UNIVERSIDADE FEDERAL DE CAMPINA GRANDE
CENTRO DE ENGENHARIA ELÉTRICA E INFORMÁTICA
COORDENAÇÃO DE ESTÁGIOS E PROJETOS DE ENGENHARIA ELÉTRICA

Trabalho de Conclusão de Curso

TRANSFERÊNCIA DE DADOS ENTRE DISPOSITIVOS VIA BLUETOOTH

Relatório de Trabalho de conclusão de curso apresentado à coordenação de Engenharia elétrica da UFCG, como parte dos requisitos de obtenção do título de Engenheiro Eletricista.

Aluno: Nilo Sérgio de Aquino Arruda
Matícula: 29821131

Campina Grande
Março de 2006

Biblioteca Setorial do CDSA. Fevereiro de 2021.

Sumé - PB

Agradecimentos

Agradeço aos professores e funcionários do curso de engenharia elétrica da Universidade Federal de Campina Grande, por todo conhecimento repassado durante o curso. Em especial ao professor Luiz Reis, que me orientou no desenvolvimento deste trabalho com bastante atenção.

Aos meus familiares, que estiveram do meu lado o tempo todo e sempre me apoiaram e incentivaram durante toda a minha formação acadêmica.

Aos meus amigos e colegas, por todo tempo e ajuda oferecida, em especial a Eisenhawer Fernandes e Vagner Vale, pois estivemos juntos durante toda a graduação compartilhando conhecimentos e dificuldades, e à Alcides Flach, pelo computador e consultoria dos conceitos chaves sobre desenvolvimento para aplicações móveis utilizando a tecnologia J2ME.

Resumo

Esse trabalho apresenta uma implementação para transferência de arquivos através da rede *bluetooth*, utilizando-se a plataforma Java para micro edição (J2ME). Primeiramente, será apresentado o se esquema de funcionamento e a maneira que ele é adaptado ao trafego multimídia, bem como, um estudo sobre a tecnologia J2ME. Finalmente, teremos uma implementação da união dessas duas tecnologias, e uma pequena demonstração do universo de aplicações nas quais pode ser utilizado.

A tecnologia *bluetooth* tem a grande vantagem de ser gratuita e de tornar a conectividade entre dispositivos eletrônicos simples e prática, por outro lado, essa tecnologia visa a substituição de cabos pequenos da ordem de metros, por esse motivo, esse tipo de conexão é de curto alcance, com limite padrão da ordem de metros.

A tecnologia J2ME tem sua maior vantagem a modularidade que torna uma ferramenta mais segura e fácil de utilizar do ponto de vista do desenvolvedor, bem como sua portabilidade, pois é possível executar tais aplicações em qualquer dispositivo, desde que os mesmo tenham a máquina virtual JAVA instalada. Mas para isso, é pago um preço alto e aplicações desenvolvidas na plataforma JAVA são pesadas e apresentam baixo desempenho, exigindo bastante memória e do processador, tornando os dispositivos caros.

Índice

Lista de Figuras

1 Introdução

Com a crescente necessidade de conexão entre os diversos dispositivos existentes na vida moderna que são lançados constantemente, surgiu a necessidade de reduzir as comunicações entre tais dispositivos através de meios fixos. Essa nova necessidade está proporcionando ao mercado, um crescimento constante das tecnologias sem fio (*Wireless*) [3]. Não é difícil imaginar diversos equipamentos eletrônicos de alta tecnologia se comunicando de forma remota sem a necessidade de fios. Por exemplo, um Computador Pessoal (PC) poderia se comunicar com um telefone celular para enviar informações referentes à Bolsa de Valores, estado do carro, lojas podem enviar mensagens *broadcast* sobre promoções de seus produtos para telefones celulares de clientes em potencial que estejam passando pelas proximidades da loja, dentre outras.

O *bluetooth* é um sistema de comunicação barato e eficaz que surgiu inicialmente com intuito de substituir o cabeamento necessário para interconexão de tais dispositivos. Entretanto, pode ser usado por dispositivos eletrônicos em geral, para se comunicarem a uma pequena distância, e atualmente está presente em diversos equipamentos, tais como aparelhos de televisão, computadores, aparelhos celulares, entre outros [1], [2].

Nesse contexto, existe a necessidade de desenvolvimento de aplicações para esses equipamentos, utilizando-se dos recursos que os mesmos suportam. Entre esses recursos há o ambiente Java, equipamentos que rodam Java estão cada vez mais presentes no dia a dia das pessoas, sejam produtos fixos ou móveis, de uso pessoal ou de consumo, e que apresentam freqüentemente, entre outras características, alguma forma de comunicação e um elevado grau de sofisticação em relação às tarefas que executam (dispositivos inteligentes) [4], [5]. Geralmente são microprocessadores embutidos em algum dispositivo, como em telefones celulares e eletrodomésticos [5].

O desenvolvimento de aplicações Java para celulares é possível utilizando-se o ambiente Java 2 *Micro Edition* (J2ME), em português JAVA para micro edição. Basicamente uma versão compacta da linguagem padronizada da Sun,

destinado ao desenvolvimento de aplicações para dispositivos móveis [7]. A plataforma J2ME oferece para tanto, uma máquina virtual Java, chamada de KVM (*KiloByte Virtual Mashine*), esse tem uma conotação metafórica e significa que a máquina é tão compacta que pode funcionar com apenas 1 KB de memória, ou seja, pequena o bastante para ser suportada dentro das restrições de memória destes dispositivos. Essa máquina, por sua vez, não suporta todas as classes e funcionalidades do Java. O J2ME é formado basicamente por duas outras especificações, como será visto nesse trabalho.

Esse trabalho apresenta uma simulação de comunicação entre aparelhos de comunicação móveis que suportam a tecnologia bluetooth, e possuem uma máquina virtual Java (KVM). Será apresentado uma abordagem teórica sobre a tecnologia *bluetooth* e sobre a plataforma Java 2, mais especificamente J2ME, que foi utilizada para o desenvolvimento desse trabalho.

Finalmente, será apresentado o funcionamento da aplicação, que propõe a comunicação entre dois aparelhos celulares da motorola para *download* de arquivos, através do ambiente de simulação WTK (wirelless tookit). Essa aplicação é interessante porque permite a conexão gratuita entre dispositivos quaisquer, dessa forma, pode-se fazer transferências de dados entre os mesmos com alta taxas de transmissão, desde que estejam a uma distância limitada.

2 Objetivos específicos

- ✓ Realizar um estudo sobre as tecnologias bluetooth, considerando-se sua aplicação para transferência de arquivos;
- ✓ Realizar um estudo sobre a tecnologia J2ME, considerando-se essencialmente sua API dedicada a *Bluetooth*;
- ✓ Desenvolver um sistema de transferência de arquivo gratuito que tem como único objetivo a substituição de cabos curtos;
- ✓ Após a implementação do sistema, realizar testes utilizando o Wireless toolkit da motorola, para validar a aplicação.

3 BlueTooth

O termo *Bluetooth* se refere a uma tecnologia que permite a comunicação de voz e dados através de um enlace de rádio de baixo alcance. Esta tecnologia foi, inicialmente, desenvolvida para a substituição dos cabos associados a dispositivos periféricos.

A especificação do *Bluetooth* [2] define um sistema completo a partir da camada de enlace até a camada de aplicação, sendo a pilha de protocolos geralmente implementada parcialmente em hardware e software. O seu núcleo é formado pela banda base *(Baseband*), o Protocolo gerenciador de conexão (*Link Ma-nager Protocol* – LMP), o Controle lógico da conexão e adaptação do protocolo (*Logical Link Control and Adaptation Protocol* - L2CAP) e o protocolo de busca de serviço (*Service Discovery Protocol* - SDP). A *Baseband* e o *Link Controller* (Controlador de conexão) controlam o enlace de rádio-freqüência (RF) entre as unidades *Bluetooth*, permitindo a formação de mini-redes de comunicação conhecidas como *piconets*, explicada detalhadamente na página 8. O LMP é responsável pela configuração do enlace estabelecido entre os dispositivos *Bluetooth* como os modos de economia de energia e os estados operacionais dos dispositivos dentro de uma mini-rede. O L2CAP é responsável pela adaptação dos dados provenientes das camadas superiores da pilha de protocolos de forma que possam ser manipulados pelas camadas inferiores. A Figura 1 mostra como essas camadas estão relacionadas.

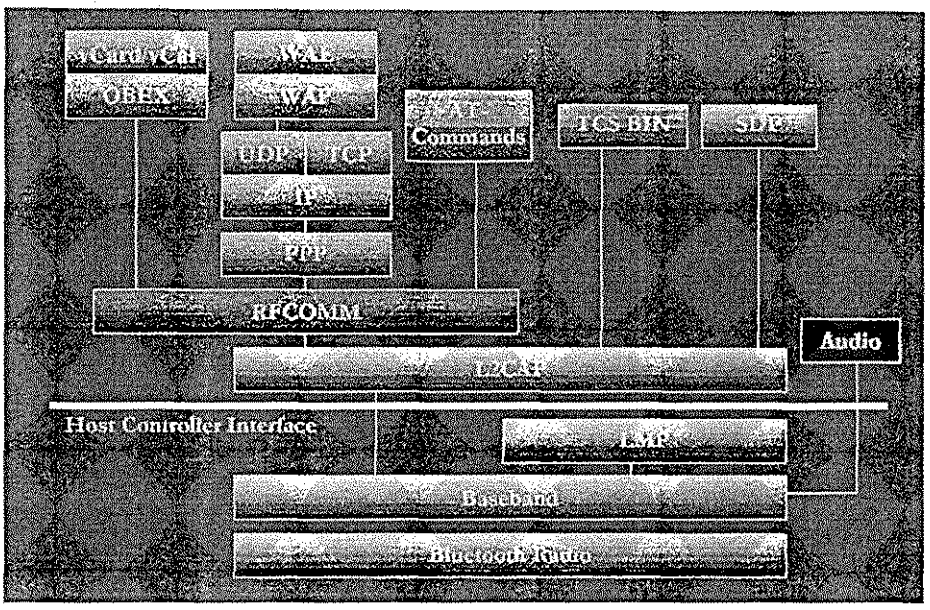

Figura 1: A Pilha de Protocolos do Bluetooth

Os dispositivos *Bluetooth* operam na faixa de freqüência de 2.4 GHz. Esta banda é reservada para uso geral em aplicações industriais, científicas e médicas, e por essa razão recebeu o nome de ISM *(Industrial,Scientific and Medica*l). Como esta faixa é utilizada por muitos tipos de aplicações é comum ocorrerem interferências durante as transmissões. Para evitar estas interferências é utilizada uma técnica de salto de freqüência *(Frequency Hopping* - FH). No *Bluetooth,* a taxa de saltos de freqüência é de 1600 *hops*/seg no modo conectado e 3200 *hops*/seg no modo de procura /sincronização. Este esquema completo é chamado de *Frequency Hoppping Code Division Multiple Access,* FHCDMA, seria uma combinação do FHC com o CDMA (múltiplo acesso por divisão de código). Nos sistemas *Bluetooth*, a banda ISM é dividida em 73 "canais de salto". Cada um desses canais é usado por 625 ms (um *slot*), ocorrendo posteriormente um salto para outro canal que é escolhido aleatoriamente. Neste novo canal, a transmissão se dará por outros 625 ms, ocorrendo novamente um salto para outra freqüência, processo este que se repete até que a transmissão seja encerrada, conforme ilustrado na figura 2.

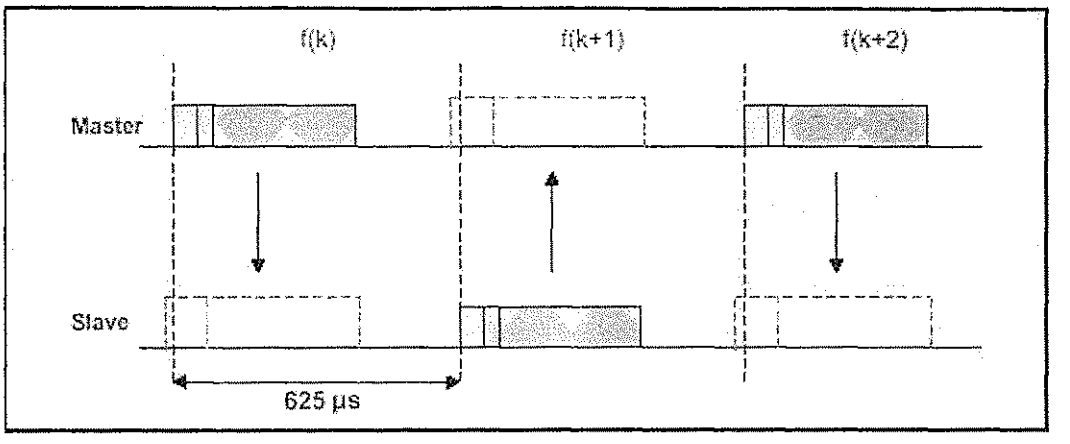

Figura 2: Multiplexação no Tempo

Está técnica é conhecida como *time-division duplex*, TDD. Desta forma, a técnica de saltos de freqüência espalha o tráfego *Bluetooth* sobre toda a banda disponível, proporcionando ao sistema uma boa proteção contra interferências.

3.1 Características de Protocolos da Camada de Enlace

O *Bluetooth* é um protocolo de comunicação da camada de enlace. No modelo OSI da ISO, ele se encontra no nível 2, onde são feitos os controles de acesso ao meio físico [1], [15]. O funcionamento de protocolos da camada de acesso ao meio em redes sem fio deve considerar alguns fatores que as diferenciam das redes fixas. Abaixo segue uma breve descrição destes:

- *Operação em Half-duplex*: quando um nodo transmite os dados, uma grande fração da energia do sinal interfere com o sistema de recepção, efeito denominada auto-interferência. Isto ocorre porque os níveis de energia para transmissão e recepção diferem em diversas ordens de magnitude. A transmissão do sinal é tipicamente muito maior do que o sinal recebido, impossibilitando efetuar a recepção do sinal no momento da transmissão. Dessa forma fica impossível a aplicação do Full Duplex, caso a freqüência se recepção e envio seja a mesma;

- *Canal variante no tempo*: os sinais de rádio propagam de acordo com três mecanismos: reflexão, difração e dispersão. O sinal recebido pelo nodo é a superposição de diferentes e atenuados composições do sinal transmitido. Como resultado, a potência do sinal recebido varia em função do tempo. Este fenômeno é chamado de propagação *multipath* (multi-caminho, ou seja, o sinal tem vários caminhos alternativos). Para verificar o canal normalmente é usada uma técnica de *handshaking* (A tradução literal seria "aperto de mão", que significa uma negociação de teste entre dois pontos de uma conexão) que verifica a qualidade do link. Quando dois nodos querem comunicar entre si, eles trocam pequenas mensagens para testar o canal, e em caso de sucesso, o processo normal de comunicação ocorre;
- *Erros de rajada:* Como conseqüência da variação do canal no tempo e da variação da energia do sinal, os erros surgem em rajadas devido ao distanciamento dos nodos, uma vez que a qualidade do canal entre eles decai. A perda de pacotes devido a rajadas de erros pode ser minimizada através do uso das seguintes técnicas:
  - ✓ *Pacotes pequenos*
  - ✓ *Códigos FEC para correção de erros*
  - ✓ *Métodos de retransmissão*

  Este *último* método é o mais usado, através de pacotes de *acknowledgments* (checagem) usados para detectar os erros;
- Detecção do canal: A sensibilidade do canal de transmissão é uma função da posição do nodo em relação ao transmissor. Essa característica acarreta as seguintes dificuldades no momento da transmissão:
  - ✓ Nodos escondidos: É um nodo que se encontra mais próximo do nodo de destino, mas fora do alcance do nodo de origem. Dessa forma esse terceiro elemento não percebe a transmissão que está ocorrendo e começa a transmitir, *interferindo no processo;*
  - ✓ *Nodos expostos*: Neste caso o nodo se encontra próximo do nodo transmissor, mas fora do alcance do destino. Assim o terceiro elemento detecta o inicio da transmissão, mas não efetua a sua transmissão,

apesar de não atrapalhar na recepção do primeiro. Dessa forma a banda fica subtilizada;

- ✓ *Captura*: A captura ocorre quando um receptor recebe claramente a transmissão simultânea de dois nodos, e a captura do sinal ocorre no que possuir maior potencia. Os protocolos de camada MAC devem ser ajustados para controlar essa percepção do sinal.

3.2 Métodos de acesso do protocolo MAC

Os protocolos da camada MAC para redes sem fio foram desenvolvidos visando diferentes tipos de aplicações e dispositivos. Desta forma, cada protocolo possui características próprias que levam em consideração diversos fatores das redes sem fio de maneira diferenciada [1], a seguir serão apresentados os tipos de acesso necessários para satisfazer tais fatores:

- *Acesso aleatório*: neste caso os nodos fazem a contenção na hora de efetuar o acesso ao meio. Quando apenas um nodo tenta efetuar a transmissão, ela ocorre com sucesso. Mas no momento que mais de um tenta acessar o meio, ocorre uma colisão e os protocolos de acesso aleatório tentam resolver este caso com um conjunto de regras de utilização ordenada do meio;
- *Acesso garantido*: neste caso os nodos acessam o meio de maneira ordenada, usualmente em *round robin*, o qual funciona executando um roteamento de endereços pelo método das filas, "atende" ao IP que está na primeira posição da fila e o coloca em último, atendendo ao próximo e assim por diante [15], [16]. Existem duas maneiras de efetuar essa transmissão. A primeira é através da configuração mestre-escravo, onde o mestre nomeia os nodos que irão transmitir, e esses retornam os dados necessários. A outra maneira é através de *tokens*, topologia em anel, cujas estações devem aguardar sua vez para transmitir e somente a estação que possui o *token* transmite;

- *Acesso híbrido*: neste caso é feita uma mistura das duas técnicas. Um nodo efetua um processo de requisição para transmissão de dados. Este processo é feito de maneira aleatória. Em seguida, o outro analisa as capacidades necessárias vindas na requisição, retornando uma confirmação para iniciar a transmissão de dados. Neste momento, todos os recursos necessários para essa transmissão já foram reservados pelo transmissor, sendo liberados de acordo com padrões pré-determinados de Qualidade de Serviço ou mesmo de maneira aleatória.

O *Bluetooth* pode ser classificado de duas maneiras: durante o processo de conexão efetua o acesso aleatório, e quando esta é estabelecida efetua o acesso garantido, usando a política de mestre e escravo. Para avaliar o perfil de funcionamento de um protocolo de enlace de acordo com seu objetivo é necessário estabelecer algumas métricas. No caso do suporte a trafego multimídia, é necessário levar em conta:

- Atraso: O atraso é definido como o tempo médio dispendido por um pacote na fila da camada MAC, a partir do momento entra nela até a transmissão completa.O suporte ao trafego multimídia em tempo real deve minimizar este atraso;
- *Throughput:* Usado para medir a fração do canal usada para transmissão de dados. O objetivo do protocolo da camada MAC é maximizar a taxa de transferência enquanto minimiza o atraso do acesso, produzindo um valor de throughput aceitável para as aplicações. Se o tamanho médio de uma mensagem é de P bits, o tempo de transmissão de cada pacote é T segundos e C b/s é a capacidade do canal, o throughput é dado por P/TC;
- *Fairness:* O protocolo MAC deve apresentar condições de preferência de escolha de canal no momento que mais de um nodo decide transmitir sobre o canal, caso contrário ocorre o mau uso da banda passante. No caso do tráfego multimídia é suportado, ele deve apresentar uma política diferenciada de acesso;

- *Estabilidade*: Devido ao *overhead* no protocolo, o sistema deve ser habilitado para controlar cargas menores que a capacidade de transmissão do canal, minimizando o desperdício de banda passante;
- *Robustez*: O meio de comunicação sem fio é muito tendencioso a erros, e o sinal de transmissão é muito variante, de forma que um protocolo da camada MAC deve ter políticas de tratamento destes comportamentos instáveis;
- *Consumo de energia: Como a maioria dos dispositivos sem fio possui uma bateria limitada, estes protocolos têm o suporte a diferentes métodos de economia de energia, sendo que eles podem influenciar nas características do tráfego gerado;*

Assim, para o suporte a multimídia são criados mecanismos que tratam os pacotes de várias aplicações, a partir de restrições associadas ao atraso de transmissão. Duas políticas comuns nestes casos são as prioridades de acesso e o escalonamento. As prioridades de acesso provêem um serviço diferenciado permitindo que certos nodos tenham acesso aos serviços de rede com maior probabilidade do que outros. O escalonamento efetua o controle do acesso ao meio de maneira ordenada, alocando os recursos preferencialmente para os pacotes multimídia.

3.3 A topologia scatternet

O sistema *Bluetooth* provê conexões ponto-a-ponto (apenas dois dispositivos *bluetooth* envolvidos), ou conexões ponto-multiponto. Nas conexões ponto-multiponto, o canal é compartilhado entre alguns dispositivos, formando uma **piconet** (rede muito pequena). Em uma piconet, um dos dispositivos funciona como *master* (mestre), enquanto os demais funcionam como *slaves* (escravos). O master controla o acesso dos dispositivos slaves, determina o clock responsável pela sincronização, dentre outras funções. Múltiplas piconets com áreas sobrepostas formam uma **scatternet.** A figura 3 ilustra alguns três possíveis

configurações de topologias, 3.a) com uma piconet com um único escravo, 3.b) uma piconet com múltiplos escravos e uma 3.c) scatternet.

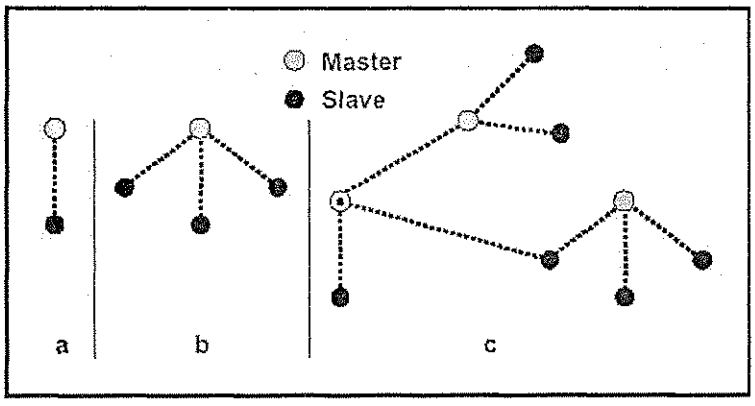

Figura 3: Topologias de Redes Bluetooth: a) Piconet com um único escravo; b) Piconet com múltiplos escravos; c) Possível configuração de uma scatternet.

Como o *Bluetooth* mescla o TDD com o FHCDMA, a comunicação *interpiconet* trás diversas peculiaridades como a necessidade de sincronização e descobrimento entre as *piconets* vizinhas. A comunicação *inter-piconets* é feita utilizando os modos de economia de energia, onde os elementos que pertencem a mais de uma *piconet* negociam os intervalos de conexão ativa entre as *piconets*. A formação das *scatternet* pode ser caracterizada de duas formas: como um evento controlado, onde uma aplicação, utilizando funcionalidades de diversas *piconets* conhecidas, cria uma *scatternet* para acessá-las; ou de maneira ad hoc, que caracterizam-se por terem topologia que varia com a mobilidade das estações e com as condições do meio de propagação [15], onde algum tipo de serviço é requerido e não se encontra em nenhuma *piconet* conhecida. O último caso requer duas funcionalidades: o roteamento, onde os nodos possuem a capacidade de encaminhamento de pacotes entre diferentes *piconets* e a descoberta de serviços, onde é necessário verificar as capacidades e funcionalidades dos nodos comunicáveis entre as diferentes *piconets*. Um dos pontos principais da formação das *scatternets* são as topologias, uma vez que os dispositivos passam por

procedimentos de descoberta e formação de links ponto-a-ponto de maneira explícita.

Em paralelo ao *Bluetooth*, o grupo de estudos do IEEE, o 802.15, estuda a criação de uma padronização para as *Wireless Personal Area Networks,* WPANs. Alguns cenários estão sendo avaliados, como a interoperação com o padrão 802.11, redes de grande capacidade de transmissão para trafego multimídia e redes de baixa capacidade energética, como as redes de sensores.

3.4 **Tipos de Enlace**

Dois tipos de enlace foram definidos pela especificação *Bluetooth*: assíncrono sem conexão assíncrona (*Asynchronous Connectionless* - ACL) e síncrono orientado à conexão (*Synchronous Connection Oriented* - SCO). Diferentes pares de mestre-escravos na mesma *piconet* podem utilizar diferentes tipos de enlace, sendo que o tipo de enlace adotado pode ser alterado durante uma sessão.

Os enlaces ACL foram definidos para transmissão de dados em rajadas como, por exemplo, tráfego de pacotes. Eles suportam conexões simétricas e assimétricas, comutadas por pacotes e ponto-a-multiponto, sendo possível o envio de mensagens de *broadcast*. Pacotes *multi-slot* utilizam o enlace do tipo ACL e podem alcançar uma taxa de transmissão de 723.2 kbit/s em uma direção e 57,6 kbit/s na direção oposta se nenhuma correção de erro for utilizada.

Apenas um único enlace ACL pode existir entre dois dispositivos durante uma transmissão, embora um mestre possa ter inúmeros enlaces ACL não ativos com diferentes escravos. Sendo assim, o mestre nem sempre transmite informações aos escravos de uma maneira regular. A escolha de qual escravo receberá ou enviará a informação é de responsabilidade do mestre. Um escravo só tem o direito de enviar um pacote ACL após receber previamente um pacote do mestre endereçado a ele. Além disso, a unidade mestra controla a largura de banda do enlace ACL, assim como a simetria do tráfego, e decide quanto da largura de banda total cada escravo da *piconet* poderá utilizar.

Os enlaces SCO suportam conexões simétricas, comutadas por circuito e ponto-a-ponto sendo, portanto, utilizadas tipicamente para o tráfego de voz. Eles provêm uma largura de banda reservada para o canal de transmissão e troca de dados com periodicidade regular sob a forma de *slots* reservados.

Um mestre pode suportar até três enlaces SCO para um mesmo escravo ou para escravos diferentes, enquanto um escravo pode suportar até três enlaces SCO para um mesmo mestre. Um mestre transmite pacotes SCO a um escravo em intervalos regulares, normalmente referenciado como o "intervalo SCO" e é contado em *slots*.

3.5 Gerenciamento de energia

Desenvolvida com o foco em economia de energia, a especificação do *Bluetooth* define três tipos diferentes de classes de radio, com valores variando de 1mW a 100mW. A titulo de comparação a potência típica de um outro padrão o IEEE802.11b que é em torno de 1000mW.

Outra característica importante do *Bluetooth*, é que a conexão entre o mestre e um escravo possui diversos modos de funcionamento que possibilitam o gerenciamento de energia. No modo *active* ocorre a comunicação direta, no modo *sniff*, o escravo desliga e periodicamente volta a escutar o *slot* de transmissão por alguma requisição do mestre, no *hold* o nodo escravo para de comunicar com o mestre por um tempo prédeterminado, voltando à comunicação após o fim deste e no modo *parked*, o nodo escravo abdica de seu endereço de comunicação ativa com o mestre, assumindo um endereço de *parked* e se desliga da *piconet*, mas periodicamente pode voltar a comunicar, sem precisar passar pelos procedimentos de *inquiry* e *paging*.

4 Java

Java é uma linguagem computacional adequada para o desenvolvimento de aplicações baseadas na rede Internet, redes fechadas ou ainda programas executáveis. Tecnologias no campo dos computadores, a linguagem Java foi das

que teve maior efeito no mercado [15]. Desde páginas web usando applets Java, até aplicações desenvolvidas completamente nesta linguagem. Foi desenvolvida na primeira metade da década de 90 nos laboratórios da Sun Microsystems com o objetivo de ser mais simples e eficiente do que suas precessoras. O alvo inicial era a produção de software para produtos eletrônicos de consumo (fornos de microondas, agendas eletrônicas, etc.). Um dos requisitos para esse tipo de software é ter código compacto e de arquitetura neutra.

A linguagem obteve sucesso em cumprir os requisitos de sua especificação, mas apesar de sua eficiência não conseguiu sucesso comercial [6]. Com a popularização da rede Internet, os pesquisadores da Sun Microsystems perceberam que aquele seria um nicho ideal para aplicar a recém criada linguagem de programação. A partir disso, adaptaram o código Java para que pudesse ser utilizado em microcomputadores conectados a rede Internet, mais especificamente no ambiente da rede mundial de computadores (*World Wide Web* - www). Com isso, a linguagem conseguiu uma grande popularização, passando a ser usada amplamente na construção de documentos Internet que permitam maior interatividade. Os principais navegadores Internet disponíveis comercialmente passaram a dar suporte aos programas Java, e outras tecnologias em áreas como computação gráfica e banco de dados também buscaram integrar-se com o novo paradigma proposto pela linguagem: aplicações voltadas para o uso em redes de computadores.

Atualmente, a linguagem Java é a força propulsora por trás de alguns dos maiores avanços da computação mundial, como:

- ✓ Acesso remoto a bancos de dados;
- ✓ Bancos de dados distribuídos;
- ✓ Comércio eletrônico;
- ✓ Interatividade em páginas WWW e em ambientes de Realidade Virtual distribuído;
- ✓ Ensino à distância;
- ✓ Jogos e entretenimento.

4.1 Código Interpretado e Portável

As diversas linguagens de programação podem ser tanto compiladas como interpretadas. Quando se utiliza uma linguagem compilada, é necessário executar um programa para traduzir os arquivos fonte, legíveis em linguagem de alto nível, em código executável. As linguagens compiladas têm a vantagem de produzir código de alta performance, o qual está ajustado para o funcionamento em um tipo específico de processador ou arquitetura de processador. Aplicativos compilados, chamados de código binário, só podem rodar no tipo de computador para o qual foram compilados, uma vez que esses aplicativos consistem, na realidade, em instruções em linguagem de máquina, entendidas e executadas pelo microprocessador.

As linguagens interpretadas só existem em código fonte. Quando em execução, um programa chamado interpretador toma o código fonte e executa as ações indicadas pelos comandos no arquivo. O interpretador é, na realidade, o único aplicativo que está sendo executado. Entre os benefícios das linguagens interpretadas está o fato dos programas interpretados poderem rodar em uma variedade de plataformas diferentes, pois estes só existem em código fonte. Além disso, são mais fáceis de depurar.

A linguagem Java é tanto compilada como interpretada. Após escrever um programa em Java, utilizando um editor de textos qualquer, salva-se o programa como código fonte. A seguir, pode-se compilar essa fonte, a fim de produzir um tipo de arquivo binário chamado de arquivo de classe. Esses arquivos não são executados diretamente, pois eles não contêm instruções que são entendidas diretamente pelos processadores atualmente disponíveis no mercado.

Os programas Java são compilados em um formato intermediário chamado bytecodes. Assim, esses programas podem ser interpretados e executados em qualquer sistema através de um interpretador Java, ou Java Virtual Machine (JVM). Com isso, o código precisa ser escrito e compilado apenas uma vez, pois os bytecodes gerados serão executados da mesma forma em qualquer plataforma de hardware e software.

4.2 Segurança

Por ter seu projeto voltado para a simplicidade de código, as possibilidades de erro de programação em Java são reduzidas. Apesar disso, a linguagem traz outros recursos para tornar seu código ainda mais eficiente. O processo de compilação - geração de bytecodes – é projetado para a detecção prévia dos possíveis erros, evitando que os mesmos se manifestem em tempo de execução.

Além de diminuir as possibilidades de erro de programação, a linguagem tem um esquema de segurança para garantir a integridade de código, principalmente no caso do código originário de rede insegura. Esses recursos de segurança são notáveis principalmente dentro do ambiente do interpretador [Hoff 1996].

Após baixar um applet (bytecodes para serem interpretados e executados em um navegador Internet) da rede, o interpretador faz uma verificação do código, buscando alterações intencionais ou não. A seguir, o interpretador determina o formato de memória para execução. Em outras palavras, não é possível acessar informações diretamente da memória ou inserir código estranho ao código original.

4.3 Java API

Uma aplicação desenvolvida em Java pode utilizar várias bibliotecas nativas desenvolvidas pela Sun, e o conjunto dessas bibliotecas Java é conhecido como Interface API (Java *Application Programming*). Nessas bibliotecas estão várias classes, organizadas em pacotes. Cada um desses pacotes traz classes com funcionalidade básica e vital para um determinado ramo de programação Java. Juntas, a JVM e a Java API constituem o ambiente de execução de Java ou ainda, a plataforma Java.

4.4 JCP

JCP (*Java Community Process*) é o processo aberto de padronização da tecnologia Java. É através desse processo que as novas APIs Java são especificadas e também que as APIs já existentes são evoluídas [SouJava, 2002]. O JCP define como a comunidade deve solicitar uma nova API, ou modificação, que ocorre através de uma JSR (*Java Specification Request*) e, uma vez aceita a solicitação, como essa solicitação deve ser encaminhada e atendida, qual o produto final (especificação + implementação de referência + teste de compatibilidade), quanto tempo o processo vai durar, etc. O JCP é o conjunto de normas que rege a evolução da plataforma Java.

4.5 Plataforma Java 2

De acordo com Sun Wireless (2002), Java 2 é a nova versão da plataforma Java da Sun Microsystems. Percebendo que um só produto não conseguiria abranger todas as necessidades do mercado, a Sun projetou a Java 2 em três edições: Java 2 Platform, Standard Edition (J2SE); Java 2 Platform, Enterprise Edition (J2EE); e Java 2 Platform, Micro Edition (J2ME). Cada edição define uma tecnologia e um conjunto de ferramentas com propósitos diferentes.

A J2SE foi a primeira edição criada da Java 2 Platform. Suporta o desenvolvimento de aplicações para desktop e estações de trabalho. J2SE é a versão básica do Java com a API padrão. Pouco tempo depois do lançamento da J2SE, uma nova edição foi criada, a J2EE, suportando aplicações mais robustas e destinada a servidores de grande porte. A API da J2EE fornece uma linha completa de funcionalidades e um ambiente integrado para a criação de aplicativos Java multi-camadas no servidor [12]. A plataforma J2EE, que expande a J2SE, é uma plataforma completa, robusta, estável, segura e de alta performance, voltada para o desenvolvimento de soluções corporativas [13]. O principal objetivo da Plataforma J2EE é reduzir a complexidade e o tempo (e, portanto o custo) do desenvolvimento de aplicações corporativas.

4.6 J2ME

Com o aumento expressivo de equipamentos eletrônicos móveis, como telefones celulares e PDAs, a Sun, de olho nesse mercado, criou a última integrante da plataforma Java 2, a Micro Edition. Essa tecnologia é uma versão compacta da linguagem padronizada pela Sun, e direcionada ao mercado de pequenos dispositivos [17]. J2ME na verdade é um conjunto de especificações que tem por objetivo disponibilizar uma JVM, API e ferramentas para equipamentos portáteis e qualquer dispositivo com poder de processamento menor que os atuais computadores de mesa.

A arquitetura J2ME é modular e escalável [11]. Esta modularidade e escalabilidade são definidas pela tecnologia J2ME em um modelo de três camadas embutidas sobre o sistema operacional do dispositivo: a camada de perfil, a camada de configuração, e a camada do interpretador. A aplicação encontra-se acima da camada de perfil. Podemos observar essas camadas na figura 5.

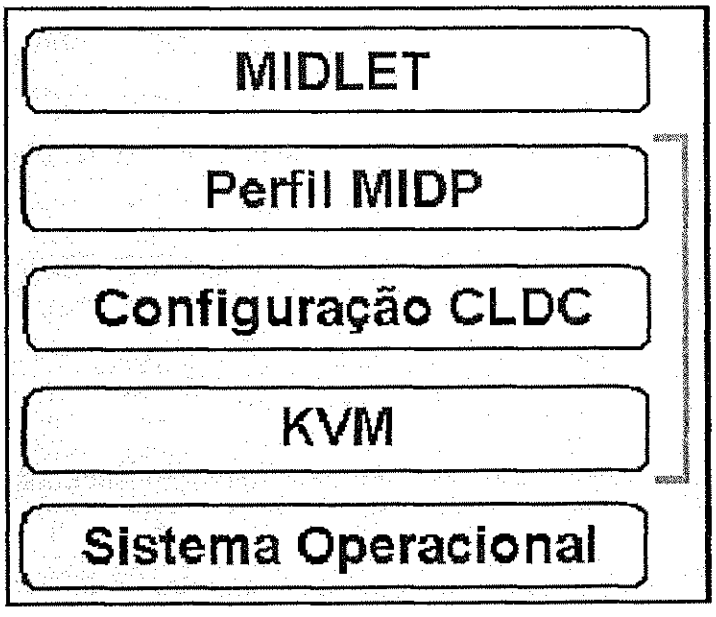

Figura 4: Camadas da J2ME

4.7 Camada De Configuração

Devido à grande quantidade de dispositivos existentes, variando desde placas com sistema operacional embutido até PDAs com grande poder de processamento, e funcionalidades diferentes, com conexões de redes velozes ou não, foi necessário definir um mínimo de funcionalidades (JVM e API) para uma faixa de equipamentos que tivessem características semelhantes (memória, processamento, conectividade) [7]. Logo, uma mesma configuração atende vários dispositivos diferentes, não possuindo detalhes específicos do dispositivo. Uma configuração serve como a base de execução e desenvolvimento das aplicações J2ME.

A API de uma configuração pode ter subconjuntos da API de J2SE. No momento existem duas Configurações: *Connected Device Configuration* (CDC) e *Connected Limited Device Configuration* (CLDC). A configuração CDC possui um conjunto de APIs que suportam equipamentos fixos de porte médio, tal como televisores. A configuração CLDC é um conjunto de APIs destinadas a aparelhos cujo poder de processamento, display e memória são limitados.

A CLDC é direcionada aos aparelhos com processamento de 16-32 MHz ou maior, 128 a 512 Kb de memória total, seja ela RAM, Flash ou ROM, disponíveis para a plataforma Java, fornecimento de energia limitado, normalmente por baterias, conectividade com algum tipo de rede, normalmente sem fio, conexões intermitentes e com largura de banda pequena (9600 bps ou menos), interfaces de usuário com recursos bastante limitados, ou mesmo sem interface visual, ou com variados níveis de sofisticação baseados em LCD, leds, etc.

Existem algumas limitações na configuração CLDC em relação à edição J2SE. As principais são:

- ✓ Não há suporte para números de ponto flutuante, portanto os tipos *float* e *double* não podem ser usados. Isso ocorre principalmente por causa do hardware que não suporta, pois operações com ponto flutuante requerem um maior processamento;
- ✓ Sem suporte a finalização (object.finalize());

- ✓ Suporte limitado a erros. Existem apenas três classes de erros: java.lang.Error, java.lang.OutOfMemoryError, e java.lang.VirtualMachineError. Erros de *rutime* são manipulados de maneira dependente de implementação, que pode terminar a aplicação ou efetuar um reset no dispositivo. Exceções de aplicação são utilizadas normalmente, como no J2SE;
- ✓ Não há suporte à Java Native Interface (JNI), por questões de segurança e performance.

A resposta do porquê dessas limitações está na memória e processamento limitados, características dos equipamentos portáteis [10].

4.8 Camada De Perfil

Uma especialização da JVM e API é necessários para determinados dispositivos (ou grupos deles), para se obter performance e outros detalhes específicos [8]. A implementação tem liberdade para estender a API e JVM para utilizar recursos do equipamento, o que causa a dependência da implementação.

Um Perfil funciona como o complemento da configuração, em que esta define o básico de funcionalidades e o perfil atende detalhes específicos do dispositivo. Alguns dos perfis existentes são: *Mobile Information Device Profile* (MIDP), RMI *Profile, Foundation Profile*, PDA *Profile, PersonalJava*, etc.

O MIDP é um perfil CLDC voltado especificamente aos dispositivos portáteis. O perfil MIDP especifica interface com o usuário, entrada e persistência de dados, manipulação de eventos, modelo de armazenamento orientado à registro, rede, mensagens, e aspectos relacionados ao ambiente de programação para esses dispositivos.

Uma aplicação Java baseada na configuração CLDC e no perfil MIDP, e portanto, voltada para dispositivos portáveis, é conhecida como Midlet.

4.9 Camada do Interpretador

Esta camada é uma implementação da máquina virtual Java customizada para dispositivos específicos.

A KVM (Kilobyte *Virtual Machine*) é o interpretador Java otimizado para dispositivos portáteis com recursos limitados. A KVM é extremamente pequena e eficiente, garantindo ótima performance. O nome Kilobyte provém do fato de que o seu consumo de memória é tão pequeno que é medido em kilobytes [SUN J2ME, 2002]. A quantidade mínima de memória requerida pela KVM é 128 Kb, incluindo o interpretador, um mínimo de bibliotecas especificadas pela configuração e algum espaço para rodar os aplicativos.

Rodando com as bibliotecas da plataforma J2ME, da configuração CLDC e do perfil MIDP, a KVM fornece um ambiente de execução específico para pequenas aplicações.

5 Materiais e métodos

No desenvolvimento desse trabalho, foi realizada uma revisão bibliográfica na tecnologia *bluetooth* para que pudéssemos conhecer suas limitações, baseado nesse estudo e na API para aplicações móveis, iniciou-se o desenvolvimento da aplicação.

Para desenvolvimento do software foi utilizada a ferramenta eclipse 3.0, que é uma ferramenta utilizada para desenvolvimento de software em Java 2, foi necessário instalar um plugin para aplicações móveis no eclipse. Conforme foi explicado no capitulo anterior, a plataforma Java utiliza API's, nesse caso utilizamos a MIDP, que oferece suporte a utilização de vários componentes para aplicações em sistemas móveis embarcadas, entre eles o *Bluetooth*.

Como não dispomos de dispositivos *Bluetooth*, precisamos de um ambiente para simulação, então utilizamos o J2ME *wireless Toolkit*, que fornece um ambiente de simulação para o desenvolvimento de aplicações MIDP.

6 Uma Aplicação Cliente/Servidor

Essa aplicação estabelece a comunicação ponto-multiponto entre dispositivos móveis para a realização de *download* de arquivos, utilizamos figuras para esses arquivos. Em sua implementação é necessário que um dispositivo seja configurado como servidor e os demais serão clientes, que deve ser configurado de maneira simples na tela de inicialização, conforme a figura 6.

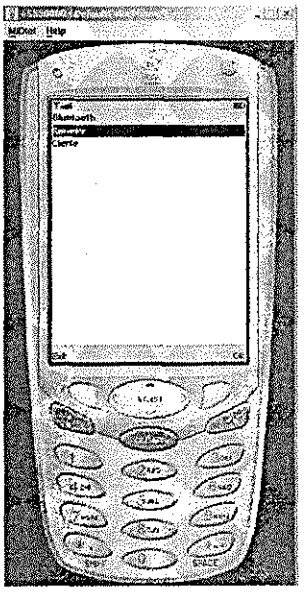

Figura 5: Tela inicial da aplicação bluetooth

Para demonstrar o funcionamento da aplicação, será configurado o servidor, em seguida, um cliente deve baixar uma figura do servidor.

Inicialmente, deve ser selecionado Servidor através da softkey (tecla comandada por software) OK, e será exibida uma tela de confirmação conforme a figura 7a. Após confirmação da conexão através da softkey Yes, a tela da figura 7.b será exibida. Então deve-se pressionar menu, e será apresentada uma tela onde poderá ser disponibilizada as figuras para que o clientes possam fazer o *download*, veja a figura 7.c. Para disponibilizar as figuras, basta selecionar "publish image" e pressionar SELECT, o ícone da imagem muda para a cor verde.

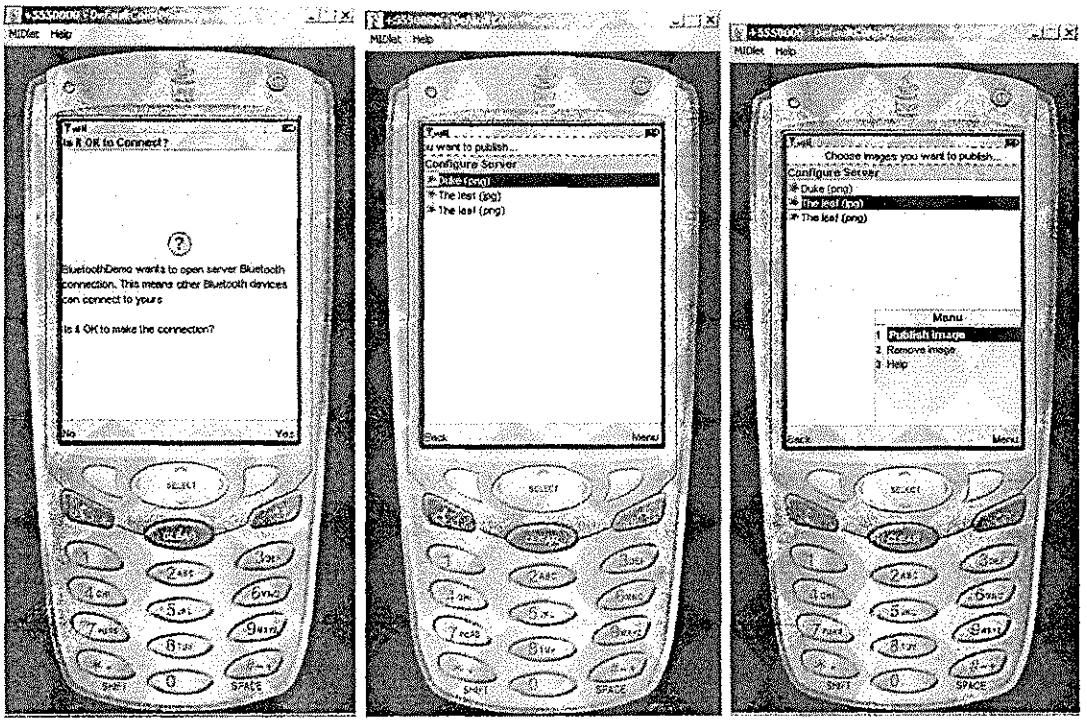

Figura 6: Telas de configuração da aplicação Servidor

Com isso, a aplicação servidora está configurada e qualquer dispositivo conectado a rede pode baixar uma dos dois arquivos disponíveis. Veremos, em seguida, como fazer isso da aplicação Cliente.

Na tela da figura 6, deve ser selecionada a aplicação Cliente, então será apresentada a tela da figura 8.a, na qual basta pressionar a softkey "find", então o cliente procura executa uma busca por dispositivos bluetooth dentro do seu raio de alcance, estando o servidor dentro desse raio, será encontrado pelo o cliente, o qual estabelecerá uma conexão e poderá visualizar os arquivos que foram disponibilizados pelo servidor, conforme ilustra a figura 8b. Então, basta selecionar uma figura disponível e pressionar a softkey Load, e o arquivo será transferido para o cliente, conforme ilustrado na figura 8c.

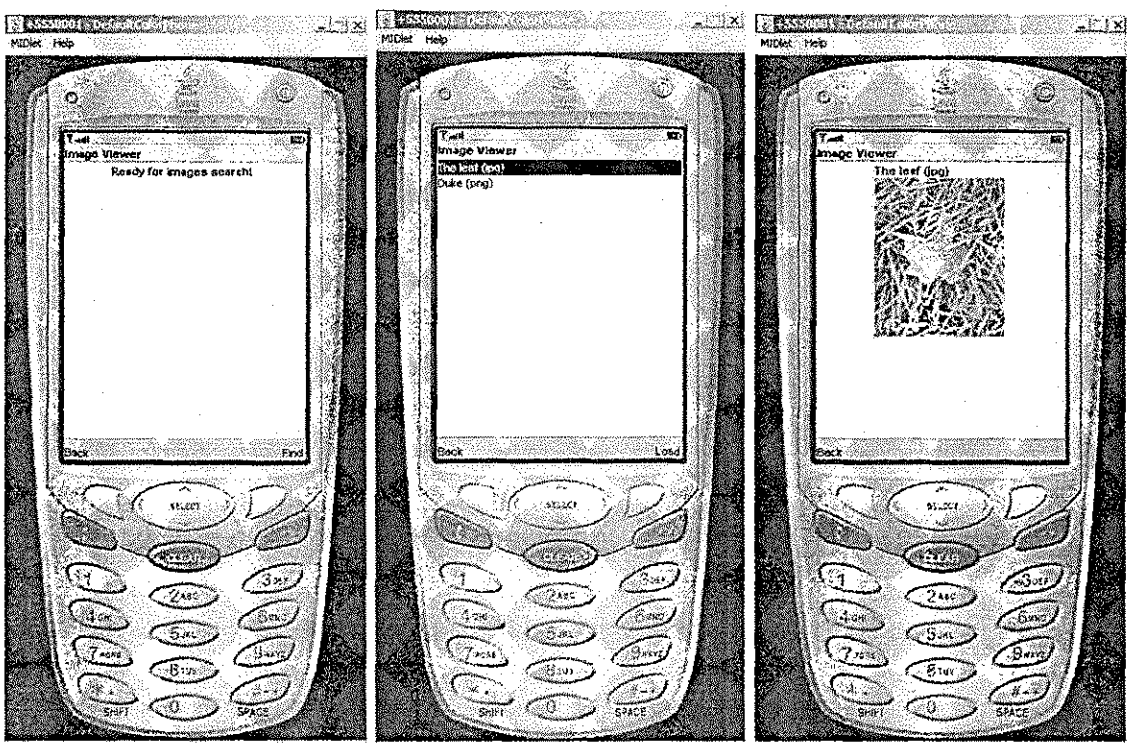

Figura 7: Tela de download de arquivos da aplicação cliente

Com isso, temos um exemplo de uma utilidade da aplicação para transferência de arquivos, através de comandos simples e de fácil navegabilidade ao usuário. Esse dispositivo foi implementado a partir da camada de aplicação e utilizando-se unicamente J2ME, que comprova sua versatilidade e portabilidade.

Através dessa aplicação é possível realizar transferência de arquivos maneira fácil, sem necessidade de cabos, instalação de *drives* necessários para que o Pc reconheça alguns hardwares. Dessa forma, o usuário pode transferir músicas do aparelho de som do carro para seu aparelho celular, atualmente, seria necessário para realizar essa operação, baixar o arquivo para um PC, e em seguida transferi o arquivo do PC para o equipamento desejado; transferir fotografias de uma câmera fotográfica digital para o equipamento da loja de revelação, atualmente o usuário pode fazer isso ou descarregando as fotos no seu PC e levando para a loja através de algum dispositivo de armazenamento, como um CD ou *jump drive* ou levando o CD de instalação do software da câmera, bem como, o cabo de transferência; o usuário pode transferir arquivos tais como,

músicas, vídeos, fotos, etc, de seu dispositivo equipado com bluetooth, um celular, por exemplo, para outro equipamento qualquer de um amigo que ocasionalmente precisou fazer tal transferência, atualmente, é necessária uma visita ao amigo e algum tempo até que ligue computador, conecte os cabos, e monte toda parafernália. Ou seja, o universo é o limite para a aplicabilidade de dispositivos equipados com essa aplicação bluetooth.

Diversas outras aplicações, inclusive já implementadas por grandes empresas, fazem de muitos dispositivos eletrônicos, o principal dele é o celular, como fone de ouvido, o qual é conectado ao aparelho via bluetooth e o usuário não precisa retirar o aparelho do bolso, ou do painel do carro, com um simples comando de voz, pode fazer chamadas ou atender ligações. Mouses e teclados sem fio, entre outros periféricos de um computador pessoal, já existem no mercado, diminuindo a quantidade de cabos, e oferecendo mobilidade, basta imaginar que o usuário não precisa esta a uma distancia limitada do PC para utilizar o mouse. Ou ainda com o aumento dos consumidores de computadores pessoais e utilização de internet de banda larga, tornou-se comum a utilização de redes domésticas nas residências, em vez de um monte de fio, pode ser usado um modem sem fio (*wireless*), possivelmente, através da tecnologia bluetooth. Ou seja, esse tipo de dispositivo tem um futuro promissor, e muitos tipos de conexões utilizando-se cabos estão com os dias contados.

6.1 Aplicações

Uma aplicação *bluetooth* como a desenvolvida neste trabalho é apenas um pequeno exemplo da vasta gama de utilização para conexões de curtas distancias que este tipo de tecnologia oferece. Como exemplo desse fato podemos citar a possibilidade de computadores conectados a seus periférico sem utilização de qualquer tipo de cabo, do teclado, mouse ou qualquer outro periférico. Mobilidade de o usuário escutar música sem incomodar outras pessoas e sem precisar ficar próximo ao equipamento que está reproduzindo o som.

O usuário pode ainda fazer transferências de arquivo de áudio do seu computador para o equipamento de som do seu automóvel sem a necessidade de nem um tipo de conexão através de fios, transferências de arquivos de mídia como fotografias ou vídeos de um aparelho celular para outro, ou até do seu aparelho celular para equipamentos como aparelhos televisores ou computadores.

Atender ao telefone celular dentro de seu automóvel sem a necessidade de segurar o aparelho próximo ao rosto, inviabilizando a utilização de uma das mãos e desviando atenção do transito. Estes são apenas algumas das inúmeras maneiras que a conexão bluetooth pode facilitar na vida dos seus usuários.

7 Conclusões

No presente trabalho, foi abordada a implementação de um dispositivo bluetooth utilizado-se a tecnologia J2ME. Nesse contexto, foi necessário realizar uma revisão bibliográfica sobre o protocolo e um estudo cuidadoso sobre a API bluetooth.

Durante o decorrer dos estudos observou-se que desenvolver aplicações em J2ME para dispositivos sem fio apresenta muitos desafios aos programadores, visto que o mesmo é uma versão compacta da linguagem padronizada da Sun, por ser justamente destinado ao desenvolvimento de aplicações para dispositivos móveis, e, portanto possuindo algumas limitações computacionais.

Dessa forma, é necessária uma preocupação especial no desenvolvimento de aplicações para esses dispositivos, pois eles têm bastante limitação em diversos aspectos, para perceber melhor esse fato basta se ater ao tamanho dos dispositivos, que geralmente deve caber na mão do usuário; limitações quanto ao consumo de energia *Shutdown* (desligamento), *startup* (reinicialização), *hivernação* (o dispositivo desliga todo o sistema, permanecendo ligado apenas alguma aplicação que esteja em execução), *standby* (o dispositivo permanece desligado até que seja enviado um comando para retira-lo de tal estado).

Tamanho reduzido de memória - deve-se produzir um código o mais enxuto possível, com bastante atenção para o uso desnecessário de memória, ou ainda quando for necessário pensar na forma que utilize a capacidade mais baixa possível, deve-se ter sempre em mente que a aplicação não é para desktop.

Baixa capacidade de processamento – a velocidade de processamento também é limitada, e aplicações que exijam muito desse recurso podem se tornar inviáveis, provocando o travamento do processador ou ainda cancelamento de funcionalidades de maior prioridade como atender uma chamada, assim deve-se otimizar o código para melhoria de desempenho.

Pequena área de interface ao usuário, o um dispositivo móvel dificilmente tem teclado alfanumérico, e quando tem, ainda é bastante desconfortável para o usuário. Dessa forma, via de regra deve-se colocar as opções de navegabilidade

no display. Sem esquecer que o display também é muito pequeno, exigindo que a interface seja bem organizada e de fácil operabilidade.

Outra preocupação é quanto à operabilidade do dispositivo, quem vai usar são pessoas que não entendem de informática, assim a curva de aprendizado deve ser curta sem necessidade de manuais, treinamentos ou vídeos. Dispositivos com interfaces complicadas têm bastante rejeição no mercado.

O meio físico utilizado para comunicação de dados é o espaço, do contrario não seria dispositivo móvel, esse meio de transmissão é instável e sofre bastante influência de fenômenos naturais. Exigindo maio tolerância à falhas e melhor tratamento de erros, latência elevada, criptografia e autenticação, pois o sinal pode ser interceptado mais facilmente.

A validação do trabalho foi possível através de simulação, pois dispositivos móveis com Máquina Virtual Java e aplicação *bluetooth* ainda custam caro. Ou seja, a aplicação foi executada em simuladores para desktop, assim não foi possível uma validação da aplicação usando dispositivos reais. De qualquer forma, o simulador é uma aproximação muito boa dos aparelhos celulares da Motorola, nos quais são executados testes dos seus produtos e exibiu bons resultados, talvez sejam necessários apenas pequenos ajustes.

As conexões *bluetooth* apresentam bastantes vantagens incontestáveis, como fácil usabilidade para os usuários, praticidade para conexão de dispositivos que são equipados com esse tipo de tecnologia, conexões dispensam a utilização de cabos e é gratuita. Mas deve-se ter em mente que a aplicação é bastante limitada e tem como objetivo principal conexões de curta distância sem a utilização de cabos, pois seu alcance padrão e de 10 metros de distância. A tecnologia J2ME tem sua maior vantagem a modularidade que torna uma ferramenta mais segura e fácil de utilizar do ponto de vista do desenvolvedor, bem como sua portabilidade, pois é possível executar tais aplicações em qualquer dispositivo, desde que os mesmo tenham a máquina virtual JAVA instalada. Mas para isso, é pago um preço alto e aplicações desenvolvidas na plataforma JAVA são pesadas e apresentam baixo desempenho, exigindo bastante memória e do processador, tornando os dispositivos caros.

8 Trabalhos Futuros

Como trabalho futuro baseado nas tecnologias utilizadas, podem ser implementado sistemas para comunicação de voz utilizando-se bluetooth, a qual não é viável a utilização da plataforma J2ME;

Desenvolvimentos de diversos tipos de aplicações em J2ME para dispositivos móveis ou portáteis, tais como jogos eletrônicos, calculadoras, etc. O desenvolvedor que optar por desenvolver em JAVA conta com as vantagens de sua portabilidade, no entanto deve está preparado para seu baixo desempenho e para a impossibilidade de controle da máquina, utilizando-se a plataforma JAVA é impossível o desenvolvedor realizar alguma ação destrutiva ao sistema, ou até ao dispositivo, mas por isso não se tem o menor controle sobre a máquina, tornando o sistema pesado, logo a tecnologia J2ME está longe de ser utilizada para construção dos sistemas embarcados na camada de sistema, podemos utilizá-ala essencialmente na camada de aplicação.

9 Referências Bibliográficas

[1] Jennifer Bray and Charles Sturman. Bluetoot Connect Without Cables. PrenticeHall,1 edição, 1999.

[2] *Bluetooth* Forum. http://www.*Bluetooth*.com.

[3] IEEE 802.15, Wireless Personal Área Network, HTTP://WWW.IEEE802.ORG/15/, 20/07/2005

[4] ANUFF, Ed. **Java sourcebook**. Nova York: Wiley, 1996.

[5] HOFF, Arthur Van; SHAIO, Sami; STARBUCK, Orca. **Ligado em Java**. São Paulo: Makron Books, 1996.

[6] INDRUSIAK, Leandro Soares. **Linguagem Java,** [S.l.], nov. 2002, http://www.inf.ufrgs.br/tools/java/introjava.pdf

[7] MICROJAVA Network. **The J2ME resource,** [S.l.], nov. 2002, http://www.microjava.com/

[8] MOTOROLA. **Developer community,** [S.l.], nov. 2002. http://idenphones.motorola.com/iden/developer/developer_home.jspSOUJAVA, 20/07/2005.

[9] **Sociedade de usuários Java,** [S.l.], nov. 2002, http://www.soujava.org.br

[10] SUN. **Technology for creating mobile devices**, [S.l.], nov. 2002, http://java.sun.com/products/cldc/wp/KVMwp.pdf

[11] SUN J2ME. **Java 2 platform, micro edition**, [S.l.], nov. 2002, http://java.sun.com/j2me/

[12] SUN Brasil. **Soluções corporativas**, [S.l.], nov. 2002. http://br.sun.com/

[13] SUN Wireless. **Java wireless developer**, [S.l.], nov. 2002, http://wireless.java.sun.com/, 03/08/2005

[14] NUNES, Nuno. A microarquitectura PicoJava, [S.l.], nov. 2002. Disponível em: <http://www.fe.up.pt/~ee99043/picojava/picojava.pdf>. Acesso em: 17 nov. 2002.

[15] TANEMBAUM, A.; Redes de Computadores. Editora Campus, 1994. (4 ex.)

[16] ALBITZ, P.; LIU, C. DNS and BIND. 3. ed. Sebastopol: O´Reilly & Associates, sep. 1998.

[17] FONSECA, Jorge Cavalcanti. Portando a KVM . 2002. 64 f. Trabalho de Conclusão de Curso (Graduação em Ciências da Computação) – Centro de Informática, Universidade Federal de Pernambuco, Recife.