Capítulo

1

Sistemas de Computação Móvel: da Mobilidade à Ubiqüidade

Iara Augustin, João Carlos Damasceno Lima, Fabio Lorenzi da Silva, Giuliano Lopes Ferreira, Tiago Antonio Rizzetti.

Centro de Tecnologia, Universidade Federal de Santa Maria (UFSM)
Av. Roraima 1000 – 97.105-900 – Santa Maria – RS – Brazil
{august,caio, lorenzi, giuliano, rizzetti}@inf.ufsm.br

Abstract.

This tutorial registers the progresses in Mobile Systems from Nomadic Computing to Pervasive/Ubiquitous Computing and it delineates the concepts that can contribute to the programming of new applications class that executes in a pervasive space. The first generation of pervasive computing environments focused middlewares for management and availability of the smart spaces with goal of understanding the requirements and needs of these. Some of the proposed concepts and case studies of programming tools are presented in this text. The potential of the Java language is analyzed with views to the programming of mobile, pervasive and ubiquitous applications.

Resumo.

Este tutorial registra os avanços dos sistemas móveis desde a computação nômade até a Computação Pervasiva/Ubíqua e delineia os conceitos advindos da mobilidade e que contribuem para uma nova classe de aplicações. A primeira geração de ambientes para computação ubíqua focalizou middlewares para gerenciamento e disponibilização do ambiente com objetivo de entender os requisitos e necessidades destes. Alguns dos conceitos propostos e estudos de ferramentas de programação são apresentados neste texto. O potencial da linguagem Java é analisado com vistas à programação de aplicações móvel, pervasiva e ubíqua.

1.1 Introdução

A Computação Ubíqua ou Pervasiva é um novo paradigma computacional oriunda das tecnologias de rede sem fio e sistemas distribuídos, em um processo evolutivo iniciado pela Computação Nômade e seguido pela Computação Móvel, estágio atual da tecnologia móvel [Satyanarayanan 2001]. Nesta, o usuário, portando dispositivos móveis como palmtops, notebooks e telefones celulares, tem acesso a uma infra-estrutura compartilhada e independente da sua localização física. Isto disponibiliza uma comunicação flexível entre as pessoas e um acesso aos serviços de rede. Observa-se que a crescente introdução de facilidades de comunicação tem deslocado as aplicações da computação móvel de uma perspectiva de uso pessoal para outras mais avançadas e de uso corporativo, como as aplicações móveis distribuídas.

Sistemas distribuídos tradicionais são construídos com suposições sobre a infra-estrutura física de execução, como conectividade permanente à rede fixa e disponibilidade dos recursos necessários. Porém, essas suposições não são válidas nos sistemas móveis. Isto impede o uso direto das soluções adotadas pelos sistemas distribuídos, as quais podem ser altamente ineficientes devido à variabilidade freqüente da conexão à rede e da disponibilidade de recursos e serviços.

Parece, portanto, ser necessário definir uma nova arquitetura para sistemas móveis, projetada com mobilidade, flexibilidade e adaptabilidade intrínsecas. A produção de software no ambiente móvel é, ainda, complexa. Seus componentes são variáveis no tempo e no espaço em termos de conectividade, portabilidade e mobilidade. O desafio que se apresenta é, portanto, projetar aplicações móveis distribuídas cujos níveis de serviço e disponibilidade de recursos são imprevisíveis. Emergem, portanto, novos requisitos para o desenvolvimento de aplicações, os quais geram uma nova classe de aplicações projetadas especificamente para este ambiente dinâmico.

Esta nova classe de aplicações tem sido referenciada de muitas formas: *environment-aware, network-aware, resource-aware, context-aware applications*. Porém, todas têm um conceito embutido: adaptação ao contexto. Isto significa que os sistemas devem ter consciência da localização e da situação onde estão inseridos, e devem tirar vantagem desta informação para (auto)configurar-se dinamicamente de um modo distribuído. A diferença entre as aplicações está no grau de adaptabilidade e nos recursos que são objetos dessa adaptação. O foco da complexidade na implementação dessas aplicações móveis, com comportamento adaptativo, está no fato de que os componentes distribuídos das mesmas podem sofrer influência dos diversos ambientes onde estão inseridos.

Neste curso, faz-se uma revisão evolutiva dos conceitos de mobilidade na seção 1.2. Na seção 1.3, abordam-se a visão futura proposta pela Computação Ubíqua. Na seção 1.4, faz-se uma revisão das tecnologias atualmente disponíveis para o gerenciamento e execução de aplicações móveis em direção à pervasividade/ubiquidade. Na seção 1.5, as ferramentas de programação de aplicações móveis mais utilizadas são resumidas. As conclusões são apresentadas na seção 1.6.

1.2 Da Mobilidade à Ubiquidade

O termo computação móvel (*mobile computing*) não está ainda bem definido e é usado pelos autores em um espectro de ambientes, que envolvem alguma forma de mobilidade. De forma geral, pode-se dizer que "computação móvel é a computação distribuída que envolve elementos (software, dados, hardware, usuário) cuja localização se altera no curso da execução" [Augustin 2004]. Esta definição torna evidente a amplitude de abrangência desta nova área da computação.

1.2.1 Cenários Possíveis da Computação Móvel

Dependendo dos elementos que possuem a propriedade de mobilidade, podem-se definir diferentes cenários. Entre eles:

- Computação Nômade (*nomadic computing ou palm computing*) – popularizada com o uso de dispositivos portáteis, tais como os palmtops (a Palm era líder desse mercado) e aplicações de gerenciamento pessoal. A mobilidade está mascarada através da portabilidade do hardware (PDA – Personal Digital Assistant) e não é transparente. Nos início dos anos 90, as facilidades de comunicação eram, basicamente, via acesso discado; a cada movimentação, uma nova conexão à rede era requerida;
- Computação com Redes Sem Fio (*wireless computing*) – usuário portando um equipamento pode se mover dentro de uma área de acesso, enquanto mantém a conexão a rede fixa;
- Mobilidade de Código *(Mobile Computation)* – os componentes da aplicação podem ser mover. Pode-se ter somente: a mobilidade de código; a mobilidade de dados; ou a mobilidade de todo o estado da execução da aplicação (por exemplo: agentes móveis);
- Computação Ubíqua *(ubiquitous/pervasive computing)* – o ambiente é impregnado de dispositivos e equipamentos computacionais conectados entre si e invisíveis ao usuário final. O usuário dispõe de seu ambiente computacional independente de localização, tempo, dispositivo e rede subjacente.

Este último cenário categoriza a mobilidade de todos os elementos, o qual permite ao usuário deslocar-se junto com seu ambiente computacional.

A computação móvel é caracterizada por três propriedades: **portabilidade**, **mobilidade**, e **conectividade** [Augustin et al 2002] que introduzem restrições no ambiente. Para ser portátil, um computador deve ser pequeno, leve e requer fontes pequenas de energia. Isto significa que um computador portátil tem restrições no tamanho de memória, na capacidade de armazenamento, no consumo de energia e na interface do usuário. Além disso, a portabilidade potencializa o risco de perda, queda ou roubo. Quando em movimento, o dispositivo móvel pode alterar sua localização e, possivelmente, seu ponto de contato com a rede fixa. Essa natureza dinâmica do deslocamento introduz questões relativas ao endereçamento dos nós, localização do usuário e informações dependentes da localização. Além da mobilidade física, a aplicação com seu código, dados e estado também pode se deslocar entre os nós da rede. A conexão à rede através do meio sem fio levanta outros obstáculos: comunicação

intermitente (desconexões freqüentes, bloqueio no caminho do sinal, ruído), restrita (e altamente variável) largura da banda, alta latência e alta taxa de erros.

Outro requisito importante a considerar no projeto de aplicações móveis é o grau de conectividade à rede. Estudos, a partir dos sistemas de arquivos, têm demonstrado que existem basicamente três **modos de operação** de um sistema móvel [Satyanarayanan 2001]: fortemente conectado – conexão sobre uma rede fixa, rápida e confiável; fracamente conectado – conexão sobre um canal sem fio com largura de banda restrita; e desconectado – sem conexão à rede. O **modo desconectado é eletivo**[1] e utilizado, principalmente, para economia de recursos do *host* móvel, como o consumo de energia. Observa-se que o *host* móvel estará, na maior parte do tempo, desconectado da rede, sendo a desconexão um modo normal no ambiente móvel, enquanto que no ambiente distribuído é uma exceção. Desta forma, o sistema deve prover uma "ilusão" de conexão para o usuário móvel.

Três outros requisitos existentes em Sistemas Distribuídos assumem maior relevância quando a mobilidade está presente. São eles: **escalabilidade,** refere-se ao tamanho do conjunto potencial de usuários; **heterogeneidade**, introduzida por diferentes equipamentos e redes móveis; e **dinamismo** introduzido pela alta variabilidade da disponibilidade de recursos e pela mobilidade do usuário. Esses aspectos não são isolados, e devem ser abordados na arquitetura do sistema móvel.

Como visto, restrições são da natureza da mobilidade e colocam novas demandas no projeto de aplicações, as quais devem ser mais flexíveis que as atuais aplicações distribuídas quando um recurso está indisponível/inacessível ou tem seu nível de disponibilidade/acessibilidade reduzido. Para serem efetivos e apresentarem um desempenho compatível com a expectativa do usuário, essas aplicações devem exibir a capacidade de **adaptação** às freqüentes e rápidas alterações no ambiente de execução durante o curso de evolução da aplicação.

As soluções para ambientes distribuídos tradicionais (redes fixas) não são apropriadas, pois não permitem uma adaptação dinâmica em face às alterações do contexto de execução. Além disso, a maioria das soluções adaptativas atuais satisfazem as necessidades de aplicações específicas, sendo insuficientes para serem reusadas em aplicações de propósito geral.

A Figura 1.1 ilustra a arquitetura de gerenciamento da mobilidade. A rede de acesso pode incluir redes celulares, redes de comunicação pessoal, redes sem fio, numa dimensão de macro a pico células. A rede *backbone* inclui várias redes de alta velocidade, como FDDI e ATM, que podem estar conectadas à Internet. O gerenciamento da mobilidade, inicialmente desenvolvido para suporte a mobilidade de usuário e terminal, é estendida para gerenciar mobilidade de serviços e recursos.

1.3 Futuro com a Computação Ubíqua

Computação Pervasiva (*Pervasive Computing*) é uma nova área de pesquisa que originou-se da proposta de Mark Weiser, chamada Computação Ubíqua (*Ubiquitous Computing*) [Weiser 1991]. Também é denominada Tecnologia Tranqüila (*Calm*

[1] Termo introduzido por Dan Duchamp, significando que o *host* pode informar antecipadamente ao sistema que ocorrerá a desconexão, e este pode executar um "protocolo de desconexão".

Tecnhology), Inteligência Ambiente (Inteligence Ambient), Computação Pró-ativa (*Proactive Computing*) e Computação Invisível (*Invisible Computing*) entre outros nomes.Porém, os termos que têm predominado são Computação Pervasiva[2] e Computação Ubíqua. A Figura 1.2 ilustra o estágio atual (a) e a visão proposta pela Computação Ubíqua.

Em um espaço pervasivo/ubíquo (também chamado *smart space*), computadores e outros (vários e variados) dispositivos digitais estão totalmente integrados ao ambiente do usuário e objetivam auxiliá-lo em suas tarefas diárias.

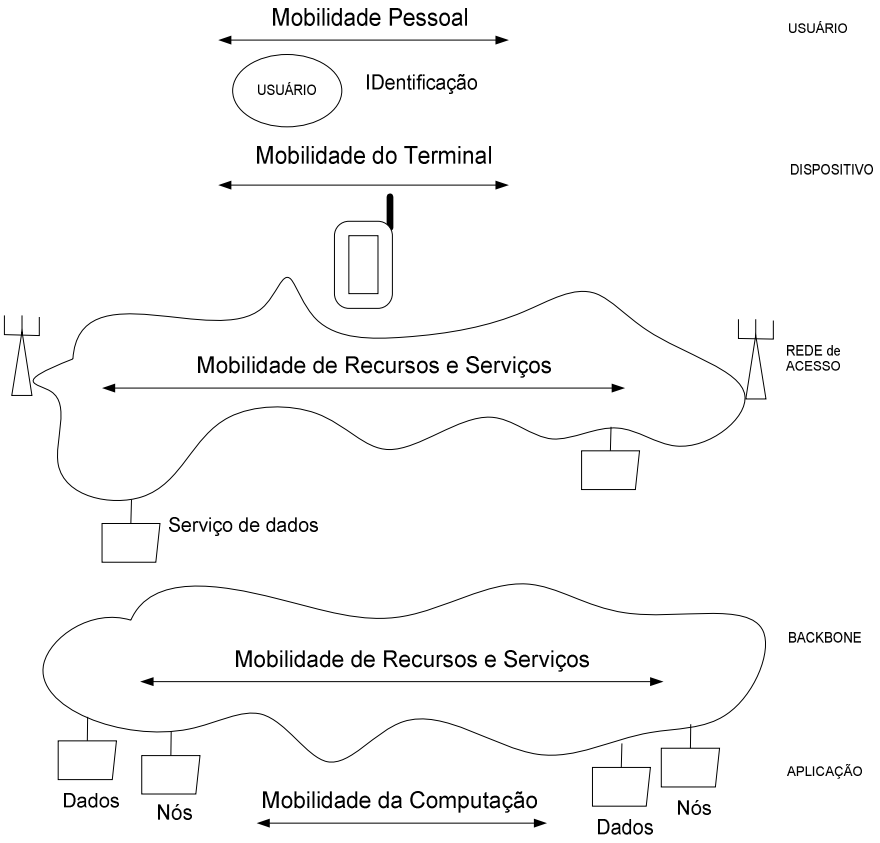


Figura 1.1. Tipos de Mobilidade

Este é um ambiente altamente dinâmico e heterogêneo. Os recursos, incluindo serviços, dispositivos e aplicações, disponíveis podem alterar-se rapidamente. Diferentes espaços têm diferentes tipos de recursos disponíveis e diferentes políticas de uso dos recursos. Programas executando neste ambiente devem ser capazes de se adaptar a troca do contexto e disponibilidade de recursos. Isto coloca um desafio para os desenvolvedores que devem especificar como o programa deve se comportar em diferentes contextos e quando diferentes tipos de recursos estão disponíveis. Além disso, diferentes espaços pervasivos podem ter diferentes modos de executar a mesma tarefa

---

[2] O termo 'pervasivo' (espalhado, integrado, universal) não existe ainda na Língua Portuguesa. Alguns autores consideram que se deve usar somente o termo Computação Ubíqua, pois este existe na Língua Portuguesa e significa onipresente.

uma vez que têm variados serviços, aplicações e recursos. O desenvolvedor não pode esperar saber antecipadamente como as várias tarefas serão executadas nos diferentes espaços pervasivos. Assim, programadores necessitam de abstrações de alto nível para programar aplicações no espaço pervasivo sem ter que ter consciência dos recursos disponíveis, contexto, políticas e preferência dos usuários [Augustin 2004].

(a) (b)

Figura 1.2. Ambiente Computacional

Ambientes para Computação Pervasiva têm sido explorados através de uma série de protótipos de pesquisa na academia, tais como o projeto Aura [Garlan and Stenskiste and Schmerl 2002], projeto Gaia [Roman et al 2002], projeto Oxigen do MIT (www.oxygen.lcs.mit.edu); a indústria também tem dado atenção a essa área, tais como IBM Pervasive Computing Unit (www.research.ibm.com/thinkresearch/ pervasive.shtml). Muitos enfatizam os requisitos de tecnologia e a previsibilidade da interação e comportamento do ambiente.

Pela integração de sensores, computadores, dispositivos e redes foi possível desenhar a **primeira geração de ambientes pervasivos**, referenciados como 'ambientes integrados'. Agora os esforços de pesquisa concentram-se em deslocar o paradigma de 'ambientes integrados' para 'espaços programáveis'. O grande desafio é que a Computação Pervasiva afeta toda a Ciência da Computação em três distintas perspectivas: da experiência, da engenharia e teórica [Chalmers 2006].

Modelos de programação e middlewares baseados no modelo de sensores-atuadores-contexto foram os mais focados para a programação de aplicações da primeira geração e resultaram no conceito de Computação Orientada a Contexto (*context-aware programming*). Busca-se, agora, encontrar abstrações de alto nível que permitam programar aplicações que comporão um novo paradigma denominado Programação Orientada a Tarefas (*Task-Oriented Programming*).

1.3.1 O Cenário Ubíquo

Para construir-se o cenário visualizado pela Computação Pervasiva/Ubíqua é necessário uma pesquisa multidisciplinar envolvendo, praticamente, todas as áreas da computação:

sistemas distribuídos, sistemas móveis, redes de sensores, banco de dados, inteligência artificial, interface homem-computador, segurança, rede, etc.

Dado o contínuo progresso técnico em comunicação e computação, parece que se está caminhando a uma total integração da computação nas atividades humanas. Previsões indicam que em poucos anos, microprocessadores se tornarão pequenos e baratos o suficiente para serem embutidos em quase tudo – não somente em dispositivos digitais, carros, eletroeletrônicos, brinquedos, ferramentas, mas também em objetos (lápis, por exemplo) e roupas. Todos esses artefatos devem estar entrelaçados e conectados em uma rede sem fio.

De fato, a tecnologia espera uma revolução na qual bilhões de pequenos e móveis processadores estejam incorporados ao mundo físico, compondo objetos 'espertos' (smart[3]) – sabem onde estão, se adaptam ao ambiente fornecendo serviços úteis em adição ao seu propósito original, formam redes espontâneas (ad-hoc) e altamente distribuída numa ordem de magnitude muito maior que a de hoje.

Este cenário, conhecido como Computação Pervasiva ou Ubíqua, está sendo considerado como o novo paradigma do século 21 [Saha and Mukherjee 2003; Satyanarayanan 2001] ou a terceira onda da computação [Jansen et al 2005], o qual permite o acoplamento do mundo físico ao mundo da informação e fornece uma abundância de serviços e aplicações onipresentes visando que usuários, máquinas, dados, aplicações e objetos do espaço físico interajam uns com os outros de forma transparente (em background) [Ranganathan et al 2005].

É claro que se está movendo gradualmente em direção à visão de uma computação onipresente/ubíqua. Está-se incrementalmente acostumando-se a usar uma coleção de heterogêneos dispositivos (*personal computer*) para suportar uma crescente faixa de atividades. A corrente geração de dispositivos interconectados é somente o ponto de partida em direção à computação ubíqua [Chalmers 2006].

1.3.2 Requisitos e Desafios das Aplicações Pervasivas

Um exemplo de ambiente pervasivo é o hospitalar. Neste ambiente, algumas atividades são previsíveis e planejadas enquanto outras são randômicas, as atividades variam de simples a complexas, algumas atividades tem prioridade enquanto outras podem ser feitas quando houver tempo, algumas atividades são ligadas a determinadas salas e presença de certos artefatos. O trabalho dos clínicos é extremamente móvel e estes não podem carregar equipamentos pesados.

Logo, é interessante no ambiente o conceito de 'computador público' que não armazena atividades computacionais de ninguém, mas serve como um portal para acesso a elas. Este conceito requer uma infra-estrutura que gerencia, armazena e distribui atividades computacionais. Por exemplo, o usuário é identificado e autenticado no sistema por 'proximidade' – procedimento deve ser rápido, e o computador público recebe o ambiente virtual deste usuário para que ele possa desenvolver suas tarefas/atividades.

---

[3] dispositivo muitos em um (many-in-one device).

Outra propriedade necessária é a inferência pró-ativa das atividades baseada na localização da pessoa e artefatos ao redor. O trabalho dos clínicos é também altamente colaborativo por natureza – o atendimento a um paciente envolve, em geral, várias especialidades. Colaboração significa interromper a tarefa em execução para atender a solicitação por demanda (chaveamento de atividades).

O ambiente também requer o acesso a um conjunto de variadas e atualizadas informações, que podem ser requeridas por vários clínicos ao mesmo tempo. Uma organização de dados e acesso pervasivo a ele é requerida. Dispositivos móveis devem se comunicar com a infra-estrutura disponível, numa organização de rede infra-estruturada, ou descobrir novos dispositivos, numa organização de rede *ad-hoc* ou *mesh*.

Como o cenário pervasivo prevê uma mobilidade física (dos equipamentos e/ou dos usuários) e lógica (componentes da aplicação e serviços), potencialmente em escala global (larga-escala), deve fornecer transparência ao usuário, de forma que o usuário possa acessar seu ambiente computacional independente de localização, do meio de acesso e do tempo. O sistema de suporte para esse ambiente usa a metáfora de um ambiente virtual do usuário, onde as aplicações têm o estilo "siga-me" (*follow-me applications*) [Augustin et al 2005].

Soluções integradas (Figura 1.3) para disponibilizar este ambiente é o foco da primeira geração de sistemas pervasivos, dos quais pode-se citar o projeto ISAM (Infra-estrutura de Suporte às Aplicações Móveis Distribuídas – www.inf.ufrgs.br/~isam) .

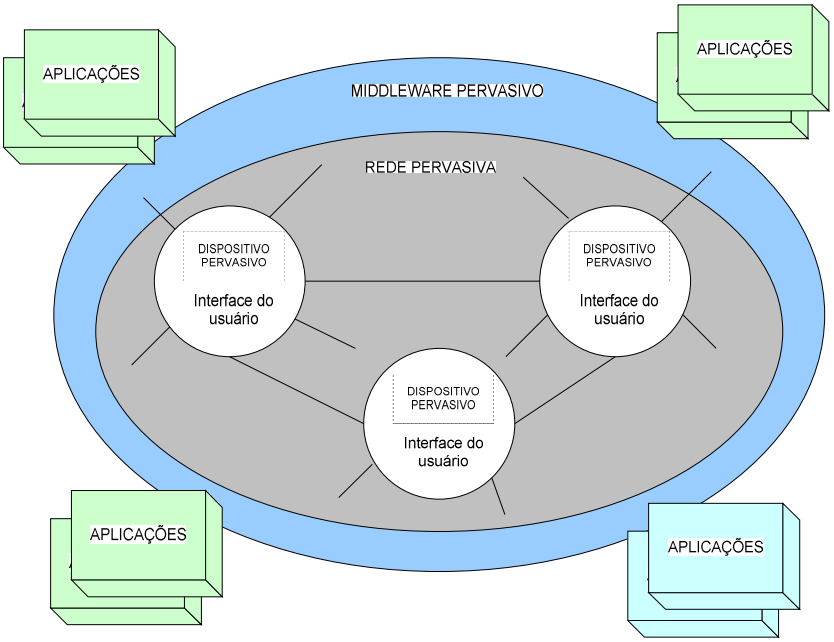


figura 1.3. Espaço Pervasivo

1.3.3 Panorama Geral da Primeira Geração de Sistemas Pervasivos

Pesquisadores têm, recentemente, desenvolvido vários sistemas e protótipos de computação pervasiva para demonstrar como este novo paradigma pode beneficiar domínios de aplicações específicos, como segurança de residências (*home security*),

educação (*pervasive learning*), saúde e emergências (*smart hospital*), casa virtual (*smart home*), e entretenimento.

Pode-se dizer que as estratégias adotadas dividem-se em:

- projeto de aplicações experimentais;
- desenvolvimento de sistemas genéricos experimentais;
- análise teórica, como o modelo Mobile Ambients [Cardelli and Gordon 1998] e a formalização do conceito de contexto [Jansen et al 2005].

Na maioria dos casos, a abordagem adotada é a integração de sistemas povoados com diversos dispositivos computacionais heterogêneos (*appliances*), incluindo sensores, atuadores, computadores, micro-controladores, etc. usando diversas redes e conectores. O objetivo desses sistemas é demonstrar a habilidade de envolver novas tecnologias emergentes e de entender os requisitos das aplicações. É uma estratégia com interesse experimental; não, necessariamente, prático. Dentre as diversas questões de pesquisas em progresso, as seguintes têm sido mais concentradas: redes móveis, redes de sensores e middlewares.

1.4 Tecnologias Disponíveis

Os principais componentes físicos da Computação Móvel envolvem os computadores portáteis, usados pelos usuários móveis, e as redes sem fio, com seus respectivos meios de transmissão de dados e padrões de comunicação de dados. Além disso, considerando que a Internet é o principal meio de comunicação usado hoje e que o uso de sua infra-estrutura é aplicável aos trabalhadores móveis, estão surgindo muitas propostas de adequação da arquitetura TCP/IP à mobilidade, dentre elas o protocolo WAP (*Wireless Application Protocol*).

1.4.1 Dispositivos Móveis e Convergência Tecnológica

Os dispositivos para o usuário final da Computação Móvel têm várias formas e configurações. As plataformas de *hardware*, o sistema operacional e as capacidades funcionais variam entre esses dispositivos. Entretanto, existem atributos que são compartilhados entre *notebooks*, *palmtops*, computadores *handhelds* e outros acessórios, tais como impressoras, fax e *scanners* móveis. Os **computadores portáteis** devem ser de tamanho reduzido e de pouco peso. Precisam ser resistentes, funcionais, práticos e portáteis. O desenvolvimento de computadores móveis deve conciliar objetivos que são conflitantes: oferecer recursos e desempenho semelhante aos do *desktop* com tamanho, peso e consumo de energia reduzidos. Também devem oferecer acesso remoto a outros computadores, servidores e *mainframes*, comunicação em rede e suporte a uma crescente variedade de mídias.

Apesar de os *notebooks* serem os pioneiros na Computação Móvel, a popularização de transportar um computador para qualquer lugar veio com o lançamento da primeira versão do Palm, da *Palm Computing* (www.palm.com),

chamado de PalmPilot, ocorrido em 1996. Desde então uma crescente evolução na capacidade e funcionalidade destes dispositivos tem sido observada.

Outra linha de dispositivos usados é a de **telefones celulares**. A 2,5ª geração dos celulares, conhecida como *Personal Communciations Services* (PCS) dispõe de serviços que facilitam a comunicação e permitem ao usuário executar certas funções, como enviar mensagens, e o acesso à Internet móvel com o protocolo WAP (http://www.wapforum.org). A 3ª geração de telefones celulares permitirá aplicações de multimídia e videoconferência, embora ainda não contemple *roaming* global.

Na década de 1990, acompanhando o avanço das redes de telecomunicações, tornou-se mais comum falar em convergência tecnológica ou convergência digital. A popularização da Internet foi um passo fundamental para que o conceito se difundisse, principalmente fora dos meios corporativos. Na verdade os primeiros internautas não dispunham de recursos adequados para obter a qualidade esperada em serviços convergentes, pois a maioria dependia de conexões discadas por enlaces analógicos sobre par-de-cobre. O usuário doméstico comum só começou a beneficiar-se da convergência com a adoção em massa de conexões de banda larga (xDSL), que pela primeira vez forneceram, a um custo acessível, capacidade de transmissão suficiente para utilização de serviços, tais como Voz sobre IP (VoIP).

Com a consolidação da Internet como a mais importante rede de informações do mundo globalizado, também se estabeleceram os padrões tecnológicos que ela emprega, tais como o protocolo IP e a comutação de pacotes. Esses elementos, aliados ao barateamento e aprimoramento dos meios de transmissão em banda larga, crescente demanda por serviços multimídia, criação de novos protocolos como o SIP e de mecanismos como MPLS, estão dando forma à arquitetura de redes convergentes que vem sendo chamada de *Next Generation Networking*.

O ponto de partida para o fenômeno da convergência tecnológica é, evidentemente, a viabilidade de desenvolvimento e comercialização em grande escala de soluções de tecnologia convergentes, sejam redes, serviços ou terminais.

A convergência atual de tecnologias de distribuição de voz, dados, imagens e sons através da digitalização da informação passa por diversas instâncias, seja a convergência de equipamentos de comunicação, telecomunicações e informática; a convergência dos modelos de consumo de informação, entre comunicação de massa e comunicação interativa; a convergência dos produtos das indústrias culturais em um único produto multimídia; e a convergência da economia das comunicações que agrupa dois setores distintos – telecomunicações e comunicação eletrônica de massa – mediados pela informática.

1.4.1.1 Convergência de redes

É a unificação entre duas ou mais redes de comunicação distintas numa única rede capaz de prover os serviços antes prestados pelas diversas redes. Um dos primeiros exemplos é a convergência entre redes de voz e dados, inicialmente através de tecnologia RDSI e, mais recentemente, pela tecnologia xDSL.

Ultimamente, aos serviços de voz e dados tem-se incluído serviços de vídeo e/ou multimídia. Muitos desses serviços não existiam antes de se começar a falar em

convergência de redes, por isso pode-se dizer que já "nasceram convergentes", como IPTV (que é diferente de simplesmente enviar a transmissão da televisão analógica tradicional por protocolo IP). A oferta combinada de serviços de voz, Internet banda larga e televisão recebe o nome de *Triple play*, esse termo tem origem no Marketing e é um modelo de negócios para comercialização dos produtos.

1.4.1.2 Convergência fixo-móvel

Nos anos 1990 começou-se a falar na convergência entre telefonia fixa e móvel, mas sem resultados práticos. Uma década depois o assunto ressurge, ainda sem uma definição clara do que seria tal convergência, embora se possa dizer em linhas gerais que "tem como objetivo disponibilizar serviços convergentes pelos ambientes fixo, móvel e Internet".

Atualmente, as operadoras de telefonia enfrentam desafios para desenvolver estratégias para convergência fixo-móvel. As tecnologias que recebem mais atenção (*Unlicensed Mobile Access, IP Multimedia Subsystem*) são centradas na própria rede e estão em estágio imaturo, despendendo esforços que divergem da real necessidade da prestação efetiva de serviços para competir com outros provedores como Skype. Ainda falta demanda de mercado consistente, tanto de consumidores quanto empresas.

O Yankee Group (2004) publicou um estudo que identifica quatro estágios sucessivos na convergência fixo-móvel:

- Convergência por pacotes. Forma mais básica de convergência que consiste simplesmente na oferta comercial de telefonia fixa e móvel num único pacote de serviços. Não há integração entre tecnologias, mas unificação do atendimento ao consumidor e cobrança de faturas.
- Convergência de recursos. Integração de recursos que, anteriormente, existiam apenas para telefones fixos ou móveis. Podem-se citar funcionalidades de transferência automática de chamadas direcionadas para um telefone fixo (como na residência do cliente) para seu celular ou vice-versa, bem como caixa de mensagens de voz integrada.
- Convergência de produto. Convergência resultante da redundância entre produtos fixo e móvel, fazendo com que efetivamente se tornem um só. É um amadurecimento da convergência de recursos, pois à medida que começam a ser oferecidos em um produto recursos que só eram disponíveis no outro (por exemplo, suporte a E911 em telefones móveis e melhoria do sinal dentro das residências), a telefonia fixa tende a cair em desuso.
- Convergência total. Quando a experiência do usuário ocorre de maneira transparente, coesa, contínua. Pode-se mudar de localização ou terminal sem sobressaltos, mantendo acesso às mesmas informações e serviços. A mesma agenda de contatos telefônicos, perfis e configurações ou arquivos multimídia estariam sempre disponíveis e sincronizados seja no telefone móvel, PDA ou computador desktop (PC).

1.4.1.3 Convergência de serviços

É a disponibilização de um mesmo serviço através de diferentes meios de comunicação. Essa modalidade de prestação de serviços tem sido utilizada por diversos segmentos, entre eles o segmento bancário. Há cada vez mais opções para o cliente consultar seu saldo: essa simples operação, que originalmente só podia ser realizada através do caixa humano ou pelo caixa eletrônico, já está disponível através da Internet, telefone fixo ou dispositivo móvel.

O pacote de serviços que inclui os três serviços do *Triple play* mais telefonia móvel vem recebendo o nome de *Quadruple play*, também sem representar qualquer inovação do ponto de vista tecnológico.

1.4.1.4 Convergência de terminais

É a utilização de um único terminal para acesso a múltiplas redes e serviços diversos. Exemplo inclui os *smartphones*, que combinam características de telefones celulares e PDAs. Desde os primeiros exemplares no início dos anos 2000, como o QCP da Kiocera ao lado, o uso desses aparelhos vem aumentando, principalmente no mercado corporativo onde há, ainda, um domínio da linha BlackBerry. Como as tarifas cobradas pelos serviços de transmissão de dados estão baixando, prevê-se que estes devem começar a substituir os telefones celulares (que estão saturados) também no mercado de massa (INFOEXAME, 2007). O iPhone, anunciado em janeiro de 2007, é um smartphone da Apple Inc. apresentado como um "telefone revolucionário", tanto que na mesma data do anúncio a empresa alterou sua razão social de "Apple Computer, Inc." para simplesmente "Apple Inc.". O objetivo das pesquisas que resultaram no iPhone foi a experimentação de telas sensíveis ao toque que, assim como o design e a facilidade de uso, é considerado um dos pontos fortes do aparelho. Por outro lado, além da comunicação por voz que se espera de qualquer telefone, o iPhone integra recursos multimídia, conexão à Internet por tecnologia EDGE com acesso à web e e-mails, e conectividade local por Wi-Fi e Bluetooth. Tais recursos conferem ao aparelho características de um terminal convergente.

Estas características também estão presentes em modelos novos de smartphones de outros fabricantes. O maior desafio para os desenvolvedores de aplicações é fazer com que essas executem satisfatoriamente em diferentes sistemas e modelos (ver seção 1.5).

Outros aparelhos considerados convergentes incluem:

* UTStarcom GF200 - telefone fixo-móvel, capaz de fazer chamadas por Wi-Fi e Bluetooth e funcionando como telefone celular convencional onde não houver um ponto de acesso.

* HTC P3300 - oferece localização por GPS e conectividade GSM/GPRS, Wi-Fi e Bluetooth, além de outros recursos triviais nos aparelhos de hoje como recepção de sinal de rádio FM.

1.4.1.5 Convergência regulatória

O surgimento de serviços convergentes cria um ponto de contato entre dois mercados: o da telefonia, tradicionalmente regulamentado, e o mercado de serviços de dados, sujeito a pouca ou nenhuma regulamentação sobre a prestação dos serviços.

Dentre os desafios a serem enfrentados pelos órgãos reguladores, incluem-se a manutenção garantida de princípios, como a defesa da justa competição no setor de telecomunicações e radiodifusão. Por exemplo, através de ligações VoIP é possível enquadrar-se na lacuna não regulamentada dos serviços de transmissão de dados para evitar acordos internacionais e prover chamadas de voz mais baratas.

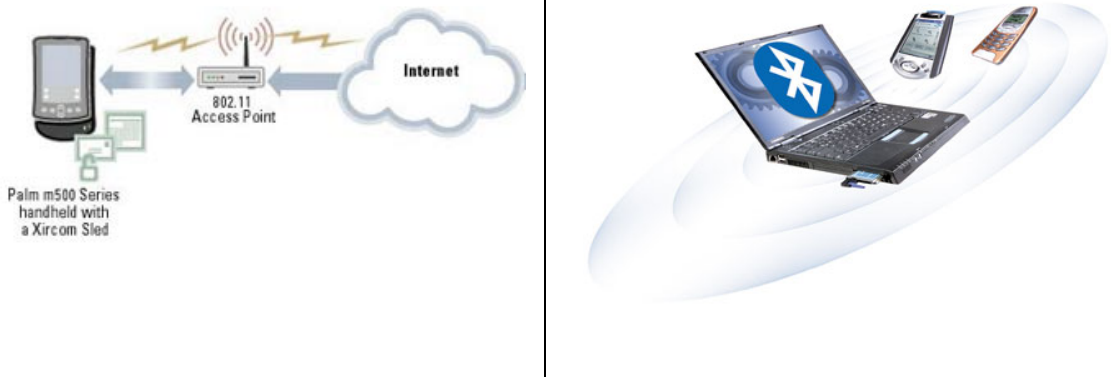
1.4.2 Redes Móveis (*Mobile Networks*)

Desde seu início na década de 70, as redes sem fio têm ganhado popularidade na indústria da computação, sendo utilizadas, principalmente, para permitir mobilidade. Atualmente, existem dois tipos principais de redes sem fio: redes infra-estruturadas e redes sem infra-estrutura (*ad-hoc*).

Nas **redes infra-estruturadas**, parte da rede é fixa e cabeada (Figura 1.4a). Os nós móveis comunicam-se com a parte fixa através de estações-base, que fazem o enlace entre a rede cabeada e a rede sem fio. Quando um nó se move para fora do alcance de uma estação-base e entra no raio de cobertura de outra, as estações trocam informações sobre o nó, e este passa a se comunicar com a nova estação de forma transparente. Aplicações típicas desse tipo de rede incluem redes locais sem fio em escritórios ou empresas.

As **redes sem infra-estrutura** (redes *ad-hoc*) não possuem nenhum tipo de estação base ou roteador (Figura 1.4b). Nela, todos os nós são móveis (comunicam-se por meio físico sem fio) e podem se conectar dinamicamente uns com os outros, formando a rede espontaneamente. Nós que não estão diretamente conectados se comunicam através do encaminhamento das mensagens através de nós intermediários. Cada nó da rede funciona como um roteador que descobre e mantém rotas para outros nós. Por isso, muitas vezes, as redes sem fio espontâneas são chamadas de redes sem fio *multi-hop*.

Mais especificamente, uma rede sem fio espontânea é um conjunto de dispositivos sem fio que podem, dinamicamente, se auto-organizar em uma topologia aleatória e temporária, para formar uma rede sem usar nenhuma infra-estrutura pré-existente. Essas redes também são conhecidas como redes móveis espontâneas (*Mobile Ad-hoc Network* - MANET), e podem formar grupos de terminais sem fio autônomos.



(a) (b)

Figura 1.4. Rede infra-estruturada e Rede Ad-hoc

Apesar de que (na prática) alguns terminais podem estar conectados a uma rede fixa, a característica principal das redes espontâneas é sua auto-configuração dinâmica, sem a intervenção de uma administração centralizada. As principais vantagens das redes espontâneas são flexibilidade, baixo custo e robustez. Aplicações típicas para essas redes incluem operações de resgate em desastres e encontros ou convenções em que pessoas precisam compartilhar informações rapidamente.

Dependendo do alcance da comunicação, as redes sem fio podem ser classificadas em: *Body Area Network* (BAN), *Personal Area Network* (PAN) e *Wireless Local Area Network* (WLAN). As BANs são formadas por um conjunto de dispositivos que têm alcance de comunicação em torno de dois metros. As PANs referem-se a comunicações entre diferentes BANs, tendo alcance de, aproximadamente, dez metros. E, finalmente, as WLANs têm alcance de comunicação na ordem de centenas de metros.

As BANs e PANs são implementadas, principalmente, com tecnologia *Bluetooth*, adotando o padrão IEEE 802.15.1. Já as WLANs, também conhecidas como Wi-Fi (*Wireless Fidelity*), são implementadas com tecnologia IEEE 802.11. A família 802.11 inclui vários padrões, como IEEE 802.11b e IEEE 802.11g, que diferem na camada física.

Desde a publicação dos protocolos para redes via rádio pela Defense Advanced Research Projects Agency (DARPA) na década de 70, numerosos protocolos têm sido desenvolvidos para redes móveis. Esses protocolos precisam lidar com as limitações desse tipo de rede, como limitado armazenamento de energia, baixa largura de banda e alta taxa de erros.

Os trabalhos de pesquisa atuais em redes móveis espontâneas (*ad-hoc*) estão direcionados, principalmente para os campos: (a) controle de acesso ao meio físico; (b) roteamento; (c) gerenciamento de recursos (*service discovery*); (d) gerenciamento de energia; (e) segurança. Além disso, tem-se levantado questões relativas aos protocolos de transporte (TCP) e interação entre diferentes camadas de comunicação (*cross-layer*).

1.4.2.1 Roteamento em Redes Ad-hoc

Como discutido anteriormente**,** uma rede móvel espontânea é um conjunto de nós móveis, formando uma topologia dinâmica. Com o intuito de facilitar a comunicação entre os nós, um protocolo de roteamento é usado para descobrir rotas entre os nós.

O objetivo principal de um protocolo de roteamento para redes espontâneas é estabelecer uma rota correta e eficiente entre dois nós para que as mensagens possam ser entregues oportunamente. Além disso, a construção da rota deve ser feita com mínimo *overhead* e baixo consumo de largura de banda.

Roteamento é o campo de pesquisa mais ativo em redes espontâneas (Hu and Perring 2004]. O maior desafio é desenvolver um protocolo de roteamento eficiente que, dinamicamente, encontre uma rota entre dois nós que precisam se comunicar. Minimizar o número de *hops* não é mais o principal objetivo dos algoritmos de roteamento, mas otimizar diversos parâmetros como taxa de erro, consumo de energia, *overhead* de roteamento, velocidade de configuração e reparo das rotas, possibilidade de estabelecer rotas paralelas, etc.

Os protocolos de roteamento podem ser classificados em duas categorias:

- protocolos baseados em tabela ou pró-ativos. Refere-se à classe de protocolos de roteamento que objetiva manter informações atualizadas sobre rotas entre todos os nós da rede. Os protocolos dessa classe necessitam que cada nó mantenha informações de roteamento e que essas informações sejam periodicamente propagadas pela rede, mantendo uma visão consistente da topologia rede. As características nas quais os protocolos diferem são o número de tabelas mantidas e a forma de propagar as modificações na topologia;
- protocolos sob-demanda ou reativos. Refere-se à classe de protocolos onde o objetivo é criar rotas somente quando elas forem requisitadas. Assim, quando um nó necessita uma rota para algum destino, ele inicia um processo de descoberta de rota. Esse processo se completa quando a rota é encontrada ou quando todas as possibilidades forem examinadas e o destino não foi alcançado. Quando uma rota é estabelecida, ela é mantida por um processo de manutenção de rotas até que o destino não seja mais alcançável, ou a rota não seja usada por longo período.

A maioria dos protocolos de roteamento existentes se enquadra em uma dessas classes. Porém, existem alguns protocolos que utilizam as duas abordagens (pró-ativa e reativa). Esse tipo de protocolo é conhecido como protocolo híbrido.

Na abordagem pró-ativa, a manutenção das tabelas independe de quando e quão freqüentemente as rotas são usadas. Nessa abordagem, um mecanismo de propagação de informações de roteamento atualiza constantemente as tabelas de cada nó, deixando as rotas sempre disponíveis. Já na abordagem reativa, quando um nó precisa de uma rota para determinado destino, ele deve aguardar até que um processo de descoberta de rota termine. Por outro lado, a constante propagação de informações de roteamento nos protocolos pró-ativos ocasiona elevado tráfego na rede e maior consumo de energia, mesmo quando não há mudanças na topologia da rede.

1.4.2.2 Segurança nas Redes Ad-hoc

Segurança em redes espontâneas sem fio ainda é uma tarefa desafiadora [Karlog and Wagner 2003]. Algumas características específicas dessas redes, tais como: transmissão por rádio é naturalmente broadcast, ausência de infra-estrutura, topologia dinâmica e comunicação *multi-hop* colaborativa, são as responsáveis pelos maiores problemas de vulnerabilidade.

Nas redes infra-estruturadas, a integridade, a autenticidade e a confidencialidade das mensagens são tratadas nas camadas mais altas, por mecanismos de segurança fim-a-fim (*end-to-end*) como SSH e SSL Porém, em redes espontâneas, esse tipo de mecanismo não é aplicável, pois os nós intermediários necessitam ter acesso a parte da mensagem. Além disso, ataques contra os protocolos de roteamento perturbam o desempenho e a confiabilidade da rede. Por isso, estão sendo propostos diversos protocolos de roteamento para redes espontâneas, que implementam mecanismos de segurança para os pacotes de controle e de dados.

Existem duas classes principais de ataques: passivos e ativos. Num ataque passivo, o nó atacante não envia mensagens, somente "escuta" as mensagens que estão sendo transmitidas. Esse tipo de ataque é uma ameaça maior para a privacidade da comunicação do que para o funcionamento dos protocolos de roteamento. Já os ataques ativos injetam mensagens na rede, além de escutar as mensagens que estão trafegando.

Os ataques a protocolos de roteamento podem ser divididos em ataques de rompimento da rota e ataques de consumo de recursos. Os ataques de rompimento da rota objetivam fazer com que os pacotes de dados trafeguem por uma rota errada (rota mais longa, rota em loop, rota comprometida, etc.). Já ataques de consumo de recursos objetivam injetar pacotes na rede de modo que aumente o consumo de recursos da rede e dos nós. Do ponto de vista da camada de aplicação, ambos os tipos de ataques são uma forma de ataque de negação de serviço (*denial-of-service* – DoS).

Um exemplo de ataque de rompimento do roteamento é a injeção, na rede, de pacotes de roteamento forjados de modo que se formem rotas em ciclo, evitando que os pacotes de dados atinjam o destino. Uma forma similar de ataque é forjar as mensagens de roteamento de forma que se estabeleça uma rota que passa por um nó específico, que irá capturar e descartar os pacotes de dados. Esse tipo de ataque é conhecido como *blackhole*. Um tipo particular de *blackhole* é quando o nó atacante descarta seletivamente os pacotes, deixando alguns passar. Esse ataque é chamado de *grayhole*.

Em protocolos de roteamento que tentam manter uma "lista negra" de nós mal intencionados, os atacantes podem fazer com que nós corretos sejam inseridos na lista negra, sendo evitados em rotas futuras. Controlar essa lista negra torna o protocolo mais complexo e aumenta o *overhead* de processamento.

Nos protocolos de roteamento sob demanda que tentam suprimir a retransmissão de pacotes duplicados, um ataque pode ser gerado disseminando-se pacotes *route request* rapidamente na rede. Isso faz com que os nós descartem pacotes legítimos recebidos após os pacotes forjados. Esse ataque chama-se *rushing attack*.

Um tipo de ataque mais robusto é a criação de um túnel na rede, com a utilização de dois atacantes em diferentes pontos da rede. Um deles captura os pacotes num ponto da rede, envia para o outro (através do túnel), e este retransmite os pacotes. Esse tipo de

ataque é conhecido como *wormhole*, e é um dos ataques mais difíceis de serem combatidos.

Um grande número de desafios ainda resta na área de segurança em redes sem fio espontâneas. Primeiramente, o problema de segurança nessas redes ainda não está bem modelado. Um modelo mais completo possibilitaria aos projetistas evoluírem seus protocolos de roteamento, e serviria de base para métodos formais de verificação da segurança dos protocolos. Outro desafio é desenvolver protocolos de roteamento eficientes, que sejam fortemente seguros e que não degradem excessivamente o desempenho da rede. Embora muitos pesquisadores desenvolverem extensões de segurança para protocolos existentes, essas extensões, muita vezes, eliminam otimizações de desempenho dos protocolos.

1.4.2.3 Descoberta de Serviços nas Redes Ad-hoc

Redes móveis espontâneas são caracterizadas por sua natureza altamente dinâmica, *multi-hop* e sem infra-estrutura. Num ambiente dinâmico, nós diferentes oferecendo diversos serviços podem entrar e sair da rede em qualquer tempo. Um protocolo de descoberta de serviços (*Service* Discovery) eficiente é um pré-requisito para melhor utilização dos recursos compartilhados na rede.

Descoberta de serviços é uma característica crucial para a usabilidade de uma rede móvel espontânea. Ela permite que dispositivos encontrem, automaticamente, serviços na rede e anunciem seus serviços para outros dispositivos.

Implementar descoberta de serviços em redes móveis espontâneas traz alguns desafios: (a) possibilitar que dispositivos sem fio com recursos limitados possam descobrir serviços dinamicamente; (b) possibilitar a descoberta de serviços em redes de larga escala; (c) possibilitar a conectividade entre redes móveis espontâneas e redes infra-estruturadas, e o compartilhamento de recursos entre elas.

As arquiteturas de descoberta de serviços podem ser divididas em *arquitetura sem diretório* e *arquitetura baseada em diretório*. Um diretório é uma entidade que armazena informação sobre os serviços disponíveis na rede.

A arquitetura baseada em diretório pode ser subdividida em *arquitetura com diretório centralizado* e *arquitetura com diretório distribuído*. Uma arquitetura com diretório centralizado utiliza um ou poucos diretórios para armazenar as descrições dos serviços disponíveis na rede. Enquanto que numa arquitetura de diretório distribuído, as informações sobre os serviços estão distribuídas em vários diretórios dinamicamente dispostos na rede.

Ainda, pode-se subdividir a arquitetura de diretório distribuído em *arquitetura de diretório distribuído sem infra-estrutura* e *arquitetura de diretório distribuído baseada em infra-estrutura*. Na primeira, os diretórios estão distribuídos nos nós da rede móvel. Enquanto que na segunda, os diretórios estão distribuídos em nós de uma rede infra-estruturada conectada à rede móvel espontânea.

Na arquitetura sem diretório, os provedores de serviço não distribuem as descrições de seus serviços para outros nós da rede, mas mantêm armazenados no próprio dispositivo. Dessa forma, os dispositivos requisitam os serviços através de *broadcast*, que é repassado por todos os nós da rede. Quando um dos nós atingidos pelo

broadcast pode prover o serviço requisitado, ele responde a mensagem para o requisitante. Esse mecanismo de *broadcast* não é praticável em redes móveis devido ao alto consumo de energia e largura de banda. Além disso, o tamanho da rede suportado é bem limitado.

Em contraste com essa arquitetura, a arquitetura com diretório centralizado utiliza um diretório central para armazenar a descrição de todos os serviços disponíveis na rede, permitindo a descoberta do serviço através de *unicast*. Essa arquitetura é melhor adaptável para redes sem fio estruturadas. Todavia, o diretório central é o gargalo do sistema.

A arquitetura de diretório distribuído é preferível quando se quer escalabilidade em redes grandes. No caso de arquitetura de diretório distribuído sem infra-estrutura, é assumido um grande número de nós em uma rede móvel. Por esse motivo, essa arquitetura é melhor adaptável às redes móveis espontâneas. Comparando com a arquitetura sem diretório, as maiores vantagens do uso de diretórios são (a) escalabilidade quando o tamanho da rede cresce, (b) redução do tempo de resposta para localizar serviços, (c) pode-se aplicar balanceamento de carga nos diretórios para diminuir o *overhead* e melhorar o desempenho dos diretórios.

Embora essa arquitetura seja melhor adaptável a redes móveis espontâneas, ela também pode ser aplicada a topologias híbridas, aplicando-se o conceito de *gateway*, que funciona como um diretório que interliga duas topologias distintas.

No caso da arquitetura de diretório distribuído baseada em infra-estrutura, é assumido que existem várias redes móveis, e que cada uma delas está ligada a uma rede fixa com uma estrutura de diretórios. Isto está baseado no fato de a maioria das redes móveis implementadas atualmente serem redes híbridas. Nessa arquitetura, um cliente pode solicitar um serviço para um nó de outra rede móvel que esteja conectada a ele por uma rede fixa. Neste caso, os diretórios de serviços encontram-se na rede fixa infra-estruturada.

1.4.2.4 Protocolo da Camada Transporte para Redes Ad-hoc

O *Transmission Control Protocol* (TCP) é um protocolo de transporte orientado a conexão, desenvolvido para prover entrega de pacotes de forma confiável e ordenada. O TCP foi projetado para ser independente da camada de enlace, podendo rodar sobre o protocolo IP em redes cabeadas e sem fio. Porém, experimentações em ambiente real têm mostrado que a implementação padrão do TCP não funciona adequadamente em redes sem fio espontâneas. Esses experimentos mostraram também que o aumento da mobilidade tem um impacto negativo no protocolo TCP.

Isso ocorre porque o TCP foi desenvolvido para redes cabeadas, com baixa taxa de erro e assumindo-se que as perdas de dados são devidas a congestionamento. Em contraste, nas redes sem fio existe alta taxa de erro e falhas na rota, que pode corromper os pacotes TCP e ACK. Então, ignorar essas características das redes sem fio pode levar a um baixo desempenho do TCP.

O problema ocorre, principalmente, porque a mobilidade e a atenuação do sinal causam a perda do enlace e enquanto o protocolo de roteamento está reconstruindo a rota, o mecanismo de recuperação do TCP continua retransmitindo os pacotes de dados

e incrementando (dobrando) o *timeout* para retransmissão. Assim, o nó não começa a transmitir os dados imediatamente após o restabelecimento da rota, aguardando o *timeout* definido no último erro de retransmissão.

Muitas propostas para melhorar o desempenho do protocolo TCP em redes sem fio têm sido feitas [Hanbali; Altman and Nain 2005]. Pode-se classificar essas propostas em dois tipos:

- protocolos que dividem a conexão de transporte. É usado em redes que combinam nós sem fio com nós cabeados. Essa abordagem insere um nó intermediário entre a rede móvel e a cabeada, que "*bufferiza*" os pacotes vindos da rede cabeada, fazendo o TCP acreditar que a rede móvel está trabalhando sem falhas. Porém, esse método quebra a semântica fim a fim (*end-to-end*) do TCP.
- protocolos que utilizam interação entre as camadas de comunicação (*cross-layer*). Usa uma abordagem de troca de informações entre as camadas de comunicação, onde o protocolo de roteamento ou da camada de acesso ao meio físico informa o de transporte sobre falhas na rota e contenções no canal. Essa idéia consiste em permitir que o TCP diferencie uma falha por congestionamento de uma por rota quebrada ou contenção da camada MAC.

A interação com outras camadas de comunicação pode ser de grande relevância para melhorar o desempenho do TCP em redes sem fio. A idéia em torno da proposta *cross-layer* é prover informações das camadas mais baixas para as mais altas, resultando em melhor desempenho de todo o sistema.

1.4.2.5 Protocolo de Acesso ao Meio Físico para Redes Ad-hoc

O projeto de um protocolo de controle de acesso ao meio físico (MAC) é uma questão importante para redes espontâneas [IEEE 1999][IEEE 2005]. O protocolo precisa tratar as limitações do canal, atenuação e ruído, bem como, prover um eficiente acesso ao meio considerando requisitos como qualidade de serviço (QoS), baixo consumo de energia e escalabilidade.

Os protocolos MAC para redes sem fio podem ser classificados em *contention-free* ou *contention-based*, dependendo da estratégia de acesso ao meio físico.

A abordagem *contention-free* predefine atribuições para permitir que os nós transmitam sem contenção do meio físico. Esse mecanismo é, geralmente, implantado para prover limites ao atraso fim a fim, privilegiando aplicações sensíveis ao atraso, como transmissões de áudio e vídeo. Exemplos de mecanismos dessa classe são TDMA, CDMA, FDMA, *polling*, e *token*.

O mecanismo *contention-based* é mais apropriado para transferências de dados esporádicas, comuns em redes móveis, devido a sua topologia aleatória e temporária. No padrão IEEE 802.11 é usado o esquema *Carrier Sense Multiple Access with Collision Avoidance* (CSMA/CA). Esse mecanismo escuta o meio antes de transmitir, para evitar colisões e implementa o esquema RTS/CTS (*Request-To-Send/Clear-To-Send*) para evitar um problema conhecido como *hidden terminal*. Esse problema é gerado quando um nó A está transmitindo para outro nó B, e um terceiro nó C, que está no alcance de B mas não no de A, escuta o canal, verificando que ele está livre. O nó C então transmite para B, causando colisão no recebimento das mensagens em B.

Dependendo da aplicação principal da rede, as abordagens com ou sem contenção podem influenciar diretamente no desempenho da rede.

Uma importante vantagem de uma abordagem *cross-layer* é a troca de informações entre o protocolo de roteamento e os protocolos da camada de acesso ao meio físico, o que permite melhor controle de consumo de energia durante a descoberta e manutenção das rotas.

1.4.2.6 Gerenciamento de Energia em Redes Ad-hoc

Dispositivos móveis são alimentados com baterias; então, as limitações de energia inerentes às baterias devem ser consideradas. Em redes móveis espontâneas, os nós precisam ser capazes de conservar energia para estender o tempo de vida das baterias.

Protocolos conscientes do estado da energia do sistema (*power-aware protocols*) podem utilizar três técnicas para reduzir o consumo de energia: (a) troca entre estados ativo e inativo, (b) configurações da potência do rádio, e (c) evitar retransmissão.

A troca entre os estados ativo e inativo evita o gasto de energia em períodos ociosos. A configuração da potência de transmissão pode ser feita de modo que o nó use o mínimo de energia necessário para que o destino receba a mensagem corretamente. E, finalmente, evitar retransmissões desnecessárias diminui a perda de energia.

Atualmente, os protocolos conscientes de energia implementam mecanismos de gerenciamento da energia ou de controle da energia. Nos mecanismo de gerenciamento de energia, os nós alternam entre períodos ativos e períodos de descanso. Os mecanismos de controle de energia adaptam a potência de transmissão de acordo com o mínimo necessário para a correta recepção da mensagem pelo destino.

O desenvolvimento de tecnologia para baterias tem avançado lentamente se comparado com a taxa de crescimento da velocidade de comunicação. Assim, é importante que várias camadas de comunicação tenham consciência do estado de energia do sistema, para poderem aplicar técnicas de redução do consumo de energia.

Em mecanismos de gerenciamento de energia é importante definir quando entrar em modo de descanso e quando voltar ao modo ativo, para não perder a conectividade. Já os mecanismos de controle de energia enfrentam os problemas de precisão nas medidas de potência e de variabilidade das condições do meio físico, como ruído e atenuação.

1.4.3 Redes de Sensores

Na computação móvel deseja-se obter um acesso contínuo às informações e outros recursos computacionais através de uma comunicação sem fio. Um dos tipos de aplicações móveis mais usuais hoje é aquela que emprega redes de sensores.

Uma definição para rede de sensores é a de uma rede sem fio formada por um grande número de sensores pequenos e imóveis plantados numa base para detectar e transmitir alguma característica física do ambiente. A informação contida nos sensores é agregada numa base central de dados. Outro enfoque que se pode ter de redes de sensores é de um conjunto de nós individuais (sensores) que operam sozinhos, mas que podem formar uma rede com o objetivo de juntar as informações individuais de cada

sensor para monitorar algum fenômeno. Estes nós podem se mover juntamente com o fenômeno observado. Por exemplo, sensores colocados em animais para observar seu comportamento. Ao observar o conjunto de sensores estar-se-ia monitorando toda a manada [Pereira, Amorin e Castro 2007].

Sensores podem ser dispositivos eletrônicos (sensores físicos) ou componentes de software (sensores lógicos) que obtém/medem algum tipo de informação. Sensores físicos hoje são utilizados para monitorar tanto ambientes de difícil acesso (oceano, vulcões, áreas de desastre, etc) quanto residências, áreas rurais (plantações e animais) e ruas (movimentação de pessoas e trânsito). Esses sensores, tipicamente, consistem de cinco componentes: detector de hardware, memória, bateria, processador embutido e transmissor-receptor.

Numa rede de sensores típica, os sensores individuais apresentam amostras de valores locais (medidas) e disseminam informação, quando necessário, para outros sensores e eventualmente para o observador [Hac 2003]. O acesso a essas informações pode ser sob demanda (reativo) ou pré-determinado pela aplicação (pró-ativo).

Para formar-se uma rede de sensores, têm-se três componentes: (i) sensores, (ii) protocolo de rede, para a comunicação entre sensores (propagação de dados) e com a aplicação, e (iii) aplicação/observador.

Redes de sensores são redes móveis e podem ser estáticas (infra-estruturadas) ou dinâmicas (ad-hoc). Os protocolos de roteamento *ad-hoc* podem ser usados como protocolos para redes de sensores, porém esses apresentam desvantagens devido às restrições de capacidade de processamento e energia dos sensores físicos. Pesquisas em andamento procuram gerar protocolos e soluções mais adequadas às redes de sensores.

Essa nova tecnologia de sensores cria um conjunto diferente de desafios provenientes dos seguintes fatores [Pereira, Amorin e Castro 2007]: (i) os nós encontram-se embutidos numa área geográfica e interagem com um ambiente físico; (ii) são menores e menos confiáveis que roteadores de redes tradicionais; (iii) geram (e possivelmente armazenam) dados detectados ao contrário de roteadores de rede e (iv) podem ser móveis.

Para o ambiente ubíquo, as redes de sensores sem fio desempenham um papel importante para detectar, coletar e disseminar informações dos objetos físicos/lógicos monitorados. Aplicações de sensores representam um novo paradigma para operação de rede, que têm objetivos diferentes das redes sem fio tradicionais.

1.4.4 Middlewares

Middlewares implementam as camadas sessão e apresentação do Modelo de Referência ISO/OSI e objetivam habilitar a comunicação entre componentes distribuídos. Para tal, fornecem aos programadores de aplicações abstrações de alto nível, construídas usando primitivas do sistema operacional de rede, que escondem a complexidade introduzida pela distribuição. Tecnologias de middlewares existentes têm sido construídas com a metáfora de caixa preta, onde a distribuição torna-se transparente ao usuário e ao projetista de software.

Essas tecnologias têm sido projetadas com sucesso para sistemas distribuídos estacionários, executando sob redes fixas. Porém, a simples adoção da tecnologia atual

de middlewares não é adequada ao ambiente móvel devido, principalmente, aos seguintes motivos: (i) as primitivas de interação, tais como transações distribuídas, requisições a objetos ou chamada remota de procedimento, assumem uma alta largura de banda e conexão permanente e disponível, o que contrasta com as características inerentes ao ambiente móvel; (ii) middlewares orientados a objetos, como Java-RMI (*Remote Method Invocation*), suportam principalmente comunicação ponto-a-ponto síncrona, a qual requer que o cliente solicite um serviço e o servidor o atenda, ambos executando simultaneamente, o que, novamente, contrasta com o ambiente móvel que requer anônima e assíncrona comunicação; (iii), ambientes distribuídos assumem que o ambiente de execução é estacionário, com confiável e alta largura de banda, localização fixa de cada *host* e serviços bem conhecidos, isto contrasta com o cenário altamente dinâmico proposto pela computação móvel, onde a localização dos host altera-se no tempo, e novos serviços podem ser descobertos dinamicamente enquanto este se move; (iv) a carga computacional para execução de middlewares é, geralmente, alta para ser carregada e executada em host móveis.

Outro aspecto a ser ressaltado é o relativo à questão transparência x consciência. Middlewares são construídos em abordagens que enfatizam a transparência, onde programadores não necessitam conhecer nenhum detalhe sobre o objeto ao qual o serviço é requerido. Enquanto que em sistemas distribuídos estacionários é possível (e desejável) esconder completamente as informações de contexto (por exemplo, localização) e detalhes de implementação da aplicação, em ambientes móveis isto se torna mais difícil e inadequado.

Para oferecer transparência, os middlewares tomam decisões em nome das aplicações, sacrificando a flexibilidade. No entanto, considerando as novas demandas das aplicações móveis e ubíquas é mais eficiente que as decisões sobre utilização dos recursos levem em conta informações específicas de cada aplicação, o host móvel e o ambiente de execução corrente. Em sistemas móveis é essencial que o middleware seja adaptativo, leve e (auto)reconfigurável.

Desta forma, é necessário o desenvolvimento de novas plataformas de middleware para atender aos requisitos impostos pela mobilidade. Por exemplo, middlewares que utilizam o paradigma de comunicação assíncrona através dos mecanismos de subscrição (*publish/subscribe*), são mais adequados às redes móveis.

Middlewares para fornecer **consciência de contexto** às aplicações móveis estão sendo tema de muitas pesquisas no momento. Alguns trabalhos desenvolvidos nessa área são:

1.4.4.1 AURA/CIS

Aura/CIS [AURA 2003] é um *middleware* que apresenta uma abordagem diferenciada quanto à forma de obtenção dos dados de contexto, pois utiliza uma base de dados onde todas as informações provindas de sensores são armazenadas e/ou obtidas diretamente dos sensores através do sistema de gerenciamento do banco de dados. A linguagem de consulta é similar à linguagem SQL, o que oferece facilidade na consulta aos dados de contexto pela aplicação; no entanto, a notificação de mudanças de contexto não pode ser registrada, cabendo então à aplicação controlar periodicamente as informações de contexto disponíveis e perceber quando alguma mudança de contexto ocorrer.

O mecanismo de interação da aplicação com o *middleware* se dá através de requisições utilizando a linguagem CSInt (*Contextual Service Interface*), a qual é fortemente baseada na linguagem SQL. Tais requisições são recebidas pelo middleware através dos *Query sysnthetizer*, que além de obter as informações, podem realizar algum processamento sobre elas para inferência de contexto de alto nível, por exemplo.

As informações de contexto, no sistema, podem ser de dois tipos: estáticas e dinâmicas. As informações de contexto estáticas são informações armazenadas em uma base de dados, e seu acesso é realizado conforme um banco de dados convencional, utilizando a linguagem de consulta SQL (*Simple Query Language*). Quanto às informações dinâmicas, estas são solicitadas diretamente à fonte no momento da consulta, em geral os sensores.

1.4.4.2 SOLAR

O sistema Solar [Chen 2004] é um *midleware* para o tratamento de informações de contexto. A ênfase dessa arquitetura é no tratamento de dados, e não em como esses dados são obtidos junto aos sensores.

Foram construídas duas versões do sistema Solar. Na primeira, utilizou-se uma arquitetura centralizada, onde um nodo é responsável por registrar e manter o catálogo de todos os recursos da rede, incluindo nodos e sensores. Em virtude da centralização observou-se a falta de escalabilidade dessa arquitetura. Então, foi criada uma nova versão, utilizando um catálogo de informações distribuído pela rede e um protocolo peer-to-peer, agindo de forma descentralizada.

A obtenção das informações dos sensores (coleta) se dá da seguinte forma: os sensores registram-se junto a um nodo da rede e fornecem um fluxo de dados com as informações que o sensor é capaz de gerar.

Como as informações obtidas por um único sensor não são diretamente úteis, algum tratamento deve ser realizado, seja através da agregação de dados de diferentes sensores, ou filtragem, ou inferência, enfim, toda a gama de operações que se deseja realizar sobre esses dados. Esse tratamento no sistema Solar é realizado pelos operadores definidos, que são entidades que residem nos *planets* (nós) da rede solar.

A disseminação das informações na rede ocorre através da implementação de um protocolo multicast, diferenciado do multicast IP. O protocolo implementado utiliza uma árvore de difusão, eliminando mensagens duplicadas aumentando a eficiência na transmissão.

Quando uma aplicação é ativada na arquitetura, ela informa, através de uma descrição XML (*Extended Markup Language*), em quais dados está interessada e quais operadores devem ser aplicados sobre esses dados. Essa descrição XML, realizada através de um arquivo, é mapeada diretamente na arquitetura e, caso, os operadores necessários não estejam já instanciados eles o serão.

Várias aplicações podem estar interessadas nos mesmos dados de alto nível, mas é difícil para o sistema perceber por si só essa relação. Para solucionar, a abordagem utilizada é a especificação de um nome para determinado contexto de alto nível. O problema gerado a partir de tal rotulação é que todas as aplicações deverão concordar com os nomes utilizados.

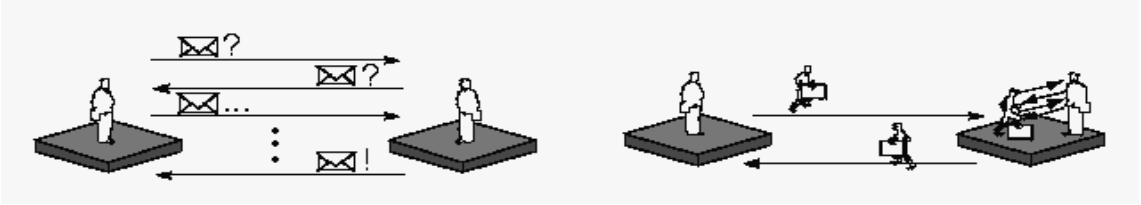
1.5 Programação de Aplicações Móveis

As principais ferramentas disponíveis hoje para o desenvolvimento de aplicações móveis estão relacionadas ao conceito de Agentes Móveis, os quais implementam a mobilidade de código, e ferramentas para programação, como as disponibilizadas pela plataforma Micro Edition da linguagem Java.

1.5.1 Agentes Móveis

Um agente móvel é um programa que pode migrar (mover-se) de um computador para outro, transportando seu código, seus dados e seu estado, para continuar a execução no computador destino. As principais características de um agente móvel são: autonomia, habilidade social, aprendizagem e mobilidade.

Agente móvel é uma forma específica do paradigma de agentes e de softwares de código móvel (Figura 1.5). Porém, os agentes móveis são mais flexíveis, na perspectiva de que eles podem decidir mover-se em qualquer instante da execução. Isso os torna uma poderosa ferramenta para implementar aplicações distribuídas ou móveis.



Modelo Troca de Mensagens Modelo Agentes Móveis

Figura 1.5. Modelo Agentes Móveis

Assim, os agentes móveis não estão restritos ao sistema em que iniciaram sua execução, pois eles têm a habilidade de se transportar de um sistema para outro através de uma rede. Este recurso permite um agente móvel mover-se para o sistema que possui um objeto com o qual o agente deseja interagir, obtendo a vantagem de residir na mesma máquina ou rede do objeto.

O principal paradigma utilizado atualmente em sistemas de objetos distribuídos é baseado na passagem síncrona de mensagens entre objetos estacionários que interagem utilizando este mecanismo. Entretanto, esse paradigma é incompleto, e necessita ser melhorado adicionando-se alguns recursos como passagem de mensagem assíncrona, mobilidade de objetos e objetos ativos.

Agentes móveis podem oferecer um paradigma uniforme para Objetos Distribuídos, englobando passagem de mensagens síncronas e assíncronas, passagem de objetos e objetos móveis e estacionários. Além de suportar os serviços existentes em uma rede, agentes móveis também tornam possíveis novos serviços e novas oportunidades de negócios.

As principais vantagens dos agentes móveis sobre agentes convencionais são:

- **Redução do tráfego da rede -** Agentes móveis auxiliam a redução do tráfego na rede, pois permitem despachar tarefas que podem executar suas interações localmente. Eles podem ainda reduzir o tráfego de dados na rede, pois permitem mover o processamento para o local onde os dados estão armazenados ao invés de transferir os dados para depois processá-los. O princípio é "Mover o processamento para os dados ao invés de mover os dados para o local de processamento";
- **Oculta a latência da rede** - Sistemas críticos necessitam de respostas em tempo real para mudanças no ambiente. Agentes móveis oferecem uma solução, pois podem ser despachados pelo controlador central para realizarem suas tarefas localmente;
- **Encapsulamento de protocolo** - Cada máquina em um sistema distribuído possui seu próprio código necessário para implementar a transferência de dados. Porém, novos requisitos de segurança e eficiência demandam mudanças no protocolo que podem ocasionar problemas na manutenção do código existente. Agentes móveis, por outro lado, podem mover-se para máquinas remotas a fim de estabelecer canais de comunicação baseados em protocolos proprietários;
- **Execução assíncrona e autônoma** – As tarefas podem ser embutidas em agentes móveis que podem ser despachados pela rede. Após serem despachados, os agentes são autônomos e independentes da criação de processo, podendo executar assincronamente. Este recurso é útil principalmente porque um dispositivo móvel pode se reconectar na rede para coletar o agente mais tarde;
- **Adaptação dinâmica** - Agentes móveis possuem a habilidade de perceber mudanças no ambiente de execução e reagir autonomamente. Múltiplos agentes podem interagir entre si e se distribuir pela rede, de modo a manter uma configuração ótima para resolver um problema em particular;
- **Independência de plataforma** - Redes de computadores, geralmente são heterogêneas, tanto na perspectiva de hardware como a de software. Agentes móveis são independentes da máquina e também da rede, sendo dependentes somente do seu ambiente de execução, não dificultando a integração de sistemas;
- **Robustez e tolerância a falhas** - A habilidade dos agentes móveis de reagirem dinamicamente a situações e eventos desfavoráveis torna fácil a construção de sistemas distribuídos robustos e tolerantes a falhas. Se uma máquina está para ser desligada, todos os agentes em execução na máquina podem ser advertidos para que possam ser despachados e continuar suas tarefas em outra máquina da rede;

O aspecto que mais diferencia entre a migração de agentes e a transferência de código móvel (também chamado de código sob demanda) é que, quando migrando entre *hosts*, os agentes transferem não apenas o código, mas também o estado da sua execução. Sendo assim, no contexto de agentes móveis, tende a desaparecer a diferença entre clientes e servidores, e todas as máquinas que suportam a execução de agentes são chamadas de hosts. Por isso, o modelo computacional de agentes móveis pode ser considerado um caso particular de tecnologia de código móvel.

Atualmente, há uma série de projetos e produtos que implementam suporte a agentes móveis, como, por exemplo, Telescript, Odyssey, Mole, Concórdia, Voyager, TACOMA, Aglets e Grasshopper. O conceito de agentes móveis vem sendo proposto para suportar diferentes tipos de aplicações, dentre elas: comércio eletrônico, gerenciamento de fluxo de trabalho, gerenciamento de redes, serviços de telecomunicação, recuperação de informações distribuídas e redes ativas [Aridor and Lange 1998].

Agentes móveis são uma atraente alternativa para construções de ambientes distribuídos e móveis, principalmente em aplicações como comércio eletrônico, sendo que suas potencialidades podem ser utilizadas da seguinte forma:

- *Observação*: em um ambiente de investimento um cliente pode despachar um agente móvel para monitorar um mercado eletrônico até que determinado produto atinja um determinado preço, assim retornando e notificando o usuário.
- *Busca*: o usuário pode delegar a um agente a tarefa de procurar um produto a um determinado preço, assim o agente cliente movimenta-se para um servidor para interagir com outros agentes (vendedores) para a busca do melhor preço.
- *Organização*: em uma aplicação voltada para o entretenimento, um cliente pode delegar a um agente a tarefa de organizar uma agenda. Com isso o agente deve interagir e buscar a melhor solução para cobrir todos os requisitos determinados pelo cliente.

1.5.1.1 Aglets

Aglets [Lange and Oshima 1998] é uma plataforma de agentes móveis em Java, que facilita o desenvolvimento de aplicações baseadas em agentes. Um aglet é um agente Java capaz de migrar de um host a outro autonomicamente. Originalmente, a tecnologia Aglets foi desenvolvida pelo IBM Tokio Research Laboratory, e se chamava Aglets WorkBench, sendo a IBM responsável pelas versões 1.X. Porém, atualmente chama-se apenas Aglets, e é um projeto de código aberto hospedado na sourceforge.net, encontrando-se na versão 2.0.2.

A implementação de Aglets segue o princípio de desenvolvimento de *frameworks*, da mesma forma que se desenvolvem Applets e Servlets em Java. Frameworks são esqueletos de uma arquitetura de sistema, suportados por uma biblioteca (API) orientada a objetos. Em conformidade com o princípio de desenvolvimento de *frameworks*, o programador que desenvolve Aglets implementa uma série de métodos com nomes pré-definidos, os quais são invocados sobre o agente durante a ocorrência de eventos característicos do ciclo de vida deste agente.

Os principais métodos que constituem o *framework* de programação de um Aglet são (http://aglets.sourceforge.net/):

- `onCreate()` ou `onClone()` - invocado sobre o agente logo após sua criação;
- `onDisposing()` - invocado sobre o agente antes de sua destruição;
- `onDispatching()` e `onArrival()` - invocado sobre o agente antes e depois de sua migração entre *hosts,* respectivamente;

- `onDeactivating()` e `onActivation()`- invocado sobre o agente antes e depois de seu estado de execução ser movido entre a memória e um meio de armazenamento permanente, respectivamente.
- `handleMessage()` - invocado sobre o agente quando ele recebe uma mensagem de outro agente ou dele mesmo;
- `run()` - invocado sobre o agente na criação de seu próprio *thread* de execução.

O *framework* algets foi completamente desenvolvido em Java, e por isso, possui alta portabilidade. Esse *framework* inclui uma plataforma para agentes móveis, um servidor autônomo chamado Tahiti, e uma biblioteca que permite desenvolver agentes móveis e embutir a tecnologia Aglets em outras aplicações.

O servidor Tahiti é mostrado na Figura 1.6. O botão "Create" exibe uma lista com os agentes disponíveis. A criação de um agente invoca, automaticamente, o método da API `onCreate()`, que deve ser implementado pelo agente. O botão "Dialog" não está associado ao nenhum método. Ao ser ativado, esse botão envia uma mensagem "dialog" ao agente selecionado. Essa mensagem pode ser tratada no método `handleMessage()`, sendo geralmente usada para mostrar uma interface gráfica. O botão "AgletInfo" exibe informações sobre o agente selecionado. O botão "Dispose" invoca o método `onDisposing()` e encerra o agente, O botão "Clone" cria outro agente idêntico ao selecionado (contendo os mesmo dados e estado de execução), e invoca o método `onClone()`. O botão "Dispatch" serve para enviar o agente para outro host. Além disso, a ativação desse botão invoca o método `onDispatching()`. Finamente, o botão "Retract" permite selecionar um agente, em determinado servidor, para trazê-lo de volta.

O método `onArrival()` é invocado toda vez que um agente chega ao um host. Os métodos `onDeactivating()` e `onActivation()` são invocados através do menu "Mobility", que permite desativar um agente por um tempo determinado, após o qual ele será automaticamente reativado, ou reativá-lo manualmente. A implementação dos métodos da API não é obrigatória. Esses métodos servem para facilitar o tratamento de eventos.

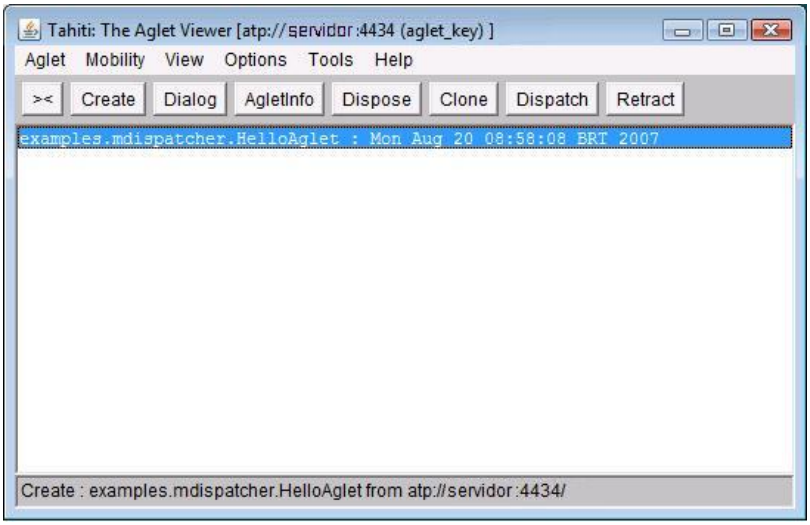


Figura 1.6. Servidor Tahiti

Dessa forma, Aglets são objetos Java que possuem a capacidade de mover-se de uma máquina para outra em uma rede, levando consigo o código do programa e o estado dos objetos que compõem o Aglet. A migração ocorre com a interrupção da execução do aglet na máquina origem, seu despacho para uma máquina remota e o reinicio da execução após sua chegada ao destino. Mecanismos de segurança impedem que aglets não autorizados (*untrusted)* tenham acesso a determinados recursos do sistema, oferecendo certo nível de segurança.

O código Java é simples, seguro, compacto e, por ser orientado a objetos, permite o reuso de código e a exploração de recursos como interfaces, encapsulamento e polimorfismo. As características de Java úteis a um ambiente baseado em agentes móveis são relacionadas a seguir:

- **Independência de plataforma** - O código Java é compilado em um formato independente de arquitetura chamado *byte code*, permitindo a execução de aplicações Java em redes heterogêneas. Isto permite criar agentes móveis sem conhecimento prévio de qual tipo de computadores na qual eles irão executar.
- **Execução segura** - Java possui diversos mecanismos de segurança. Os programas Java são proibidos de acessar dados privados de objetos. Este fato torna possível construir um ambiente seguro a ataques de agentes móveis mal intencionados.
- **Carga dinâmica de classes** - A máquina virtual Java carrega e define classes em tempo de execução, fornecendo um espaço de endereçamento privado para cada agente, que pode executar independentemente e com segurança em relações a outros agentes.
- **Programação *multithread* -** O modelo de programação *multithread* permite a implementação de agentes como entidades autônomas. As primitivas de sincronização nativas da linguagem Java habilitam a interação entre agentes.
- **Serialização de objetos** - A linguagem Java permite a serialização de objetos. Este mecanismo permite que objetos sejam empacotados com informação suficiente que permitam posterior reconstrução. Esta é uma característica chave para implementação de agentes móveis.
- **Reflexão** - Java possui mecanismos para obter informações sobre classes carregadas, permitindo construir agentes com maior conhecimento de si próprios e de outros agentes.

No é apresentado um exemplo de implementação de um Aglet. Neste caso, apenas são mostrados os métodos mais relevantes, pois os outros são apenas métodos auxiliares que não apresentam nenhuma característica de agente.

Pode-se observar que um aglet deve estender a classe Aglet. Devido à migração dos agentes, todos os membros (variáveis) de um aglet devem ser serializáveis. Por isso, o membro “my_dialog”, que representa uma interface gráfica criada para o agente e que não é serializável, foi declarado como *transient*. Dessa forma, esse membro não irá migrar com o aglet. Porém, após o aglet migrar para outro host, sua interface só poderá ser invocada se houver uma implementação da classe neste host.

```
public class HelloAglet extends Aglet {
        transient Frame my_dialog = null;       // não serializado
        String message = null;
        String home = null;
        SimpleItinerary itinerary = null;

        public boolean handleMessage(Message msg) {
                if (msg.sameKind("atHome")) {
                        atHome(msg);
                } else if (msg.sameKind("startTrip")) {
                        startTrip(msg);
                } else if (msg.sameKind("sayHello")) {
                        sayHello(msg);
                } else if (msg.sameKind("dialog")) {
                        dialog(msg);
                } else {
                        return false;
                }
                return true;
        }
        public void onCreation(Object init) {
                setMessage("Hello World!");    // mensagem padrão

                createGUI();   // Cria GUI para controlar o Aglet
                itinerary = new SimpleItinerary(this);
                home = getAgletContext().getHostingURL().toString();
      }
        public synchronized void startTrip(Message msg) {
                String destination = (String)msg.getArg();
                // Migra para o destino e
                // envia mensagem "sayHello" para ele mesmo (this)
                try {
                        itinerary.go(destination, "sayHello");
                } catch (Exception ex) {
                        ex.printStackTrace();
                }
        }
}
```

Quadro 1 - Exemplo de Aglet

No método `handleMessage()`, são tratados os tipos de mensagens esperados pelo aglet. Nesse exemplo, para cada mensagem que o aglet recebe é invocado um método auxiliar que faz o processamento necessário e a migração do agente. O método `onCreate()`, que é chamado logo após a criação do agente, inicializa a mensagem padrão, a interface gráfica, o itinerário (que nesse caso está vazio) e uma variável com o endereço do host de origem (para poder voltar). O itinerário é usado quando se tem vários hosts para visitar. Finalmente, o método `startTrip()` é chamado pela interface gráfica, e inicia a migração do agente.

Em síntese, mobilidade de código [Vigna 1998] já é um tópico consolidado e disponível em linguagens atuais como Java. Agentes Móveis é, também, um conceito dominado e disponível em aplicações reais. Porém, permitir e gerenciar a mobilidade de todos os componentes do espaço pervasivo permitindo às aplicações expressarem a semântica 'siga-me' é ainda um tema em aberto [Augustin 2005]. Novas abordagens são necessárias para permitir a flexibilidade e adaptabilidade tanto dos dispositivos móveis quanto das entidades de software. Em nível de programação, este conceito está diretamente relacionado à adaptação dinâmica ao contexto.

1.5.2 Plataforma Java Micro Edition (J2ME)

Para a programação de aplicações de propósito geral, os dispositivos móveis dispõem de algumas ferramentas. As mais utilizadas hoje são: (i) APIs para o sistema operacional Symbian, presente em muitos telefones celulares, tais como os da Nokia; (ii) BREW (*Binary Runtime Environment for Wireless*), ferramenta proprietária da Qualcomm que utiliza C++ e frameworks; (iii) .NET Mobile da Microsoft, e (iv) Java Micro Edition.

Java Micro Edition (J2ME), uma das três partes que compõem a plataforma Java, como ilustra a Figura 1.7, disponibiliza um eficiente ambiente para o desenvolvimento e execução de aplicações para dispositivos móveis como Personal Digital Assistants (PDA), telefones celulares e dispositivos embarcados. Assim como as outras partes, J2ME compõe-se de uma máquina virtual Java e de um conjunto de Application Programming Interface (APIs).

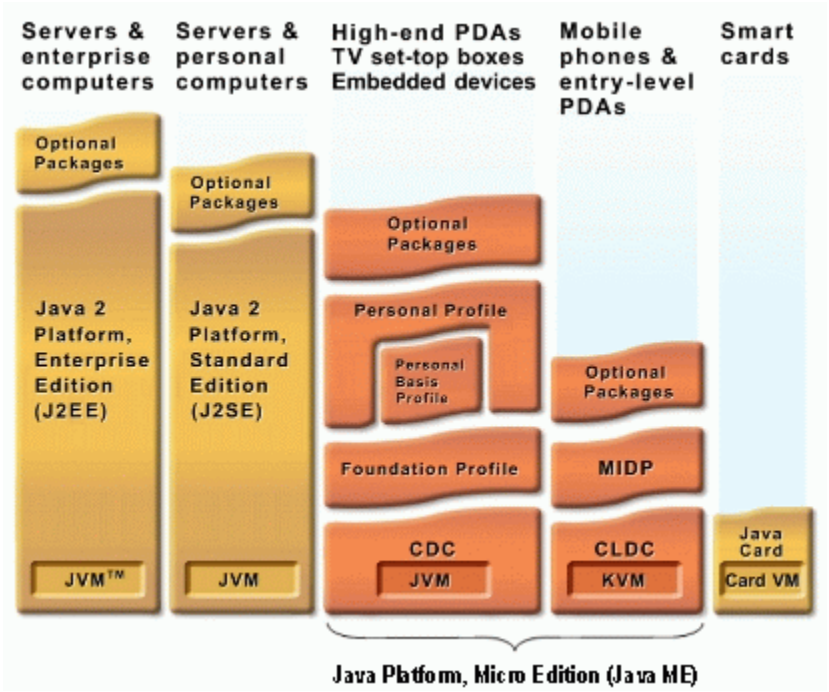


Figura 1.7. Plataforma Java

Para atender a diversidade de dispositivos móveis, a arquitetura do J2ME engloba uma variedade de configurações, perfis e pacotes opcionais que os desenvolvedores podem combinar para construir aplicação Java adequada a determinado dispositivo e atendendo aos requisitos exigidos. Cada combinação é otimizada para uma

categoria de tamanho de memória, poder de processamento e capacidade de entrada e saída.

Configurações abrangem uma máquina virtual e um conjunto mínimo de classes de biblioteca. Elas provêem as funcionalidades básicas para uma gama de dispositivos que compartilham características similares, como conectividade e capacidade de memória. Atualmente, existem duas configurações Java Microedition: (i) *Connected Limited Device Configuration* (CLDC), definem as funcionalidades comuns aos dispositivos com restrições de recursos, tais como os telefones celulares; e (ii) *Connected Device Configuration* (CDC), definem as funcionalidades comuns aos dispositivos de maior capacidade, tais como os PDAs baseados em Mobile Windows e Linux.

A fim de prover um ambiente de execução completo para uma categoria específica de dispositivos, uma configuração deve ser combinada com um **perfil** – um conjunto de APIs de alto nível que melhor define o acesso às propriedades específicas dos dispositivos. Um perfil dá suporte a um subconjunto de dispositivos de uma determinada configuração. A combinação amplamente adotada é a configuração CLDC com o perfil *Mobile Information Device Profile* (MIDP) que provê um ambiente de execução para telefones celulares e outros dispositivos (PDAs) com capacidades similares a eles.

A plataforma J2ME pode ser estendida através da adição de **pacotes opcionais** à pilha de tecnologias que inclui tanto CLDC ou CDC. Criados para resolver diversos requisitos de aplicações, os pacotes opcionais oferecem APIs padrão para usar tecnologias existentes ou emergentes como conexão a base de dados, multimídia, Bluetooth e *web services* (ver Figura 1.8).

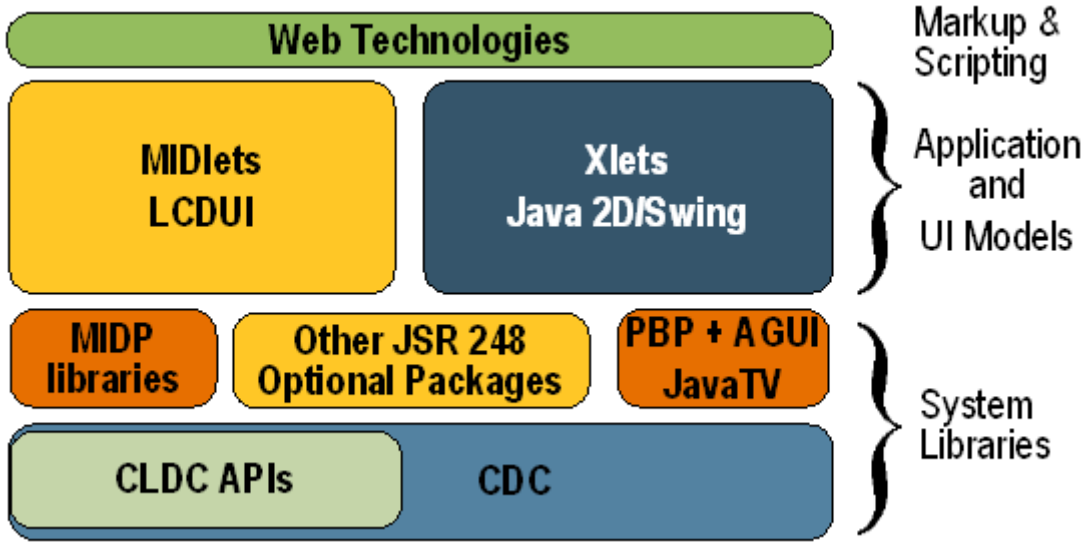


Figura 1.8. Pilha de Tecnologias do Microedition

1.5.2.1 Programando com J2ME – CLDC/MIDP

Para criar programas destinados a dispositivos móveis e portáteis utilizando J2ME, utiliza-se a API Java `javax.microedition` que contém inúmeras classes que permitem a implementação de programas em Java. Essa plataforma considera todas as

possíveis limitações que dispositivos móveis possuem, como as relativas à memória, processamento e consumo de energia.

Aplicações implementadas em CLDC/MIDP possuem três estados que são:

- Ativo: realiza-se a aquisição de recursos que serão necessários e, ainda, inicializa-se a execução (chamada do método `startApp`);
- Pausado: realiza-se a liberação de recursos em um modo de espera (chamada do método `pauseApp`);
- Destruído: libera-se todos os recursos (chamada do método `destroyApp`).

A Figura 1.9 ilustra as possíveis transições de estados acionadas pelos métodos.

A aplicação CLDC/MIDP é uma aplicação gerenciada pelo *Application Manager* (AMS), o qual implementa as funcionalidades adequadas a cada dispositivo móvel e gerencia sua execução, e é chamada **MIDlet**.

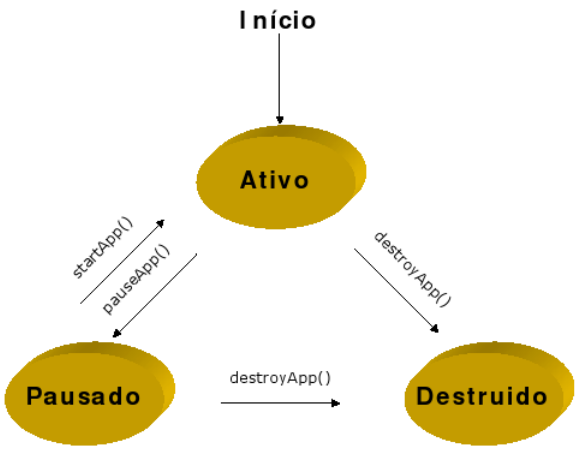


Figura 1.9. Estados do MIDP

A programação é realizada criando-se uma subclasse da classe `Midlet`, e preenchendo-se os métodos para transição de estado. A transição entre os estados (ciclo de vida) é gerenciada pelo AMS (`midp.exe`). A adequação dos elementos definidos para a interface gráfica às restrições da tela do dispositivo é também responsabilidade do AMS. Abaixo está o exemplo de uma aplicação simples implementada em J2ME/MIDP.

```
import javax.microedition.midlet.*;
import javax.microedition.lcdui.*;

public class OlaMidlet extends MIDlet implements CommandListener{

    private Display display; //Referencia para o objeto Display
    privateTextBox tbMain; //TextBox para mostrar a mensagem
    private Command cmSair; //Botão para sair do MIDlet

    public OlaMidlet(){
         display = Display.getDisplay(this);
         cmSair = new Command("Sair", Command.SCREEN, 1);
         tbMain = new TextBox("Bem Vindo","OlaMidlet", 50, 0);
```

```
            tbMain.addCommand(cmSair);
            tbMain.setCommandListener(this);
    }

    //Chamada do gerenciador de aplicação para iniciar o Midlet.
    public void startApp(){
      display.setCurrent(tbMain);
    }

    public void pauseApp(){}
    public void destroyApp(boolean unconditional) { }

    // Checa se o comando de saída foi selecionado
    public void commandAction(Command c, Displayable s) {
      if (c == cmSair) {
        destroyApp(false);
        notifyDestroyed();
      }
    }
}
```

Conforme o exemplo acima, percebe-se que a aplicação utiliza os pacotes javax.microedition.midlet que inclui a classe base para o MIDLET a ser implementado provendo os três métodos para o controle dos possíveis estados da aplicação, e também a `javax.microedition.lcdui` que fornece diversas elementos para a criação de interfaces gráficas a ser utilizados nos aplicativos MIDP. Além disso, têm-se alguns métodos que são disponibilizados para a comunicação com o AMS. No código acima dois desses possíveis métodos são utilizados como o `destroyApp()` que destruirá a aplicação, e o `notifyDestroyed()` que notifica ao gerenciador que a aplicação está sendo terminada.

Outros pacotes permitem (i) gerenciar o armazenamento em arquivos, usando o conceito de registro (`javax.microediton.rms`), (ii) realizar a comunicação com a rede (`javax.microedition.io`) usando um framework genérico para conexões (GCF) para permitir a aplicação usar qualquer protocolo com a mesma interface de conexão; são implementados vários protocolos (http – padrão, htpps, ssl, socket, file, etc), (iii) acessar dados armazenados no dispositivo em código nativo (`javax.microedition.io.pim`), (iv) envio de mensagens sms (`javax.wireless.messaging`), (v) sistemas baseados em localização (`javax.microedition.location`), etc.

1.6 Conclusões

As restrições naturais do ambiente móvel colocam novos desafios para os projetistas de aplicações e exigem novas tecnologias para que as aplicações sejam úteis em sistemas com recursos limitados. Sente-se a necessidade de sistemas mais flexíveis que dividam a responsabilidade entre o projetista da aplicação e o sistema de suporte (*middleware*) para fornecer o comportamento dinâmico e adaptativo que a aplicação requer.

Em nossa visão, novas linguagens de programação deverão ser desenvolvidas nas quais programas sejam construídos pelos usuários dinamicamente; porém, neste momento esforços estão sendo feito a partir de linguagens existentes para se entender os

problemas e as necessidades envolvidas na Computação Pervasiva. A próxima geração de programas terá de ser sempre disponível (*always-on*), dinamicamente adaptável, em constante interação com numerosos outros programas residindo em diferentes máquinas virtuais e físicas.

As ferramentas disponíveis para a programação de aplicações móveis estão incorporando novos conceitos e facilidades de gerenciamento do ambiente móvel. Há uma tendência ao uso de padrões como forma de amenizar a heterogeneidade dos dispositivos móveis (PDAs, telefones celulares, smartphones).

1.7 Referências

Aridor, Y. and Lange, D. B. (1998) "Agent Design Patterns: Elements of Agents Application Design". In *Proceedings of the Second International Conference on Autonomous Agents*.

Augustin, I. et al (2005) "Managing the Follow-me Semantics to Build Large-scale Pervasive Applications". In *Middleware Conference 2005 - Workshop on Middleware for Ad-hoc and Pervasive Computing*, Grenoble, France.

Augustin, I. (2004) "Abstrações para uma Linguagem de Programação visando Aplicações Móveis em um Ambiente de Pervasive Computing". Tese de Doutorado, II/UFRGS, Porto Alegre, janeiro.

Augustin, I. et al. (2004a) "ISAM, Joing Context-awareness and Mobility to Building Pervasive Applications", *Mobile Computing Handbook*. Mahgoub, I. and Ilyas, M. Editors, CRC Press, New York.

Augustin, I et al. (2002) "Towards Taxonomy for Mobile Applications with Adaptive Behavior". In *International Symposium on Parallel and Distributed Computing and Networks* (PDCN 2002), Innsbruck, Austria. feb.

AURA (2003). Project Aura - Distraction-free Ubiquitous Computing. Disponível em http://www-2.cs.cmu.edu/~aura/internal.html. Acesso em agosto de 2007.

Cardelli, L. and Gordon, A. D. (1998) "Mobile Ambients". In First International Conference on Foundations of Software Science and Computation Structure, mar. p.140-155.

Chalmers, D. et al. (2006) "Ubiquitous Computing: Experience, Design and Science", http://www-dse.doc.ic.ac.uk/Projects/UbiNet/GC/index.html, abril.

Chen, G. (2004) "Solar: Building A Context Fusion Network for Pervasive Computing", PhD Thesis. Dartmouth College, Hanover, New Hampshire, USA.

Cho, C. and Lee, D. (2005) "Survey of Service Discovery Architectures for Mobile Ad Hoc Networks", http://folk.uio.no/paalee, agosto.

Cunha, D. d.; Costa, L. H., and Duarte, O. C. (2004) "Analyzing the Energy Consumption of IEEE 802.11 Ad Hoc Networks". In *Conference on Mobile and Wireless Communication Networks* .

Garlan, D. and Steenkiste, P. and Schmerl, B. (2002) "Project Aura: Toward Distraction-free Pervasive Computing". In *IEEE Pervasive Computing*, New York, v.1, n.3, sept.

Hac, A. (2003) “Wireless Sensor Network Designs”, John Wiley & Sons, dec., 391 p.

Hanbali, A. A.; Altman, E., and Nain, P. (2005) “A survey of TCP over Ad Hoc Network”. In *IEEE Communications Surveys & Tutorials* , v.7, n.3, p.22-36.

Hu, Y.-C. and Perring, A. (2004) “A survey of secure wireless ad hoc routing”. In *IEEE Security & Privacy Magazine* , v.2, n.3, p.28-39.

IEEE (1999). “Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications”. ANSI/IEEE Std 802.11.

IEEE (2005). “Part 15.1: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Wireless Personal Area Networks (WPANs)”. IEEE Std 802.15.1.

INFOEXAME (2007). “Smartphones – porque é a hora de comprar um e aposentar seu celular”. n. 257, agosto.

Jansen, E. et al (2005) “A Programming Model for Pervasive Spaces”, In *International Conference on Service-Oriented Computing*, Netherlands, dec.

Karlof, C. and Wagner, D. (2003) “Secure Routing in Wireless Sensor Networks: Attacks and Countermeasures”. *Elsevier Ad Hoc Networks Journal*, v. 1, n.2-3, p.293-315.

Lange, D. B. and Oshima, M. (1998) “Mobile agents with Java: The Aglet API”. *World Wide Web* ,v. 1, n.3, p.111-121.

Pereira, M. R; Amorim, C. L e Castro, M. C. S. “Tutorial sobre Redes de Sensores”. http://magnum.ime.uerj.br/cadernos/cadinf/vol14/3-clicia.pdf

Ranganathan, A. et al (2005) “Towards a Pervasive Computing Benchmark”. In *3rd International Conference on Pervasive Computing and Communications Workshops* (PerCom), New York, IEEE Computer Society.

Roman, M. et al. (2002) “Gaia: a Middleware Infrastructure to Enable Active Spaces”, *IEEE Pervasive Computing*, New York, v.1, n. 4, dec.

Saha, D. and Mukherjee, A. (2003) “Pervasive Computing: a Paradigm for the 21st Century”, *IEEE Computer*, v.36, n.3, p.25-31, mar.

Satyanarayanan, M. (2001) “Pervasive Computing: Vision and Challenges”, *IEEE Personal Communications*, New York.

The Yankee Group. “Divergent Approach to Fixed/Mobile Convergence”. November 2004.

Vigna, G. (1998) “Mobile Code, Technologies, Paradigms, and Applications”. Tesi di Dottorato, Politecnico di Milano, Italy.

Weiser, M. (1991) “The Computer of the 21st Century”. *Scientific American*, New York, v.265, n.9, sep.

Wikipedia - Convergência Tecnológica (2007). http://pt.wikipedia.org/wiki/Convergência_tecnológica, agosto.

Wikipedia - Mobile Agent. (2007). http://en.wikipedia.org/wiki/Mobile_agent, agosto.

LISTA DE ABREVIATURAS

sigla	significado
AMS	Application Management System
API	Application Programming Interface
BAN	Body Area Network
CDMA	Code Division Multiple Acess
CDC	Connected Device Configuration
CLDC	Connected Limited Device Configuration
FDMA	Frequency Division Multiple Access
GSM	Global System Mobile
GPS	Global Positioning System
GPRS	General Packet Radio Service
IPTV	Internet Protocol Television
ISO/OSI	International Standard Organization's Open System Interconnect
MAC	Medium Access Control
MANET	Mobile Ad Hoc Network
MIDP	Mobile Information Device Profile
MPLS	Multiprotocol Label Switching
PAN	Personal Area Network
PCS	Personal Communication Service
PDA	Personal Digital Assistant
RDSI	Rede Digital de Serviços Integrados
SSH	Secure Shell
SIP	Session Initiation Protocol
SSL	Secure Sockets Layer
TCP / IP	Transmission Control Protocol / Internet Protocol
TDMA	Time Division Multiple Access
VoIP	Voz sobre IP (Internet Protocol)
xDSL	notação generalizada da família de protocolos Digital Subscriber Line
WAP	Wireless Application Protocol
WLAN	Wireless Local Area Network
WiFi	Wireless Fidelity (IEEE 802.11b)