Computação Ubíqua: Princípios, Tecnologias e Desafios

Regina Borges de Araujo

Departamento de Computação – Universidade Federal de S. Carlos (UFSCar)
Caixa Postal 676 – 13565-905 – São Carlos - SP – Brasil

regina@dc.ufscar.br

Abstract. *The basic idea behind ubiquitous computing is that computing moves out of the workstation and personal computers to become pervasive in ours day to day lives. Marc Weiser, considered the father of ubiquitous computing, shimmered one decade ago that, in the future, computers would inhabit the most trivial objects: from clothes tags, coffee mugs, light interrupters, pen, watches, etc., in an invisible way to the user. This text discusses some of the main issues and technologies involved in this Weiser´s world, in which we should learn to live with computers and not only interact with them.*

Resumo. *A idéia básica da computação ubíqua é que a computação move-se para fora das estações de trabalho e computadores pessoais e torna-se pervasiva em nossa vida cotidiana. Marc Weiser, considerado o pai da computação ubíqua, vislumbrou há uma década atrás que, no futuro, computadores habitariam os mais triviais objetos: etiquetas de roupas, xícaras de café, interruptores de luz, canetas, etc, de forma invisível para o usuário. Este texto discute as principais questões e tecnologias envolvidas neste mundo de Weiser, em que devemos aprender a conviver com computadores, e não apenas a interagir com eles.*

Prefácio

A convergência das tecnologias de rádio, dos microprocessadores e dos dispositivos eletrônicos digitais pessoais está levando ao conceito de ubiqüidade no qual dispositivos inteligentes, móveis e estacionários, coordenam-se entre si para prover aos usuários acesso imediato e universal a novos serviços, de forma transparente, que visam aumentar as capabilidades humanas.

Os termos computação pervasiva, computação ubíqua e computação móvel são, muitas vezes, utilizados como sinônimos por algumas pessoas, entretanto, será visto que eles são conceitualmente diferentes. A computação ubíqua integra mobilidade em larga escala com a funcionalidade da computação pervasiva. O potencial de aplicações da computação ubíqua é limitado apenas pela imaginação - com a conexão, monitoramento e coordenação de dispositivos localizados em casas, edifícios e carros inteligentes, através de redes sem fio locais e de longa distância com alta largura de banda, as aplicações variam desde o controle

de temperatura, luzes e umidade de uma residência, até aplicações colaborativas com suporte à mobilidade.

Pesquisas em computação ubíqua estão sendo realizadas por pesquisadores do mundo todo em tópicos que vão de protótipos de rede que provêem acesso básico a qualquer tipo de dispositivo sem fio, suporte à mobilidade na rede, de forma transparente, segurança, tratamento de contexto, otimização de espaço de armazenamento, largura de banda e uso de energia; formatação, compressão, entrega e apresentação de conteúdo multimídia que se adapta a diferentes condições de largura de banda e de recursos de dispositivos; até a adaptação da aplicação e da apresentação multimídia aos dispositivos do usuário etc.

O objetivo deste texto é dar uma visão geral das questões envolvidas nesta área emergente de pesquisa.

Agradecimentos

Gostaria de agradecer a todos os meus alunos de mestrado do Programa de Pós Graduação em Ciência da Computação do DC da UFSCar, que estão trabalhando em algumas das questões discutidas aqui, e com quem tive várias discussões sobre o assunto. Agradecimentos especiais para Taciana Novo Kudo, Richard Werner Nelém e Alessandro Rodrigues da Silva pela contribuição em algumas partes do texto.

1. Introdução

A idéia básica da computação ubíqua é que a computação move-se para fora das estações de trabalho e computadores pessoais (PCs) e torna-se *pervasiva*[1] em nossa vida cotidiana. Marc Weiser, considerado o pai da computação ubíqua, vislumbrou há uma década atrás que, no futuro, computadores habitariam os mais triviais objetos: etiquetas de roupas, xícaras de café, interruptores de luz, canetas, etc, de forma invisível para o usuário. Neste mundo de Weiser, devemos aprender a conviver com computadores, e não apenas interagir com eles [1].

Dois cenários são utilizados para ilustrar o conceito de computação ubíqua, as tecnologias envolvidas e os desafios a serem vencidos. O primeiro cenário descreve o potencial da computação ubíqua para aumentar as capabilidades dos profissionais. O segundo cenário ilustra as diferentes visões que se pode ter de uma mesma aplicação na dimensão da computação ubíqua. A apresentação desses cenários demonstra algumas das questões envolvidas nesta desafiadora área emergente de pesquisa.

Cenário 1. Potencializando as Capabilidades da Jovem Executiva (adaptada de [2])

Janete é a diretora de uma organização que depende de serviços de computação e está participando de uma reunião importante na matriz de sua empresa, em São Paulo. Ela está em uma sala de conferencia com três colegas de trabalho e dois participantes remotos que

[1] O termo “pervasivo” não existe no vocabulário português. Entretanto, para evitar confusão com outros termos também emergentes na área, “pervasivo” será usado neste texto como a tradução do termo em inglês “*pervasive*”, cuja definição é dada no texto principal.

participam através de um sistema de videoconferência por computador. O sistema permite que esta equipe de Janete veja e ouça os participantes remotos bem como use um quadro branco compartilhado para edição de documentos e exploração de dados em conjunto. Infelizmente Janete tem que deixar a reunião mais cedo, pois terá que pegar um avião para visitar um fornecedor em Paris. Felizmente ela pode continuar a participar da reunião através de seu PDA inteligente. Tão logo ele (PDA) detecta que ela saiu da sala de reuniões, ele redireciona a parte de áudio da reunião para o telefone celular dela. Quando ela entra no carro que vai levá-la ao aeroporto, o seu PDA procura por uma tela maior a sua volta e acha a tela embutida atrás do banco do motorista. O PDA então passa a exibir os fluxos de vídeo da reunião na tela do carro, incluindo as pessoas e o quadro branco compartilhado. Além disso, a porção de áudio da reunião é transferida para o sistema de alto-falantes do carro. Conforme o carro se movimenta rumo ao aeroporto, o vídeo se adapta automaticamente para refletir a mudança de qualidade da rede, alterando a sua resolução.

Conforme a reunião progride, uma mensagem instantânea é mostrada na tela informando Janete sobre um incêndio na central de dados da empresa em Pernambuco. Ela toca na tela sensível a toque sobre a mensagem com o objetivo de discar o numero de telefone do transmissor da mensagem. O transmissor, o gerente da central de dados, relata que o fogo na central danificou seriamente alguns servidores importantes, mas que felizmente os *back-ups* estão seguramente armazenados em um outro local. Janete decide mudar seus planos e voar para Pernambuco imediatamente para ajudar nos reparos. Ela pede ao gerente da central de dados para enviar a ela um relatório detalhado dos danos sofridos para que ela possa planejar os reparos. Ela ativa seu agente pessoal de software usando o teclado embutido no carro e solicita ao agente para mudar as suas reservas de vôo de Paris para Pernambuco. Ela também pede que todos os arquivos referentes à configuração da central de dados sejam transferidos para o seu PDA assim que a largura de banda de rede necessária esteja disponível. O agente rearranja os vôos. Além disso, o agente infere que ela não será capaz de cumprir seus compromissos em Paris e então cancela a reserva de hotel usando os serviços on-line do hotel, bem como as reuniões com o fornecedor através do envio de e-mail apropriado de notificação. Assim que Janete sai do carro, o agente pessoal de software "salta" do computador do carro para o PDA e varre a área a procura de uma conexão de rede melhor. Conforme ela entra no terminal do aeroporto, o agente detecta uma LAN sem fio na qual ele se autentica e começa a baixar os dados solicitados sobre a configuração da central de dados.

Enquanto Janete espera pelo seu vôo no terminal, ela chama o gerente da central de dados para ouvir os últimos acontecimentos. Assim que ela embarca no avião, seu PDA novamente varre a área em busca de possíveis serviços em seu ambiente e detecta que ele pode se conectar a tela e teclado embutidos no assento do avião usando uma rede sem fio interna ao avião. O PDA mostra o ambiente de trabalho (desktop) de Janete na tela enfatizando as informações da central de dados. Depois do avião decolar, Janete planeja cuidadosamente os reparos necessários, usa os serviços de compra rápida para pedir as partes necessárias de equipamentos e/ou software e solicita *free-lancers* com as habilidades necessárias para aliviar o trabalho de seus próprios funcionários que estão trabalhando 24 horas por dia. O agente pessoal do PDA automaticamente prioriza suas solicitações e transfere alguns arquivos via a rede lenta do avião. Quando o avião pousa e Janete vai para o

terminal de desembarque, o PDA pode transferir arquivos usando a LAN sem fio do terminal. Quando Janete chega na central de dados, as primeiras ofertas de *free-lancers* já chegaram, e os fornecedores de equipamentos já receberam os pedidos para substituição das partes danificadas.

Cenário 2. Uma aplicação para vários dispositivos (adaptado de [3])

Um geólogo do Instituto Americano de Geologia foi mandado para um local remoto no oeste dos EUA para examinar os efeitos de um terremoto recente. Usando um PC, e um software denominado MANNA[2], que suporta as atividades do geólogo em qualquer dispositivo que ele usar, o geólogo baixa mapas e relatórios existentes sobre a área de modo a se preparar para a visita. Como o PC não é um equipamento móvel e o geólogo precisa viajar, os documentos são transferidos para um *laptop* e o geólogo pega um avião até o local onde ocorreu o terremoto.

Neste avião não há suporte para rede, portanto o *laptop* desabilita a conexão de rede e oferece apenas computação local. Quando o geólogo examina os vídeos do local, a interface do usuário chaveia automaticamente para monitor branco e preto e reduz a taxa de quadros por segundo, para ajudar a conservar bateria. Ao chegar ao aeroporto, o geólogo aluga um carro e dirige até o local. Ele recebe uma mensagem através do sistema MANNA pelo celular, alertando-o para examinar um local em particular. Como o fone celular oferece espaço de tela extremamente limitado, o mapa da região não é mostrado. Em vez disso, o celular mostra a localização geográfica, orientações de como chegar lá, e a posição atual do geólogo (GPS). Um recurso que permite ao geólogo responder a mensagem também é oferecido pelo software.

Chegando no local, o geólogo usa o seu *palmtop* para tomar notas sobre a região. Como o *palmtop* conta com uma caneta de toque (styllus) para interação, não é permitida a interação através de cliques duplos ou cliques à direita. Mais ainda, como o tamanho da tela também é um problema no *palmtop*, um *layout* mais conservador é adotado para ser mostrado ao geólogo. Ao completar a investigação, o geólogo prepara uma apresentação em dois formatos. Primeiro uma apresentação do seu percurso pelo local, com anotações, é feita para o HUD *(heads-up display)*. Como o HUD tem capacidade limitada para tratar entrada de texto (usa o próprio dispositivo móvel), a aplicação MANNA oferece interação baseada em voz. Uma outra apresentação, mais convencional, é preparada para ser visualizada em telão. Como o propósito da apresentação no telão não é o de interação, mecanismos de interação são removidos.

Questões observadas nos cenários acima descritos:

- A informação é acessada através de múltiplos dispositivos heterogêneos
- A aplicação segue o usuário em movimento
- Os dispositivos interagem entre si

[2] MANNA – aplicação multimídia que deve executar em varias plataformas e que pode ser usada de forma colaborativa via Internet.

- Algumas tarefas são executadas de forma autônoma
- Dispositivos diferentes apresentam visões diferentes da mesma aplicação
- O ambiente troca informações com os dispositivos e vice-versa
- A aplicação responde a mudanças no ambiente

Questões que devem ser respondidas para que os cenários acima tornem-se cenários amplamente difundidos:

- Como mudar de uma rede para outra de forma transparente, sem que isto seja refletido na aplicação?
- Como fazer com que dispositivos "descubram" outros dispositivos ao seu redor e interajam entre si para a realização de serviços?
- Como fazer com que um ambiente inteligente aprenda sobre os dispositivos e as ações do usuários no ambiente e reflita na aplicação esse contexto aprendido.
- Como projetar aplicações acessadas/executas de/em diferentes dispositivos sem ter que projetar as interfaces e funcionalidades para cada dispositivo separadamente?

Estas e outras questões perfazem o assunto deste livro e serão discutidas no decorrer dos próximos capítulos.

1.1 Definições

Por ser uma área emergente de pesquisa, termos como computação ubíqua, computação pervasiva, computação nomádica, computação móvel e outros tantos, têm sido usados muitas vezes como sinônimos, embora sejam diferentes conceitualmente e empreguem diferentes idéias de organização e gerenciamento dos serviços computacionais. Na medida em que a área evolui, esses conceitos vão sendo melhor compreendidos e suas definições se tornam mais claras. A definição e diferenciação entre estes conceitos são mostradas a seguir [4].

O que é Computação Móvel?

A computação móvel baseia-se no aumento da nossa capacidade de mover fisicamente serviços computacionais conosco, ou seja, o computador torna-se um dispositivo sempre presente que expande a capacidade de um usuário utilizar os serviços que um computador oferece, independentemente de sua localização. Combinada com a capacidade de acesso, a computação móvel tem transformado a computação numa atividade que pode ser carregada para qualquer lugar.

Limitações da Computação móvel

Uma importante limitação da computação móvel é que o modelo computacional não muda enquanto nos movemos, isto é, o dispositivo não é capaz de obter flexivelmente informação sobre o contexto no qual a computação ocorre e ajustá-la corretamente. Numa solução para acomodar a mudança de ambiente, os usuários poderiam manualmente controlar e configurar a aplicação à medida que se movem, o que seria inviável e inaceitável pela maioria dos usuários

O que é Computação Pervasiva?

O conceito de computação pervasiva implica que o computador está embarcado no ambiente de forma invisível para o usuário. Nesta concepção, o computador tem a capacidade de obter informação do ambiente no qual ele está embarcado e utilizá-la para dinamicamente construir modelos computacionais, ou seja, controlar, configurar e ajustar a aplicação para melhor atender as necessidades do dispositivo ou usuário. O ambiente também pode e deve ser capaz de detectar outros dispositivos que venham a fazer parte dele. Desta interação surge a capacidade de computadores agirem de forma "inteligente" no ambiente no qual nos movemos, um ambiente povoado por sensores e serviços computacionais.

O que é Computação Ubíqua?

A computação ubíqua beneficia-se dos avanços da computação móvel e da computação pervasiva. A computação ubíqua surge então da necessidade de se integrar mobilidade com a funcionalidade da computação pervasiva, ou seja, qualquer dispositivo computacional, enquanto em movimento conosco, pode construir, dinamicamente, modelos computacionais dos ambientes nos quais nos movemos e configurar seus serviços dependendo da necessidade. A tabela 1 mostra as dimensões da computação ubíqua.

	Computação Pervasiva	**Computação Móvel**	**Computação Ubíqua**
Mobilidade	Baixa	Alta	Alta
Grau de "embarcamento"	Alto	Baixo	Alta

Tabela 1 – Dimensões da Computação Ubíqua – adaptado de [4]

O grau de embarcamento indica, de maneira geral, o grau de inteligência dos computadores, embutidos em um ambiente pervasivo, para detectar, explorar e construir dinamicamente modelos de seus ambientes.

Como se relacionam as computações ubíqua, pervasiva e móvel?

Tendo em vista todas as definições mencionadas acima, o termo computação ubíqua será usado aqui como uma junção da computação pervasiva e da computação móvel. A justificativa de se realizar uma diferenciação desses termos é que um dispositivo que está embutido em um ambiente, não necessariamente é móvel. Devido a isso, quando for utilizado o termo *computação ubíqua*, considerar-se-ão o alto grau de dispositivos embarcados da computação pervasiva juntamente com o alto grau de mobilidade da computação móvel, conforme mostra a Figura 1.

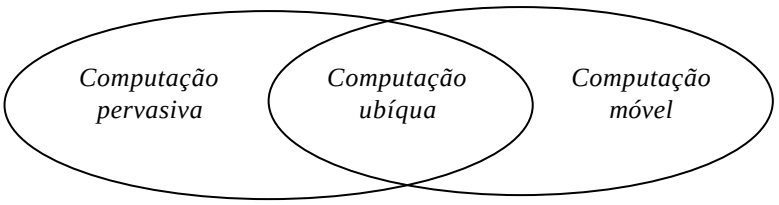

Figura 1 – Relação entre Computação Ubíqua, Pervasiva e Móvel

1.2 Princípios da Computação Ubíqua (adaptado de [5])

Pelos menos três princípios são identificados na Computação Ubíqua, a saber:

Diversidade - Ao contrario do PC, que é um dispositivo de propósito geral que atende varias necessidades distintas do usuário, tais como: edição de texto, contabilidade, navegação na web, etc., os dispositivos ubíquos acenam com uma nova visão da funcionalidade do computador, que é a de propósito especifico, que atende necessidades especificas de usuários particulares. Apesar de vários dispositivos poderem oferecer funcionalidades que se sobrepõem, um pode ser mais apropriado para uma função do que outro. Por exemplo, o *palmtop* é bom para fazer anotações rápidas, mas não é o melhor dispositivo para navegar na web. Um telefone de tela pode ser o dispositivo escolhido pelo usuário em sua casa para navegar pela internet, com todo o potencial de cores e boa resolução que este dispositivo pode oferecer. Um outro aspecto da diversidade é o de como gerenciar as diferentes capabilidades de diferentes dispositivos, uma vez que cada dispositivo tem características e limitações próprias tornando difícil oferecer aplicações comuns.

Descentralização - Na computação ubíqua as responsabilidades são distribuídas entre vários dispositivos pequenos que assumem e executam certas tarefas e funções. Estes dispositivos cooperam entre si para a construção de inteligência no ambiente, que é refletida nas aplicações. Para isso uma rede dinâmica de relações é formada, entre os dispositivos e entre dispositivos e servidores do ambiente, caracterizando um sistema distribuído. Quando os dispositivos têm que atualizar informações com os servidores ou com outros dispositivos, em especial com dispositivos de diferentes capacidades, questões de sincronização emergem e serão discutidas no capitulo sobre sincronização. Outra questão relacionada à distribuição inclui o gerenciamento, pelos servidores, das aplicações que executam nos dispositivos. Os servidores precisam manter registros de perfis de usuários e de dispositivos com capacidades diferentes. Além disso, o servidor deve ser altamente flexível, bem como poderoso, para tratar um vasto número de dispositivos em movimento.

Conectividade - Na computação ubíqua, tem-se a visão da conectividade sem fronteiras, em que dispositivos e as aplicações que executam neles movem-se juntamente com o usuário, de forma transparente, entre diversas redes heterogêneas, tais como as redes sem fio de longa distância e redes de media e curta distancia. Para que se atinja a conectividade e interoperabilidade desejada é preciso basear as aplicações em padrões comuns, levando ao desafio da especificação de padrões abertos.

1.3 Aplicações

Uma das primeiras demonstrações de como a computação ubíqua pode ajudar os usuários em suas tarefas diárias foi realizada nos laboratórios da XeroxPARC com o *Xerox PARCTab* [6], que localizava um usuário dentro de um edifício, dentre outras funcionalidades. Outras áreas de aplicação da computação ubíqua incluem: ensino, trabalho colaborativo, residências e automóveis inteligentes, descritas logo abaixo.

Exemplos de Computação Pervasiva no Ensino

O objetivo é dar suporte aos professores em suas atividades de produção de material didático e aos alunos na anotação das aulas de forma personalizada. Exemplos de projetos nesta área incluem: o *Classroom 2000* [7] do *Georgia Institute of Technology* que compreende um conjunto de tecnologias de hardware e software para a captura de aulas presenciais e posterior disponibilização do material capturado sob a forma de hiperdocumentos multimídia customizados; o *Lecture Browser* [8] é um projeto da Universidade de *Cornell* que difere do *eClass* ao utilizar técnicas de visão computacional e combinação de mais de uma fonte de vídeo para produzir seus hiperdocumentos multimídia resultantes do processo de captura da aula dada pelo professor; o *Authoring on the Fly* [9] é um projeto de suporte ao ensino da Universidade de *Freiburg* na Alemanha que difere de outros sistemas dessa natureza em função do poderoso modelo de sincronização das mídias capturadas da aula, das diversas formas de representação do material capturado, e das funcionalidades fornecidas para acesso posterior a esse material.

Exemplos de Computação Pervasiva no Trabalho Colaborativo

Oferece suporte a reuniões formais e/ou informais entre participantes de um mesmo grupo. Exemplos de projetos nesta área incluem: o sistema *DUMMBO* (*Dynamic, Ubiquitous, Mobile Meeting Board*) [10], do *Georgia Institute of Technology* voltado para a captura de atividades de uma reunião informal. Esse sistema registra os desenhos feitos em uma lousa eletrônica compartilhada e a voz de cada membro da reunião, e disponibiliza essa informação sob a forma de hiperdocumentos multimídia sincronizados; o sistema *TeamSpace* [11] que é um projeto colaborativo entre o *Georgia Institute of Technology*, IBM e Boeing com foco na captura de atividades envolvidas em uma reunião formal local ou distribuída. Esse sistema registra uma parcela maior dos artefatos apresentados durante uma reunião, incluindo slides, anotações, itens de agenda, vídeo e áudio de cada membro da reunião; o projeto *iRoom* (*Interactive Room*) [12] da Universidade de *Stanford* que focaliza as interações com dispositivos computacionais, tais como PCs, laptops, PDAs e superfícies eletrônicas, em um espaço de trabalho conectado a uma rede sem fio. Um de seus principais desafios envolve o suporte à movimentação de dados e de controle entre esses vários dispositivos de interação de forma transparente, para não interromper o processo de colaboração.

Exemplos de Computação Ubíqua em Aplicações que exigem maior movimentação do usuário

Guias turísticos eletrônicos trabalham normalmente com informações de localização dos usuários e utilização de dispositivos portáteis para conferir maior mobilidade aos usuários. Os

principais objetivos dessa classe de aplicações são registrar os locais visitados e identificar a posição atual do usuário dentro de um espaço de interação, como campus universitário, museu, etc. O *CampusAware* [13] é um sistema de turismo que utiliza informações de localização para fornecer serviços aos usuários. O sistema permite que o usuário faça anotações em seus dispositivos sobre os locais que estão visitando, fornecendo também informações de como encontrar determinados locais de interesse e a posição atual do usuário. O *CampusAware* possui um repositório central com três bancos de dados para informações contextuais e sociais, como as anotações dos usuários e locais visitados. O *CampusAware* utiliza o GPS para obtenção de informações de localização em ambientes abertos. O sistema *Rememberer* [14] é parte do projeto *Cooltown* [15], dirigido pelo *Hewlett-Packard Laboratories*. Seu objetivo é capturar experiências pessoais e estimular discussões ou outras formas de interação pessoal através de dispositivos portáteis sem fio. Para capturar informação de localização em recintos fechados, esse sistema utiliza RFID (*radio frequency identification)* [16]. Um exemplo de aplicação desse sistema é em visita a museus, onde câmeras fotográficas registram interações dos usuários no museu, e disponibiliza essas informações na ordem em que os locais foram visitados, permitindo ainda que o usuário faça anotações sobre esses lugares.

Exemplos de Computação Ubíqua na Residência Inteligente

Basicamente, as aplicações de computação ubíqua que abrangem o domínio doméstico têm por objetivo conhecer as atividades dos moradores de uma casa e fornecer serviços que aumentem a qualidade de vida deles. O *EasyLiving* [17] é um projeto da *Microsoft Research* que se preocupa com o desenvolvimento de arquiteturas e tecnologias para ambientes inteligentes, focando particularmente em uma sala de estar residencial. O ambiente contém um computador, telas eletrônicas (incluindo uma tela grande), caixas de som, sofás e mesa de café, entre outros itens. Os serviços são fornecidos para melhorar o ambiente, como por exemplo, automatizar o controle de luz, tocar música baseado na localização (dependendo da preferência do usuário), e ainda transferir automaticamente o conteúdo de uma tela para outra. O *Adaptive House* [18] é um projeto da Universidade de Colorado e tem como objetivo desenvolver uma casa que se programe observando o estilo de vida e desejos dos habitantes e aprendendo a se antecipar às suas necessidades. O sistema desenvolvido no *Adaptive House* é chamado *ACHE* (*Adaptive Control of Home Environments*), e foi desenvolvido basicamente para controlar o sistema de AVAC (Aquecedor, Ventilador, Ar condicionado), iluminação e água.

Exemplos de Computação Ubíqua em outros domínios

Outros domínios de aplicação são automóveis, penitenciárias, laboratórios, etc. No domínio automotivo a atenção principal é na posição do usuário/dispositivo. Segundo Herrtwich [19] o setor automotivo é um bom atrativo para a computação ubíqua, pois os dispositivos de comunicação já podem estar integrados nos automóveis (devido ao seu tamanho), os equipamentos de comunicação podem utilizar as fontes de energia do próprio automóvel, o preço dos equipamentos de comunicação é relativamente pequeno comparado com o valor do automóvel, e o mais importante, é que muitos serviços, como pedido de socorro e rastreamento remoto, são de interesse dos compradores e dos produtores de automóveis.

1.4 Considerações finais

Novas arquiteturas devem ser criadas para tratar da diversidade e das limitações dos dispositivos ubíquos e de outras tecnologias envolvidas.

Entretanto, com o aumento do potencial de processamento e armazenamento de dispositivos que estão cada vez mais baratos, a emergência de discos minúsculos, processadores milimétricos, células de energia em miniatura, telas pequenas com resolução cada vez melhor, além dos avanços no reconhecimento da fala e da escrita manual, do aumento das redes de comunicação sem fio com capacidade cada vez maior, do surgimento de padrões abertos para conexão e interoperabilidade entre redes diferentes, a computação ubíqua não parece tão distante e certamente vai causar um enorme impacto na sociedade e no estilo de vida das pessoas. O próximo capitulo descreve diferentes tipos de dispositivos.

2. Dispositivos

Os cenários mostrados no capitulo 1 oferecem uma pequena amostra do potencial de aplicação da computação ubíqua. Na medida em que a residência, o meio de transporte e o trabalho do usuário tornam-se os cenários de aplicação da computação ubíqua, oferecendo serviços como segurança, comodidade, informação, entretenimento, e muito mais, um universo incrível de dispositivos diferentes deverá existir para suportar estes serviços. Exemplos incluem: controles, sensores e atuadores para residências e automóveis, eletrodomésticos, ar-condicionado, aquecedor, relógios e etiquetas inteligentes, além de toda a linha branca de utensílios domésticos, TVs, celulares, PDAs, consoles de jogos e muito mais.

A existência de múltiplos dispositivos, seja para o acesso a informação, ao entretenimento, embutidos em utensílios domésticos, embarcados em ambientes inteligentes, etc., constitui-se em um dos desafios da computação ubíqua, conforme visto no capitulo 1. Uma tentativa de classificar esses dispositivos é ilustrada na tabela 2. Por questão de falta de espaço, será apresentado, no texto a seguir, apenas um resumo das características principais de cada classe de dispositivos [5] [20].

2.1 Controles Inteligentes

Os controles inteligentes caracterizam-se por serem muito pequenos podendo ser integrados a lâmpadas, interruptores, termostatos, sensores, atuadores etc., em aplicações que variam de controle de segurança residencial (sensores em portas e janelas para detectar a entrada de intrusos, atuadores para acender/apagar lâmpadas específicas em horários específicos etc), a controle de comodidades para o usuário (sensores de temperatura, atuadores para ligar/desligar/programar sistemas de aquecimento/resfriamento de ambiente residencial etc.). Os controles são conectados a redes domesticas e gerenciados local ou remotamente, através da Web ou applets Java, em aplicações locais.

Cartões Inteligentes

Os cartões inteligentes *(smartcards)* merecem uma seção a parte dentro dos controles inteligentes, devido ao alto potencial de uso, em larga escala, em aplicações do tipo: Autenticação de usuário, assinatura eletrônica e criptografia. O cartão inteligente pode funcionar como um cartão telefônico (nos telefones GSM), sistema de acesso a um local físico de trabalho, em aplicações de *home banking*, bolsa eletrônica (dinheiro digital) e muito mais. A próxima geração de cartões para transações financeiras possuem confiabilidade muito superior ao cartão magnético – não dá para copiar. Estas aplicações podem executar fora e/ou no próprio cartão.

Controles Inteligentes	**Utensílios Inteligentes**
Controles de processo de manufatura	Quiosques
Controles residenciais	Terminais de Ponto de Venda
Termostatos	Centrais de telecomunicações
Etiquetas inteligentes	Eletro domésticos da linha branca (geladeiras,
Controles de bombas de ar, água, gás etc.	maquinas de lavar roupa etc.)
Cartões inteligentes	Terminais de Caixa eletrônico
Controles em sistemas automotivos	Maquinas de venda automática
	Instrumentos de monitoramento medico

Dispositivos de Acesso a Informação	**Sistemas de Entretenimento**
Telefones Celulares	TV
Telefones de tela	Caixas digitais – *set-top-box*
Agentes Pessoais Digitais - PDAs	Consoles de jogos
Computadores de mão	Câmeras digitais
Computadores portáteis	Brinquedos inteligentes
Pagers	Reprodutores de musica MP3

Tabela 2. Uma classificação dos dispositivos ubíquos (adaptado de [5])

Um cartão inteligente é composto basicamente de um *chip* de no máximo 25 mm^2 de tamanho, com os seguintes componentes e respectivas configurações típicas: CPU de 8, 16 ou 32 bits (co-processador opcional para criptografia), Memória RAM de 4KB, ROM de 16KB que contém o sistema operacional e funções de comunicação e de segurança (DES, RSA), além de EEPROM para dados permanentes (aplicação). Os Sistemas operacionais existentes para cartões inteligentes incluem: JavaCard, Mutos e Windows para smartcards. As propriedades físicas, elétricas, mecânicas e de programação (sistema de arquivos) dos cartões inteligentes são especificadas pelo Padrão ISO 7816 [21]. A padronização da interface de acesso às aplicações é também um aspecto importante para que aplicações possam ser acessadas por leitoras diferentes ou levadas para leitoras de outros fabricantes. Iniciativas de padronização de APIs para aplicações de cartões inteligente incluem: PC/SC [22] e *OpenCard Framework* – OCF [23].

2.2 Utensílios Inteligentes

É difícil estabelecer uma linha divisória clara entre as classes de dispositivos, em especial os controles inteligentes e os utensílios inteligentes. Os controles inteligentes podem estar embutidos em utensílios que são controlados por um ou mais microprocessadores. Desta forma, podemos considerar os utensílios como sendo mais inteligentes e complexos do que os controles inteligentes. Os utensílios inteligentes aumentam a funcionalidade de equipamentos já conhecidos e que hoje prestam um conjunto de serviços dedicados. Os utensílios inteligentes, em ambientes de computação ubíqua, interagem entre si para aumentar o conforto dos usuários, seja em casa, no carro, no escritório, no banco, no hospital, nas ruas, nos *shoppings* etc. Exemplos de aplicação de utensílios inteligentes incluem: Otimização do consumo de energia (aquecimento seletivo – por cômodo da casa; aquecimento diferenciado por ocupação da casa; aquecimento de água em função do perfil dos moradores); manutenção de utensílios (diagnóstico e atualização remota de micro-código em utensílios da linha branca), comunicação entre etiquetas de roupa e a máquina de lavar. Esses dispositivos podem ser acessados e operados remotamente via web, por exemplo.

Computação Automotiva

Um numero cada vez maior de microprocessadores, controladores, sensores e atuadores estão sendo embutidos nos vários componentes dos automóveis de mais alto custo. Atualmente, estes componentes interagem entre si, mas em breve estarão se comunicando também com o mundo exterior para estender a interface tradicional do motorista e oferecer serviços do tipo: sistemas de navegação (uso do GPS para determinação de localização do veiculo com o objetivo de orientar o motorista sobre as melhores rotas para chegar ao destino); telemática (componentes do carro comunicam-se com componentes de fora do carro, através de comunicação sem fio, para informação sobre alternativas de rotas face a congestionamentos, bloqueios, condições climáticas etc); informe sobre acidente ou falhas em componentes do automóvel (problemas no *air bag* são notificados para a montadora via fone celular); monitoramento dos dados sobre o veiculo pelas montadoras (temperatura do óleo, informes da montadora para o motorista, envio de atualizações de software); acesso a informação(serviços de e-mail, acesso a Internet); entretenimento (difusão de áudio digital – DAB; TV) e muito mais.

Três fatores da indústria automotiva estão levando as montadoras a implementarem em seus carros sistemas genéricos de barramento que interconectam componentes eletrônicos, através de barramento e não mais de fiação com interconexão ponto-a-ponto [20]: (1) aumento da complexidade da montagem do veiculo devido ao peso da fiação necessária para interconectar um numero cada vez maior de componentes eletrônicos, aumentando também o custo do veiculo; (2) o custo cada vez maior dos componentes personalizados para cada fabricante; (3) a diferença grande de tempo entre os ciclos de desenvolvimento dos dispositivos eletrônicos e do modelo de um carro novo – o sistema eletrônico de um carro novo que entra em linha de produção já está defasado em termos de custo e tecnologia em relação a nova geração de eletrônicos disponível nas prateleiras.

Com o barramento, que age como uma rede de comunicação, diferentes componentes, com interfaces bem definidas, podem ser conectados. Estas interfaces têm que

ser padronizadas para que os componentes possam ser desenvolvidos por terceiros, e de forma independente do carro em si.. Existem varias iniciativas de padronização das redes automotivas em andamento, tais como: os americanos J1850 definido pela Sociedade dos Engenheiros Automotivos e *Onboard Diagnostics* - ODB-II, o europeu Controller área Network – CAN, o padrao aberto *Local Interconnect Network* – LIN, o barramento IDB – *Intelligente Transportation Systems Data Bus*, entre outros.

2.3 Dispositivos de Acesso a Informação

Os dispositivos de acesso à informação provêem comunicação entre usuários, anotação, acesso à informação etc., visando aumentar as capabilidades humanas, em especial no trabalho. Exemplos desse tipo de dispositivo incluem: Assistentes Digitais Pessoais – PDA, PC de bolso (baseados no W/CE), *Sub-notebooks* (intermediários entre computador de mão e *lap-top)*, Telefone celular; Telefones inteligentes (combinam telefone móvel com PDA), Telefones de tela (convergência do telefone com terminal de acesso à internet), etc.

Dispositivos de Mão

Os primeiros dispositivos moveis de acesso á informação eram descendentes dos organizadores e possuíam teclado e tela relativamente grandes, com vários conectores e *slots* de expansão (exemplos: Psion e W/CE *Handheld Pro*). Já os PDAs caracterizam-se pela tela sensível a toque. A interação com o dispositivo é baseada em apontador *(stylus)*, a entrada de texto é feita por reconhecimento de escrita manual (exemplos: Palm, WorkPad da IBM, Pocket PC etc.). Outros dispositivos incluem: leitores multimídia portáteis e reprodutores de conteúdo para ler/ouvir livros eletrônicos (exemplo: e-BookMan).

Nos dispositivos de mão, tudo é menor que no PC, com tecnologias normalmente muito diferentes das do PC. Alguns exemplos de serviços oferecidos incluem: livro de endereços e telefones, calendário, atividades a fazer, editor de texto, calculadora, despertador, e-mail, sincronização de dados e jogos, navegadores, relógios globais, gerente de arquivos, planilhas, aplicações financeiras, reprodutores de mídia, recursos de desenho etc.

Telefones Celulares

Os celulares oferecem vários serviços, entre eles: *Paging*, transmissão de dados, fax, mensagens curtas - SMS (até 160 caracteres), WAP (para acesso a serviços de escalas de vôo, previsão do tempo, informações de trânsito, cotação de ações, reserva de hotel etc). O oferecimento de serviços multimídia é o alvo principal dos fabricantes de celulares, o que está se tornando possível na medida em que aumenta-se a implantação e abrangência das redes sem fio de maior capacidade de largura de banda, aumenta-se o potencial de processamento dos processadores de celulares e também a capacidade das baterias.

A seção abaixo descreve aspectos das tecnologias de hardware dos dispositivos, que diferem muito das tecnologias do PC [5] [20]. Apesar de algumas dessas tecnologias poderem ser usadas nas outras classes de dispositivos, elas se referem, em especial, aos dispositivos de acesso a informação.

2.4 Aspectos de Hardware dos Dispositivos Ubíquos [20]

Bateria: Tecnologia Limitante

Uma tecnologia que limita a velocidade de desenvolvimento de novos serviços para os dispositivos é a bateria. A velocidade de desenvolvimento das baterias é menor do que a de outras tecnologias. Desta forma, qualquer avanço é rapidamente superado pelo aumento de consumo de energia dos processadores mais rápidos. Pesquisadores do mundo todo trabalham com o objetivo de descobrir novas fontes de energia que sejam mais leves, suportem mais tempo de fala e de processamento de aplicações, além de serem menos agressivas ambientalmente. Baterias de Niquel-Cadmo (NiCad), pesadas e sujeitas a perda de capacidade, foram substituídas por tecnologias como as de Niquel-Metal Híbrido (NiMH), mais leves, maior capacidade e menor agressividade ambiental. Baterias de Lítio-íon (Li ion), encontradas hoje em vários tipos de equipamentos eletrônicos, possuem baixa capacidade de energia mas suportam um tempo mais longo de fala graças a redução da necessidade de energia dos dispositivos modernos. Células de lítio polímero, que usam gel para o eletrólito, refletem uma das tecnologias mais modernas na área de baterias. Feita de camadas finas e flexíveis, podem assumir qualquer forma ou tamanho.

Telas

Nos dispositivos de acesso a informação, o CRT - Tubo de Raios Catódicos - é substituído pela LCD – Tela de Cristal Liquido. Apesar de ter um custo mais alto, a LCD pesa menos e consome menos energia. As tecnologias de LCD são: OLED - Diodo Orgânico Emissor de Luz, inventado há 15 anos mas só agora disponível comercialmente; Semicondutor Cristalino, compostos orgânicos que permitem a construção de dspositivos flexíveis (construído em qualquer tamanho e cor); LEP - Polímero Emissor de Luz; CoG - Chip-no-vidro; LCoG - Cristal sobre Vidro Líquido. Essas tecnologias permitem a criação de telas minúsculas (tamanho de pixel de 10 micrômetros). Como são extremamente pequenas, requerem alguma forma de magnificação (canhão multimídia, capacetes etc).

Memória

O desenvolvimento da tecnologia de memórias é orientado, em parte, pelos telefones inteligentes, câmeras digitais, tocadores MP3 e PDAs. Atualmente, é viável a integração de vários megabytes em dispositivos móveis, através de tecnologias como a dos micro discos rígidos removíveis (*Microdrive* da IBM que suporta de 340MB a G1B), e memórias *Flash* não voláteis (em telefones inteligentes e PDAs). A memória RAM nos telefones inteligentes e PDAs requer menos energia e provê acesso mais rápido. Com capacidade típica de 2 a 16MB, contém o Sistema Operacional e os dados da aplicação. Esses dispositivos normalmente possuem s*lots* de expansão para memória adicional. Os tipos de memória RAM disponíveis incluem: RAM estática – SRAM que é ideal para cache. Requer pouca energia, não faz *refresh*, possui esquema mais simples de endereçamento e guarda os dados que mudam com freqüência – os outros são armazenados em flash ou DRAM; RAM Unitransistor - Ut-RAM, estática e dinâmica em um único *chip*, possui maior capacidade de memória em *chips* menores, que requer menos energia; RAM magneto-resistiva – MRAM e RAM ferroelétrica – FRAM, ambas são uma tendência na combinação de memória magnética e

semicondutora com comportamento similar ao da RAM estática, que requer menos energia, pois não tem os ciclos de regeneração dos bits *(refresh)* que a RAM dinâmica tem. Com estas duas ultimas tecnologias de memória, é necessário apenas um único tipo de memória nos dispositivos móveis, pois elas substituem a combinação de volátil e não volátil atualmente em uso.

Processadores

As melhorias na fabricação do CMOS têm levado a estruturas cada vez menores com um número cada vez maior de transistores. A voltagem abaixou de 3,3V em 1995 para 1,35V em 2002. Há menor emissão de calor e caches maiores. Com o problema critico da limitação das tecnologias de baterias, a tecnologia dos processadores tem avançado a ponto de oferecer mecanismos que permitem a redução da voltagem do dispositivo por software, o que reduz o ciclo do *clock* (roda mais devagar), economizando energia. O processador *SpeedStep* da Intel faz o gerenciamento da energia ajustando a freqüência interna do *clock* e a voltagem do núcleo. Ele se adapta a mudanças no fornecimento de energia, desligando as partes da CPU que não são necessárias no atual processamento. A transição entre os modos econômico e normal é transparente.

Entrada de dados – Teclados

O tamanho dos dispositivos de acesso a informação é imposto pela necessidade de E/S de dados, uma vez que o tamanho dos componentes de E/S (teclado, LCD) influencia o tamanho total do dispositivo móvel. Quando os dispositivos de E/S são integrados em um mesmo dispositivo, é difícil esperar que se tornem menores do que já são nos dias de hoje. Entretanto, quando considerados separados, os avanços na tecnologia vão certamente torná-los menores. Dependendo do tamanho do dispositivo, o teclado pode oferecer um conjunto completo de teclas ou um conjunto limitado para a entrada dos dados. Quanto menor o numero de teclas, mais complexa será a operação do dispositivo. Por outro lado, um teclado completo acoplado a um dispositivo certamente vai deixá-lo maior.

Teclados Virtuais em Telas Sensíveis a Toque

Quando os teclados são omitidos, por exemplo, por falta de espaço no dispositivo, outras tecnologias são usadas, tais como: *Teclados na Tela* - em dispositivos com telas grandes, substitui-se o teclado mecânico pelo teclado virtual (tela sensível a toque) e *Teclado Fitaly* – teclado que apresenta *layout* especial, diferente do QWERTY. As letras são organizadas baseadas na freqüência individual e probabilidade de transições na língua Inglesa. As letras i, t, a, l, n, e, d, o, r, s, espaço perfazem 73% das teclas usadas na língua inglesa. Adicionando-se as letras c, h, u, m, chega-se a 84%. As demais teclas estão todas no máximo a duas teclas de distancia da área central. A figura 2 mostra um teclado Fitaly na tela de um Pocket PC (W/CE).

Teclados Tegic T9

O padrão T9 *(Text on 9 digits)*.[24] usa números para entrada de texto, onde cada tecla numérica é associada a um conjunto de letras. O número é pressionado várias vezes, por exemplo, as letras “A”, “B” e “C” são associadas ao digito “2” no teclado. A tecla “2” tem que ser pressionada uma vez para “A”, duas vezes para “B” e três vezes para “C”. Quando

surge ambigüidades, isto é, quando uma palavra é representada pela mesma seqüência de teclas, o usuário tem que ajudar. O T9 é usado para entrada de texto em telefones moveis e PDAs. Um outro tipo de teclado é o Octave que mapeia cada letra do alfabeto em um de oito traços *(strokes)* distintos. Os traços são baseados em símbolos que representam uma parte comum característica das letras que eles representam, localizados ao redor das pontas do botão/tela sensível a toque em forma de estrela. Desenhos e textos explicativos sobre o Octave podem ser encontrados em [25].

Figura 2. Exemplo de um teclado Filtaly na tela de um Pocket PC (W/CE)

Alguns dispositivos, por questões, por exemplo, de falta de espaço, podem omitir totalmente o uso do teclado. Nestes casos, tecnologias alternativas, como reconhecimento de escrita a mão ou reconhecimento de voz são utilizadas.

Reconhecimento de escrita a mão

Tecnologia viável quando estiverem disponíveis: poder de processamento suficiente e telas sensíveis a toque. O maior desafio desta tecnologia é o reconhecimento de palavras completas, que é muito mais difícil do que o reconhecimento de letras individuais, pois requer uma captura muito precisa dos dados e retorna o reconhecimento com atraso. Outros métodos limitam-se ao reconhecimento de caracteres separados. Aqui, requer-se que o *stylus* seja levantado depois de cada caractere. O reconhecimento de caractere normalmente atinge alta taxa de reconhecimento, mas requer cooperação do usuário. O numero de formas que uma letra pode ser desenhada para ser reconhecida é muitas vezes limitado. A figura 3 mostra um exemplo de reconhecimento de escrita a mão. Para o reconhecimento de caracteres de linguagens asiáticas, o sistema operacional – SO do Palm pode ser estendido para a versão **CJKOS** – extensão de SO que adapta o SO do Palm para E/S em **c**hinês, **j**aponês e **k**oreano), como mostra a figura 4.

Reconhecimento de Voz

É o método de entrada mais natural. Requer espaço mínimo para ser integrado no dispositivo. É uma tecnologia cara (requer alto poder computacional). A fala contínua já está disponível

em PCs e é uma questão de tempo até que esteja disponível nos dispositivos móveis. O telefone é o dispositivo mais óbvio para a integração com reconhecimento de voz (entrada de números de telefone em serviços de agenda, via voz, já disponível). Os desafios dos futuros sistemas de reconhecimento de voz incluem: reconhecimento de fala contínua, compreensão de solicitações complexas, tradução para outras línguas etc.

2.5 Sistemas de Entretenimento

Os sistemas de entretenimento envolvem dispositivos cujo uso é voltado principalmente ao lazer. Exemplos incluem: TV (via cabo, satélite, microondas), *Set-top-boxes* (caixa sobre a TV que promove a interface entre os provedores do serviço de difusão e a TV do consumidor), Console de jogos (exemplos: *Dreamcast* da Sega, *Playstation2* da Sony, *Dolphin* da Nintendo e X-box da Microsoft), Câmeras digitais, Brinquedos inteligentes, Reprodutores de musica MP3 etc.

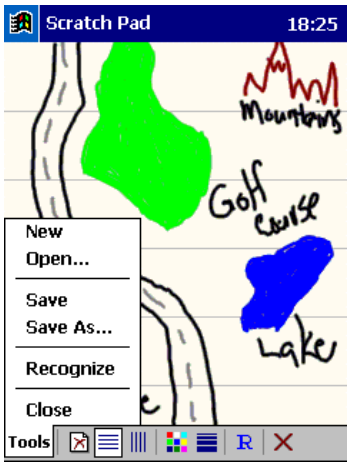

Figura 3. Exemplo de reconhecimento de escrita

Figura 4. Exemplo de tela com CJKOS no Pocket PC

3. Software de Sistemas de Computação Ubíqua

Devido as limitações de memória, capacidade de processamento e bateria dos dispositivos, além da diversidade de tipos, alta latência das redes de comunicação sem fio, etc., as aplicações para os dispositivos têm que ser projetadas tendo em mente essas limitações. Os Sistemas operacionais, por sua vez, têm que ser projetados tendo em mente as características do dispositivo e o objetivo de uso. A computação ubíqua traz novos requisitos para as infraestruturas de serviço, ou *middlwares*, tais como: descoberta de serviços, adaptação da aplicação, além de *frameworks* que atendem necessidades especificas. Estas e outras questões relacionadas ao software de sistemas de computação ubíqua são discutidas a seguir.

3.1 Aplicação

Os desenvolvedores de aplicações para ambientes ubíquos têm uma seleção limitada de linguagens de programação disponíveis, dentre elas: Java, C e C++ (além de algumas linguagens proprietárias). A linguagem Java oferece independência de plataforma do código compilado e atende os requisitos de dispositivos pequenos. As linguagens C e C++ podem produzir código nativo muito pequeno, além de altamente otimizado para uma plataforma específica. Estas linguagens são indicadas para programação em ambientes com fortes restrições de memória, como é o caso da maioria dos dispositivos ubíquos.

Os dispositivos ubíquos processam executáveis e dados direto de sua localização em RAM ou ROM. A quantidade de memória nos dispositivos ubíquos é uma fração da de um PC (normalmente não tem disco rígido). A programação da aplicação deve ser eficiente pois o poder de processamento e a memória são recursos limitados. Deve-se evitar perder dados quando a bateria está com carga baixa. Muitas vezes o sistema operacional e as aplicações não são atualizáveis no tempo de vida do dispositivo. As aplicações normalmente não são reiniciadas e podem executar para sempre.

3.1.1 O desenvolvimento de aplicações para ambientes de computação ubíqua

Ao desenvolver aplicações para dispositivos ubíquos, os programadores devem ter em mente as seguintes restrições:

- tamanho limitado da tela, capacidade limitada de entrada de dados, poder limitado de processamento, memória, armazenamento persistente e vida da bateria
- alta latência, largura de banda limitada e conectividade intermitente (o que os dispositivos esperam encontrar em termos de conectividade)

Frente às limitações listadas acima, as aplicações devem:

- Conectar-se a rede apenas quando necessário
- Consumir da rede apenas os dados que realmente precisa
- Permanecer útil mesmo quando desconectado

Algumas estratégias para o desenvolvimento de aplicações para dispositivos pequenos incluem:

- **Manter simples.** Remover características desnecessárias. Se possível tornar essas características como uma aplicação secundária, separada.
- **Menor é melhor.** Aplicações menores usam menos memória no dispositivo e requerem tempo mais curto de instalação.
- **Deixe o servidor fazer a maior parte do trabalho.** Mover as tarefas computacionalmente intensivas para o servidor. Deixar o dispositivo móvel tratar a interface e quantidades mínimas de computação. Claro que isto depende de quão fácil e com que freqüência o dispositivo pode se conectar no servidor.
- **Escolha a linguagem cuidadosamente.** Linguagens da família Java, como o J2ME, não são a única opção. Linguagens como C++ também devem ser consideradas.

- **Se linguagem usada é Java, minimizar uso de memória do sistema de execução (*run-time*).** Usar tipos escalares em vez de tipos objetos. Não depender do *garbage collector* (você mesmo deve gerenciar a memória eficientemente colocando as Referências a objetos em NULL quando terminar de usar). Somente alocar objetos quando estes forem necessários. Liberar os recursos rapidamente, reusar objetos e evitar exceções.

Estratégias de programação para codificação visando o melhor desempenho incluem:

- **Usar variáveis locais.** É mais rápido acessar variáveis locais do que acessar membros de classes.
- **Evitar concatenação de *string*.** A concatenação de *strings* diminui o desempenho e pode aumentar o uso de memória da aplicação.
- **Usar *threads* e evitar sincronização.** Qualquer operação que levar mais de 1/10 de um segundo para executar requer um *thread* separado. Evitar sincronização também pode aumentar o desempenho.

3.1.2 Java para Ambientes Ubíquos

Applets, Servlets, JavaBeans e aplicações Java são todos tipos de programas em Java e bem conhecidos dos profissionais da área de Computação. Por questão de falta de espaço, suas definições e características podem ser encontradas em [26], [27] e [28]. A independência de plataforma que a Java oferece, a grande quantidade de bibliotecas disponíveis (de suporte a construção de interfaces gráficas a suporte de rede), e a existência de maquinas virtuais embutidas em vários dispositivos tornou a linguagem Java uma tecnologia chave para o desenvolvimento de software na computação ubíqua.

A figura 5 mostra como funciona a execução de programas Java em diferentes dispositivos. O código em Java é compilado num código neutro e padronizado denominado *bytecode*. O sistema operacional de cada dispositivo alvo utiliza um ambiente de execução Java (*Java runtime environment* – JRE) para interpretar e executar o *byte code* gerado na compilação, em tempo de execução. Um interpretador de *bytecode*, também chamado de maquina virtual – MV, traduz as instruções genéricas em comandos nativos, específicos da maquina que está executando a aplicação. Desta forma, qualquer dispositivo que contenha uma maquina virtual Java pode executar um programa em Java sem que este tenha que ser recompilado.

As diferentes especificações da família Java

Como os requisitos das diferentes classes de computadores são diferentes, uma linguagem Java de propósito geral que acomodasse todos esses requisitos diferentes estava tornando-se ingerenciavel. A SUN então decidiu dividir a linguagem Java em diferentes especificações, ilustradas na figura 6. Cada uma das versões de Java possui um Kit de Desenvolvimento de Software – SDK *(Software Development Kit)* que implementa a versão correspondente. Segue abaixo a descrição da família Java.

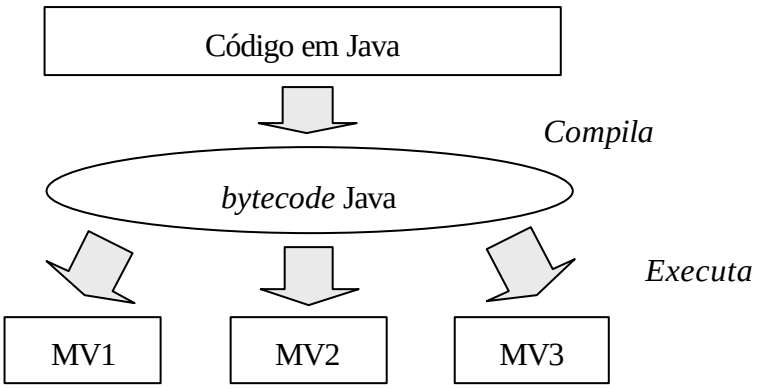

Figura 5. A execução de código Java em diferentes dispositivos

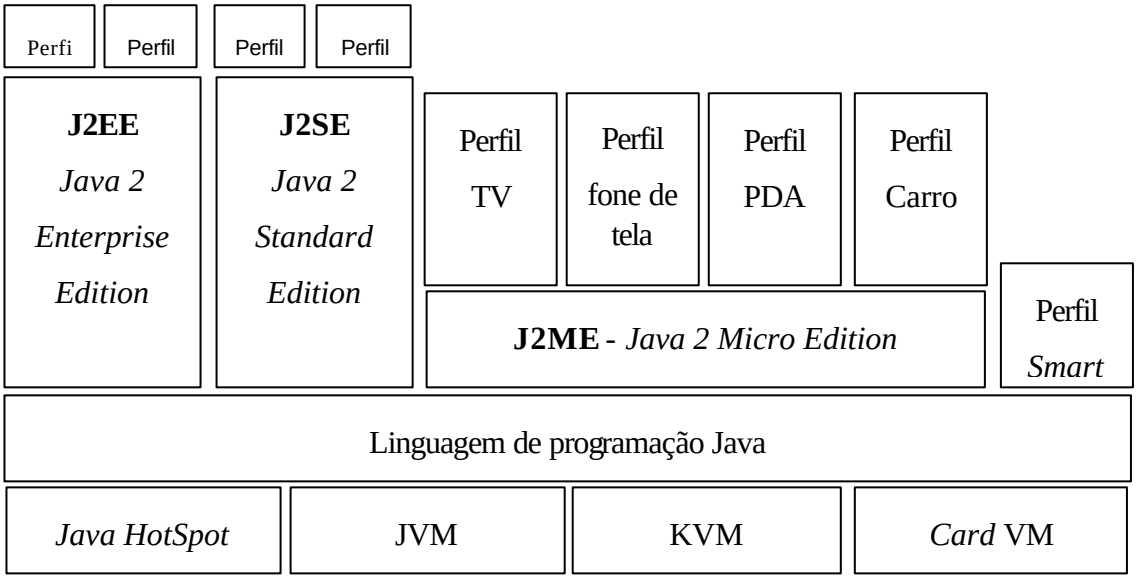

Figura 6. As diferentes especificações da linguagem Java

J2SE - *Standard Edition*

É a especificação da linguagem Java para estações de trabalho e PCs, para aplicações dedicadas e applets. As maquinas virtuais são otimizadas. O objetivo é maximizar desempenho e segurança ao custo de maior uso de memória.

J2EE - *Enterprise Edition*

O J2EE visa os sistemas servidores de grande porte corporativos para aplicações de comercio eletrônico e de negócios na Web *(e-business)*. Oferece características adicionais ao J2SE: servlets JSP, JDBC, *Java Message Service*, Java *Naming* e Interface de Diretório, além de Suporte a transações *(Java Transaction)*, correio eletrônico *(Java Mail)*, XML, CORBA e EJB.

J2ME –*Micro Edition*

Objetiva atender os requisitos de dispositivos ubíquos com as seguintes limitações: pouca memória, energia limitada, conexão intermitente à rede; capabilidades gráficas restritas, dentre outras. O J2ME introduz o conceito de Perfis que são subconjuntos da linguagem Java para diferentes grupos de dispositivos. É composto de funcionalidade básica mínima obrigatória. Classes adicionais, necessárias para suportar características típicas de um grupo especifico de dispositivos, são incluídas em um perfil correspondente. Caso uma aplicação precise de funcionalidade adicional, não especificada no perfil do dispositivo, as bibliotecas correspondentes podem ser trazidas da rede de forma dinâmica (código de aplicação mais bibliotecas de classes). Inclui MVs, bibliotecas de APIs, ferramentas para colocar aplicação no dispositivo, e configura-lo. O J2ME será detalhado mais adiante no texto.

3.1.3 A edição J2ME para Dispositivos ubíquos - o lado do cliente

A edição J2ME é um conjunto de produtos, tecnologias, ferramentas e padrões necessários para criar aplicações para dispositivos ubíquos. Inclui suporte para desenvolvimento de aplicações no dispositivo e fora do dispositivo. O J2ME é disponível em duas versões:

CLDC - Configuração de dispositivo limitado, conectado *(Connected, Limited Device Configuration)* para fones celulares, PDAs e *set-top boxes* de média capacidade. Contém a máquina virtual KVM, um interpretador de linguagem Java para microprocessadores de 16-bit ou 32-bit RISC/CISC em dispositivos com 128 a 512KB RAM, e conexão de rede intermitente. Oferece suporte total a linguagem Java (exceto ponto flutuante, finalização e tratamento de erro). Contem versão minimizada dos pacotes java.lang, java.io, and java.útil do J2SE. Implementa API javax.microedition.io, para conexões de rede (fones móveis e aparelhos de TV).

CDC - Configuração de Dispositivo Conectado *(Connected Device Configuration)* permite desenvolver aplicações para telefones de tela, *set-top boxes*, PDAs de alta capacidade, pontos-de-venda, navegação automotiva e utensílios domésticos. Contém uma máquina virtual completa - CVM para microprocessadores de 32-bit RISC/CISC/DSP em dispositivos com memória ROM maior que 512KB e memória RAM maior que 256KB, e conexão de rede sempre ativa. É uma versão “enxuta” do J2SE mais classes do CLDC. Possui bibliotecas de UI restritas (telefones de tela, *Set-top-boxes)*.

O CLDC e CDC são aninháveis, isto é, qualquer aplicação que executa uma configuração mínima é capaz de executar numa configuração com recursos mais poderosos

MIDP - Os Perfis J2ME

O Perfil é uma forma adicional de especificar o subconjunto de APIs Java, bibliotecas de classe e recursos de máquina virtual para uma família específica de dispositivos. Os perfis objetivam atender necessidades de segmentos específicos da indústria. O MIDP – Perfil do Dispositivo de Informação móvel *(Mobile Information Device Profile)* é a especificação para um perfil J2ME. O MIDP foi projetado para operar acima do CLDC com o objetivo de capacitar as aplicações Java para executarem em dispositivos móveis de informação (MIDs). Contem APIs para o ciclo de vida da aplicação, IU, rede e memória persistente

(java.sun.com/products/midp). Provê um ambiente de tempo de execução padrao que permite a colocação dinâmica de novos serviços e aplicações nos dispositivos do usuário. O MIDP é um perfil padrao comum para dispositivos moveis, independente de fabricante. É uma estrutura completa e de suporte para o desenvolvimento de aplicações moveis.

O MIDP estende o suporte à conectividade oferecida pelo CDLC (via *Connection framework).* Ele suporta um subconjunto do protocolo HTTP protocol, que pode ser implementado usando protocolos IP (TCP/IP) e não IP (WAP e i-mode – através de um *gateway* para acesso aos servidores HTTP na internet). A estrutura de conexão *Connection framework* é usada para suportar redes cliente-servidor e datagramas. O uso de protocolos especificados pelo MIDP permite que a aplicação seja "portável" para todos os MIDs. As implementações MIDP têm que oferecer suporte para acessar servidores e serviços HTTP 1.1. Pode haver amplas variações nas redes sem fio (e conseqüente uso de *gateways)*. A aplicação cliente e o servidor na internet não precisam estar cientes que redes não IP estão sendo usadas, tampouco precisa saber das características destas redes. Entretanto, a aplicação pode tirar vantagem deste conhecimento para otimizar as suas transmissões. Quando um MID não suportar IP, este pode utilizar um *gateway* para acessar a Internet. O *gateway* seria responsável por serviços do tipo DNS. O dispositivo e a rede podem definir e implementar políticas de segurança e acesso à rede que restringem o acesso.

MIDLets – Aplicações MIDP

Um MIDLet é uma aplicação MIDP que compreende um conjunto de classes projetado para ser executado e controlado pelo software de gerenciamento da aplicação, também chamado de software de gerenciamento MIDLet. O software de gerenciamento MIDLet é uma aplicação que controla como os MIDlets são instalados, atualizados e desinstalados no dispositivo móvel. Ele gerencia dois tipos de MIDLets: *Permanentes* e do *Sistema*:

- ***MIDLets Permanentes*** - As MIDlets Permanentes residem, pelo menos parcialmente, em memória não volátil, tais como ROM ou EEPROM. Elas podem ser trazidas para um dispositivo e escritas na memória persistente do MID. Depois disso, um usuário pode executar uma MIDLet permanente repetidamente sem que esta tenha que ser trazida novamente.
- ***MIDlets do Sistema*** - São MIDlets permanentes especiais criadas pelo fabricante do MID. Executam funcionalidades especificas do dispositivo móvel e têm acesso a funcionalidades não publicas do dispositivo e possuem, normalmente, restrições especiais para sua recuperação e instalação.

Fases do ciclo de vida de uma MIDLet

O Software de gerenciamento MIDLet é responsável por mover entre as seguintes fases: Recuperação, Instalação, Lançamento, Gerenciamento de versão e Remoção.

Ações resumidas que devem ser feitas para desenvolver aplicações J2ME [29]:

1. Buscar e instalar CLDC e MIDP para Windows ou UNIX (Win32, Solaris, e Linux)
2. Instalar a KVM no dispositivo (através do *HotSync* do PDA)
3. Buscar e instalar o emulador do Palm OS (POSE)
4. Transferir a imagem da ROM de um Palm para o PC para ser usada com o emulador (disponível de alguns fabricantes)
5. Buscar e instalar o *Toolkit Wireless* do J2ME para começar a desenvolver aplicações J2ME para dispositivos MIDP

Produtos do Java2ME necessários para o desenvolvimento de aplicações do lado do dispositivo

- APIs MIDP
- APIs do CLDC - *Connected Limited Device Configuration*
- Bibliotecas para: Programação da Interface do Usuário em um dispositivo que usa a API LCDUI - *Limited Connected Device User Interface;* RMS – Armazenamento persistente de dados em um dispositivo
- Conectividade com um servidor ou com outro dispositivo através das APIs do GCF - *Generic Connection Framework*

Ferramentas de desenvolvimento para Java

- Visual Café (Symantec)
- Jbuilder (Borland)
- Visual Studio J++ (MS)
- Visual Age (IBM)
- Outras ferramentas podem ser consultadas em [30].

Três outras tecnologias são definidas para potencializar o uso do Java: Java Card, Java Embarcado e Java para Tempo-Real, definidos a seguir.

Java Card

Menor e mais limitada versão da família Java. Não suporta *strings*, interface gráfica, formato de *bytecode* comprimido. O Fórum Java Card [31] especifica o subconjunto da linguagem para smart cards. As MVs são construídas por desenvolvedores de sistemas operacionais e ficam residentes na ROM do cartão

Java Embarcado

Objetiva os controladores industriais, *switches*, e outros dispositivos com restrições severas de memória, em dispositivos com visor orientado a caractere ou sem visor. Não existe um

core mínimo obrigatório – classes desnecessárias podem ser omitidas, o que, naturalmente, limita a interoperabilidade.

RTJS - Java para Tempo-Real

O Java RTJS oferece suporte na criação de aplicações para dispositivos automotivos, industriais etc., que exigem um comportamento de execução previsível. Resolve problemas de coleta de lixo, realiza escalonamento personalizado de processos e threads por compartilhamento de tempo, oferece gerenciamento avançado de memória, além de prover acesso direto à memória de sensores e atuadores, suporta a sincronização de threads e objetos, trata eventos assíncronos e etc.

3.1.4 Maquinas Virtuais Java

KVM (Sun)

A maquina virtual K da Sun, é derivada da especificação da JVM. É uma maquina otimizada para executar em sistemas com memória > 128KB. Objetiva o menor uso possível de memória possível, em vez de desempenho máximo. Executa em processadores de 16/32 bits. É modular e extensível, facilmente transportável para plataformas diferentes. Não suporta AWT ou swing - usa *drivers* de E/S especiais nativos. Não implementa algumas características da MVJ padrão (RMI, agrupamento de *threads* etc), outras são opcionais (ponto flutuante, arranjos multidimensionais, verificação de arquivo de classe) e outras ainda, são parcialmente implementadas (java.net e java.io). Para se ter uma idéia do tamanho de uma maquina virtual, a KVM para PalmOS ocupa 50 a 70KB e 128KB quando em execução. Executa em mais de 20 plataformas, entre elas: Solaris; W/32; telefones Sony, Nokia, Motorola e SO EPOC.

Waba (Wabasoft)

É um produto de fonte aberto, com tecnologia baseada no Java – similar mas não compatível. A linguagem Waba, a Máquina Virtual Waba - MVW e o formato de arquivo de classe foram projetados para otimizar desempenho (omite características desnecessárias e/ou que usam muita memória). O tamanho da MVW é de 64KB (32KB para o executável e 32KB para as classes fundamentais). Os formatos de Arquivo de classe e *bytecode* são subconjuntos dos correspondentes em Java. Permite aos programadores usar ferramentas de desenvolvimento Java, desde que seja usado apenas o subconjunto suportado pelo Waba. Possui um conjunto de classes fundamentais (a menor possível que contem funcionalidade para escrever programas complexos para dispositivos pequenos). Possui classes “ponte”, que permitem executar programas em Waba onde o Java for suportado. Usa bibliotecas nativas do dispositivo, o que melhora o desempenho.

J9 Visual Age ME (IBM)

É uma maquina virtual menor e mais rápida que a KVM. Suporta adição de cores. Possui um ligador inteligente *(linker)*, que Remove classes, métodos e campos desnecessários de suas aplicações nas bibliotecas de classes. Suporta depuração remota (executa e depura a aplicação de um PDA num PC). O J9 é usado pelo VAME *(Visual Age ME)* da IBM – ambiente integrado para desenvolvimento de aplicações Java embarcadas.

A tabela 3 mostra uma comparação entre as maquinas virtuais para dispositivos ubíquos.

3.1.5 A edição J2EE para Servidores em Ambientes de Computação Ubíqua - o lado do servidor

A plataforma Java2, *Enterprise Edition* (J2EE) define e simplifica aplicações corporativas tratando de varios detalhes do comportamento da aplicação de forma automática, sem a necessidade de programação complexa. J2EE inclui características da plataforma J2SE, tais como portabilidade, APIs JDBC para acesso a base de dados, CORBA para interação com recursos corporativos legados, além de um modelo de segurança que protege os dados mesmo em aplicações da Internet. A plataforma J2EE trata de complexidades que são inerentes a aplicações corporativas, tais como gerenciamento de transações, gerenciamento do ciclo de vida, compartilhamento de recursos etc., que podem ser usados pelas aplicações e por componentes, facilitando assim o trabalho dos desenvolvedores que podem se concentrar na lógica da aplicação e na interface do usuário.

	KVM	**Waba**	**J9**
Plataformas	Palm, W/CE, EPOC, Linux, W/MS, Solaris	Palm, W/CE	Palm, W/CE, Neutrino, Linux,
Desempenho	Baixo	Alto	Alto
Uso de Memória	70KB	64KB	< 70KB
Bibliotecas	Independente de dispositivo	Dependente de dispositivo	Dependente de dispositivo
Padrões	J2ME	-	Java, J2ME
Fonte	Sob licensa da Sun	Aberto	Proprietário
Ambiente de desenvolvimento	Ferramentas padrão do Java com KVM na plataforma de desenvolvimento	Ferramentas padrão do Java com Waba VM na plataforma de desenvolvimento	Ambiente de desenvolvimento integrado com depuração remota

Tabela 3. Comparação entre as maquinas virtuais para dispositivos ubíquos[5][20].

O J2EE prove opções de interfaces para dispositivos dos mais diferentes tipos, com suporte a linguagens simples de marcação (HTML, WML, etc). Permite o carregamento de *plug-ins* Java para suporte a applets por navegadores Web, além de suportar clientes como aplicações Java independentes.

O J2EE Edition oferece suporte total para componentes EJB - Enterprise JavaBeans, API JavaServlets, JSP - JavaServer Pages, além de XML. O padrão J2EE inclui

especificação completa e testes de conformidade para assegurar a portabilidade das aplicações no âmbito dos sistemas corporativos que suportam o J2EE. A edição J2EE compreende as seguintes tecnologias:

- ***EJBs (Enterprise JavaBeans)*** – Arquitetura padrão de componentes para a construção de aplicações corporativas orientadas a objeto em Java. Constitui-se de API que permite a criação, uso e gerenciamento de componentes da aplicação, reusáveis em diferentes plataformas.
- ***JSPs (Java Server Pages)*** – permite a criação de conteúdo dinâmico na web. Uma pagina JSP é um documento que descreve como processar uma solicitação para criar uma resposta, através da combinação de formatos padrão de marcação com ações dinâmicas (através de scripts).
- ***Java Servlets*** – API que estende a funcionalidade de um servidor Web, através de módulos de aplicação implementados em Java
- *Conectores* – Permite que a plataforma J2EE seja conectada a sistemas de informação heterogêneos
- ***CORBA*** –suporta interoperabilidade de aplicações Java com outras aplicações empresariais, através de uma linguagem de definição de interface – IDL.
- ***JDBC (Java Database Connectivity)*** – API que promove a integração com bancos de dados relacionais, além de fornecer uma base comum sobre a qual ferramentas de alto nível e interfaces possam ser construídas.

3.1.6 XML

De modo a superar as limitações da linguagem de marcação HTML que não separa a representação dos dados da apresentação em paginas da Web, foi criada a Linguagem de Marcação Extensível – XML *(Extensible Markup Language).* XML é uma sintaxe baseada em texto que é legível tanto por computadores quanto por humanos. Oferece portabilidade de dados e reusabilidade em diferentes plataformas e dispositivos. Algumas das aplicações do XML incluem: suporte a publicação independente de mídia, o que permite que documentos sejam escritos uma vez e publicados em múltiplos formatos de mídia e dispositivos. Do lado do cliente, o XML pode ser usado para criar visões personalizadas dos dados. Informações sobre XML são abundantes na Web e na literatura em geral, por isso, e também pelo pouco espaço disponível, não entraremos em mais detalhes sobre o funcionamento do XML.

3.1.7 O Padrão MVC para o desenvolvimento de aplicações

O modelo MVC – Modelo-Apresentação-Controlador *(Model-View-Controller)* é usado como padrão para o desenvolvimento de aplicações para dispositivos moveis, que oferece flexibilidade e uma nítida separação de responsabilidades. Ele visualiza a aplicação sob três aspectos: Modelo *(model),* Apresentação *(View)* e Controlador *(Controller):*

- **Modelo -** Responsável pelos dados e transações da aplicação que podem ser associadas a ela. O modelo encapsula a lógica do negocio
- **Apresentação –** Responsável por apresentar os dados

- **Controlador** - Recebe todas as solicitações que chegam e controla as suas execuções. Para executar uma solicitação, o controlador acessa o modelo e exibe as apresentações conforme necessário.

Exemplos de uso do MVC na construção de aplicações para celulares pode ser vistos em [32].

Benefícios do Uso do Padrão MVC

Separando o controlador do modelo e da apresentação, o fluxo da aplicação é isolado. Desta forma, os desenvolvedores podem entender a perspectiva do usuário apenas olhando o controlador. O fluxo da aplicação pode ser alterado modificando-se o controlador sem alteração (ou pouca) do restante do código

Separando o modelo da apresentação, o modelo é isolado de alterações. Desta forma, os desenvolvedores podem alterar a aparência da interface do usuário sem ter que mudar o modelo. Separando a apresentação do modelo, a apresentação é poupada dos detalhes de como funciona o modelo. O modelo pode pegar seus dados de um armazenamento local, através da API RMS, de um servidor usando HTTP, de uma cache de memória ou de uma combinação dessas fontes

Para usar a rede intermitentemente e ainda permanecer útil quando desconectado, a porção cliente da aplicação deve decidir sobre quando buscar os dados do servidor e quando buscar os dados do armazenamento local. As estratégias de dados locais podem ser baseadas em cache ou sincronização para melhorar a responsividade e manter a coerência dos dados. O particionamento destes detalhes no modelo (do MVC) torna a implementação, teste e manutenção mais fácil do que se estivessem espalhados pela aplicação

Limitações do padrão MVC

O MVC pode aumentar o tamanho do código (grande parte no controlador). Entretanto, a pior limitação do padrão MVC é que, apesar dele separar o modelo da apresentação, de tal forma que os desenvolvedores possam alterar a aparência da interface do usuário sem ter que mudar o modelo, o MVC não muda o modelo da aplicação em si, que é um requisito da computação ubíqua. A mudança do modelo é o resultado da adaptação da aplicação frente a limitações de recursos da rede e do sistema. Pesquisas têm que ser feitas para o desenvolvimento de novas soluções de engenharia de software que permitam a especificação e desenvolvimento desses sistemas.

3.2 Sistemas Operacionais

Com a proliferação de inúmeros dispositivos ubíquos diferentes, reacendeu-se o mercado para novos sistemas operacionais personalizados para estes dispositivos.Assim como nos PCs, os sistemas operacionais de dispositivos ubíquos são responsáveis por gerenciar os recursos do dispositivo, e por oferecer aos programadores e usuários, uma maquina virtual que esconde a complexidade do hardware do dispositivo, tornando mais fácil a tarefa de programar suas aplicações.

Um dispositivo possui, tipicamente, os seguintes recursos: CPU, memória (RAM, Flash, ROM, EEPROM, etc), bateria, telas de diferentes tamanhos e tecnologias, diferentes

dispositivos de entrada (pequenos teclados, apontadores - *stylus*, tela sensível a toque, processamento de voz) e sistema de arquivo. Com exceção das tecnologias envolvidas nos dispositivos, os recursos não mudam muito em relação a aqueles encontrados no PC. O que diferencia então o sistema operacional de uma maquina tradicional do sistema operacional de um dispositivo ubíquo? Uma das diferenças é a especificidade do dispositivo. O PC é uma maquina de propósito geral, enquanto que o dispositivo é muitas vezes otimizado para realizar muito bem uma ou mais tarefas. A especialização do dispositivo é um dos aspectos que determina o projeto do sistema operacional do dispositivo.

No mundo dos PCs, há poucos competidores na área de sistemas operacionais, mas no mundo dos dispositivos ubíquos, o numero é cada vez maior. Alguns exemplos de SO para esses dispositivos incluem: Palm OS (PDA), EPOC (celular), W/CE *(Consumer Electronics)*, Java Card e W/Smart Card para *Smart Cards*, QXN, VxWorks etc. Provavelmente não haverá monopólio equivalente ao da MS/Intel no mundo dos dispositivos ubíquos. Nas seções abaixo, serão mostrados os sistemas operacionais de PDAs e celulares, bem como as características de alguns sistemas operacionais considerados como sistemas embarcados [5] [20] [33].

3.2.1 PALM OS

Desenvolvido pela Palm Inc., domina cerca de 70% do mercado de dispositivos de mão, apesar de proprietário. O sucesso do PALM deve-se ao fato dele ser projetado especificamente para PDAs, ser fácil de usar, oferecer um número limitado de características, mas que são altamente otimizadas, levando ao uso de pouca memória e de UCP, o que garante vida mais longa de bateria.

As novas versões do PalmOS suportam comunicação bluetooth, 64K cores, em PDAs multimídia integrados a fones móveis. Além da Palm Inc., o Palm OS é o sistema operacional de dispositivos da Sony, IBM, HandSpring entre outras. Para manter o SO pequeno e rápido, o Palm OS não separa as aplicações umas das outras, o que traz problemas de estabilidade (se uma aplicação cai, o sistema todo cai) e de segurança, já que cada aplicação pode ler e alterar dados de outras aplicações. As aplicações têm que ser extremamente bem construídas e testadas.

Toda a memória de um Palm reside em cartões de memória (um cartão é uma unidade lógica de RAM, ROM ou ambas). Cada cartão de memória tem um espaço de endereço máximo teórico de 256MB. A memória é dividida em *Heap dinâmico* e *Heap de armazenamento*, que são gerenciados separadamente. O espaço que é deixado para a aplicação e muito pequeno (<36KB), por isso a memória da aplicação deve ser mantida a menor possível, com ações do tipo: evitar geração de estruturas grandes de pilhas, evitar uso de variáveis globais e copias de base de dados na memória dinâmica etc. O *Heap dinâmico* é alocado para pilha e *heap* de aplicações em execução. O *Heap de Armazenamento* retém dados permanentes, tais como base de dados e arquivos fonte da aplicação. É uma memória protegida contra escrita , acessada apenas através do gerente de memória. Provê acesso rápido de leitura e lento de escrita. A integridade é garantida através de suporte a transações.

O SO e aplicações embutidas ficam em memória ROM. O gerente de memória aloca (pedaços de 64KB), desaloca, muda o tamanho e impede o acesso a outros pedaços da memória que podem estar na memória dinâmica ou de armazenamento. O Palm OS não usa sistema de arquivo. O armazenamento é estruturado em base de dados. Cada base de dados, gerenciada por um gerenciador de base de dados, possui múltiplos registros. Apenas uma aplicação roda por vez. Não suporta threads - –uma aplicação de busca pode invocar outras aplicações para que elas façam, por ex., uma consulta em suas bases de dados. O controle volta para a aplicação que chamou. O Palm OS é orientado a eventos que são tratados por tratadores de eventos. Eventos típicos incluem: ações do usuário em interação com a aplicação; notificações do sistema (alarme de tempo) e outros específicos da aplicação. Os programadores devem evitar código que faça processamento enquanto espera por um evento (espera ociosa) para economizar bateria.

As linguagens de programação disponíveis são C, C++, Java e proprietárias. Dois pacotes para desenvolvimento de aplicações são suportados: SDK (APIs para desenvolver aplicações, funções de interface do usuário, gerenciamento do sistema e comunicação) e o CDK *(Conduit Development Kit)* que suporta a implementação de "conduites" para troca e sincronização de dados entre uma aplicação de mesa e uma aplicação que roda no dispositivo.

Suporte para Programação do Palm

O suporte para programação inclui: CodeWarrior da Metrowerk e Compilador GNU C para Palm OS. Assim como ocorre para a maioria dos dispositivos ubíquos, cada aplicação tem que ter um ID único (creator ID) obtido da empresa, no caso, a Palm.

3.2.2 O Sistema Operacional Symbiam EPOC

Criado pela Psion (agora mantido pela Symbian – fundada pela Psio, Motorola, Panasonic, Ericsson e Nokia em 1998), o EPOC está disponível para o NEC V30H (16 bits) e o ARM/StrongARM (32 bits) e possui as seguintes características:

- Escrito em C++
- Suporta sistemas de tempo-real
- Mono-usuário
- Multitarefa – tem baixa latência no tratamento de interrupção e no chaveamento de contexto de tarefas, o que o torna ideal para responder ao usuário e receber dados do ar ao mesmo tempo
- É preemptivo, com escalonador orientado a prioridade, isto é, quando um processo de maior prioridade tem que ser executado, o processo atual é suspenso
- Suporta e-mail e troca de mensagens, sincronização de dados (entre dispositivo e PC)
- Suporta diferentes tipos de plataforma: **Crystal** (comunicadores com teclado pequeno); **Quartz** –(pequenos comunicadores); **Pearl** (fones móveis inteligentes); VGA completo.

- Suporte para bluetooth e WAP.

A arquitetura do EPOC constitui-se de 4 níveis:

1. **Base** - Contém sistema de execução e núcleo com dois componentes: F32, que fornece carregador de *bootstrap*, acesso e monitoramento de sistemas de arquivo; e E32, que fornece escalonador, interrupção de relógio, gerenciamento de *drivers* do dispositivo e gerenciamento de memória, com o conceito de MMU para separar os espaços de endereço para cada aplicação, além de ferramentas para checagem de erros de acesso invalido á memória, desalocação de memória, verificação e limpeza de pilha e *heap*.

2. **Middleware –** Oferece suporte a aplicação com SGBD, armazenamento de fluxos de caracteres, impressão, servidor de janelas, etc.

3. **Interface Gráfica do Usuário (EIKON) - U**ma estrutura para GUIs que é parte do SDK do EPOC e que oferece elementos de interface gráfica padrão, tais como botões de diálogos e menus; além de tratar a entrada de dados e de comandos. A Eikon separa entre a função da interface e a apresentação da interface, o que é importante quando os tamanhos de tela e as funcionalidades de dispositivos mudam muito.

4. **Aplicações** - Cada aplicação no EPOC tem ID único para identificação de arquivo e associação – a faixa 0x01000000 a 0x0fffffff reservada para propósito de desenvolvimento e teste. Antes da aplicação ser liberada, um ID único tem que ser solicitado a Symbian.

3.2.3 O Sistema Operacional W/CE *(Consumer Electronics)*

O W/CE não é um SO de "prateleira". Cada fabricante de dispositivo precisa configurar o W/CE para a plataforma de processador específica de seu dispositivo. Normalmente o SO é entregue em ROM, em contraste com os SOs baseados em disco, orientados a *desktop*, como o Linux ou BeOS. O W/CE é oferecido em blocos modulares, o que permite a criação de um sistema apenas com as partes essenciais, economizando um espaço precioso – o tamanho do SO é mantido consistente com o tamanho do dispositivo: um computador de mão, por exemplo, inclui mais funcionalidades do que um telefone inteligente.

O W/CE diferencia-se totalmente do Palm OS que foi projetado de forma altamente otimizada para uma classe alvo de dispositivos. As tarefas executadas pelo núcleo do SO W/CE incluem:

- Gerenciamento de Processos e *Threads*
- Gerenciamento de memória
- Escalonamento de tarefas
- Tratamento de interrupção

Características do W/CE

- Possui um Gerente de Gráficos/Janelas/Eventos - GWE
- Suporta Armazenamento de Objetos (arquivos, registros e uma base de dados)

- Suporta Interfaces de comunicação Infravermelho via IrDA, TCP/IP e *drivers* seriais
- Suporta 32 processos simultaneamente
- Suporta número ilimitado de *threads*, usadas para monitorar eventos assíncronos, tais como:interrupção de hardware e atividades do usuário (depende de memória física disponível)
- Suporta comunicação entre processos – IPC (seções criticas, mutex e eventos)
- Suporta oito níveis de prioridades de *threads* (do ocioso ao tempo critico). As prioridades não podem mudar, exceto quando um *thread* de baixa prioridade retém recurso que um *thread* de alta prioridade precisa
- Suporta Interrupções que são usadas para notificar o SO sobre eventos externos
- Atende requisitos de um S.O. de tempo real não critico para aplicações do tipo: comutação de telefonia, controle de processos de fabricação, sistemas automotivos de navegação etc.
- Trata os vários tipos de armazenamento da mesma forma: cartões de memória flash, *drivers* de disco, ROM, RAM
- Possui memória virtual protegida de até G4B com Unidade de Gerenciamento de Memória – MMU
- Suporta até 32MB por processo

A memória disponível é dividida em dois blocos no W/CE[20]:

1. ***Memória de programa*** - Alocada e gerenciada como memória virtual que é usada para pilha e *heap* em RAM. O SO e as aplicações embutidas, como o *Pocket Word* – versão compacta do WinWord -, são armazenadas e executadas diretamente da ROM, o que economiza no tempo de carregamento
2. ***Memória de armazenamento*** - Inclui os dados persistentes (normalmente armazenado em disco rígido). Parte da memória volátil é referida como armazenagem de objetos (que retém registros do sistema, diretórios, todas as aplicações e dados do usuário) com tamanho máximo de 16MB. Fora da área de armazenagem de objetos, as bases de dados e os sistemas de arquivos podem ser instalados em RAM, ROM ou dispositivos externos. Cada sistema de arquivos pode ser dividido em múltiplos volumes, representados como pastas do diretório raiz dos sistemas de arquivos. Os dados podem ser salvos em objetos arquivos ou bases de dados. Os arquivos são automaticamente comprimidos. O W/CE suporta *Heap* especial para o Sistema de Arquivos, registros e armazenagem de objetos (até 256MB) com serviço de transação para assegurar integridade dos dados.

Os esquemas de gerenciamento de memória suportados incluem a memória paginada com uso de cache e sem uso de cache e memória não paginada com uso de cache e sem uso de cache. A tabela de páginas é mantida pelo hardware.

O Gerente de Gráficos/Janelas/Eventos - GWE

A interface do W/CE com o usuário é similar ao do MS/W, com suporte a menus, caixas de diálogo, ícones, som, vídeo, e reconhecimento de escrita a mão. Toda aplicação tem que ter pelo menos uma janela, mesmo que esta não possa ser mostrada. Todas as mensagens que uma aplicação envia ou recebe são passadas pela fila de mensagens do *thread* da janela correspondente. Uma janela é o ponto de entrada de cada aplicação.

O Subsistema de Gráficos, Janelas e Eventos – GWE suporta APIs de programação da interface. Inclui funções de gerenciamento de energia e troca de mensagens. O Unicode é o formato de texto nativo (2 bytes por caractere). Suporta biblioteca criptográfica (CAPI) que armazena informação segura na memória. A autenticação de aplicações é feita através de assinatura com chave pública. Suporta o uso de cartão inteligente para segurança de dados sensíveis (dados de até 64KB em cartão – acesso mais lento(EEPROM x RAM).

O W/CE suporta as seguintes características de Comunicação e Rede:

- Comunicação serial – SLIP e PPP (via cabo e infravermelho)
- Conexões de Rede (protocolos da Internet e de troca de arquivos)
- API WinInet (HTTP e FTP)
- API WinSock (sockets que suportam TCP/IP)
- API IrSock – suporte a comunicação infravermelha (via IrDA – *Infrared Data Association)*
- RAS – *Remote Access Service* suporta *upload/download* de arquivos (via PPP para conectar no servidor)
- API Telephony – suporta comunicação via modem para um hospedeiro remoto (só chama – não recebe chamadas)

Os *drivers* necessários incluem: bateria, tela, porta serial, InfraVermelho, painel de toque. Os tipos de *drivers* no W/CE são:

- *Nativos* – *drivers* de baixo nível que usam características embutidas de um dispositivo (fonte ou teclado), ligados com o núcleo e residentes em ROM. Os d*rivers* nativos são integrados na configuração do W/CE pelo fabricante do dispositivo
- Interface de fluxo de dados *(serial)* – carregados como DLLs dedicadas
- Baseados na especificação da Interface de *Driver* de Rede (NDIS), em que protocolos podem ser implementados independentes dos *drivers* de hardware
- Outros tipos de *drivers* podem ser instalados (impressora, modem)

3.2.4 Linux Embarcado

O Linux Embarcado é na verdade um Linux “enxuto” com suporte especial para dispositivos ubíquos. Possui as seguintes características :

- Arquitetura microkernel - serviços e características podem ser compiladas no núcleo ou carregadas como módulos dinamicamente ligados em tempo de execução
- Suporte multiusuário
- Sistema multitarefa, preemptivo, com escalonadores de tempo-real opcionais
- Suporte básico para múltiplos processadores
- Tamanho do núcleo pode variar de 200KB a vários MB (uma critica ao linux para dispositivos ubíquos é a de que ele necessita de 2 a 8 MB de memória em tempo de execução)
- Suporta memória virtual com paginação
- Suporta versões "enxutas" do X-Window para interface do usuário
- Possui versões para processadores MIPS, ARM, Motorola e Intel

W/SmartCards

Os menores SOs executam em cartões inteligentes, com severas restrições de memória e poder de processamento. Alguns SmartCards possuem uma maquina virtual Java em ROM que permite que applets Java sejam carregadas para o cartão e executadas. Múltiplas applets podem ser executadas em alguns SOs, o que torna o SO para este ambiente bastante complexo, com o suporte a conceitos do tipo multiprogramação, escalonamento de tarefas, gerenciamento de recursos, proteção de memória etc.

Um exemplo de SO complexo, porem primitivo, para SmartCards, é o Windows para SmartCards. O SO W/SmartCard oferece para os programadores uma API bem definida, que esconde detalhes do cartão e oferece funcionalidade independente de cartão. Em vez de Java, usa bytecode gerado pelo VisualBasic, que é executado em um ambiente de execução no cartão.

O Sistema de arquivo é baseado em uma FAT – *File Access Table*. O número e tamanho de partições são estabelecidos durante configuração - apenas uma partição é suportada no momento.

Em termos de segurança, suporta autenticação, com o conceito de "principais conhecidos", em que "principais" refere-se a usuário, grupo de usuários, ou sistemas. É um sistema configurável, onde um arquivo no cartão é criado para definir como é o processo de autenticação de um "principal conhecido". A autenticação é similar há do PC, mas oferece um protocolo mais sofisticado de autenticação (padrão de Encriptação de Dados - DES). Além da Autenticação, provê também um esquema de Autorização, com o uso de Listas de Controle de Acesso – ACLs que estabelecem os direitos de acesso, através de mecanismos de autorização (regras lógicas que definem *quem* tem permissão para fazer *o que* com um arquivo em particular). As ACLs são armazenadas dentro de arquivos, o que permite proteger as ACLs através de ACLs associadas.

O W/SmartCard suporta applets que são implementadas em código nativo ou Visual Basic (compiladas em bytecode, interpretadas por um sistema de execução e executadas por

MV genérica). Apenas um sub-conjunto do Visual Basic pode ser usado, entretanto funcionalidades extras podem ser adicionadas, através de *plug-ins*.

Outra iniciativa em sistemas operacionais para SmartCards é o **JavaCard.** A interoperabilidade entre os dois é difícil. Por exemplo, os modelos de direitos de acesso nos dois são completamente diferentes.

Dados os sistemas operacionais para dispositivos ubíquos vistos acima, pode-se observar que existem pelo menos duas linhas distintas de desenvolvimento: de um lado, um SO "enxuto", altamente otimizado para aplicações especificas, com interfaces nativas que exploram todas as características do hardware. Exemplo deste tipo de SO é o PalmOS. Do outro lado, tem-se SOs complexos, de tempo-real, multitarefa, que tratam interrupções e chaveiam a CPU para outros processos mais prioritários, como é o caso do EPOC, usado em celulares, onde a aplicação mais importante é o atendimento a chamadas ou a recepção de mensagens curtas, que podem chegar a qualquer momento, inclusive no momento em que o usuário está interagindo com outra aplicação. O projeto de sistemas operacionais para dispositivos ubíquos devem ser pensados em função dos tipos de aplicações que deverão ser oferecidas por um dispositivo, bem como as capacidades de hardware daquele dispositivo. O mercado acena com inúmeras possibilidades e necessidades, o que leva a muitas oportunidades de desenvolvimento de novos SOs.

3.3 *Middleware*

Uma das principais funções de um *middleware* é prover abstração das dependências do SO. O *middleware*, tipicamente uma camada entre o sistema operacional e as aplicações, oferece certas operações e estruturas de dados que permitem aos processos e usuários interoperarem em maquinas diferentes de forma consistente. Estas estruturas e operações são muitas vezes denominadas na literatura como APIs, *frameworks*, bibliotecas, etc., que são consideradas como componentes de *middleware*. Segue abaixo uma breve descrição de alguns componentes construídos como resultado de iniciativas de empresas e outras de laboratórios de Universidades e Institutos de Pesquisa.

3.3.1 Componentes de *Middleware* no mercado

Interface de Programação da Aplicação – APIs *(Application Programming Interface)*

As APIs Java funcionam como extensões para a plataforma Java, para suportar serviços específicos da aplicação. Exemplos incluem:

- *APIs JavaPhone* – suporta serviços de telefonia, tais como: funcionalidades básicas para tratar chamadas, funcionalidades de gerenciamento de informação pessoal – PIM *(Personal Information Management)*, suporte para tratamento de mensagens curtas, independente de rede ou hardware particular, suporte para monitoramento de bateria, e outras mais.
- *JavaTV* – suporta serviços de TV. Exemplos incluem: funcionalidades para sobreposição de vídeo e de gráficos na tela da TV, suporte a serviços que oferecem

diferentes visões sobre um mesmo dado, suporte a Guias Eletrônicos de Programação etc.

- *PC/SC para SmartCards*– Interface de programação padronizada que permite acessar cartões inteligentes de forma independente da aplicação. Exemplos de serviços incluem: APIs que abstraem os protocolos específicos dos terminais de leitura de cartões (leitoras), APIs para acesso a arquivos e autenticação, recursos de detecção de inserção e remoção do cartão, etc.
- *Framework OpenCard* (OCF)– Assim como o PC/SC, o OCF é uma API Java para desenvolver aplicações de *smartcards* no lado do terminal (leitora). O objetivo do OCF é tornar partes de uma solução para *smartcards*, fornecidas tipicamente por diferentes fabricantes, independentes umas das outras. O OCF define dois conceitos básicos: a camada de Terminal do cartão (abstração da leitora de cartões) e a camada de Serviço do cartão (suporte a serviços de acesso a arquivos no cartão, assinatura digital, instalação/desinstalação de applets etc.

Frameworks

Os *frameworks* pode ser definidos como estruturas básicas para uma certa classe de aplicações. Exemplo de classe de aplicação suportada por *frameworks*:

WebTV – iniciativas como a Especificação de Conteúdo Melhorado do Fórum de Melhorias para a TV Avançada (ATEVF) definem formas padronizadas de difundir a TV de maneira melhorada. A WebTV é parte desta especificação e consiste num software residente em uma *set-top-box* (caixa digital que age como interface entre o usuário e o provedor de conteúdo de TV) que objetiva receber dados e apresenta-los na forma de paginas da web na tela da TV. Para isso, consiste de três elementos: “gatilho” para entregar eventos entre paginas ligadas; “formatação de conteúdo” para HTML e “avisos” para iniciar ao conteúdo da TV melhorada.

Outros componentes de *middleware* incluem: Estruturas de suporte a base de dados, como por exemplo, o DB2 *Everywhere*, que oferece, além de serviços de armazenamento e recuperação de dados no dispositivo, suporte a sincronização entre os dados do dispositivo e do servidor correspondente; Infraestruturas escaláveis de suporte a mensagens, como o *ME everyplace*, disponível para o PalmOS, W/CE, EPOC e outros.

3.3.2 *Middlewares* na Academia

Bibliotecas, *frameworks*, *toolkits* e infra-estruturas de serviços têm sido construídas para suportar aplicações em ambientes ubíquos. Nos laboratórios de pesquisa de Universidades e empresas, soluções têm sido pesquisadas que convertem um espaço físico e seus dispositivos de computação ubíqua em um sistema de computação programável. Um dos desafios nesta tarefa é como capturar e tratar os dados que são lidos de sensores e que, quando interpretados, definem um contexto que pode influenciar no fluxo da aplicação.

Todas as estruturas vistas acima tratam do suporte a uma classe especifica de serviços (telefonia, TV, *smartcards)*. Os *middlewares* sendo pesquisados no meio acadêmico são mais ambiciosos e oferecem soluções mais completas para o suporte da mobilidade,

adaptabilidade de aplicações, tratamento de contexto, descoberta de serviços, etc. Segue abaixo um resumo de alguns desses projetos:

Context Toolkit [34], um conjunto de ferramentas desenvolvido pelo *Georgia Institute of Technology* preocupa-se com tudo que é referente a dados de contexto capturados através de sensores em ambientes de computação ubíqua. Seu objetivo é auxiliar o desenvolvimento de soluções para tratar as dificuldades originadas da natureza da informação de contexto e, então, facilitar a construção de aplicações cientes de contexto. O Context Toolkit utiliza o conceito de *widgets* que é utilizado nos *toolkits* de GUI (*Graphical User Interface*). Um *widget* de contexto é um componente de software que fornece à aplicação acesso a informação de contexto de seu ambiente operacional. O desafio a ser superado pelo Context Toolkit é o tratamento de contexto.

Gaia [35] é uma **infra-estrutura** desenvolvida pela Universidade de Illinois que trata um espaço ativo (*ActiveSpace*) e seus dispositivos de forma análoga a um SO tradicional, fornecendo serviços básicos, incluindo eventos, presença de entidades (dispositivos, usuários e serviços), notificação de contexto, descoberta e sistema de nomes etc. O Espaço Ativo corresponde a qualquer ambiente de computação ubíqua que pode ser gerenciado pela infra-estrutura Gaia [36]. Gaia utiliza uma nova abstração para a computação que é chamada de espaço virtual do usuário (*User Virtual Space*). Um espaço virtual do usuário é composto por dados, tarefas e dispositivos que estão associados a um usuário; ele é permanentemente ativo e independente de dispositivo; move-se com o usuário e mapeia dados e tarefas no ambiente de computação ubíqua do usuário de acordo com seu contexto atual. Gaia converte um espaço físico e seus dispositivos de computação ubíqua em um sistema de computação programável.

iROS (*Interactive Room Operating System*) [37] é uma infra-estrutura construída pela Universidade de Stanford para dar suporte a aplicações dentro de um espaço físico utilizando também a abordagem de SO. A infra-estrutura consiste de um meta-SO que une dispositivos com seus SOs específicos. O Espaço de Trabalho Interativo (*Interactive Workspace*) considerado pelo iROS, consiste de um ambiente com alta tecnologia em que pessoas podem interagir de maneira colaborativa. A arquitetura do iROS é composta por três subsistemas: memória de eventos (*EventHeap*), memória de dados (*Datáeap*) e ICrafter. O núcleo do iROS é a memória de eventos, um repositório central no qual todas as aplicações do espaço de trabalho interativo podem publicar eventos [38]. A memória de dados é responsável pela transformação e movimentação dos dados, permitindo que qualquer aplicação coloque dados num lugar associado ao ambiente local. O ICrafter é responsável pelo controle dos recursos no ambiente. Trata da adaptação de interfaces, bem como contexto, e descoberta de serviços.

Com exceção do *Context Toolkit*, que enfatiza o tratamento do contexto, outros projetos, tais como o **Aura** [39], o **RCSM** (*Reconfigurable Context-Sensitive Middleware*) e varios outros encontrados na literatura têm objetivos comuns, que são na verdades, os desafios da computação ubíqua, a saber: tratamento da heterogeneidade de dispositivo, tratamento de contexto (localização do usuário, autenticação, atividade, presença de outras pessoas por perto, etc.); adaptação e descoberta de serviço, dentre outros.

4. Segurança

Os objetivos gerais dos sistemas computacionais com relação a segurança são:

- Integridade da informação contra a adulteração dos dados
- Confidencialidade dos dados contra a exposição dos mesmos
- Disponibilidade do sistema contra a recusa de serviço

No caso dos sistemas de computação ubíqua, estes três objetivos se mantêm e podem tornar-se críticos dependendo dos cenários de aplicação. Quando o ambiente é, por exemplo, uma rede domestica, que conecta dispositivos do tipo sensores de temperatura, atuadores que ligam aquecedores, ar-condicionado etc., e que podem ser configurados a distancia, via web, a segurança no tangente ao controle de acesso e integridade dos dados é critica, uma vez que "crackers" *(hackers* do mal), podem entrar no sistema e alterar valores de configuração, como temperatura do quarto do bebe, que pode levar ao comprometimento do bem estar dos moradores. Os riscos de acesso indevido e adulteração de dados se repetem em outros ambientes, como dentro de um automóvel, por exemplo, quando prevê-se que dispositivos internos poderão ser acessados e mantidos remotamente pela montadora ou pelos fabricantes de peças.

Em outros cenários, como os de *m-commerce* (comercio móvel), os usuários acessam a Internet e efetuam compras, fornecendo dados pessoais como o numero do cartão de credito e endereço do cliente para a entrega de mercadorias. Este tipo de cenário não difere muito do cenário da web com acesso via PC ou wap. O agravante aqui é que o acesso é feito por uma multiplicidade de dispositivos diferentes, onde cada um pode ter um sistema operacional diferente, que juntamente com as aplicações e a limitação de recursos do próprio dispositivo, podem abrir novas brechas de segurança que são difíceis de serem identificadas.

Os cenários que ilustram possíveis problemas ocasionados por brechas na segurança são inúmeros e é assustador o que pode acontecer caso estas brechas sejam exploradas . Não se tem a pretensão aqui de identificar e enumerar todos os possíveis problemas de segurança em sistemas ubíquos, até porque este é um tema importante de pesquisa que está sendo explorado atualmente.

Desta forma, e como existe vasta literatura na área de segurança de sistemas tradicionais e da web, serão discutidos neste capitulo, apenas os principais motivos que provocam as brechas nos sistemas de segurança de uma maneira geral e os conceitos básicos de criptografia, uma das técnicas mais importantes de segurança.

4.1 O que origina as brechas na segurança?

Os principais motivos que originam brechas na segurança e por conseqüência possibilitam invasões são [40]:

- **Má configuração de Computadores e Sistemas:** a má configuração ocorre tanto em softwares específicos de segurança, como *firewalls*, quanto em outros softwares não propriamente relacionados com segurança. A utilização de versões antigas de softwares,

por exemplo, também contribui para o aparecimento de falhas de segurança, pois estes podem conter erros já conhecidos que facilitem uma invasão. Por exemplo, o uso de senhas padrão, após a instalação do sistema, é uma causa comum de invasões. Nos sistemas ubíquos, dados e aplicações podem estar parcial ou totalmente localizados em servidores que são acessados por dispositivos diferentes. Estes servidores podem estar sujeitos a má configuração.

- **Falhas de Softwares:** este tipo de falha é bastante freqüente e pode causar o funcionamento inadequado de um software sob certas condições, comprometendo o funcionamento do sistema ou possibilitando exploração de fraquezas para danificar ou ganhar acesso (como a não criptografia de senhas pelos protocolos *telnet* e *ftp*). Estes problemas, às vezes, são difíceis de serem corrigidos se não existir uma atualização do software para corrigir a falha. Neste caso, a solução mais óbvia seria não utilizar o programa, mas isto muitas vezes pode ser inviável devido à grande utilização deste. No caso da computação ubíqua, os problemas de software podem estar no dispositivo e no servidor (sistema operacional, aplicações, *frameworks*, *middleware* etc.)

- **Falta de Conhecimento por Parte do Usuário:** muitas falhas poderiam ser evitadas se os responsáveis pelos sistemas e usuários tivessem o conhecimento sobre os problemas e soluções. Por exemplo, um administrador de rede deveria ter conhecimento sobre as falhas dos softwares utilizados em sua rede e nos seus servidores ou o usuário deveria verificar se o seu navegador está utilizando uma conexão segura no momento que este deseja efetuar uma compra, via Web, de um produto. Desta maneira todas as pessoas que utilizam um sistema devem ter noções básicas de segurança. Entretanto este é um dos problemas de segurança mais difíceis de ser resolvido. Em ambientes de computação ubíqua, onde um dos objetivos é tornar o computador e a rede invisíveis para o usuário, embarcados em ambientes inteligentes que trocam informações e engatilham ações independentemente do usuário, é difícil visualizar o usuário se preocupando com questões de segurança. Ainda há muito a avançar nesta área.

A Autenticação como forma de barrar o acesso indevido aos sistemas

No processo de autenticação, um usuário prova que é quem ele diz que é (dispositivo e servidor podem sofrer autenticação também). Dependendo da aplicação, apenas uma senha não é suficiente ou não é viável. No controle do aquecimento doméstico, por exemplo, a idéia de proteger o patrimônio e o bem estar apenas com uma senha que eventualmente pode ser "quebrada" por *crackers*, não é muito alentador. Alternativas como a Identidade Eletrônica, que usa cartões inteligentes e criptografia, são usadas.

Criptografia

A criptografia endereça três questões: ajuda a autenticar pessoas e transações, provê acesso seguro a dados ou serviços e protege a privacidade da comunicação. Na dinâmica da criptografia, um transmissor encripta dados usando uma chave. O recipiente dos dados decripta os dados de volta numa forma utilizável usando uma chave, como mostra a figura 7. Os algoritmos criptográficos classificam-se em: Simétricos e Assimétricos.

Os algoritmos simétricos podem ser divididos em dois grupos, baseados na forma como os dados são processados:

- ***Bloco cifrado*** - quebra os dados em blocos de tamanho fixo (64 bits). Estes algoritmos são padronizados na indústria e são os mais usados. Exemplos incluem: DES *(Data Encryption Standard)*, DES Triplo, AES *(Advanced EncryptionStandard)*, RC2, RC4, RC5
- ***Cadeia cifrada*** - encripta cada byte separadamente

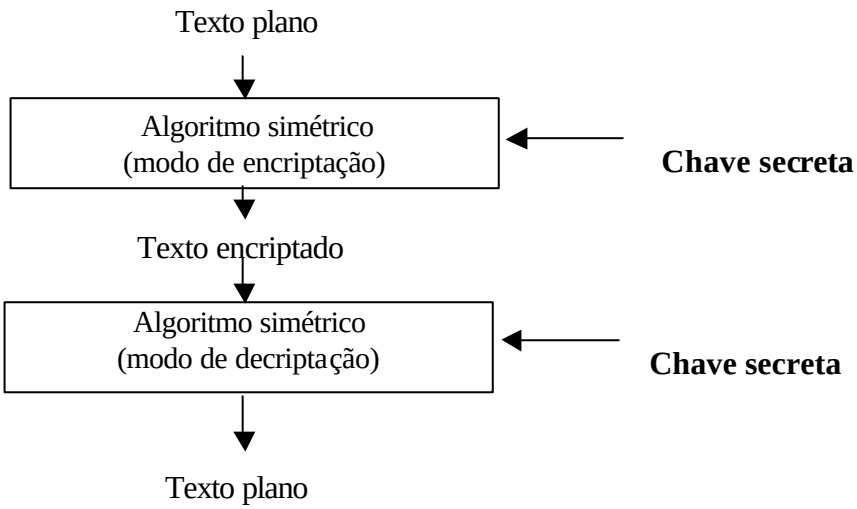

Figura 7. Dinâmica da criptografia

Problemas com os Algoritmos Simétricos

Os algoritmos simétricos apresentam varios problemas: usam a mesma chave para encriptar e decriptar; a chave em si tem que ser distribuída de forma segura, por exemplo, através de meio de transmissão não eletrônico como telefone ou correio; quando a troca de dados é entre vários participantes, deve haver chaves separadas para cada um.

Algoritmos Assimétricos

Para resolver o problema de distribuição de chaves que ocorre com os algoritmos simétricos, foram desenvolvidos os algoritmos assimétricos, também conhecidos como algoritmos de chave publica. As principais áreas de uso da criptografia assimétrica incluem: distribuição de chaves, geração de assinaturas digitais e encriptação e decriptação de informação. Nos algoritmos assimétricos, todos têm duas chaves: uma pública e uma privada. As chaves são usadas em pares: uma para encriptação e outra para decriptação. A chave pública é acessível a todos, e pode ser solicitada de um terceiro confiável, que garante que uma chave pública específica realmente pertença a aquela pessoa específica. A chave privada fica com o dono e é mantida em segredo. É impossível computar uma chave dada outra chave. O cartão inteligente é o meio ideal de armazenar esta chave – o cartão gera as assinaturas ou dados encriptados no cartão e a chave privada nunca deixa o cartão.

Exemplo de uso do algoritmo assimétrico para encriptação de dados: se A quer enviar informação para B que só pode ser lido por B, A usa a chave publica de B para encriptar a informação. Somente a chave privada de B pode decriptar a informação e só B pode ler a informação enviada por A. Os principais algoritmos assimétricos incluem: RSA (Ron Reveste, Adi Chamar e Leonard Aclimam), DAS (Algoritmo de Assinatura Digital) e ECC (Criptografia por Curvas Elípticas).

Forma híbrida de utilização dos Algoritmos Simétricos e Assimétricos

Os algoritmos simétricos são 10 a 1000 vezes mais rápidos que os algoritmos assimétricos, dependendo se a implementação é em software ou em hardware. Os dois algoritmos podem ser usados de forma híbrida: o assimétrico para fazer a distribuição das chaves e o simétrico para encriptar dados.

4.2 Aspectos de Segurança nos Dispositivos

O nível de segurança de um dispositivo varia muito de acordo com tipo de dispositivo, seu hardware, sistema operacional e a forma como executa as aplicações. Alguns aspectos que influem na segurança dos dispositivos incluem:

- a execução de software que não muda, no dispositivo, fica menos sujeita a brechas de segurança do que a execução de software que é carregado arbitrariamente no dispositivo.
- existência de hardware de proteção de memória é importante para isolar as aplicações umas das outras.
- o poder computacional do dispositivo pode influenciar na segurança - quando este é limitado, o suporte a assinaturas e encriptação pode ser implementado apenas com chaves de pequeno comprimento, deixando o sistema mais vulnerável.

Um ponto importante na segurança dos dispositivos está em poder dos projetistas da aplicação: na implementação de aplicações que precisam de segurança, o projetista deve estar ciente do nível de segurança que o dispositivo cliente suporta!!!

4.3 Aspectos de Segurança no Servidor

Todos os aspectos relativos a segurança de servidores nos sistemas tradicionais, devem ser considerados nos sistemas de computação ubíqua, tais como:

- Autenticação do servidor da aplicação e/ou dos dados, por exemplo, através de certificados digitais, para garantir que a interação está ocorrendo com o verdadeiro servidor para o dispositivo cliente.
- Restrição de acesso ao servidor
- Atualização de todos os softwares no servidor
- Utilização de ferramentas de busca de vulnerabilidades
- Restrição de funcionalidade

- Restrição de relações de confiança com outras máquinas etc.
- Proteção das chaves utilizadas pelo algoritmo de criptografia na proteção da comunicação com os dispositivos cliente
- Proteção do servidor contra comprometimento por usuários internos da rede/sistema e atacantes externos.
- Proteção da base de dados (que pode estar no mesmo servidor da aplicação, o que não é aconselhável, ou em servidores diferentes)
- Proteção contra acesso e alteração não autorizados às informações
- Proteção contra acesso e alteração não autorizados a *backups* ou base de dados replicada
- Proteção contra comprometimento do servidor (invasões e ataques do tipo recusa de serviço - *Denial-of-Service*).
- Utilização de criptografia para proteger dados confidenciais, tais como os números de cartões de crédito;
- Instalação e configuração de *firewall* entre a Internet e a *intranet* da empresa provedora de aplicações, visando a proteção do servidor de aplicações e/ou dados contra acesso não autorizado e ataques de *hackers*
- Instalação e Configuração de *firewall* entre a rede interna e o servidor de aplicação e/ou dados, visando a limitação do acesso de dentro da Intranet a dados ou aplicações que devem ser protegidos inclusive dos usuários da rede interna da empresa provedora de aplicações/dados.

O controle de acesso e a autenticação devem ser capazes de tratar diferentes tipos de dispositivos cliente e suas linguagens de marcação, tais como HTML para clientes PC, WML para clientes WAP, VoiceXML para telefones apenas de voz, HTML simples para PDAs etc.

A privacidade deve ser garantida através da proteção do trafego de dados contra a exposição dos mesmo. Conexões seguras devem ser estabelecidas fim-a-fim entre os dispositivos e o provedor da aplicação. No PC, o protocolo SSL *(Secure Socket Layer)* é usado para garantir a privacidade dos dados que trafegam na rede. Quando os dispositivos não têm capacidade de encriptação, um *gateway* pode ser colocado entre os dispositivos e o provedor da aplicação, de tal forma que a transmissão seja feita com texto claro até o *gateway* e de lá até o provedor, segue encriptado [20].

Conforme será visto no capitulo de Conectividade, algumas tecnologias de conexão, tais como Bluetooth, asseguram a privacidade na comunicação entre dispositivos. Entretanto, a rede sem fio, principalmente a de longa distancia, ainda está sujeita a muitos problemas de segurança que certamente abrirão brechas para invasões nos sistemas de computação ubíqua.

5. Conectividade

Conforme visto nos cenários do capitulo 1, a conectividade é um dos aspectos chave no conceito da computação ubíqua. O conceito da comunicação de qualquer lugar e a qualquer momento é o conceito das redes que oferecem serviços de comunicação pessoal, como voz e dados. Os serviços de Comunicação Pessoal – PCS referem-se a sistemas celulares que operam em bandas de alta freqüência. Com a evolução da tecnologia de radio, novas gerações de sistemas celulares têm surgido que oferecem serviços multimídia mais sofisticados de comunicação pessoal. As gerações G1, G2, 21/G2 e G3 refletem essa evolução. Entretanto, os serviços a serem oferecidos na computação ubíqua vão além dos serviços de comunicação pessoal. Outras tecnologias sem fio de media e curta distancia surgiram para prover a interação entre dispositivos, de forma transparente para o usuário. Esta interação vai desde a conexão sem fio de um PC a seus periféricos, eliminando assim o cabeamento excessivo, até a conexão e comunicação entre dispositivos nos mais diversos ambientes (residência, escritório, chão de fabrica, sala de aula, lojas e shoppings, hotéis, aeroportos, automóvel etc) para a realização das mais diversas tarefas. Tecnologias de comunicação sem fio de curta e media distancia, tais como o Bluetooth, Wi-Fi e HomeRF, juntamente com as redes de longa distancia, compõem uma estrutura básica de suporte aos sistemas de computação ubíqua.

Este capitulo dá uma visão geral sobre as tecnologias sem fio de rede de longa, media e curta distancia.

5.1 O Sistema Celular

Um único sistema celular interconecta varias pequenas áreas de cobertura de radio, denominada de “células”. A idéia básica do sistema celular é o reuso da freqüência, que proporciona que um mesmo conjunto de canais possa ser reutilizado em áreas geográficas diferentes, distantes o suficiente umas das outras de forma que interferências estejam dentro de limites toleráveis. O conjunto de todos os canais disponíveis no sistema é alocado a um grupo de células denominado *cluster*. O numero de células por *cluster* constitui o padrão de reuso que está intimamente relacionado com a capacidade e a qualidade da transmissão [41]. Quanto menor o tamanho da célula, maior o numero de células que podem ser empregadas, aumentando assim o numero de canais que podem ser atribuídos aos usuários. Desta forma, a capacidade de um sistema celular é determinada pelo tipo de padrão de reuso, bem como a cobertura das células. Os sistemas sem fio são normalmente projetados usando um padrão de reuso de 4, 7 ou 12 células. Um padrão típico de reuso é o de sete células. Conforme aumenta a demanda por mais capacidade ou os níveis de serviço precisam ser melhorados, células adicionais têm que ser empregadas.

Os três principais componentes de um sistema celular, mostrados na figura 8, são:

- Estação móvel
- Estações Base
- Equipamento central de processamento (ou Centro de Comutação móvel - MSC)

A **Estação Móvel** é a entidade que provê a função de telefone móvel utilizado com o sistema celular. É equipada com uma unidade de controle, um transceptor e sistema de antena que permite a comunicação com a estação-base mais próxima. A estação móvel gera um sinal de identificação para que a estação-base possa identificá-la quando ativada. Para colocar uma chamada, sinaliza o inicio da chamada para a estação-base via um canal de controle. Para operar em modo analógico e digital, tanto a estação rádio-base quanto a estação móvel precisam ser modo dual (unidade de modo dual responde primeiro a um canal digital).

As **Estações-Base** gerenciam os canais de rádio dentro da célula. Contêm uma unidade de controle, equipamento de radio-base e uma antena. A estação-base fornece a ligação entre a estação móvel e o equipamento central de processamento. Supervisiona chamadas, diagnostica problemas, passa mensagens de dados para o equipamento central de processamento e para as estações móveis, monitora a potência do sinal da estação móvel, varre todas as estações móveis ativas nas células adjacentes e relata a potência do sinal para a central, que mapeia todas as unidades móveis - este mapeamento determina qual célula deve servir uma estação móvel quando esta cruza de uma célula para outra (*handoff*). A área de cobertura é configurável e pode ser projetada baseada na geografia, densidade demográfica e requisitos de capacidade geral. três descrições gerais de tamanhos de células são possíveis [42]:

- *Macrocélula* – célula típica em um sistema celular. Oferece cobertura em um raio geralmente maior que 300m e até 40km
- *Microcélula* – Cobertura típica num raio entre 60 e 300 m. As microcelulas são muitas vezes usadas para aumentar a capacidade do sistema, preencher pontos mortos ou para reduzir congestionamentos de handoff e de trafego em geral.
- *Picocélulas* – Oferece cobertura em um raio de menos de 60 m. É ideal para o interior de prédios ou campus local.

A potência do sinal de transmissão e o tamanho da antena ditam a cobertura de uma célula.

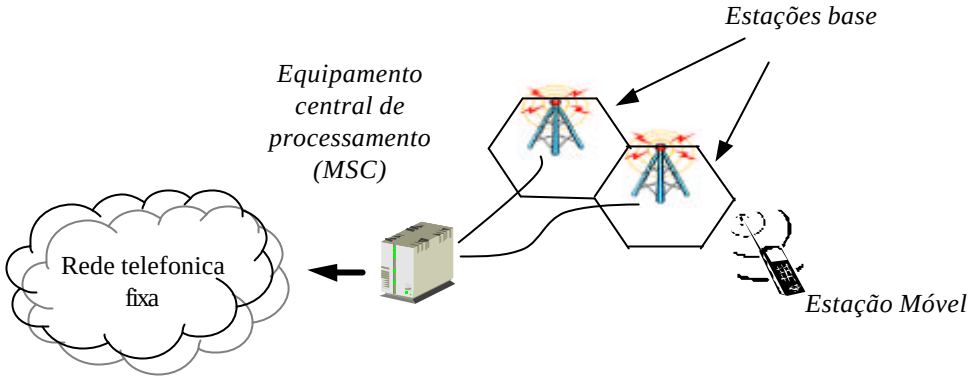

Figura 8. Um sistema celular típico

O **equipamento central de processamento** é conhecido como Centro de Comutação Móvel – MSC. É o centro nervoso do sistema, pois controla a operação móvel dentro das células. Pode alojar sistema de bilhetagem, arquivos de assinantes, informação estatística sobre volume de tráfego e contabilizações em geral. O MSC é conectado às estações-base, através de linhas terrestres, enlaces microondas ou uma combinação das duas. O número de estações-base que podem ser controladas varia. Num sistema pequeno, todas as estações base são conectadas a um MSC que faz as atribuições de canais. Em um sistema grande pode haver uma hierarquia de MSCs. O MSC é conectado à Rede de Telefonia Pública Comutada.

Handoff

A transferência de uma chamada de um sistema de serviço para outro enquanto a chamada está em progresso faz parte de qualquer sistema de comunicação móvel. Quando uma estação móvel se move entre células que são servidas pelo mesmo MSC, ocorre o denominado *Handoff* Intra-sistema. O *Handoff* intersistemas ocorre quando a estação móvel se move entre células servidas por MSCs diferentes. Neste caso, coordenação e sinalização adicional são necessárias entre os dois MSCs.

Roaming

O *Roaming* define a capacidade de uma estação móvel se mover para fora de sua área normal de serviço, através da implementação de funções de rede que rastreiam a localização do assinante e transferem essa informação do sistema de sua área normal de serviço para o sistema da área temporária sendo visitada por ele. O serviço de *roaming* é habilitado, através de uma rede de sinalização entre os MSCs, que podem ser de provedoras diferentes. A expansão da rede de sinalização e os acordos entre operadoras tornam o *roaming* mais fácil.

5.2 A evolução das Redes Sem Fio de Longa Distância

As tecnologias das redes de comunicação móvel sem fio evoluíram através de múltiplas gerações, que são descritas a seguir.

5.2.1 Geração 1: Os Sistemas Celulares Analógicos

A primeira geração de telefonia celular caracterizava-se basicamente por ser analógica, utilizando modulação em freqüência para voz e modulação digital FSK (*Frequency Shift Keying*) para sinalização. Os primeiros sistemas analógicos foram: NMT - *Nordic Mobile Telephone,* NTT - *Nippon Telephone and Telegraph,* AMPS - *Advanced Mobile Phone Service* e TACS - *Total Access Communications Systems.*

O problema com o celular da geração 1 – G1, é que as taxas de dados oferecidas eram muito baixas e quaisquer novas características de serviços exigiam alterações de hardware tanto nas estações moveis quanto nas redes celulares. A capacidade típica de processamento dos sistemas G1 era de cerca de 500 mil instruções por segundo (versus 10 a 40 MIPS na G2) [43]. Serviços de encriptação de dados, por exemplo, não poderiam ser implementados sem a digitalização da rede.

5.2.2. Geração 2: Os Sistemas Celulares e os Serviços de Comunicação Pessoal/ Redes de Comunicações Pessoais– PCS/PCN

A segunda geração de sistemas celulares caracterizou-se pela digitalização do sistema que oferece as seguintes vantagens sobre os analógicos: técnicas de codificação digital de voz mais poderosas, maior eficiência espectral, melhor qualidade de voz, maior capacidade de utilização, provimento de segurança nas comunicações e transmissão de dados. O sistema celular e o PCS operam em faixas diferentes: 1.8-1.9 para PCS e 800-900MHz e competem entre si pelo mercado. O PCS possui tecnologia superior ao do sistema celular digital, e visa o mercado de massa oferecendo serviços multimídia a baixo custo. Entretanto os sistemas celulares, com toda a infra-estrutura existente de milhões de assinantes no mundo todo, começou a incorporar serviços ditos do PCS. Como a implantação do PCS exige a superação de limitações complicadas, tais como implantação de infra-estrutura e liberação do espectro usado por outros serviços [41], além da superioridade em relação ao sistema celular ficar cada vez menos evidente na medida que estes começam a oferecer os serviços até então inerentes ao PCS, fica também mais difícil a visão da entrada maciça do PCS no mercado. Os principais sistemas G2 incluem [43]:

- PDC/JDC – *Personal Digital Cellular* ou *Japanese Digital Cellular*. (Amplamente implantado no Japão)
- TDMA (D-AMPS/IS-54/IS-136) - Implantado nos EUA e América do Sul
- CDMA (IS-95) – amplamente implantado nos EUA
- GSM – *Global System for Mobile Communication*. Padrão desenvolvido na Europa e implantado amplamente no mundo todo

Dentre os métodos de acesso da segunda geração destacam-se: o TDMA (*Time Division Multiple Access*) e o CDMA (*Code Division Multiple Access*). O GSM (*Groupe Speciale Mobile/Global System for Mobile Communications*), padrão europeu muito difundido no mundo todo, utiliza o método de acesso TDMA, padronizado pela norma IS-136. O CDMA, padronizado pela norma IS-95, é muito utilizado pelas operadoras americanas, sul americanas e asiáticas. O GSM foi padronizado em 1991 e oferece *roaming* internacional, além de serviços de localização automática, encriptação do sinal, autenticação e autorização do usuário, anonimato, mensagens curtas - SMS, fax e serviço de dados. Dentre os serviços oferecidos pelo GSM, o SMS é o serviço mais popular (mais de 12 bilhões de mensagens/mês no mundo todo – dados de 2000) [44].

O padrão GSM está sendo adotado em mais de 160 paises (mais de 400 milhões de assinantes no mundo todo – inclui GSM 1.8 e 1.9). Nos EUA prevalece o AMPS e , devido a falta de padronização, foram adotados três sistemas incompatíveis: AMPS analógico, TDMA e CDMA na faixa de 1.9GHz. Também foi adotado o padrão GSM na faixa de 1.9GHz. No Brasil, está sendo adotado o padrão GSM na faixa de 1.8GHz [45].

Após a implantação dos sistemas G2, que possibilitou atender a demanda reprimida por telefonia móvel, um novo cenário de evolução para as comunicações sem fio se delineou, apontando para o surgimento de uma nova geração de sistemas. Os principais fatores que motivam esse cenário são [46]:

- Demanda por serviços de banda larga, isto é, serviços que requerem altas taxas de transmissão (serviços de Multimídia);
- A enorme expansão da Internet e de seus serviços associados, como o correio eletrônico, que demandam acesso à rede a qualquer instante;
- Grande desenvolvimento de meios de transmissão via fibra óptica, o que capacitou a rede ISDN (*Integrated Service Digital Network*) para o fornecimento de serviços de altas taxas de transmissão. Com isso a interface aérea passou a ser uma limitação à transmissão desses serviços para usuários móveis.

A implantação de um sistema que suporte às demandas acima exigiria uma atualização considerável dos equipamentos existentes no sistema G2. Para evitar esta atualização, que implicaria em altíssimos custos, uma geração intermediaria foi criada, denominada 2,5 G, descrita a seguir.

5.2.3 A Geração 2,5

A geração G2,5 refere-se aos sistemas celulares com serviços e taxas adicionais àquelas oferecidas pelos sistemas G2, baseados no GSM, porém ainda não caracterizados como G3. A geração G2,5 compreende: o Serviço Geral de Radio por Comutação de Pacote - GPRS (*General Packet Radio Service*) e Dados a alta velocidade por comutação de circuito - HSCSD (*High Speed Circuit-Switched Data*).

O GRPS, assim como a Internet, usa a comutação de pacotes que, diferentemente da comutação de circuito, só usa a rede quando há dado para ser enviado. Com o GPRS, os mesmos serviços da Internet, tais como e-mail e navegação on-line, são mais eficientemente compartilhados pelos seus usuários. Padronizado pelo ETSI – *European Telecommunications Standardizations Institute*, representa a primeira implementação de comutação de pacotes dentro do GSM. O objetivo do GPRS é conduzir voz nos pacotes, mas, por enquanto, precisa coexistir com o GSM – circuito comutado e comutação de pacotes.

O HSCSD supera a taxa de 9,6kbps do GSM original, agregando múltiplos canais básicos de trafego para serviços de dados, que atingem até 57,6kbps. Diferentes números de canais podem ser atribuídos para transmissão de dados na direção terminal cliente ? provedor *(upstream)* e na direção provedor ? terminal cliente *(downstream)*. Apresenta uma capacidade máxima de taxa de dados igual a 64 kbps com a possibilidade de aumento de 2 a 4vezes, se utilizada a tecnologia de compressão de dados do GSM, possibilitando aplicações de transmissão de arquivos longos e recepção de vídeo em movimento. As vantagens e desvantagens da geração G2,5 são descritas abaixo.

Vantagem do HSCSD

Compatível com o GSM , isto é, não requer grandes alterações ou investimentos pelo provedor do serviço (envolve apenas atualização do software da rede e adição de *gateways* que permitam conexão para as redes de dados) [44].

Desvantagens do HSCSD

- Exige novo aparelho para o assinante (terminal do usuário)
- Não adequado para atender requisitos típicos da Internet, como o envio de pequenos surtos de dados do cliente, devido à atribuição estática de canais
- Pode apresentar possível perigo para a saúde pois transmite mais que um telefone GSM comum, causando aumento de consumo de energia e superaquecimento (primeiros terminais se incendiaram
- Desperdiça largura de banda, devido a utilização de comutação de circuito. Enquanto estiver on-line, cada usuário precisa manter um circuito aberto inteiro de 9.6kbps ou mais

Vantagens do GPRS

- Oferece transição suave para as redes G3 (arquitetura de rede, serviços e modelo de negócios são similares)
- Oferece conexão permanente com a Internet com taxas de transmissão de dados de 20 a 171kbps
- Otimiza tempo de ar e consumo de energia, uma vez que, com a comutação de pacotes, o ar é usado apenas quando dados estão sendo enviados
- Suporta conexões virtuais permanentes para fontes de dados , ou seja, a informação pode chegar automaticamente
- Acesso a Internet móvel não interfere com a recepção de chamadas (sessão de dados é suspensa enquanto a chamada é atendida)
- Capacita serviços de mensagem por difusão, difusão seletiva ou *unicast*.

Desvantagens do GPRS

- Não garante largura de banda (importante para vídeo em tempo-real)
- Apresenta problema de superaquecimento e mesmo limite de velocidade do HSCSD

Apesar do avanço da tecnologia G2 para a G2,5, existe ainda demanda por largura de banda cada vez maior, o que leva a tecnologia da comunicação sem fio para a terceira geração – G3, que visa uma total integração, em nível global, da comunicação pessoal.

5.2.4 A Geração G3

Os celulares de terceira geração representam a convergência das seguintes tecnologias: Internet (navegação web, e-mail, informações, m-commerce), telefonia (voz, vídeo, fax etc) e mídia de difusão (TV, radio, entretenimento e informação, serviços de localização) para o suporte a seis grande classes de serviços: voz, mensagens, comutação de dados, multimídia, multimídia de alto padrão e multimídia interativa de alto nível.

As principais características dos sistemas G3 são conhecidas coletivamente como IMT-2000 *(International Mobile Telecommunications)* e compreendem: [47] utilização no

mundo todo; utilização em todas as aplicações moveis; suporte a comutação de pacotes e comutação de circuito; taxa de dados de até 2Mbps e alta eficiência do espectro. Os tipos de serviços a serem oferecidos pelo IMT-2000 são mostrados na tabela 4.

A proposta mais importante do IMT-2000 é o UMTS (W-CDMA). O W-CDMA foi projetado para permitir interoperação com o GSM, apresentando um canal de 5 MHz que representa quatro vezes a capacidade do CDMAOne e vinte e cinco vezes a capacidade do GSM, com técnica de codificação que oferece velocidade máxima de 4Mbps [44].

Cada terminal W-CDMA pode acessar vários serviços diferentes ao mesmo tempo, entre eles: filmes, videoconferência, acesso a Internet, jogos multimídia, e outros como o *roaming* global, esperado para 2005. Até que o WCDMA esteja totalmente implantado, os usuários deverão ter dispositivos multi-modo, capazes de comutar para qualquer tecnologia disponível no momento [48]. A tabela 5 mostra o tempo de entrega de dados nas diferentes gerações dos sistemas celular.

Motivos para o atraso na adoção do celular G3

Alguns dos problemas que podem atrasar a entrada de novos serviços em operação incluem [41] [20]:

- O espectro alocado para o UMTS não está disponível em alguns paises. Nos EUA, por exemplo, essas freqüências são usadas por estações de TV cujos contratos expiram em 2004 com opções para extensão do prazo de concessão. A liberação deverá ser progressiva, de acordo com a demanda, o que pode levar a atrasos na entrada em operação dos serviços
- A G3 requer novo hardware para transmissão, de modo que uma ampla coberturaG3 pode demorar.

Motivo para Otimismo

O W-CDMA foi originado no Japão e adotado para uso no G3 pelo ETSI. - *European Telecommunications Standardizations Institute*. Resolvidos os problemas de patenteamento do W-CDMA, ele deverá ser amplamente disponibilizado uma vez que ele é suportado pela comunidade GSM, que fornece a tecnologia dominante hoje [20]. A tabela 6 mostra um resumo das características das gerações G2, G2,5 e G3.

Os sistemas celulares atuais, por comutação de circuitos, provêem conexão dependendo da disponibilidade de um circuito. No UMTS, a conexão por comutação de pacotes, usando o Protocolo da Internet (IP), provê uma conexão virtual que está sempre disponível para qualquer ponto da rede. UMTS promete realizar um ambiente *Virtual Home* em que um usuário em *roaming* poderá ter acesso aos mesmos serviços que um usuário está acostumado a ter quando está em casa ou no escritório.

Classe de Serviços	Velocidade de saída dos dados	Velocidade de entrada dos dados	Exemplo	Comutação
MM interativa	256kbps	256kbps	Video-conferencia	Circuito
MM de alto nivel	20kbps	2Mbps	TV	Pacote
Multimidia	19.2kbps	768kbps	Navegação web	Pacote
Comutação de dados	43.2kbps	43.2kbps	Fax	Circuito
Mensagens simples	28.8kbps	28.8kbps	E-mail	Pacote
Fala	28.8kbps	28.8kbps	Telefonia	Circuito

Tabela 4. Tipos de serviços no IMT-2000 (adaptado de [48]).

Serviço	RDSI	G2 (GSM)	G2,5 (GPRS)	G3 (UMTS)
E-mail (10KB)	1s	8s	0,7s	0,04s
Página web (9KB)	1s	9s	0,8s	0,04s
Arquivo texto (40KB)	5s	33s	3s	0,2s
Relatório (2MB)	2m	28m	2m	7s
Clip de Video (4MB)	4m	48m	4m	14s
Filme com qualidade TV (6GB)	104h	1100h	52h	~5h

Tabela 5. Tempo de entrega nas diferentes gerações [44]

Sistemas G2	Sistemas G2,5	Sistemas G3
▪ Chamadas telefônicas ▪ Correio de voz ▪ Recebimento/envio de simples mensagens de e *email* ▪ **Velocidade:** 10 Kbps	▪ Chamada telefônica/fax ▪ Correio de voz ▪ Recebimento/envio de mensagens maiores ▪ Navegador *Web* ▪ Navegação/Mapas ▪ Noticias atualizadas ▪ **Velocidade:** até 171.2 Kbps (GPRS) e até 256Kbps (HSCSD com compressão de dados)	▪ Chamadas telefônicas/ fax ▪ *Roaming* Global ▪ Recebimento/envio de mensagens maiores ▪ Web de alta velocidade ▪ Navegação/Mapas ▪ Vídeo Conferência ▪ TV *Streaming* ▪ Agenda eletrônica ▪ Aplicações multimídia ▪ **Velocidade:** 144Kbps – 2Mbps

Tabela 6 – Comparação entre as gerações de celulares (adaptada de [49])

5.2.5 A Geração G4

A quarta geração das comunicações móveis envolve uma mistura de conceitos e tecnologias para inovações na alocação e utilização do espectro, comunicações a rádio, redes, serviços e aplicações. Consiste na evolução da rede G3, de suporte a redes de pacotes e de comutação de circuitos, para uma rede apenas de pacotes, além da possível convergência com as redes locais sem fio - WLAN para o suporte a aplicações de conteúdo multimídia mais sofisticados, tais como: vídeo colorido com imagens de alta qualidade, jogos com animações gráficas 3D, serviços de áudio em vários canais, correio multimídia, serviços baseados na localização do usuário e muito mais. Prevê-se aqui taxas de dados superiores a 100Mbps e 1Gbps. Novas tecnologias de acesso como MC-CDMA (*Multicarrier-CDMA)* e OFDM (*Orthogonal Frequency Division Multiplexing*), assim como antenas inteligentes, técnicas de codificação espaço-temporal etc., devem caracterizar as tecnologias dos sistemas G4.

5.3 Redes Sem Fio de Curta e Media Distância

O objetivo maior dos sistemas celulares vistos acima é o provimento de serviços de comunicação pessoal que permitam ao usuário acessar, de forma única, serviços sofisticados de qualquer lugar do mundo e a qualquer momento. Estes serviços atendem parte dos requisitos de comunicação da computação ubíqua. Um outro requisito é a provisão da comunicação dentro de ambientes fechados, como uma sala, de tal forma que dispositivos de um lado da sala possam se comunicar com dispositivos do outro lado. Esta seção descreve as principais tecnologias de rede de media e curta distancia: Bluetooh, FIR, HomeRF e 802.11 [50].

5.3.1 A Tecnologia Bluetooth

A comunicação sem fio para comunicações locais surgiu de uma necessidade de eliminar os fios que ligam dispositivos a acessórios. A Ericsson foi uma das primeiras empresas a reconhecer essa necessidade e, a partir daí, a identificar outras aplicações com enorme potencial, tais como: um telefone móvel se comunicando com uma impressora, ou um PDA se comunicando com o PC para realizar alguma tarefa, sem qualquer configuração ou intervenção manual do usuário.

Em 1998, um consorcio liderado pela Ericsson, com a participação da Nokia, IBM, Toshiba e Intel foi criado para desenvolver a especificação de um padrão global aberto para a conectividade sem fio entre os dispositivos de telecomunicações e os de computação. Esta especificação foi denominada Bluetooth. Muitas outras empresas se associaram ao consorcio deste então, mais de 2000. O nome Bluetooth foi dado em homenagem a Herald I. Bluetooth – rei da Dinamarca (910-985), primeiro a unificar o país.

A comunicação com o Bluetooth é onidirecional com alcance de até 10m. Suporta serviços de transmissão síncronos e assíncronos. A taxa de transferência de dados é de até 1Mbps. Conecta dois ou mais dispositivos que não estão na linha de visão um do outro. Exemplos incluem: o envio de fotos de uma câmera digital para o disco rígido; a conexão de um PCs com seus periféricos - impressoras, *scanners*, teclados, mouse, microfones etc.; a conexão de voz em tempo-real entre fone de ouvido e telefone móvel. O Bluetooh oferece uma solução em um chip único. Prevê-se que a tecnologia seja utilizada em mais de 100 milhões de dispositivos até 2002.

O Bluetooth utiliza a faixa de freqüência ISM - Medico-Cientifica-Industrial de 2,45GHz (2,4000 a 2,4835). Esta faixa é muito ruidosa pois é também utilizada por outros dispositivos, tais como: portões automáticos, monitores de bebes, telefones sem fio etc. De modo a minimizar os problemas de interferência, os rádios Bluetooth utilizam uma técnica denominada espalhamento do espectro por de salto de freqüência – FHSS *(Frequency Hopping Spread Spectrum)*, que consiste no seguinte: a banda é dividida em 79 freqüências portadoras (com espaços de 1MHz entre elas); um sinal salta de uma freqüência para outra no decorrer de uma mesma transmissão. Isto significa que uma transmissão Bluetooth não permanece numa mesma freqüência o tempo bastante para ser afetada por ruídos nessa freqüência. Os padrões de salto de freqüência são pseudo-aleatórios e têm que ser sincronizados para poder haver comunicação.

Piconet

Um grupo de dispositivos Bluetooth comunicantes constituem uma Piconet. Na piconet, um dispositivo age como principal e os outros são secundários. O primeiro dispositivo a iniciar a comunicação assume o papel de principal. Dispositivos secundários sincronizam seus relógios internos e seqüências de salto de freqüência com os do principal. Cada piconet pode conter até oito dispositivos Bluetooth diferentes, entretanto, numa transmissão de voz full-duplex (comunicação nas duas direções), apenas três dispostivos são suportados ao mesmo tempo. O mesmo canal de salto de freqüência é compartilhado por todos os dispositivos de uma mesma piconet. Os relógios internos dos dispositivos secundários são sincronizados ao relógio do dispositivo principal, de tal forma que, todos os dispositivos na piconet saltam de

freqüência na mesma seqüência. Cada piconet tem então uma identidade própria, baseada em canais de salto de freqüência diferentes. Assim, varias piconets podem conviver num mesmo espaço compartilhado sem interferências entre si [50].

Redes de Dispersão

As redes de dispersão *(scatternet)* são formadas por duas ou mais piconets que se sobrepõem. Até dez piconets, com 80 dispositivos bluetooth diferentes, podem ser incluídas numa rede de dispersão – mais que isso pode saturar a rede, uma vez que a banda é dividida em apenas 79 freqüências. Um dispositivo em uma piconet pode comunicar-se com dispositivos em outra piconet. Um dispositivo principal em uma piconet pode ser secundário em outra. Uma rede de dispersão com duas piconets é mostrada na figura 9.

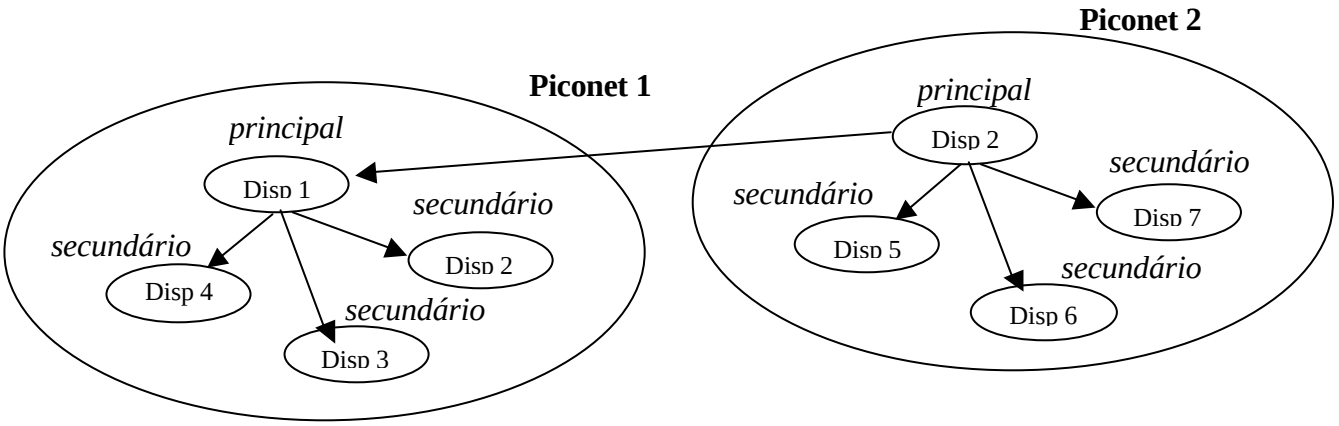

Figura 9. Uma Redes de dispersão com duas piconets, onde dois dispositivos principais podem servir de ponte entre as duas redes [44].

Dispositivos dentro da piconet estabelecem enlaces para comunicação sob demanda. Dois tipos de enlaces são definidos:

- **SCO** – Síncrono orientado a conexão *(Synchronous Connection-Oriented)* comunicação ponto-a-ponto dedicada entre dispositivos principal e secundário. O dispositivo principal reserva *slots* de tempo fixos para comunicação e pode suportar até três enlaces SCO com o mesmo dispositivo secundário (ou outros) – cada um de até 64Kbps para voz em cada direção.
- **ACL** – Assíncrono sem conexão *(Asynchronous Connection-Less)* – utiliza quaisquer s*lots* não reservados para um enlace SCO. O dispositivo principal pode se comunicar com um secundário numa base "por-*slot*". Um dispositivo secundário só pode se comunicar com o principal depois deste tê-lo endereçado. Os dispositivos Bluetooth suportam também uma mistura de canais SCO e ACL.

A transmissão *full-duplex* no Bluetooth utiliza o esquema de TDD – Duplex por Divisão de Tempo *(Time Division Duplex)*. No TDD, cada freqüência portadora é dividida em *slots* de tempo de 625µs de duração. Os dispositivos principais transmitem em *slots* pares e os secundários em *slots* impares. Desta forma, alternando-se de um lado para outro, uma mesma freqüência pode ser compartilhada por duas transmissões diferentes.

Quaisquer dispositivos Bluetooth, dentro de um raio de dez metros, podem estabelecer uma conexão ponto-a-ponto ou ponto-a-multiponto de forma ad-hoc (imediata).

Conexões seguras com o Bluetooth

A segurança oferecida com o Bluetooth inclui autenticação baseada em chaves privadas de 128 bits e encriptação. Três mecanismos básicos são especificados:

- *Gerenciamento de Chave* – utiliza três tipos de chaves no processo de gerenciamento de chave: uma para o usuário (ou dispositivo), em que digita-se uma senha numérica; o dispositivo gera uma chave privada para o enlace e autentica com o segundo dispositivo; por fim, o dispositivo gera uma chave privada para encriptação, a partir da chave do enlace, e autentica com o segundo dispositivo.
- *Autenticação do dispositivo* – Um esquema de desafio-resposta é utilizado para verificar se o outro dispositivo reconhece uma chave secreta compartilhada. A autenticação é um sucesso quando os dois dispositivos reconhecem a mesma chave, caso contrario a conexão é abortada.
- *Encriptação de pacotes* – a especificação Bluetooth define três modos de encriptação: modo 1 - não há encriptação; modo 2 – apenas o trafego ponto-a-ponto é encriptado (o ponto-a-multiponto não); e modo 3 – todo o trafego é encriptado.

5.3.2 Infravermelho

A Associação de Dados Infravermelho – IrDA *(Infrared Data Association)* estabeleceu vários padrões de comunicação através de luz infra-vermelha, onde o IrDA-Data e o IrMC são os mais importantes. A tecnologia infravermelha suporta apenas conexão ponto-a-ponto. Tem alcance de curta distancia, de até um metro, com ângulo estreito entre o transmissor e o receptor – o cone de visão é de 30º. Qualquer sinal fora deste angulo não será recebido - os patrocinadores do IrDA sustentam que o IrDA deverá suportar um angulo de visão de 60º, o que possibilitaria suporte a um numero maior de serviços, tais como: a conexão de mouse, *joystick*, etc., no PC. O inicio da troca de dados requer que os dispositivos estejam em linha direta de visão, uma vez que os raios infravermelho não atravessam objetos sólidos. Essas duas ultimas características da tecnologia infravermelha mencionadas representam as suas maiores desvantagens.

O infravermelho é adequado para conexões de dados de alta velocidade (por exemplo, quando se quer conectar um dispositivo a uma rede com fio). Tipicamente, os dispositivos que utilizam a conexão IrDA são os periféricos de computadores (modems, impressoras, scanners, etc.), além de PDAs, relógios, equipamentos industriais etc. O infravermelho é projetado para a conexão entre dois pontos. Tem uma base de instalação bastante grande, entretanto é raramente utilizado. Isto porque no inicio de sua implantação, em dispositivos do tipo *lap-top*, por exemplo, não funcionava bem. Os problemas diminuíram muito mas percepção negativa ficou marcada, o que acabou gerando um ciclo vicioso: os fornecedores não implementam suporte total a IrDA porque os usuários não solicitam e os usuários não solicitam IrDA a menos que tenha suporte total. Apesar de barato (menos de $5), poucos fornecedores o instalam nos computadores de mesa e outros dispositivos

(exceção para a Apple). Nos serviços de fala, suporta apenas um canal digital de voz. A segurança não é especificada – é deixada para o protocolo de nível mais alto [5][20].

Apesar das limitações, a IrDA tem algumas características interessantes: a largura de banda é de 4 a 16 Mbps, muito além do 1Mbps do Bluetooth.. Os varios tipos de IrDA incluem:

- SIR *(Serial Infrared)*
  - padrão original
  - 115 kbps
- MIR *(Medium Infrared*)
  - 1.152Mbps (suficiente para transmissão e recepção de vídeo com qualidade de TV)
  - Não implementado amplamente (escolha recai no SIR e FIR)
- FIR *(Fast Infrared)*
  - Até 4Mbps
  - Embutido na maioria dos novos computadores (opção padrão do W/98 e 2000)
- VFIR *(Very Fast Infrared)*
  - Chega a 16Mbps
  - Ainda não implementado amplamente
  - Novas atualizações podem levar a velocidades de 50Mbps.

O padrão IrMC *(Infrared Mobile Communications)* define como transmitir dados de voz, *full-duplex*, por um enlace IrDA. A tabela 7 mostra uma comparação entre a IrDA e outras tecnologias concorrentes.

5.3.3 Tecnologia HomeRF

A HomeRF é uma iniciativa do grupo de trabalho HomeRF, um consorcio liderado por uma empresa parcialmente possuída pela Intel, a Proxim. Outras empresas participantes incluem: Motorola, Compaq, Intel, Cayman etc. A tecnologia HomeRF foi projetada tendo em mente redes pequenas de baixo trafego, em residências e pequenos ambientes comerciais. Visa a comunicação entre o PC e seus periféricos, bem como entre dispositivos inteligentes de uma residência. Ela não é projetada para a comunicação entre dispositivos portáteis (outras tecnologias, como o bluetooth, são melhores para isso), tampouco para a formação de redes ad hoc (imediatas) – é do tipo de rede “configure-a e deixe-a”[50].

HomeRF é baseada no Protocolo de Acesso Sem Fio Compartilhado – SWAP *(Shared Wireless Access Protocol)*, que suporta seis canais de voz baseados no padrão *Digital Enhanced Cordless Telephone* – DECT, além de um canal de dados baseado na especificação IEEE 802.11.

Versões atuais da HomeRF suportam velocidades de até 10Mbps. Assim como o Bluetooth, a HomeRF utiliza a faixa de freqüência ISM - Medico-Cientifica-Industrial de

2,45GHz. Para minimizar os problemas de interferência, os rádios HomeRF utilizam espalhamento do espectro por salto de freqüência. Para alguns tipos de dispositivos, o padrão de salto é de 50 saltos por segundo, em intervalos de 1MHz.

As conexões são do tipo ponto-a-ponto com alcance de 25 a 60 metros. Permite diversas redes no mesmo ambiente físico, com suporte de até 127 dispositivos por rede. Como tem alto consumo de energia, não é adequada para utilização em dispositivos portáteis. Não se integra facilmente a outras redes sem fio existentes. Entretanto, seu custo é relativamente baixo (menos de $200 dólares por dispositivo), é fácil de instalar, além de não exigir ponto de acesso.

A HomeRF pode coexistir com o Bluetooth da seguinte maneira: bluetooth para substituir o cabeamento domestico, por exemplo, entre o PC e seus periféricos e a HomeRF como rede local de interconexão, por exemplo, entre mais de um PC.

5.3.4 Tecnologia 802.11 – Wi-Fi

O Instituto dos Engenheiros Elétricos e Eletrônicos - IEEE desenvolveu uma especificação de rede local sem fio, denominada 802.11b, ou ainda Wi-Fi – Fidelidade Sem Fio *(Wireless Fidelity)*. A Aliança de Compatibilidade da Ethernet Sem Fio – WECA *(Wireless Ethernet Compatibility Alliance)*, patrocinada por empresas do tipo 3Com, Apple, Lucent, Nokia, Compaq etc., adotou a 802.11b como solução para redes corporativas sem fio.

Assim como o Bluetooth e HomeRF, a 802.11 utiliza a faixa de freqüência ISM de 2,45GHz. Para minimizar os problemas de interferência, a 802.11b utiliza a técnica de espalhamento do espectro por seqüência direta – DSSS *(Direct Sequence Spread Spectrum)* em que, diferentemente do FHSS utilizada pelo Bluetooth e HomeRF, os sinais são fixos dentro de um canal de 17MHz mas encobertos por ruídos que são criados com o intuito de reduzir a interferência de outros ruídos. A utilização do DSSS resulta em uma tecnologia de rede com taxa de transmissão de dados de até 11 Mbps. Por outro lado, essas redes ficam mais suscetíveis a ruídos de outros dispositivos que também utilizam a banda de 2.G4Hz – por isso esta tecnologia não é tão adequada para atender as necessidades de redes residenciais que possuem dispositivos que utilizam esta mesma banda, tais como: controle de portões automáticos, forno de micro-ondas, telefone sem fio entre outros.

As redes 802.11b são rápidas, confiáveis e têm alcance de até 100 metros, que pode ser ainda maior em áreas abertas. Do lado negativo, exigem hardware de ponto de acesso para a interconexão entre os dispositivos da rede, que tem um custo relativamente alto (equipamentos mais sofisticados podem custar acima de $1200). Esses pontos de acesso podem ter suporte para conexões Ethernet comum, tornando a 802.11b facilmente integrável com outras redes Ethernet com fio em uma corporação). A configuração desta rede é mais complexa e mais cara do que, por exemplo, a HomeRF, o que a torna não muito adequada, pelo menos no momento, para uso em residências, apesar dos fabricantes promoverem-na para este fim. Mais ainda, a 802.11b não suporta serviços de telefonia, sendo um padrão de uso estrito para a comunicação de dados – serviços de voz podem ser suportados pela 802.11b quando a voz é transmitida como dado, como é o caso na tecnologia de voz-sobre-IP. Como a 802.11 não é facilmente configurável, redes *ad hoc*, que se formam de modo

espontâneo, não são suportadas, o que pode tornar a tecnologia Bluetooth mais adequada para redes em locais públicos, tais como cybercafes, do que a 802.11.

Outras bandas de freqüência estão sendo consideradas para a especificação de redes locais sem fio que suportem velocidades mais altas. O IEEE está trabalhando na banda de radio freqüência de 5GHz para a especificação do novo padrão 802.11a que visa redes de até 54Mbps. A tabela 7 resume as principais vantagens e desvantagens de cada uma das tecnologias de redes sem fio de curta e media distancia discutidas aqui. A tabela 8, adaptada de [50], compara as tecnologias descritas acima.

5.3.5 Redes Domésticas

Os lares de hoje estão repletos de aparelhos eletro-eletrônicos sem nenhuma ou pouca interação entre eles, com um numero crescente de controles remoto e cabos. As redes domésticas surgem para integrar todos esses aparelhos, para que o usuário final possa usufruir conforto, praticidade e comodidade em sua casa.

As tecnologias discutidas nas seções acima, tais como o bluetooth e HomeRF, utilizam radio freqüência para a comunicação entre os dispositivos. Soluções que eliminam ou minimizam o cabeamento e possibilitam a interconexão de dispositivos domésticos a baixo custo, envolvem a utilização das redes de energia elétrica e de telefonia.

Vários projetos que fazem uso essas duas redes têm surgido, tais como: HomePlug e X10, que utilizam a rede elétrica, e HomePNA, que usa a linha telefônica [66] [67]. Um dos mais abrangentes é o X10, descrito a seguir.

O Padrão X10

O X10 é um padrão para a conexão e controle de aparelhos domésticos, através do cabeamento de fornecimento de energia elétrica. Sinais de controle são transmitidos entre dispositivos, através dos cabos de energia elétrica existentes na casa do usuário, usando uma portadora de sinal modulado sobre a corrente alternada normal (AC). Isto torna o X10 adequado para a conexão dos mais diversos dispositivos numa casa comum.

Os controles de comando são limitados e não há verificações de compatibilidade entre um dispositivo e os controles de comando. O endereçamento para os dispositivos é limitado a 256 endereços os quais devem ser administrados manualmente, o que em alguns casos podem causar conflitos entre dispositivos em casas diferentes, mas com o mesmo endereço. Mais informações sobre o X10 em [67]

Neste capitulo foram vistas as tecnologias de rede de longa, media e curta distancia para a interconexão de dispositivos. O próximo capitulo descreve as iniciativas que propiciam que esses dispositivos comuniquem-se entre si não apenas para a troca de informações, mas também para a busca de serviços.

	Bluetooth	**IRDA**	**HomeRF**	**802.11b (Wi-Fi)**
Aplicação	Substituição de cabeamento e na formação de redes *ad hoc*	Substituição de cabeamento e na formação limitada de redes *ad hoc*	Formação de redes domesticas e/ou de pequenas empresas	Formação de redes corporativas e/o campi
Tecnologia	RF 2.G4Hz	Luz Infravermelha	RF 2.G4Hz	RF 2.G4Hz
Tipo de Rede	1 para muitos (<= 8)	Ponto-a-ponto	1 para poucos	Muitos para muitos
Velocidade	1Mbps	4Mbps (típica)	10Mbps	11Mbps
Serviço suportado	Voz e dados	Voz e dados	Voz e dados	Dados (voz só sobre dados)
Alcance	10 ms	1 m	50 ms	100 ms
Angulo de conexão	360 graus	30 graus	360 graus	360 graus
Técnica de espalhamento do espectro	FHSS	-	FHSS	DSSS
Necessidade de ponto de acesso	Não	Não	Não	Sim
Carga de energia exigida	Baixa	Baixa	Alta	Alta
Custo por dispositivo	$ 70 a 120	< $3	$100 a 300	$250 a >1200 (custo do equipamento de ponto de acesso)

Tabela 7. Comparação entre Bluetooth xIRDA x HomeRF x Wi-Fi (adaptada de [50])

	Vantagens	Desvantagens
Bluetooth	▪ Adequado para substituir o cabeamento ▪ Adequado quando se requer a formação de redes *ad hoc* ▪ Custo relativamente baixo	▪ Sofre interferências ▪ Conexão lenta com rede local sem fio de alta velocidade (1Mbps x 11Mbps)
IrDA	▪ Rápida (4Mbps) ▪ Baixo Custo (< $3) ▪ Baixo consumo de energia ▪ Não sofre interferências ▪ Segura ▪ Amplamente instalada	▪ Faixa de alcance limitada (1 m) ▪ Cone estreito de conexão (30 graus) ▪ Requer linha direta de visão entre dispositivos comunicantes ▪ Topologia um-para-um (apenas um dispositivo se comunica por vez)
HomeRF	▪ Muito rápida (10Mbps) ▪ Fácil de instalar ▪ Suporta até 127 dispositivos por rede ▪ Suporta varias redes no mesmo ambiente físico	▪ Alto consumo de energia ▪ Sofre interferências ▪ Não integra facilmente com outras redes sem fio
Wi-Fi	▪ Muito rápida (11Mbps) ▪ Fácil integração com redes locais com fio existentes ▪ Longo alcance (100 ms ou mais)	▪ Requer equipamento de ponto de acesso (pode ter alto custo) ▪ Alto consumo de energia ▪ Sofre interferências ▪ Não suporta serviços de voz ▪ Difícil configuração e manutenção

Tabela 8. Resumo das principais vantagens e desvantagens das tecnologias discutidas (adaptada de [50]).

6. Descoberta de Recursos

No capitulo 5, foram discutidas tecnologias que suportam a conectividade entre dispositivos. As expectativas de aplicações no ambiente da computação ubíqua foram ilustradas nos cenários do capitulo 1. Naqueles cenários, os dispositivos comunicam-se entre si não apenas para a troca/sincronização de dados mas para a realização de tarefas mais complexas, como,

por exemplo, um celular que procura, dentro de um automóvel, um dispositivo que possa oferecer um serviço de alto-falante, ou ainda, um PDA, que dentro de uma rede interna de um avião, busca pelo serviço de uma tela maior para servir de saída para os dados processados no PDA. Estes cenários não podem ser realizados apenas com a conectividade. Algo mais é necessário que propicie a identificação de serviços em uma rede. Este algo a mais é a chamada Descoberta de Serviço.

Alguns grupos têm trabalhado na especificação de padrões para a descoberta de serviços. Uma dessas especificações, a especificação de Referência para Computadores em Redes Moveis - MNCRS *(Mobile Network Computer Reference Specification),* propõe um conjunto de padrões para a interação entre aplicações, servidores e protocolos de rede. Em particular, uma das especificações deste conjunto é a Descoberta de Serviço [51].

No mundo dos dispositivos móveis especializados e heterogêneos prontos para operar num ambiente ubíquo, uma arquitetura para coordenar a interação entre dispositivos é necessária. Essas arquiteturas são essencialmente *frameworks* de coordenação que sugerem certos modos e meios de interação entre dispositivos. Algumas das arquiteturas de descoberta de serviço disponíveis que merecem destaque incluem: Jini [52], Universal Plug and Play [53] e Salutation [54] – todas iniciativas de indústrias.

A coordenação da interação entre dispositivos deve oferecer aos dispositivos as seguintes funcionalidades:

- Habilidade de anunciar sua presença à rede
- Descoberta automática de dispositivos na vizinhança e também daqueles localizados remotamente
- Habilidade de descrever seus serviços e reconhecer os serviços (capacidades) dos outros dispositivos
- Capacidade de se auto-configurar sem a intervenção de um administrador

Um *framework* de coordenação pode fazer com que dispositivos tornem-se cientes uns dos outros. Para que isto ocorra, algum padrão deve ser seguido pelos dispositivos. Um dos desafios na descoberta de serviços é manter o equilíbrio entre as necessidades de padronização e a autonomia de dispositivos [55]. Jini leva padronização para um extremo enquanto UPnP leva autonomia para um outro extremo. Essas tecnologias são descritas a seguir.

6.1 Jini

A tecnologia Jini, da Sun Microsystems, é um *framework* de coordenação evoluído e adaptado de pesquisas acadêmicas e implementado especificamente para Java. Jini usa o termo *federação* para representar a coordenação entre dispositivos. Uma federação é uma coleção de dispositivos autônomos os quais podem tornar-se cientes uns dos outros e cooperar se for necessário. Um subsistema Jini contém serviços de consulta que mantêm informação dinâmica sobre dispositivos disponíveis. Esses serviços são a chave para o funcionamento adequado do sistema Jini.

Os Serviços de Consulta Jini

Os serviços de consulta do Jini, que executam numa rede, podem ser descobertos por dispositivos através de protocolos bem definidos. Todo dispositivo deve descobrir um ou mais desses serviços de consulta antes que possa entrar para uma federação. A localização desses serviços pode ser conhecida de antemão, ou eles podem ser descobertos usando difusão seletiva.

É possível formar grupos de serviços de consulta e associar nomes a eles, o que permite que um dispositivo procure por um grupo específico na sua vizinhança. Por exemplo, numa empresa, decisões administrativas podem forçar que alguns dispositivos de armazenamento pertençam a algum grupo especial, com acesso restrito. Quando um dispositivo localizar um serviço de consulta de interesse ele pode agora dizer sobre ele mesmo ao serviço (registrar), ou pedir ao serviço informações sobre outros dispositivos da federação. Quando o dispositivo se registra, ele pode associar um conjunto de propriedades que podem ser usadas por consultas de outros dispositivos.

Expondo as Interfaces

Durante o processo de registro é possível para um dispositivo enviar código Java para o serviço de consulta. Esse código é essencialmente um "proxy" que pode ser usado para contatar uma interface no dispositivo e invocar métodos desta interface. Um dispositivo pode então automaticamente transferir esse "proxy" para si mesmo e chamar métodos dentro de outro dispositivo. Isso é realizado usando a Invocação de Método Remoto – RMI (*Remote Method Invocation*) que permite a um programa Java chamar um método num programa Java executando remotamente, através de alguma interface exposta. O ponto principal aqui é que ambos dispositivos devem ter Java embutido para que tudo funcione. Porém, é possível evitar o uso de RMI e usar serviços de consulta do Jini simplesmente para extrair informação sobre um dispositivo. Um protocolo proprietário pode então ser usado para contatar o dispositivo. Nesse caso o serviço de consulta é reduzido a um diretório de serviços.

Os serviços de consulta podem formar grupos e trocar informações entre si, de forma hierárquica, para resolver uma determinada solicitação de um dispositivo. Esses serviços também tentam certificar-se que eles possuem um acurado registro do conjunto de dispositivos ativos no momento. Isso é feito através de um "contrato" de registro, por um tempo determinado, que os dispositivos devem renovar periodicamente para que os serviços de consulta mantenham suas informações atualizadas. Após o término do tempo determinado, os dispositivos que não atualizaram seus registros terão esses registros automaticamente removidos das bases de dados

6.2 Universal Plug and Play

O *Plug and Play* Universal - UPnP, desenvolvido pela Microsoft, é um *framework* definido num nível mais baixo que o Jini. Embora seja considerado, em alguns aspectos, uma extensão do *Plug and Play,* que detecta automaticamente a existência de dispositivos novos no barramento de PCs, para o ambiente de redes, seu *framework* de implementação é muito diferente do *Plug and Play*. O UPnP trabalha basicamente com as camadas mais baixas dos protocolos de rede TCP/IP, implementando padrões nesse nível ao invés do nível de

aplicação. Isto envolve a adição de protocolos adicionais ao conjunto de protocolos implementados nos dispositivos. Ao contrario do Jini, que focaliza na autonomia, o UPnP focaliza na padronização de protocolos específicos e a aderência dos fabricantes a estes novos padrões. Com a definição desses protocolos, o UpnP permite que os dispositivos construam suas próprias APIs que implementam esses protocolos, em qualquer linguagem ou plataforma que desejarem.

A Descoberta de Serviços no UPnP

O UPnP usa um protocolo especial, SSDP *(Simple Service Discovery Protocol)* que habilita dispositivos para anunciarem sua presença à rede, bem como descobrir dispositivos disponíveis. Ele pode funcionar com ou sem um serviço de diretório.

O protocolo de descoberta (SSDP) usa HTTP sobre UDP nos modos de comunicação *unicast* e *multicast*. Esses protocolos podem ser referidos como HTTPU (UDP *unicast)* e HTTPMU (UDP *multicast)* respectivamente. O processo de registro/solicitação envia e recebe dados no formato HTTP, mas contendo uma semântica especial. Uma mensagem de anúncio chamada ANNOUNCE e uma mensagem de requisição chamada OPTIONS estão embutidas em HTML para facilitar esse processo.

Um dispositivo entrando na rede pode então enviar um ANNOUNCE *multicast*, informando ao mundo sua presença. Um serviço de diretório, se presente, pode armazenar essa informação, ou outros dispositivos talvez vejam a informação diretamente. A mensagem *multicast* é enviada por um canal *multicast* reservado (endereço) ao qual todos dispositivos devem escutar. A mensagem ANNOUNCE deve também conter uma URI que identifica o recurso (por exemplo, “dmtf:printer”) e uma URL para um arquivo XML que fornece a descrição do dispositivo anunciante. Este arquivo usa uma folha de estilos adaptada para vários tipos de dispositivos. Uma requisição para a descoberta de dispositivo também pode ser *multicast* (uma mensagem OPTIONS), para a qual os dispositivos podem responder diretamente, ou pode ser direcionada a um serviço de diretório, se presente.

Configuração

O UpnP também resolve o problema da associação automática de endereços IP e nomes de DNS a um dispositivo sendo conectado a rede. Para isso, mais alguns protocolos são introduzidos. Uma maneira de associar um endereço IP é consultar um servidor DHCP na rede. Se não estiver presente, então o dispositivo pode escolher um IP de uma faixa reservada de endereços IP, conhecida como rede LINKLOCAL (faixa de endereços 169.254.x.x ?). O dispositivo deve usar o protocolo ARP para achar um endereço que não está sendo usado nesta faixa e associar a si mesmo. Isto é conhecido como AutoIP.

Para conseguir isso, um plano de DNS *multicast* foi proposto, que permite aos dispositivos conectados:

- Descobrir servidores DNS via *multicast;*
- Resolver consultas de DNS sobre eles próprios via *multicast*

Quando conectado a uma rede que não possua nenhum servidor DNS localmente, um dispositivo pode enviar um pacote *multicast* e descobrir um servidor DNS remoto e então

usá-lo para resolução de nomes na Internet (como resolução de URLs). O próprio dispositivo pode opcionalmente escutar o canal *multicast* e responder a consultas para seu próprio nome. Assim que termina o processo de descoberta e a descrição XML de um dispositivo é recebida, protocolos proprietários podem passar a controlar a comunicação entre dispositivos.

6.3 Salutation

O *framework* de coordenação *Salutation* parece ser um *framework* mais rigoroso e útil, na perspectiva de coordenação, do que Jini ou UPnP. O *Salutation* posiciona-se entre autonomia de dispositivos e padronização, o que permite a vários fabricantes adaptar muitos de seus produtos à especificação e interoperarem uns com os outros.

Descoberta de Serviços no Salutation

No *Salutation*, um dispositivo quase sempre comunica-se diretamente com um gerenciador *(Salutation Manager)*, que pode estar no mesmo dispositivo ou localizado remotamente. Um conjunto de gerenciadores (SLMs) coordenam-se uns com os outros. Eles agem como agentes que fazem tudo em nome de seus clientes. Mesmo a transferência de dados entre dispositivos, incluindo mídias diferentes e transportes, é mediada por eles. Toda tentativa de registrar um dispositivo é feita com o SLM local ou mais próximo disponível. Os SLMs descobrem outros SLMs próximos e trocam informação de registro, que é possível utilizando módulos de transporte *(Transport Managers)* que podem usar difusão internamente. Chamada remota a procedimentos (RPC) é usada opcionalmente.

O conceito de um "serviço" é quebrado numa coleção de unidades funcionais, cada unidade representando alguma função essencial, como Fax, Impressão, Scan. Uma descrição de serviço é então uma coleção de descrições de unidades funcionais, cada uma possuindo uma coleção de registros de atributos (nome, valor). Esses registros podem ser consultados e comparados durante o processo de descoberta de serviço. Certas funções de comparação bem definidas podem ser associadas com uma consulta que procura por um serviço. A requisição de descoberta de serviço é enviada ao SLM local que poderá invocar outros SLMs, através de chamadas remotas de procedimentos – RPC (mais especificamente a RPC ONC da SUN). O Salutation define APIs para que clientes possam invocar aquelas operações e recuperar os resultados.

Comunicação no Salutation

Uma das funcionalidades importantes que o Salutation tenta prover é a independência de protocolo de transporte. A comunicação entre clientes e unidades funcionais (serviços) pode ser feita de diversas maneiras. No modo nativo, os clientes podem usar protocolos nativos e falar uns com os outros diretamente sem envolver os SLMs na transferência de dados. No modo emulado, os SLMs gerenciam a sessão e agem como um conduíte para os dados, entregando-os como mensagens. Isso proporciona algo muito interessante – a independência de protocolo de transporte. No modo salutation, os SLMs não carregam apenas os dados, mas também definem os formatos de dado a serem usados na transmissão. Nesse modo, padrões bem definidos de interoperação com unidades funcionais são seguidos, permitindo a interoperabilidade genérica. Toda comunicação nos últimos dois modos mencionados envolve

APIs e eventos bem definidos. Descrições de dados seguem uma notação (codificação) popular chamada ASN.1 *(Abstract Syntax Notation One)* da OSI. Ao invés de gerenciamento com contratos temporários, os SLMs podem ser consultados para checar periodicamente a disponibilidade de unidades funcionais e reportar o estado. Isso permite que um cliente ou unidade funcional (serviço) torne-se ciente quando qualquer um deles não estiver mais disponível. O Salutation já possui inúmeras unidades funcionais definidas para vários propósitos como fax e correio eletrônico por voz.

O Jini é independente de plataforma. O UPnP implica na padronização que nem todos os fabricantes podem estar dispostos a seguir. O Salutation mostra ser uma solução mais flexível devido a sua independência de protocolo e de linguagem de programação utilizada. Qual dos três deverá dominar o mercado é uma questão ainda não respondida pelo mercado. Até o momento, não há um líder incontestável.

7. Adaptação de Aplicações em Ambientes Ubíquos

A adaptação em ambientes de computação ubíqua é uma questão atual importante de pesquisa. Os ambientes de computação ubíqua são caracterizados pela grande diversidade de dispositivos, heterogeneidade de redes e conexão intermitente. As aplicações têm que se ajustar a mudanças nesses ambientes, de forma transparente para o usuário [4], [56].

Uma aplicação pode sofrer adaptação de dados e/ou de controle. A adaptação de dados implica numa transformação dos dados usados com freqüência pela aplicação para versões que se adaptam aos recursos disponíveis, sem a perda da representação, por exemplo, uma textura pode ser transformada para uma outra versão de textura com resolução menor. Já a adaptação de controle implica na modificação do fluxo de controle da aplicação, ou seja, no seu comportamento. Por exemplo, uma aplicação pode, a partir da notificação de aumento de atraso na rede, introduzir um buffer de recepção de dados, em tempo de execução, para a diminuição da latência [57].

A adaptação é orientada por uma *política de adaptação*. O tipo da aplicação, bem como o tipo dos dados da aplicação, muitas vezes dita a escolha da política de adaptação. Um modelo de adaptação deve levar a política de adaptação em consideração (quando adaptar, o que adaptar, quanto adaptar, considerações de custo etc), além de outras questões, tais como: detecção de conflito de interesses e como resolve-los, grau de intervenção do usuário na política de adaptação etc. [58].

7.1 Estratégias de adaptação

As estratégias de adaptação podem ser classificadas de três maneiras [59]:

1. *Transparente para a aplicação* – o sistema toma para si a responsabilidade integral da adaptação, ficando entre a aplicação e o dado. Esta abordagem provê compatibilidade retroativa com as aplicações existentes e não necessita de modificação na aplicação. Entretanto, as políticas de adaptação, neste caso, são dependentes do tipo dos dados e também do tipo da aplicação. Assim, para cada

tipo de aplicação uma nova política deve ser estabelecida, não sendo possível o uso de políticas genéricas.

2. *Laissez-faire* - independente do sistema, somente a aplicação é alterada. Nesta estratégia não há um gerenciador central de adaptação, somente a aplicação é alterada - o sistema não está ciente de adaptação. Isto torna mais difícil a escrita do código das aplicações, porem, a adaptação é feita sob medida resultando num melhor resultado final e permitindo ao usuário entrar com parâmetros.
3. *Ciente da aplicação* – promove uma sociedade entre a aplicação e o sistema para prover a adaptação. Esta estratégia envolve muitas vezes a existência de uma camada entre a aplicação e o sistema operacional, que é responsável pelo monitoramento e controle de recursos, bem como as decisões sobre a aplicação de políticas de adaptação.

As estratégias de adaptações *Laissez-faire* e *ciente de aplicação* permitem a adaptação tanto do controle quanto dos dados. Já a adaptação *transparente para a aplicação* é limitada apenas a dados.

7.2 Parâmetros de adaptação normalmente considerados

A adaptação de aplicações normalmente se baseia no monitoramento dos recursos de rede, de hardware e de software, e na conseqüente modificação de tipos de dados, tais como vídeo, imagens, áudio, texto [60]. Os recursos de rede que podem sofrer flutuação incluem: largura de banda, latência e taxa de erro na rede. Já os recursos de hardware dos dispositivos que podem variar incluem: tamanho e resolução da tela, quantidade de cores, memória, poder de processamento, carga da bateria etc. Os recursos de software que variam se restringem às capacidades do sistema operacional ou da aplicação para suportar um determinado tipo de dado (por exemplo, Post script ou extensões HTML não padronizadas) ou extensões de protocolo, como o IP multicast. Modificações no áudio, por exemplo, incluem alterações do número de canais (*Surround*, estéreo ou mono), redução da freqüência de amostragem do sinal sonoro e algoritmos de compressão. Em situações mais estritas, como no caso de mensagens de voz, um sintetizador é utilizado para converter áudio em texto.

Várias ações de adaptação podem ser aplicadas aos tipos de dados multimídia: a compressão pode ser usada tanto em quadros de vídeo quanto em quadros de imagens, resultando ou não em perda da informação. A redução da dimensão ou da resolução do vídeo ou da imagem pode ser feita durante o processo de compressão ou aplicada separadamente. Filtros podem ser aplicados a vídeos para descartar quadros semelhantes. Quanto aos textos, formatações e *aliasing* de fontes podem ser retirados para uma representação mais dinâmica na tela. Fontes muito complexas podem ser descartadas em prol do uso de outras mais simples. Em ambientes virtuais tridimensionais, os recursos monitorados são os mesmos, entretanto os parâmetros sujeitos a alterações são outros. A adaptação da aplicação normalmente está relacionada ao monitoramento dos recursos do sistema, da rede e das políticas do usuário e eventual adaptação da aplicação ao estado atual desses recursos. A adaptação de aplicações 3D normalmente envolve parâmetros do tipo: quantidade de quadros exibidos por segundo, qualidade da imagem, que é afetada por várias outros parâmetros, dentre eles, o número de polígonos exibidos num determinado momento, a quantidade de

fontes de luz e a qualidade de texturas (resolução, quantidade de cores). Estes últimos parâmetros dificilmente são monitorados em tempo real, pois o ônus disso é muito alto. Em vez disso, a adaptação da aplicação é tratada localmente, normalmente pelos motores de sintetização da cena, tendo como base as capacidades máximas do terminal. Já a adaptação de controle em ambientes virtuais tridimensionais, em tempo-real, é raramente discutida na literatura e é assunto de pesquisa [61]. A adaptação da aplicação (apresentação, dados e controle) é uma área bastante ativa de pesquisa dentro da computação ubíqua.

8. Sincronização

Para entender as questões relacionadas a sincronização, cinco situações são descritas a seguir [5] [20]:

1. *Situação 1*. Com a multiplicidade de dispositivos que podem ser usados pelos usuários e que oferecem um mesmo serviço, tal como agenda de nomes, dados gerados em um dispositivo devem ser replicados nos outros dispositivos daquele usuário, caso contrario, se o usuário quiser acessar a sua agenda através de um dispositivo diferente, verá uma versão diferente dos dados. Por exemplo, um usuário que adiciona um endereço em sua agenda no celular, poderá querer acessar a agenda a partir do PDA em um outro momento. Se a informação nova que estava no celular não foi replicada para a agenda do PDA, o usuário vai se frustrar pois não terá acesso aos últimos dados gerados.
2. *Situação 2*. Um usuário em movimento, por exemplo, dentro de um carro, pode estar acessando uma aplicação, cujos dados estão armazenados em um servidor remoto. Num determinado momento, a conexão pode ser interrompida, seja por falta de cobertura da rede sem fio, ou para economizar nos custos de conexão, o usuário deve ser capaz de continuar o trabalho localmente, por exemplo, gerando novos dados ou modificando dados que foram carregados previamente na memória local do dispositivo. Quando a conexão do dispositivo com o servidor for restabelecida, os dados alterados no dispositivo local devem ser replicados para o servidor, pois estes dados poderão ser acessados via outros dispositivos.
3. *Situação 3*. Ocorre devido as diferenças de software ou hardware dos diferentes dispositivos. Considere o seguinte exemplo [5]: um celular, com menos memória e sistema operacional diferente do PC, tem o potencial de armazenar uma forma reduzida do nome e numero de telefone de uma pessoa. Mais ainda, pode armazenar no máximo 300 registros, onde cada registro é identificado por um ID de um byte. Por outro lado, no PC, cada registro pode ser identificado por um ID de 16 bytes ou mais. Se os registros tiverem que ser atualizados nos dois equipamentos, como fazer a correspondência entre os identificadores?
4. *Situação 4*. Quando registros são alterados (bem como adicionados ou excluídos) na base de dados do servidor ou do dispositivo, as alterações efetuadas devem ser registradas, bem como o tempo em que foram realizadas (possivelmente através de estampilhas de tempo), para que o sincronizador possa saber quando os dados foram

alterados pela ultima vez e assim solicitar a lista dos registros alterados desde a ultima sincronização bem sucedida. Desta forma, as alterações poderão ser reproduzidas de forma integra nas replicas dos dados em outros dispositivos ou servidores.

5. *Situação 5*. Quando duas replicas exatas de dados, armazenadas em dois locais diferentes, forem acessadas e modificadas por dois usuários diferentes, ao mesmo tempo, como atualizar esses dados?

O objetivo (e também desafio) da sincronização é manter dois (ou mais) conjuntos de dados idênticos mesmo quando alterações são feitas em ambos os conjuntos, ao mesmo tempo.

Não se pretende discutir aqui todas as soluções possíveis de sincronização para as situações descritas acima, mas sim ilustrar a complexidade desta questão, até porque os casos descritos representam apenas algumas situações que envolvem sincronização, e certamente não abarcam todas as situações possíveis, e apresentar algumas soluções, e produtos, descritos na literatura.

A sincronização é tratada, através de protocolos de sincronização. Estes protocolos consistem, tipicamente, das seguintes fases [20]

1. **Fase de pré-sincronização** – Nesta fase são realizadas: a *autenticação* do servidor e do dispositivo, que assegura que o servidor é quem ele diz que é e o dispositivo idem; a *autorização* que verifica se o dispositivo está autorizado a executar a alteração solicitada; e a *determinação das capabilidades* do dispositivo para que o servidor possa otimizar o fluxo de dados a ser enviado para o dispositivo.
2. **Fase de sincronização** – Nesta fase os dados de sincronização são trocados entre dois parceiros de sincronização. Cada parceiro tem uma tabela de mapeamento de IDs locais para IDs globais. Todos os IDs locais das entradas de dados são mapeados em IDs globais e, na sincronização, apenas as entradas novas, alteradas ou excluídas são trocadas entre os dois parceiros. Haverá conflito quando ambos atualizaram a mesma entrada. Este conflito pode ser resolvido de varias maneiras: tentativa de fusão das entradas, duplicação das entradas, prevalecimento de uma entrada sobre a outra, notificação do usuário parta que ele decida o que fazer.
3. **Fase de pos-sincronização** – Nesta fase, tarefas como a atualização de tabelas de mapeamento, reportagem de conflitos não resolvidos etc., são realizadas.

Algumas das soluções para tratar a sincronização incluem: o Protocolo de sincronização SyncML[62], IrMC, Mobile Application Link [63]. Produtos existentes incluem: AvantGo [64], DB2 [65], etc.

9. Desafios da Computação Ubíqua

Há vários desafios a serem superados na computação ubíqua nos níveis tecnológico, social e organizacional. No nível social, pesquisadores afirmam que a computação ubíqua trará problemas de segurança e privacidade e mudará a forma como os trabalhadores e empresas

interagem entre si, o que gera novas preocupações, por exemplo, como o empregador supervisiona os empregados, já que computadores estarão por toda parte conectados uns aos outros por redes. Igualmente desafiador é entender como os avanços tecnológicos podem ajudar ou prejudicar o ser humano e o bem-estar social. Por exemplo, a agregação de dados capturados por sensores pode resultar numa maior comunicação social; mas isso também pode permitir a organizações discernir mais e mais sobre os padrões de atividades e preferências pessoais das pessoas, e dessa forma levar a invasão de privacidade e por em risco o nível de confiança nas relações sociais.

Um resumo dos principais desafios da computação ubíqua no nível tecnológico é dado a seguir:

- Novas Arquiteturas - Projeto e a implementação de arquiteturas computacionais que possibilitem a configuração dinâmica de serviços ubíquos em larga escala;
- Tratamento de Contexto - um dos grandes desafios no tratamento do contexto é a coleta dos dados de diversos sensores e o processamento desses dados em informações de contexto e a disseminação dessas informações para centenas de aplicações executadas em diversos dispositivos.
- Integração da mobilidade em larga escala com a funcionalidade da computação pervasiva
- Integração das redes sem fio de forma transparente para o usuário
- Segurança nas redes sem fio e nos sistemas ubíquos
- Tratamento da multiplicidade de dispositivos
  - Melhor utilização dos recursos dos dispositivos pessoais - armazenar e gerenciar informações a partir do dispositivo
  - Realização da tarefa apropriada no dispositivo apropriado - aplicações em diferentes dispositivos que exploram as características únicas de cada um deles
- Criação de metodologias de desenvolvimento de aplicações em que a aplicação se move juntamente com o usuário
- Etc.

Referências

1. Weiser, M. (1991), “The Computer for the 21st Century”, Scientific American, vol.265, no.3,Setembro., pp.94-104.
2. Banavar, G. E Bernstein, A. (2002) “Software Infrastructure and Design Challenges for Ubiquitous Computing Applications”. Communications of the ACM, vol.45, no.12, Dezembro., pp.92-96.
3. Eisenstein, J., Vanderdonckt, J. e Puerta, A. (2000) “Adapting to Mobile Contexts with User-Interface Modelling”. IEEE Computer.

4. Lyytinen, K. e Yoo, Y. (2002) "Issues and Challenges in Ubiquitous Computing", Communications of the ACM, vol.45, no. 12, Dezembro.
5. Hansmann, U., Merk, L., Nicklous, M.S., Stober, T. (2001) "Pervasive Computing Handbook", Ed. Springer. 409 pags.
6. Schilit, B. N., Adams, N., Gold, R., Tso, M., and Want, R.(1993) *"The ParcTab mobilecomputing system"*. Proceedings of the Workshop onWorkstation Operating Systems, p.34–39.
7. Abowd, G.D.(1999) "Classroom 2000: An experiment with the instrumentation of a living educational environment". *IBM Systems Journal*, v. 38, Outubro.
8. Mukhopadhyay, S., Smith, B. (1999) "Passive Capture and Structuring of Lectures". *Proceedings of ACM Multimedia*.
9. Hürst, W., Müller, R. (1999) "A Synchronization Model for Recorded Presentations and its Relevance for Information Retrieval". *Proceedings of ACM Multimedia*.
10. Brotherton, J.A., Abowd, G.D., Truong, K.N. (1998) "Supporting Capture and Access Interfaces for Informal and Opportunistic Meetings". *Georgia Institute of Technology Technical Report: GIT-GVU-99-06*, Atlanta, GA.
11. Richter, H., at el. (2001) "Integrating Metting Capture within a Collaborative Team Environment". *Proceedings of the International Conference on Ubiquitous Computing (UbiComp-2001)*, Atlanta, GA.
12. Johanson, B., Fox, A., Winograd, T. (2002) "The Interactive Workspaces Project: Experiences with Ubiquitous Computing Rooms". *Pervasive Computing Magazine Special Issue on Systems.*
13. Burrell, J., Gay, G. K., Kubo, K., Farina, N.(2002) "Context-aware computing: A test case". *Proceedings of the International Conference on Ubiquitous Computing*, p.1–15.
14. Fleck, M., Frid, M., Kindberg, T., O'Brien-Strain, E., Rajani, R., Spasojevic, M. (2002) "From informing to remembering: Ubiquitous systems in interactive museums". *IEEE Pervasive Computing*, p.13–21.
15. Kindberg, T., Barton, J. [2001] "A Web-based Nomadic Computing System". *Computer Networks: The International Journal of Computer and Telecommunications Networking*, v.35, nº 4, p. 443–456, Março.
16. AIM (2003). Radio Frequency Identification (RFID) home page. Disponível em http://www.aimglobal.org/technologies/rfid/. *The Association for Automatic Identification and Data Capture Technologies*.
17. Brumitt, B., Meyers, B., Krumm, J., Kern, A., Shafer, S. A. (2000) "Easyliving: Technologies for Intelligent Environments". *Proceedings of the Handheld and Ubiquitous Computing Second International Symposium (HUC'2000)*, p.12-29.

18.Mozer, M. C. (1998) “The Neural Network House: An Environment that Adapts to its Inhabitants”. *In American Association for Artifcial Intelligence Spring Symposium on Intelligent Environments*, p.110-114.

19.Herrtwich, R. G. (2002) “Ubiquitous Computing in the Automotive Domain”. *Proceedings of the Pervasive Computing – First International Conference*, p. 15, Agosto

20.Burkhard, T J., Henn, H., Hepper, S., Rindtorff, K., Schack, T. (2002) “Pervasive Computing: Technology and Architecture of Mobile Internet Applications”. Ed. Addison-Wesley. 410 pags.

21.International Organization for Standards (1999). Em www.iso.ch. Consultado em 30 de Março de 2003.

22.PC/SC Workgroup (1999) “PC/SC Specification 1.0”. Em www.pcscworkgroup.com

23.OCF – Open Card Framework. Em www.opencard.org

24.Sistema de Entrada T9. Em www.t9.com/demo_page2.html

25.Sistema de entrada Octave da e-acute.Em www.e-acute.fr

26.Como aplicar a tecnologia Java para dispositivos pequenos: www.javaworld.com.

27.Documentos sobre Java: http://java.sun.com/infodocs

28.Tutoriais sobre Java: http://java.sun.com/docs/books/tutorial

29.Sobre o J2ME em. http://www.java.sun.com/products/j2me

30.Ferramentas para programação de dispositivos: http://www.chicagowireless.org/ tools.html

31.Forum Java Card: www.javacardforum.org

32.MVC em: http://java.sun.com/blueprints/guidelines/designing_enterprise_applications_2e/app-arch/app-arch2.html

33.Tanenbaum, A. S. (2001) “Modern Operating Systems”. Second Edition. Prentice Hall, 951 pags.

34.Salber, D., Dey, A.K., Abowd, G.D. (1999) “The Context Toolkit: Aiding the Development of Context-Enabled Applications”. *CHI’99*, ACM Press, p.434-441, Pittsburgh, PA, Maio.

35.Román, M., Hess, C., Cerqueira, R., Ranganathan, A., Campbell, R.H., Nárstedt K. (2002) “A middleware infrastructure for Active Spaces”. *IEEE Pervasive Computer*, v.1, n.4, p. 74-83, Outubro-Dezembro.

36.Hess, C.K., Roman, M., Campbell, R.H., (2002) “Building Applications for Ubiquitous Computing Environments”. *Proceedings of the Pervasive Computing – First International Conference*, p. 16-29, Agosto.

37.Johanson, B., Fox, A., Winograd, T. (2002) “The Interactive Workspaces Project: Experiences with Ubiquitous Computing Rooms”. *Pervasive Computing Magazine Special Issue on Systems.*

38.Ponnekanti, S., Johanson, B. (2003) “Portability, Extensibility and Robustness in iROS”. *Proceedings of the IEEE International Conference on Pervasive Computing and Communications*. Dallas-Fort Worth, Texas, Março.

39.Garlan, D. Et al.(2002) “Project Aura: Toward Distraction-Free Pervasive Computing”. *IEEE Pervasive Computing*, p.22-31, Abril-Junho.

40.Freire, L. (2002) “Um Ambiente Integrado de Ensino e Avaliação de Segurança em Sistemas da Web”, Tese de Mestrado defendida em Agosto de 2002 no PPG-CC do DC UFSCar.

41.Waldman, H. e Yacoub, M. D.(1997) "Telecomunicações - Princípios e Tendências", Serie Universidade, Editora Erica, 287 págs.

42.Christensen, G., Florack, P. G. e Duncan, R. (2001) "Wireless Intelligente Networking". Mobile Communications Series, Artech House, 418 pags.

43.Harte, L., Levine, R. e Kikta, R.(2002) "3G Wireless Demystified". McGraw-Hill, 496 pags.

44.Dornan, A. (2001) "Wireless Communication: o guia essencial de comunicação sem fio". Editora Campus, Rio de Janeiro.

45.GSM World (2002) "The Wireless Revolution" em http://www.gsmworld.com. Consultado em 20/12/ 2002.

46.Massaud, E. M.(2000) "Estudo de técnicas de alocação dinâmica de recursos e sincronismo para serviços de multimídia num sistema móvel celular CDMA de banda larga". Dissertação de Mestrado. Departamento de Engenharia de Telecomunicações e Controle. Escola Politécnica, USP, São Paulo.

47.UMTS (2001) "Universal Mobile Telecommunications System (UMTS)". Web Proforum Tutorials.Em http://www.iec.org

48.Underhill, W. E Dickey, C. (2001) "Lost in a mobile maze". Newsweek. pp. 17. May.

49.Foroohar, R. (2001) "The Other Bubble". Newsweek. pp.13-16. May.

50.Miller, M. (2001) "Descobrindo o Bluetooth", Editora Campus, 289 pags.

51.MNCSDS (1999) "Mobile Network Computer Service Discovery Specification, version 1.1", Março. Em ttp://www.oadg.or.jp/activity/mncrs/servdisc/spec/ serv_disc.html.

52.Jini. Em http://www.jini.org/

53.UPnP. Em http://www.upnp.org/

54.Salutation. Em http://www.salutation.org/

55.Rekesh, J. (1999), “UPnP, Jini and Salutation - A look at some popular coordination frameworks for future networked devices”. Publication from California Software Labs.

56.Raatikainen, K. (2002) “Middleware for Mobile Applications Beyond 3G”, Nokia Research Center and University of Helsinki. Disponível em: www.cs.helsinki.fi/u /kraatika/Papers/MobileMiddleware.pdf.

57.Lara, E., Wallach D. S. and Zwaenepoel, W. (2001) "Puppeteer: Component-based Adaptation for Mobile Computing," Proceedings of the 3rd USENIX Symposium on Internet Technologies and Systems, San Francisco, California, in http://www.cs.rice.edu/~delara/papers.

58.Rao, D. (2001) “Efficient and Portable Middleware for application-level Adaptation”, Master of Science thesis, Department of Computer Science and Applications Virginia Tech, Blacksburg, Virginia.

59.Noble, B. (1998) “Mobile Data Access”, doctoral thesis, School of Computer Science, Carnegie Mellon university, May, CMU-CS-98-118.

60.Fox, S., Gribble, E., Brewer, A. and Elan, A. (1997) “Adapting to Network and Client Variability via On-Demand Dynamic Distillation” University of California at Berkeley, July.

61.Silva, A. R..; Laffranchi, M., Araujo, R. B.(2003). "A Adaptação de Jogos 3D Multiusuário a Ambientes Ubíquos", submetido ao Semish´2003.

62.SyncML. Em www.syncml.org.

63.MAL - *Mobile Application Link*. Em www.mobilelink.org.

64.AvantGo. Em www.avantgo.com/frontdoor/index.html

65.DB2 - *IBM DB2 Everywhere*. Em www.ibm.com/software/data/db2/everyplace

66.Weatherall, J. and Jones, A. (2002) “Ubiquitous Networks and their Applications”. IEEE Wireless Communications, Fevereiro.

67.X10 Technology. Em: www.extremetech.com/article2/0,3973,133763,00.asp