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

Daniel Pezzi da Cunha

UM ESTUDO DAS ESTRATÉGIAS DE REPLICAÇÃO E RECONCILIAÇÃO DE BANCO DE DADOS MÓVEIS EM UM AMBIENTE WIRELESS

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

Orientador: Mário Antônio Ribeiro Dantas

Florianópolis-SC, Novembro de 2003.

UM ESTUDO DAS ESTRATÉGIAS DE REPLICAÇÃO E RECONCILIAÇÃO DE BANCO DE DADOS MÓVEIS EM UM AMBIENTE WIRELESS

Daniel Pezzi da Cunha

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

Prof. Dr. Raul Sidnei Wazlawick
Coordenador

Banca Examinadora

Prof. Dr. Mário Antônio Ribeiro Dantas

Prof. Dr. Nelson Francisco Favilla Ebecken

Prof. Dr. Rosvelter João Coelho da Costa

"A mente que se abre a uma nova idéia, jamais
volta a seu tamanho original".

(Albert Einstein)

Dedico este trabalho aos meus mestres de vida, meu pai Gilberto e minha mãe Nair. A minha noiva, amiga e companheira Taís, que em todos os momentos desta jornada me prestou o seu irrestrito apoio, sua compreensão e o seu enorme amor.

Ao meu orientador, Prof. Mário Dantas, que estimulou e indicou, de forma esclarecedora, as diretivas que conduziram ao sucesso deste trabalho.

Aos colegas do LABWEB, LACPAD, LAPS e LSD que, com alegria e muita amizade tornaram muitos momentos inesquecíveis que sempre me recordarei com felicidade.

Ao Prof. Fernando Gauthier e Vera Sodré, pela simpatia e prestatividade, quer seja com relação a assuntos da secretaria, quer seja com palavras de carinho e incentivo.

SUMÁRIO

ÍNDICE DE FIGURAS

ÍNDICE DE TABELAS

RESUMO

Recentemente tem sido verificado um notável crescimento na utilização da computação móvel influenciado pelo desenvolvimento de computadores portáteis com maior capacidade de processamento, armazenamento e transferência de dados. No âmbito de sistemas de banco de dados é possível permitir a manipulação otimizada de dados replicados de forma *ad-hoc* sem a obrigatoriedade da interligação a uma base de dados consolidada. Porém, algumas considerações importantes precisam ser analisadas, tais como: o grau de divisão e propagação de dados, a eficiência das respostas às requisições e a distribuição do controle. O problema fundamental é a combinação adequada destas e de outras ponderações para que se obtenha um sistema confiável e com alto desempenho. Neste trabalho é apresentada uma análise de diferentes estratégias de replicação e reconciliação de dados baseando-se em um estudo de caso experimental, considerando dois ambientes operacionais convencionais sob o paradigma da comunicação wireless. Os resultados indicam soluções para o desenvolvimento de sistemas de bancos de dados móveis que operam em condições de comunicação descontinuada. A análise de metodologias diferenciadas para gerenciamento de réplicas de dados permitiu concluir que é possível a obtenção de um nível satisfatório de funcionalidades com a combinação de estratégias de replicação e reconciliação diferenciadas conforme o tipo de solução pretendida.

PALAVRAS-CHAVE: Computação móvel, Banco de dados, Replicação de dados.

ABSTRACT

Nowadays it has been verified an intensive use of the mobile computing mainly because of the development of portable computers with larger processing and storage capacity, and large ration of data transfer. In the database systems scenario it is possible to allow an optimistic manipulation of replicated data in ad-hoc fashion without the necessary interconnection to a consolidated database. However, some important considerations need to be analyzed such as: division degree and propagation of data, efficiency answers to requisitions and distribution control. The fundamental problem is the combination of these factors and another reflections, leading to obtained a reliable system and with high performance. In this work we present an analysis of different replication strategies and reconciliation of data considering an experimental study case, where two conventional operational environments are used under the paradigm of wireless communication. Our results indicate interesting answers for the development of mobile databases systems that operate in conditions of discontinued communication. The analysis of methodologies differentiated for management of responses of data allowed to end that is possible obtaining of a satisfactory level of functionality with a combination of replication strategies and differentiated reconciliation according to the type of intended solution.

KEYWORDS: *Mobile computing, Database, Replication of data.*

1. INTRODUÇÃO

Atualmente, com o advento das redes de comunicação sem fio e a miniaturização dos recursos computacionais, usuários móveis tem adquirido maior flexibilidade na manipulação das suas bases de dados. Com a utilização da tecnologia *wireless*[1] a partir de PDAs (*Personal Digital Assistants*), por exemplo, pode-se obter o acesso a informações provenientes da rede fixa em bancos de dados distribuídos. Essa mobilidade provoca um estilo novo de computação que exige a criação de novas metodologias para gerenciamento dos dados. Alguns dos mecanismos que compõem tais exigências envolvem a criação e a manipulação de réplicas de dados que devem ser otimizadas de tal forma a permitir operações consistentes viabilizando o processo de reconciliação de dados em bases consolidadas. Além disso, podem ser determinantes para a realização de certas ações, principalmente aquelas que implicam em operações sobre comunicação descontinuada.

A geração de réplicas de itens de dados e a admissão de operações em modo de desconexão permitem que transações possam ser feitas com base em registros contidos na memória de um dispositivo portátil. Este fato promove ao usuário um alto desempenho de processamento, extensa disponibilidade de informações e ampla mobilidade. Além disso, com a existência de réplicas de dados, não é preciso haver comunicação contínua entre unidades móveis e fixas, permitindo a utilização de computadores portáteis de baixo custo.

A distribuição e o gerenciamento de cópias de dados em um sistema móvel é extremamente desafiador. Diversas metodologias aplicadas em bancos de dados distribuídos podem ser adaptadas para bancos de dados móveis, porém, em alguns casos, tornam-se inviáveis por gerarem situações de inconsistência. É conveniente a adoção de estratégias para a concepção e o tratamento de cópias de itens de dados para que, ao serem reintegradas ao banco de dados consolidado, não comprometam o estado

[1] Ambiente de transmissão de dados para sistemas computacionais conectados sem fio.

global do sistema. E algumas considerações importantes precisam ser analisadas, tais como o grau de divisão dos dados, a eficiência das respostas às requisições de usuários, a forma de propagação dos dados e a distribuição do controle sobre os bancos de dados.

Este trabalho compreende o estudo e a experimentação de estratégias de replicação e reconciliação de dados em um ambiente de computação móvel que opera em condição de comunicação descontinuada. O enfoque foi dado à investigação de novas soluções para problemas envolvendo o gerenciamento de bancos de dados sob o paradigma da computação móvel. A concepção das réplicas de dados envolveu o estudo com base em duas linhas de inquirição, são elas, o modelo de replicação otimista e o pessimista. Sob o paradigma da comunicação *wireless*, foi realizada uma análise das formas possíveis de reconciliação e manutenção da consistência do banco de dados. O trabalho também envolveu o desenvolvimento de um protótipo experimental visando a aplicação de proposições que visam a otimização na sincronização de dados. A arquitetura empregada no experimento incorporou ambientes operacionais convencionais e recursos de *hardware* de baixo custo com o propósito de ser aplicada em um projeto na área médica.

O trabalho é organizado como segue. O Capítulo 2 engloba a fundamentação sobre computação móvel, incluindo suas características essenciais, categorias de aplicações, principais arquiteturas, requisitos de *hardware* e meios de comunicação sem fio. O Capítulo 3 compreende uma investigação aprofundada sobre a concepção, organização, tipos, modelos e propostas de replicação de dados existentes. O Capítulo 4 envolve o estudo da reconciliação de dados replicados, com base em modelos de transferência e gerenciamento, alternativas para disseminação de dados e outras características pertinentes a ações de reintegração de bancos de dados. O Capítulo 5 expõe o ambiente de experimentação e o protótipo desenvolvido considerando as estratégias de replicação e reconciliação abordadas nos capítulos anteriores. O Capítulo 6 contém as conclusões e perspectivas de continuação.

2. COMPUTAÇÃO MÓVEL

A computação móvel, também chamada de computação nômade, é um novo paradigma computacional advindo da tecnologia de rede sem fio e sistemas distribuídos. Permite que o usuário dessa tecnologia obtenha uma comunicação flexível e acesso a diversos serviços, desde que localizado dentro dos limites de uma infra-estrutura de comunicação. A visão de diversos computadores autônomos é transparente para os usuários e a distribuição das atividades é feita automaticamente pelo sistema. Além disso, a especialização das tarefas computacionais é estabelecida conforme a natureza da função de cada computador.

A computação móvel é a combinação de três importantes e inter-relacionadas propriedades (LIU, MARLEVI & MAGUIRE, 1996): computação, comunicação e mobilidade, conforme representada na Fig.1. A computação inclui os dispositivos portáteis que provêem o processamento necessário para a realização de operações móveis. A comunicação compreende a troca de mensagens entre os elementos do sistema através de uma rede do tipo *wired* ou *wireless*. A mobilidade está relacionada à mudança de localização dinâmica de um usuário móvel.

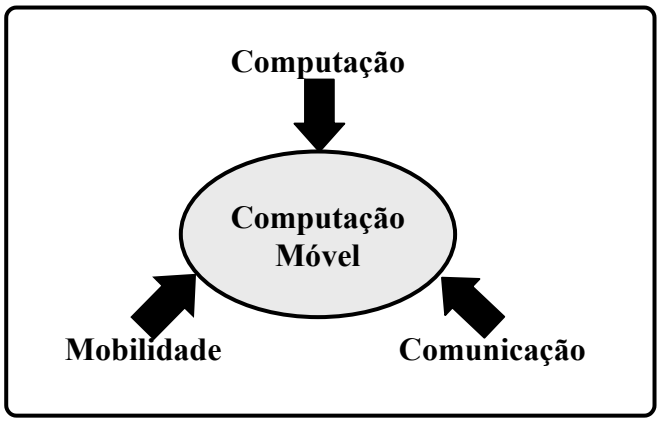

FIGURA 1 – Componentes fundamentais da computação móvel.
Fonte: (LIU, MARLEVI & MAGUIRE, 1996).

Para MATEUS, LOUREIRO (1998) a computação móvel amplia os conceitos tradicionais de computação distribuída por utilizar a comunicação sem fio, que elimina a necessidade de um usuário manter-se conectado a uma estrutura de rede fixa, cuja localização é estática.

A iniciativa do desenvolvimento da tecnologia que envolve a combinação da computação pessoal com a mobilidade surgiu em 1946, através da *Illinois Bell Telephone Company*. Conforme BATES, GREGORY (1997), inicialmente tratava de serviços de telefonia móvel, que permitia aos usuários de veículos automotores comunicarem-se com um sistema telefônico. Com isso, foi permitido o estabelecimento da comunicação bidirecional sem fio. A partir deste avanço tecnológico, diversos centros de pesquisa mobilizaram-se na elaboração de recursos para tratar e fornecer serviços que garantissem a troca de informações entre elementos móveis.

A comunicação *wireless* dá suporte a computação móvel através de transmissões de dados via satélite, serviços de rádio, serviços móveis públicos, serviços para comunicação pessoal, dentre outras facilidades. Tecnologias essas, associadas a equipamentos como os PDAs (*Personal Digital Assistants*), permitem ao usuário um conjunto de serviços oferecidos em um sistema distribuído de computadores estáticos com mobilidade de acesso às informações.

Conforme definido em BARBARÁ (1999), as unidades móveis podem ser também agrupadas em sub-redes, cada qual mantida por uma estação de suporte à mobilidade composta por um ponto de acesso a rede *wireless* (Fig. 2). Cada ponto de acesso provê comunicação entre as unidades fixas (*hosts*), ligadas por rede *wired,* e as unidades móveis. Além disso, diversas funcionalidades podem ser agregadas com emprego de bancos de dados distribuídos e de um sistema de comunicação interoperável, como o descrito em ZHANG, HELAL, HAMMER (2003).

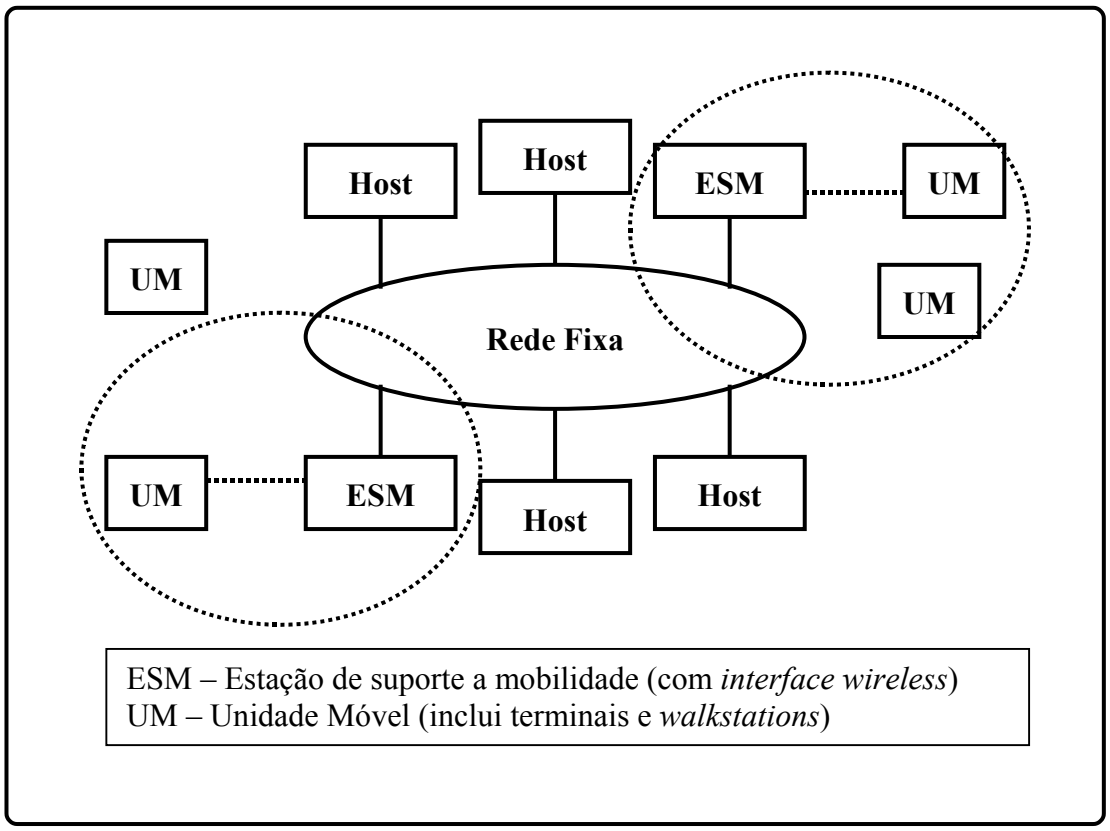

FIGURA 2 – Um ambiente de computação móvel.
Fonte: Adaptação de BARBARÁ (1999).

2.1. Características

2.1.1. Portabilidade

Um PDA é projetado para ser pequeno, leve, durável, operacional em uma variedade de condições e requerer o mínimo de consumo de energia, assegurando longa vida útil para a bateria. Algumas concessões podem ser feitas em cada uma dessas qualidades para que se chegue a uma boa relação de funcionalidade/desempenho. Para BARBARÁ (1999) isso envolve a adaptação às limitações impostas pela arquitetura dos dispositivos móveis e das características relativas à computação móvel.

A seguir, são listadas algumas dificuldades de projeto impostas pela portabilidade de um dispositivo móvel:

- **Energia**: a bateria enfrenta problemas de tamanho/peso e de necessidade de recarga. Técnicas para minimização do consumo de energia podem aumentar a portabilidade e aumentar a durabilidade da bateria.

- **Riscos de perda e extravio de dados**: a portabilidade aumenta o risco de danos físicos, acessos não autorizados, perda e roubo de um dispositivo móvel, tornando tais computadores menos seguros e confiáveis. Para reduzir estes riscos, o usuário deve ter uma preocupação ainda maior com operações de *backup* e o armazenamento de dados importantes em uma base remota que possa ser acessada quando necessário.

- **Interface limitada**: a *interface* com o usuário é através de botões e canetas *stylus*. Para as aplicações que necessitam da abertura de várias telas ao mesmo tempo, os PDAs são inapropriados, levando a desafios de projetos gráficos e, muitas vezes, obrigando a redução de funcionalidades em um sistema móvel.

- **Capacidade de armazenamento**: o espaço de armazenamento em um computador móvel é comprometido pelo tamanho físico e pela disponibilidade de energia. Algumas estratégias para contornar o problema são aplicadas, tais como: compressão automática de arquivos, acesso a dados armazenados remotamente, compressão de páginas de memória virtual, e compartilhamento de bibliotecas de código. Uma abordagem para reduzir o tamanho do código em programas executados em um dispositivo móvel é a interpretação de linguagens de *scripts* ao invés de execução de objetos de código, que são tipicamente maiores.

2.1.2. Mobilidade

Na atmosfera que envolve a computação móvel é estabelecido um ambiente altamente dinâmico. Isso é explicado pelas diferenças estruturais de um sistema móvel, assim como as variações de tráfego decorrente do deslocamento e das requisições de usuários (LI, 2001). A capacidade de uma unidade móvel mudar de localidade enquanto conectada à rede aumenta a volatilidade das informações manipuladas. Alguns dados considerados estáticos para computadores fixos poderão se tornar dinâmicos em computadores móveis.

O grande desafio que envolve sistemas móveis é a criação de algoritmos e estruturas de dados para minimizar o custo de comunicação quando é estabelecida uma gerência de localização de uma unidade móvel. O emprego adequado de protocolos de comunicação influencia diretamente na eficiência de um sistema deste tipo. A carga da aplicação e a heterogeneidade não devem ser desprezadas, uma vez que determinam a disponibilidade de serviços aos usuários da rede sem fio. Tais usuários podem ter equipamentos que adotem diferentes protocolos, com velocidades variáveis. Também são introduzidas questões relacionadas à segurança, que introduzem mais problemas que uma rede fixa já que redes sem fio tendem a ser mais vulneráveis à interceptação de mensagens e ao rastreamento de estações móveis.

Tendo em vista os possíveis cenários que envolvem o espectro da computação móvel, dependendo da mobilidade de cada elemento do sistema podem ser definidas as seguintes subdivisões:

- *Nomadic computing*: o *hardware* pode se mover.

- *Wireless computing*: o usuário pode se mover entre um conjunto fixo de estações conectadas à rede.

- *Mobile code/Mobile agent*: a aplicação pode se mover.

- *Pervasive computing*: o usuário se locomove executando aplicações com dados e código móveis.

Um usuário pode obter uma taxa de transmissão de dados em uma área e, com o seu deslocamento, alterar essa taxa de transmissão. No entanto, para MATEUS, LOUREIRO (1998) essa versatilidade é dependente de diversos fatores, tais como: determinação da localidade de um *host* móvel, disponibilidade de canais de comunicação, interferências nas transmissões de dados, projeto de protocolo, heterogeneidade, segurança e energia disponível na unidade móvel.

Quando aplicada sob o modelo *wireless computing* a mobilidade introduz alguns problemas, tais como:

- **Migração de endereço**: com o deslocamento das unidades móveis, diferentes pontos de acesso à rede são utilizados. Um sistema distribuído deve ser capaz de armazenar o endereço de um *host* e invalidar entradas defasadas sem afetar o desempenho do processamento.

- **Informação dependente de localidade**: como o endereço de rede de um computador móvel muda dinamicamente, o caminho de comunicação aumenta quando o mesmo visita uma célula vizinha e sua localização atual afeta parâmetros de configuração assim como respostas a consultas.

- **Migração de localidade**: a distância física entre dois pontos em um ambiente móvel não reflete necessariamente à distância das duas redes. O tamanho do caminho de comunicação pode crescer desproporcionalmente, levando à existência de ações intermediárias que aumentam a latência e o risco de desconexões.

2.1.3. Adaptação

Uma das propriedades da computação móvel é a capacidade de operar sobre um ambiente de comunicação sem fio. Esse ambiente tem características diferentes daquelas encontradas em redes fixas, tais como: maior taxa de erro, menor velocidade de transmissão e maior latência de rede, que podem depender da localização geográfica atual de um dispositivo móvel. Além disso, os dispositivos portáteis empregados na computação móvel freqüentemente têm limitações quanto à velocidade de processamento e capacidade de armazenamento.

Para oferecerem serviços aceitáveis em um ambiente móvel, as unidades clientes precisam de algum mecanismo de gerenciamento das alterações do ambiente e reagirem a estas mudanças (MENKHAUS, 2002). Isso pode ser feito através da adaptação, que é a capacidade de um algoritmo fornecer diferentes saídas válidas dependendo das características do ambiente onde o dispositivo móvel se encontra. Em uma rede fixa, onde as condições são praticamente estáveis, isso não é comum. No entanto, em uma rede *wireless*, onde o ambiente é mutável, essa solução é extremamente importante, pois deve garantir que operações possam ser finalizadas com êxito, mesmo que ocorram freqüentes desconexões.

Segundo SATYANARAYANAN, NARAYANAN (1999), existem três planos de adaptação, descritos a seguir:

- ***Laissez-Faire***: a aplicação móvel é responsável por toda a adaptação realizada.

- ***Application-Transparent***: o sistema de gerenciamento fixo é responsável pela adaptação. Para a aplicação, a adaptação é feita de forma transparente.

- **Application-Aware**: estratégia colaborativa onde a adaptação é feita tanto pelo sistema fixo quanto pela aplicação móvel.

Uma arquitetura de adaptação em computação móvel pode ser implementada, por exemplo, através de um módulo (ou biblioteca). Para se adaptar melhor a cada aplicação esse módulo pode permitir a incorporação de adaptações através de *plug-ins*. Vários aspectos podem ser observados para a definição de uma arquitetura de adaptação, tais como:

- **Recursos disponíveis**: a escalabilidade de um sistema aplicado em um ambiente móvel depende dos recursos oferecidos na rede fixa. Ao receber requisições, um *host* servidor deve ser capaz de processar de forma rápida sem comprometer o funcionamento das unidades móveis. Já os recursos disponíveis na rede móvel precisam congregar funcionalidades reduzidas, adaptando-se às restrições de processamento e de energia.

- **Dados**: para cada tipo de dado a ser transmitido há um tipo de adaptação diferente. Em um serviço de FTP (*File Transfer Protocol*), por exemplo, as variações da taxa de transmissão não afetam a aplicação. Já na transmissão de dados multimídia a ocorrência de variações na taxa de transmissão obriga a realização de adaptações conforme a largura de banda disponível.

- **Desempenho**: uma adaptação precisa ser planejada de forma a não ocasionar perda no desempenho geral do sistema. Na detecção de incapacidades é satisfatório que sejam geradas notificações ao usuário móvel. Além disso, uma transferência de dados deve incluir técnicas que reduzam seu volume, adaptando-se as condições de comunicação e otimizando a velocidade de transmissão.

- **Broadcast/Multicast**: em redes fixas a maior parte da comunicação é feita através de transmissões *unicast*. Devido às características do ambiente móvel, a utilização de técnicas de *broadcast* e *multicast* é altamente recomendada. A utilização destas técnicas evita que um mesmo dado seja enviado várias vezes em um ambiente onde a largura de banda é reduzida.

2.1.4. Gerência de energia

Em dispositivos móveis, o aproveitamento eficiente da energia é fundamental. As baterias costumam ter durabilidade da carga muito pequena, comprometendo a disponibilidade do recurso de *hardware* (FLINN & SATYANARAYANAN, 1999). Isso se torna um desafio para construtores de equipamentos móveis e para projetistas de sistemas, pois é necessário que ocorra uma minimização do consumo durante o funcionamento de uma unidade móvel.

Para KREMER, HICKS, REHG (2001) a limitação de consumo ocasionou a criação de técnicas, como por exemplo a transmissão de um valor exato de potência na transmissão de um sinal, o controle da iluminação do visor do aparelho, o desligamento da unidade de armazenamento quando não está sendo usada e a confecção de processadores que consomem menos energia em modo *doze*[2].

2.1.5. Fraca conectividade

Na computação móvel podem existir momentos em que a unidade móvel esteja momentaneamente desconectada da unidade fixa. Tal situação pode ser ocasionada pela desconexão voluntária, no intuito de liberar canais de comunicação, por problemas de comunicação como variações na taxa de sinal ruído, pela quantidade de energia disponível no dispositivo portátil ou pela distribuição da largura de banda da rede de comunicação. Para tanto, MUMMERT (1996) descreve que é preciso existir suporte a desconexão através de mecanismos que permitam ao cliente móvel manipular itens de dados de forma autônoma.

Segundo PITOURA, SAMARAS (1998), operações de desconexão envolvem três estados distintos (Fig. 3):

- ***Hoarding***: pré-carregamento de itens de dados na unidade móvel. Com isso, os dados podem ser replicados ou movidos para uma *cache* local, podendo

[2] Modo em que um processador fica inerte a espera de uma nova requisição. Este estado de espera é aplicado na computação móvel para manter a durabilidade da energia de um computador portátil.

ficar inacessíveis a outros membros da rede. Os tipos de dados acumulados dependerão do modelo de aplicação que o sistema está utilizando.

- **Operações locais**: período em que a unidade desconecta-se da rede fixa, podendo apenas realizar operações sobre dados localizados na memória local do dispositivo após o *hoarding*. Caso tais dados não estejam disponíveis momentaneamente pode ser estabelecida uma fila para execução de operações futuras que será consultada na etapa de reintegração.

- **Reconciliação**: durante a re-conexão os dados são atualizados na unidade fixa conforme as informações decorrentes dos nodos que operaram sobre esses dados. Isso gera uma situação concorrente que deve ser tratada por uma semântica de serialização adequada, garantindo a não ocorrência de inconsistências nos dados atualizados.

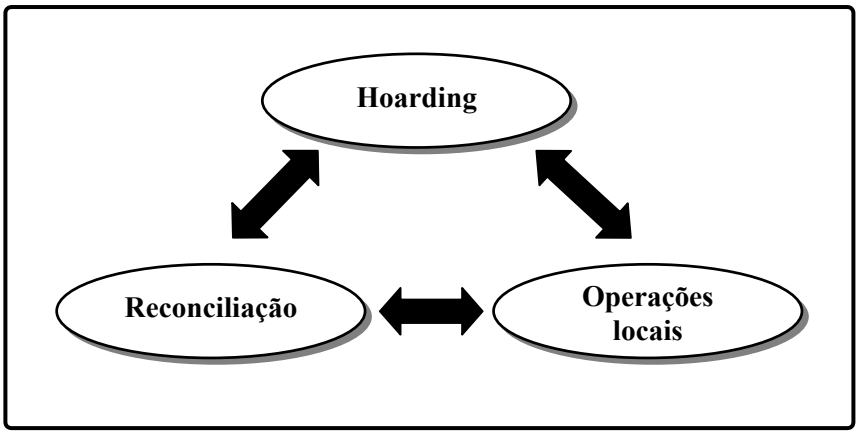

FIGURA 3 – Os estados de uma operação de desconexão.
Fonte: (PITOURA & SAMARAS, 1998).

Quando a conexão entre os elementos da rede é freqüentemente perdida por curtos períodos de tempo pode influenciar na execução de operações, acarretando em perda de desempenho. Para MATEUS, LOUREIRO (1998) a fraca conectividade é identificada quando se alternam períodos de transferência de dados entre a unidade fixa e a unidade móvel ou quando ocorre o caso extremo de falta total de conectividade entre esses elementos. Os efeitos da desconexão são refletidos diretamente na execução das

operações, que tendem a não serem finalizadas. Isso também poderá afetar a *cache* do cliente, principalmente quando ele deseja validar algum valor no servidor. Nesse caso, a unidade depende de uma resposta efetiva para continuar uma determinada tarefa.

Os problemas descritos anteriormente devem ser atenuados com a utilização de protocolos de comunicação adequados e com a disponibilização de uma vasta largura de banda para permitir uma melhor transferência de dados. Podem ser também aplicados mecanismos que visam a reintegração da conexão de forma branda. Outro método é a busca de informações em *sites* replicados, identificando-se qual possui melhor disponibilidade para estabelecimento de comunicação. Os principais protocolos que dão suporte a problemas desse tipo são:

- **Protocolo de desconexão**: tem a finalidade de assegurar que o dispositivo móvel permaneça atuando sobre os dados durante um período de desconexão.

- **Protocolo de desconexão parcial**: possui a característica de dispor condições autônomas antecipadas para uma provável ocorrência de desconexão.

- **Protocolo de *handoff***: tem a capacidade de retransmitir os dados pertencentes a uma unidade móvel que está se deslocando entre as células de uma área de cobertura.

2.1.6. Replicação de dados

A essência de computação móvel consiste na manutenção das funcionalidades oferecidas por computadores portáteis enquanto os usuários migram entre diferentes localizações. A necessidade de manter a conectividade entre os elementos do sistema de forma transparente para o usuário final gera um aumento significativo da carga de mensagens que transitam entre o terminal móvel e a rede de computadores. Para amenizar esse problema é conveniente que exista a possibilidade do usuário manipular réplicas do banco de dados sem estar conectado a rede sem fio.

Segundo BARBARÁ (1999), a criação de réplicas de um arquivo ou grupo de arquivos tem como objetivos principais o aumento da disponibilidade de dados e um melhor desempenho das aplicações sobre esses dados em unidades móveis. Dessa forma, dispositivos portáteis carregam cópias redundantes das informações contidas no banco de dados distribuído. A replicação é uma técnica estratégica e importante, pois otimiza os recursos de *hardware* e de rede, maximizando a flexibilidade e a escalabilidade em ambientes heterogêneos.

A base de dados que contém os itens a serem replicados é chamada de base consolidada. As bases remotas são aquelas que contêm um subconjunto das informações da base consolidada em um *host*.

A reconciliação é o caminho inverso dos dados replicados. Após serem feitas operações de atualização, inserção ou exclusão sobre cópias de itens de dados nas estações móveis, as modificações são reintegradas na base de dados consolidada, como mostrado na Fig. 4. Tal conjunto de operações permite manter atualizados os registros do sistema de banco de dados, eliminando possíveis inconsistências (BADRINATH & PHATAK, 1998). Além disso, as mudanças são propagadas para os demais locais onde existam réplicas. Estas alterações podem ser disseminadas através de vários tipos de protocolos ou canais de comunicação.

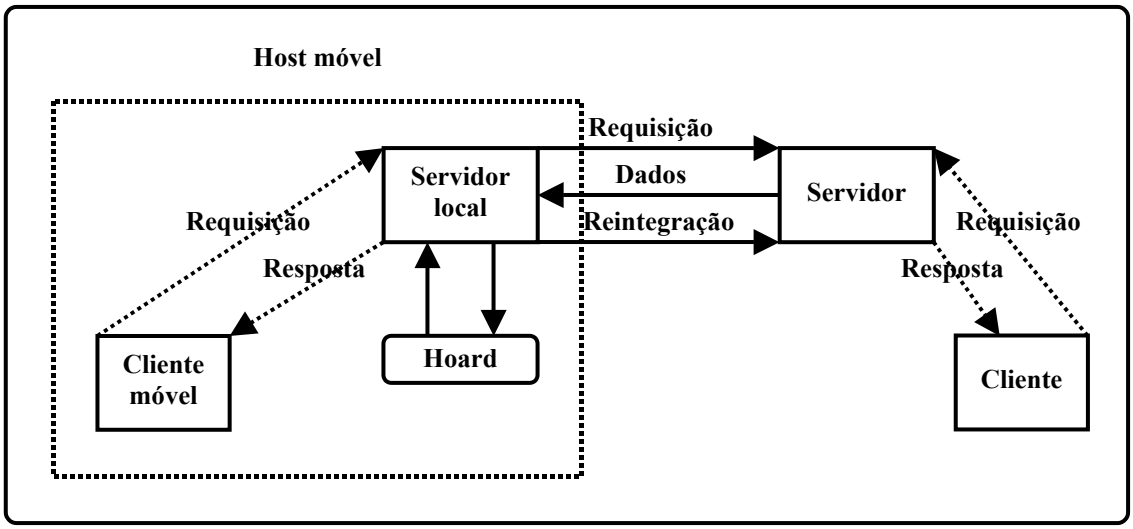

FIGURA 4 – Modelo de reintegração de dados.

Fonte: (BADRINATH & PHATAK, 1998).

O procedimento de *hoard* é o acúmulo das mudanças realizadas no *host* móvel para serem propagadas conjuntamente no instante da reintegração com o servidor. Tal mecanismo visa diminuir a quantidade de requisições entre os elementos do sistema, minimizando os custos envolvidos com a comunicação de rede.

A sincronização entre a base de dados fixa e a consolidada pode ser feita a curta ou longa distância através de vários meios, como infravermelho, satélite, radiofreqüência, *spread spectrum* e por vários protocolos de comunicação de dados, incluindo: HTTP, WSP (*Wireless Session Protocol*), OBEX (*Bluetooth, IrDA*[3]), SMTP, TCP/IP e protocolos proprietários.

2.2. Tipos de aplicações

As aplicações da Computação móvel podem ser divididas em três categorias: aplicações horizontais, aplicações horizontais genéricas e aplicações verticais.

2.2.1. Aplicações horizontais

Correspondem às aplicações destinadas a solucionar processos simples e que requerem poucos custos. Por exemplo, o correio eletrônico, o serviço de *paging*, automação de vendas e multimídia.

2.2.2. Aplicações horizontais genéricas

Esta categoria inclui aplicações que requerem acesso à banco de dados para fornecer grande volume de informações. Segue abaixo alguns exemplos:

- Aplicações para usuários móveis dentro de edifícios;

[3] *Infrared Data Association* – Organização da indústria de fabricantes de computadores, componentes e telecomunicações que estabelecem os padrões para comunicação via infravermelho entre computadores e dispositivos periféricos.

- Servidor de Informação: usuários móveis podem requerer acesso constante a um servidor de banco de dados de uma rede local e interagir com diferentes pacotes de busca *front-end* (por exemplo, o Personal 2000 da Oracle, Power Builder da Sybase) para submeter uma consulta diretamente ao servidor ou através de um *gateway* ou por meio de um *software back-end* do servidor que recupera a informação do banco de dados;

- CAD (*Computer Aided Dispatcher*): o envio auxiliado por computador é uma aplicação muito comum para o envio de informações de serviços representativos para clientes. Existem três tipos de implementação para esta aplicação, conforme descrito a seguir:

    - Modelo simples de serviços representativos: um computador móvel conectado através de uma rede sem fio a um sistema central de CAD pode fazer diversas requisições de serviços. Geralmente CADs são utilizados por grandes lojas, para comunicação entre vendedores e veículos de entrega. Um exemplo é o *dispatcher* (conjunto de rotinas localizados em um servidor) que recebe requisições remotas e envia para os veículos requisitantes;

    - Modelo CAD complexo para segurança pública: agentes de segurança pública como polícia e bombeiros podem ser capazes de responder a situações críticas rapidamente através do uso de GPS (*Global Positioning System*). Nestes casos em que o CAD está integrado com o GPS, um mapa do Sistema de Informação Geográfico (SIG) é ligado a um banco de dados com os números de ruas;

    - *Dispatch* de Táxis: táxis são equipados com terminais adequados e o *dispatcher* envia a posição do cliente ao táxi mais próximo. Combina GPS com redes sem fio MOBITEX e *software* CAD.

- Aplicações marítimas com conexão a sistemas de informações integrados a redes de dados sem fios baseadas em satélites que cobrem mares e lagoas;

- Aplicações baseadas em GPS (*Global Positioning System*) que podem indicar a localização de veículos ou objetos baseando-se em cálculos de sinais constantemente transmitidos de satélites geosíncronos.

- Aplicações baseadas em GLS (*Global Location System*). O GLS é um sistema ativo baseado em um transmissor montado em um veículo que propaga um sinal captado por múltiplas torres de recepção onde a posição geográfica é calculada. Tais aplicações são usadas para rastreamento de veículos, pelas agências de locação de veículos e por centrais meteorológicas.

2.2.3. Aplicações verticais

As aplicações verticais são empregadas de forma específica em grandes corporações como seguradoras, bancos, linhas aéreas, governo e transporte. Algumas áreas que podem envolver aplicações verticais da computação móvel são descritas a seguir:

- Seguradoras: vendedores de seguros podem ser equipados com unidades portáteis para consulta de produtos e preços com mobilidade. Com uma conexão por *modem* as informações adicionais armazenadas em um banco de dados podem ser rapidamente recuperadas.

- Hospitais: médicos e enfermeiros equipados com *laptops* ou *palmtops* podem fazer consultas a um banco de dados para obter informações sobre pacientes e medicamentos. Além disso, podem registrar diagnósticos e gerar receitas.

- Transporte: caminhões podem ser equipados com terminais especiais que utilizam a comunicação baseada em satélites para enviar/receber mensagens e rastrear encomendas. Outra opção é o uso de dispositivos especiais para venda de passagens aéreas e programação de vôos, trazendo melhorias de serviços aos clientes.

2.3. Arquiteturas de computação móvel

2.3.1. Modelo cliente/servidor

A computação móvel agrega-se ao modelo distribuído baseado em uma estrutura conhecida como modelo Cliente/Servidor. Segundo SATYANARAYANAN (1996), são chamadas de aplicações clientes àqueles sistemas computacionais que requisitam serviços para uma aplicação que está sendo executada em um outro sistema computacional, chamada aplicação servidora. Uma parte da comunicação é feita por cabeamento fixo (lado do servidor) até ser estabelecido o enlace sem fio, a partir desse ponto a informação é transmitida por sinais de rádio (meio sem fio) até a aplicação cliente (Fig. 5).

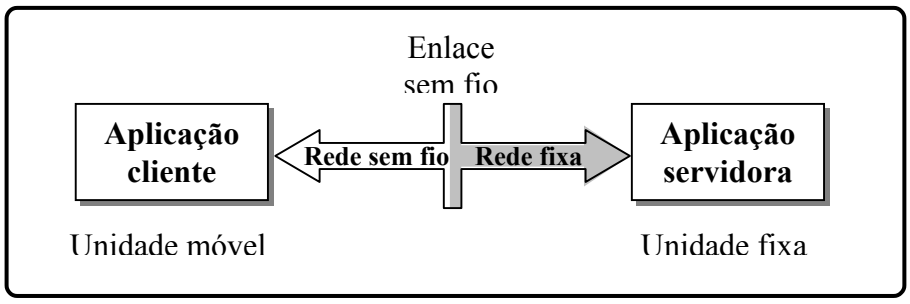

FIGURA 5 – Modelo cliente/servidor.

Fonte: Adaptação de ITO (2001).

De modo geral, um dispositivo móvel tem o papel do cliente e a unidade fixa tem o papel do servidor. Em alguns casos, operações e dados são distribuídos entre diversos servidores fixos que podem ter que se comunicar entre si para atender as requisições de um cliente. Dessa forma, são atribuídas funcionalidades distintas aos servidores de um mesmo sistema, uns dando suporte a informações e serviços e outros sendo utilizados para a replicação de dados, que permite maior disponibilidade e escalabilidade do banco de dados. Em casos especiais, como o modo desconectado (momento em que a unidade móvel não está comunicando-se com a rede fixa), o dispositivo móvel emula funcionalidades de um servidor para poder continuar operando.

Na arquitetura cliente/servidor existem dois mecanismos de comunicação entre seus elementos: troca de mensagens e RPC (*Remote Procedure Call*), entretanto a utilização de primitivas RPC tradicionais é inadequada para sistemas móveis, pois não permitem a ocorrência de desconexões freqüentes da unidade móvel. Para solucionar o problema JOSEPH, TAUBER, KAASHOEK (1997) citam que pode ser aplicado o modelo RPC assíncrono que permite a otimização dos custos de comunicação entre a base móvel e a base fixa. Ao se conectar com a estação fixa, a estação móvel envia chamadas remotas de forma transparente ao usuário. Nesse modelo estendido é atribuído um componente chamado agente, responsável por implementar a otimização de comunicação apropriada. O agente atua como um intermediário, provendo facilidades na transmissão de mensagens. A utilização de um agente favorece o estabelecimento das seguintes associações:

- **Cliente/Agente/Servidor (C/A/S)**: apenas um agente interage sobre a comunicação (Fig. 6), atuando junto à unidade fixa ou junto à unidade móvel, conforme o tipo ação (nível de comunicação ou aplicação) que necessita realizar. Quando ocorre a movimentação do usuário entre células o agente também deve mover-se de forma a ficar o mais próximo possível do cliente e continuar fornecendo recursos. Conforme relatado em TENNENHOUSE, SMITH, SINCOSKIE (1996), o agente opera como um *proxy*, mantendo a comunicação cliente/servidor, tendo acesso a canais de comunicação de alta velocidade e funcionalidades extras. Com isso, muitas ações podem ser otimizadas, tanto do lado do cliente quanto do servidor. O tempo de resposta entre operações remotas pode diminuir, já que a carga de trabalho do servidor é menor. Operações antes realizadas pelo servidor, como por exemplo a compressão de dados, tornam-se responsabilidade do agente, que trata a informação e envia o pacote para o cliente. O modelo C/A/S é mais apropriado para clientes que não possuem muitas demandas de operações e que possuem recursos computacionais limitados, já que a capacidade operacional do agente é limitada.

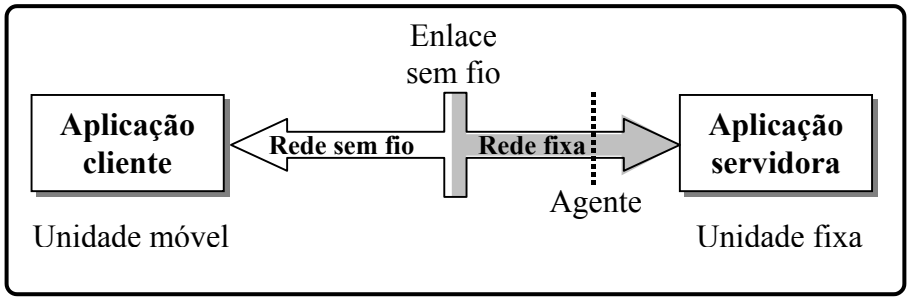

FIGURA 6 – Modelo cliente/agente/servidor.

Fonte: Adaptação de ITO (2001).

- **Cliente/Interceptador/Servidor (C/I/S)**: dois agentes se comunicam para aumentar a eficiência na troca de informações entre a unidade fixa e a unidade móvel, sendo que um localiza-se junto ao cliente e outro junto ao servidor (Fig. 7). Os agentes possuem a denominação de interceptadores por atuarem como porta de entrada e saída de informações tanto do cliente como do servidor, executando otimizações para minimizar os efeitos da transmissão de dados no canal de comunicação (SAMARAS & PITSILLIDES, 1997). A arquitetura C/I/S é indicada para aplicações com alto poder de processamento e armazenamento de informações, visto que é composta por uma camada *middleware* de monitoramento constante de transações entre uma unidade fixa e uma unidade móvel.

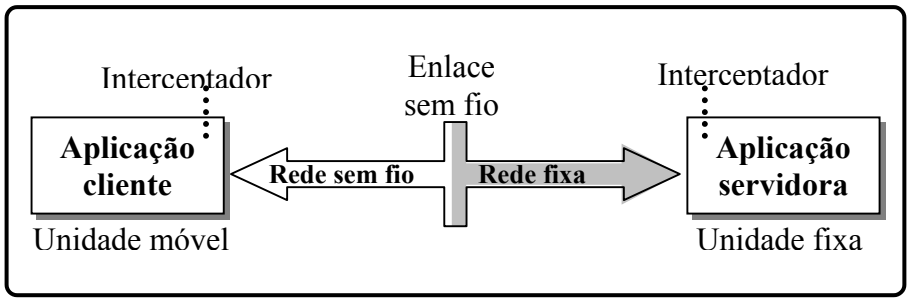

FIGURA 7 – Modelo cliente/interceptador/servidor.

Fonte: Adaptação de ITO (2001).

2.3.2. Modelo ponto-a-ponto

Neste tipo de arquitetura distribuída cada *site* tem dupla funcionalidade, comportando-se tanto como cliente quanto um servidor. RAPPAPORT (1996) verificou que a adaptação do modelo ponto-a-ponto tradicional (aplicado em redes fixas) com a computação móvel têm-se unidades móveis e unidades fixas emulando semelhantes configurações. O modelo compreende estruturas onde existe uma intensa ligação entre os dispositivos móveis e pouca ou nenhuma ligação com a unidade fixa. A inexistência de um canal de comunicação entre os seus componentes acarreta um efeito negativo sobre a ocorrência de desconexões.

2.3.3. Modelo de agentes móveis

Esse modelo é uma extensão do mecanismo de comunicação baseado em RPC, que possibilita o envio de requisições entre clientes e servidores, porém, acrescenta os agentes móveis, que são formados por processos que têm por finalidade a realização de tarefas específicas. Os agentes, descritos por PITOURA, SAMARAS (1998), têm a capacidade de executar funções remotamente, sendo constituídos por instruções, dados e um estado próprio. Além disso, possuem alguns aspectos que tornam viáveis a execução de transações em ambientes móveis, tais como:

- manipulação ativa de dados conforme a localização;
- habilidade de interagir com outros agentes na execução de operações;
- interoperabilidade com relação aos diferentes protocolos de comunicação;
- capacidade responder a eventos externos;
- otimização da comunicação entre os elementos móveis;
- permissão para que unidades móveis possam trabalhar desconectadas às unidades fixas.

Localizado entre a aplicação do cliente e do servidor, o agente interage de forma independente com cada um deles. Isso permite que protocolos diferentes sejam aplicados em cada interação. Além disso, o agente pode assumir um papel ativo em alguns casos, notificando o cliente apropriado sobre a ocorrência de determinado evento, manipulando prioridades na transmissão de dados e mudando a ordem de envio quando existem múltiplas respostas. Outra característica é a possibilidade de uma economia considerável de energia ao cliente através da solicitação de desconexões e re-conexões nos momentos oportunos. Isso ocorre, por exemplo, quando o cliente submete uma requisição ao agente e, automaticamente, entra em modo *doze* (estado de baixa energia) até receber um aviso do agente que a resposta tornou-se disponível. O agente também pode se comunicar com outro agente, um do lado do cliente e outro do lado do servidor, ambos com intuito de permitir uma melhor eficiência de comunicação e oferecer maior flexibilidade na manipulação de desconexões. Além disso, um agente localizado no servidor pode substituir o *host* fixo através do estabelecimento de um protocolo de ligação especial para *hosts* móveis e a personalização da informação sobre sua disponibilidade de sua localização, durante o período de desconexão.

2.4. Dispositivos móveis

Os dispositivos PDAs para usuários finais assumem várias formas e configurações. Alguns computadores portáteis são especialmente projetados para instalação e uso permanente em veículos e outros são fabricados de forma compacta para serem carregados no bolso. No entanto, todos eles têm que ir de encontro às necessidades do usuário e dar suporte as exigências das aplicações móveis.

Diversos tipos e modelos de computadores portáteis podem ser utilizados em um ambiente de computação móvel. A definição do equipamento ideal depende de fatores como: capacidade de memória, velocidade de processamento, formas de comunicação sem fio (*IrDA*, *Bluetooth, Wi-Fi*) e a portabilidade. Dentre os dispositivos móveis de uso geral destacam-se os *laptops, handhelds, palmtops* e telefones celulares. Algumas considerações sobre cada um deles estão descritas a seguir:

- ***Laptop***: pode ser aplicado com sucesso em sistemas móveis que requerem alta capacidade de processamento e armazenamento. Em alguns casos poderá assumir o papel de servidor. Apesar se ser um computador com características e funções similares a um *desktop*, tem limitações ergonômicas que o caracteriza como computador de média portabilidade.

- ***Handheld***: caracteriza-se por ser pequeno, leve, com baixa velocidade de processamento e armazenamento de dados. Um *handheld* inclui um pequeno teclado similar a um *laptop* e suporta sistemas operacionais resumidos que são embutidos na memória ROM na concepção do *hardware*.

- ***Palmtop***: tem características similares a um *handheld*, porém diferencia-se por receber entrada de dados através de botões e de uma caneta chamada *stylus pen*. Atualmente têm sido incorporadas novas funcionalidades, como a disponibilidade de teclado e conexão com a telefonia móvel. Já os *handhelds* têm diminuído de tamanho, fazendo com que a distinção entre eles diminua, assim como a tênue fronteira entre estes dispositivos e os telefones celulares de terceira geração.

- **Telefone celular**: além da tradicional comunicação por telefonia móvel, o telefone celular atual permite a navegação na *Internet* através de um *microbrowser*, troca de *e-mails*, acesso a serviços bancários, efetuação de pagamentos, acesso a informações sobre previsão do tempo, índices financeiros, dentre outras funcionalidades pessoais. A perspectiva é que a próxima geração (3G) inclua funcionalidades similares aos *palmtops*, porém adaptados a dimensões reduzidas.

2.4.1. Evolução

Em 1984, a empresa de tecnologia britânica Psion® ao lançar o Psion Organizer I®, que possuía calculadora, relógio, calendário e capacidade de programação em uma linguagem similar a BASIC, inaugurou o gênero PDA. Paralelamente, também em 1984, a Epson lançou o modelo HX-20, o primeiro *laptop*. O sucesso da Psion, logo

chamou atenção de grandes fabricantes como Apple, Hewlett-Packard, Motorola, Sharp Electronics e Sony Electronics, que também acabaram por lançar seus dispositivos portáteis, tornando-os populares. Mas foi em 1996 que a Palm Inc, provocou um verdadeiro impacto neste mercado ao lançar o Pilot 1000® e Pilot 5000®, que introduziram o conceito de sincronização de dados com um *desktop* remoto. Ou seja, as mesmas informações de um *desktop* poderiam ser acessadas pelo computador portátil localmente ou por uma rede WAN (*Wide-Area Network*) ao toque de um botão.

Desde o lançamento dos primeiros Pilots inúmeras funcionalidades têm sido adicionadas aos PDAs modernos. Sem interferir na portabilidade, avanços na tecnologia vêm permitindo melhor visualização, maior autonomia de bateria, maior capacidade de processamento e uma compatibilidade cada vez maior com dados multimídia.

Uma consideração importante é que portabilidade e mobilidade são conceitos diferentes. Um dispositivo pode ser portátil e não possuir capacidade de computação móvel. Por exemplo, o uso de um *laptop* em um carro realizando tarefas comuns (como processamento de texto e planilha eletrônica), mas sem capacidade de se conectar a uma rede sem fio. Analogamente, em um escritório de uma empresa que funcione em um prédio sem fiação, a LAN (*Local Area Network*) será uma rede *wireless*, mas não será um caso de computação móvel.

2.5. Comunicação *wireless*

A comunicação sem fio introduz diversos requisitos de projeto de protocolos, como o uso de técnicas de criptografia na comunicação devido a confidencialidade, a utilização de mecanismos de compressão de dados devido à baixa largura de banda e o emprego de métodos para detecção de colisão deve ser substituído por técnicas para evitá-las a todo custo.

A mobilidade do computador/usuário de um local para outro pode ser modelado como uma mudança no nodo da rede onde ocorre o acesso a infra-estrutura

cliente/servidor. Nesse caso, a mobilidade pode ser tratada naturalmente como uma mudança no roteamento de datagramas destinados ao computador móvel de tal forma que os pacotes cheguem ao ponto de acesso à rede.

Uma conexão típica *end-to-end* para computação móvel consiste de partes sem fio (*wireless*) e de uma parte com fio (*wired*). Uma estação base provedora de serviços de rede pode estar conectada através de uma linha física a uma estação central, que por sua vez está conectado ao servidor de comunicação do cliente (SNOEREN & BALAKRISHNAN, 2000). O servidor de comunicação do cliente fica conectado via *back-end* a uma aplicação e ao servidor de banco de dados onde as informações ou caixas de correio residem. O dispositivo móvel tem que estar equipado com modem de rede ou celular que tenha capacidade suficiente para receber e transmitir sinais.

Redes sem fio são mais susceptíveis a erros de transmissão e perdas temporárias de conexões do que redes terrestres. Isso ocorre devido, em grande parte, ao alcance de células de cobertura que provêem comunicação entre as unidades móveis e as condições atmosféricas. Estas suscetibilidades podem ter um impacto significativo nas camadas de comunicação de *software* como TCP/IP que foram projetados para WAN's mais seguros.

2.5.1. Arquitetura

A arquitetura de redes sem fio, assim como de redes convencionais, é formada por níveis, *interfaces* e protocolos (MANIATIS, ROUSSOPOULOS, & SWIERK, 1999). Cada nível oferece um conjunto de serviços ao nível superior usando funções realizadas no próprio nível e serviços disponíveis nos níveis inferiores.

Um nível pode ser um programa ou um processo implementado por *hardware* ou *software*, que se comunica com processos correspondentes em outra máquina. As regras que governam a conversão de um nível qualquer são chamadas de protocolos de níveis. Os limites entre cada nível adjacente são as *interfaces*. Na computação móvel a arquitetura lógica é vista sob a perspectiva de dois coeficientes, como segue:

- **Nível de fluxo de dados da aplicação**: os níveis de aplicação são implementados tanto no cliente quanto no servidor. O *software* de aplicação pode acessar serviços de transporte de infra-estrutura do nível anterior através do *midleware*, especialmente se é necessária a comunicação entre plataformas diferentes. Normalmente um *midleware* é concebido com o protocolo TCP/IP móvel. Os pacotes são transmitidos para as camadas de acesso ao meio através de redes sem fio, usando camadas de *drivers* de *softwares* apropriados.

- **Nível de controle de sistema**: deve estabelecer condições para conexão e transferência de dados através do uso de protocolos confiáveis e seguros.

2.5.2. Redes locais

As redes locais têm se tornado um padrão de conectividade física entre *desktops*, *workstations* e servidores. Na maioria dos casos o uso de cabos (fibra ótica, cobre) é uma necessidade inevitável para realizar esta conectividade física. Entretanto, alguns tipos de usuários, como profissionais executivos que precisam tomar decisões longe do ambiente do escritório, exigem mobilidade. Neste contexto redes locais sem fio encontram um importante foco de aplicação.

Existem três cenários de conectividade onde redes locais sem fio são usadas comumente:

a) Redes sem fio verdadeiras

Para mudanças e movimentos constantes, provê conexões rápidas locais. Este tipo de rede é muito usado em ambientes muito dinâmicos, como no interior de edifícios comerciais.

b) Redes locais sem fios flexíveis

Para situações em que o usuário precisa sair fora do ambiente de um edifício. A mobilidade de acesso pode ser alcançada através do uso de conexões sem fio para *hosts* ou servidores.

c) Conexões LAN-LAN

As conexões LAN-LAN envolvem conexões entre universidades, filiais de lojas, etc. Implica na comunicação entre edifícios distanciados por alguns quilômetros e conexões entre estabelecimentos localizados em zonas rurais. Mesmo existindo a possibilidade de ser aplicada conexão por cabo a alternativa sem fio pode se tornar mais conveniente, principalmente quando se quer rapidez na instalação da rede e baixos custos.

2.5.3. Componentes

Os componentes essenciais de redes locais sem fio são similares àqueles utilizados em redes convencionais. As principais diferenças são a substituição de cartões de *interface* de rede *Ethernet* e *Token Ring* pelos seus similares nas LAN's sem fio e a ausência de conectores e fios. Dentre os recursos de rede sem fio, temos:

- Cartões de *interface* de rede (NIC - *Network Interface Cards*). Exemplo: cartões PCMCIA para *notebooks*;

- Antenas para captar e difundir sinais de rádio;

- Pontos de acesso, módulos de controle ou *hubs* controladores de cartões NIC.

3. REPLICAÇÃO DE DADOS

Replicação de banco de dados é o processo de compartilhamento de dados entre bancos de dados com localizações diferentes (ÖZSU & VALDURIEZ, 1999). Através do emprego de mecanismos de replicação são geradas cópias especiais, chamadas de réplicas, com o propósito de aumentar a disponibilidade e a velocidade de acesso a dados aos usuários remotos. Esses dados, após serem submetidos a mudanças, são reintegrados ao servidor por meio de primitivas de sincronização.

Uma característica chave de sistemas de banco de dados móveis é a habilidade para lidar com conexões raras, esporádicas e fracas. Isso exige a aplicação de mecanismos para aumentar a disponibilidade do banco de dados aos usuários. Apesar dos aspectos inerentes, como o grau de divisão, a localização e fatores de integridade das réplicas de itens de dados, LUBINSKI e HEUER (2000) destacaram que a utilização de replicação tem sido extremamente comum. Cópias de dados são mantidas no equipamento móvel para evitar o estabelecimento de uma conexão com a base fixa a cada consulta/atualização, aumentando a disponibilidade desses dados durante um período de desconexão.

3.1. Concepção e tratamento de réplicas

A concepção e o tratamento de dados replicados segue três fases distintas (preparação para a desconexão, operações em modo de desconexão e re-conexão), descritas resumidamente na Tabela 1. Durante um período em que uma unidade móvel esteja conectada com a base fixa é preciso que ocorra a preparação para a desconexão. Neste ponto são determinados quais dados serão armazenados na memória do dispositivo portátil, conforme as diretrizes estipuladas no plano de replicação. De acordo com o plano, podem ser adotadas as seguintes medidas:

- A definição dos dados que serão copiados para a unidade móvel fica a critério do usuário. Nesse contexto é preciso que haja uma aplicação que atue como um assistente de sincronização, associada ao banco de dados consolidado;

- A medida em que um usuário executa operações na unidade móvel, o sistema monitora suas ações e armazena os dados antecipadamente;

- Os dados são encaminhados à unidade móvel conforme o tipo de usuário e as permissões de acesso.

TABELA 1 – Planos para geração e tratamento de réplicas de dados.

	Desafio	**Plano**
Preparação para a desconexão	O que armazenar na memória	- Especificado pelo usuário. - Baseado no histórico das operações. - Definido pela aplicação.
	Quando armazenar na memória	- Antes da desconexão. - Em um período definido.
Operações em modo de desconexão	Requisição de dados não-armazenados	- Criação de exceções. - Fila para futuros serviços.
	O que guardar no *log*	- Dados. - Operações. - *Timestamps*.
	Como usar o *log*	- De forma incremental. - Associado ao banco de dados .
Re-conexão	Como integrar registros	- Submeter todos os dados. - Re-executar o *log*.
	Como resolver conflitos	- Uso de semânticas de aplicações. - Soluções automáticas. - Uso de especificações.

Além da definição de quais dados deverão ser submetidos à unidade móvel, também é necessário que seja estipulado o momento mais apropriado para a transferência desses dados. Devem ser levados em consideração o desempenho do sistema, a manutenção da consistência do banco de dados e a capacidade dos recursos de *hardware* disponíveis. Com base nesses critérios, podem ser adotados dois planos distintos: o armazenamento é feito antes da conexão (cópias parciais ou completas), na fase de sincronização; ou o armazenamento é feito conforme a demanda de requisições, com intervalos de conexão pré-definidos.

As operações realizadas em modo de desconexão podem requisitar dados que não estejam na memória local do dispositivo móvel. Para solucionar o problema, pode-se bloquear os processos conflitantes ou gerar uma fila para execução posterior. A escolha do plano mais adequado depende do grau de abstração definido aos usuários e da freqüência de conexão das unidades móveis com a unidade fixa.

Quanto às operações efetivadas localmente, é conveniente que sejam resguardadas para uma posterior reconciliação com o banco de dados consolidado. As ações ocorridas no cliente podem ser conservadas através de um registro de *log*, que lista dados, operações ou *timestamps* associados às mudanças ocorridas em modo de desconexão. Tal registro de *log* pode ser concebido e aplicado de forma incremental (na forma de uma lista) ou associado ao banco de dados (com utilização de atributos marcadores - *flags*).

Em um período de re-conexão, os dados replicados são reintegrados à base fixa. Esse procedimento, também chamado de reconciliação, pode envolver todos os registros ou apenas aqueles associados ao *log*. A resolução de conflitos, provenientes de atualizações concorrentes, pode ser baseada no uso de semânticas de aplicações (intervenção da aplicação móvel), em soluções automáticas (todos os dados originários do *host* móvel são considerados válidos) ou no uso de especificações (definição de *timestamps* ou outros modelos para controle de concorrência).

3.2. Distribuição dos dados

Em um sistema de banco de dados os registros armazenados podem ser divididos em um ou mais bancos de dados. Em alguns casos, uma parte ou a totalidade dos dados podem ser mantidos por diferentes computadores, em diferentes localizações geográficas. Para HELAL, HEDDAYA, BHARGAVA (1996) esta fragmentação torna transparente a localização dos dados de tal forma que o usuário, ao acessar o banco de dados, desconhece onde e como a transação solicitada está atuando.

A replicação de dados é um método muito aplicado em sistemas de banco de dados distribuídos, onde um banco de dados é replicado em diferentes nodos, de maneira que cada cópia é uma instância da base de dados servidora. Em alguns casos aplica-se a fragmentação horizontal ou vertical da base de dados, contendo porções inteiras de registros, que são manipulados em estações remotas. Essa metodologia tem sido adotada para agilizar as consultas realizadas por usuários geograficamente distantes do banco de dados central. Cada procedimento de busca tende a não gerar transações adicionais que resultem no acesso a base de dados servidora. Dessa forma, fragmentos específicos são destinados a determinados terminais conforme o tipo de demanda. Além disso, pode haver cópias completas da base de dados em locais remotos a fim de garantir maior segurança do conteúdo armazenado.

Em ambientes móveis dificilmente é adotada a transmissão completa de dados, como ocorre em bancos de dados distribuídos, devido às restrições impostas pela arquitetura de *hardware*. Conforme descrito por TERRY, PETERSEN, SPEITZER (1998), uma unidade móvel normalmente contém réplicas parciais de itens de dados a fim de suprir a demanda do usuário até a ocorrência da próxima solicitação de sincronização. Tal manutenção de dados replicados é conhecida como *caching*, que tem como objetivo o acesso mais eficiente aos dados, sendo utilizada a fim de aumentar a probabilidade do item procurado estar armazenado localmente na unidade móvel. Faz com que permaneçam em memória os dados que serão possivelmente reutilizados de modo a reduzir a contenção desses dados e o tempo de repostas às consultas, melhorando com isso o desempenho do sistema. Por outro lado, as replicações são tratadas do ponto de vista de itens de dados, que são parcelas resumidas das

informações contidas na base fixa. Uma unidade móvel pode conter localmente réplicas parciais compartilhadas que estão sujeitas a atualizações que devem ser propagadas ao resto do sistema na fase de reintegração. Entretanto, essas cópias de itens de dados potencialmente conflitam com outras mudanças na rede distribuída, sendo necessários esquemas para descobrir e solucionar tais conflitos.

A implementação de um sistema de banco de dados com distribuição de cópias de dados entre unidades remotas implica na utilização de estratégias de replicação. Tais estratégias visam permitir a manipulação de registros de forma autônoma em cada unidade cliente, mantendo a consistência geral do sistema. Conforme uma análise descrita por RATNER (1998), dependendo da característica do projeto poderá ser desejável que transações sejam processadas em unidades isoladas, com emprego de comunicação assíncrona para reintegração de dados, ou o uso de comutatividade entre as estações, de forma síncrona, no intuito de manter constante a atualização de todas as réplicas distribuídas pelo sistema de banco de dados móvel.

3.3. Graus de replicação

A obtenção e a manipulação das cópias podem seguir eixos diferentes, dependendo do contexto em que o banco de dados é definido e dos tipos de solicitações feitas pelos usuários. Também estão atrelados fatores limitantes como a disponibilidade de acesso e a necessidade de armazenamento dos dados. Com base nestas definições, BADRINATH, PHATAK (1998) descreveram os seguintes graus de replicação:

3.3.1. Ausência de replicação

O banco de dados é fragmentado de tal forma que cada parte fica armazenada em apenas um *site*[4]. Após o término das operações sobre os fragmentos ocorre o re-agrupamento das linhas de dados, remontando o banco de dados. Cada estação detém o

[4] Em sistemas distribuídos entende-se como uma unidade computacional (microcomputador, estação de trabalho, minicomputador, entre outros).

controle de uma parte do banco de dados, conforme o tipo e o número de requisições realizadas localmente. Este grau de replicação é aplicado em casos onde não há necessidade da existência de cópias concorrentes no sistema.

3.3.2. Replicação parcial

Na replicação parcial, apenas os fragmentos de dados solicitados pelos usuários são replicados, com isso deverão ser empregadas técnicas de controle de concorrência e a execução de conjunções quando ocorrido o término das transações sobre os itens de dados. A finalidade de tal método é dar autonomia e aumentar o desempenho com a disponibilidade de registros na memória das estações remotas, minimizando as transferências entre as unidades do sistema.

3.3.3. Replicação total

Para que a disponibilidade das informações aumente a medida em que o usuário executa operações sobre os dados, utiliza-se a replicação completa do banco de dados em todos os *sites*. Isso afeta diretamente os mecanismos de controle de concorrência, que precisam gerenciar atualizações realizadas em locais distintos, não permitindo que ocorram inconsistências.

Com a existência de replicações completas e a generalização do conteúdo entre diversos usuários, aumenta confiabilidade na mesma proporção que o número de cópias. No caso da ocorrência de perda de dados em um arquivo, estes poderão ser recuperados, de forma a manter a estabilidade do sistema, suprindo as necessidades do usuário. Quando as operações sobre itens de dados locais são finalizadas no dispositivo móvel é restabelecida a conexão com o servidor e os arquivos são reintegrados.

3.4. Tipos de replicação

3.4.1. Replicação síncrona

Em uma replicação síncrona todas as cópias de dados existentes em um sistema são analisadas no instante da sincronização. As alterações feitas em alguma réplica do banco de dados são imediatamente aplicadas a todas as outras instâncias dentro de uma transação. A replicação síncrona é apropriada em aplicações comerciais onde a consistência exata das informações é de extrema importância.

3.4.2. Replicação assíncrona

Neste tipo de replicação as alterações realizadas em cópias de dados são propagadas em um segundo passo, conforme descrito em PARK, YEOM (2000). Em um procedimento de sincronização são transferidas as modificações de uma réplica para o banco de dados centralizado e os possíveis conflitos gerados por atualizações concorrentes são tratados individualmente. A sincronia entre as réplicas de dados dependerá da ocorrência da comunicação entre as unidades clientes e a unidade servidora.

A replicação assíncrona pode ser dividida em:

- **Replicação em lotes**: A comunicação é estabelecida em intervalos de tempo e é baseada em transações enfileiradas que são agendadas para processamento posterior nos servidores que possuam réplicas dos dados. As políticas de agendamento variam, podendo ocorrer em um determinado intervalo de tempo ou em função da quantidade de alterações acumulada ou, ainda, em instantes específicos de um ciclo de processamento. Uma vez que a comunicação é estabelecida, conjuntos de dados são copiados para o servidor com dados replicados. Os protocolos mais utilizados são FTP e RDIST. Esta forma é muito usada para sincronização de espelhos de *sites Web* na *Internet*.

- **Replicação por demanda**: Os servidores replicados obtêm o conteúdo quando necessário devido à demanda das aplicações. Quando uma aplicação solicita um recurso que não esteja no conjunto de dados do servidor replicado ou que esteja desatualizado é atendido o pedido buscando o recurso no servidor original. Os protocolos mais utilizados são FTP, HTTP e ICP.

3.5. Modelos de replicação

Alguns modelos que abordam replicação foram propostos ao longo dos anos. Um importante parâmetro é fazer com que todas as suas réplicas fiquem disponíveis para uso pleno das unidades móveis, dando suporte inclusive para atualizações e acesso a dados compartilhados. Para RATNER, REIHER, POPEK (2001), os modelos de replicação estão dispostos como segue:

3.5.1. Replicação ponto-a-ponto

O modelo de replicação ponto-a-ponto permite que réplicas de itens de dados sejam atualizadas a partir de qualquer outra cópia, sem serem necessariamente advindas de um banco de dados central. Normalmente as réplicas envolvidas na sincronização são constituídas por estruturas idênticas, que ficam localizadas em *sites* que se comunicam entre si (Fig. 8). Tal modelo é aplicado principalmente quando existe alta conectividade entre dispositivos portáteis, o que cria uma limitação considerável do número de elementos replicados.

Para REIHER, POPEK, GUNTER (1996) o custo da replicação ponto-a-ponto compreende principalmente a necessidade de recursos, pois envolve uma considerável complexidade nos algoritmos de controle de sincronização que necessitam estarem localizados na *cache* dos dispositivos móveis. Cada unidade portátil deve ser capaz de comunicar-se com outros elementos e executar transferências de dados e operações de controle de concorrência.

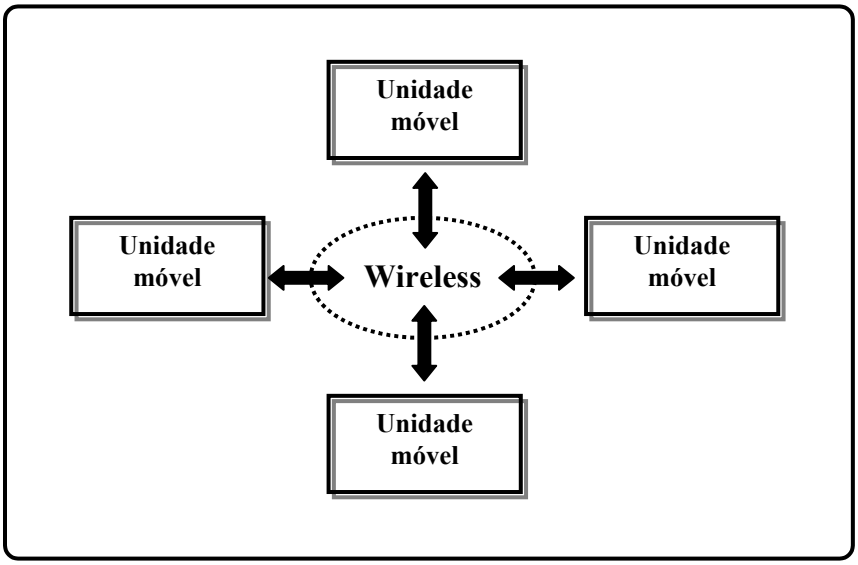

FIGURA 8 – Modelo de replicação ponto-a-ponto.

3.5.2. Replicação cliente-servidor

A computação móvel agrega-se ao modelo distribuído baseado em uma estrutura conhecida como modelo Cliente/Servidor. Segundo SATYANARAYANAN (1996), são chamadas aplicações clientes àqueles sistemas computacionais que requisitam serviços para uma aplicação que está sendo executada em um outro sistema computacional, nomeada aplicação servidora.

De modo geral, um dispositivo móvel tem o papel do cliente e a unidade fixa tem o papel do servidor. Em alguns casos, operações e dados são distribuídos entre diversos servidores fixos que podem ter que se comunicar entre si para atender as requisições de um cliente, conforme representado na Fig. 9. Dessa forma, são atribuídas funcionalidades distintas aos servidores de um mesmo sistema, uns dando suporte a informações e serviços e outros sendo utilizados para a replicação de dados, o que permite maior disponibilidade e escalabilidade do banco de dados. Em casos especiais, como o modo desconectado (momento em que a unidade móvel não está comunicando-se com a rede fixa) o dispositivo móvel emula funcionalidades de um servidor para poder continuar operando.

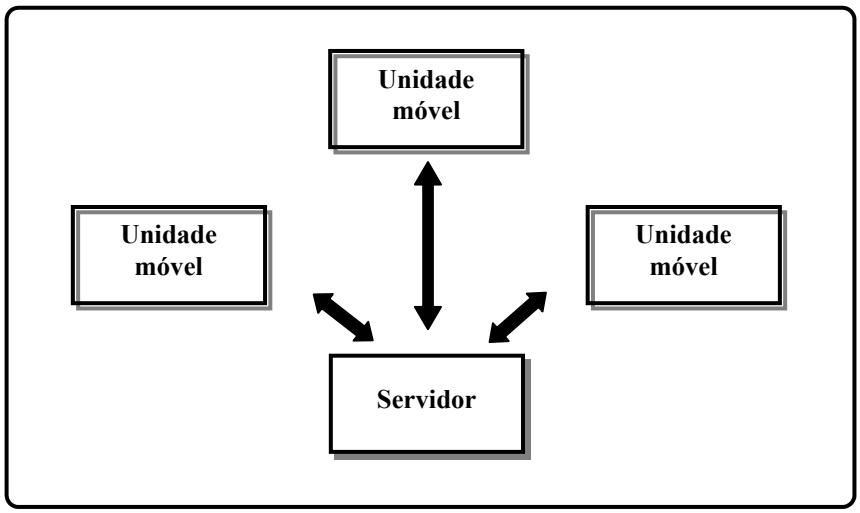

FIGURA 9 – Modelo de replicação cliente/servidor.

O modelo Cliente/servidor possibilita que as atualizações de uma réplica de item de dado sejam somente transmitidas a uma ou mais réplicas especialmente designadas no servidor. O servidor é tomado como ponto de referência para atualização dos dados, não sendo permitida a atuação entre os clientes. Os registros completos ficam armazenados em servidores, que gerenciam replicações, obrigando os clientes a utilizarem apenas os dados mais relevantes para determinada consulta. As atualizações são transmitidas de um servidor para todos os clientes.

Os computadores portáteis que contém réplicas de itens de dados não podem apenas transmitir as atualizações diretamente ao banco de dados fixo. Ao invés disso, cada cliente tem que conectar a um servidor, submeter às próprias atualizações e aguardar a confirmação resultante das operações de controle concorrência.

A simplicidade operacional só é alcançada quando há um único elemento servidor. Porém, sistemas centralizados acarretam em menor confiabilidade dos dados à medida que se aumenta a quantidade de dispositivos, que dificulta a disseminação das atualizações. Já os sistemas cliente/servidor que suportam múltiplas réplicas de servidor para obter um índice mais elevado de confiabilidade e desempenho, necessitam embutir algoritmos do tipo ponto-a-ponto.

3.5.3. Replicação WARD

A replicação WARD (*Wide Area Replication Domain*), conhecida como híbrida, é a mescla entre os modelos anteriores. Permite atualizações dinâmicas sobre banco de dados replicados através de topologias de sincronização. Como visto em RATNER, REIHER, POPEK (2001), tal plano consiste em um grupo de clientes móveis próximos geograficamente (Fig. 10). Cada uma das unidades clientes pode se comunicar entre si e com o servidor, embora tal interação não seja intermitente. Alguns componentes são designados como estações-mestre, aos quais é atribuída a permissão para realização de sincronizações diretas sobre todos os demais membros.

Segundo COGLIANESE (2000), um cliente pode, potencialmente, fazer parte de múltiplos WARDs que são construídos de forma hierárquica. Cada WARD possui um elemento-mestre que se conecta com um sub-nível de forma a conhecer a localização das réplicas de dados dos elementos de níveis inferiores. O gerenciamento de consistência nesse modelo é realizado, essencialmente, através da comunicação *one-way* entre os clientes.

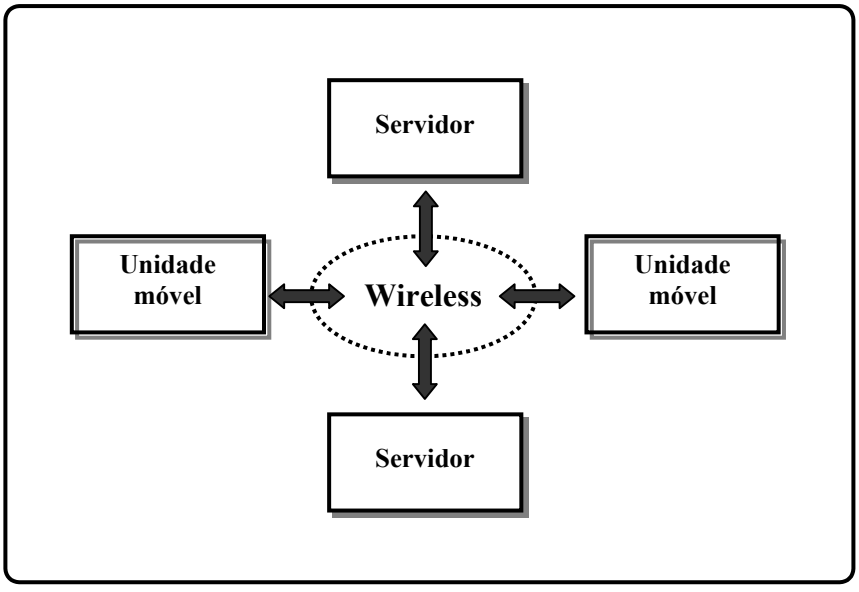

FIGURA 10 – Modelo de replicação WARD.

Para se chegar a uma distribuição eficiente do banco de dados em *sites,* devem ser analisadas diferentes estratégias de replicação de dados. Conforme PALAZZO, PULIOFITO, SCARPA (2000), duas linhas podem ser adotadas: o desenvolvimento de uma rede de *Petri* para a análise das conseqüências derivadas de nodos em diferentes níveis de um banco de dados distribuído e a investigação do desempenho oferecido pelo sistema em um modelo de rede encadeada. Com isso, alguns nodos podem ser considerados como críticos para a confiabilidade do sistema, permitindo a estruturação correta com relação ao número de nodos e a estratégia de replicação a ser adotada.

3.5.4. Replicação Eager e Lazy

Em bancos de dados, mecanismos de controle de réplicas asseguram a consistência dos dados entre as cópias. Para KEMME, ALONSO (2000) esses mecanismos podem ser caracterizados de acordo com o momento no qual são propagadas atualizações e quais cópias podem ser atualizadas (Tabela 2). Propagações de atualização podem ser feitas dentro ou fora dos limites de uma transação. No primeiro caso, a replicação é *Eager*, caso contrário pode ser considerada *Lazy*.

TABELA 2 – Aplicações dos mecanismos *Eager* e *Lazy*.
Fonte: (KEMME & ALONSO, 2000)

Onde / **Quando**	**Eager**	**Lazy**
CÓPIA PRIMÁRIA	Soluções no Ingres	Sybase/IBM/Oracle
Atualizações distribuídas	ROWA/ROWAA Quorum based Oracle Synchr.Repl.	Oracle Advanced Repl. Weak Consistency Strat.

A replicação *Eager* permite a detecção de conflitos antes que a transação seja efetivada no banco de dados (GRAY et al, 1996). Este método provê consistência de dados de um modo direto, mas pode resultar em *overhead* na comunicação aumentando significativamente o tempo de resposta.

Para manter um tempo de transmissão de dados relativamente curto, a replicação Lazy prorroga a propagação de mudanças para após o término da transação, implementando essa disseminação como um procedimento *in background*. Desde que sejam permitidas cópias divergentes neste tipo de replicação, inconsistências podem acontecer.

Em termos de atualizações de cópias, existem duas possibilidades de gerenciamento: centralizar as atualizações através de uma cópia primária ou aplicar um plano distribuído. Na primeira alternativa, as transações ocorridas nos clientes são encaminhadas para o servidor, que possui a versão referencial do banco de dados. Todas as atualizações são aplicadas primeiramente na cópia principal e, somente após isso acontecer, são propagadas para as demais réplicas existentes. Já na segunda possibilidade é permitida a existência de múltiplas cópias de itens de dados coerentes em um determinado instante de tempo. As atualizações podem ser propagadas a partir da unidade cliente que realizou alterações em sua réplica local, que detém momentaneamente a cópia válida de parte banco de dados, disseminando determinadas mudanças para as demais cópias distribuídas em um ambiente móvel.

3.6. Publicações de cópias

Para um entendimento mais claro, pode ser feita uma analogia a metáfora editor/publicação/artigo/assinante. Há a figura do editor, que é a pessoa responsável por gerar a publicação (nesse caso, uma revista). Toda publicação é composta de um ou mais artigos (que são as informações a serem entregues). Para receber uma revista é necessário assinar a publicação, tornando-se assinante. Analisando a metáfora no âmbito de banco de dados, o editor seria o banco de dados que disponibilizaria tais dados para os demais bancos de dados (assinantes). Uma publicação seria um conjunto de dados replicados para um determinado *site*, composta de diversos artigos, ou seja, tabelas (ou parte delas) a serem replicadas. Um assinante, visto como um segundo banco de dados, então receberia as informações de uma publicação. Além disso, torna possível que um

banco de dados seja assinante de uma publicação em uma outra base de dados que seja assinante de alguma publicação no primeiro. Desta forma, a replicação dos dados se faz de maneira bi-direcional.

3.7. Protocolos de replicação

Na computação móvel podem existir momentos em que a unidade cliente esteja momentaneamente desconectada da parte fixa (denominada como servidor). Segundo PITOURA, SAMARAS (1998), isso pode ser ocasionado por problemas de comunicação, no intuito de liberar canais de comunicação ou ainda, promover uma economia de energia na unidade móvel. Para tanto, sistemas móveis dão suporte a desconexão através de protocolos de replicação que permitem ao cliente móvel manipular itens de dados de forma autônoma. As réplicas de itens de dados decorrentes das operações de consultas realizadas pelo cliente no servidor ficam na memória principal do dispositivo móvel até a solicitação de sincronização. Essa chamada é bilateral, podendo ser realizada no sentido servidor/cliente, caso ocorra alterações na base consolidada, ou no sentido cliente/servidor, caso o usuário acione explicitamente o comando de sincronização.

3.7.1. Replicação otimista

A abordagem otimista envolve a capacidade de acesso paralelo e desconectado para mais de uma cópia sem necessidade da fase de reintegração depois do acesso isolado. Pode ser comparada com certos mecanismos de coerência de *cache* utilizados por alguns sistemas de arquivos, como em *Coda File System* (KHUSHRAJ, HELAL & ZHANG, 2002). Cabe, então, as unidades clientes atuarem sobre as cópias de dados locais e após a finalização completa de um determinado trabalho reintegrarem as modificações com a base fixa. Isso amplia a capacidade de mobilidade do dispositivo móvel, já que não existe a necessidade da comunicação constante com o banco de dados fixo. Além disso, serve como um artifício para se obter um melhor gerenciamento do consumo de energia dos dispositivos móveis.

Algoritmos de replicação otimistas permitem que dados apresentados aos usuários possam ficar desatualizados, mas de um modo controlado. Uma característica chave que separa algoritmos de replicação otimista da contraparte pessimista são as atualizações. Com algoritmos pessimistas, atualizam-se todas as réplicas de um bloco lido durante uma solicitação de usuário. No caso otimista são propagadas atualizações em *background* permitindo leituras de quaisquer réplicas diretamente na memória local do dispositivo, sem haver dependência de comunicação. Esta característica faz com que algoritmos otimistas sejam mais disponíveis e eficientes, além de serem mais indicados quando se dispõe de recursos limitados de *hardware*.

No esquema otimista cada unidade móvel mantém réplicas de itens de dados que podem sofrer operações de leitura e escrita conforme as requisições do usuário. Essa abordagem tem sido utilizada para gerenciar a consistência de dados sobre grupos de servidores interligados, porém a conectividade dos dispositivos em ambientes móveis pode variar freqüentemente. Entretanto, o modelo de replicação otimista incorre em certos custos de atualização de dados concorrentes. Para AHUJA, et al (1999), em muitos casos, esses custos podem ser tolerados por três razões. Primeira, em um pequeno período de tempo, a maioria dos arquivos tende a sofrer raras escritas, conseqüentemente, são geradas atualizações concorrentes esporádicas sobre réplicas de dados. Segunda, muitas aplicações trabalham bem com informações desatualizadas, assim, a propagação imediata de modificações para todas as réplicas não é freqüentemente vital. Terceira, a maioria de acessos a dados concorrentes podem ser executados em paralelo por muitas aplicações, portanto, com a manipulação correta, as modificações podem ser combinadas automaticamente ou manualmente, sem perda de dados.

3.7.1.1. Taxonomia de algoritmos otimistas

Examinando alguns sistemas que utilizam o mecanismo de replicação otimista de dados, COGLIANESE (2000) observou que decisões quanto ao projeto de sistemas devem incluir as seguintes áreas:

- **Sistema de arquitetura e comunicação**: deve ser levado em consideração o grau de comunicação permitido entre diferentes elementos na rede. Pode ser adotada a arquitetura cliente/servidor, onde clientes móveis conectam-se somente com uma aplicação servidora, ou usar o modelo ponto-a-ponto, onde clientes móveis são capazes de se intercomunicar.

- **Gerenciamento de *cache***: outra característica importante a ser observada é a maneira como cada dispositivo representa e administra a memória local. Em particular, deve-se determinar qual informação adicional precisa ser armazenada para que dados possam ser reintegrados à base de dados consolidada, mantendo a consistência do sistema.

- **Propagação de atualizações e reconciliação**: uma importante decisão é como as operações de escrita na memória local são propagadas para as demais máquinas do sistema. De forma inerente, deve ser levado em consideração como as atualizações conflitantes são reconciliadas na base fixa. Alguns sistemas podem optar por atualizações comutativas, que são aplicadas em ambos os elementos interligados.

- **Ambiente de aplicação**: em alguns casos é permitido que aplicações móveis façam parte do gerenciamento das cópias em suas próprias *caches* aplicando políticas de atualização de dados. No entanto, é preciso que seja feita uma análise de como será feita a integração dos dados no sistema sem a necessidade da intervenção do usuário e se nenhuma aplicação esteja adotando outra política conflitante.

A busca pela detecção e resolução de conflitos com o mecanismo otimista, descrita em KUENNING et al (1998) e SAITO, LEVY (2001), pode ser aplicada não apenas no momento da reconciliação dos dados, mas também durante procedimentos isolados na unidade móvel. Com isso é dada uma maior autonomia na manipulação do banco de dados à aplicação móvel, incorrendo em ações irreversíveis sobre registros.

Em contrapartida, aumenta a complexidade dos algoritmos, exigindo maior desempenho do dispositivo portátil.

3.7.2. Replicação pessimista

Um assunto importante em replicação é modo com que são apresentadas as cópias de itens de dados aos usuários. Algoritmos de replicação pessimistas tradicionais oferecem uma semântica onde os usuários têm a ilusão de existir apenas uma cópia de um objeto, altamente disponível, mantendo réplicas idênticas todo o tempo.

Para SAITO (2001) a expressão "pessimista" é porque proíbe acessos a uma réplica a menos que o conteúdo seja atualizado. Embora algoritmos deste tipo sejam essenciais em algumas classes de aplicações, como por exemplo, para transações bancárias, têm a desvantagem de exigir determinadas funcionalidades de *hardware*, considerando a necessidade de freqüentes conexões com outros elementos do sistema.

O método pessimista garante a consistência dos dados pela propagação de atualizações. Nessa abordagem, as transações móveis são completadas após a reconciliação dos dados na base consolidada. A técnica ROWA - *Read-One, Write-All* - adota o protocolo pessimista, assegura a integridade do banco de dados, mas acarreta em uma intensa comunicação na rede, como descrito em LUBINSKI, HEUER (2000).

3.7.3. Análise comparativa entre as propostas otimista e pessimista

Em sistemas de computação distribuída a replicação de dados em múltiplos *sites* acarreta em um aumento de desempenho e disponibilidade. Porém, dependendo do tipo de usuário e de sua localização geográfica o emprego de determinados mecanismos pode ser fundamental para se obter êxito em transações sobre bancos de dados. No âmbito das propostas otimista e pessimista algumas considerações foram descritas baseadas nos aspectos fundamentais de sistemas de gerenciamento de dados aplicados à computação móvel, como segue:

a) Tolerância à falhas: algoritmos otimistas oferecem resultados satisfatórios em uma rede com conexão lenta ou incerta porque podem propagar atualizações entre réplicas sem bloquear acessos para qualquer outra cópia existente no ambiente móvel. Esta propriedade é essencial em um ambiente móvel no qual nodos comunicam-se entre si ocasionalmente. Em contraste com a contra-parte pessimista, os acessos às réplicas são impedidos até ser armazenado o conteúdo mais atual.

b) Transmissão de dados: algoritmos otimistas melhoram a autonomia requerendo menos coordenação entre *sites*. Algumas operações podem ser executadas completamente sobre uma réplica sem qualquer obrigatoriedade de mudanças em outros *sites* e também da aplicação de sincronizações adicionais, como normalmente ocorre em sistemas com ênfase em replicação pessimista.

c) Disponibilidade de informações: dependendo da demanda de requisições de usuários à base de dados fixa, poderá haver momentos onde existam mais transações pendentes que o sistema gerenciador do banco de dados possa suportar. Isso acarreta em perda de desempenho e aumento da latência das transações para as unidades clientes. Na proposta pessimista a situação descrita anteriormente é comum, apesar de manter as réplicas atualizadas constantemente. Já no modelo otimista as consultas restringem-se a memória local, permitindo que respostas às consultas de usuário sejam mais rápidas, porém com menor grau de consistência.

d) Mobilidade: como no modelo otimista as operações sobre o banco de dados móvel são realizadas localmente, não é preciso que o usuário fique dentro do limite estabelecido pela zona de cobertura (célula) de uma rede *wireless*. Para o modelo pessimista essa prática inviabilizaria uma operação sobre dados, uma vez que depende da existência de comunicação com a base fixa para efetivar mudanças ou submeter dados resultantes de operações de consulta.

e) Coerência do banco de dados: no modelo otimista a existência de alterações conflitantes sobre itens de dados replicados em diversos *sites* força a invalidação de operações de usuários. Além disso, procedimentos de consulta podem retornar dados desatualizados, uma vez que as modificações são efetivadas apenas no momento da

sincronização, determinado de forma explícita pelo usuário. Já no modelo pessimista um mecanismo de controle de concorrência, localizado em um servidor, consegue garantir a integridade do banco de dados e impedir ações inválidas antecipadamente.

e) Necessidade de recursos: na proposta pessimista todas as atualizações recaem diretamente no banco de dados consolidado através do envio e recebimento de mensagens. Essa associação entre cliente/servidor ocorre por meio de uma rede *wireless*. A incidência de requisições ao servidor torna-o um "gargalo" no sistema, que deve suportar uma grande quantidade de requisições paralelas de usuários. Além disso, a unidade cliente deve incluir suporte a comunicação via *wireless* e quantidade de energia suficiente para manter a transferência de dados até o fim de uma transação. A proposta otimista dá ênfase na alta disponibilidade de informações mantendo parte considerável de itens de dados na memória da unidade cliente, conforme o perfil do usuário. Com esse método espera-se que requisições sejam supridas por dados locais, diminuindo assim a necessidade de recursos adicionais de suporte a comunicação.

De acordo com as conclusões de PITOURA, SAMARAS (1998), uma combinação dos métodos otimista e pessimista pode ser implementada. Tal associação pode ser classificada como híbrida. Cópias de dados centrais ou aparentes são geradas, de forma a suprir as necessidades do usuário móvel. As cópias centrais são visíveis por todo o sistema, favorecendo a resolução de conflitos na própria unidade-cliente. Já as cópias aparentes, são visíveis apenas por um grupo selecionado de elementos na rede, e aplicadas em sistemas onde o encargo do gerenciamento de réplicas se dá apenas no servidor fixo.

3.8. Propagação de réplicas

Observa-se que o uso da replicação de dados é muito útil por vários motivos, no entanto, deve ser levado em consideração que usuários poderiam estar geograficamente distantes da unidade servidora e, ao reconectarem-se, poderiam dirigir um alto volume de requisições em um único servidor. Para tratar situações deste tipo são especificados

métodos de propagação de réplicas, que podem envolver a propagação síncrona ou assíncrona de réplicas de banco de dados.

As soluções síncronas tradicionais normalmente são empregadas para administrar um número pequeno de réplicas e não são satisfatórias para ambientes amplamente distribuídos, onde a conexão entre as réplicas pode ser incerta. Como a necessidade de replicação de dados tende a crescer com o incremento de novas unidades em um sistema, KELEHER (1999) observou que soluções assíncronas podem ser adotadas para administrar a propagação de dados replicados. A aplicação Lotus Notes, por exemplo, usa replicação baseada em valor (*value-based*) na qual são executadas atualizações localmente e um mecanismo de propagação é provido para aplicar estas atualizações para outras réplicas. Além disso, um número de versão é usado para descobrir e solucionar possíveis inconsistências no banco de dados.

Embora os mecanismos aplicados no Lotus trabalhem razoavelmente bem na atualização de um único objeto (como em ambientes de sistemas de arquivos), falha quando objetos múltiplos são envolvidos em uma única atualização (ambientes orientados a transação). Em particular, é preciso mecanismos mais formais para propagação de atualizações e descoberta de conflitos com replicação assíncrona de bancos de dados.

3.8.1. Modelo epidêmico

No modelo epidêmico as atualizações de itens de dados podem ser executadas localmente no cliente. Após isso, as unidades móveis comunicam-se para trocar informações de atualização. Os usuários executam operações locais sem esperar por comunicação e as modificações são propagadas entre diversos elementos do sistema assincronamente, no momento mais conveniente. O modelo epidêmico é adotado por diversas companhias, como Oracle, Sybase, Informix, dentre outras, e discutido em diversas pesquisas, como em AGRAWAL, ABBADI, STEINKE (1997), STEINKE (1997), ANDERSON et al (1998), BREITBART et al (1999) e HOLLIDAY et al (2000).

Um algoritmo de propagação epidêmica originalmente fornece um baixo nível de consistência do banco de dados preservando a ordem causal de operações de atualização. Segundo HOLLIDAY et al (2003), para algumas aplicações isto é suficiente. Opcionalmente, para determinados casos onde se estima que conflitos serão raros, pode ser usada uma aplicação de compensação específica. Fundamentalmente, a meta é assegurar que todas as réplicas de um único item de dados convirjam em um único valor final. Para o processamento de transações isto é insuficiente porque transações podem criar dependências entre os valores de itens de dados diferentes.

Em um contexto de banco de dados, a execução de um conjunto de transações deve ser equivalente a uma ordem total. Vejamos um exemplo de execução não serializável: consideremos duas transações, “t1” e “t2”, que executam concorrentemente em locais diferentes do banco de dados. Além disso, admitimos que “t1’ lê um valor de um objeto “x” e executa atualizações no objeto “y” e “t2” lê “y” e escreve em “x”. Neste cenário, existe uma dependência entre transações que manipulam os mesmos conjuntos de dados. Caso seja aplicada uma associação epidêmica que mantenha e propague apenas operações de escrita, como descrito no algoritmo proposto por RABINOVICH, GEHANI, KONONOV (1996), as operações de atualização dos valores de objetos não poderiam garantir a consistência do banco de dados.

Para tratar a ocorrência de inconsistências com a aplicação do modelo epidêmico, podem ser incorporadas algumas alternativas, como por exemplo, o uso do protocolo de replicação otimista ou o pessimista (HOLLIDAY et al, 2003). Em um algoritmo epidêmico otimista as transações locais monitoram a ocorrência de conflitos e as inconsistências são tratadas posteriormente, durante a reconciliação do banco de dados. Já a versão epidêmica pessimista obriga a serialização e a garantia de execuções estritas durante a execução de uma transação.

4. RECONCILIAÇÃO DE DADOS

Por definição, unidades móveis não estão sempre conectadas a uma rede. As unidades móveis recuperam dados da rede e os armazenam na memória local, onde usuários têm acesso e manipulam esses dados. Isso gera a necessidade periódica de realização de re-conexões com a rede para enviar mudanças locais para o repositório de dados da rede fixa. Ocasionalmente é necessário que sejam solucionados conflitos entre as atualizações feitas localmente com os dados da rede fixa. Essa operação de reconciliação efetuada sobre conjuntos de dados de aspectos idênticos, onde são trocadas atualizações e são resolvidos os conflitos, é conhecida como sincronização de dados.

Para PHATAK, BADRINATH (2001) a sincronização em ambientes móveis pode ser definida como o ato de estabelecer equivalência entre duas coleções de dados, entre cliente e servidor, após a ocorrência de alteração nos registros armazenados. Depois de sincronização de dados cada elemento de dados em uma coleção é mapeado por um elemento de dados em outra, sendo que seus dados são equivalentes, entretanto não necessariamente idênticos.

Durante o processamento de uma consulta sobre banco de dados replicados, as transações são também replicadas, porém de forma assíncrona, propagando-se de acordo com a distribuição dos dados. O sincronismo ocorre somente após a reorganização dos dados, que poderá acontecer em um banco de dados fonte, seguido pelo envio das informações requeridas aos dispositivos móveis.

4.1. Modelos de transferência de dados

Quanto aos métodos de propagação usados, a replicação de dados é dividida nos seguintes modelos: modelo de baseado em sessão (*Session-Based*), modelo de baseado em mensagens (*Message-Based*) ou modelo de baseado em conexão (*Connection-Based*).

4.1.1. *Session-based*

Neste esquema de sincronização a replicação de dados ocorre através uma linha de comunicação direta entre a base consolidada e a base remota, como aplicado por SNOEREN (2002). A base remota cliente inicia a comunicação com o servidor de sincronização, enviando uma lista completa das modificações feitas naquela base desde a última sincronização. O servidor, então, atualiza a base consolidada e envia de volta as modificações relevantes. O *host* remoto incorpora o conjunto de mudanças e fecha a sessão logo após enviar uma confirmação de sucesso da operação junto com uma tabela que mapeia o conteúdo que foi atualizado (*mapping table*).

A sessão de comunicação neste contexto consiste de:

- Validação de segurança em cada sessão;
- Tarefas de comunicação, tais como a transferência de arquivos iniciada pelas aplicações dos clientes, comandos do sistema operacional e processos dos servidores centrais;
- Tarefas de aplicação, tais como responder as requisições de um cliente remoto específico;
- Distribuição eletrônica de *software* sob direção do servidor central;
- Monitoramento de recursos e diagnostico de tarefas, tais como espaço de disco e consumo de memória.

Staging é o processo na qual atividades a serem executadas durante a sessão subseqüente de comunicação são enfileiradas em um servidor central, assim como nos dispositivos remotos. Neste contexto, *staging* pode ser considerado como uma quantidade de trabalho a ser executado em momento posterior.

Programação de sessão é um processo na qual a sessão de trabalho é enfileirada e iniciada no servidor remoto. O servidor remoto pode acumular sessões de trabalho, analisar a sua execução e controlar o tempo, tipo de conexão ou disponibilidade de recursos (memória disponível na unidade móvel).

4.1.2. *Message-based*

A troca de dados entre as bases pode ocorrer também através de mensagens, que normalmente são arquivos armazenados em algum diretório particular ou mensagens de correio eletrônico com formato especial. De acordo com PARK, WOO, YEOM (2002) um agente de mensagens vinculado a cada elemento da rede pode enviar mensagens informando as mudanças recentes em sua base e receber mensagens de um ou mais outros agentes, modificando seus dados de acordo com o conteúdo delas.

Este modelo de sincronização permite replicação entre bancos de dados que não estão diretamente conectados. Cada uma das mensagens carrega informações de controle como o seu endereço de destino para que nenhuma conexão seja necessária entre as aplicações que se comunicam remotamente.

4.1.3. *Connection-based*

Em um esquema baseado em conexão, unidades móveis dependem da existência de uma zona de cobertura para ser estabelecida a conexão com um provedor de serviços. Em um ambiente móvel com diversas unidades móveis ocorrem freqüentes re-conexões devido à instabilidade da comunicação pelo meio *wireless*. Para amenizar esta inconveniência FUKUSHIMA et al (2001) propuseram um método que estima o tempo de conexão e a velocidade de movimentação do cliente usando o envio de sinais provenientes de uma estação-base. Baseando-se nessa alternativa é possível estabilizar a

conexão entre os *hosts* de um ambiente móvel, favorecendo a transferência completa de dados em operações de sincronização.

4.2. Gerenciamento dos Dados

4.2.1. Controle Centralizado

O gerenciamento da reconciliação de dados replicados cabe a um nó central, que fica responsável por reintegrar as atualizações efetuadas por instâncias de um banco de dados. Essa função é normalmente atribuída a um servidor estático apto a receber solicitações de todos os elementos da rede. As modificações ocorridas nas réplicas de itens de dados devem ser submetidas ao servidor que controla a instância global do banco de dados e propaga as modificações para as demais réplicas existentes no ambiente móvel.

Quando é aplicado um gerenciamento centralizado de réplicas de dados considera-se que exista apenas uma cópia válida de um item de dado (localizada no servidor). Na ocorrência de dados contraditórios entre os elementos do sistema, cabe ao servidor aplicar primitivas de controle de concorrência que garantam a integridade do banco de dados consolidado. O gerenciamento dos dados pode envolver tanto operações no próprio servidor quanto nas unidades remotas em decorrência da serialização dos bancos de dados.

Em uma de arquitetura de controle centralizado todos os clientes ficam dependentes do servidor para efetivar as mudanças executadas sobre as cópias de itens de dados, gerando um gargalo que pode influenciar o desempenho geral do sistema. Além disso, o fato de ser mantida apenas uma cópia válida do banco de dados diminui a segurança quanto à durabilidade do armazenamento dos registros do banco de dados.

4.2.2. Controle Distribuído

A alternativa de controle distribuído propõe que todos os elementos da rede tenham a capacidade de gerenciar as réplicas de dados localmente. Nesse caso, qualquer modificação deve ser disseminada para as demais instâncias do banco de dados. Sua aplicação permite manter o banco de dados consistente com a atualização constante dos dados, porém tende a gerar um alto custo de comunicação de rede.

4.3. Alternativas para Disseminação de Dados

De acordo com as conclusões de HUANG, SISTLA, WOLFSON (1994) é essencial que terminais móveis acessem suas bases consolidadas de forma a minimizar ao máximo o custo de comunicação. Isto pode ser obtido por meio da alocação apropriada de dados. Se um usuário realiza consultas freqüentes a um determinado item de dado que sofre raras atualizações, então torna-se importante alocar uma cópia do dado na memória do *host* móvel. Deste modo, as leituras subseqüentes do item de dado são realizadas na cópia local, não requerendo nova comunicação com o servidor. Na ocorrência de alterações, a nova versão atualizada do dado deve ser transmitida para o servidor a fim de garantir coerência da base de dados fixa. Em contrapartida, na ocorrência de consultas esporádicas a um determinado item de dado, não devem existir cópias em dispositivos móveis. Em vez disso, o acesso deve ser por demanda (toda operação de leitura deve ser transmitida diretamente para o servidor que contém a base de dados fixa).

Durante o período de conexão de uma unidade móvel com uma unidade fixa são feitas diversas requisições de serviços. Tais solicitações, que podem partir tanto do cliente quanto do servidor, têm a finalidade de executar operações ou simplesmente consultar ou transmitir dados remotamente. Para que se obtenha um bom desempenho na transferência de informações, JING, HELAL, ELMAGARMID (1999) descrevem os métodos *pull-based* e *push-based*, ambos focados à disseminação dos dados.

4.3.1. *Pull-based*

A estratégia de disseminação *pull-based*, descrita por RATNER (1998), atribui a unidade cliente o controle da reintegração dos dados com o servidor. Fica a critério do cliente móvel dar início ao procedimento de sincronização do banco de dados através do envio de uma requisição explícita ao servidor de sincronização. Ao servidor cabe apenas submeter dados ao cliente em resposta a uma consulta do cliente, conforme representado na Fig. 11.

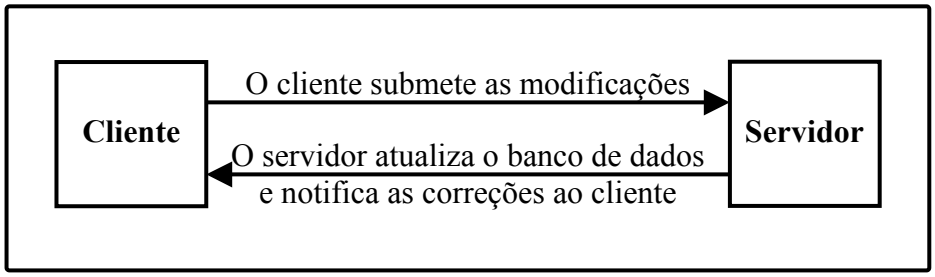

FIGURA 11 – Disseminação de dados com *pull-based*.

Conforme KARAKAYA, ULUSOY (2001) o emprego desse mecanismo de reconciliação de dados traz certos benefícios, tais como:

- **Baixo custo de comunicação**: o método sincronização proposto permite que o cliente acumule mudanças por um período de tempo indeterminado. Isso reduz o *overhead* computacional da rede, tendo em vista que serão necessárias poucas conexões com transferência de uma extensa quantidade de atualizações.

- **Simplicidade do algoritmo de transmissão de dados**: como a demanda de requisições ao servidor de sincronização é limitada pelo acúmulo de operações remotas, podem ser adotados mecanismos de comunicação mais simples que não precisem monitorar continuamente o canal de comunicação, como ocorre, por exemplo, com o uso de agentes móveis.

Por outro lado, o emprego do método *pull-based* faz com que servidores sejam continuamente interrompidos em decorrência das requisições externas. Durante o procedimento de sincronização o servidor fica impedido de executar transações adjacentes. O tempo para finalização das etapas de reconciliação dos dados submetidos pelo cliente dependerá do volume transferido e do relacionamento entre as tabelas do banco de dados. Enquanto tal conjunto de operações não é finalizado, os demais *hosts* devem tentar nova conexão ou apenas transferir as requisições para serem inseridas em uma fila de espera.

Outro fator negativo é que as operações de consulta ficam limitadas aos dados armazenados na memória do cliente. As transações baseadas na memória da unidade móvel não podem ser consideradas confiáveis, visto que os dados podem estar desatualizados com relação à base de dados consolidada.

4.3.2. *Push-based*

A filosofia de funcionamento do mecanismo *push-based* é a repetição simultânea do encaminhamento de mensagens, iniciada a partir da transferência de informações constituídas do servidor aos clientes (Fig. 12). O controle da disseminação dos dados é do servidor, que é capaz de enviar dados aos clientes sem serem consultados e ainda incorporar alguma forma de perfil dos usuários para decidir o que e quando transmitir.

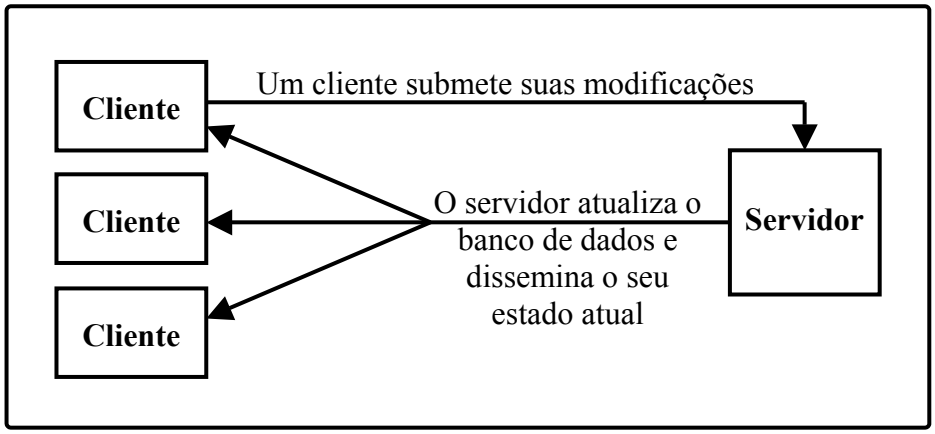

FIGURA 12 – Disseminação de dados com *push-based.*

Para ACHARYA, FRANKLIN, ZDONIK (1997) uma estação fixa de suporte à mobilidade pode ser utilizada para o envio de mensagens de forma difundida, conhecida como *broadcasting*. Essa metodologia consiste no envio simultâneo de informações para diversas unidades ao mesmo tempo. Dessa forma é possível otimizar a exploração da largura da banda, que é um recurso limitado em redes sem fio. Possui alto potencial de escalabilidade, uma vez que o desempenho não depende do número de destinatários. Cada transmissão é estabelecida de forma síncrona, onde o cliente necessita apenas fazer o monitoramento do canal de comunicação para coletar o item almejado. Além disso, se consegue uma considerável economia de energia dos dispositivos móveis, que não realizariam tentativas frustradas de conexão, ficando a encargo da unidade fixa encaminhar atualizações de forma consecutiva e conjunta.

A alternativa *push-based* assegura a rápida disseminação de dados, porém PITOURA, CHRYSANTHIS (1999) descrevem algumas restrições, tais como:

- **Necessidade de rede sem fio**: exige o emprego de algum meio de transmissão que permita a localização imediata da unidade móvel. Deve ser aplicada em ambientes formados por células que compõem uma zona de cobertura, dando a possibilidade do usuário mover-se dentro de um espaço delimitado.

- **Isolamento de dados**: os clientes tendem a desconhecer os dados armazenados nos servidores. Normalmente manipulam apenas a parte do banco de dados que lhe é atribuída.

- **Dependência externa**: os clientes dependem da iniciativa do servidor para manter a consistência dos dados.

- **Volatilidade dos dados**: em certas situações, a organização dos dados armazenados nos servidores pode mudar com freqüência, dificultando a formulação explícita de consultas.

- **Assimetria de comunicação cliente-servidor**: como os servidores tendem a ser em número bem menor do que os clientes e os dados enviados do servidor para o cliente são em muito maior volume do que no sentido contrário, converge para a ocorrência de uma sobrecarga de atividades que poderá influenciar no desempenho geral do sistema móvel.

4.3.4. *Hybrid mode*

Além da possibilidade da adoção dos mecanismos *pull-based* e *push-based* separadamente, ACHARYA, FRANKLIN, ZDONIK (1997) descrevem que é possível algum sistema móvel valer-se de ambos ao mesmo tempo. Essa combinação é denominada entrega de dados híbrida, que utiliza o canal de comunicação de forma compartilhada tanto para a difusão de mensagens quanto para requisições. Para tanto, são propostas estratégias para evitar uma sobrecarga de requisições ao servidor. Uma delas é a obrigar que o envio de uma mensagem sob demanda ocorra apenas se for atingido um valor predeterminado de difusões. Outra possibilidade é a divisão do banco de dados em duas partes, uma encarregada de suprir informações referentes às solicitações feitas pelos usuários e outra composta por dados classificados para disseminação.

4.4. Ordem da propagação de réplicas

Na comunicação via difusão os dados são transmitidos de tal forma que o computador móvel fica monitorando a comunicação para identificar a informação almejada. Para que esse processo funcione de maneira satisfatória é importante haver mecanismos de organização dos dados. Esses mecanismos devem ser utilizados para minimizar o tempo de transmissão entre o servidor e os clientes e preservar a escassa energia disponível na unidade móvel. Três formas de organização para transferência de dados são propostas: organização plana, organização por discos de difusão e organização por indexação.

4.4.1. Organização plana

Representa a organização mais simples para transmissão de dados. Após o servidor reunir as informações necessárias, elas são enviadas repetidamente até serem capturadas pelos clientes (Fig. 13). Visto da unidade móvel, este mecanismo não representa uma economia satisfatória de energia, pois exige que a comunicação se estabeleça até o recebimento completo do conteúdo requerido.

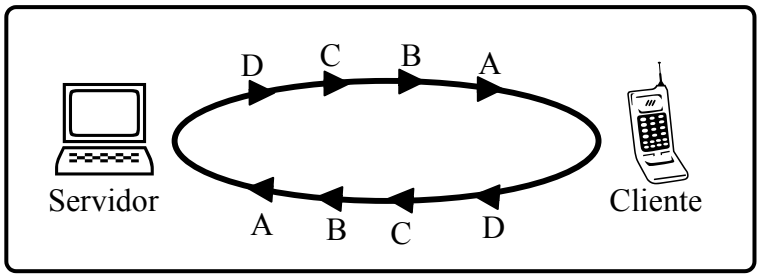

FIGURA 13 – Difusão de dados através da organização plana.
Fonte: (JING, HELAL & ELMAGARMID, 1999).

4.4.2. Organização por discos de difusão

Na abordagem por discos de difusão leva-se em consideração uma estimativa sobre os dados que mais interessam aos clientes. A freqüência em que estes dados são transmitidos definem os discos, isto é, os elementos transmitidos na mesma constância são considerados pertencentes ao mesmo disco (Fig. 14). A diferença de freqüência se dá devido as diferentes velocidades de processamento das informações solicitadas, já que se trata de um ambiente heterogêneo.

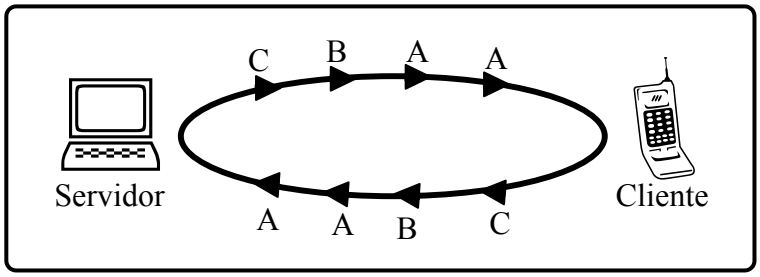

FIGURA 14 – Difusão de dados através da organização por discos de difusão.
Fonte: (JING, HELAL & ELMAGARMID, 1999).

4.4.3. Organização por indexação

A estrutura proposta aborda a utilização de índices atrelados aos dados transmitidos. Juntamente ao dado fica estabelecida a sua própria chave e também aquelas que ainda restam para finalizar a transferência, antes da nova difusão de índice (Fig. 15). Com isso, é possível que o dispositivo móvel faça uma estimativa de tempo para recebimento de um novo segmento de dado, ficando em modo de espera, economizando energia.

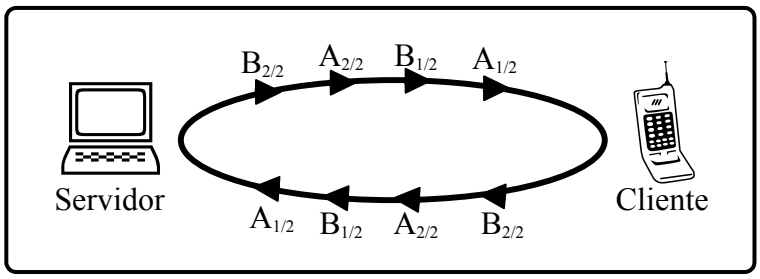

FIGURA 15 – Difusão de dados através da organização por indexação.
Fonte: (JING, HELAL & ELMAGARMID, 1999).

4.5. Mecanismo de *caching*

Com o objetivo de haver uma otimização na transmissão de dados são utilizadas as técnicas de *caching* e difusão, conjuntamente. Nesse caso, é mantida na *cache* da unidade móvel os dados com maior probabilidade de utilização, diminuindo assim a dependência com o servidor. Conforme LEONG, SI (1997) a manutenção através de *caching* é especialmente importante para prover uma melhoria no desempenho da aplicação móvel e para aumentar a durabilidade das operações em modo desconectado. A mescla dos dois mecanismos ocorre quando é executada a difusão para atualizar a memória *cache* na unidade móvel, onde ocorre a tentativa de otimização da difusão de dados para um conjunto de clientes com necessidades diferentes. Com isso, se um cliente específico não obter uma difusão ótima, poderá ser compensada pelo desempenho do *caching*.

Pode-se melhorar ainda mais o desempenho do *caching* aplicando-se uma metodologia chamada busca antecipada (similar àquela aplicada em arquiteturas de processadores). Tal metodologia baseia-se no armazenamento prematuro de informações que provavelmente serão acessadas, adotando-se heurísticas dinâmicas para cálculo de prioridades.

A atualização da *cache* no cliente é muito importante para manter a consistência dos dados armazenados. Isso significa que o cliente deve receber notificações para decidir se as informações são válidas ou não. Tais notificações podem ser assíncronas (quando o servidor transmite as notificações tão logo seus valores sejam alterados) ou síncronas (quando as notificações são transmitidas periodicamente). Em ambos os casos descritos anteriormente podem ser adotadas algumas estratégias de atualização, tais como:

- **Difusão por *timestamp***: junto com a notificação está atrelada uma etiqueta de tempo para identificar a última alteração realizada em um determinado dado.

- ***Amnestic terminals* (AT)**: são transmitidos apenas os identificadores dos itens que sofreram modificações.

- ***Signature***: é enviada uma assinatura (*checksun*), calculada sobre vários itens através de técnicas de compressão de dados.

No caso das notificações assíncronas, utiliza-se um método baseado em seqüência de *bits*. Cada notificação é organizada como um conjunto de pares, onde o primeiro elemento é um *bit* e o segundo representa o *timestamp*. O *bit* 1 representa uma alteração realizada conforme o *timestamp* subseqüente. Esse algoritmo pode ser empregado para grandes bancos de dados, alterando-se a granularidade do *bit*. Resumidamente pode-se dizer que, para cada conjunto de dados alterados é associado um *bit* de controle.

De acordo com a utilidade dos dados registrados na *cache* e os tipos de operações que serão realizadas sobre esses dados, há a seguinte subdivisão:

- **Dados somente para consulta**: são aqueles que não são submetidos a atualizações. Podendo ser acessados diversas vezes sem a necessidade da ocorrência de reintegração ao banco de dados.

- **Dados não concorrentes**: as atualizações no servidor de arquivos são bloqueadas durante o período em que a unidade móvel está alterando as informações localmente. Nesse caso, podem ser alterados apenas na unidade móvel, garantindo assim consistência geral do banco de dados.

- **Dados concorrentes**: possui um algoritmo de controle de concorrência complexo, pois deve garantir a integridade dos dados, mesmo quando cliente e servidor realizam atualizações sobre eles.

4.6. Freqüência de disseminação de dados

Conforme algumas considerações, AKSOY, ALTINEL, BOSE (1998) relatam que a freqüência com que dados são propagados em um ambiente móvel pode influenciar na manutenção da consistência do banco de dados. Cabe ressaltar que, quando existe a possibilidade de unidades móveis atuarem paralelamente sobre os mesmos itens de dados, torna-se importante que ocorra a reintegração para que os possíveis conflitos não sejam perpetuados pelo ambiente. Para tanto, são definidas duas formas de disseminação: propagação periódica e propagação condicional.

4.6.1. Propagação periódica

Em um ambiente que mantém os dispositivos móveis constantemente conectados entre si e com a base fixa é possível que seja aplicada a propagação periódica de mensagens de atualização. Segundo KIM, SONY, STANKOVICY (2003), propagações

deste tipo determinam a constante manutenção da consistência das réplicas distribuídas em um sistema móvel. A comunicação contínua facilita a disseminação de atualizações, porém acarreta em um crescente tráfego de rede e maior necessidade de recursos de *hardware.*

4.6.2. Propagação condicional

No caso da propagação condicional, dados atualizados em unidades móveis são propagados conforme a determinação do usuário ou da aplicação que gerencia procedimentos de sincronização, o que favorece o acúmulo de modificações locais, porém podem influenciar no estado geral do banco de dados. A alternativa de disseminação por condição pode ser aplicada em sistemas com fraca conectividade entre os elementos de rede. Em contrapartida, deve ser associada a mecanismos mais flexíveis para controle de concorrência, visto que modificações em cópias de dados localizadas em *hosts* móveis podem ficar isoladas por um longo período de tempo.

4.7. Formas de comunicação

4.7.1. Any-to-any

Considerando que os usuários esperam que computadores vizinhos devam sincronizar entre si, não podem ser previstas quais estarão dispostos geograficamente em determinado momento. Nesse caso, RATNER, REIHER, POPEK (2001) apontam a necessidade da replicação ser capaz de apoiar a comunicação *any-to-any*, que permite quaisquer máquinas comunicarem-se entre si, não podendo haver hierarquia entre clientes no sistema.

A comunicação *any-to-any* é equivalente ao modelo de replicação ponto-a-ponto, visto no Capítulo 3; se qualquer cliente pode sincronizar diretamente com outro, então, todos elementos do sistema devem ser, por definição, iguais; pelo menos com relação às habilidades de executar atualizações.

A consistência do banco de dados pode ser mantida corretamente até mesmo se duas máquinas não puderem sincronizar diretamente entre si, como no caso de alguns sistemas baseados no modelo cliente/servidor. Porém, o armazenamento local aumenta a usabilidade e o nível de funcionalidade, enquanto diminui o custo de sincronização inerente.

4.7.2. *Ad-hoc*

Redes *Ad-Hoc* são redes que não necessitam de uma infra-estrutura previamente projetada para funcionar. Segundo BUSZKO, LEE, HELAL (2001), os nós deste tipo de configuração de rede possuem a capacidade de se autoconfigurar através de algoritmos escaláveis de roteamento e disseminação de informação.

Várias vantagens e desvantagens podem ser citadas ao se comparar redes *Ad-Hoc* com redes infra-estruturadas e com as redes fixas (CANO & MANZONI, 2000), como por exemplo:

Vantagens:

- **Rápida instalação**, uma vez que as redes *Ad-Hoc* podem ser estabelecidas dinamicamente em locais onde não haja previamente uma infra-estrutura de rede instalada.

- **Tolerância à falhas**: a permanente adaptação e re-configuração das rotas em redes *Ad-Hoc* permitem que perdas de conectividade entre os nós possam ser facilmente resolvidas desde que uma nova rota possa ser estabelecida.

- **Conectividade**: dois nós móveis podem se comunicar diretamente desde de que cada nó esteja dentro da área de alcance do outro. Em redes infra-estruturadas ou em redes fixas, mesmo que dois nós estejam próximos é necessário que a comunicação passe pela estação de suporte à mobilidade (no caso de redes infra-estruturadas) ou, no caso de redes fixas, haver uma

ligação por meio de cabo entre os dois nós (DESAI, VERMA & HELAL, 2003).

- **Mobilidade**: esta é uma vantagem primordial com relação às redes fixas.

Desvantagens:

- **Roteamento**: a mobilidade dos nós e uma topologia de rede dinâmica contribuem diretamente para tornar a construção de algoritmos de roteamento.

- **Localização**: outra questão importante em redes *Ad-Hoc* é a localização de um nó. Pois, além do endereço de uma estação não ter relação com a posição atual do nó, também não existem informações geográficas que auxiliem na determinação do seu posicionamento no ambiente móvel.

- **Taxa de erros**: a taxa de erros associada a enlaces sem-fio é mais elevada.

- **Banda passante**: enquanto em meios de conexão por cabo a banda passante já chega em 1 Gbps, os enlaces sem-fio suportam tipicamente taxas de até 2 Mbps.

4.8. Condições de comunicação

As redes sem fio são normalmente de custo mais elevado, largura de banda reduzida e de menor confiabilidade se comparadas com redes fixas. Alguns fatores que contribuem para esse quadro, apontados por HAARDT, MOHR (2000), são a divisão da largura de banda entre usuários de uma área de cobertura (divisão em canais), altas taxas de ruídos, erros de comunicação e as freqüentes desconexões.

As desconexões podem se caracterizadas como voluntárias (o próprio usuário evita intencionalmente o acesso à rede) ou forçadas (quando uma unidade móvel entra em uma região onde não existe acesso à rede fixa por falta de um canal de comunicação ou cobertura do sinal de rádio). O *handoff* é um exemplo de desconexão freqüente e de curta duração, onde o *software* do dispositivo móvel opera em modo desconectado, de forma a diminuir a carga de comunicação entre a ERB [5] e a unidade móvel.

A telefonia IP pode ser adaptada de forma a possibilitar a comunicação entre os elementos que constituem um ambiente de computação móvel. Algumas características determinantes para o estabelecimento de condições de comunicação neste tipo de ambiente são descritas a seguir:

- Tráfego de pacotes em tempo real, sem a necessidade de retransmissão. Esta característica é garantida pelo protocolo de transporte RTP;

- Exigência de largura de banda constante, determinada pelo algoritmo de codificação da voz, compressão e supressão de silêncio;

- Serviço muito sensível a erros. A perda de um pacote pode implicar em ruídos na saída de dados e voz;

- O uso de diversos algoritmos de codificação de voz influi na largura de banda necessária para o serviço e na qualidade da reprodução da voz no outro extremo da conversação, de forma a ter possivelmente uma reprodução de voz mais mecanizada, com perda de alguns sons da fala;

- A adaptabilidade a ser trabalhada para telefonia IP pode ser feita sobre algoritmos de codificação e decodificação, definidos pela ITU-T, além da possibilidade de compactação e supressão de silêncio. Assim, a estação fixa pode alterar dinamicamente estes métodos, atuando diretamente sobre os dados, de acordo com a disponibilidade de largura de banda.

[5] Estação Rádio Base – empregada em sistemas móveis que transmitem dados por meio de ondas de rádio. O conjunto de ERBs formam uma zona de cobertura.

4.9. Tipos de sincronização

4.9.1. *One-way*

A sincronização *one-way* compreende a transferência de atualizações de dados em sentido único (PALAZZO & PULIOFITO, 2000). Um elemento de rede estabelece uma conexão com outro e submete todas as alterações que foram realizadas localmente. Após a finalização da transmissão, os dados são processados e aplicados no banco de dados do receptor. Nenhuma mensagem retorna a unidade requisitante. Essa metodologia visa minimizar o custo de comunicação com a baixa incidência de mensagens entre as unidades do sistema móvel.

Quando um elemento de rede submete as alterações realizadas localmente para outro elemento integrante do ambiente móvel, espera-se que não exista a possibilidade das transações recorrentes serem invalidadas pelo plano de controle de concorrência. Isso se deve ao fato que a unidade requisitante não recebe qualquer notificação de aceite ou erro na efetivação das requisições ao banco de dados.

O plano *one-way* pode ser aplicado de duas formas: no sentido cliente/servidor ou ao contrário, dependendo do tipo de solução projetada.

a) *One-way* cliente/servidor

Nesse tipo de configuração o cliente é responsável por iniciar a sincronização e submeter todas as modificações ao servidor, mas o servidor não retorna o resultado final das ações realizadas no banco de dados. A Fig. 16 representa a seqüência das etapas seguidas durante o procedimento da sincronização *one-way* motivada pela tentativa de reconciliação dos dados modificados ou inseridos a partir do cliente.

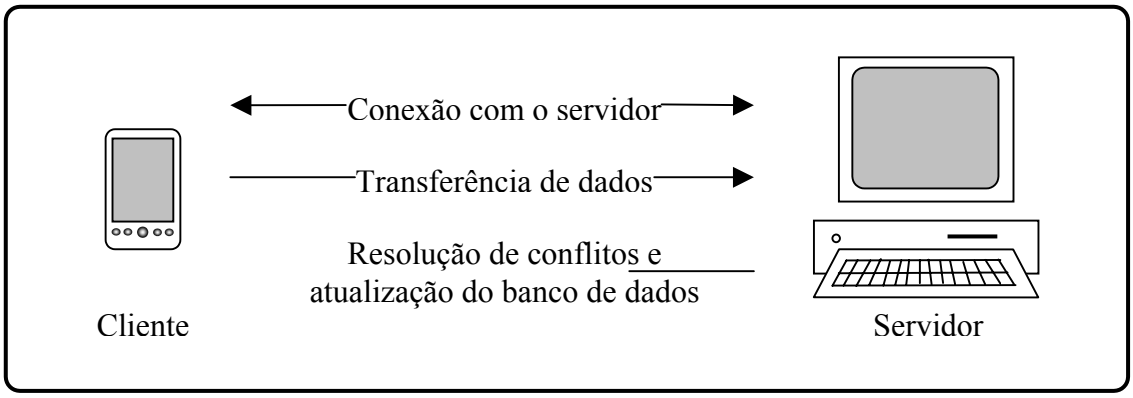

FIGURA 16 – Sincronização *one-way* cliente/servidor.

b) *One-way* servidor/cliente

A iniciativa de transmissão é do servidor, que conecta-se com o cliente e submete todos os dados atualizados no banco de dados consolidado (Fig. 17). Após o término da transferência, o cliente contém a versão atualizada de todos registros do sistema, mas o servidor não captura qualquer alteração do cliente.

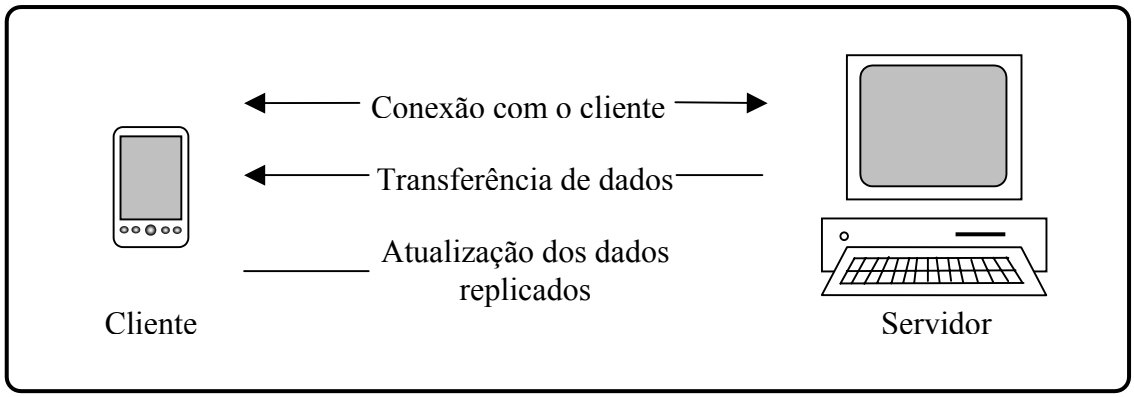

FIGURA 17 – Sincronização *one-way* servidor/cliente.

4.9.2. *Two-way*

O modelo *two-way* compreende a transmissão bilateral das modificações ocorridas tanto do cliente quanto no servidor. O procedimento de sincronização é iniciado pelo cliente, que submete as atualizações/inserções para o servidor (Fig. 18). Durante esse processo o servidor identifica e trata os conflitos existentes, retornando ao cliente o resultado efetivo das requisições ao banco de dados. No final da sincronização, o cliente e o servidor têm exatamente os mesmos dados armazenados.

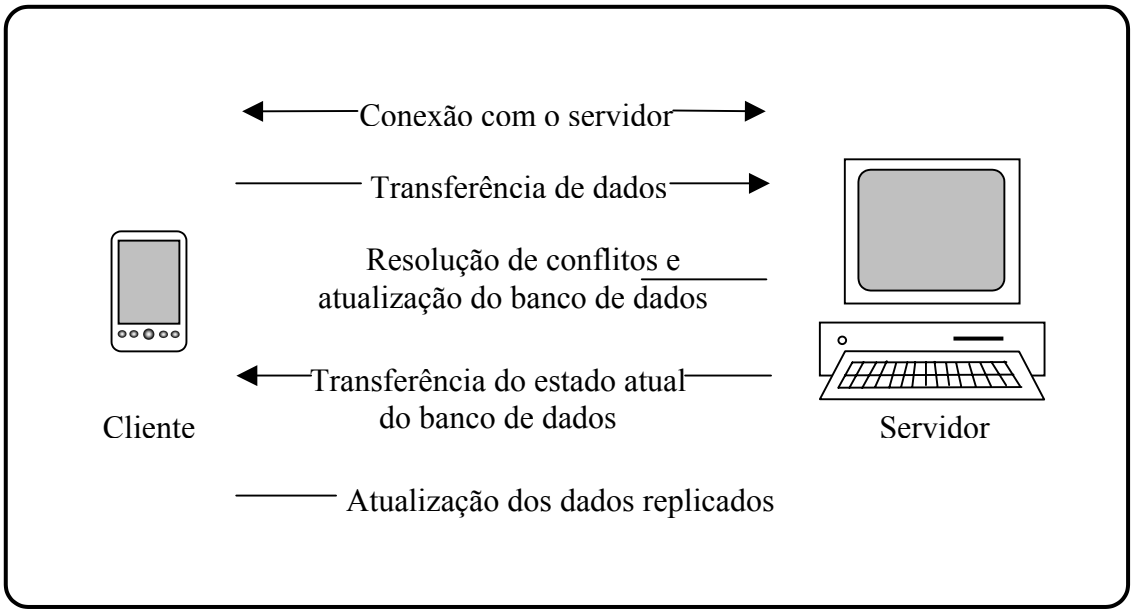

FIGURA 18 – Sincronização *two-way*.

4.10. Protocolos de sincronização

Um protocolo de sincronização define o fluxo de trabalho para a comunicação durante uma seção de sincronização de dados quando o dispositivo móvel é conectado à rede fixa. Para BREITBART, KOMONDOOR, RASTOGI (1999), deve dar suporte a identificação de registros, comandos de protocolo comuns para sincronização de dados local e de rede, e poder apoiar a identificação e resolução de conflitos de sincronização.

Atualmente diversos fabricantes oferecem uma variedade de produtos de sincronização de dados. Cada protocolo funciona apenas para transportes específicos implementados para alguns dispositivos. A proliferação de tecnologias de sincronização não-interoperáveis complica as tarefas de usuários, fabricantes de dispositivos, provedores de serviços e desenvolvedores de aplicações. A falta de um protocolo de sincronização de dados comum impede o crescimento do uso de dispositivos móveis, restringindo a habilidade de usuários para ter acesso dados e limitando a entrega de operações realizadas sobre dados móveis.

Aplicações que reconciliam e solucionam mudanças em dados entre duas fontes distintas são amplamente utilizadas por empresas de geração de *softwares* para sistemas móveis. Porém, o mercado de sincronização atual consiste em soluções proprietárias múltiplas, que não satisfazem todo o espectro de funcionalidades e o suporte requerido aos dispositivos pela maioria das organizações. Muitas organizações não estão dispostas a desenvolver múltiplos canais e a implementar várias soluções de sincronização para permitir o acesso de múltiplos dispositivos a um único banco de dados devido à complexidade e o custo de manutenção requerido.

Um protocolo necessita ser interoperável, sincronizar dados da rede com qualquer dispositivo móvel e sincronizar dados de um dispositivo móvel com qualquer rede. Isto inclui um espaço de mercado onde os dispositivos móveis têm recursos reduzidos de processamento e de memória, e são empregados servidores de rede capazes de executar uma lógica de sincronização sofisticada.

4.10.1. Protocolo de sincronização interoperável

Devido às diferenças estruturais em um sistema móvel e a imprevisibilidade de movimentação dos usuários ao longo das regiões do sistema, o ambiente computacional passa a ser altamente dinâmico (SILVA & ENDLER, 2000). A transmissão adequada de certas informações, como gráficos e figuras, exige uma capacidade que pode ser inviável para o sistema gerir. Tais condições levam a necessidade do projeto de aplicações com capacidade de interoperabilidade ao longo de diferentes ambientes de acesso sem fio. Pontos chaves no projeto de tais aplicações são: capacidade de identificação das condições do ambiente, adaptabilidade do modo de apresentação das informações e continuidade da prestação do serviço ao longo das mudanças fronteiriças. Como exemplo, pode-se imaginar uma arquitetura cliente/servidor, onde o servidor tem autonomia de escolha de dos dados a transmitir, baseado nas condições de tráfego de sua área de abrangência, assim como o cliente, que deve ser capaz de se adaptar a tais condições.

Um protocolo de sincronização comum tem que trabalhar sobre todas as redes usadas por dispositivos móveis. As várias redes *wireless* comumente usadas por

dispositivos móveis exigem muito de um protocolo, pois compartilham características comuns como alta latência de rede, largura de banda limitada, o custo relativamente alto de pacotes e a baixa confiabilidade de dados e conectividade.

• **Alta latência de rede**: é a demora introduzida quando um pacote é momentaneamente armazenado, analisado e então remetido de um porto a outro da rede. Redes *wireless*, com alta latência requerem um protocolo de sincronização robusto.

• **Largura de banda limitada**: para minimizar requerimentos de largura de banda e as demandas de processo associadas, o protocolo deve permitir alternativas como técnicas de codificação binárias para os dados e os comandos de sincronização. O WBXML (*Wap Binary Extensible Markup Laguage*), padrão adotado pelo Foro WAP[6] (*Wireless Application Protocol*) e submetido ao W3C[7](*Word Wide Web Consortium*) faz uso da técnica de codificação binária para ambientes de largura de banda limitada (DANTAS, 2002).

• **Custo de pacotes relativamente alto**: o protocolo tem que minimizar o número de iterações de *request-response* entre o dispositivo e os dados da rede. Um ótimo protocolo geraria um único par de *request-response* de mensagens. Nesse modelo, um pedido de um dispositivo móvel poderia conter todas as atualizações para seus dados locais. A mensagem de resposta proveria as atualizações, com conflitos já identificados e possivelmente resolvidos.

• **Baixa confiabilidade de dados e conectividade**: o protocolo tem que sobreviver a desconexões inesperadas durante o procedimento de sincronização. Mesmo na ocorrência de uma desconexão, deve assegurar que os dados do dispositivo e do repositório da rede permaneçam consistentes; e continuar a operação quando a conexão for restabelecida.

---

[6] ***Wireless Application Protocol*** – protocolo que define padrões para a transmissão de dados via redes sem fio.

[7] Comitê regulador que supervisiona a criação da XML (***Extensible Markup Language***), definindo quais recursos deve oferecer.

Com a utilização de um protocolo genérico, um usuário poderia ter acesso e manipular o mesmo conjunto de dados de dispositivos diferentes. Por exemplo, um usuário poderia ler um *e-mail* em um *handheld* ou em um telefone móvel, e ainda mantê-lo consistente, atualizando registros das mensagens que tenham sido lidas.

Semelhante a sincronização *any-to-any*, dispositivos móveis poderiam apoiar mais tipos de dados, inclusive *e-mail*, calendário, dados de empresas armazenados em bancos de dados e documentos na rede. Com tal funcionalidade um usuário que recebeu uma ordem por *e-mail* poderia acessar o sistema de inventário da empresa no mesmo dispositivo para determinar uma data de entrega. Para que isto seja possível, o protocolo precisa das seguintes características:

- Operar efetivamente sobre redes *wireless* e *wireline*;
- Suportar diversos protocolos de transporte;
- Tolerar dados arbitrários de rede;
- Possibilitar o acesso a dados por diversas aplicações;
- Suportar as limitações de recurso do dispositivo móvel;
- Assistir implementações para *Internet* e tecnologias de rede existentes.

5. AMBIENTE EXPERIMENTAL E RESULTADOS OBTIDOS

5.1. Descrição do ambiente experimental

O ambiente empregado no experimento é composto por dois clientes móveis que requisitam operações de consulta e atualização no banco de dados consolidado, vinculado a um servidor, que contém um repositório de dados concentrando todos os registros do sistema, conforme representado na Fig. 19.

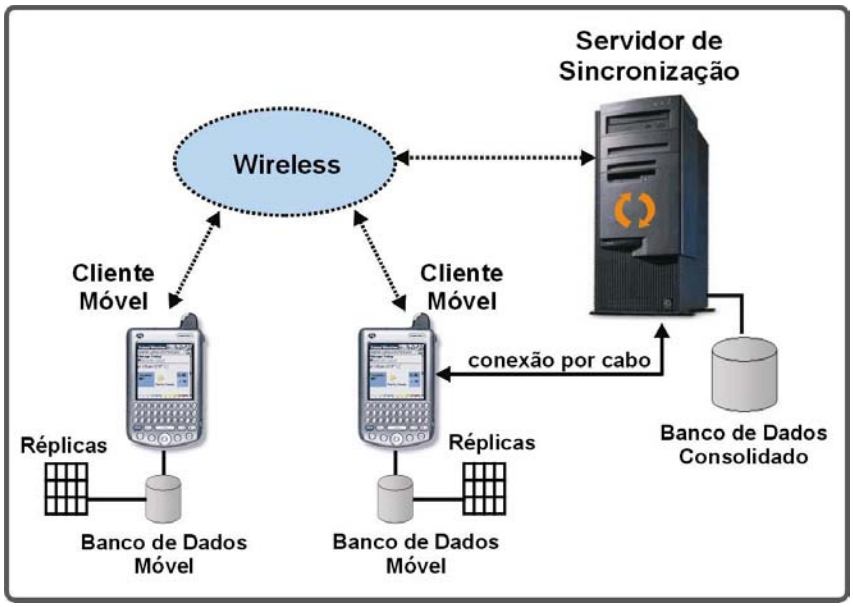

FIGURA 19 – Ambiente experimental.

Foram utilizados dois *palmtops*, titulados clientes móveis, baseados na plataforma PalmOS e um computador *desktop*, com Windows, atuando como servidor de sincronização no gerenciamento da base de dados fixa (Tabela 3). A comunicação entre esses elementos envolveu tanto a transmissão sem fio, via *wireless*, para a troca de dados entre as unidades móveis e o servidor de sincronização e entre elas próprias, quanto por cabo, com *interfaceamento* serial.

TABELA 3 – Descrição dos computadores envolvidos no experimento.

Nó	Modelo	Processador	Capacidade de memória	Sistema ional
Cliente móvel	Sony Clié PEG-T415	DragomballVZ 33 Mhz	8 MB	PalmOS 4.1
Cliente móvel	Palm Zire 71	OMAP 310, 144 Mhz	16 MB	PalmOS 5.2
Servidor de sincronização	Compaq D31M	Pentium IV, 1,7 Ghz	256 MB	Windows XP

Os *palmtops* são equipados com porta de infravermelho, oferecendo suporte à implementação ao padrão de comunicação IrCOMM, estabelecida pela *Infrared Data Association* (IrDA). Isso significa que dados podem ser transferidos para qualquer dispositivo próximo, contanto que tenha suporte para implementação IrCOMM nos padrões da IrDA. É possível, por exemplo, transferir uma entrada de um aplicativo para um computador com PalmOS®, ou um telefone celular ou um computador equipado com uma porta de infravermelho e que possa identificar dados do tipo vCard.

5.2. Protótipo desenvolvido

O protótipo desenvolvido é uma aplicação vertical para a área médica baseada no modelo de computação distribuída cliente/servidor. A Fig. 20 ilustra a tela inicial do sistema. O sistema permite que usuários móveis compartilhem dados provenientes de um banco de dados distribuído. Os clientes móveis atuam de forma dinâmica no ambiente através do envio e recebimento de requisições de/para o servidor de sincronização. As transações decorrentes das chamadas remotas são reproduzidas de uma forma assíncrona, propagando-se de acordo com a distribuição dos dados consolidados. As réplicas temporárias, localizadas na memória RAM, permitem que uma unidade móvel fique desconectada da rede por um período de tempo indeterminado

realizando operações sobre dados locais. Os dados são reintegrados durante um procedimento de sincronização executado com o servidor, que gerencia a base de dados consolidada e remete as devidas correções para as unidades clientes que detém as cópias de itens de dados.

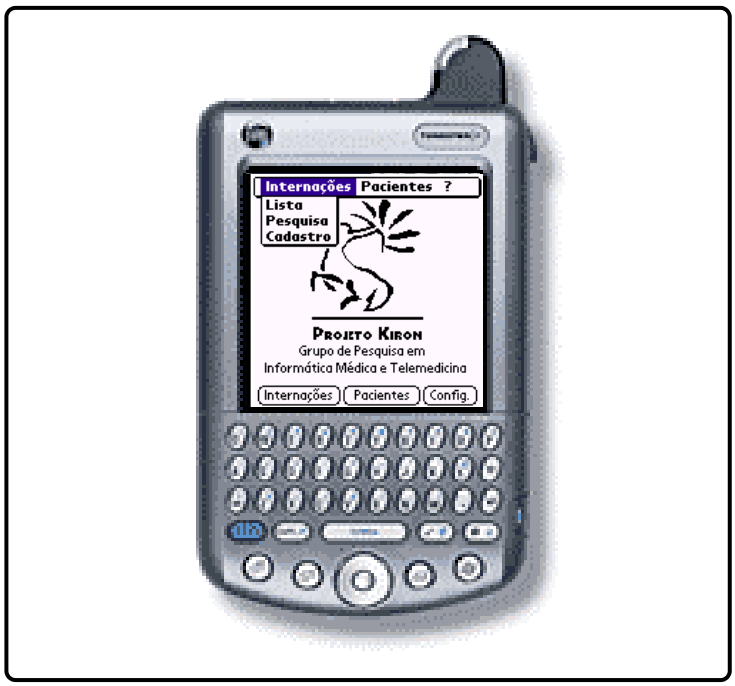

FIGURA 20 – Tela principal da aplicação cliente.

Visto sob a perspectiva do desenvolvimento de *software*, o sistema pode ser dividido em sete planos de aplicações, conforme mostrado na Fig. 21, que se comunicam entre si para prover desde o acesso até a organização dos dados do sistema de banco de dados distribuído.

A funcionalidade de cada parte do sistema é descrita a seguir:

1°) A *interface* da aplicação cliente permite que o usuário manipule as réplicas de itens de dados localizadas na memória do dispositivo móvel. A ações concebidas são: consultas, atualizações ou inserções de dados, conforme as definições de acesso que delimitam ações para tipos de usuários específicos.

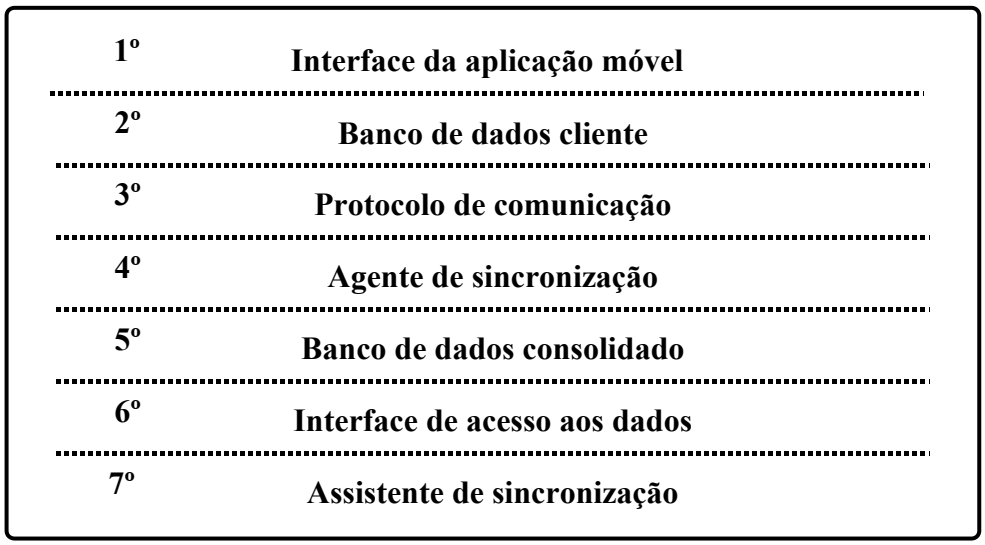

FIGURA 21 – Disposição das aplicações do experimento.

2º) O banco de dados cliente é organizado em um conjunto estruturado de relações que armazena dados replicados provenientes da base de dados fixa.

3°) O protocolo de comunicação é responsável por fazer a conexão e a transferência de dados entre o cliente e o servidor. Para este experimento foi utilizado o HotSync®, que é o *software* para sincronização de dados incorporado ao PalmOS e que possui uma versão para o *desktop* chamada HotSync Manager® (associado ao Palm™ Desktop – organizador de dados pessoais). Esta aplicação desenvolvida pela PalmSource[8] é capaz de utilizar vários meios de comunicação, tais como: serial, IrDA, *Bluetooth* e *Wi-Fi*.

4°) O agente de sincronização é uma aplicação acionada pelo HotSync®, desenvolvida para executar os procedimentos de sincronização entre o banco de dados do cliente e do servidor. Para a aplicação do cliente foi definido um conjunto de opções que determinam como os registros são tratados durante a sincronização, incluindo a varredura em busca de modificações na base remota e a geração de transações sob o banco de dados principal para repercutir o estado das réplicas e dos novos registros, se houverem. Além disso, atua na notificação e correção de possíveis inconsistências provenientes da atualização simultânea de réplicas em unidades remotas.

[8] Empresa responsável pelo desenvolvimento do sistema operacional PalmOS e de diversos aplicativos para PDAs.

5°) O banco de dados representa a organização lógica das tabelas e relacionamentos. Engloba o modelo conceitual do banco de dados utilizado como referência para as demais instâncias de itens de dados distribuídas pelo sistema, também definido como banco de dados consolidado.

6º) A *interface* de acesso aos dados, implantada no servidor, serve como mecanismo de entrada e gerenciamento de dados, bem como para emissão de relatórios e geração de gráficos estatísticos com base no banco de dados consolidado.

7°) O assistente de sincronização possibilita a seleção dos registros que serão transferidos para o *palmtop*, evitando a emissão desnecessária de dados que não serão utilizados pelo usuário. Também é responsável restringir determinadas replicações, conforme a permissão definida para cada tipo de usuário no sistema.

5.3. Clientes móveis

Os *palmtops* são definidos como clientes móveis, que mantém réplicas de registros do banco de dados do servidor. Possuem um papel importante no sistema, pois fornecem subsídios para que usuários possam realizar consultas e modificações nos próprios dispositivos móveis, sem precisarem estar conectados por cabos ou dentro de uma região de cobertura composta por antenas retransmissoras de sinais de rádio.

Cada cliente móvel é composto por um banco de dados resumido (planejado a partir do banco de dados consolidado) e uma *interface* de acesso aos dados (responsável por fornecer um ambiente intuitivo para a manipulação de registros médicos).

5.3.1. Banco de dados resumido

O banco de dados foi projetado para fornecer uma estrutura suficiente para armazenar apenas os dados necessários para suprir certos tipos de requisições de usuário. Criado a partir do esquema conceitual do banco de dados centralizado. Tem o

papel de repositório temporário de dados para suprir transações que deverão ser preservadas até a ocorrência de uma sincronização. Depende tanto do gerenciamento do *software* de acesso local quanto das ações emitidas pelo servidor de sincronização.

Todas as réplicas de dados armazenadas são gravadas na memória RAM do dispositivo móvel através de um formato específico indicando que é um banco de registros, diferenciando de outros tipos de informações provenientes de aplicações proprietárias, como calendário, agenda, dentre outros *softwares* embutidos no sistema operacional PalmOS.

Para este experimento foram definidos quatro bancos de dados para o *palmtop*, descritos na Tabela 4, que darão suporte as requisições da aplicação cliente.

TABELA 4 – Descrição dos bancos de dados para o *palmtop*.

Identificação	**Finalidade**
PacientePDB	- Armazenar dados essenciais para identificação de um paciente. O conteúdo armazenado reflete uma porção resumida das informações cadastradas no banco de dados principal.
InternacaoPDB	- Conter informações referentes às internações relacionados às cópias de registros pacientes.
AcompPDB	- Servir com base para inserções de registros que informam o estado atual do paciente. - Fornecer subsídios para a geração de dados estatísticos e de controle médico.
ConfigPDB	- Identificar o usuário. - Monitorar atualizações.

A arquitetura do banco de dados para o *palmtop* não é organizada por atributos, índices e chaves primárias, como ocorre em um banco de dados convencional. É uma estrutura seqüencial de registros classificados através de uma função da API do PalmOS que recebe um identificador único do registro. Os dados são arranjados em blocos com

tamanhos variáveis (*Raw Data*) representando um conjunto de informações dentro do banco de dados. A estrutura básica pode ser vista na Fig. 22. Esta estrutura foi utilizada no protótipo para permitir ao servidor de sincronização identificar as mudanças nos dados e fazer as devidas correções.

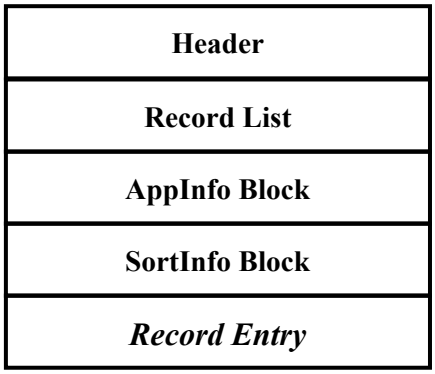

FIGURA 22 - Arquitetura do banco de dados do *palmotp*.

Header: cabeçalho de identificação do banco de dados. Local onde estão detalhadas as localizações dos blocos de informações. É representando por um conjunto de 72 bytes que determinam as características de acesso aos dados, conforme descrito na Tabela 5. Para o experimento, tem extrema importância por fornecer informações que favorecem a correção de erros ocasionados por operações ilegais sobre o banco de dados. Um exemplo é a análise da data da última modificação, que evita a exame completo dos registros pelo servidor em busca de atualizações.

TABELA 5 – Composição do *header* do *Palm Database*.

Posição (bytes)	Conteúdo
1 a 32	Nome do banco de dados.
33 a 34	*Flags* que identificam as características do banco de dados: 0x0002 – Somente para Leitura. 0x0004 – AppInfoBlock Modificada. 0x0008 – *Backup* do banco de dados. 0x0010 – Aceita a sobreposição de dados. 0x0020 – Na instalação, não inclui registros. 0x0040 – Impede a transmissão por infravermelho (*beaming*).
35 a 36	Versão do banco de dados.

37 a 40	Data de criação do banco de dados.
41 a 44	Data de Modificação do banco de dados.
45 a 48	Data do último *backup* do banco de dados.
49 a 52	Número de modificações no banco de dados.
53 a 56	Posição absoluta a partir do início do *Header*, identificando a localização do *AppInfo Block*.
57 a 60	Posição absoluta a partir do início do *Header*, identificando a localização do *SortInfo Block*.
61 a 64	Identifica o tipo do Banco de Dados.
65 a 68	Identificador único do criador do banco de dados (*Creator ID*).
69 a 72	Armazena um número de identificador único para o PDB (*Unique ID Seed*).

Record List: relação seqüencial de todos os registros gravados no banco de dados. Informa a posição absoluta dos dados a partir do *Header*. Cada entrada de registro corresponde a 8 *bytes*. Engloba o identificador único do registro (*Unique ID*), que é definido automaticamente pelo PalmOS quando um registro é criado. Além disso, a *Record List* é composta por códigos binários que refletem o estado do banco de dados, conforme descrito a seguir:

- *Secret*: registro privado;
- *Busy*: registro ocupado pelo sistema operacional;
- *Dirty*: registro modificado recentemente;
- *Deleted*: registro excluído.

Conforme o resultado do teste sobre o *Record List* é possível determinar se é preciso haver sincronização de determinado conjunto de dados para o banco de dados consolidado.

AppInfo Block: área destinada a informações diversas a critério do desenvolvedor. Também armazena a categoria de um banco de dados ou de registros. O PalmOS dispõe de rotinas que permitem filtrar a busca de registros por categoria. Um exemplo típico é a aplicação *Address©*, onde é possível classificar os registros como *Business, Personal, QuickList*, etc. Atualmente são permitidas 16 categorias por banco de dados.

SortInfo Block: área sem formato e tamanho pré-definido, destinada a aplicações específicas do desenvolvedor. Sua localização é dada pela posição absoluta marcada no *Header*.

Record Entry: os registros que foram apontados na *Record List*, são colocados logo após o último bloco de dados específicos da aplicação (*AppInfo/SortInfo*), se houver.

5.3.2. Interface da aplicação cliente

O desenvolvimento do protótipo incluiu a criação de uma aplicação cliente que tem a finalidade de prover o acesso e o gerenciamento do banco de dados localizado na unidade móvel. Todas as ações realizadas sobre os dados locais repercutem na base consolidada durante um procedimento de sincronização.

Através da distribuição homogênea dos registros de pacientes na GUI (*Graphical User Interface*) da aplicação cliente (ilustrada parcialmente na Fig. 23), o usuário móvel tem acesso às listas de pacientes internados na Unidade de Tratamento Intensivo e as respectivas informações que envolvem o monitoramento de cada paciente. Como padrão, a navegação entre as diversas telas do sistema é feita por meio dos botões direcionais e de caneta *stylus*. Para a facilitar a entrada de dados fez-se necessária a disponibilização de *menus* suspensos, diminuindo a necessidade do uso de recursos adicionais, como teclado virtual e área de reconhecimento de escrita.

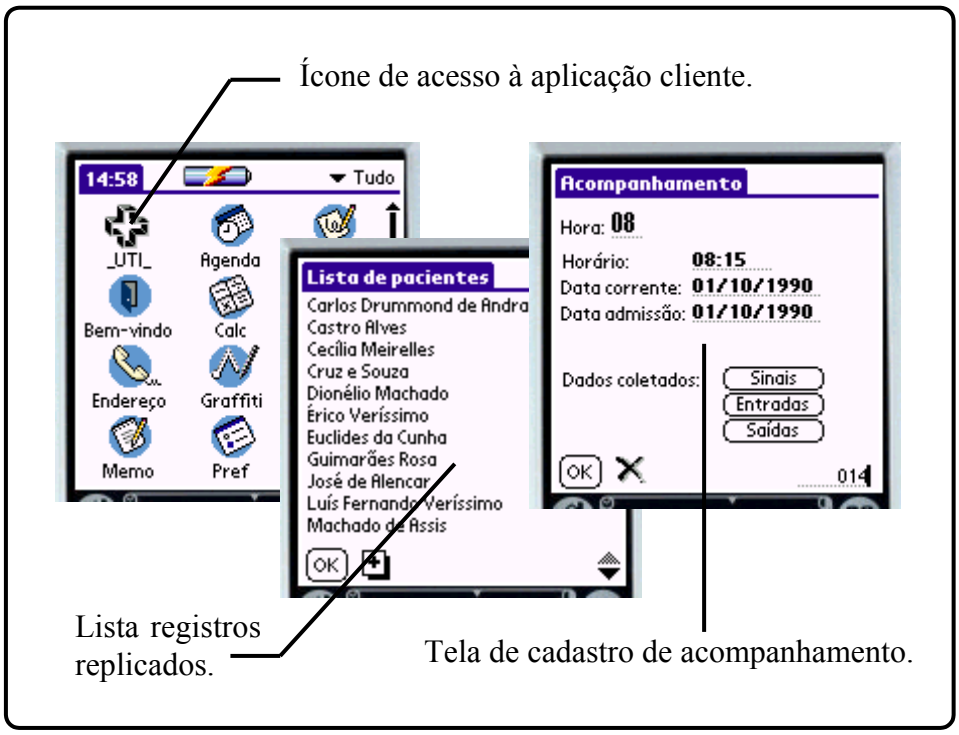

FIGURA 23 – Telas da aplicação cliente.

A linguagem de programação utilizada para a construção da aplicação cliente foi a PocketStudio Professional©. Desenvolvida pela Pocket-Technologies™ desde 1999 e lançada em 2001, a PocketStudio© conta com um compilador que gera aplicações nativas PalmOS de tamanhos reduzidos, sem a necessidade de *runtime* (POCKET-TECHNOLOGIES, 2003). Combina um compilador de 32 *bits* com a linguagem *Object Pascal*, com acesso as APIs do sistema operacional PalmOS. Oferece suporte a diversos equipamentos como HandEra, HandSpring, Kyocera, Palm, Symbol, Sony, impressoras, leitores de códigos de barras e celulares, além de bibliotecas de comunicação remota, como o ASTA.

A ferramenta PocketStudio© engloba três versões.

- ***Standard***: inclui componente simples para a criação de aplicações horizontais (conforme descrito no Capítulo 2).

- ***Professional***: versão desenvolvida para dar suporte a uma ampla variedade de aplicações (verticais e horizontais) com acesso a bancos de dados. Inclui recursos para múltiplos segmentos (para aplicações que excedem 32 KB), tecnologia TDK *bluetooth*, *conduit wizard* (construção automática de *conduits* genéricos) e o *POSE (Palm OS Emulator* – programa que emula o sistema operacional PalmOS em um computador *desktop* com Windows).

- ***Enterprise/Wireless***: compõe todas as características da versão *Professional*, mais um componente chamado *Windows Server Component*, que possibilita a criação de aplicações que conectam os *palmtops* a qualquer servidor de banco de dados usando *Internet* por *wireless*, com autenticação de usuários, criptografia e compactação de dados.

Para a concepção da aplicação cliente foram utilizadas estruturas de dados conhecidas como *PalmOS Resources*, vista em MASS (2002), que descrevem as características dos elementos que compõem a *interface* de usuário. Uma *interface* é composta por formulários, caixas de diálogo, menus e outros elementos que abstraem a complexidade operacional do sistema ao usuário (OSTREM, 2002). A título de experimentação foi utilizado a PSLibrary, que é uma *interface* de acesso a objetos visuais de formulários e a bancos de dados. Os testes foram efetuados em um *emulador* para PalmOS (POSE Emulator©) integrado ao *compilador* da linguagem de programação PocketStudio©.

A combinação de um conjunto de *resources* forma a aplicação cliente. No caso do experimento descrito neste trabalho, englobou cinco categorias de *resources*, descritas a seguir:

- ***Main Project File*** (ProjectUTI.ppr): arquivo principal do projeto do sistema, composto por uma lista com todas as *units* incluídas no sistema e o ciclo de processamento de todos os eventos relacionados às GUIs (telas, menus e mensagens de alertas).

- ***Resource File*** (ProjectUTI_RC.rc): contém uma descrição textual de todos os recursos usados no projeto.

- ***Project Event File*** (ProjectUTI_EVN.evn): código para a manipulação de *handlers* (monitoramento de ações sobre botões e listas suspensas), funções e procedimentos.

- ***Source Code File*** (*.pas): fornece o código fonte para cada *unit* do sistema baseando-se na linguagem Object Pascal.

- ***Compiled Application File*** (BDMKiron.prc): resultado da compilação do código-fonte para o sistema operacional PalmOS. Inclui a aplicação e o banco de dados PDB[9] associado.

5.3.3. Adaptações ao ambiente móvel

Algumas adaptações tiveram que ser realizadas para suprir restrições de tipos de dados impostas pelo PalmOS. Observou-se que domínios de diversos atributos, tais como data, hora e números reais possuem incompatibilidades nos formatos entre os sistemas operacionais PalmOS e Windows. Essas diferenças se devem aos padrões de representação de valores adotados por diferentes países. A solução encontrada foi transformar esses atributos em valores alfanuméricos (tipo *String*) para serem manipulados no *palmtop*. Ao serem sincronizados, são automaticamente convertidos para os formatos definidos no banco de dados consolidado, por meio de operações como: *StrToDate*, *StrToDateTime* e *StrToFloat*.

Um dos problemas fundamentais das operações sobre réplicas de itens de dados sob comunicação descontinuada é a possibilidade do usuário estar manipulando registros inconsistentes. Isso ocorre devido à existência de cópias paralelas de registros comuns entre diferentes unidades clientes. Para diminuir a incidência de problemas deste tipo foram adotadas algumas medidas conservativas, descritas a seguir:

---

[9] ***Palm Database*** – Banco de dados para *palmtops*. São manipulados pelos aplicativos (arquivos PRC).

- Bloqueio de escrita: determinados atributos são bloqueados para a realização de alterações na unidade cliente, para diferentes tipos de usuários.

- Baixa durabilidade: ao final de um procedimento de sincronização, no sentido cliente/servidor, todos os registros são removidos do banco de dados cliente. Para recompor o banco de dados é preciso que o usuário informe quais dados pretende manipular e requisitar nova sincronização.

5.4. Tratamento das limitações de recursos

5.4.1. Utilização da memória

Em um sistema móvel cada acesso ao banco de dados pode desencadear diversas transações remotas. Essas transações podem incidir direta ou indiretamente sobre o banco de dados consolidado. As possibilidades envolvem a utilização de um servidor para submissão de requisições constantes a base de dados fixa ou a sustentação de segmentos completos de registros na unidade móvel.

Na primeira alternativa espera-se que a unidade móvel tenha possibilidade de transmissão constante de dados via *wireless*, bem como permaneça dentro da área de cobertura desse serviço. A disponibilidade de memória para este caso não é fundamental, mas sim a velocidade de processamento e de transmissão de dados.

Na segunda alternativa o enfoque é dado a quantidade de memória disponível, já que as transações atuam no próprio *host* móvel. Nesse caso, os dados armazenados na unidade cliente devem ser classificados de forma a atender as requisições momentâneas do usuário sem degradar da velocidade de processamento em decorrência do uso extensivo da memória.

No experimento optou-se pela sustentação de parcelas do banco de dados na memória da unidade móvel. Com isso, foi possível fazer com as requisições ao banco de

dados gerassem apenas transações locais. As principais vantagens observadas no uso dessa abordagem são:

- Inexistência de limites geográficos para a localização das estações móveis durante operações sobre os itens de dados;

- Menor exigência de recursos de *hardware*, já que não obriga o uso de pontos de acesso à rede;

- Facilidade de expansão da capacidade de memória dos dispositivos portáteis com a adição de cartões de memória *flash*.

Uma situação crítica do uso da memória da unidade cliente como ponto de referência para a aplicação é a inconsistência do banco de dados ocasionada pela sustentação de dados na memória por um longo período de tempo. Diversos clientes podem fazer consultas e modificações sobre itens de dados que estão em desigualdade com a réplica precedente. Para contornar o problema, todos os registros remanescentes na unidade móvel, após a ocorrência da sincronização com o banco de dados são eliminados. Isso faz com que o usuário tenha que recarregar a base de dados remota apenas com os registros que serão necessários para satisfazer as novas transações locais.

5.4.2. Monitoramento da fonte de energia

Os dispositivos móveis, de modo geral, oferecem baixa durabilidade da fonte de energia, comprometendo a preservação dos dados. Uma forma de tentar preservar os registros é através da inspeção da quantidade de carga da bateria. Na ocorrência de baixa carga, equivalente a 10% da capacidade total, a aplicação cliente pode submeter automaticamente os dados à base fixa. Essa medida conservativa diminui a probabilidade da perda de registros em conseqüência da falta de energia, porém obriga a existência de comunicação constante entre o cliente móvel e o servidor de sincronização. Portanto, é somente apropriada enquanto o usuário estiver na área de cobertura de rede. Na impossibilidade de conexão com o servidor, não existe garantiria a durabilidade do banco de dados. Uma ação paliativa encontrada para aumentar a

durabilidade do banco de dados foi dar autonomia a aplicação cliente impedir novas inserções e modificações nas ocasiões em que a fonte de energia esteja comprometida.

5.5. Planos de replicação investigados

Para a obtenção e manipulação de réplicas de itens de dados, podem ser seguidos diferentes planos, dependendo do contexto em que o banco de dados é estabelecido, os tipos de usuários e as possíveis requisições. Os níveis de replicação investigados abrangem a ausência de replicação, a replicação parcial e a replicação total do banco de dados. Também estão atrelados fatores limitantes como a disponibilidade de acesso e a necessidade de armazenamento dos dados, como descrito a seguir:

5.5.1. Ausência de replicação

A não geração de cópias de linhas de dados envolve a transferência completa de determinadas *tuplas* do banco de dados consolidado para o banco de dados móvel. Torna único no sistema, um registro submetido ao evento de sincronização.

No âmbito da aplicação desenvolvida para a área médica, dados de pacientes foram transferidos via procedimento de sincronização, incluindo as tabelas relacionadas. Observou-se que o emprego desse método de replicação garantiu o controle de concorrência, já que impediu a existência de cópias de itens de dados no sistema. Porém, na prática, observou-se que esse tipo de metodologia não é interessante de ser aplicada, visto o impedimento da realização de consultas multi-usuários, mesmo que não resultem em modificações no estado do banco de dados. Em alguns casos, operações de consulta envolvem a junção de itens localizados em diversos pontos da base de dados, que podem estar “bloqueados” por tempo indeterminado e, em muitos casos, sem uso, já que não são constituídos fragmentos. A liberação desses itens para a realização de transações depende do evento de sincronização, emitido pelo usuário voluntariamente, no sentido cliente/servidor.

5.5.2. Replicação parcial

Na replicação parcial, somente os fragmentos solicitados pelos usuários são replicados. Quando tal modelo é aplicado, réplicas de itens de dados podem estar distribuídas por diversos clientes em um determinado instante, sem haver ciência de quais as linhas do banco de dados central possuem cópias remotas. Em alguns casos as cópias são estendidas entre linhas adjacentes, conforme o relacionamento existente entre as tabelas do banco de dados. No período de reintegração, são realizadas operações de controle de concorrência para garantir a integridade do sistema, que podem invalidar mudanças nos dados, conforme o estado do *flag,* do tipo *timestamp,* associado ao registro da *tupla* reintegrante.

A replicação parcial foi aplicada no experimento através do desenvolvimento de um assistente de sincronização (ilustrado na Fig. 24), responsável por marcar as linhas de dados para serem replicadas durante um evento de sincronização. Tal assistente foi criado para atuar de forma passiva, configurando e ligando tabelas, conforme requisições estabelecidas por usuários restritos. Para tanto, foi necessária a execução de adaptações no banco de dados fixo, que teve o incremento de dois novos de atributos de marcação (*flags)*, "palm_n" e "palm_m", com domínio *booleano*. O primeiro, definido para identificar as linhas de dados a serem submetidas ao evento de sincronização, tem função temporária, sendo inspecionado e restabelecido na submissão dos dados à base remota.

O papel do agente de sincronização (*conduit*) é mapear as linhas de dados da base fixa que estão assinaladas e incluí-las na base de dados remota. No procedimento inverso, com a transferência cliente/servidor, em decorrência de pedido explícito de sincronização pelo usuário, as linhas de dados devem ser reintegradas e removidas automaticamente da base remota, evitando a existência de réplicas desnecessárias. O segundo atributo de marcação (palm_m) foi criado para assinalar as linhas modificadas por operações de usuários na própria unidade fixa. O domínio desse atributo pode ser substituído por *timestamp*, conforme o grau de concorrência previsto no sistema. Nesse caso, o *conduit* se encarrega de comparar os dois bancos de dados e determinar a serialização mais adequada.

A utilização do método de replicação parcial é importante para que se tenha um melhor gerenciamento de memória do dispositivo portátil. Para o uso médico, por exemplo, na manipulação dos dados de pacientes internados na UTI, não seria conveniente a transmissão completa dos registros, já que, nem todos os dados seriam necessários. Além disso, o volume de informações pode ser maior que a capacidade de armazenamento da memória, o que inviabilizaria o procedimento de sincronização.

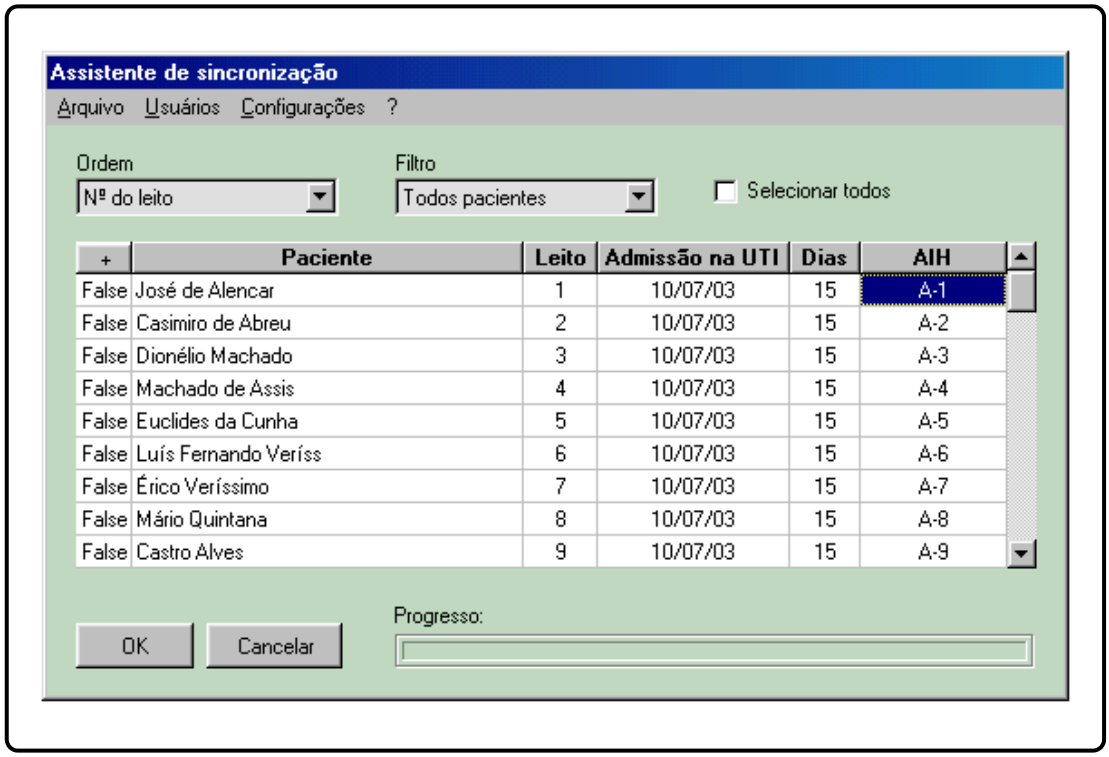

FIGURA 24 – O assistente de sincronização.

5.5.3. Replicação total

O modelo de replicação total sugere a replicação completa do banco de dados. Tal procedimento aumenta a necessidade de mecanismos otimizados para controle de concorrência, já que tende a existir diversas cópias de itens de dados espalhadas entre as estações móveis. Um ponto positivo é o fato de tornar a unidade cliente auto-suficiente, com alta disponibilidade de dados às requisições de usuário. Isso também repercute no índice de comunicação entre cliente e servidor, pois as réplicas tendem a suprir as

necessidades das operações de consultas, tornando esporádicas as transferências de dados adicionais. Como ponto negativo, pode ser apontada a probabilidade da manipulação de itens de dados desatualizados pelo cliente, em decorrência da não serialização do banco de dados, tornando o sistema freqüentemente inconsistente.

5.5.4. Replicação híbrida

O desenvolvimento do protótipo do experimento combina os modelos cliente/servidor e ponto-a-ponto. Um servidor é tomado como base para atualização de registros provenientes das estações móveis, definidas como clientes, centralizando o gerenciamento do banco de dados fixo. Além disso, é dada autonomia para transferências de itens de dados entre os elementos móveis do sistema.

A aplicação do modelo híbrido tem o propósito de dividir o controle das operações envolvidas com o banco de dados consolidado dando autonomia tanto às estações fixas quanto às que tem mobilidade. As atualizações são encaminhadas pelas unidades clientes durante procedimentos de sincronização com a base fixa. A efetivação de possíveis mudanças ou inserções de itens de dados, ocorridas remotamente, devem ser serializadas pelo servidor por meio de um agente de sincronização (*conduit*), responsável por analisar e tratar possíveis conflitos.

5.6. Emprego do protocolo de replicação otimista

O protocolo de replicação otimista visa fornecer subsídios que permitam garantir a execução de transações sobre dados localizados nas estações móveis sem a necessidade do acesso constante ao banco de dados consolidado, conforme descrito no Capítulo 3. Envolve a transmissão de um conjunto de itens de dados suficientes para dar suporte a requisições de usuário, diminuindo a necessidade de freqüentes conexões com o servidor.

Para FELBER, SCHIPER (2001), no desenvolvimento de algoritmos de replicação de dados e serviços distribuídos deve ser levado em consideração fatores como a eficiência das respostas as solicitações de usuários, com o uso adequado de mecanismos de transmissão e garantia de consistência das réplicas de itens de dados. Baseando-se nestas considerações, somadas a demanda crescente de registros de pacientes e as necessidades de segurança, concluímos que o emprego do modelo de replicação otimista é mais conveniente.

No intuito de tornar o sistema menos suscetível a erros decorrentes das operações simultâneas a réplicas concorrentes e de minimizar a complexidade operacional optou-se pelo uso de uma arquitetura cliente/servidor. Coube a unidade servidora gerenciar de forma centralizada o banco de dados fixo, validando conjuntos de transações acumuladas durante o período de desconexão das unidades móveis. Para que as operações remotas possam ser efetivadas no banco de dados, devem ser submetidas ao servidor de sincronização. As ações ocorridas nos clientes são entendidas como temporárias e propensas a invalidações mediante a constatação da existência de dados contraditórios.

Uma característica do modelo otimista é permitir que visões do banco de dados móveis possam ficar desatualizadas. Isso ocorre devido às modificações paralelas sobre réplicas de itens de dados espalhadas por diferentes pontos do ambiente móvel. Uma solução para amenizar este problema é tornar a aplicação cliente capaz de identificar situações que possam comprometer a integridade dos dados. Para tanto, baseando-se na proposta de ARAÚJO, FERREIRA (2000), foram definidas regiões de risco (tituladas como “zonas quentes”) e regiões seguras (“zonas frias”). As “zonas quentes” representam os pontos do banco de dados que apresentam maior probabilidade de geração de conflitos. Tais pontos compõem registros que sofrem freqüentes modificações. Já as “zonas frias” são caracterizadas por englobar registros com valores inalteráveis ou com pouca incidência de atualizações. Quando um usuário apodera-se de dados provenientes de uma região de risco, por exemplo, cada *tupla* recebe uma etiqueta de tempo que auxiliará na identificação da cópia válida no momento da reintegração com o servidor. Já as *tuplas* advindas de regiões seguras não são repassadas para análise no servidor, pois não oferecem risco de gerar problemas consistência.

Para que o modelo otimista pudesse ser aplicado de maneira a não comprometer a consistência geral do sistema de informação buscou-se diminuir a incidência de propagação de réplicas entre os elementos móveis. Para isso foi desenvolvida uma solução, nomeada como assistente de sincronização, com finalidade de gerenciar a disseminação de cópias. Essa aplicação, localizada junto ao servidor, recebe designações dos usuários e marca as *flags* apropriadas que contém vínculos com os registros que atendem as expectativas do usuário. Somente os dados "marcados" são encaminhados ao evento de sincronização subseqüente. Dessa forma, otimiza-se o tempo de transferência de dados, acarretando em um melhor aproveitamento da energia disponível no equipamento portátil e também se consegue uma disponibilidade de dados suficientes para o processamento de consultas sem ultrapassar a capacidade de memória existente.

Com a replicação parcial do banco de dados é possível tornar a unidade cliente mais autônoma. Essa característica é muito importante para dar maior mobilidade ao usuário, que não fica limitado a uma rede com áreas de cobertura pré-estabelecidas. No entanto a autonomia não deve permitir que intervenções de usuários causem posteriores invalidações de transações. A solução encontrada foi adicionar novas funcionalidades às aplicações móveis. Um dos métodos é fazer com que todas as operações incidentes sobre dados replicados sejam atômicas e gerem um arquivo de recuperação. Tal adaptação, oriunda de sistemas distribuídos, aumenta a confiabilidade do sistema, promovendo alternativas para contornar possíveis falhas operacionais. Além disso, tais políticas abstraem do usuário a complexidade do restabelecimento do controle do sistema.

5.6.1. Benefícios obtidos

a) Disponibilidade

A transposição de réplicas completas de *tuplas* do banco de dados para as unidades clientes permitiu que grande parte do processamento de transações pudessem ser realizados sobre os dados locais, sem a necessidade de freqüentes sincronizações com o servidor. Isso acarretou em um bom desempenho operacional, visto que as

respostas aos procedimentos de consultas não ficam sujeitas aos altos índices de latência das redes sem fio.

b) Confiabilidade

Na sincronização do banco de dados deve ser priorizada a segurança de forma a tentar impedir que registros sejam interceptados durante a transferência entre as unidades cliente e servidor. Neste aspecto o emprego do modelo otimista mostrou-se eficaz, pois gera poucas transferências entre os elementos do sistema e possibilita a aplicação de metodologias de propagação de réplicas adequadas para atender as normas de segurança estabelecidas.

c) Gerenciamento de energia

Com a agilidade no atendimento das requisições de usuário é possível obter-se um melhor aproveitamento de energia da unidade portátil. Outra vantagem observada é que não há um consumo de energia adicional relativo a tentativas de conexão, como ocorre em redes sem fio. As operações de sincronização, que são realizadas por cabo ou infravermelho, visam submeter conjuntos de registros suficientes para atender o maior número possível de consultas ao banco de dados, mantendo os procedimentos locais o tanto quanto possível.

d) Baixa necessidade de recursos

Como o controle de concorrência é atribuído ao servidor de sincronização, não obriga as unidades móveis oferecerem alta velocidade de processamento. Em, vez disso, prioriza-se a disponibilidade de memória do dispositivo portátil. O uso do modelo otimista permite que sejam aplicados meios de transmissão de dados convencionais. Como tal pesquisa visa a aplicação prática para o setor público, a solução não deve despender de grandes investimentos para que sua implantação possa se tornar viável.

5.6.2. Limitações identificadas

a) Custos de atualização de dados concorrentes

A liberdade da realização de modificações em réplicas de dados localizadas nas unidades clientes possibilita a ocorrência de situações conflitantes no sistema de banco de dados. Vejamos um exemplo em que diversos usuários atuam sobre os mesmos itens de dados. Nenhum sabe da existência de acessos paralelos. Caso algum deles realize alterações locais, os demais elementos remotos estariam obtendo visões desatualizadas sobre determinados itens de dados. Além disso, poderiam estar sendo feitas diversas atualizações simultâneas nos mesmos registros.

Como na replicação otimista as mudanças são aplicadas diretamente na memória local do cliente móvel todos os conflitos ocasionados por atualizações sobre os mesmos dados, em diferentes nodos, devem ser solucionados durante o período de reintegração com o banco de dados principal. Não existe limite de tempo para a reintegração dos dados, ficando a critério do usuário a decisão de quando suas modificações serão encaminhadas ao banco de dados consolidado, o que pode tornar o banco de dados inconsistente por um longo período de tempo.

No momento da sincronização o servidor deverá identificar a ocorrência de operações conflitantes sobre os itens de dados replicados, aplicar no banco de dados central os dados que exprimem o estado atual do sistema e invalidar as versões contraditórias restantes.

Uma alternativa para solucionar problemas deste tipo é impedindo a existência de cópias de itens de dados entre as unidades clientes. Isso não é viável já que nem todos os usuários farão mudanças nos dados. A melhor forma encontrada foi dar permissões a tipos de usuários distintos, de forma a minimizar a ocorrência de atualizações paralelas. No caso disso ocorrer, são oferecidas duas medidas: advertir o usuário da existência do conflito, possibilitando que ele próprio tome a decisão sobre qual dado é o oficial, ou deixar que o sistema decida, tomando por base o *timestamp*

mais atual. Inclui também a possibilidade de reverter o banco de dados para o estado inicial com a manipulação do *log* de registro.

Para solucionar os problemas descritos anteriormente é preciso que o mecanismo de controle de concorrência tome decisões arbitrárias. No entanto, tende a limitar as condições de consistência, ocasionando um aumento de encargos no lado do servidor, que precisa tomar decisões arbitrárias para solucionar conflitos, podendo anular ou reiniciar transações realizadas nas unidades móveis. Desde que essas inconsistências e conflitos de atualizações permaneçam até a reintegração, o protocolo otimista torna-se interessante. Portanto, é mais indicado para soluções que possuem pequena probabilidade de atualizações nos dados remotos ou não exista obrigatoriedade de justeza dos dados armazenados. Não sendo conveniente em aplicações especificas que possuem regras rígidas de reconciliação.

b) Manipulação de dados inválidos

A execução paralela de transações sobre réplicas de itens de dados em diferentes unidades móveis pode fazer com que usuários manipulem temporariamente dados inválidos. O problema se agrava quando existem muitas mudanças nesses dados e as sincronizações são esporádicas. Em ambientes móveis com comunicação contínua, por exemplo, as cópias de dados que sofreram modificações são atualizadas imediatamente por meio de difusão de mensagens para todos os membros da rede. Porém, em sistemas com comunicação descontinuada não é possível encaminhar imediatamente atualizações para uma unidade cliente para manter seus registros constantemente atualizados. Um cliente só deterá valores atualizados mediante uma sincronização solicitada explicitamente pelo usuário.

Para evitar a ocorrência de dados inválidos optou-se por bloquear alterações em certos tipos de dados, definidos como do tipo “somente leitura”. Esses dados não têm restrição quanto ao número de réplicas. Para fazer uma mudança é preciso que seja alterada a permissão no servidor. Os demais dados não podem conter mais que uma réplica por vez. Em contrapartida, essa medida diminui a autonomia das unidades móveis e pode tornar os dados bloqueados por um período de tempo indeterminado.

5.7. Atualizações de cópias

Em termos de atualizações de cópias, existem duas possibilidades de gerenciamento: centralizar as atualizações através de uma cópia primária ou aplicar um plano distribuído. Na primeira alternativa, as transações ocorridas nos clientes são encaminhadas para o servidor, que possui a versão referencial do banco de dados. Todas as atualizações são aplicadas primeiramente na cópia principal e, somente após isso acontecer, são propagadas para as demais réplicas existentes. Já na segunda possibilidade, é permitida a existência de múltiplas cópias de itens de dados coerentes em um determinado instante de tempo. As atualizações podem ser propagadas a partir da unidade cliente que realizou alterações em sua réplica local, que detém momentaneamente a cópia válida de parte banco de dados, disseminando determinadas mudanças para as demais cópias distribuídas no ambiente móvel.

Neste estudo foi aplicado o gerenciamento de atualizações por cópia primária para definir a versão atual do banco de dados. A escolha justifica-se pelo fato que as unidades clientes manipulam suas cópias de dados locais de forma desconectada por longos períodos de tempo, não tendo, então, autonomia para interagir com outros elementos da rede. O servidor detém o controle sobre o banco de dados principal, monitorando as réplicas de dados existentes, executando procedimentos de controle de concorrência durante o procedimento de sincronização com uma unidade móvel. Algumas técnicas foram empregadas no sentido de aumentar o desempenho do sistema e diminuir a possibilidade de erros de transmissão, descritas a seguir.

5.7.1. Redução o overhead de mensagens

A serialização por cópia primária evita atualizações simultâneas em réplicas diferentes e simplifica o controle de concorrência, mas também introduz um “gargalo” potencial em um único ponto. Para reduzir tal problema de *overhead* resultante do considerável número de mensagens por transação, as operações de atualização foram empacotadas em uma simples mensagem com Esquema Otimista e Replicação Lazy. Assumindo que muitas ações executadas por usuários móveis têm quantidade significante de operações de escrita, optou-se pelo uso da solução Cliente/Servidor. As

operações *commit* sobre cópias de dados foram adiadas até a ocorrência de uma solicitação de sincronização. Desta forma, o conjunto de operações sobre as réplicas de itens de dados, realizadas durante um período de desconexão, são reintegradas de forma conjunta, evitando a comunicação contínua entre os elementos, o que diminui expressivamente o volume de mensagens submetidas ao servidor.

5.7.2. Eliminação de *deadlocks*

Em sistemas de bancos de dados distribuídos que operam sobre diversas réplicas de itens de dados, a probabilidade de completa paralisação é diretamente proporcional ao número de cópias existentes. Da mesma forma ocorre em sistemas móveis. Um modo para evitar a ocorrência de *deadlocks* é pré-ordenar transações; com isso é possível assegurar que todas as mensagens sejam tratadas na mesma ordem no servidor. Na ocorrência de alguma inconsistência no banco de dados, as operações de uma transação em uma única mensagem são canceladas. A ordem total na entrega de transações não implica uma execução consecutiva, dessa forma, operações não-contraditórias podem ser executadas em paralelo.

5.7.3. Otimizações com diferentes níveis de isolamento

Freqüentemente bancos de dados comerciais usam critérios de justeza que permitem diferentes níveis de isolamento para transações, como por exemplo, *read committed* e isolamento serializável, aplicados pelo PostgreSQL. Tais níveis são utilizados na tentativa de aumentar o desempenho, maximizando o grau de concorrência e reduzindo o percentual de conflitos entre transações.

As exigências de isolamento sobre a execução de transações sobre dados, pertinentes em ambientes distribuídos, podem ser adaptadas para sistemas móveis. O problema principal é quando são aplicados em cópias de dados que operam em modo de desconexão. Como as efetivações das transações tendem a acontecer de modo tardio, conduzem a taxas de conflito mais altas. Quando uma transação processa sob o nível de isolamento *read committed*, operações de consulta retornam apenas os dados efetivados antes da consulta começar; nunca “enxerga” dados não efetivados, ou as mudanças

ocorridas durante a execução da consulta pelas transações concorrentes. Isso significa que sincronizações sucessivas podem submeter dados diferentes às unidades clientes. Em contrapartida, o isolamento serializável mostra-se mais apropriado. Este opção emula a execução serial da transação, como se todas as transações fossem executadas em série, ao invés de concorrentemente. Quando uma transação está no nível serializável, as operações de consulta são aplicadas sobre os dados efetivados antes da transação começar; com isso, não ficam visíveis os dados não efetivados ou mudanças efetivadas durante a execução da transação por transações concorrentes. Transações "enxergam" os efeitos de atualizações anteriores, executadas dentro de sua própria transação, mesmo que ainda não tenham sido efetivadas. Portanto, sincronizações sucessivas tendem a reproduzir dados semelhantes, mesmo sendo executadas no subseqüentemente.

5.7.4. Log de transações

Ao serem realizadas alterações ou exclusões de registros no *host* móvel, uma estrutura de recuperação por *log* de registro pode ser constituída. A técnica, aplicada no término de um procedimento de sincronização, permite o restabelecimento do banco de dados na identificação de inconsistências.

5.7.5. Transações atômicas

As transações decorrentes de uma solicitação de sincronização normalmente geram mudanças significativas na base de dados fixa. Estas alterações devem ocorrer de forma completa para não gerar inconsistências no banco de dados. Para tanto, é fundamental que tais procedimentos sejam atômicos. Na circunstância de um erro de conexão, por exemplo, transações parciais não devem ser efetivadas. Nesse caso seria necessária uma nova tentativa de transferência do cliente para o servidor.

5.8. O servidor de sincronização

O servidor de sincronização é um computador *desktop* composto por recursos de *software* que combinados gerenciam a transferência e o armazenamento dos dados do sistema distribuído. Monitora as requisições externas e garante a consistência e a integridade dos registros armazenados nos bancos de dados, em decorrência das reconciliações de dados provenientes dos clientes móveis.

Durante um procedimento de sincronização, o servidor identifica e corrige possíveis situações de conflito nos dados modificados pelos usuários móveis, atuando tanto na base consolidada quanto nos itens de dados replicados em um dispositivo móvel.

5.8.1. Protótipo de sincronização

Para que o servidor de sincronização possa atuar é preciso que exista um intercâmbio entre ele e os demais elementos do sistema. Com este objetivo, desenvolveram-se diferentes módulos (representados na Fig. 25) que, associados com soluções proprietárias, possibilitam a execução de procedimentos de sincronização na base de dados consolidada e nas réplicas de itens de dados.

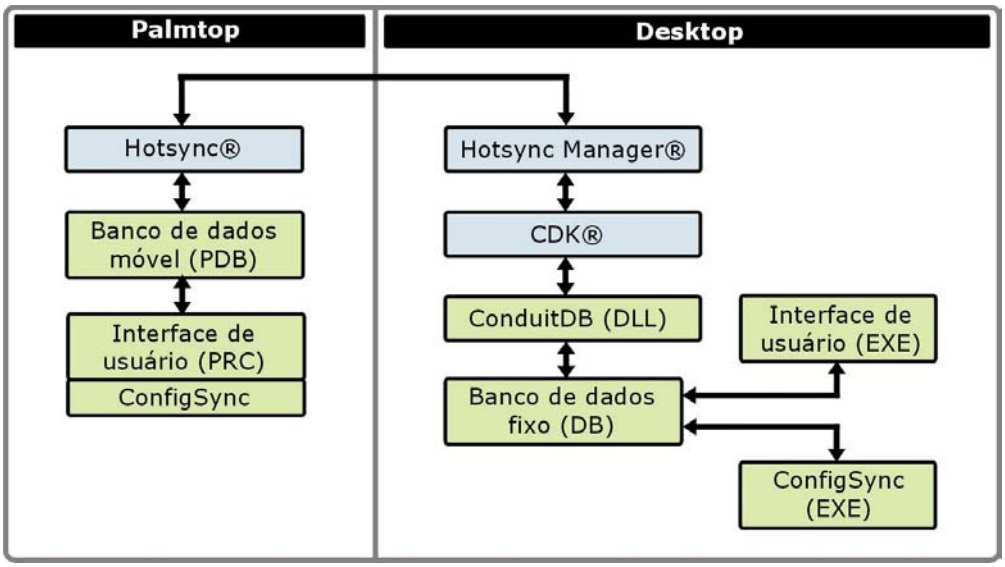

FIGURA 25 – Protótipo de sincronização.

Um evento de sincronização é iniciado a partir de um cliente móvel, que submete, através do Hotsync®, requisições ao servidor. O Hotsync® é uma aplicação, desenvolvida pela PalmSource™, que atua no lado do cliente, responsável por acionar o processo de sincronização. Ao ser executada, ativa o protocolo de comunicação escolhido pelo usuário no dispositivo móvel, que realiza a chamada de serviços no *desktop*, o que desencadeia o procedimento de troca de dados com o HotSync Manager®, responsável por gerenciar as requisições locais e remotas, conforme descrito nos *conduits*.

Uma operação de HotSync® é uma sincronização bidirecional de registros entre a unidade móvel e a unidade fixa. As alterações feitas na unidade móvel são atualizadas nas duas plataformas após uma operação de HotSync. Também podem ser instalados aplicativos no sentido servidor/cliente ou em um cartão de expansão compatível com Palm OS® durante este tipo de operação.

O Hotsync Manager® é uma aplicação que atua em *background* no lado do servidor inspecionando uma ou mais portas de comunicação para localizar ocorrências de chamadas de sincronização feitas pelos clientes móveis. Quando recebe algum sinal, aciona os *conduits* instalados e configura os usuários autorizados para realizar a sincronização dos dados. Cada *conduit* desempenha suas próprias operações de sincronização, gerenciando a sincronização de bancos de dados específicos.

O módulo *plug-in* do tipo *conduit* desenvolvido para este experimento tem a finalidade de analisar e solucionar conflitos de sincronização ocasionados pela reconciliação de dados modificados entre o banco de dados cliente e o banco de dados consolidado. O sistema ainda inclui aplicações tanto no lado do cliente quanto no servidor, que atuam como *interface* para acesso e manipulação dos dados e para o controle das configurações de usuários e métodos.

O Hotsync Manager® provê uma *interface* de comunicação para *conduits* que atuam sobre um *handheld* chamada Sync Manager API®. Através dela, é possível que a atuação do *conduit* permaneça independente do tipo de conexão entre o equipamento móvel e o servidor. Além disso, são fornecidos *notifiers*. Quando uma aplicação *desktop*

e seu correspondente *conduit* podem modificar dados de usuários, o HotSync Manager® usa notificações para assegurar que ambas não estão alterando-os ao mesmo tempo. Uma notificação é acionada através de uma mensagem da aplicação localizada no *desktop*, baseando-se em DLLs (*Dinamic Link Librarys*) do Windows, invocadas pelo HotSync Manager® durante o procedimento de sincronização. A Fig. 26 mostra o fluxo de processo destes componentes.

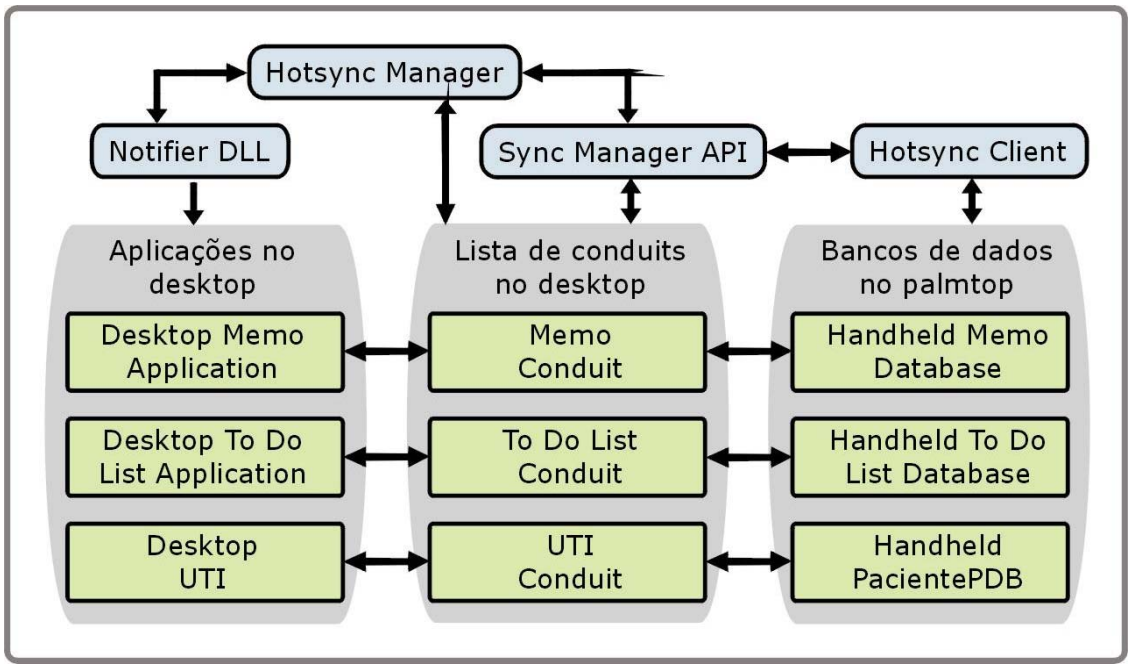

FIGURA 26 – Fluxo do processo de sincronização.

5.8.3. Módulo *conduit*

Conduits são Bibliotecas de Vínculo Dinâmico (DLL) administradas pelo HotSync Manager® que têm a capacidade de importar, exportar e sincronizar dados entre um computador *desktop* e uma unidade móvel com plataforma PalmOS. Podem acessar bancos de dados e executar procedimentos de consulta e atualizações sobre registros. Cada aplicação que mantém dados replicados remotamente contém um *conduit* que efetua o tipo de transferência de dados mais apropriado, que pode ser *one-way* ou *two-way* (ver Capítulo 4).

Tipos de *conduits*

- ***Conduit* de instalação**: tem como finalidade a instalação de aplicações para PalmOS e bancos de dados na memória principal ou em cartões de expansão de dispositivos móveis. No Windows, o HotSync Manager® executa o *conduit* de instalação durante um evento de sincronização. Isso faz com que instalações provenientes do servidor sejam executadas no *host* móvel independente da solicitação do usuário móvel.

- ***Conduit* de *backup***: copia para o *desktop* quaisquer itens de bancos de dados ou dados de aplicações que sejam configuradas para replicar informações armazenadas na unidade portátil.

- ***Conduit* de sincronização**: tipicamente sincroniza bancos de dados móveis com suas contra-partes em banco de dados distribuídos, e vice-versa. As funcionalidades são descritas pelo programador, que pode adotar heurísticas complexas, aumentando a necessidade de longas conexões.

Para o experimento descrito neste trabalho foi desenvolvido um *conduit* de sincronização, titulado ConduitUTI, onde foram aplicados métodos para tratamento de réplicas de dados, incluindo o modelo otimista e o híbrido, que favorecem a disponibilidade de dados ao usuário e a permanência de longos períodos de desconexão com a base fixa. Este *plug-in* foi projetado para executar diversos tipos de operações sobre dados locais e remotos de forma a otimizar o processo de troca de dados entre os dispositivos envolvidos na sincronização, incluindo mecanismos para transferência mínima de dados e a notificação de conflitos decorrentes de modificações contraditórias entre as cópias de dados. Além disso, não requer interação do usuário em qualquer operação de sincronização.

5.8.4. O ConduitUTI

O ConduitUTI foi construído através da linguagem de programação Borland Delphi© associada à solução TurboSync (TABDEE, 2003), que é um conjunto de componentes VCL e classes para a linguagem Delphi que permite o acesso a bancos de dados para PalmOS.

Para inserir o *conduit* desenvolvido à lista de programas de sincronização do Hotsync Manager® foi utilizado o CDK® - *Conduit Development Kit*, que é um conjunto de funções específicas para a criação de aplicações *conduit* através das Linguagens de Programação Microsoft Visual C++ 6.x, Visual Basic 6.0 e WebGain VisualCafe 4.0/4.0a. Inclue suporte para Java© JRE 1.3 e COM[10] (*Component Object Model*). Segundo WILSON (2002), através do CDK é possível inserir e remover programas de sincronização da lista de execução e vinculá-los as respectivas aplicações móveis (com a especificação do *CreatorID*).

5.8.5. Replicação e reconciliação de dados

Inicialmente a unidade cliente contém um banco de dados vazio; é preciso que o usuário solicite explicitamente na aplicação cliente a transferência de dados a partir da *interface* de configuração de sincronização. Com o estabelecimento da conexão com o servidor são executados os *conduits* inscritos no Hotsync Manager® de forma seqüencial. Ao receber requisições externas, o *conduit* desenvolvido para este experimento identifica (através de *flags* no banco de dados consolidado) os dados que devem ser enviados para a unidade cliente. A partir daí o banco de dados da unidade móvel é preenchido por réplicas parciais de registros originados do banco de dados fixo (Fig. 27).

[10] Tecnologia de comunicação da Microsoft extensamente usada para comunicação e interoperabilidade entre aplicações e fontes de dados externas.

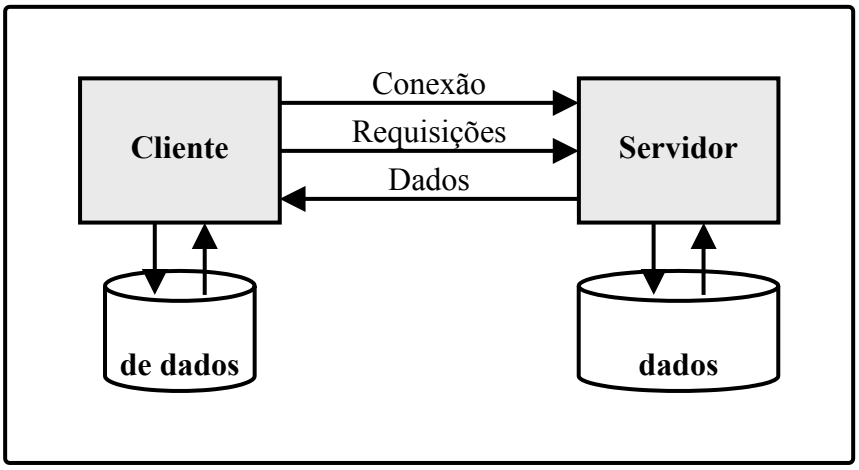

FIGURA 27 – Replicação de dados pelo *conduit*.

Com a existência de réplicas de itens de dados na unidade cliente, o usuário pode fazer consultas e modificações através da *interface* de acesso. Após isso, independente do tempo percorrido, o usuário pode solicitar nova sincronização com o servidor. Nessa nova conexão, o *conduit* identifica a existência de registros modificados na unidade cliente. As modificações são então transferidas para o servidor, que analisa a existência de conflitos e submete as devidas correções para serem introduzidas na unidade cliente, conforme representado na Fig. 28. Alternativamente, o agente de sincronização pode ser configurado para efetivar as modificações no banco de dados fixo e, após isso, eliminar todos os registros do banco de dados cliente, liberando a memória do dispositivo móvel para o armazenamento de novos registros e evitando problemas de incoerência de *cache*.

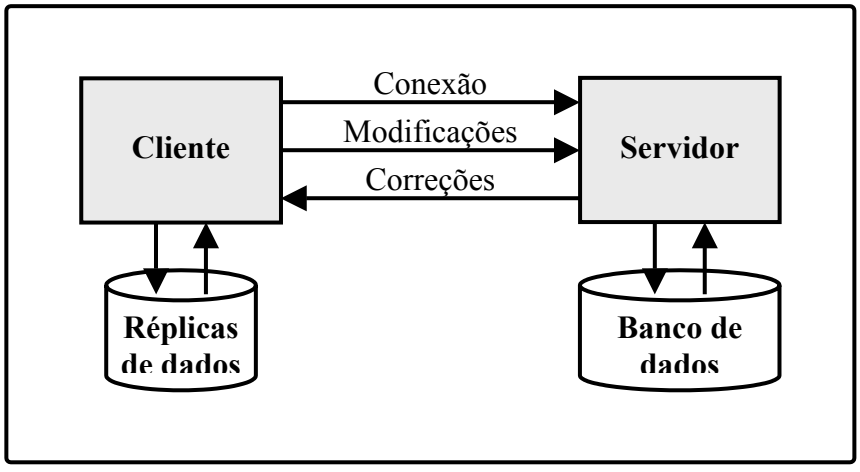

FIGURA 28 – Reconciliação de dados pelo *conduit*.

A comunicação entre os componentes analisados baseou-se em rede do tipo *wireless*, com transmissão de dados por infravermelho e rede ponto-a-ponto, com conexão por cabo serial. O que permite a difusão de mensagens entre todos os componentes do sistema. Isso inclui a possibilidade das unidades clientes trocarem dados entre si, para suprirem necessidades de transações móveis, decorrente da incapacidade de comunicação com o servidor.

5.9. Transferência de dados

O modelo de transferência de atualização determina para onde uma atualização sobre dados (transação) pode ser emitida e como é propagada. Foram levados em consideração os seguintes aspectos: a detenção do controle na emissão de atualizações entre os elementos do sistema, o tipo de conteúdo que é transferido na reconciliação das réplicas de dados e a forma com que são propagadas as modificações ocorridas remotamente.

5.9.1. Controle centralizado de atualizações

Neste experimento foi adotado um esquema com único mestre, designado como servidor de sincronização, que armazena a cópia principal de um objeto. Todas as atualizações são aceitas conforme o banco de dados principal, sendo então propagadas para as réplicas existentes no ambiente móvel. A vantagem principal de sistemas deste tipo é a simplicidade. A descoberta e a solução de conflitos emprega o mínimo de comunicação, já que existe um único ponto de referência. A desvantagem principal é o gerenciamento direcionado a um único ponto, que pode falhar, comprometendo todo o funcionamento do sistema.

Conforme SAITO (2001b), em sistemas com múltiplos mestres, quaisquer réplicas podem emitir atualizações a qualquer momento, disseminando dados entre os *hosts* móveis, que transferem dados de forma assíncrona. A complexidade é compartilhada entre os membros que operam sobre réplicas de dados. Essa alternativa

não foi aplicada no experimento, pois tende a gerar várias versões de dados, ditas "atualizadas", espalhadas no sistema, dificultando a processamento de consultas e diminuindo a confiabilidade do banco de dados. Além disso, necessita de intensa comunicação entre os elementos, aumentando a latência de rede e ocasionando perda de energia na tentativa de freqüentes re-conexões. A vantagem é que cada nodo da rede tem a autonomia para solucionar conflitos e disseminar as modificações para aqueles que possuem cópias do mesmo item de dado.

5.9.2. O Conteúdo da reconciliação

As mudanças ocorridas em um objeto podem ser expressas pelo seu conteúdo atual ou por meio de um registro (*log*). No primeiro caso, o conteúdo do banco de dados modificado por uma unidade cliente é transferido integralmente. No caso da transferência de *log*, somente a definição da operação é propagada (representada por comandos do tipo SQL).

O método por transferência de *log* permite gerenciar conflitos com flexibilidade, especialmente em sistemas com múltiplos mestres. Reduz o *overhead* computacional e da rede, tendo em vista que, em muitos casos, um objeto contém uma extensa quantidade de valores que sofrem mínimas alterações. Por outro lado, tende a ser mais complexo que o método por transferência de dados, devido a três razões. Primeira, cada réplica precisa conter um histórico das atualizações realizadas. Dessa forma, o *site* não precisa apenas de uma estrutura complexa para manter localmente o registro, mas também de um algoritmo para monitorar novas entradas de *log*. Segunda, para manter a consistência da réplica, o sistema tem que determinar o conjunto de atualizações a ser enviado para outras réplicas e a ordem que estas atualizações serão aplicadas. Terceira, deve ser implementado de maneira transparente, de forma a separar a semântica da aplicação, independentemente do algoritmo de replicação, possibilitando sua reusabilidade.

Dentre os métodos de transferência de modificações de dados apresentados anteriormente, foi aplicado no desenvolvimento deste sistema a alternativa de transmissão por conteúdo do banco de dados modificado. Quando a unidade cliente

solicita a reconciliação dos dados replicados, o servidor de sincronização testa o cabeçalho (*Header*) de cada cópia para identificar a ocorrência de alguma alteração desde a criação da réplica. Caso sejam localizadas atualizações, somente os itens de dados que foram modificados ou inseridos são submetidos para o banco de dados consolidado. Os dados que não sofreram alterações são conservados na memória da unidade móvel até o término da operação de sincronização ou até a solicitação de exclusão comandada pelo usuário.

5.9.3. Propagação de atualizações

A escolha de um modelo de propagação de dados determina de que forma as atualizações são disseminadas no ambiente de computação móvel (LANHAM, 2002). Algoritmos *pull-based*, por exemplo, determinam que cada cliente tem o controle da reintegração de dados atualizados com o servidor. No caso do modelo *push-based*, o controle é dado ao servidor, que deve ser capaz de enviar dados aos clientes sem serem consultados e ainda incorporar alguma forma de perfil dos usuários para decidir o que e quando transmitir.

A estratégia de disseminação aplicada foi o modelo *pull-based.* Neste caso, fica a critério do cliente móvel dar início ao procedimento de reconciliação com o banco de dados consolidado através do envio de uma requisição explícita ao servidor de sincronização. Observou-se que o emprego desta metodologia evita o *overhead* de mensagens entre os elementos de rede, visto que normalmente os dados são propagados em lotes, após a realização de diversas atualizações. Além disso, favorece a utilização de diferentes alternativas de comunicação, como por exemplo o infravermelho, já que não é obrigatória a continuidade do canal de comunicação. A desvantagem na utilização de *pull-based* é que as operações de consulta ficam limitadas aos dados armazenados na memória do cliente. Com isso, as transações ocorridas nas unidades clientes não podem ser consideradas confiáveis, já que os dados podem estar desatualizados com relação à base de dados consolidada. Sendo assim, uma transação remota é considerada temporária e não definitiva. Será somente efetivada após a inserção no banco de dados consolidado em decorrência da reconciliação.

No caso do modelo *push-based*, um conjunto de atualizações é difundido entre os elementos da rede por intermédio de *polling*. Esta propriedade torna o mecanismo *push-based* mais adequado para ambientes que operam com *Internet* (GU, 2003). Outra característica é que oferece uma redução no tempo de propagação de atualizações, especialmente em redes completamente conectadas, disseminando dados imediatamente após serem modificados. Por outro lado, exige o emprego de algum meio de transmissão que permita a localização imediata da unidade móvel, normalmente composto por zonas de cobertura. Isso restringe a mobilidade do usuário, delimitando o espaço de funcionamento com base em sinais de rádio. Além disso, essa alternativa diminui a autonomia das unidades clientes, que dependem da iniciativa do servidor para manter a consistência dos dados.

5.9.4. Comunicação por infravermelho

A transferência de dados por infravermelho foi implementada de acordo com os protocolos e padrões da IrDA. Esses padrões são projetados para aceitar componentes de baixo custo e reduzir a demanda de energia. A IrDA é uma tecnologia *half-duplex* de transferência de dados de curta distância. Os protocolos IrDA especificam os procedimentos que oferecem suporte para o estabelecimento do vínculo, a descoberta de endereços de dispositivos, a negociação da inicialização da conexão e da taxa de dados, a troca de informações, o desligamento do vínculo e a resolução de conflitos de endereços de dispositivos.

Atualmente há transceptores[11] de infravermelho instalados em quase todos os novos computadores portáteis, como no caso dos *palmtops* testados no experimento. Nestes casos, a funcionalidade de infravermelho é fornecida através do recurso de transferência de arquivos por conexão sem fio (*wireless*). O PalmOS usa uma *interface* própria (Hotsync®) para fornecer conexões de infravermelho para impressoras, *modens*, *pagers* digitais, assistentes digitais pessoais, câmeras eletrônicas, organizadores, telefones celulares e computadores de modo geral. A implementação IrDA Serial (SIR) de IrDA possibilita uma velocidade máxima de transferência de dados de 115,2 Kbps.

[11] Transceptor: dispositivo que pode transmitir e receber sinais. Em redes locais (LANs), um transceptor é o elemento que conecta um computador à rede e converte sinais entre formas seriais e paralelas.

Em alguns casos, computadores não incluem transceptor de infravermelho interno, como o ocorre no servidor de sincronização utilizado neste experimento. Nesse caso foi preciso instalar um transceptor de infravermelho externo. Foi utilizado um adaptador que captura sinais de infravermelho e retransmite para a porta de comunicação USB (*Universal Serial Bus*) do *desktop* a uma taxa de transferência máxima de 2 Mbps.

No vínculo entre dispositivos com infravermelho todas as transmissões partem do dispositivo principal (de comando) para o dispositivo secundário (de recebimento). O papel principal é determinado dinamicamente quando se estabelece o vínculo e continua até que a conexão seja fechada. Uma das unidades móveis cria um vínculo detectando o servidor ou outra unidade móvel através de uma solicitação do usuário. Esse elemento detém o comando, enviando uma solicitação de conexão a 9.600 bps para o outro dispositivo (incluindo informações como endereço, taxa de dados e outros recursos). O dispositivo de resposta assume o papel secundário e retorna informações que contêm seu endereço e seus recursos. A estação principal e a secundária alteram a taxa de dados e os parâmetros do vínculo passando a utilizar a configuração comum definida pela transferência inicial de informações. Finalmente, a estação principal envia dados para a estação secundária confirmando a conexão. Os dispositivos são conectados e iniciam a transferência de dados sob o controle do dispositivo principal.

5.9.5. Comunicação por cabo USB

Outra implementação de transferência de dados testada foi por cabo, feita através de um conector universal do tipo *cradle*[12], que é conectado a uma porta USB (*Universal Serial Bus*) do computador *desktop*. Com essa conexão direta é possível atingir uma velocidade de transmissão entre 2400 Mbps à 57600 Mbps. Além de servir como meio de transmissão de dados, também funciona como uma fonte para recarga da bateria do dispositivo móvel.

[12] Suporte em que um *palmtop* é afixado para realizar a sincronização e o carregamento da bateria. Pode conectado ao computador *desktop* por um cabo serial convencional ou USB.

6. CONCLUSÕES E TRABALHOS FUTUROS

Este trabalho envolveu o estudo e a aplicação de estratégias de replicação e reconciliação de dados que permitem a um usuário ter acesso a uma grande quantidade de informações em um computador portátil e ainda poder fazer modificações em registros sem a necessidade de conexão por cabos e sem limitações quanto à mobilidade física.

Com o emprego de métodos diferenciados de replicação e reconciliação de dados foi possível obter as seguintes conclusões:

- A geração de réplicas parciais otimizou a transmissão e o armazenamento de dados por não serem sincronizados todos os registros do banco de dados para a unidade móvel.

- O mecanismo de *caching* tornou os clientes menos dependentes do servidor e com maior desempenho devido à disponibilidade de registros na memória.

- A independência de conexão permitiu ampla mobilidade aos usuários móveis, uma vez que não existe necessidade da aproximação contínua com um ponto de acesso.

- O controle de transações na unidade móvel favoreceu o balanceamento de carga de trabalho no sistema, não sobrecarregando o servidor com transações sobre o banco de dados consolidado.

- A transmissão moderada de modificações entre os clientes e o servidor otimizou o uso do canal de comunicação.

- O emprego de replicação assíncrona equacionou os acessos ao servidor.

- A sincronização *two-way* proporcionou a organização da disseminação de cópias de dados, pois permitiu um controle abrangente sobre os dados existentes tanto no servidor quanto nas unidades móveis.

- Os meios de comunicação adotados (infravermelho e cabo USB) deram segurança na transmissão de dados, dificultando a interceptação durante o período de sincronização do banco de dados.

- Atualizações paralelas podem influenciar temporariamente na consistência geral do sistema, pois não é possível monitorar as réplicas de itens de dados constantemente.

- O controle centralizado da sincronização é um ponto de vulnerabilidade no sistema. Com a sobrecarga de solicitações externas o servidor pode tornar-se indisponível por tempo indeterminado. A solução para situações deste tipo é a utilização de unidades de armazenamento redundantes.

A análise de metodologias diferenciadas para gerenciamento de réplicas de dados permitiu concluir que é possível se obter um nível satisfatório de funcionalidades com a combinação de estratégias de replicação e reconciliação diferenciadas conforme o tipo de solução pretendida. Para isso, alguns critérios fundamentais devem ser observados:

- **O controle da geração de réplicas**: é importante uma definição no que tange o controle do lançamento de linhas de dados na memória de um dispositivo móvel. É conveniente conceder o controle ao usuário quando não existe a possibilidade de conexão instantânea. Na situação inversa, o controle passa para a aplicação. Nesse caso tanto o cliente quanto o servidor (ou os dois) podem assumir tal comando.

- **O grau de replicação atribuído**: o mais conveniente é aplicar a replicação parcial, já que tal método dá autonomia à unidade cliente e aumenta o desempenho com a disponibilidade de registros na memória das estações remotas. As alternativas possíveis geram muitos problemas que podem tornar inviáveis suas aplicações práticas.

- **Quando gerar réplicas**: no intuito de aumentar a disponibilidade de informações na unidade cliente e otimizar a velocidade de resposta às requisições de usuários a despeito da ocorrência de desconexões é interessante ser feito o armazenamento antecipado de dados. A geração de consultas freqüentes pode degradar o desempenho e aumentar a latência das transações.

- **O tipo adequado de replicação**: no tipo assíncrono as alterações realizadas em cópias de dados são propagadas em um segundo passo. Essa escolha é a mais apropriada na maioria das aplicações, pois ocasiona em uma diminuição do tempo de conexão durante uma transferência de dados entre o servidor e um cliente.

- **O modelo de replicação**: não existe um modelo que possa ser considerado o ideal. Depende do problema que se quer resolver. O modelo cliente/servidor foi aplicado no experimento deste trabalho por fornecer uma infra-estrutura adequada ao tipo de desafio a ser implementado.

- **O protocolo de replicação**: dentre os protocolos descritos neste trabalho, o modelo otimista trás mais vantagens do que a contraparte, o modelo pessimista. Destacam-se os seguintes benefícios: tolerância à falhas, otimização quanto à transmissão de dados, disponibilidade de informações e mobilidade. Porém, diminui a confiabilidade das operações por não garantir a coerência do banco de dados.

- **O modelo de transferência de dados**: o modelo baseado em mensagens é mais adequado quando não existe uma conexão contínua entre clientes e o servidor. Já os modelos baseados em sessão e conexão destinam-se a casos particulares de comunicação ininterrupta.

- **O controle da reconciliação**: pela maior facilidade de implementação e gerenciamento do banco de dados indica-se o controle centralizado, já o controle distribuído demanda de uma complexa interação entre as unidades clientes.

- **A ordem da propagação de réplicas**: a organização plana representa a organização mais simples para transmissão de dados. Após o servidor reunir as informações necessárias, elas são enviadas repetidamente até serem capturadas pelos clientes. As demais alternativas somente são aplicadas em ocasiões onde exista um volume intenso de transmissão de dados.

- **A freqüência de disseminação de dados**: a propagação condicional favorece o acúmulo de modificações locais na ocorrência de desconexão, porém pode influenciar no estado geral do banco de dados. A propagação periódica determina a constante manutenção da consistência das réplicas distribuídas em um sistema móvel. Nesse caso a escolha dependerá da existência de modificações paralelas entre as réplicas de dados. A quantidade de mensagens necessárias para atender solicitações de usuários deve ser mínima a fim de evitar sobrecargas na rede e preservar a energia dos dispositivos portáteis.

Como trabalhos de futuros propõe-se a aplicação de comunicação contínua entre as unidades do sistema, que permitiria a atualização contínua das réplicas de itens de dados no cliente móvel; a implementação de um controle distribuído de sincronização de dados, dando autonomia de gerenciamento para todas as unidades móveis em decorrência da perda de conexão ou indisponibilidade do servidor; a fragmentação do banco de dados conforme a localização do usuário, visando um aumento na eficiência na execução de transações e transmissão de dados; e o desenvolvimento de um protocolo de transferência de dados interoperável, tornando o sistema apto a operar com diferentes sistemas operacionais.

REFERÊNCIAS

ACHARYA, S.; FRANKLIN, M.; ZDONIK; S. **Balancing Push and Pull for Data BroadCast**. ACM SIGMOD International Conference on Management of Data and Symposium on Principles of Database Systems, p.1983-194, 1997.

AGRAWAL, D.; ABBADI, A. El; STEINKE, R. **Epidemic Algorithms in Replicated Databases**. ACM Symposium on Principles of Database Systems, p.161-172, 1997.

AHUJA, R.; BAGRODIA, R.; BAJAJ L.; et al. **Evaluation of Optimistic File Replication in Wireless Multihop Networks**. GLOBECOM'99, p.2-4, 1999.

AKSOY, D.; ALTINEL, M.; BOSE, R.; et al. **Research in Data Broadcast and Dissemination**. 1st International Conference on Advanced Multimedia Content Processing, p.1-15, 1998.

ANDERSON, T.; BREITBART, Y.; KORTH, H.; et al.. **Replication, Consistency and Practicality: Are These Mutually Exclusive?** ACM SIGMOD International Conference on Management of Data, p.484-495, 1998.

ARAÚJO, L.; FERREIRA, J. **Cache Semântico para Computação Sem Fio Baseado na Abstração de Composição dos Dados**. Workshop de Sistemas de Informação Distribuída de Agentes Móveis. São Paulo, p.83-89, 2000.

BADRINATH, B. R.; PHATAK, Shirish. H. **An Architecture for Mobile Databases**. Research Work DCS-TR-351 , 1998.

BARBARÁ, D. **Mobile Computing and Database: A Survey**. IEEE Transactions on Knowledge and Data Engineering, p.1-14, 1999.

BATES, R.; GREGORY, D. **Voice and Data Communications Handbook**. McGraw-Hill Series on Computer Communications, 1997.

BREITBART, Y.; KOMONDOOR, R.; RASTOGI, R.; et al. **Update Propagation Protocols for Replicated Databases**. ACM SIGMOD International Conference on Management of Data, p.97-108, 1999.

BUSZKO, D.; LEE, W.; HELAL, A. **Decentralized Ad-Hoc Groupware API and Framework for Mobile Collaboration**. ACM International Conference on Supporting Group Work, 2001.

CANO, J.C.; MANZONI; P. **A Performance Comparison of Energy Consumption for Mobile Ad Hoc Network Routing Protocols**. IEEE/ACM Eighth International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems, p.57-63, 2000.

COGLIANESE, M. **Optimistic Data Replication for Mobile Applications**. International Conference on Mobile Data Access, p.1-8, 2000.

ÇETINTEMEL, U.; KELEHER, P.; FRANKLIN, M. **Support for Speculative Update Propagation and Mobility in Deno**. 21st IEEE International Conference on Distributed Computing Systems (ICDCS'01), 2001.

ÇETINTEMEL, U.; KELEHER, P. **Performance of Mobile, Single-Object, Replication Protocols**. 19th IEEE Symposium on Reliable Distributed Systems (SRDS'00), p.2-8, 2000.

DANTAS, M. **Tecnologias de Redes de Comunicação e Computadores**. Rio de Janeiro: Axcel Books, 2002.

DESAI, N.; VERMA, V.; HELAL, S. **Infrastructure for Peer-to-Peer Applications in Ad-Hoc Networks**. 2nd International Workshop on Peer-to-Peer Systems, 2003.

FELBER, P.; SCHIPER, A. **Optimistic Active Replication**. 21st International Conference on Distributed Computing Systems, p.2-9, 2001.

FLINN, J.; SATYANARAYANAN, M. **Energy-aware adaptation for mobile applications**. 17th ACM Symposium on Operating Systems Principles, p.48-63, 1999.

FUKUSHIMA, T.; TAKAHASHI, E.; NARAZAKI, H.; et al. **A "Wireless Agent" for automatic connection based on connection time prediction using RF information**. 5th World Multi-Conference on Systemics, Cybernetics and Informatics, p.1-6, 2001.

GRAY, J.; HELLAND, P.; O'NEIL, P.; et al. **The dangers of Replication and a Solution**. SIGMOD International Conference on Management of Data, p.173-182, 1996.

GU, W.; HELAL, A. **Extended Internet Caching Protocol: A Foundation for Building Ubiquitous Web Caching**. ACM Symposium on Applied Computing, 2003.

HELAL, A.; HEDDAYA, A.; BHARGAVA, B.. **Replication Techniques in Distributed Systems**. Kluwer Academic Publishers, 1996.

HOLLIDAY, J.; STEINKE, R.; AGRAWAL, D.; et al. **Epidemic Algorithms for Replicated Databases**. IEEE Transactions On Knowledge And Data Engineering, p.2-17, 2003.

HOLLIDAY, J.; STEINKE, R.; AGRAWAL, D.; et al. **Epidemic Quorums for Managing Replicated Data**. 19th IEEE Int'l Performance, Computing and Comm. Conf. - IPCCC 2000, p.93-100, 2000.

HUANG, Y.; SISTLA, P.; WOLFSON, O. **Data Replication for Mobile Computers**. SIGMOD Conference, p.11-13, 1994.

ITO, G. **Bancos de Dados Móveis: Uma Análise de Soluções Propostas para Gerenciamento de Dados**. Dissertação de Mestrado em Ciência da Computação, UFSC, 2001.

JAGADISH, H.; MENDELZON, A.; MUMICK, I. **Managing Conflicts Between Rules**. ACM Symposium on Principles of Database Systems, p.192-201, 1996.

JING, J.; HELAL, A.; ELMAGARMID, A. **Client-Server Computing in Mobile Environments**. ACM Computing Surveys, vol.31, n.02, 1999.

JOSEPH, A.; TAUBER, J.; KAASHOEK, M. **Mobile Computing with the Rover Toolkit**. IEEE Transactions on Computers, 1997.

KARAKAYA, M.; ULUSOY, Ö. **An Efficient Broadcast Scheduling Algorithm for Pull-Based Mobile Environments**. IEEE/ACM Transactions on Networkin, p.1-23, 2001.

KELEHER, P. **Decentralized replicated-object protocols**. 18th ACM Symposium on Principles of Distributed Computing, p.143-151, 1999.

KEMME, B.; ALONSO, G. **A New Approach to Developing and Implementing Eager Database Replication Protocols**. ACM Transactions on Database Systems, p.336-343, 2000.

KHUSHRAJ, A.; HELAL, A.; ZHANG, J.. **InCoda: Incremental Hoarding and Reintegration in Mobile Environments**. IEEE/IPSJ International Symposium on Applications and the Internet, p.2-11, 2002.

KIM, S.; SONY, S.; STANKOVICY, J.; et al. **SAFE: A Data Dissemination Protocol for Periodic Updates in Sensor Networks**. 23rd International Conference on Distributed Computing Systems Workshops, p.19-22, 2003.

KREMER, U.; HICKS, J.; REHG, J. **A Compilation Framework for Power and Energy Management on Mobile Computers**. 14 th International Workshop on Parallel Computing, p.1-12, 2001.

KUENNING, G.; BAGRODIA, R.; GUY, R.; et al. **Measuring the Quality of Service of Optimistic Replication**. 12th European Conference on Object-Oriented Programming, p.1-3, 1998.

LANHAM, M.; KANG, A.; HAMMER, J.; et al. **Format-Independent Change Detection and Propagation in Support of Mobile Computing**. XVII Brazilian Symposium on Databases, 2002.

LEE, S.; HWANG, C.;YU, H. **Supporting Transactional Cache Consistency in Mobile Database Systems**. MobiDE Seattle, p.1-7, 1999.

LEONG, H.; SI, A. **On Adaptive Caching in Mobile Databases**. ACM Symposium on Applied Computing Table of Contents, p.302-309, 1997.

LIU, G.; MARLEVI, A.; MAGUIRE, G. **A Mobile Virtual-distributed System Architecture for Supporting Wireless Mobile Computing and Communications**. 1st International Conference on Mobile Computing and Networking Communications, p.111-118, 1996.

LUBINSKI, A.; HEUER, A. **Configured Replication for Mobile Applications**. Workshop Grundlagen von Datenbanken, p.1-13, 2000.

MADRIA, S. **Timestamps to Detect R-W Conflicts in Mobile Computing**. International Workshop on Mobile Data Access in conjunction with ER'98, p.242-253, 1998.

MANIATIS, P.; ROUSSOPOULOS, M.; SWIERK, E.; et al. **The Mobile People Architecture**. ACM Mobile Computing and Communications Review, 1999.

MASS, B. **Constructor for Palm OS**. Technical Report PalmSource 3007-004, May, 2002.

MATEUS, G.; LOUREIRO, A. **Introdução a Computação Móvel**. Rio de Janeiro: NCE/UFRJ, 11ª Escola de Computação, 1998.

MENKHAUS, G. **Adaptive User Interface Generation in a Mobile Computing Environment**. PhD Thesis, Universität Salzburg, 2002.

MUMMERT, L. **Exploiting Weak Connectivity for Mobile File Access**. PhD. thesis, Carnegie Mellon University, School of Computer Science, Pittsburgh, 1996.

OSTREM, J. **Palm OS User Interface Guidelines**. Technical Report PalmSource 3101-001, January, 2002.

ÖZSU, M.; VALDURIEZ, P. **Principles of Distributed Database Systems**. New Jersey: Prentice Hall, 2ª ed., 1999.

PALAZZO, S.; PULIOFITO, A.; SCARPA, M. **Design and Evaluation of a Replicated Database for Mobile Systems**. Wireless Networks 6, p.131-133, 2000.

PARK, T.; WOO, N.; YEOM, H. **An Efficient Optimistic Message Logging Scheme for Recoverable Mobile Computing Systems**. IEEE Transactions on Mobile Computing, p.265-277, 2002.

PARK, T.; YEOM, H. **An Asynchronous Recovery Scheme based on Optimistic Message Logging for Mobile Computing Systems**. The 20th International Conference on Distributed Computing Systems, p.436-443, 2000.

PHATAK, S.; BADRINATH, B. **Multiversion Reconciliation for Mobile Databases**. New Brunswic, NJ 08903, Rutgers University, p.1-2, 2001.

PITOURA, E.; BHARGAVA, B. **Data Consistency in Intermittently Connected Distributed Systems**. Knowledge and Data Engineering, vol.11, n.6, p.896-915, 1999.

PITOURA, E.; CHRYSANTHIS, P. **Scalable Processing of Read-Only Transactions in Broadcast Push**. 19th IEEE Int'l Conference on Distributed Computing Systems, 1999.

PITOURA, E.; SAMARAS, G. **Data Management for Mobile Computing**. Kluwer Academic Publishers, 1998.

POCKET-TECHNOLOGIES. **PocketStudio Professional Edition: Write PalmOS Applications With Your Delphi Skills**. <http://www.pocket-technologies.com>. Acesso em Junho de 2003.

RABINOVICH, M.; GEHANI, N.; KONONOV, A. **Scalable Update Propagation in Epidemic Replicated Databases**. International Conference on Extending Data Base Technology, p.207-215, 1996.

RAMALHO, J.. **Sybase SQL Anywhere Studio**. São Paulo: Berkley Brasil, 2000.

RAPPAPORT, T. **Wireless Communications: Principles and Practice**. Prentice Hall, US, 1996.

RATNER, D. **Roam: A Scalable Replication System for Mobile and Distributed Computing**. Dissertation for the Degree Doctor of Philosophy in Computer Science, UCLA, 1998.

RATNER, D.; POPEK, G.; REITHER, P. **The Ward Model: A Replication Architecture for Mobile Environments**. Technical Report CSD-960045, p.1-5, 1996.

RATNER, D.; REIHER, P.; POPEK, G.; et al. **Replication Requirements in Mobile Environments**. Mobile Networks and Applications, 2001.

REIHER, P.; POPEK, J.; GUNTER, M.; et al. **Peer-to-Peer Reconciliation Based Replication for Mobile Computers**. ECOOP'96 II Workshop on Mobility and Replication, p.1-3, 1996.

SAITO, Y.; LEVY, H. **Optimistic Replication for Internet Data Services**. 14th International Conference on Distributed Computing, p.297-314, 2001.

SAMARAS, G.; PITSILLIDES. **A Computational Model for Wireless Environments**. 4ª International Conference on Telecomunications, p.1-19, 1997.

SATYANARAYANAN, M. **Fundamental Challenges in Mobile Computing**. ACM Symposium on Principles of Distributed Computing, Filadélfia, p.1-7, 1996.

SATYANARAYANAN, M.; NARAYANAN, D. **Multi-fidelity Algorithms for Interactive Mobile Applications**. 3rd International Workshop on Discrete Algorithms and Methods for Mobile Computing and Communications, p.1-6, 1999.

SILVA, F.; ENDLER, M. **Requisitos e Arquiteturas de Software para Computação Móvel**. I Workshop SIDAM - Sistemas de Informação Distribuída de Agentes Móveis, 2000.

SNOEREN, A. **A session-Based Architecture for Internet Mobility**. PhD thesis, Massachusetts Institute of Technology, December, 2002.

SNOEREN, A.; BALAKRISHNAN, H. **An End-to-End Approach to Host Mobility**. 6th ACM/IEEE International Conference on Mobile Computing and Networking, p.1-12, 2000.

STEINKE, R. **Epidemic transactions for replicated databases**. Master's Thesis, University of California at Santa Barbara, Dept. Computer Science, UCSB, 1997.

TABDEE. **Turbosync: Connecting Delphi to your Palm**. (Fev.2003). <http://www.tabdee.ltd.uk>. Acesso em Junho de 2003.

TENNENHOUSE, D.; SMITH J.; SINCOSKIE, W.; et al. **A Survey of Active Network Research**. IEEE Communication Magazine, p.2-6, 1996.

TERRY, D.; PETERSEN, K.; SPEITZER, M.; et al. **A Case for Non-transparent Replication: Examples from Bayou**. IEEE International Conference on Data Engineering, p.12-20, 1998.

WILSON, G.; OSTREM, J.; LIU, C.; et al. **Palm OS Programmer's API Reference.** Technical Report PalmSource 3003-005, May, 2002.

WOLFSON, O.; JAJODIA, S.; HUANG, Y. **An Adaptive Data Replication with Selective Control**. In Interactive Conference on Mobile Database Systems, p.1-8, 1997.

XIA, Y.; HELAL, A. **A Dynamic Data/Currency Protocol for Mobile Database Design and Reconfiguration**. SAC2003, p.2-6, 2003.

ANEXO I

PUBLICAÇÕES

Título: **Replicação e Sincronização de Banco de Dados Móveis em Ambientes Wireless.**
Evento: II Simpósio de Informática da Região Centro do RS, Santa Maria-RS.
Data: Agosto de 2003.
Autores: Daniel Pezzi da Cunha e Mario Antônio Ribeiro Dantas.

Título: **Um Estudo das Estratégias de Replicação e Sincronização de Banco de Dados Móveis em Ambientes Wireless.**
Evento: XI Escola de Informática da SBC-PR, Londrina-PR.
Data: Setembro de 2003.
Autores: Daniel Pezzi da Cunha e Mario Antônio Ribeiro Dantas.

Título: **Reconciliação de Dados em Ambiente Móvel com Wireless.**
Evento: VIII Simpósio de Informática da PUC-RS, Uruguaiana-RS.
Data: Outubro de 2003.
Autores: Daniel Pezzi da Cunha e Mario Antônio Ribeiro Dantas.