritmo AT obteve resultados interessantes em clientes que estão a maior parte do tempo conectados. Isto acontece porque o parâmetro L, utilizado como

comparação antes de invalidar todo o cache do dispositivo, geralmente é muito menor que parâmetro $w.L$ (janela de *broadcast* vezes o intervalo de *broadcast*) utilizado nas estratégias TS e CCS-IUP. Desta forma, em ambientes onde o período de desconexão não ultrapassa o valor L, o algoritmo AT demonstra-se bastante efetivo. Isto porque a estratégia AT transmite uma quantidade menor de informações nas NIs. Esta redução da quantidade de dados transmitidos resulta em uma melhor utilização da largura de banda e um menor tempo para atualizar o cache do dispositivo móvel. Em contrapartida, as estratégias TS e CCS-IUP demonstram ser mais atrativas para clientes que eventualmente se desconectem da rede sem fio por períodos curtos de tempo ($<w.L$), porém superiores ao parâmetro L.

Nas estratégias TS e AT, os registros informados na NI que estão em cache e não atendem as suas condições são excluídos. Sendo assim, se um determinado dispositivo móvel possui algum registro, que foi excluído do cache, armazenado na lista Q_i, este deverá, através do canal de *uplink*, consultar o servidor para verificar os valores destes registros. Ao contrário, a estratégia CCS-IUP, mantém na NI o novo valor do item alterado. Deste modo, esta estratégia não elimina os registros de seu cache, ela apenas substitui o valor antigo pelo atual. Com isso, se um determinado dispositivo móvel necessita fazer muitas alterações no banco de dados, é mais vantajoso que este já tenha os novos valores dos dados alterados em cache, a ter que solicitar que os novos valores sejam reenviados pelo servidor. Neste caso, a estratégia CCS-IUP demonstrou ser a mais indicada.

6.1 Trabalhos Futuros

- Pretende-se implementar uma função que consiga definir, através do ambiente desenvolvido, qual é a estratégia mais indicada para as particularidades de um determinado dispositivo móvel. Esta função utilizará as estatísticas do cliente para obter informação sobre o seu comportamento. Com base em informações como período de desconexão e quantidade de *uplinks* necessários para atualizar o cache, será determinado, de forma dinâmica, qual o algoritmo mais apropriado a ser utilizado

pelo cliente móvel;

- Adicionar ao sistema, a implementação da estratégia *Grouping with COld update-set REtention* (GCORE) [WU 96]. Esta estratégia visa reduzir o número de invalidações desnecessárias. Nela, o servidor particiona o banco de dados em um número de grupos. Desta forma, o servidor consegue dinamicamente identificar os itens de dados *hot* e *cold* em cada grupo. Os itens de dados *hot* são aqueles que foram incluídos na mais recente notificação de invalidação enviada para as unidades móveis. Ao invés de examinar todo o grupo, o GCORE exclui os itens de dados *hot* do grupo quando checa a sua validade. Com as alterações *hot* excluídas do grupo, o servidor pode concluir que os objetos que não foram alterados no grupo, podem ser retidos no cache;

- Dividir as estratégias estudadas em dois conjuntos, os de *propagação de alteração* e os de *invalidação de alteração* conforme taxonomia proposta por MURTHY [MUR 98]. Nesta divisão, TS e CCS-IUP são classificadas como estratégias de propagação de alteração, e AT e GCORE são classificadas como estratégias de invalidação de alteração. Desta forma, para realizar a análise de desempenho destes conjuntos, o banco de dados deverá ser dividido em grupos e o servidor seria responsável por especificar os itens de dados *hot* e *cold* em cada grupo. Assim, será possível determinar qual conjunto é mais eficiente em um banco de dados com um número maior ou menor de alterações;

- Analisar as estratégias já implementadas quanto à quantidade de energia consumida. Desta forma, será possível verificar qual é a estratégia mais econômica, em termos de consumo de bateria. Estes valores poderão contribuir para a escolha da estratégia a ser utilizada, de acordo com a carga disponível no dispositivo móvel.

Referências Bibliográficas

[ADV 96] ADVE, S. V.; GHARACHORLOO, K. Shared memory consistency models: a tutorial. [S.l.], v.29, n.12, p.66–76, 1996.

[AGR 99] AGRAWAL, P.; FAMOLARI, D. Mobile computing in next generation wireless networks. In: PROCEEDINGS OF THE 3RD INTERNATIONAL WORKSHOP ON DISCRETE ALGORITHMS AND METHODS FOR MOBILE COMPUTING AND COMMUNICATIONS, 1999. ACM Press. p.32–39.

[BAR 95] BARBARá;, D.; IMIELíNSKI, T. Sleepers and workaholics: caching strategies in mobile environments (extended version). **The VLDB Journal**, [S.l.], v.4, n.4, p.567–602, 1995.

[BAR 99] BARBARá, D. Mobile computing and databases-a survey. **IEEE Transactions on Knowledge and Data Engineering**, [S.l.], v.11, n.1, p.108–117, 1999.

[BER 04a] BERKENBROCK, C. D. M.; DANTAS, M. A. R. Análise comparativa entre estratégias de coerência de cache em um ambiente de computação móvel. **3rd *International Information and Telecommunication Technologies Symposium***, [S.l.], 2004.

[BER 04b] BERKENBROCK, C. D. M.; DANTAS, M. A. R. Estratégias para coerência de cache em ambientes de computação móvel. **III Simpósio de Informática da Região Centro do RS**, [S.l.], 2004.

[BJO 02] BJORNSSON, M. E.; SHRIRA, L. Buddycache: high-performance object storage for collaborative strong-consistency applications in a wan. In: PROCEEDINGS OF THE 17TH ACM SIGPLAN CONFERENCE ON OBJECT-ORIENTED PROGRAMMING, SYSTEMS, LANGUAGES, AND APPLICATIONS, 2002. ACM Press. p.26–39.

[CAI 97] CAI, J.; TAN, K.-L.; OOI;, B. C. On incremental cache coherency schemes in mobile computing environments. In: PROCEEDINGS. 13TH INTERNATIONAL CONFERENCE ON DATA ENGINEERING, 1997. [s.n.].

[CAI 99] CAI, J.; TAN, K.-L. Energy-efficient selective cache invalidation. **Wirel. Netw.**, [S.l.], v.5, n.6, p.489–502, 1999.

[CAO 02] CAO, G. On improving the performance of cache invalidation in mobile environments. **Mob. Netw. Appl.**, [S.l.], v.7, n.4, p.291–303, 2002.

[CAR 91] CAREY, M. J. et al. Data caching tradeoffs in client-server dbms architectures. In: PROCEEDINGS OF THE 1991 ACM SIGMOD INTERNATIONAL CONFERENCE ON MANAGEMENT OF DATA, 1991. ACM Press. p.357–366.

[CHA 03] CHAN, D.; RODDICK, J. F. Context-sensitive mobile database summarisation. In: PROCEEDINGS OF THE TWENTY-SIXTH AUSTRALASIAN COMPUTER SCIENCE CONFERENCE ON CONFERENCE IN RESEARCH AND PRACTICE IN INFORMATION TECHNOLOGY, 2003. Australian Computer Society, Inc. p.139–149.

[CHE 95] CHESS, D. et al. Itinerant agents for mobile computing. [S.l.], v.2, n.5, p.34–49, 1995.

[CHI 02] CHIASSERINI, C. F.; NUGGEHALLI, P.; SRINIVASAN, V. Energy-efficient communication protocols. In: PROCEEDINGS OF THE 39TH CONFERENCE ON DESIGN AUTOMATION, 2002. ACM Press. p.824–830.

[CHU 98] CHUNG, H.; CHO, H. Data caching with incremental update propagation in mobile computing environments. In: AUSTRALIAN COMPUTER JOURNAL, 1998. [s.n.].

[COR 03] CORTES, S.; LIFSCHITZ, S. **Banco de Dados para um Ambiente de Computação Móvel**. http://ftp.inf.pucpcaldas.br/CDs/SBC2003/pdf/arq0246.pdf, Acessado em 21/01/2005.

[COU 01] COULOURIS, G.; DOLLIMORE, J.; KINDBERG, T. **Distributed systems: concepts and design**. Addison-Wesley, 2001.

[DAN 02] DANTAS, M. **Tecnologias de Redes de Comunicação e Computadores**. Axcel Books do Brasil Editora, 2002.

[dC 03] DA CUNHA, D. P. **Um Estudo das Estratégias de Replicação e Reconciliação de Banco de Dados Móveis em um Ambiente Wireless**. Universidade Federal de Santa Catarina, 2003. Dissertação de Mestrado.

[DIR 00] DIRCKZE, R. A.; GRUENWALD, L. A pre-serialization transaction management technique for mobile multidatabases. **Mob. Netw. Appl.**, [S.l.], v.5, n.4, p.311–321, 2000.

[DUN 95] DUNHAM, M. H.; HELAL, A. Mobile computing and databases: anything new? **SIGMOD Rec.**, [S.l.], v.24, n.4, p.5–9, 1995.

[ELM 95] ELMAGARMID, A.; JING, J.; FURUKAWA, T. Wireless client/server computing for personal information services and applications. **SIGMOD Rec.**, [S.l.], v.24, n.4, p.16–21, 1995.

[FIF 03] FIFE, L. D.; GRUENWALD, L. Research issues for data communication in mobile ad-hoc network database systems. **SIGMOD Rec.**, [S.l.], v.32, n.2, p.42–47, 2003.

[FLI 99] FLINN, J.; SATYANARAYANAN, M. Powerscope: a tool for profiling the energy usage of mobile applications. In: PROCEEDINGS OF THE SECOND IEEE WORKSHOP ON MOBILE COMPUTING SYSTEMS AND APPLICATIONS, 1999. IEEE. p.2–10.

[GUP 01] GUPTA, S.; SRIMANI, P. A strategy to manage cache consistency in a disconnected distributed environment. In: IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS, 2001. IEEE. v.12, p.686–700.

[HAE 83] HAERDER, T.; REUTER, A. Principles of transaction-oriented database recovery. **ACM Comput. Surv.**, [S.l.], v.15, n.4, p.287–317, 1983.

[HAR 02] HARA, T. Cooperative caching by mobile clients in push-based information systems. In: PROCEEDINGS OF THE ELEVENTH INTERNATIONAL CONFERENCE ON INFORMATION AND KNOWLEDGE MANAGEMENT, 2002. ACM Press. p.186–193.

[HOL 02] HOLLIDAY, J.; AGRAWAL, D.; El Abbadi, A. Disconnection modes for mobile databases. **Wirel. Netw.**, [S.l.], v.8, n.4, p.391–402, 2002.

[HOU 01] HOU, W.-C. et al. An optimal construction of invalidation reports for mobile databases. In: PROCEEDINGS OF THE TENTH INTERNATIONAL CONFERENCE ON INFORMATION AND KNOWLEDGE MANAGEMENT, 2001. ACM Press. p.458–465.

[HUA 94] HUANG, Y.; SISTLA, P.; WOLFSON, O. Data replication for mobile computers. In: PROCEEDINGS OF THE 1994 ACM SIGMOD INTERNATIONAL CONFERENCE ON MANAGEMENT OF DATA, 1994. ACM Press. p.13–24.

[IMI 94] IMIELINSKI, T.; VISWANATHAN, S.; BADRINATH, B. R. Energy efficient indexing on air. In: PROCEEDINGS OF THE 1994 ACM SIGMOD INTERNATIONAL CONFERENCE ON MANAGEMENT OF DATA, 1994. ACM Press. p.25–36.

[JIN 97] JING, J. et al. Bit-sequences: an adaptive cache invalidation method in mobile client/server environments. **Mob. Netw. Appl.**, [S.l.], v.2, n.2, p.115–127, 1997.

[KOR 96] KORDALE, R. M.; AHAMAD, M. A scalable technique for implementing multiple consistency levels for distributed objects. In: PROCEEDINGS OF THE 16TH INTERNATIONAL CONFERENCE ON DISTRIBUTED COMPUTING SYSTEMS, 1996. IEEE. p.369–376.

[LAI 03] LAI, K. Y.; TARI, Z.; BERTOK, P. Cost efficient broadcast based cache invalidation for mobile environments. In: PROCEEDINGS OF THE 2003 ACM SYMPOSIUM ON APPLIED COMPUTING, 2003. ACM Press. p.871–877.

[LAM 01] LAM, K.-Y.; LI, G. H.; KUO, T.-W. A multi-version data model for executing real-time transactions in a mobile environment. In: PROCEEDINGS OF THE 2ND ACM INTERNATIONAL WORKSHOP ON DATA ENGINEERING FOR WIRELESS AND MOBILE ACCESS, 2001. ACM Press. p.90–97.

[LAU 02] LAUZAC, S. W.; CHRYSANTHIS, P. K. Personalizing information gathering for mobile database clients. In: PROCEEDINGS OF THE 2002 ACM SYMPOSIUM ON APPLIED COMPUTING, 2002. ACM Press. p.49–56.

[LEO 97] LEONG, H. V.; SI, A. On adaptive caching in mobile databases. In: PROCEEDINGS OF THE 1997 ACM SYMPOSIUM ON APPLIED COMPUTING, 1997. ACM Press. p.302–309.

[LIU 96] LIU, G. Y.; MARLEVI, A.; Maguire, Jr., G. Q. A mobile virtual-distributed system architecture for supporting wireless mobile computing and communications. **Wirel. Netw.**, [S.l.], v.2, n.1, p.77–86, 1996.

[LOH 00] LOH, Y.-H. et al. A hybrid method for concurrent updates on disconnected databases in mobile computing environments. In: PROCEEDINGS OF THE 2000 ACM SYMPOSIUM ON APPLIED COMPUTING, 2000. ACM Press. p.563–565.

[MAT 98] MATEUS, G. R.; LOUREIRO, A. A. F. **Introdução á Computação Móvel**. 11a Escola de Computação COPPE/Sistemas. http://www.ime.usp.br/ brito/mac5743/, Acessado em 18/05/2004.

[Mck 04] Mckoi SQL Database. **Mckoi SQL Database**. http://mckoi.com/database/, Acessado em 07/09/2004.

[Mun 04] Mundo Java. **J2ME - Programando para dispositivos Móveis**. Mundo Java, 2004.

[MUR 98] MURTHY, V. K. Mobile computing deploying agents. In: TWELFTH INTERNATIONAL CONFERENCE ON INFORMATION NETWORKING, 1998. IEEE. p.127–130.

[PAL 99] PALAZZO, S.; PULIAFITO, A.; SCARPA, M. Design and evaluation of a replicated database for mobile systems. **Wirel. Netw.**, [S.l.], v.6, n.2, p.131–144, 1999.

[PER 01] PERRY, M. et al. Dealing with mobility: understanding access anytime, anywhere. **ACM Trans. Comput.-Hum. Interact.**, [S.l.], v.8, n.4, p.323–347, 2001.

[PIT 94] PITOURA, E.; BHARGAVA, B. Building information systems for mobile environments. In: THIRD INTERNATIONAL CONFERENCE ON INFORMATION AND KNOWLEDGE MANAGEMENT (TO APPEAR), 1994. [s.n.].

[PIT 98] PITOURA, E.; SAMARAS, G. Data management for mobile computing. In: KLUWER ACADEMIC PUBLISHERS, 1998. [s.n.]. v.10.

[PIT 99] PITOURA, E.; CHRYSANTHIS, P. K. Exploiting versions for handling updates in broadcast disks. In: THE VLDB JOURNAL, 1999. [s.n.]. p.114–125.

[RAP 95] RAPELI, J. Umts: targets, system concept, and standardization in a global framework. **IEEE Wireless Communications**, [S.l.], v.2, p.20–28, 1995.

[RAT 96] RATNER, D.; POPEK, G. J.; REIHER, P. The ward model: A replication architecture for mobile environments. **Technical Report CSD-960045**, [S.l.], p.1–5, 1996.

[SAT 01a] SATYANARAYANAN, M. Pervasive computing: vision and challenges. [S.l.], v.8, n.4, p.10–17, 2001.

[SAT 01b] SATYANARAYANAN, M.; NARAYANAN, D. Multi-fidelity algorithms for interactive mobile applications. **Wireless Networks**, [S.l.], v.7, n.6, p.601–607, 2001.

[STE 98] STEVENS, W. R. **UNIX network programming**. Prentice Hall, 1998.

[Sun 04a] Sun Microsystems. **Java 2 Platform, Micro Edition (J2ME)**. http://www.umts-forum.org/servlet/dycon/ztumts/umts/Live/en/umts/Home, Acessado em 01/06/2004.

[Sun 04b] Sun Microsystems. **Java 2 Platform, Micro Edition (J2ME) JSR 68 Overview**. http://java.sun.com/j2me/overview.html, Acessado em 01/06/2004.

[UMT 04] UMTS Forum. **UMTS Forum**. http://www.umts-forum.org/servlet/dycon/ztumts/umts/Live/en/umts/Home, Acessado em 05/05/2003.

[VOR 02] VORA, A. et al. A mobile cache consistency protocol using shareable read/write time locks. In: NINTH INTERNATIONAL CONFERENCE ON PARALLEL AND DISTRIBUTED SYSTEMS, 2002. IEEE. p.284–290.

[WAN 91] WANG, Y.; ROWE, L. A. Cache consistency and concurrency control in a client/server dbms architecture. In: PROCEEDINGS OF THE 1991 ACM SIGMOD INTERNATIONAL CONFERENCE ON MANAGEMENT OF DATA, 1991. ACM Press. p.367–376.

[WAN 03] WANG, Z. et al. Saccs: scalable asynchronous cache consistency scheme for mobile environments. In: 23RD INTERNATIONAL CONFERENCE ON DISTRIBUTED COMPUTING SYSTEMS WORKSHOPS, 2003. IEEE. p.797–802.

[WU 96] WU, K.-L.; YU, P. S.; CHEN, M.-S. Energy-efficient caching for wireless mobile computing. [S.l.], v.26, n.1, p.336–343, 1996.

[XU 03] XU, J.; TANG, X.; LEE;, D. L. Performance analysis of location-dependent cache invalidation schemes for mobile environments. In: IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING, 2003. IEEE. v.15, p.474–488.

[YE 98] YE, T.; JACOBSEN, H.-A.; KATZ, R. Mobile awareness in a wide area wireless network of info-stations. In: PROCEEDINGS OF THE 4TH ANNUAL ACM/IEEE INTERNATIONAL CONFERENCE ON MOBILE COMPUTING AND NETWORKING, 1998. ACM Press. p.109–120.

[YUE 00] YUEN, J. C.-H. et al. Cache invalidation scheme for mobile computing systems with real-time data. **SIGMOD Rec.**, [S.l.], v.29, n.4, p.34–39, 2000.

Apêndice A

Publicações

Título: Estratégias para Coerência de Cache em Ambientes de Computação Móvel
Evento: III Simpósio de Informática da Região Centro do RS
Local: Santa Maria - RS
Data: Agosto de 2004
Autores: Carla Diacui Medeiros Berkenbrock e Mario Antonio Ribeiro Dantas

Título: Análise Comparativa entre Estratégias de Coerência de Cache em um Ambiente de Computação Móvel
Evento: 3rd *International Information and Telecommunication Technologies Symposium*
Local: São Carlos - SP
Data: Dezembro de 2004
Autores: Carla Diacui Medeiros Berkenbrock e Mario Antonio Ribeiro Dantas

Título: *Investigation of Cache Coherence Strategies in a Mobile Client/Server Environment*
Evento: *International Conference on Computational Science* 2005 (ICCS 2005)
Local: Atlanta - USA

Data: Maio de 2005

Autores: Carla Diacui Medeiros Berkenbrock e Mario Antonio Ribeiro Dantas

Título: *Performance Analysis of Cache Coherence Strategies over a Wireless Environment*

Evento: *19th International Symposium on High Performance Computing Systems and Applications*

Local: Ontario - Canada

Data: Maio de 2005

Autores: Carla Diacui Medeiros Berkenbrock e Mario Antonio Ribeiro Dantas

Apêndice B

Configuração do Ambiente de Desenvolvimento

Nesta seção serão apresentados alguns dos programas utilizados para configuração do ambiente de desenvolvimento.

B.1 Ambiente de Software

B.1.1 *Java 2 Micro Edition* (J2ME)

A Plataforma *Java 2 Micro Edition* fornece ambiente para desenvolvimento em dispositivos com memória, tela e poder de processamento limitados. J2ME inclui a máquina virtual Java e um conjunto de APIs padrão do Java definidos através da *Java Community Process* [Sun 04b].

A arquitetura do J2ME define *configurations* (configurações), *profiles* (perfis) e *optimal packages* (APIs opicionais). Esta divisão permite ao desenvolvedor conhecer informações específicas sobre as diferentes famílias de dispositivos e as APIs disponíveis em cada uma delas [Mun 04].

Sua configuração é composta de uma máquina virtual e um conjunto reduzido de bibliotecas. Estas bibliotecas fornecem as funcionalidades básicas para uma grande variedade de dispositivos que compartilham características similares. Entre as configurações dispo-

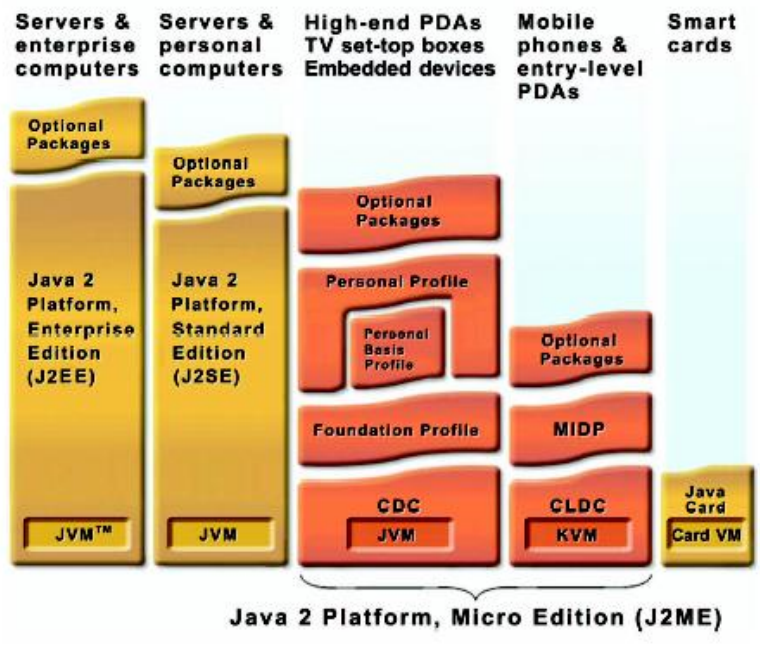

Figura B.1: A arquitetura do J2ME [Sun 04a]

níveis para o J2ME destacam-se:

- *Connected, Limited Device Configuration* (CLDC): A CLDC consiste em uma máquina virtual reduzida (KVM) e um conjunto de classes mais apropriadas para dispositivos conexões de rede intermitentes, processadores lentos e memória limitada. Nesta dissertação foi utilizado o CLDC versão 1.1.

- *Connected Device Configuration* (CDC): O CDC inclui todas as características da máquina virtual java. Ele foi desenvolvido para dispositivos com UCP de 32 bits e no mínimo 2MB de memória disponível para a plataforma java e aplicações associadas.

De forma a fornecer um completo ambiente de execução para categorias de dispositivos específicas, configurações devem ser combinadas com um conjunto de APIs, ou *profiles* que definam o ciclo de vida da aplicação, a interface com o usuário, e acesso a características especificas dos dispositivos, conforme descrito abaixo:

- *Mobile Information Device Profile* (MIDP): O perfil MIDP fornece funcionalidades requeridas pelas aplicações movéis, incluindo interface com o usuário, acesso a rede e armazenamento local. Combinado com o CLDC, o MIDP fornece um completo ambiente de execução java e alavanca as capacidades dos dispositivos portáteis e minimiza o consumo de memória e energia.

- *Foundation Profile* (FP): O FP é o perfil de mais baixo nível do CDC. Ele é utilizado em dispositivos de rede sem interface gráfica.

- *Personal Profile* (PP): O PP é o perfil CDC direcionado a dispositivos que requeiram completo *Graphical User Interface* (GUI) ou suporte *Internet Applet*.

- *Personal Basis Profile* (PBP): O PBP é um subconjunto do PP, fornecendo um ambiente de aplicação para conectar à rede dispositivos que suportem níveis básicos de representação gráfica ou que requeiram a utilização de ferramentas gráficas especializadas para determinadas aplicações.

A plataforma J2ME pode ser estendida pela combinação de vários pacotes com o CLDC, CDC e seus correspondentes perfis. As APIs opcionais permitem a utilização de tecnologias como: *bluetooth*, *web services*, *wireless messaging*, multimídia, e conexão com o banco de dados.
Nesta dissertação, para o desenvolvimento da aplicação no cliente móvel, optou-se pela utilização do MIDP 2.0. Esta escolha foi realizada pelo fato desta versão do MIDP fornecer suporte a uma maior variedade de protocolos de comunicação, conforme abordado na seção B.1.2.

B.1.2 Protocolos de Comunicação

A versão 1.0 do *Mobile Information Device Profile* (MIDP) não suporta rede de baixo nível para *sockets* TCP/IP e conjunto de dados (datagramas) UDP/IP [Sun 04b]. Esta deficiência foi suprida na especificação MIDP 2.0, como resposta às necessidades das redes de computadores da 2.5G e 3G. Este suporte esta baseado no *Generic Connection Framework* (GCF) do *Connected Limited Device Configuration* (CLDC).

B.1.3 CLDC *Generic Connection Framework*

Os pacotes java.io.* e java.net.* do J2SE necessitam de uma quantidade de memória que ainda não esta disponível nos dispositivos portáteis de mão, como *Palmtops*. Conforme mencionado pela SUN MICROSYSTEMS [Sun 04b], dispositivos portáteis possuem mecanismos específicos para comunicação. Desta forma, o GCF fornece um único conjunto de abstrações que podem ser utilizadas para programar várias formas de comunicação, ao invés de utilizar diferentes abstrações para cada protocolo. Isto pode ser exemplificado na tabela B.1, onde a sintaxe do *open* mantém isolada as diferenças existentes entre as configurações de um protocolo e outro.

Tabela B.1: Algumas formas de conexão

Tipo de Conexão	Método utilizado	Exemplo
HTTP	Connector.open	Connector.open("http://java.sun.com/wireless")
Socket	Connector.open	Connector.open("socket://java.sun.com:port")
Datagrama	Connector.open	Connector.open("datagram://java.sun.com:port")

B.1.4 *Java 2 Standard Edition* (J2SE)

Existem dois produtos principais na família J2SE: *Java 2 Runtime Environment, Standard Edition (JRE)* e *Java 2 Software Development Kit, Standard Edition (SDK)*. O JRE fornece as do APIs java, a máquina virtual java, e outros componentes necessários na execução de *applets* e aplicações escritas na linguagem de programação java. O *Java 2 SDK* contém ferramentas como compiladores e *debuggers* necessários para o desenvolvimento de *applets* e aplicações.
Nesta dissertação foi utilizada a versão do 1.4.2 do J2SE.

B.1.5 Mckoi

O Mckoi SQL [Mck 04] é um sistema de banco de dados implementado em Java. Ele oferece funcionalidades como transações, gatilhos e integridade referencial. Este programa

pode operar como multi-client, servidor de banco de dados multi-threaded, ou pode ser utilizado como um banco de dados em uma aplicação java.

O Mckoi foi utilizado nesta dissertação, por ser de código aberto e fornecer e suprir todas as características necessárias ao ambiente desenvolvido.