UNIVERSIDADE FEDERAL DO CEARÁ
CENTRO DE TECNOLOGIA
DEPARTAMENTO DE ENGENHARIA DE TELEINFORMÁTICA

JMED: UMA ARQUITETURA *PEER-TO-PEER* PARA APLICAÇÕES EM TELEMEDICINA

Autor: Cláudio Pedrosa Teles
Orientador: Prof. Phd. Helano de Sousa Castro

Dissertação apresentada junto ao Departamento de Engenharia de Teleinformática da UFC para obtenção do título de Mestre em Engenharia de Teleinformática.

Fortaleza – Ceará
23 de Junho de 2006.

CLÁUDIO PEDROSA TELES

JMED: UMA ARQUITETURA *PEER-TO-PEER* PARA APLICAÇÕES EM TELEMEDICINA

Esta Dissertação foi submetida como parte dos requisitos necessários à obtenção do Grau de Mestre em Engenharia de Teleinformática, outorgado pela Universidade Federal do Ceará, e encontra-se à disposição dos interessados na Biblioteca da referida Universidade.

A citação de qualquer trecho desta Dissertação é permitida, desde que seja feita de conformidade com as normas da ética científica.

_____

Cláudio Pedrosa Teles

Banca Examinadora:

_____

Prof. Phd. Helano de Sousa Castro
Departamento de Engenharia de Teleinformática – UFC

_____

Prof. Dr. Paulo César Cortez
Departamento de Engenharia de Teleinformática – UFC

_____

Prof. Dr. Giovanni Cordeiro Barroso.
Departamento de Engenharia de Teleinformática – UFC

_____

Prof. Phd. Julius César Barreto Leite.
Departamento de Ciências da Computação – UFF

Fortaleza, 23 de Junho de 2006.

Este trabalho é dedicado ao meu Senhor, meu Pai amado. Por causa da Tua Palavra me moveste até aqui, e esta obra é um dos frutos colhidos dos dias que me deste. Não é para minha honra, mas para honra Tua, pois quem impedirá o teu agir, oh Senhor! Louvado seja sempre e eternamente.

Agradecimentos

Primeiro agradeço à DEUS por ter me sustentado e me animado durante todo o período em que estive desenvolvendo este trabalho. Graças te dou Pai, pela vitória conquistada nesses dias. Tudo isto é vitória Tua.

Aos meus pais, que mesmo estando distantes, estiveram sempre presentes no meu coração. E ao meu irmão e irmã, representando suas famílias, que me proporcionaram descanso quando precisei.

Ao Prof. Helano de Sousa Castro, meu orientador, pelo seu apoio, ensino e colaboração efetiva para o desenvolvimento deste trabalho. Obrigado pelas palavras de encorajamento e decisão.

Ao Prof. Giovanni Cordeiro Barroso, pela sua atenção, apoio e além de poder ter me coordenado em projetos de pesquisa que trouxeram os recursos financeiros para meu sustento durante este período de estudo. E ao Prof. Paulo César Cortez, pelas palavras de motivação.

Aos meus amigos e amigas que contribuíram direta e indiretamente para realização deste trabalho, em especial Atslands, Ana Flávia, Marcos Aurélio, Marcelo Sousa e Rodrigo Santana.

E não poderia deixar de agradecer a uma pessoa muito especial que exerceu um papel muito importante para que eu viesse a concluir este trabalho, Samia Jainara (minha esposa). Meu amor, eu te agradeço pela tua companhia, por tuas orações e lágrimas derramadas, eu te agradeço pelo sorriso e carinho derramado sobre minha vida quando necessitei. Esta vitória também é vitória tua, pois DEUS nos fez um só corpo, uma só alma e um só espírito. DEUS te abençoe!

Resumo

O avanço tecnológico na área de arquiteturas computacionais tem proporcionado o uso de sistemas de computação em muitas aplicações que ainda não se haviam se beneficiado com esta tecnologia. Em particular, as arquiteturas computacionais distribuídas, associadas com a computação móvel e a comunicação móvel, alargaram o horizonte de aplicabilidade em grande proporção. Dentre as aplicações que se beneficiaram com estes avanços, a Telemedicina desponta com diversas áreas médicas fazendo uso desta tecnologia. Por Telemedicina, entende-se a distribuição de assistência médica e a colaboração do conhecimento médico à distância, por intermédio do uso dos atuais meios de comunicação. Muitas soluções propostas para suportar Telemedicina se baseiam na arquitetura Cliente-Servidor. Esta arquitetura apresenta uma baixa confiabilidade pelo fato de contar com um elemento central (Servidor) para prover os serviços solicitados. É claro que redundância pode ser incorporada em nível de servidor, mas ao alto preço de ter-se que garantir a consistência dos servidores replicados. Nesta dissertação, é proposta uma arquitetura computacional *peer-to-peer* (P2P), que elimina o elemento central encontrado na arquitetura Cliente-Servidor. É discutida a importância de se utilizar esta tecnologia na oferta de serviços de assistência médica especializada a distância, propondo um estilo de colaboração direto entre médicos, a partir da interligação entre equipamentos computacionais. Dentro deste contexto, é apresentada uma solução, denominada JMED, que é desenvolvida por meio de uma plataforma de programação para o ambiente da computação de rede distribuída *Peer-to-Peer*, conhecida por JXTA. JMED foi implementada e sua aplicação em Telemedicina foi validada a partir de uma aplicação voltada para segunda opinião médica, fazendo uso tanto de sistemas de computadores, como de telefones celulares.

Abstract

In the last years Research and Development (R&D) in the area of computer systems and communication systems have started a technological revolution which has made it possible to conceive a number of new products, particularly embedded systems, from cell phones to medical equipments. The area which makes use of both technologies above is frequently known as Information Technology (IT). IT has boosted a number of research projects and has established as a research area in its own. Among the applications which have benefited from IT, Telemedicine is one that stands out. Telemedicine is the distribution of medical care, as well as medical knowledge cooperation at distance by using IT. Many solutions proposals to support Telemedicina if base on the architecture Client-Server. This architecture presents low Reliability for the fact to count on a central element (Server) to provide the requested services. It is clearly that redundancy can be incorporated in server level, but to the high price to have itself that to guarantee the consistency of the talked back servers. In this dissertation we propose a communication architecture based on peer-to-peer (P2P) computing, that it decides the problem of the architecture Client-Server. The importance is argued of if to use this technology in offers in the distance of services of specialized medical assistance, considering a direct style of contribution between doctors, from the interconnection between computational equipment. The proposed system, named JMED, was developed using programming platform for distributed P2P computing known as JXTA. JMED was implemented and we showed that is can be used in Telemedicine application by designing an application known as "second opinion", where computers and cell phones were used as communication stations.

Sumário

Lista de Figuras

Lista de Tabelas

CAPÍTULO 1
INTRODUÇÃO

As tecnologias da informação (TI), associadas à microeletrônica, têm proporcionado um avanço crucial na área da medicina, sobretudo no que concerne a prestação de assistência médica, diagnóstico e exames clínicos.

Com o constante desenvolvimento de TI, diversos setores sociais e científicos, principalmente na área de assistência médica, sentiram o impacto da evolução tecnológica. A facilidade, agilidade e segurança com que estas tecnologias manipulam os dados para gerar informações, causaram uma mudança no ambiente de trabalho e na natureza dos dados, que passaram a ser operados de forma eletrônica ou digital. Neste contexto, realiza-se a fusão das áreas de Telecomunicação, Tecnologia da Informação e Medicina, nascendo uma área conhecida por Telemedicina.

A Telemedicina pode ser conceituada como sendo a distribuição da assistência médica e a colaboração do conhecimento médico à distância, por intermédio do uso dos atuais meios de telecomunicação (PATTICHIS, 2002). Ela compreende diversas áreas de atuação, tais como: educação à distância, formação de comunidades virtuais, Teleconsulta, Telepatologia, Teleradiologia, Telecirurgia e Telemonitoração, dentre outras (SEABRA, 2003).

Embora os termos pareçam modernos, os mesmos são bastante antigos, ao desvincular-se o uso da informática como mediadora da comunicação. Hoje, a Internet tem mudado a maneira de promover a Telemedicina. Por este meio é possível transmitir e receber dados de pacientes, imagens médicas, compartilhar experiências em grupos de discussão sobre determinados temas, além de despertar o desenvolvimento de soluções que visem o intercâmbio de informações para diagnósticos, prevenção e tratamento de doenças, nos casos em que a distância e a ausência de profissionais especialistas são fatores críticos.

Questionamentos, como por exemplo: se as tecnologias de informação serão ou não utilizadas nos projetos terapêuticos, preventivos e propedêuticos em favor dos níveis de vida e de saúde das populações, não cabem serem mais discutidos atualmente (FRANÇA, 2001). Cabe-se então discutir novas técnicas de comunicação de dados que

facilitem o intercâmbio da informação entre médicos e também entre médicos e pacientes, de forma segura e confiável.

Ao se analisar a evolução das telecomunicações, considerando sua influencia na área de assistência médica em saúde e comunicação de dados, podem-se destacar algumas tendências que deverão determinar sua evolução, bem como características que se assemelham às encontradas em tecnologias de computação distribuída. São elas (SALEMA, 2003):

1. naturalidade, no sentido de tornar a comunicação cada vez mais próxima da comunicação presencial; a naturalidade que se procura ao progredir da transmissão de símbolos (como no telégrafo), para o som (telefone e rádio), das imagens fixas (como no fax), para as imagens em movimento acompanhadas de som (videofonia, televisão), até as imagens 3D e som envolvente na realidade virtual;
2. simplificação e facilidade de uso dos procedimentos, tornados possíveis pela automação e pela inteligência embutida em dispositivos eletrônicos através de sistemas embarcados;
3. integração de serviços e aplicações sobre uma mesma rede de comunicação e, na medida do possível, através de um só terminal micro-processado;
4. interatividade, definida como capacidade de reagir à comunicação e transmitir esta reação ao interlocutor, privilegiando a bidirecionalidade;
5. personalização, entendida como a adaptação do conteúdo das mensagens, tanto ao equipamento terminal como às preferências do destinatário;
6. ubiqüidade, isto é, a possibilidade de se comunicar em qualquer local, em qualquer instante, de qualquer modo, com qualquer dispositivo interlocutor.

Sob um ponto de vista técnico, baseado no contexto tecnológico acima, tem-se uma perspectiva futura, e que já pode ser explorada hoje, para digitalização dos conteúdos a transmitir (armazenar); a integração de serviços e aplicações numa mesma rede através do uso de um único terminal; alterações dos suportes de transmissão, com substituição dos cabos metálicos por cabos de fibra óptica ou ligações à rádio (SALEMA, 2003).

Os limites geográficos foram eliminados através da interconexão das redes de telefonia móvel com as redes de computadores, trazendo uma sociedade global ligada

eletronicamente, o desafio é desenvolver novos modelos e práticas colaborativas que possam inovar a assistência médica à distância. Acredita-se que através do uso de modelos de sistemas computacionais distribuídos, que vise um processo de comunicação e colaboração máquina-a-máquina sobre a infra-estrutura da *Internet*, criar-se-ão novos sistemas de assistência médica a pacientes que ampliarão a margem dos benefícios para pessoas que não dispõem de acesso a especialistas, ou para aquelas cuja atenção básica é precária ou inexistente.

Assim, tais modelos de computação distribuída poderão contribuir para que centros de excelência em saúde possam oferecer serviços de assistência médica especializada à distância, de forma mais eficaz e colaborativa, como por exemplo, através da prática da "Telemedicina móvel"[1]. Nisto pode-se dizer que se ganha na redução de tempo e despesas na locomoção dos pacientes, na interação entre profissionais, na qualidade da reciclagem médica, no concurso rápido de profissionais de diversas áreas em acidentes de massa, no gerenciamento dos recursos em saúde, na descentralização da assistência à saúde, entre tantos (FRANÇA, 2001).

Apesar de todos os benefícios tecnológicos apresentados acima, o acesso à assistência médica remota é limitado devido ao fato das soluções computacionais aplicadas na área de Telemedicina serem comumente baseada em arquitetura Cliente-Servidor. Isto reside no fato dessas soluções não serem tolerantes à falhas, e apresentarem baixo desempenho. Estas duas desvantagens estão relacionadas com o fato da arquitetura Cliente-Servidor ser centralizada em servidores, que constituem componentes fundamentais para o funcionamento do sistema. Outro problema nesta área está relacionado com a necessidade de um sistema de comunicação que proporcione uma colaboração direta entre profissionais de saúde, e que suporte a mobilidade destes no exercício da assistência médica remota. Para resolver os problemas citados acima, é proposta, nessa dissertação, uma arquitetura *Peer-to-Peer*[2] que foi implementada dentro do contexto de Telemedicina.

[1] Telemedicina Móvel, termo aqui usado para caracterizar a prática da Telemedicina através do uso de dispositivos móveis e portáteis, como por exemplo: telefone celular, PDA (*Personal Digital Assistent*) e outros.

[2] *Peer-to-Peer* é um modelo de computação distribuído caracterizado por um estilo de comunicação descentralizado e direto entre os participantes de uma rede de dados. Ela permite a colaboração fim-a-fim entre dispositivos com funcionalidades semelhantes, ou seja, cada dispositivo pode atuar tanto como cliente como também servidor. Tal modelo é explorado no capítulo 2.

1.1 Objetivos do trabalho

Este trabalho está relacionado com o estudo da aplicação de tecnologias de computação distribuída na área da Telemedicina, visando alcançar, por meio destas, a mobilidade da informação de pacientes e as disponibilizando para médicos dentro de uma comunidade virtual, sustentada pela infra-estrutura da Internet.

Em termos gerais, o objetivo principal do trabalho é destacar a importância da tecnologia *Peer-to-Peer* como sendo uma tecnologia propulsora para o desenvolvimento de novos modelos de serviço de assistência médica especializada a distância, sendo capaz de promover uma genuína colaboração entre médicos e também entre médicos e pacientes. O objetivo específico do trabalho consiste em propor, desenvolver e avaliar uma arquitetura computacional que possibilite:

- criação de uma comunidade virtual *Peer-to-Peer*, sustentada por informações compartilhadas entre dispositivos de comunicação de médicos, tais como *desktops*, PDA e telefone celular;
- permitir o compartilhamento de arquivos digitais dentro da comunidade formada (favorecendo a transmissão de imagens médicas);
- trocar mensagens instantâneas entre os dispositivos participantes da comunidade, independente da localização de seu ambiente computacional;
- prover um serviço de assistência médica colaborativa através da *Internet*.

Por meio dos objetivos especificados acima, acredita-se que a Telemedicina utilizará a infra-estrutura da Internet de forma mais intuitiva, possibilitando ao médico, independente de sua localização geográfica, ter acesso a uma estrutura de serviços especializados, dentro de uma comunidade de saúde organizada. Esta estrutura visa favorecer o desenvolvimento de inovações na área de "segunda opinião médica"[3] através de uma arquitetura de comunicação de suporte ao serviço de análise e diagnóstico clínico, sustentado por um modelo de computação distribuída descentralizada, máquina-a-máquina.

[3] Segunda opinião médica, termo que é utilizado para abordar o intercambio de informações entre médicos que objetivam obter opiniões de outros colegas para alcançar um diagnóstico preciso sobre um determinado caso clínico.

1.2 Estrutura do trabalho

Este trabalho está organizado em 6 capítulos descritos a seguir.

No segundo Capítulo, são discutidas algumas tecnologias emergentes e relacionadas com a mobilidade da informação. Este capítulo está focado principalmente na descrição da tecnologia *Peer-to-Peer*, a qual é aqui considerada como uma das tecnologias importantes para novos modelos de sistemas distribuídos na área de Telemedicina. Ele aborda seus principais conceitos, arquitetura de rede, modelos de comunicação e a classificação de suas aplicações, além de diferenciá-la da tradicional tecnologia Cliente-Servidor.

No terceiro Capítulo são descritos as principais características e conceitos definidos na especificação da tecnologia JXTA, concebida pela *Sun Microsystems* como plataforma de programação para computação distribuída *Peer-to-Peer*.

No quarto Capítulo é descrito o ambiente de desenvolvimento de ferramentas implementadas sobre a infra-estrutura da Internet para prover a prática da Telemedicina. Neste capítulo, além de serem expostas as características de tal ambiente, também abordar-se-ão as vantagens concernentes ao uso da tecnologia JXTA para prover soluções para prática da Telemedicina.

No quinto Capítulo é contextualizada a importância da aplicação da tecnologia *Peer-to-Peer* na área de Telemedicina. Neste capítulo é proposta uma arquitetura de sistema computacional colaborativo, que visa alcançar a mobilidade das informações de pacientes, sustentando um serviço de assistência médica à distância sobre a infra-estrutura da Internet. As características do sistema, citadas acima, são viabilizadas a partir da interação entre equipamentos computacionais de médicos, incluindo celular, PDA e computadores, formando uma nuvem *Peer-to-Peer* global de apoio à assistência médica especializada.

Finalmente, no sexto Capítulo serão apresentados uma conclusão para este trabalho, fundamentada nos resultados obtidos pelo experimento, bem como sugestões para trabalhos futuros.

CAPÍTULO 2

Tecnologias emergentes: em busca da mobilidade da informação

A convergência das redes de telefonia móvel com as redes de computadores proporciona ao usuário de um sistema computacional a possibilidade de acesso irrestrito às suas informações por meio de uma comunicação transparente. Como resultado disto, observa-se a tendência das tecnologias de informação convergirem em busca da mobilidade e portabilidade da informação, tornando-a cada vez mais presencial e disponível. Para esta conquista, três dimensões são consideradas (KLEINROCK, 2003):

1. mobilidade, possibilidade do usuário se deslocar, mantendo seu acesso a uma rede de comunicação e serviços que seja transparente e adaptativo;
2. sistemas embarcados (*embedded systems*), inteligência de pequenos sistemas computacionais embarcados em equipamentos micro-processados, permitindo, por exemplo, a automação de procedimentos para conexão com a Internet;
3. ubiqüidade, a possibilidade de acesso constante ao serviço de comunicação, independente do dispositivo utilizado.

A partir do desenvolvimento das dimensões citadas, o acesso às informações e o processamento de dados deixaram de estar limitados a servidores, *desktops* e *notebooks* e passaram a abranger novos dispositivos computacionais, como por exemplo, celular, PDA, e outros equipamentos com características de conectividade de rede e recursos multimídia. Desta forma, as plataformas computacionais estão migrando de sua antiga natureza de serviços centralizados (baseada em tecnologias Cliente-Servidor), para uma natureza distribuída (máquina-a-máquina), mudando assim o contexto para computação *Peer-to-Peer*.

A partir dos aspectos citados acima, estima-se que aplicações relacionadas com a prestação de assistência à saúde logo passarão por uma grande inovação na maneira de executá-las. O ambiente de interconexão entre os dispositivos de diversas redes de comunicação contribui em muito para o aperfeiçoamento de novos modelos de assistência médica à distância, sendo este o motivo que destaca a importância de se aprofundar conhecimentos na tecnologia *Peer-to-Peer*.

2.1 Sistemas de computação rede distribuída – tecnologia *Peer-to-Peer* e Cliente-Servidor

Segundo Coulouris (COULOURIS, 2001), um sistema distribuído é todo aquele em que os componentes estão localizados numa rede de computadores, comunicando e coordenando as suas ações por troca de mensagens. Mesmo separados geograficamente, estes componentes devem estar orientados para trabalhar cooperativamente no sentido de cumprir com determinada tarefa. Ao analisar a evolução destes sistemas, pode-se observar que ela está relacionada ao desenvolvimento tecnológico das redes de computadores.

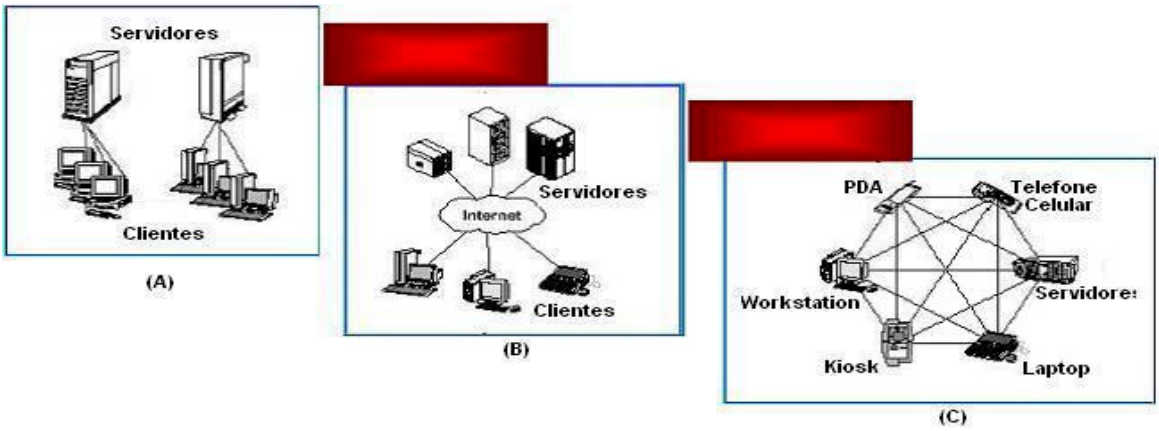


Figura 2.1: evolução da arquitetura de computação distribuída.

Conforme está ilustrado na Figura 2.1, a arquitetura de sistemas distribuídos iniciou-se através do modelo hierárquico Cliente-Servidor, representado na Figura 2.1.a. Este modelo foi inicialmente utilizado em redes locais de computadores ou *Intranets*. Ele é caracterizado por sua limitada, ou inexistente, interoperabilidade, e suas aplicações apresentam a necessidade de uma plataforma homogênea de sistema operacional, tanto para clientes como para servidores da rede (JXTA, 2005).

A partir do modelo Cliente-Servidor foram surgindo variantes, como por exemplo, *Web-Services*, tecnologia utilizada na *Internet*, e representada na Figura 2.1.b. Este modelo introduz uma melhoria em nível da interoperabilidade pela comunicação, baseada em padrões *XML*, alcançando a heterogeneidade de clientes e servidores, além de permitir, por parte dos servidores centralizados, o fornecimento de serviços e recursos para clientes universais distribuídos (JXTA, 2005).

Finalmente, na Figura 2.1.c, encontra-se a representação do conceito *Peer-to-Peer*, mais conhecido por P2P, o qual abstrai a forma descentralizada no fornecimento de serviços e recursos, eliminando a presença de uma autoridade central de serviços, além de adicionar a capacidade de auto-estruturação ou organização dos elementos da rede. Tal natureza descentralizada e distribuída, torna os sistemas P2P inerentemente robustos a certos tipos de problemas muito comuns em sistemas centralizados, como por exemplo, ter um servidor como um ponto de falha único da rede, tornando-a suscetível a falhas que acabam comprometendo a segurança e a disponibilidade de serviços.

Nos sistemas P2P, todas as entidades são potencialmente tratadas como servidores e clientes ao mesmo tempo, o que permite a tais sistemas possuírem duas características importantes (DING, 2003): escalabilidade (*Scalability*) e confiabilidade (*Reliability*).

A escalabilidade está relacionada ao fato de que os sistemas P2P normalmente crescem em quantidade de recursos disponíveis à medida que o número de usuários da rede também cresce. Desta forma, a complexidade de funcionamento do sistema deve se manter constante e indiferente da quantidade de componentes do sistema.

Já a característica de confiabilidade dos sistemas P2P reside no fato de que a não operação de quaisquer componentes pertencentes ao sistema não deve afetar o funcionamento do mesmo (a tolerância à falhas pode ser obtida pela distribuição das responsabilidades do fornecimento dos serviços entre os componentes do sistema).

A idéia da computação P2P não é nova, pois, ela já existe desde a comunicação inicial entre duas máquinas para trocar informações. Historicamente, isto é comprovado através do USENET[4], partindo do fato que este sistema possui características da tecnologia P2P (como por exemplo, nele não existe nenhuma autoridade central, a distribuição dos documentos é gerida por cada componente e o conteúdo da rede é replicado por todos os componentes da mesma). Além disto, tal estilo de computação pode ser encontrado também nas atividades de servidores de correio eletrônico, quando interagem diretamente uns com os outros para enviar, rotear e receber mensagens eletrônicas, e na operação da resolução de nomes (na *Internet*) que é realizada de

[4] USENET é um sistema que foi criado em 1979 por estudantes universitários dos Estados Unidos (EUA), para permitir que dois computadores trocassem informação (isto antes da conectividade oferecida pela Internet). A primeira iteração consistia na capacidade de um computador ligar via modem a outro, pesquisar documentos novos e transferir estes documentos para armazenamento local. Tal sistema evoluiu para o *newsgroups* conhecido hoje em dia (http://www.newsgroup.net/) .

maneira distribuída através da cooperação entre diversas máquinas (servidores de domínio de nomes – *DNS*).

2.2 Computação P2P e Cliente-Servidor

Um conceito fundamental que norteia a diferença básica entre o modelo de computação P2P e o modelo Cliente-Servidor, está relacionado ao fato de que as informações e os serviços disponíveis na rede P2P poderem ser geridos a partir dos *edges*[5] da rede, não sendo somente disponibilizados por elementos centrais da mesma (DTSOUMA, 2003). Desta forma, a computação P2P possui a característica de dar a máquinas individuais, a capacidade de fornecer serviços umas às outras, mudando o paradigma existente em tecnologias Cliente-Servidor, à medida que não depende de uma organização central ou hierárquica, além de disponibilizar aos seus integrantes as mesmas capacidades e responsabilidades.

O modelo de computação P2P, por não depender de servidores centrais, normalmente utiliza um esquema de endereçamento diferente do tradicional sistema de nomeação utilizado em tecnologias Cliente-Servidor. Sistemas P2P necessitam de conexões diretas entre os participantes de sua rede (formando uma espécie de "rede virtual"[6] de dispositivos) para alcançar a interconectividade entre eles, como está ilustrado na Figura 2.1.c. Desta forma, seu esquema de endereçamento é geralmente baseado em identificadores únicos e não dependentes de endereços IP.

A Tabela 2.1 a seguir destaca as principais características que diferenciam a tecnologia P2P da tecnologia Cliente-Servidor.

[5] *Edge* é o termo utilizado para caracterizar os elementos de uma rede computacional que estão hierarquicamente posicionados nas bordas da mesma, ou seja, que dependem da conexão de um elemento central de serviços. Desta forma, os *edges* tornam-se pontos incapazes de fornecer serviços a outros elementos da rede por estarem dependentes de outros elementos. Por exemplo: temos os servidores de acesso à *Internet* (elementos centrais de serviços) e temos os clientes (que são os *edges*) que só podem ter acesso à rede a partir dos servidores (o que os torna coadjuvantes na *Internet*, acessando recursos providos apenas por outras máquinas intermediárias).

[6] Rede virtual é formada pela interconexão de componentes da rede, executando sobre a infra-estrutura de uma rede física. Exemplo: a Internet, com infra-estrutura IP.

Característica	Cliente-Servidor	*Peer-to-Peer* (P2P)
Tráfego na rede	Assimétrico, ocasionando baixo *upstream* na rede (distinção explícita entre clientes e servidores).	Simétrico (todas as entidades participantes da rede podem interagir através de comunicações bidirecionais).
Armazenamento da informação	Centralizada em servidores.	Distribuída entre entidades.
Endereçamento	Primeiramente estático usando *DNS*, utiliza *NAT* para alcançar mascaramento de poucos endereços IP válidos para uma grande rede de computadores, abusa no uso da porta 80.	Dinâmico, registrado em tempo real e dispensa o uso de *NAT*, e *firewalls* para filtragem de protocolos e portas específicas.
Desempenho	Tende a diminuir com o crescimento de usuários na rede (sobrecarga nos servidores e requer maior largura de banda).	Tende a aumentar com o crescimento de usuários na rede (distribuição dos recursos entre as entidades da rede).
Tolerância à falhas	Reduzida, pelo fato da existência de um elemento centralizador (Servidor). A replicação de servidores pode significar um aumento de complexidade pelo fato de se ter que garantir a consistência dos mesmos.	Alta, devido ao fato da responsabilidade dos serviços oferecidos na rede entre suas entidades ser distribuída entre os *peers*.

Tabela 2.1: diferenças entre arquitetura P2P e Cliente-Servidor.

Neste momento, após a descrição das principais diferenças entre a tecnologia Cliente-Servidor e P2P, é importante ressaltar que estas tecnologias não são concorrentes e nem que uma surgiu para substituir a outra, pois, cada uma foi projetada com um foco de aplicação e portanto elas são necessárias em soluções específicas. Enquanto que a tecnologia Cliente-Servidor prioriza seus esforços na garantia de uma melhor gerência de seus serviços e recursos através da centralização dos mesmos (acarretando carências em outras áreas), a tecnologia P2P focaliza seus esforços na garantia de uma melhor distribuição destes serviços e recursos (ocasionando uma maior dificuldade de controle e gerenciamento). Desta forma, a vantagem de uma destas tecnologias supre a deficiência da outra, e com isto, pode-se dizer que elas tendem a conviver em paralelo, ou em conjunto, na busca de alcançar a combinação mais adequada de técnicas de compartilhamento e gerenciamento de recursos.

Um exemplo prático do contexto da afirmação acima, tem-se a proposta da *Internet* que é fornecer acesso irrestrito aos recursos computacionais disponíveis na rede

a seus usuários, porém sua infra-estrutura (baseado na arquitetura Cliente-Servidor) limita o nível desejado de acessibilidade destes recursos, gerando a necessidade de uma comunicação direta, máquina-a-máquina, entre os usuários da rede (aplicando assim o conceito P2P).

2.3 Paradigma da computação P2P

No campo da computação distribuída, a tecnologia P2P trouxe a mudança de paradigma relacionada aos modelos existentes de processamento distribuído e compartilhamento de recursos. Segundo Tsoumakos (TSOUMAKOS, 2003), este paradigma sugere o desenvolvimento de uma rede cooperativa em que seus membros, coletivamente, formam um sistema sem qualquer supervisão, podendo eles se comunicar e compartilhar seus recursos computacionais diretamente uns com os outros, independentes da necessidade de um gerenciamento central de serviços.

Segundo Wilson (WILSON, 2001), a idéia de se eliminar a oferta centralizada de serviços numa rede, ocasiona o surgimento do questionamento: como pode se conectar um conjunto de dispositivos de maneira que eles possam compartilhar informações, recursos e serviços? A solução de tal questionamento está associada com a organização da topologia da rede, aliada ao desenvolvimento do ambiente da comunicação P2P, o qual se baseia numa coleção de dispositivos interconectados num estilo de comunicação *ad-hoc*[7]. Nela, os processos de conexão e comunicação ocorrem dentro de um ambiente lógico (executado sobre uma infra-estrutura de uma rede física) formado pela ligação direta entre dispositivos com interesses comuns, que estão conectados através do mesmo sistema de comunicação, formando uma "teia" de dispositivos conhecida por comunidade P2P (ISHIKAWA, 2003).

As comunidades P2P criam suas arquiteturas com níveis mais altos de abstração da rede física subjacente, de modo que possibilitam uma interação direta entre seus componentes. Elas possuem ainda as seguintes características:

- os componentes são conectados de forma aleatória, não havendo restrição sobre o número de integrantes que participam da comunidade;

[7] Comunicação *Ad-hoc* é basicamente uma comunicação entre estações sem a necessidade de um ponto de acesso. Na essência, um componente da rede envia suas consultas para seus vizinhos conhecidos, e estes repassam adiante para outros conhecidos.

- a conexão de um componente à comunidade se estabelece através de outro integrante que já pertence à mesma;
- os componentes podem entrar e sair da comunidade a qualquer momento, sem prévio conhecimento dos demais participantes, sem causar danos ao bom funcionamento do sistema.

Partindo destas considerações, três modelos básicos de comunicação entre os dispositivos pertencentes a comunidades P2P passaram a ser implementados: Comunicação Ponto-a-Ponto, *Broadcast* e Multiponto (BROOKSHIER, 2002) (para exemplificação a seguir, toma-se como base uma rede IP).

A comunicação Ponto-a-Ponto é o modelo mais óbvio de comunicação entre componentes de uma comunidade P2P (que são conhecidos por *peers*[8]). Ela utiliza mecanismos, como por exemplo o HTTP, para estabelecer conexão entre dispositivos a partir de portas de comunicação abertas. Este modelo é considerado o mais usual para atividades que envolvem a necessidade da entrega de mensagens diretamente endereçadas para um *peer* específico da rede (BROOKSHIER, 2002).

No caso do estilo *Broadcast*, tal modelo de comunicação é mais utilizado entre *peers* pertencentes a sub-redes (envolve a participação de pelo menos uma entidade emissora e receptora). As mensagens ou consultas são enviadas apenas uma única vez para dentro da rede e os demais *peers* ouvem o meio. Apesar de ser um modelo econômico (por apenas uma única mensagem ser enviada), algumas destas mensagens são bloqueadas ou filtradas (devido adoção de políticas de segurança implementadas na rede física por meio de *firewalls*, "servidores *Proxy*"[9] e *NAT*), impedindo a comunicação direta entre os *peers.* A comunicação *Broadcast* é bastante aplicável nas seguintes situações: processo de descoberta dos *peers* ativos na rede; envio de mensagens idênticas para um grupo de *peers* (isto a um custo baixo de consumo da largura de banda); e entrega de consultas que podem ser respondidas por um grupo de *peers* (casos em que não se há conhecimento do *peer* que possui a resposta) (BROOKSHIER, 2002).

[8] Um *peer* é caracterizado por uma entidade de comunicação das redes P2P, que pode ser: dispositivo móvel, PDA, computador pessoal, um servidor ou uma estação de trabalho, ou qualquer variedade de dispositivos com perfil de comunicação (KATO, 2003).

[9] Servidor *Proxy* é um equipamento, posicionado entre a *Internet* pública e a rede local, que implementa serviços *Proxy*, como por exemplo: filtragem de *sites* indesejados e monitoramento de tráfego.

A comunicação Multiponto opera num estilo de comunicação semelhante ao utilizado no *Broadcast*. A diferença é que nela, múltiplas transmissões de uma mesma mensagem são enviadas para dentro da rede e/ou repassadas para outras redes. O modelo Multiponto é baseado em múltiplas conexões Ponto-a-Ponto, que podem ser utilizadas de diversas formas para criar transporte de mensagens (alguns *peers* podem especializar como *routers*, *gateway* ou controladores de mensagens) (BROOKSHIER, 2002).

O problema do modelo Multiponto é que ele não garante a entrega de mensagens simultâneas aos *peers* desejados e este comportamento faz com que o tempo de resposta de uma certa consulta (por exemplo: envio para um *peer* de outra rede) tenha características não-determinísticas, podendo ocorrer perda de mensagens durante a transmissão entre as redes (BROOKSHIER, 2002).

2.4 Topologias de rede utilizadas para prover a comunicação P2P

O termo topologia de rede na computação P2P refere-se à estrutura organizacional de *peers* que formam uma comunidade de dispositivos interconectados logicamente sobre uma rede física existente (KURMANOWYTSCH, 2003). A topologia é comparável a grafos com “nós” que se conectam através de vértices; conforme ilustrado na Figura 2.2.

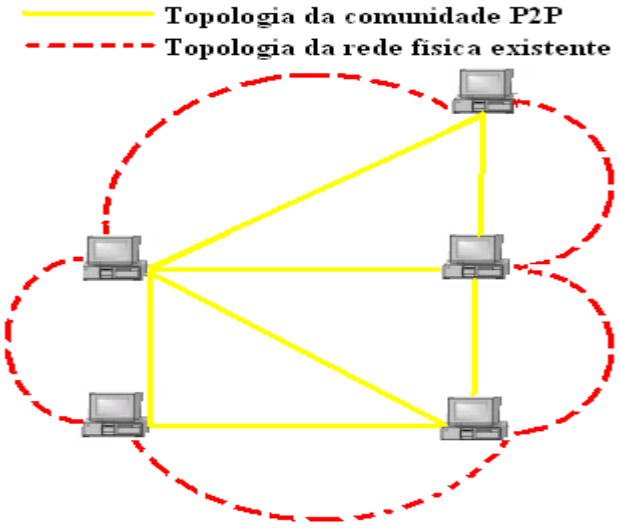


Figura 2.2: topologia de uma rede P2P.

Na Figura 2.2 é ilustrada a interligação de equipamentos numa rede física, bem como as ligações lógicas na rede P2P. A topologia P2P influencia significativamente o desempenho, a eficiência e a escalabilidade do sistema, pois, os fatores que afetam a comunicação P2P estão relacionados com a auto-organização da rede, que pode apresentar diversas características, como por exemplo, conectividade variável.

As redes P2P são tipicamente de natureza dinâmica e apresentam uma conectividade variável, pois, um *peer* pode ser adicionado e removido a qualquer momento, criando e destruindo as ligações entre os demais *peers* momentaneamente. Desta maneira, a organização dos *peers* não depende do fato destes estarem ligados ou não, e a comunidade P2P pode, de forma natural, organizar-se de acordo com o interesse dos próprios *peers* numa rede de grande escala de máquinas. Nesta rede, os *peers* podem possuir os mesmos privilégios que todos os outros já presentes, não havendo nenhum outro com autoridade maior que a dos demais. Assim, adicionar *peers* na rede não requer reorganização central. (CABAÇO, 2003).

Diante das características apresentadas, diferentes categorias de arquitetura P2P passam a existir, e diversos algoritmos de busca e roteamento de informações são implementados para tentar otimizar o encaminhamento de uma mensagem de um *peer* para outro. Segundo Kurmanowytsch (KURMANOWYTSCH, 2003), as topologias P2P mais comuns podem ser diferenciadas em duas categorias de arquitetura de rede: P2P Pura (conhecida também por Descentralizada) e P2P Híbrida. Por outro lado, os modelos de encaminhamento de mensagens mais comuns são centralizado e inundação.

2.4.1 Arquitetura P2P Pura ou Descentralizada

Uma rede P2P Pura é uma rede em que não há um ponto central e cada *peer* tem o mesmo nível funcional. Segundo Brookshier (BROOKSHIER, 2002), este tipo de arquitetura P2P é caracterizada por possuir somente entidades individuais que são capazes de se conectar e comunicar com qualquer outra entidade da rede; conforme está representado na Figura 2.3.

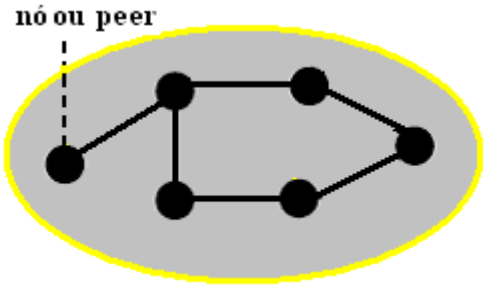


Figura 2.3: arquitetura P2P Pura ou Descentralizada.

Nesta arquitetura P2P, cada *peer* é uma entidade independente que pode participar e abandonar a rede, a qualquer momento, além de possuir total responsabilidade pela troca de recursos e gerenciamento. Eles podem se comunicar de maneira direta ou através de vizinhos comuns (o que causa melhor escala de informações de gerenciamento e pior escala de tráfego de rede). Desta forma, os *peers* utilizam mecanismos capazes de descobrir uns aos outros, propagando os trabalhos e as consultas sem a necessidade de uma autoridade central (GRADECKI, 2002).

A vantagem da arquitetura P2P Pura diz respeito ao fato da não existência de um ponto de falha único (não há um *peer* de controle central). Desta forma, a probabilidade da rede P2P sobreviver aumenta. Por outro lado, o principal problema existente neste tipo de arquitetura está relacionado ao processo de *bootstrapping*, pois quando o sistema é iniciado pela primeira vez numa máquina, pelo menos um *peer* de toda rede deve ser conhecido para efetivar sua conexão, tornando-o ativo na rede, do contrário ele não pode interagir com outro *peer* (KURMANOWYTSCH, 2003).

2.4.1.1 Modelo de Inundação

O modelo de Inundação de requisições não se baseia na publicação de recursos compartilhados na rede. Ao invés disso, cada requisição de um *peer* é enviada para todos os *peers* diretamente conectados, e assim sucessivamente até que a requisição seja respondida, ou que ocorra o número máximo de encaminhamento, conforme ilustrado na Figura 2.4 (DING, 2003).

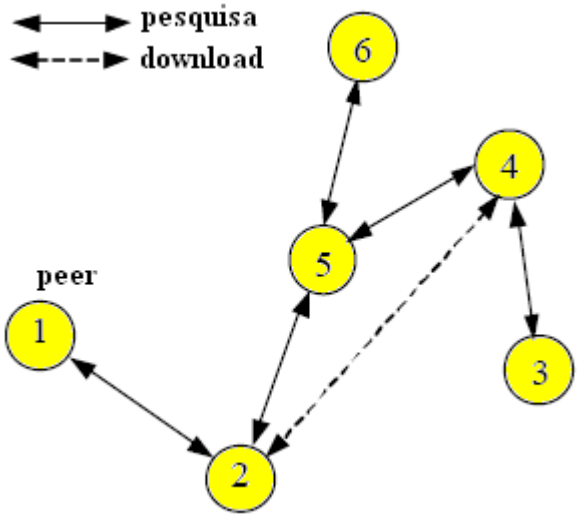


Figura 2.4: arquitetura do modelo de inundação.

Na Figura 2.4 está sendo ilustrado o uso do modelo de Inundação durante um processo de pesquisa e *download* de um arquivo numa rede P2P. Neste processo, o *peer* 2 envia uma mensagem de pesquisa aos *peers* 1 e 5, os quais, encaminham a mesma mensagem aos seus *peers* vizinhos, procedendo os mesmos da mesma forma. O *peer* 4 ao receber a mensagem de pesquisa, por meio do *peer* 5, confirma a posse do arquivo solicitado pelo *peer 2*, notificando ao mesmo sua disponibilidade em servir o arquivo por meio de uma nova mensagem diretamente enviada ao *peer* requisitante. Desta etapa em diante o processo de *download* é realizado sem a participação de outros *peers* (conforme está ilustrado na Figura - seta tracejada).

Para evitar o grande volume de uma mesma mensagem encaminhada na rede, o modelo de inundação trabalha com um valor que indica o número máximo de vezes que o pacote da mensagem pode ser propagado, o campo *TTL* (*Time-To-Live*). Cada *peer* que recebe o pacote de busca decrementa o contador interno e quando este chega ao valor zero, este pacote não é mais repassado a outro *peer*. Desta forma, o modelo de Inundação pode ser utilizado por uma rede com grande quantidade de "nós" (proporção de centenas e milhares) (GRADECKI, 2002).

Este modelo de encaminhamento de mensagens descrito é utilizado pelo *Gnutella*[10], e requer alta capacidade dos enlaces de comunicação para proporcionar bom

[10] *Gnutella* é um sistema P2P que objetiva o compartilhamento de arquivos entre usuários de sua comunidade (maiores informações na secção 2.5.1.2).

desempenho, e tende a ser um modelo eficiente em comunidades limitadas (quantidade pequena de integrantes) e em redes corporativas.

2.4.2 Arquitetura P2P Híbrida

Uma rede P2P Híbrida, também conhecida por semi-centralizada, é uma rede em que há diferença de relevância entre os *peers*, existindo um "nó" central para informações de controle, ou um conjunto de "nós", que assume tal função. Esta topologia é denominada de híbrida, pois a busca e a indexação de informações são centralizadas em *peers* de controle, enquanto que a transferência de arquivos e a comunicação ocorrem nas margens da rede, sem a necessidade de interferência de um "nó" central, conforme está ilustrado na Figura 2.5 (GRADECKI, 2002).

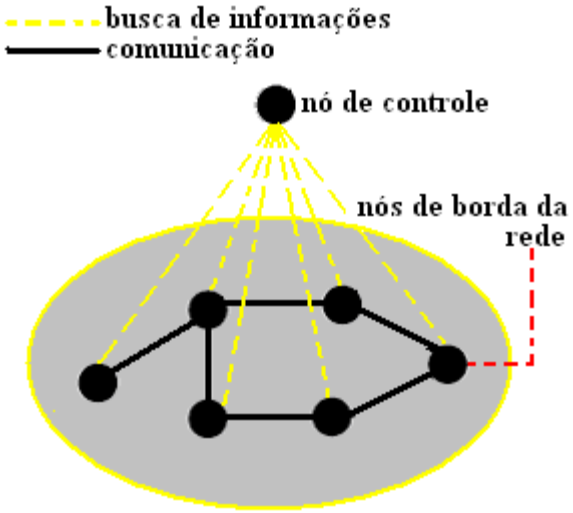


Figura 2.5: arquitetura P2P Híbrida.

Na arquitetura P2P Híbrida, os *peers* da rede se conectam a um servidor central para obter apenas informações vitais, como por exemplo, localização de outros *peers* e consulta a informações disponíveis na rede. Após uma pesquisa efetuada junto ao "nó" de controle, os "nós" de borda da rede (ou *peers edge*) passam a se comunicar diretamente. Este tipo de arquitetura geralmente utiliza o modelo de encaminhamento de mensagens centralizado.

A arquitetura P2P Híbrida combina a vantagem da arquitetura baseada em servidor (eficiência em gerenciar serviços da rede) com a idéia da arquitetura P2P Pura (em que não há ponto de falha única do sistema), o que traz benefícios, como por exemplo:

indexação central, permitindo a rápida localização de informações para *peers* solicitantes. Uma desvantagem relacionada a este tipo de topologia, diz respeito à perda de anonimato dos *peers*, pois o "nó" de controle acaba tendo o conhecimento de informações individuais dos outros *peers* da rede, além de que o mesmo pode ser visto como ponto de falha único, quando não existir outros "nós" de controle na rede (KURMANOWYTSCH, 2003).

2.4.2.1 Modelo Centralizado

O modelo Centralizado caracteriza-se pela conexão dos *peers* a um "nó" de controle (operando como um diretório central), em que eles publicam informações sobre o conteúdo que disponibilizam para a rede. Como está ilustrado na Figura 2.6, o "nó" de controle, ao receber uma requisição de *peers* da rede (1, 2, 3, 4 e 5 na Figura), escolhe o *peer* que melhor se adequar à resposta, e encaminha as informações do mesmo ao *peer* requisitante. Daí por diante, a comunicação e a troca de arquivos é realizada diretamente entre os dois *peers*, 4 e 5 na Figura (KURMANOWYTSCH, 2003).

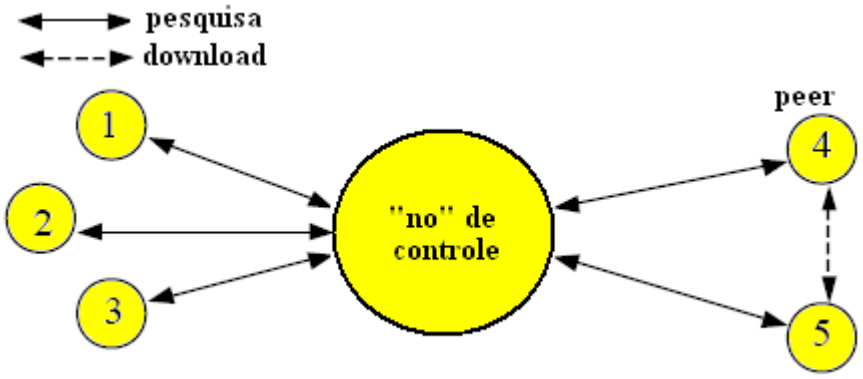


Figura 2.6: arquitetura do modelo centralizado.

A desvantagem do modelo centralizado é que podem ser gerados limites de escalabilidade, pois ele requer uma infra-estrutura de gerenciamento centralizado (o "nó" de controle). Isto pode ocasionar a necessidade de mais espaço de armazenamento, à medida que a quantidade de usuários cresce na rede. O *Napster*[11] é um exemplo de aplicação P2P que utiliza tal modelo.

2.5 Classificação dos sistemas baseados na arquitetura P2P

Segundo Ji (JI, 2003), as funcionalidades e características dominantes de sistemas P2P possibilitaram a classificação destes sistemas em quatro categorias principais de aplicação: compartilhamento de arquivos, sistema de comunicação e colaboração, plataforma P2P e sistemas de computação distribuída.

2.5.1 Compartilhamento de arquivos

A categoria de sistema de compartilhamento de arquivos pode ser considerada a de maior sucesso das redes P2P, pois, usa a idéia básica de compartilhamento de arquivo, aproveitando o espaço ocioso do disco de um dispositivo de rede, e a disponibilidade da largura de banda de uma rede para armazenamento, *download* e localização de arquivos (JI, 2003). Neste contexto, vários sistemas têm sido usados por usuários da *Internet* para burlar as limitações de largura de banda dos servidores que, em geral, impedem a transferência de arquivos grandes. Tais sistemas compreendem desde simples aplicações de compartilhamento direto de arquivos à aplicações mais sofisticadas, que criam um sistema de armazenamento distribuído, oferecendo as seguintes características:

- área de armazenamento – sistemas que fornecem ao usuário uma área potencialmente ilimitada para o armazenamento de arquivos;
- alta disponibilidade do conteúdo armazenado – para garantir alta disponibilidade dos arquivos armazenados, alguns sistemas adotam uma política de replicação múltipla do conteúdo, ou seja, um arquivo pode ser armazenado em mais de um "nó" na rede (STOICA, 2001);
- anonimato – alguns sistemas P2P garantem que o documento publicado será mantido no anonimato de seus autores e publicitários (WALDMAN, 2000);
- gerenciamento – a maioria dos sistemas P2P fornece mecanismos eficientes de localização e recuperação dos conteúdos armazenados na rede.

Atualmente, existem inúmeros sistemas que se enquadram nesta classificação de sistemas P2P, dentre os quais destacamos: *Napster* e *Gnutella* (ANDROUTSELLIS, 2005).

---

[11] *Napster* é um sistema P2P de busca de arquivos MP3 bastante popular na Internet (maiores informações consultar secção 2.5.1.1).

2.5.1.1 *Napster*

Napster (http://www.napster.com) é basicamente uma máquina de busca de arquivos MP3. Ela permite a procura e o compartilhamento de arquivos entre computadores interconectados à infra-estrutura da *Internet*. Tal sistema surgiu em 1999, e consiste numa aplicação P2P híbrida que, por sua vez, utiliza o modelo de encaminhamento de mensagens centralizado (consultar secção 2.4.2.1), no qual um servidor central é utilizado para armazenar uma lista com as músicas disponibilizadas pelos usuários. Neste sistema, todos os "*peers Napster*"[12] utilizam as informações obtidas no servidor para encontrar seus arquivos desejados, conforme está ilustrado na Figura 2.7 (GRADECKI, 2002).

Figura 2.7: arquitetura da comunidade *Napster*.

2.5.1.2 *Gnutella*

O *Gnutella* é considerado a primeira solução puramente P2P (http://www.gnutella.com). Diferente do *Napster*, não existe um servidor central que armazena informações sobre arquivos disponíveis na rede e, desta forma, seu sistema é caracterizado por uma topologia P2P Pura. Seus *peers* (identificados em sua comunidade por *servents*) atuam na rede tanto como clientes como servidores de informações. Para eles se conectarem na rede, precisam encontrar o endereço de um *servent* que já esteja conectado à mesma. Para tanto, eles devem utilizar endereços armazenados em uma *cache* local para enviar mensagem de requisição de conexão. Uma vez conectados, periodicamente os *servents* procuram por outros *servents*, através de consultas realizadas por vizinhos conhecidos (DING, 2003).

[12] *Peers Napster* é um programa cliente *Napster* instalado nos computadores de usuários da comunidade.

O mecanismo de localização e busca do *Gnutella* é baseado no modelo de encaminhamento de mensagem por Inundação (consultar secção 2.4.1.1). Desta forma, para procurar um arquivo na rede, um *servent* deve enviar uma mensagem de consulta para todos os seus vizinhos conectados.

2.5.2 Sistema de colaboração e comunicação

A classe de sistemas de colaboração e comunicação inclui soluções P2P que fornecem uma infra-estrutura que facilita a interação direta entre *peers* em tempo-real. Nesta classificação destacam-se aplicações de mensagens instantâneas, jogos *on-line* e programas de trabalhos colaborativos. Tais sistemas geralmente são implementados a partir de uma topologia híbrida (GRADECKI, 2002). Para exemplificar, tem-se o *ICQ Instant Messeger* (sistemas de mensagens instantâneas) e o *Groove* (programa de trabalho colaborativo).

2.5.2.1 ICQ Instant Messeger

A possibilidade de observar usuários se conectando na rede e de enviar mensagens em tempo real, tem tornado as aplicações de mensagens instantâneas uma das mais populares da *Internet*. Para exemplificar este tipo de aplicação, destaca-se o *ICQ Instant Messeger* (*ICQIM*). Sua primeira versão surgiu em 1996, fornecendo aos seus usuários uma forma rápida de comunicação e é o precursor dos programas de mensagens instantâneas atuais (http://www.icq.com/).

O *ICQIM* utiliza uma arquitetura P2P híbrida para prover seus serviços, possuindo basicamente um servidor central (“nó” de controle) para monitorar, quais usuários estão *on-line*, e fornecer aos mesmos uma lista de contatos individual, com um mecanismo capaz de identificar o estado dos usuários de sua lista, por exemplo: ativo, inativo ou ocupado. Já o processo de comunicação entre os usuários ocorre de maneira direta, sem a interferência do servidor central.

2.5.2.2 Groove

Segundo Brookshier (BROOKSHIER, 2002), a colaboração dentro de um grupo de trabalho envolve compartilhamento de idéias e recursos através de interações entre usuários com interesses comuns, visando uma melhor produtividade. Programas de trabalho colaborativo, ou *groupware*, podem ser definidos como programas que suportam colaboração, comunicação e coordenação de vários usuários em uma rede. Isto inclui a integração de características como correio eletrônico, calendário, espaços de trabalho, listas de discussão, sistemas de gerenciamento de documentos, vídeo conferência, entre outros (ROCHA, 2004).

Um exemplo típico de *groupware* é o *Groove* (http://www.groove.net). Ele utiliza espaços de trabalho que são criados e compartilhados entre participantes de um mesmo grupo e é baseado numa topologia P2P Híbrida para encontrar seus *peers* e iniciar seus serviços.

2.5.3 Computação distribuída

Segundo Wilson (WILSON, 2001), a computação distribuída é uma forma de resolver algum problema difícil, dividindo o mesmo em subproblemas que possam ser resolvidos por um grande número de computadores. A idéia básica é realizar grandes tarefas computacionais com o auxílio de recursos de máquinas ociosas.

Os sistemas de computação distribuída têm como objetivo delegar e migrar tarefas paralelas para diversos dispositivos espalhados pela rede, visando obter vantagem dos ciclos de processamento disponíveis por eles. Diante deste modelo de computação, diversas áreas da pesquisa científica ou mesmo aplicações comerciais que necessitam de processamento pesado podem se beneficiar desta abordagem, como por exemplo, o sistema *SETI@home*.

2.5.3.1 SETI@home

SETI@home (http://setiathome.ssl.berkeley.edu/) é um projeto de sistema de computação distribuída que é criado e é mantido pela Universidade de Berkley (Califórnia – EUA). Este sistema utiliza o tempo ocioso de máquinas na rede para realização de um *grid* computacional para analisar dados coletados, a partir de

telescópios, com o objetivo de encontrar algum sinal de rádio artificial vindo do espaço**.** O sistema possui um poder computacional de aproximadamente 25 TFlops/s (trilhões de operações de ponto flutuante por segundo), coletado de mais de três milhões de computadores conectados à Internet.

O sistema *SETI@home* inicialmente divide o problema computacional em partes pequenas e independentes. A seguir, cada computador na rede realiza alguma tarefa de processamento relacionado a um destes pequenos problemas, e os resultados obtidos são coletados por um servidor central (que é responsável também em distribuir as tarefas entre os computadores na Internet). Em cada computador registrado é instalado um *software* cliente que executa alguma computação requisitada pelo servidor. O processamento é realizado toda vez que o computador entra em período de inatividade (caracterizado pela ativação da proteção de tela). Após o término da computação o resultado é retornado ao servidor e uma nova tarefa é alocada para o cliente (WILSON, 2001).

2.5.4 Plataforma P2P

Uma plataforma P2P restringe-se à categoria de sistemas que objetivam fornecer um ambiente de programação para o desenvolvimento de aplicações P2P. Eles disponibilizam protocolos e métodos computacionais que facilitam a implementação da comunicação P2P. Esta classe de sistemas surgiu devido a maioria das soluções P2P serem proprietárias (normalmente operam com protocolos incompatíveis), resultando na impossibilidade de comunicação entre sistemas P2P (WILSON, 2001; ARORA, 2002).

Neste contexto, explorar as possibilidades da tecnologia P2P, a interoperabilidade e a capacidade de reuso de *software*, tornaram-se objetivos importantes a serem alcançados por projetistas desta área. Para tanto, eles precisavam de uma linguagem comum que permitisse que os *peers* se comunicassem e exercessem funcionalidades básicas de uma rede P2P. E nesta busca de normalização e redução dos esforços para tornar mais fácil o desenvolvimento de aplicações P2P, a *Sun Microsystems* cria o projeto JXTA, como sendo a primeira plataforma de programação para computação de rede distribuída P2P (GRADECKI, 2002).

O projeto JXTA não tem por objetivo definir o que é programação P2P, mas definir um conjunto base de especificações de protocolos que dêem suporte ao desenvolvimento de aplicações nesta área.

CAPÍTULO 3
Plataforma P2P JXTA

Diversas aplicações P2P atuais não utilizam nenhum padrão para serem desenvolvidas. Entretanto, já é possível utilizar algumas plataformas de desenvolvimento para iniciar a implementação de tais aplicações, como por exemplo JXTA.

Segundo Brookshier (BROOKSHIER, 2002), JXTA pode ser entendido como um *framework* que define um conjunto de especificações de protocolos (definidos por mensagens *XML*), com o propósito de criar uma plataforma comum de serviços e aplicações distribuídas P2P. JXTA não é uma aplicação e não define um tipo de aplicação a ser construída. Os protocolos especificados neste *framework* não são rigidamente definidos e, desta forma, suas funcionalidades podem ser expandidas para atender necessidades específicas de diversas aplicações P2P.

O conjunto de especificações JXTA (abreviação da palavra inglesa *jxtaposed*, trazendo a idéia de que a arquitetura P2P e Cliente-Servidor não são concorrentes, posicionam-se justapostas) manuseia um núcleo de funções que implementam a abstração de uma camada de rede virtual no topo de uma rede física existente, mascarando a complexidade da mesma (imposta por barreiras de proteção, como por exemplo: *firewall* e servidor *proxy*, que impedem a comunicação direta entre os dispositivos). Nesta rede virtual, qualquer *peer* pode interagir com outros *peers* de maneira descentralizada e independente de suas localizações (posicionada como *edges* ou atrás de *firewalls* e *NAT*). Além disto, esta rede virtual independe de sistema operacional e tecnologias de transporte de rede (JXTA, 2005).

Baseado na discussão exposta, o acesso aos recursos de rede deixa de ser limitado por incompatibilidade de plataforma, permitindo que dispositivos com característica de conectividade de rede estejam aptos a se comunicarem, colaborarem e compartilharem recursos da maneira definida por P2P.

A tecnologia JXTA funciona em qualquer dispositivo computacional de rede com capacidade de comunicação, incluindo aparelhos celulares, PDAs, sensores inteligentes, *desktops* e servidores.

Os objetivos do projeto JXTA são intencionalmente direcionados aos principais problemas encontrados na implementação de sistemas P2P, que são: interoperabilidade, independência de plataforma e ubiqüidade (BROOKSHIER, 2002; GRADECKI, 2002).

A interoperabilidade concerne na habilidade de múltiplos dispositivos computacionais serem capazes de interagir uns com os outros de forma eficiente, permitindo a comunicação e a troca de informações. A maioria dos sistemas P2P apresenta soluções proprietárias (conseqüentemente incompatíveis), impossibilitando a interconexão entre esses sistemas P2P.

O problema da independência de plataforma deve-se ao fato de que boa parte dos sistemas P2P desenvolvidos possuem alguma limitação relacionada à restrição de sua utilização, ou seja, são normalmente dependentes de linguagem de programação, sistema operacional e mecanismos de transporte de rede. Tudo isto leva ao contexto da impossibilidade da implementação de sistemas interoperáveis.

Por fim, o problema da necessidade de ubiqüidade dos sistemas P2P está relacionado com o fato de grande parte desses sistemas estarem limitados a plataformas de *hardware* homogêneas, o que acaba ocasionando a dependência de dispositivos para acesso ao mesmo.

A organização da arquitetura do projeto JXTA está modelada em três níveis de camadas funcionais, com o propósito de alcançar a solução dos problemas citados anteriormente. Sua organização pode ser vista como uma pilha de protocolos P2P, divididos em camadas sobrepostas, a fim de que cada camada implemente funcionalidades que possam ser utilizadas por camadas subseqüentes, em que procedimentos e funções complexos possam ser adicionados como serviço P2P (GRADECKI, 2002).

3.1 Arquitetura e Protocolos JXTA

Segundo Gong (GONG, 2001), durante a análise de diferentes sistemas P2P, a equipe de pesquisadores do projeto JXTA encontrou uma estrutura comum de camadas bem definidas em níveis conceituais para construção de sistemas clássicos P2P. Desta forma, a arquitetura JXTA passou a ser baseada nesta estrutura, sendo dividida em três camadas principais: camada de núcleo ou *core*, camada de serviços e camada de aplicações. Na Figura 3.1 é ilustrada a organização da arquitetura de sistemas JXTA.

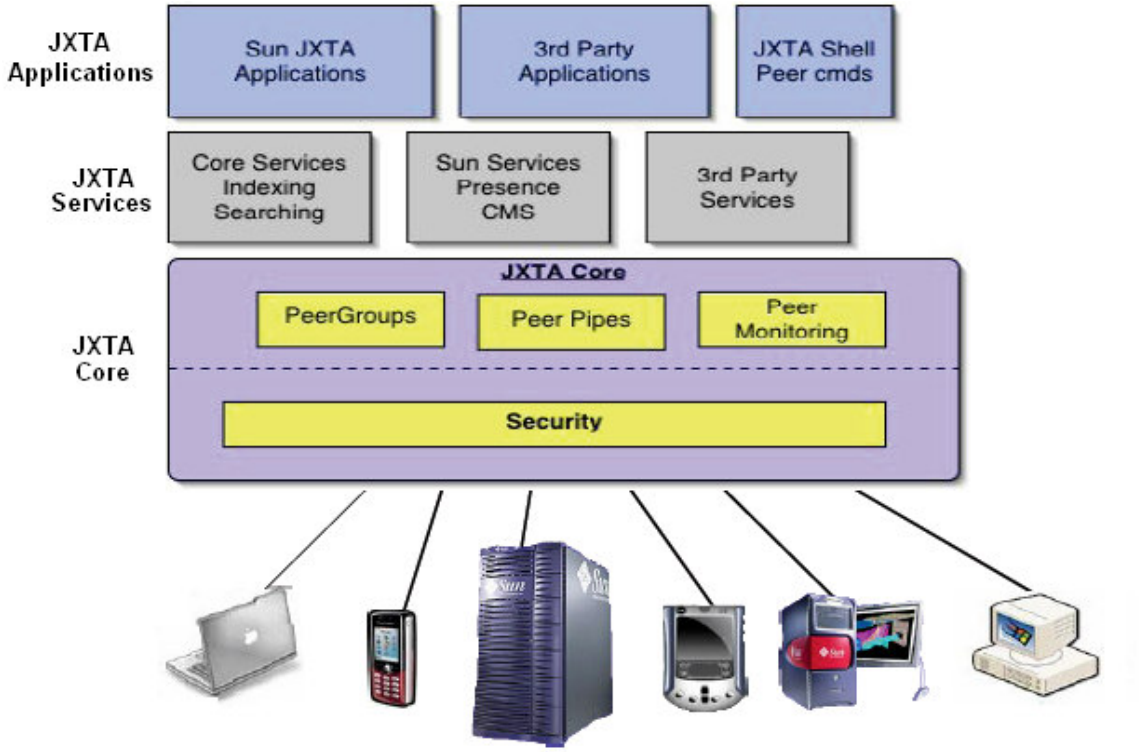


Figura 3.1: organização da arquitetura JXTA.

Conceitualmente, cada *peer* da plataforma JXTA abstrai as três camadas definidas. A primeira camada, conhecida por JXTA *Core*, é responsável por gerenciar os protocolos JXTA. Nela estão implementados os protocolos que disponibilizam o núcleo de suas funcionalidades, afim de alcançar uma infra-estrutura necessária para o desenvolvimento de qualquer sistema P2P.

A camada de serviços (JXTA *services*) contém as funcionalidades comuns que podem ser usadas como componentes em sistemas P2P, ou seja, ela provê funcionalidades semelhantes a de uma biblioteca que pode ser controlada por aplicações JXTA, através de lógica de programação numa aplicação. E finalmente, a camada de aplicação (JXTA *applications*), é onde a aplicação P2P reside propriamente. Ela utiliza

a camada de serviços para acessar a rede JXTA e suas utilidades, com o propósito de construir sistemas P2P colaborativos. No apêndice "A" podem ser encontradas informações mais detalhes sobre cada uma destas camadas.

Tomando por base a arquitetura de sistema apresentada acima, o projeto JXTA passa a ser composto por seis protocolos assíncronos (projetados especificamente para computação de rede P2P), definidos por uma representação textual XML e obedecendo ao modelo *query/response*. Os protocolos especificados são (JXTA SPEC, 2004):

- *Peer Resolver Protocol* (PRP) – usado para enviar consultas a qualquer número de outros *peers* e também para receber as respostas;
- *Peer Discovery Protocol* (PDP) – usado para anunciar e descobrir conteúdos na rede JXTA;
- *Peer Information Protocol* (PIP) – usado para obter informações de status de outros *peers*;
- *Pipe Binding Protocol* (PBP) – usado para criar endereço de comunicação entre *peers*;
- *Peer Endpoint Protocol* (PEP) – usado para encontrar rotas entre *peers*;
- *Rendezvous Protocol* (RVP) – usado para propagar mensagens na rede.

Na Figura 3.2 é ilustrado como os protocolos (destacados acima) estão empilhados para que possam permitir a colaboração de suas funcionalidades. Cada um deles endereça exatamente um aspecto fundamental da comunicação P2P, e a conversação entre eles é dividida dentro de uma porção conduzida por um *peer* local e outra porção por um *peer* remoto. A porção do protocolo referente ao *peer* local é responsável pela geração de mensagens (ou consultas) e envio para o *peer* remoto. Já a porção referente ao *peer* remoto é responsável pelo manuseio de mensagens recebidas e pelo seu processamento para execução de alguma tarefa. Desta forma, os protocolos JXTA possuem a característica de serem semi-independentes um do outro. Portanto, em um *peer* pode estar implementado apenas um subconjunto dos protocolos para atender suas funcionalidades desejadas (WILSON, 2001).

O conjunto de protocolos definidos na plataforma JXTA possibilita a criação de aplicações interoperáveis, promovendo a revolução da tecnologia P2P na Internet, ao padronizar a maneira com que *peers* descobrem e monitoram uns aos outros, auto-

organizam-se em "grupos de *peers*"[13], e anunciam e descobrem recursos existentes na rede, tudo isto a partir de um modelo de comunicação P2P (HAJAMOHIDEEN, 2003; JXTA SPEC, 2004).

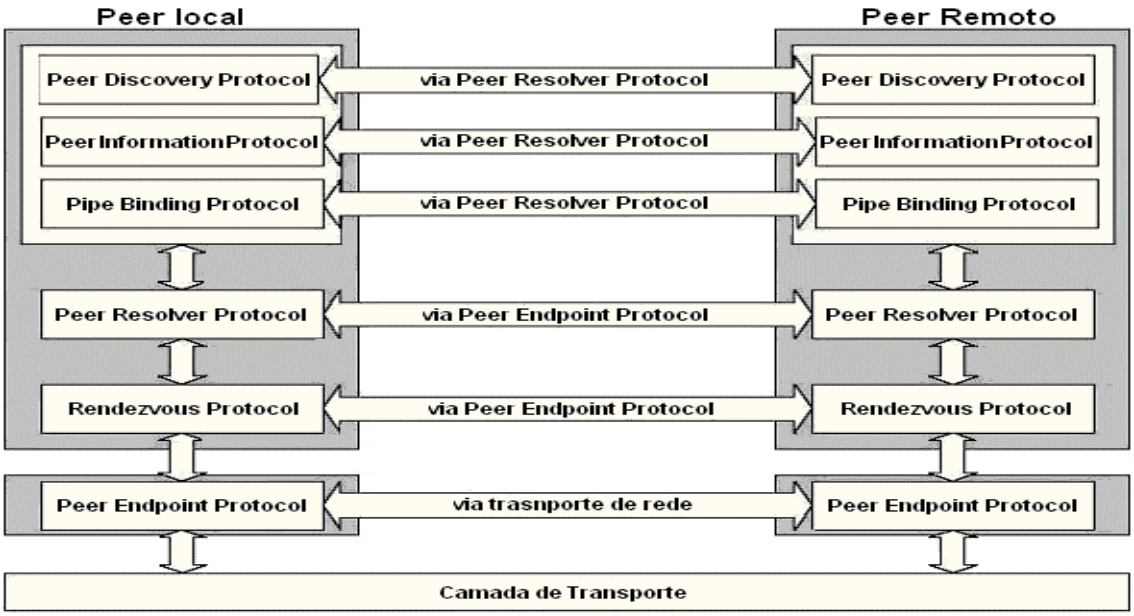


Figura 3.2: pilha de protocolos JXTA.

Segundo Gradecki (GRADECKI, 2002), todos os protocolos ilustrados na Figura 3.2 são necessários para que um *peer* possa operar autonomamente numa rede virtual JXTA auto-organizada, ou seja, opere sem a necessidade da presença de um servidor de controle. Um *peer* não é obrigado a implementar todos os protocolos ilustrados, porém faz-se necessário o *Peer Endpoint Protocol* e o *Peer Resolver Protocol*, pois, são essenciais para que um *peer* seja endereçável e capaz de estabelecer comunicação dentro de uma comunidade P2P JXTA (o apêndice "B" contém mais informações sobre cada um dos protocolos identificados na Figura).

3.2 Rede JXTA e seus principais componentes

A rede JXTA é uma rede *ad-hoc*[14], *multi-hop*[15] e *pervasive*[16], composta de uma série de *nodes* ou *peers* interconectados, conforme ilustrado na Figura 3.3. Suas

[13] Grupos de *peers* são *peers* que possuem funções similares sob um conjunto unificado de regras e restrições.
[14] Rede *Ad-hoc* é uma rede sem infra-estrutura e com controle descentralizado, na qual seus "nós" podem entrar e sair a qualquer momento.
[15] Rede *Multihop* é uma rede que possui sua arquitetura caracterizada pela interconexão entre quaisquer pares de seus "nós". Ela representa uma estratégia para expandir a área de cobertura total da rede, pois aumenta o número de rotas disponíveis para um "nó" específico.

conexões de rede podem ser transientes, e as rotas entre *peers* são não-determinísticas, ou seja, *peers* podem entrar ou abandonar a rede em qualquer momento, fazendo com que as rotas se alterem freqüentemente (JXTA PROGUIDE, 2003).

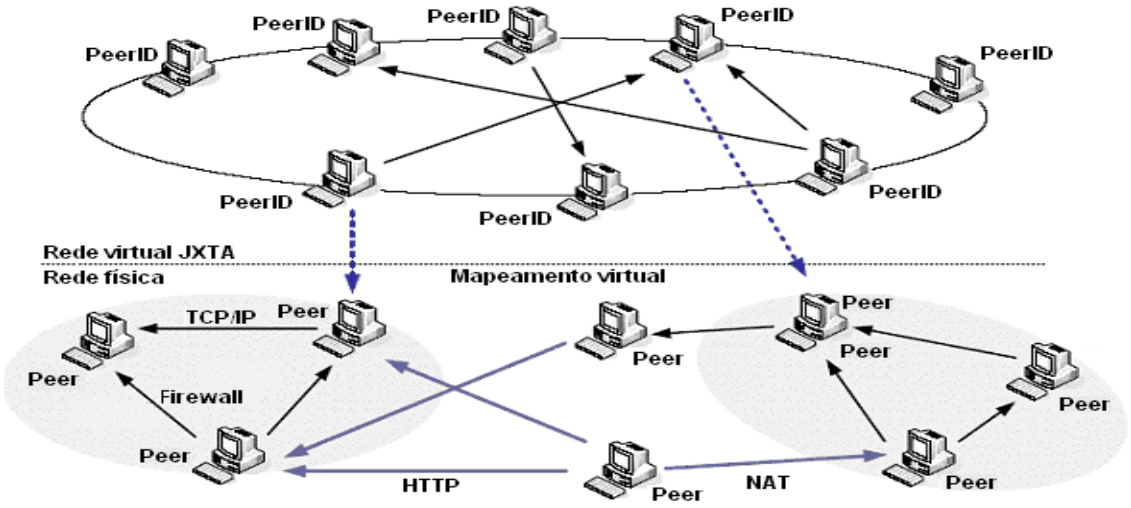


Figura 3.3: arquitetura da rede virtual JXTA.

Na Figura 3.3 é mostrado o principal propósito da rede JXTA, que é ocultar toda complexidade da topologia de rede física, por meio de um mapeamento virtual acima da infra-estrutura da rede existente. Isto é realizado com o intuito de alcançar uma espécie de "nuvem virtual JXTA"[17], em que serviços e aplicações possam ser implementados a partir de funcionalidades dos protocolos JXTA. Nesta nuvem virtual, cada *peer* é associado a um identificador único (*peerID*) e passa a operar em grupos de *peers*. Desta forma, o projeto da rede JXTA permite a comunicação entre *peers* sem a necessidade dos mesmos entenderem ou gerenciarem a complexa e dinâmica modificação da topologia de rede física. Tal ambiente fornece um endereçamento de trabalho uniforme para todos os *peers* da rede, com o intuito de manter facilmente a interoperabilidade entre diferentes sistemas P2P e comunidades (GRADECKI, 2002; WILSON, 2001).

Segundo Traversat (TRAVERSAT, 2003), a tecnologia JXTA define algumas abstrações de rede que são fundamentais para serem usadas em seus serviços e aplicações. Tais abstrações estão relacionadas no Tabela 3.1.

---

[16] Rede *pervasive* é uma rede que permite o acesso rápido e conveniente às informações, a qualquer momento e de qualquer lugar. Por exemplo: a partir de um *handheld* ou PDA, ou mesmo de um celular, é possível acessar dados de uma rede de qualquer lugar do mundo.

[17] Nuvem virtual JXTA representa a idéia de diversos conjuntos de *peers* que cooperam num grande espaço virtual JXTA (semelhante a nuvens dispostas no céu).

Advertisement	Documentos XML que descrevem os atributos de componentes da rede JXTA (*peer*, *peer group*, *pipe*, *service*).
Message JXTA	Objeto que é enviado entre *peers* JXTA, que pode ser representado tanto no formato binário ou XML. Caracteriza um volume de dados que guardam conteúdo de roteamento, dados credenciais e tipos de conteúdo de informações gerais, transmitido via *pipes*.
Peer JXTA	Qualquer entidade de rede que implementa um ou mais protocolos JXTA (contém informação de um usuário da rede). Eles são caracterizados em três tipos básicos: *edge*, *rendezvous* e *relay* (consultar apêndice C).
Pipes	Canal de comunicação virtual usado para estabelecer uma conexão de comunicação entre *peers*.
Peer Group	Coleção de *peers* que cooperam fornecendo um conjunto comum de serviços e recursos, formando uma partição lógica de rede.
Service	Recurso de rede disponível de acesso compartilhado por *peers* dentro de *um peer group*.

Tabela 3.1: principais componentes da tecnologia JXTA.

Os componentes apresentados na Tabela 3.1 estão descritos com maiores detalhes no apêndice C. Tais componentes levam a um melhor entendimento na formação do ambiente de comunicação JXTA, que nada mais é do que uma série de *peers* interconectados que se auto-organizam em *peer groups*, fornecendo um conjunto comum de serviços. Tais *peers* anunciam seus serviços e recursos através de documentos XML (conhecido por *advertisement*), possibilitando outros *peers* da rede saberem como se conectar e interagir com eles (JXTA PROGUIDE, 2003).

O sistema, descrito anteriormente, é suportado por um mecanismo transferidor assíncrono e unidirecional de *message* JXTA, usado no serviço de comunicação entre *peers*. Este serviço de comunicação baseia-se no esquema de estabelecimento de conexão direta ponto-a-ponto entre *peers* (envolvendo os *pipes* de comunicação).

Os *pipes* são construídos a partir de *endpoints*[18], que são especificados de acordo com o mecanismo de transporte da rede, como por exemplo, a associação de uma porta *TCP* a um endereço *IP*. Desta forma, cada *peer* deve publicar uma ou mais interfaces de rede, anunciadas como um *endpoint*, por meio de *advertisement* (JXTA PROGUIDE, 2003). Dentro deste contexto, um *peer* ao receber um *advertisement* de um *endpoint* de outro *peer*, pode selecionar a maneira mais eficiente para se comunicar com o mesmo, a partir da lista de endereços disponíveis para a formação de *pipes*. Isto possibilita a interação e o compartilhamento de recursos entre eles, como por exemplo, usando TCP/IP diretamente quando possível, ou HTTP sobre TCP/IP, para transpor *firewall* ou *NAT*.

O esquema de endereçamento da rede JXTA é baseado em identificadores únicos, conhecidos por "*JXTA ID*"[19]. Neste esquema, cada *peer* é identificado unicamente por um *JXTA ID*, que permite que o *peer* encapsule não só o transporte físico, mas também protocolos de transferência lógica, como por exemplo, HTTP. Desta forma, o *peerID* (tipo de *JXTA ID*) é usado para tornar transparente todos os meios possíveis em que um *peer* possa se comunicar (TRAVERSAT, 2002; JXTA PROGUIDE, 2003).

No apêndice C podem ser encontradas maiores informações sobre *endpoint* e *IDs* JXTA.

3.3 Comunicação na Rede JXTA

A comunicação é o problema principal de qualquer rede P2P, pois, faz-se necessário realizar a troca de serviços entre os dispositivos conectados na rede. Segundo Wilson (WILSON, 2001), para que esta comunicação possa existir, é fundamental saber como encontrar um *peer* e seus serviços, além de saber como um *peer* pertencente a uma rede privada deve interagir com a rede P2P. Como já mencionado anteriormente, o sistema de comunicação da rede JXTA é baseado na publicação de *advertisements* de recursos disponíveis na rede. Desta forma, um dos pontos cruciais para comunicação está relacionado com a localização destes *advertisements* publicados. A plataforma

[18] *Endpoint* é uma abstração de componente da rede JXTA que encapsula todo endereço de interface de rede física disponível para um *peer*. Eles são semelhantes a cartões de visita, listando diversos meios possíveis que se pode obter contato com uma pessoa (telefone, celular, fax, e-mail, e outros).

[19] JXTA ID são identificadores utilizados para diferenciar cada recurso existente na rede JXTA, incluindo *peers*, *pipes*, *peer groups* e *advertisements* (consultar apêndice C – JXTA ID).

JXTA considera três escopos básicos de descoberta: Local ou descoberta em *cahce*, Direta e Propagada ou descoberta indireta (GRADECKI, 2002).

A primeira considera o uso de técnica de descoberta passiva, pois não envolve conectividade na rede. As duas outras envolvem conexão na rede, e por isto consideram o uso de técnicas de descoberta ativa.

3.3.1 Local ou Descoberta em *cache*

A descoberta em *cahe* é a forma mais simples e rápida para encontrar *advertisements*, pois parte do princípio que *peers* JXTA possuem uma espécie de páginas amarelas, ou seja, baseia-se em *advertisements* disponíveis em *cache* local, fornecendo informações sobre outros *peers* ou recursos da rede conhecidos. Desta forma, este método elimina totalmente o processo de descoberta, o que diminui o tráfego na rede.

Na Figura 3.4 é ilustrada a arquitetura da descoberta em *cache*. Pode-se ver nesta Figura que, quando um *peer* iniciar a procura por *advertisements*, ele faz primeiro uma consulta em seu *cache* local para descobrir os endereço de *peers*. Caso o resultado da procura seja positivo, o *peer* passa a possuir a opção de conexão com os *peers* listados a partir de informações de sua *cahe;* caso contrário, uma descoberta remota deve ser ativada.

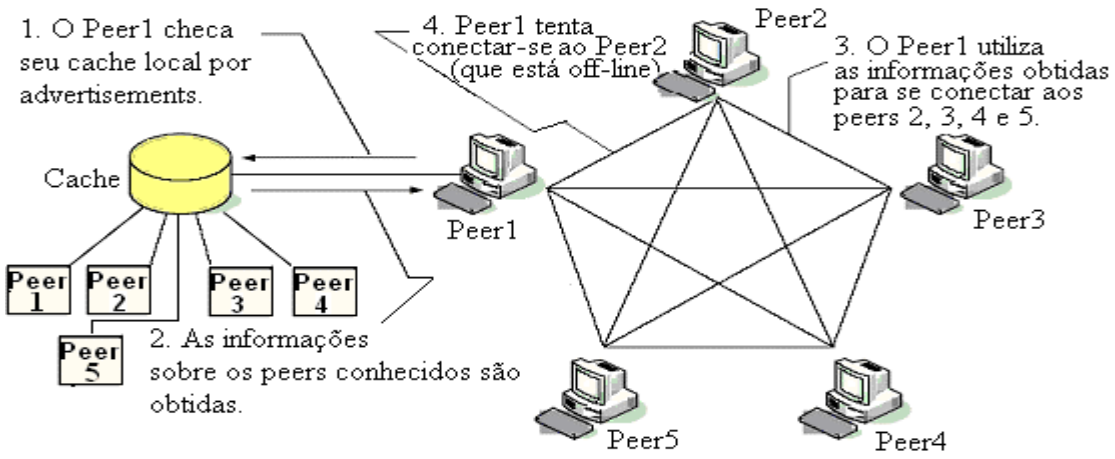


Figura 3.4: descoberta de *peers* usando busca de *advertisement* em *cache*.

Segundo Wilson (WILSON, 2001), o principal problema deste método está relacionado à possibilidade de referenciar um *peer* ou serviço que não esteja mais disponível na rede, causando um tráfego desnecessário na mesma. Para resolver este

problema é necessário estipular um tempo de vida para cada *advertisement*, desta forma pode-se remover as referências desatualizadas da lista em *cache* e atualizá-las constantemente através de algum mecanismo de descoberta remota.

3.3.2 Descoberta Direta

O processo de descoberta de *advertisements* JXTA pode ser realizado autonomamente a partir de suas conexões de rede, ou seja, cada *peer* é capaz de descobrir recursos da rede por meio de uma técnica de descoberta baseada no modelo de inundação (consultar Secção 2.4.1.1). A especificação JXTA permite a descoberta direta através do uso de *broadcast* e *multicast*, que são formas nativas da sua rede de transporte, no caso TCP/IP. Desta forma, os *peers* podem enviar requisições de descoberta ao longo da rede por meio de mensagem *broadcast* com critérios de procura que podem ser por exemplo, nome de *pipes* ou *JXTA ID*. Neste caso, todos os *peers* de uma rede local recebem a mensagem e respondem apropriadamente (GRADECKI, 2002).

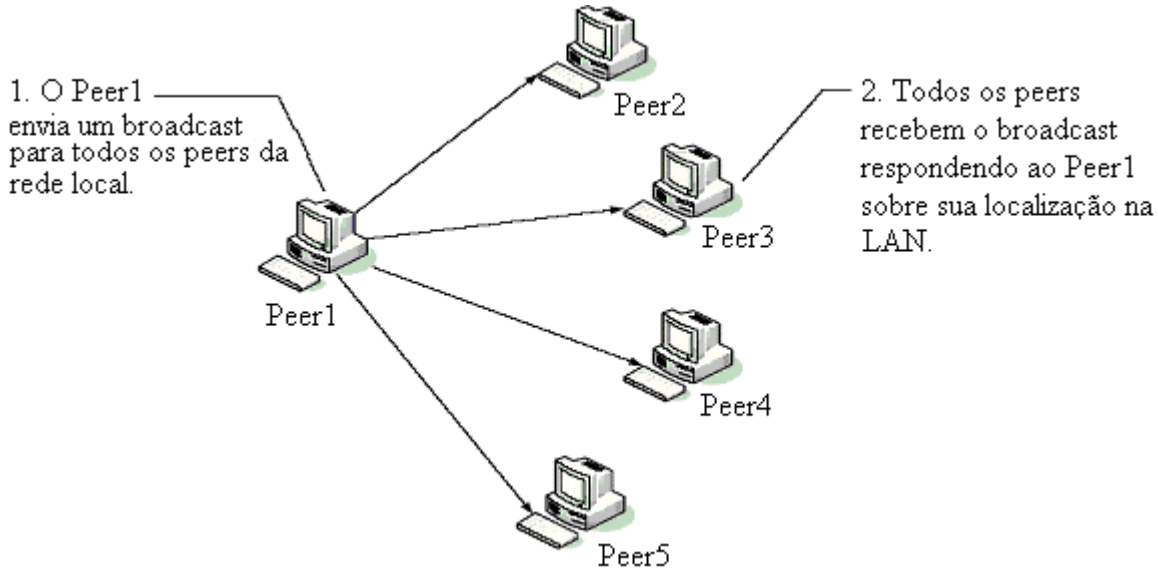


Figura 3.5: exemplo de descoberta direta.

Segundo Wilson (WILSON, 2001), a desvantagem deste processo é que ele é limitado em ser utilizado apenas em redes locais. Para a comunicação com outros *peers* fora da rede local é necessário utilizar a descoberta indireta.

3.3.3 Descoberta indireta ou propagada

A descoberta indireta é uma técnica que requer o uso de um *peer Rendezvous* para localizar *advertisements* fora da rede local. O *peer rendezvous* funciona como uma fonte de informações sobre *advertisements* de *peers*, além de ser utilizado como intermediário para propagar mensagens de descoberta. Desta forma, este procedimento caracteriza-se como uma técnica de descoberta baseada no modelo centralizado (consultar secção 2.4.2.1).

Segundo Wilson (WILSON, 2001), o *peer rendezvous* opera em dois modos possíveis para localização de *advertisements*: propagação e por *advertisements* armazenados. No modo de propagação, o *peer rendezvous* propaga uma requisição de descoberta para outros *peers* na rede JXTA que ele conhece, incluindo outros *peers rendezvous*, os quais também propagam a requisição para outros *peers*. Já no estilo de localização baseado em a*dvertisement* armazenados, o *rendezvous* usa sua *cache* local para responder a requisições de descoberta solicitadas por outros *peers*.

Para uma solução eficaz de busca de recursos numa rede JXTA, é necessário que o *peer rendezvous* utilize as duas formas citadas para armazenar *advertisements* e servir a uma grande quantidade de *peers* (WILSON, 2001).

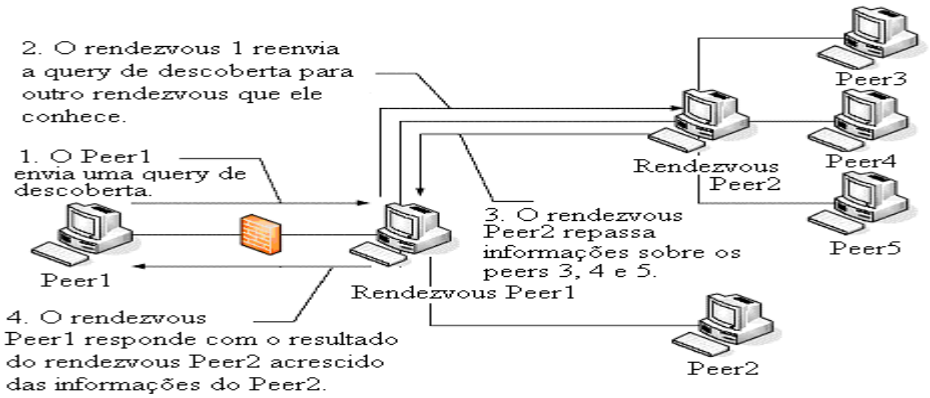


Figura 3.6: exemplo de descoberta indireta.

Um dos problemas encontrados na descoberta indireta é que os *peers rendezvous* podem propagar indefinidamente seus pedidos, criando um tráfego desnecessário na rede. Para solucionar este problema é necessário estabelecer um tempo de vida para os pedidos, desta forma, os mesmos são descartados após circular por um determinado tempo na rede (WILSON, 2001).

CAPÍTULO 4

Abordagem do ambiente de desenvolvimento de soluções computacionais na área de Telemedicina

A Telemedicina propõe o deslocamento de serviços de assistência médica especializada para regiões desprovidas da mesma, através do uso de tecnologias de informação e comunicação. Sua prática traz o desafio estratégico de beneficiar pacientes em suas próprias localidades ao possibilitar maior acesso e diminuição dos custos na oferta de serviços médicos para: prevenção, diagnóstico e tratamento de doenças, além de possibilitar a educação à distância de profissionais de saúde (LOPES, 2004; HERTZOG, 2005).

Atualmente, a *Internet* é o "palco" principal de atuação em que se desenvolvem soluções para a área de Telemedicina, devido a sua abrangência global e por ela ser alvo estratégico de convergência de dados entre diversas tecnologias de comunicação, como por exemplo, redes de telefonia móvel, *ISDN* (*Integrated Services Digital Network*), dentre outras. Isto nos traz a perspectiva de desenvolvimento de novas soluções computacionais para a área de assistência médica à distância, incluindo o desenvolvimento de redes interativas para programas educacionais de ensino, além de soluções que se baseiem na idéia de facilitar o acesso e a representação de informações de pacientes no ambiente que abrange a prestação do serviço médico.

4.1 Soluções computacionais aplicadas na Telemedicina

As soluções computacionais aplicadas na área de Telemedicina são classificadas basicamente em dois grandes grupos: *Store and Forward* e Sistemas de tempo real.

O grupo *Store and Forward* caracteriza-se por aplicações que consistem no armazenamento e envio de informações sobre pacientes entre dois profissionais de saúde, através de um estilo de comunicação assíncrona. Esta classificação, caracteriza aplicações que são tipicamente utilizadas em situações formais (de não emergência), quando o diagnóstico ou consulta não necessita ser realizado imediatamente.

O grupo de sistemas de tempo real caracteriza-se por aplicações que consistem na comunicação em tempo real entre dois ou mais intervenientes na prática clínica, sendo

necessário um estilo de comunicação síncrona. É utilizada quando uma consulta fim-a-fim é necessária entre médicos localizados em regiões geográficas distintas, possibilitando colaboração direta por meio do fornecimento de opinião ou diagnóstico remoto imediato (LAMMINEN, 2001; SEABRA, 2003).

A partir da classificação apresentada, observa-se que a prática da assistência médica é inerentemente colaborativa, pois, existe a necessidade de intercâmbio contínuo de informações entre grupos de profissionais para se obter diagnósticos clínicos precisos. Desta forma, sistemas computacionais devem possuir pelo menos como base estratégica de engenharia de *software*, o conceito de descentralização de conhecimento e de fornecimento de serviços (conceitos pertencentes a sistemas de natureza distribuída P2P), pois, tanto o conhecimento como os serviços de assistência médica estão distribuídos entre médicos e centros de saúde, e não centralizados. Por outro lado, esta idéia não tem sido bem considerada na prática, pois, estudos relacionados a sistemas computacionais têm revelado que a tecnologia Cliente-Servidor tem sido convencionalmente utilizada nesta área. Para exemplificar, destacam-se os PACS (*Picture Archiving and Comunications System*) ou sistemas de arquivamento e comunicação de imagens (BARDRAM, 2003).

Os PACS são sistemas que proporcionam ambientes distribuídos de trabalho para transmissão, processamento e visualização de imagens médicas. Alguns deles integram PEPs[20] sobre alguns "padrões"[21] abertos de representação de informação, como por exemplo: CORBAmed[22], DICOM[23], HL7[24] e XML. Tais sistemas estabelecem um gerencimaneto centralizado de imagens capturadas por "modalidades"[25] que são armazenadas localmente em depósitos de dados pertencentes a intranets e disposnibilizadas para clientes *web* (LOPES, 2004; Hira, 2002).

[20] PEP (Prontuário Eletrônico de Pacientes) é um registro digital sobre informações de um paciente, incluindo consultas, diagnósticos, laudos, procedimentos, prescrições, cirurgias e entre outros dados relevantes pana análise clínica.
[21] Padrão pode ser definido como um "objeto" de *software* que pode ser usado por diversos sistemas que desejam "conversar" entre si.
[22] CORBAmed (*Commom Request Broker Architecture - Medical*) é um organização internacional que faz parte da OMG (*Object Management Group*) cuja função é especificar interfaces padronizadas de serviços na área de saúde, ou seja, tenta trazer um consenso sobre a definição de interfaces orientadas a objetos entre serviços e funções médicas inter-relacionadas para promover a interoperabilidade entre plataformas.
[23] DICOM (*Digital Imaging and Communication in Medicine*) define a forma de efetuar o armazenamento e transmissão de imagens médicas.
[24] HL7 define normas para o intercâmbio eletrônico de informação clínica, financeira e administrativa entre serviços de saúde independentes orientado por computador, como por exemplo: sistemas de informação de hospitais, clínicas e laboratórios.
[25] Exemplo de modalidades de captura de imagens médicas: tomografia computadorizada, ressonância magnética e ultra-sonografia.

O acesso, visualização e processamento de imagens pertencentes a PACS são geralmente realizados através de aplicações baseadas na tecnologia "*Applet Java*"[26], tornando estas soluções dependentes de servidores de imagens e de filtros de processamento, além de limitá-las ao uso de *web browsers*. Todos estes fatores implicam também na impossibilidade de intercâmbio direto de imagens entre médicos, pois qualquer atividade de intercambio de informação requer a interação com o servidor central de imagens (KHLUDOV, 2000; JAIN, 1997).

Outro exemplo típico da má utilização da tecnologia Cliente-Servidor, na área de Telemedicina, está relacionado com aplicações de "segunda opinião médica" e de comunidades virtuais. Nestas áreas, grupos de profissionais de saúde geralmente utilizam ferramentas *web* sobre uma infra-estrutura Cliente-Servidor, como por exemplo, *Applets* e *Servlets Java*, ASP, CGIs e CORBA, para discutir protocolos de tratamento, temas e casos clínicos, ou elaborar e publicar eletronicamente conteúdo científico que reflita suas experiências e opiniões técnicas (SEABRA, 2003; SABBATINI, 1999).

O uso das tecnologias descritas acima no desenvolvimento de aplicações para as áreas referenciadas, normalmente gera soluções inviáveis, ou com funcionalidades limitadas, devido a:

- necessidade de uma grande base de dados para armazenamento de informações (causando elevado custo);
- manutenção contínua do sistema, pois, requer atualizações constantes no *layout* e conteúdo da página *web* onde é oferecido o serviço;
- solução avançada para busca de conteúdos, isto no caso das comunidades virtuais, além de um "alto nível de colaboração"[27] entre médicos para soluções de "segunda opinião", constituindo assim, problemas com soluções viáveis e facilmente resolvidas por meio da arquitetura de comunicação P2P.

[26] *Applet* Java é uma pequena aplicação Java que é executado em *web browser*.

[27] Alto nível de colaboração é aqui entendido por atividades que resultem no fornecimento de dados em tempo-real para colaboradores de um mesmo trabalho, como por exemplo, permitir uma conversação e o fornecimento de dados ou arquivos entre um grupo de médicos com localizações geográficas distintas. Tudo isso a nível de interação presencial como se eles estivessem num único ambiente de trabalho.

4.1.1 Exemplo de aplicação de análise médica

MedJava é um sistema típico *Store and Forward*, desenvolvido na Universidade de Washington (EUA) com a finalidade de implementar uma solução completa na área de sistemas distribuídos de imagem médica eletrônica, ou seja, que possibilite o processamento e a visualização remota de imagens para análise de diagnósticos. Para alcançar tal solução, o sistema MedJava é desenvolvido sob uma arquitetura de serviço Cliente-Servidor baseada em *Applet* Java, possibilitando a execução de suas funcionalidades a partir de navegadores *web*. Na Figura 4.1, é ilustrada a topologia deste tipo de solução (JAIN, 1998).

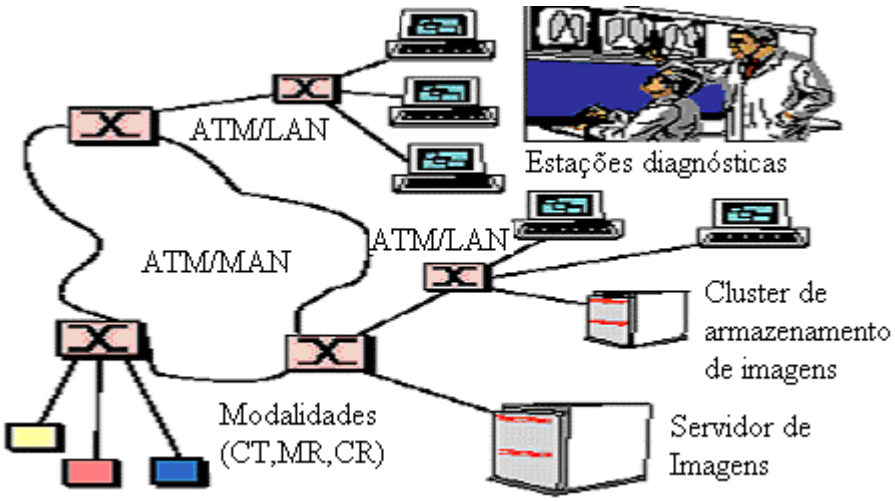


Figura 4.1: topologia do sistema MedJava.

No ambiente ilustrado na Figura 4.1, imagens médicas são capturadas por modalidades e armazenadas em servidores de imagens. Neste momento, radiologistas e médicos podem fazer requisição destas imagens (identificando suas URLs) por meio de estações de trabalho para realização de diagnóstico remoto. Desta forma, a arquitetura proposta pelo sistema MedJava fornece aos seus usuários transferência de imagens sobre uma rede de comunicação, além de processamento e visualização das mesmas, porém pode oferecer mais eficiência na disponibilidade de seus serviços através de uma arquitetura de comunicação P2P.

Por meio do modelo de comunicação P2P, o processo de busca de imagens na solução MedJava não estaria mais limitado a identificação de uma URL pertencente a uma máquina, e sim baseada na busca de arquivos entre as máquinas da rede, e o armazenamento das imagens poderia ser transparente e expansível através do uso do

espaço de armazenamento das estações de trabalho ou diagnósticas (ver Figura 4.1), além de que haveria a possibilidade de interação direta entre dois ou mais especialistas para composição de um diagnóstico preciso. A solução MedJava com esta nova modalidade de comunicação, alcançaria os seguintes benefícios:

- diminuição dos custos na aquisição de equipamentos para armazenamento das imagens (aproveitamento do espaço de disco ocioso das estações diagnósticas);
- maior agilidade no processo de localização e distribuição de imagens (sistema de busca baseada nas imagens e não em URL, além de não depender de servidor central de imagens para atender as requisições);
- interatividade no processo de diagnóstico (possibilitar a interação, em tempo real, entre diversos especialistas).

Dentro do contexto acima, destaca-se o uso da tecnologia JXTA para dar suporte ao desenvolvimento do modelo de comunicação P2P referenciado.

4.2 Cenário de aplicações P2P na área de Telemedicina

Ao considerar o uso da arquitetura de computação P2P na área de Telemedicina, agrega-se ao ambiente de desenvolvimento de soluções computacionais desta área o contexto de sistema de saúde distribuído ou *pervasive healthcare* (KORHONEN, 2005). Ele traz o caráter de descentralização à prática de assistência médica, promovendo uma maior acessibilidade a quem a necessite, dentro de uma rede global de comunicação existente. Desta forma, independente da origem de oferta do serviço médico e do local que o solicite, o serviço está disponível por meio de uma interação direta entre médicos, portanto, equipamentos computacionais, que trocam informações entre si, transferindo conhecimentos específicos para auxiliar a prática de assistência médica.

Com a possibilidade de interação direta entre os profissionais de saúde, novas soluções colaborativas podem ser desenvolvidas, tendo como resultado propostas de sistemas ideais para auxiliar a prática de assistência médica à distância. Estes profissionais podem adquirir uma mobilidade de comunicação ajustável a seus ambientes de trabalho (dentro e fora de hospitais), mantendo acesso contínuo a recursos que contribuem com o sucesso de suas atividades, por meio de um estilo de comunicação P2P. Na Figura 4.2 pode ser visualizado um exemplo do cenário de

colaboração médica com um cenário que considera um ambiente P2P formado a partir da tecnologia JXTA (mais detalhes serão abordados no Capítulo 5).

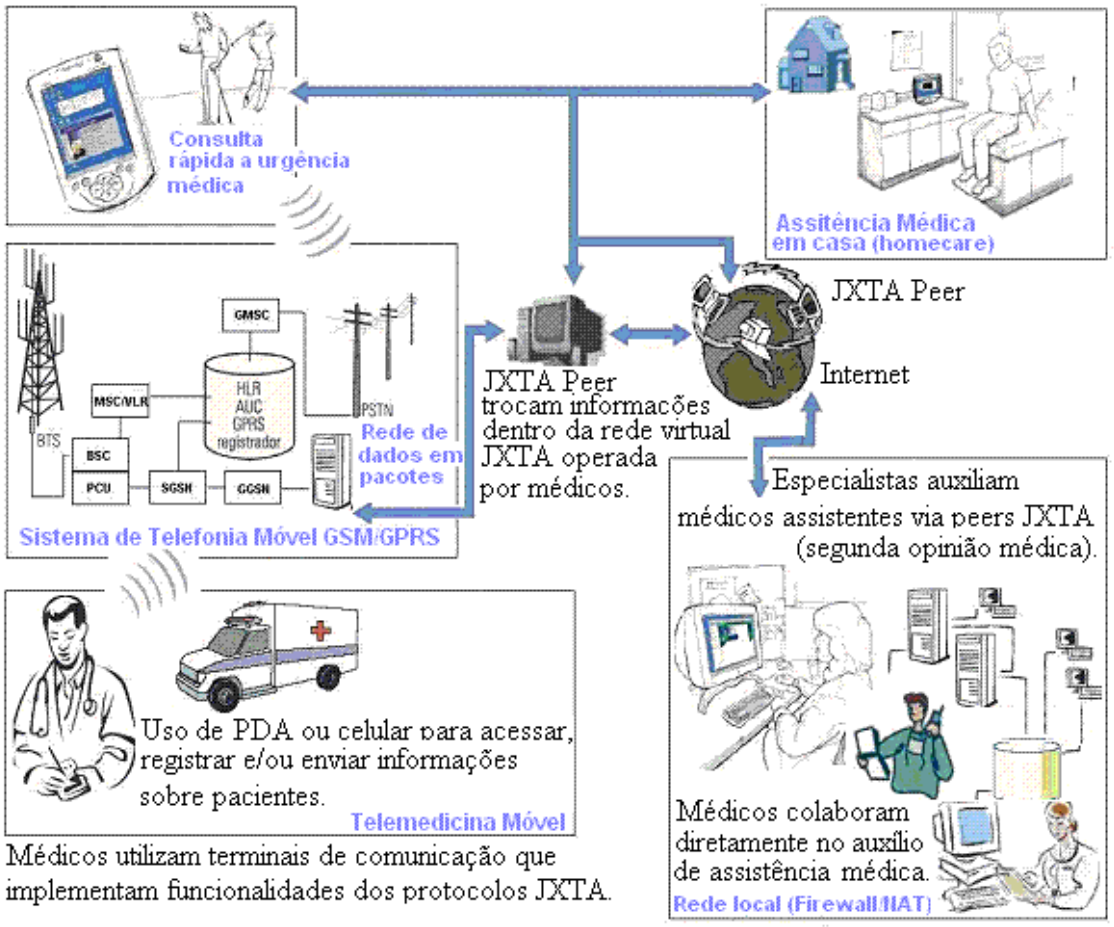


Figura 4.2: cenário de colaboração médica suportado por uma arquitetura P2P.

Na Figura 4.2, são ilustrados pelo menos seis ambientes de colaboração médica suportados pela arquitetura de comunicação P2P, são eles (TELES, 2004):

- intra – centro médico, informações e médicos estão presentes numa mesma área de atuação para intercâmbio de conhecimentos;
- móvel – centro médico, médicos providos com dispositivos computacionais móveis podem manipular informações existentes em/para centros médicos;
- médico – médico, profissionais podem trocar informações diretamente através do uso de dispositivos de comunicação;
- médico – centro médico, médicos por meio de computadores podem acessar, inserir e receber informações de/para centros médicos;

- paciente – centro médico, pacientes podem acessar informações de seus prontuários eletrônicos ou enviar informações sobre seus sinais vitais à centros médicos;
- inter – centro Médico, centros médicos podem trocar diretamente informações sobre pacientes sem alterar suas políticas de segurança de rede.

Tais ambientes, descritos anteriormente propõem a criação de uma rede virtual de colaboração médica que auxilie a prática de assistência à saúde de forma mais interativa e distribuída, ampliando o acesso à oferta remota de assistência médica especializada por meio do modelo computacional P2P.

O uso da arquitetura P2P, em sistemas computacionais que suportam a prática da assistência médica remota, oferece mais vantagens a estes sistemas em nível de interoperabilidade, tolerância à falhas e desempenho, quando comparados a outros sistemas baseados na arquitetura Cliente-Servidor (consultar Tabela 2.1). Isto destaca a importância de se avaliar uma arquitetura P2P que proponha apoiar o desenvolvimento de serviços de saúde por meio de inovações na Telemedicina.

CAPÍTULO 5

Desenvolvimento de uma rede virtual de colaboração médica P2P – experiência com JXTA

Neste capítulo, é apresentado o projeto de uma arquitetura de sistema P2P (identificado por JMED) com funcionalidades semelhantes às descritas no cenário da secção 4.3, a qual é usada para avaliar a viabilidade de aplicações P2P na área de auxílio a prestação de assistência médica à distância. Para tanto, o projeto baseia-se na implementação de uma rede virtual de colaboração médica (operada sobre a infra-estrutura da *Internet*) que visa suportar a troca de informações sobre pacientes entre médicos de maneira direta, por meio de computadores e/ou celulares interligados num estilo P2P. O projeto considera o uso da plataforma JXTA para prover a colaboração direta entre os dispositivos, a qual é avaliada neste sentido.

Para melhor entendimento do projeto, este Capítulo está divido em cinco secções, nas quais abordam-se a metodologia do projeto, sua arquitetura, as principais funcionalidades do sistema, além de uma explanação sobre a implementação dos principais métodos utilizados para a operação do mesmo e os resultados obtidos.

5.1 Metodologia do projeto JMED

O projeto JMED é composto por dois subsistemas, JMED Controle e JMED Móvel, que devem operar de forma integrada no auxílio à prestação de assistência médica remota. O primeiro consiste de uma ferramenta computacional P2P cuja funcionalidade principal é suportar um ambiente de interação, em tempo real, entre médicos para: conversação, compartilhamento e visualização de imagens de pacientes por meio de computadores interligados à *Internet*. O segundo subsistema complementa a funcionalidade do primeiro, ao possibilitar sua operação por meio de dispositivos de comunicação pertencentes a uma rede de telefonia móvel GSM/GPRS e que suportem a tecnologia J2ME[28].

Os dois subsistemas pertencentes ao projeto JMED devem atuar em conjunto, via camada de rede virtual JXTA, construída acima da *Internet*, em que microcomputadores, identificados por *peers* JXTA, possam se comunicar e compartilhar

arquivos diretamente uns com os outros, e/ou com aparelhos celulares identificados por "*peers* JXME"[29]. Para estas atividades ambos os *peers* devem pertencer ao grupo principal de *peers* da rede JXTA, conhecido por *NetPeerGroup*, em que eles podem disponibilizar seus recursos e serviços. Neste ambiente, alguns dos *peers* JXTA devem implementar funcionalidades de um *peer Rendezvous* e um *peer Relay*, a fim de possibilitar a interligação de dispositivos pertencentes a redes diferentes e a comunicação com os *peers* JXME.

O sistema JMED em si concebe a formação de uma topologia de rede P2P Híbrida baseada num estilo de comunicação ponto-a-ponto e multiponto, através dos quais *peers* podem publicar e localizar recursos disponibilizados por/aos integrantes da rede (via *advertisements* JXTA). Além disto, ele possibilita a troca de informações direta por meio de *pipes* JXTA estabelecidos entre eles. Para implementação de tal sistema é considerada a necessidade de utilização dos seguintes recursos de *software* (todos constituem plataformas de uso gratuito):

1. API JXTA J2SE e JXTA J2SE CMS – desenvolvida para a linguagem Java (http://download.jxta.org/build/release/; http://download.jxta.org/index.html);
2. API JXTA-J2ME – conhecida pela comunidade por *JXME* (baseado em *Proxy*) (http://jxme.jxta.org);
3. interface de desenvolvimento – *Netbeans* 3.6 ou 5.0 (http://www.netbeans.com);
4. ambiente de execução Java – J2SDK versão 1.4.2 (http://java.sun.com/j2se/1.4.2/download.html);
5. ferramenta de suporte ao desenvolvimento de aplicações J2ME – Java 2 *Micro Edition Wireless Toolkit* versão 2.0 a 2.2 (http://java.sun.com/j2me/download.html).

5.2 Arquitetura do sistema JMED

Com a finalidade de se implementar a solução descrita na secção anterior (baseada no uso da plataforma JXTA para prover o ambiente de colaboração P2P), a arquitetura do sistema JMED é proposta de acordo com a Figura 5.1.

---

[28] J2ME é uma plataforma de programação que permite o desenvolvimento de aplicações Java para dispositivos que possuem recursos limitados de processamento, memória e *display* (constitui uma versão simplificada da plataforma J2SE). Produtos podem incluir celular, PDA, Pager e dentre outros (MUCHOW, 2002).

[29] *Peer* JXME é um *peer* da rede JXTA que implementa as características especificadas no projeto JXTA para dispositivos J2ME, conhecido por JXME (consultar a secção do apêndice C que aborda sobre projeto JXME).

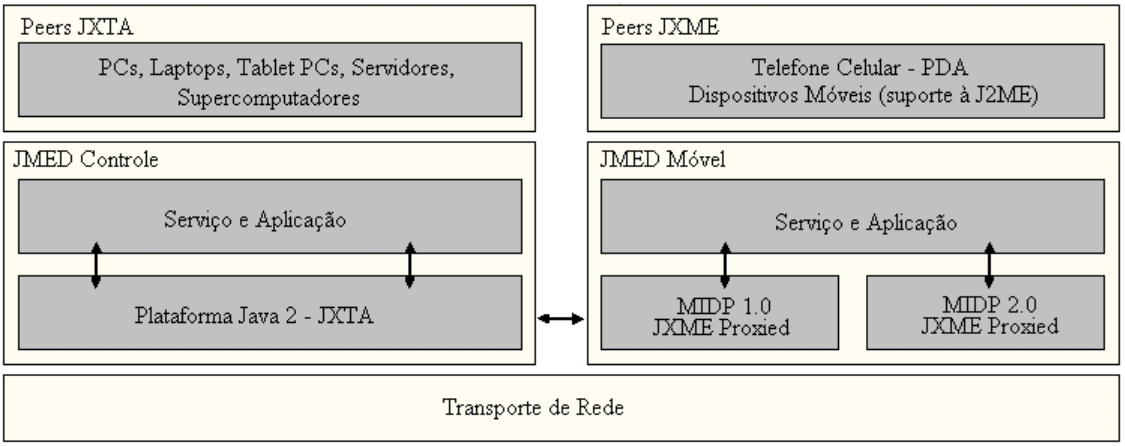


Figura 5.1: arquitetura de sistema JMED.

Conforme está ilustrado na Figura 5.1, a arquitetura do sistema JMED está dividida em dois blocos (referenciando a aplicabilidade em equipamentos com diversos perfis – *peers* JXTA e *peers* JXME – ubiqüidade), nos quais camadas distintas estão sobrepostas para dar suporte à solução JMED. No bloco para *peers* JXTA, tem-se a presença da arquitetura JXTA, fornecendo o desenvolvimento do ambiente de comunicação P2P, por meio de seus protocolos, à camada de Serviço e Aplicação. Nesta segunda camada, o serviço de troca de mensagens e de transferência de arquivos é implementado para dar suporte às funcionalidades disponíveis aos usuários através da aplicação. De forma similar, em nível organizacional de camadas funcionais, tem-se o bloco para *peers* JXME.

Na Figura 5.1, o bloco de níveis direcionado para *peers* JXME possui como base de apoio ao desenvolvimento do ambiente de comunicação P2P o perfil MIDP (1.0 ou 2.0), que é compatível com a API JXME *proxied*, e esta por sua vez com a plataforma JXTA (o que possibilita o intercâmbio de informações entre os dois subsistemas). Desta forma, a camada de Serviço e Aplicação faz uso dessa API para implementar o serviço de transferência de arquivos e de mensagens textuais, fornecendo ao usuário da aplicação uma visão transparente da rede JXTA.

5.3 Funcionalidades do JMED

O JMED Controle é operado por meio de uma interface gráfica projetada para facilitar o processo de conversação, compartilhamento e visualização de imagens de pacientes. Tal interface é dividida em duas abas de operação: uma destinada para interagir com usuários da rede que operam o JMED Controle (aba JMED - SO), e a

outra destinada para interação com usuários que operam o JMED Móvel (aba JMED - Móvel). Em ambas as abas existe uma caixa de texto para digitação e outra para visualização de todas as mensagens enviadas e recebidas, além de um painel de controle (usado para consultar informações de operação do sistema) e duas listas que identificam os *peers* correntes conectados ao *peer* em operação (listas montadas dinamicamente).

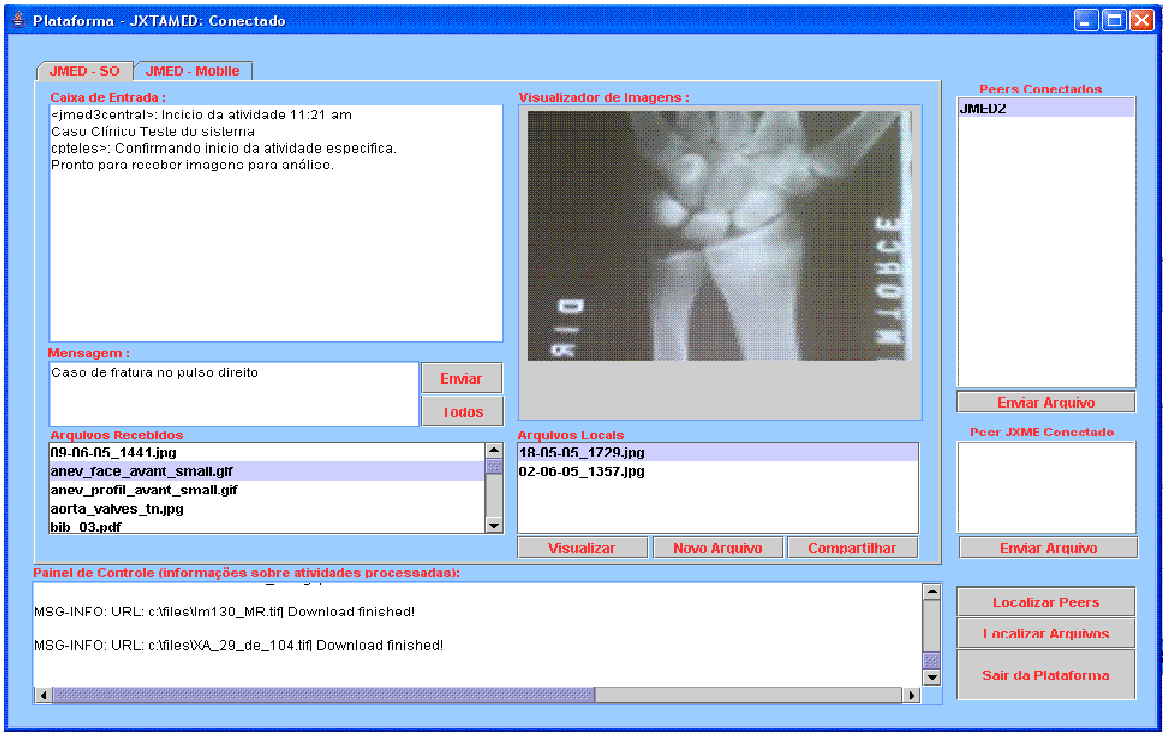


Figura 5.2: interface gráfica do subsistema JMED Controle (aba JMED - SO).

Na Figura 5.2 está ilustrada a interface gráfica do JMED Controle. Um médico operando neste sistema está apto a realizar as seguintes tarefas:

- enviar mensagens de texto (espécie de *chat* de conversação) para um único integrante do grupo ou para todos os usuários ao mesmo tempo (mensagens instantâneas);
- compartilhar arquivos diversos (.doc, .pdf, .gif e outros) com os participantes da rede, ou distribuir individualmente um arquivo específico a um usuário;
- visualizar imagens de pacientes compartilhadas por usuários da rede;
- consultar, em tempo real, usuários que estabeleceram canais de comunicação com o *peer* em atividade (disponibilizando estas informações em listas de *peers* conectados);
- localizar *peers* da rede para estabelecer canais de comunicação de dados;

- receber arquivos compartilhados na rede;
- visualizar casos de urgência médica registrada individualmente por um usuário, operando o JMED Móvel (para consultar informações dos atendimentos registrados), além de poder colaborar com estes usuários enviando procedimentos de auxílio de assistência médica que se aplique ao caso clínico analisado (formato de mensagens de texto).

Diante das funcionalidades especificadas acima, o JMED Controle contribui para a realização de atividades de grupos de profissionais de saúde que visem diagnosticar e/ou indicar o melhor tratamento a ser aplicado a um determinado caso clínico, pois, ele possibilita que uma junta de médicos, em suas próprias localidades ou ambientes de trabalho, colabore mutuamente (trocando idéias, arquivos de laudos e imagens de exames de pacientes) como se estivessem num mesmo local de atividade presencial. Isto é possibilitado a partir do uso de equipamentos computacionais interligados pelo ambiente de comunicação P2P.

Geralmente, reuniões de junta médica ocorrem numa localidade acordada entre os participantes que discutem sobre um determinado caso clínico, analisando exames e/ou laudos que normalmente estão digitalizados (é comum médicos não participarem destas reuniões por estarem em cidades diferentes da localidade acordada pela junta médica). Além de inovar na realização da prática destas reuniões (conforme exposto no parágrafo anterior), o JMED Controle permite a interação, em tempo real, entre médicos que estejam socorrendo um paciente em casos de urgência por meio da comunicação com o JMED Móvel (podendo enviar mensagens, receber dados sobre os sinais vitais do paciente, receber imagens do paciente capturadas pela câmera do aparelho celular do médico que presta socorro, além de enviar imagens para serem consultadas por médicos operando o JMED Móvel).

O JMED Móvel basicamente possui funcionalidades semelhantes às disponíveis aos usuários do JMED Controle, sendo que estão adaptadas para executar em equipamentos com limitados recursos computacionais (como por exemplo, aparelho

celular). O JMED Móvel é um "*MIDlet Java*"[30] formado por um menu de opções de atividades que permite seus usuários de realizarem as seguintes tarefas:

- registrar casos de atendimento de urgência médica numa espécie de formulário (armazenando no aparelho dados de sinais vitais do paciente e informações sobre o atendimento de urgência, em formato alfanumérico, podendo consultar, alterar e excluir estes dados posteriormente);
- capturar imagens e armazená-las no aparelho para posteriormente distribuí-las na rede virtual de colaboração médica P2P;
- participar da rede de colaboração médica através do estabelecimento de comunicação com um *peer relay JXTA* em atividade (operando o JMED Controle), sendo capaz de enviar e receber mensagens instantâneas;
- consultar um "sistema especialista"[31] através do envio dos sinais vitais de um paciente (como entrada de dados) para análise, recebendo como resposta um possível diagnóstico (em formato textual) que possa auxiliar à assistência médica a ser prestada ao paciente em atendimento;
- receber e enviar imagens de pacientes que devem ser analisadas por especialistas atuantes na rede virtual de colaboração médica (o formato das imagens que podem ser visualizadas através do aparelho celular de médicos depende do formato suportado pelo aparelho, sendo mais comum o formato png e jpeg).

Na Figura 5.3 a seguir, está ilustrado o sistema JMED Móvel em execução num "emulador"[32] de aparelho celular (disponível na ferramenta *wireless toolkit*). Tal sistema em operação com o JMED Controle fornece à rede virtual de colaboração médica P2P a construção de um ambiente global de assistência médica remota distribuída, interligando médicos que estejam na cobertura de uma rede de telefonia móvel GSM/GPRS, com outros médicos conectados à *Internet*.

---

[30] *MIDLet Java* é a unidade básica de execução ou modelo de aplicação executado em dispositivos móveis de informação que possuem recursos limitados, incluindo celular, *pagers* e outros. Ele estende a classe *javax.microedition.midlet.MIDlet* que implementa alguns métodos que definem as características chave de uma aplicação J2ME (MUCHOW, 2002).

[31] No JMED Controle existe um simulador de um sistema especialista que recebe os sinais vitais de um paciente, como dados de entrada, enviando por um *peer* que opere através do JMED Móvel. O sistema especialista ao receber esses dados, realiza uma espécie de tratamento (passando por blocos condicionais) com o propósito de montar uma mensagem textual que indique um possível diagnóstico e procedimento a ser executado. Tal mensagem automaticamente é enviado ao *peer* JMED Móvel que solicitou a requisição de análise.

[32] *Emulador* ou simulador é um ambiente de execução computacional que simula fielmente a operação de um determinado aparelho existente no mundo real.

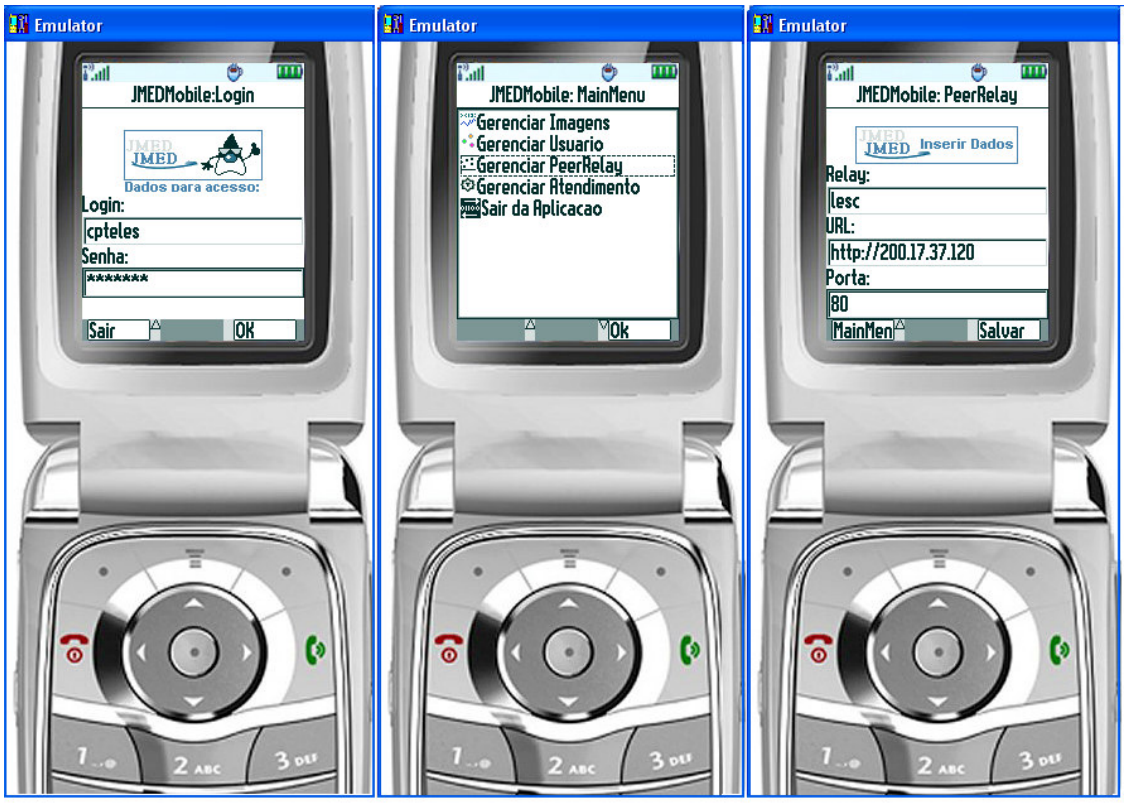


Figura 5.3: interface gráfica da aplicação JMED Móvel.

5.4 Explanação sobre a implementação do JMED

O sistema JMED foi desenvolvido em duas etapas: a primeira voltada ao ambiente de comunicação P2P entre computadores, e a segunda destinada a interação entre computadores e aparelhos celulares. Para a implementação da primeira etapa nós destacamos alguns requisitos importantes para se alcançar a interconexão entre os equipamentos alvo e o conseqüente ambiente de colaboração P2P. São eles os requisitos de:

- iniciação dos *peers* na rede virtual JXTA;
- estabelecimento de canais de comunicação entre os *peers*;
- escalonamento de atividades dos *peers*;
- compartilhamento de arquivos.

No caso da interconexão entre computadores e aparelhos celulares, os mesmos requisitos são considerados, sendo que adaptados para operarem segundo a plataforma JXME (API J2ME utilizada para estabelecer comunicação P2P com componentes da rede JXTA) (JXME, 2005).

5.4.1 Requisito de iniciação dos *peers* na rede virtual JXTA

Para que um *peer* JXTA passe a atuar na rede virtual de colaboração médica, é necessário que ele primeiro se apresente ativo dentro da rede JXTA, sendo preciso sua presença dentro do *NetPeerGroup*[33]. Este processo é facilmente implementado a partir da API J2SE JXTA (JXTA PROGUIDE, 2003).

5.4.2 Requisito de estabelecimento de canais de comunicação entre os *peers*

Após a execução do requisito anterior, o *peer* está pronto para iniciar quaisquer atividades com outros *peers* do mesmo grupo, sendo que para isto, faz-se necessário traçar estratégias para reconhecimento destes integrantes. No processo de inicialização do JMED Controle, convencionou-se que todo *peer* deve inicialmente publicar na rede, o *advertisement* usado para identificar seu *input Pipe* de conexão, e posteriormente enviar o nome deste *advertisement* para o conhecimento automático de outros *peers*, via um *pipe* de escuta comum, do tipo *propagate* (consultar apêndice C), que deve ser implementado por todo *peer* que passe a atuar na rede de colaboração médica. O formato do nome utilizado para identificar o *advertisement* de conexão baseia-se, no seguinte modelo: JXTA:JXTAMED-CONECTOR:NOME_DO_PEER.

```
<?xml version="1.0"?>
<!DOCTYPE jxta:PipeAdvertisement>
<jxta:PipeAdvertisement xmlns:jxta="http://jxta.org">
      <Id>
            urn:jxta:uuid-59616261646162614E50472050325033C0D0C4F8F94F4E5BA8C687A8D946326C04
      </Id>
      <Type>
            JxtaPropagate
      </Type>
      <Name>
            JXTA:JXTAMED-PROPAGATE
      </Name>
</jxta:PipeAdvertisement>
```

Figura 5.4: *advertisement* do *inputpipePropagation* do sistema JMED.

[33] *NetpeerGroup* é o grupo principal da rede JXTA onde qualquer *peer* que implemente os protocolos dessa rede deve iniciar suas atividades. Esse grupo disponibiliza todos os serviços básicos que podem ser implementados por qualquer *peer* JXTA (consultar apêndice C).

A estrutura XML apresentada na Figura 5.4 ilustra o *advertisement* predefinido para criação do *inputpipe*[34] que forma o *pipe propagate* especificado anteriormente. Este *pipe* atua como um canal de propagação de mensagens comum a todos os integrantes do grupo; idéia semelhante a uma "caixa postal" comum aos integrantes de uma comunidade. Partindo destas considerações, a entrada de um médico na rede passa a ser automaticamente identificada pelos integrantes do grupo, a partir do processo de reconhecimento de *advertisements* e de um processo para estabelecimento de canais de comunicação entre os integrantes da rede. A execução destes processos é realizado de forma individual por todos os componentes pertencentes à rede de colaboração médica através do sistema JMED Controle.

5.4.3 Requisito de escalonamento de atividades dos *peers*

O processo de escalonamento de atividades de cada *peer* que utiliza o JMED Controle é coordenado por um manipulador de mensagens JXTA, implementado na forma de um "*listener JAVA*"[35] associado ao *inputpipe* de comunicação. Neste processo, o manipulador filtra todas as mensagens recebidas pelo canal de comunicação do *peer* de acordo com "parâmetros"[36] que identificam ações a serem executadas.

A lógica utilizada para classificar os parâmetros usados pelo manipulador de mensagens do sistema JMED Controle está representada no pseudocódigo a seguir.

```
public void receiverMessage(PipeMsgEvent event){
   //Objeto message jxta recebe as mensagens do evento
   obter as mensagens do evento sinalizado;
    Enquanto(existir mensagens obtidas)Faça{
      capturar o nome da mensagem;
      Se(nome da mensagem == "PIPE"){//localizar pipe JXTA
            //conteúdo da mensagem traz o nome do advertisement do
            //pipe do peer de conexão(JXTA:JXTAMED-CONECTOR:nomePeer)
      realizar parser para obter o nome do peer;
      Se(peer não existir na lista de peer conectados){
              //utiliza nome do advertisement para procura
        procurar o pipe de conexão do peer encontrado;
```

[34] *Inputpipe* usado como uma "janela" de escuta do canal de comunicação estabelecido para um serviço (consultar apêndice C).
[35] *Listener Java* pode ser entendido como uma *thread* nativa ao objeto Java que monitora todas as ações e eventos relacionados a ele.

```
    }
  }Se Não Se(nome da mensagem == "MPIPE"){
        //localizar pipe para conexão com o peer mobile JXME
        //obedece a mesma lógica anterior para localizar pipe
}Se Não Se(nome da mensagem == "PEER"){
        //conteúdo da mensagem traz o nome de um peer
        Se(peer não existir na lista de peer conectados){
          //utiliza nome do peer para procurar
    procurar o pipe de conexão do peer encontrado;
    }
}Se Não Se(nome da mensagem == "MSG"){
        //indica mensagem de texto enviada por um peer remoto
        //mensagem de texto obedece o formato(nomePeer>mensagem)
        realizar parser para obter o nome do peer;
        Se(peer não existir na lista de peer conectados){
          //utiliza nome do advertisement para procura
    procurar o pipe de conexão do peer encontrado;
    }
        mostrar a mensagem de texto na caixa de entrada;
 }Se Não Se(nome da mensagem == "MMSG"){
        //mensagem de texto enviada por um peer mobile JXME
        mostrar a mensagem de texto na caixa de entrada;
 }Se Não Se(nome da mensagem == "MSGOUT"){
        //indica aviso de desconexão de um peer
        //obedece o formato(nomePeer>msg_de_desconexão)
        realizar parser para obter nome do peer;
        remover peer da lista de peers conectados;
 }Se Não Se(nome da mensagem == "ESP"){
        //indica solicitação de analise ao sistema especialista
        //essa mensagem só é enviada por um peer mobile JXME
        //conteúdo da mensagem traz sinais vitais de paciente
        obter sinais vitais do paciente;
        realizar análise dos sinais vitais do paciente;
        //enviar resultado através de uma mensagem ao peer mobile
        //nome da mensagem "RESP"
        retornar resultado da análise realizada;
 }Se Não Se(nome da mensagem == "PRO"){
        //indica mensagem de propagação comum a todos os peers
        //usado para sinalizar sua existência na rede para que
```

[36] Parâmetro é o termo aqui usado para indicar palavras reservadas ao sistema JMED.

```
            //os demais peers conectados realizam conexão com ele
            //o conteúdo da mensagem é o nome do pipe de conexão
            //formato(JXTA:JXTAMED-CONECTOR:nomePeer)
            realizar parser para obter nome do peer;
            Se(peer não existir na lista de peer conectados){
              //utiliza nome do peer para procurar
          procurar o pipe de conexão do peer encontrado;
          }
       }Se Não Se(nome da mensagem == "RSIN"){
            //sinaliza o recebimento de resposta de sincronização.
            //conteúdo da mensagem fornece todos os atendimentos
            //registrados por um peer mobile JXME
            armazenar os atendimentos recebidos para visualização;
       }Se Não{
            //peer mobile envia um arquivo a ser compartilhado
            //arquivo é armazenado localmente para visualização
            processar mensagem de arquivo compartilhado;
       }
      }
}
```

5.4.4 Requisito de compartilhamento de arquivo

Para o processo de distribuição e compartilhamento de arquivos, a arquitetura de sistema JMED implementa duas maneiras de se realizar tais atividades. Uma delas utiliza as funcionalidades do projeto CMS JXTA, implementado pela comunidade JXTA (JXTA_CMS, 2005) e a outra através de mensagens JXTA. JMED compartilha arquivos com todos os *peers* JXTA através da API JXTA CMS e distribui um arquivo específico para um determinado *peer* da comunidade através de mensagem JXTA. A idéia do projeto CMS é compartilhar um *advertisement* comum a todos os *peers*, registrando no mesmo, todos os arquivos compartilhados por um *peer*. Quando um *peer* deseja obter os arquivos compartilhados na rede, este *advertisement* é requisitado, e todos os *peers* que o possuem, respondem a requisição informando seus arquivos disponíveis. A partir daí, o processo de transferência de arquivos é realizado.

5.4.5 Lógica aplicada no desenvolvimento do JMED Móvel

O JMED Móvel por ser um sistema focado em operar de forma compatível com as funcionalidades do JMED Controle (consultar secção 5.2), suas operações são acionadas por "menus de opções" que facilitam a operação do *peer* JXME, a partir de uma interface gráfica implementada usando os recursos disponíveis no perfil "MIDP J2ME"[37]. Inicialmente, quando o *MIDlet* for acionado pelo usuário (a partir da secção de aplicações Java disponíveis no aparelho), um formulário de controle de acesso ao JMED Móvel é acionado, a fim de permitir a interação, ou não, do usuário com o menu de opções do JMED Móvel (ver Figura 5.3).

Os dados de acesso ao JMED Móvel são "gerenciados"[38] a partir de uma opção do menu principal ("Gerenciar Usuário"), bem como os dados referentes aos atendimentos de urgência médica de pacientes ("Gerenciar Atendimento"), captura de imagens estáticas ("Gerenciar Imagens") e dados de identificação de *relays* JXTA ("Gerenciar PeerRelay", usado para conectar a rede JXTA). O armazenamento persistente destes dados no aparelho é implementado via *Record Management System* (RMS) (Muchow, 2002), contemplando as seguintes tabelas de armazenamento de bytes: "HealthCare", "Relays", "UserConfig", "Images" e "ImageDados". A criação ou a ativação dessas tabelas de armazenamento persistente constitui pré-requisito para operação do JMED Móvel, ou seja, antes de qualquer interação do usuário, o sistema deve automaticamente criá-las (caso não exista) ou iniciá-las.

Uma funcionalidade importante do JMED Móvel, que está vinculada ao menu principal, diz respeito ao processo de conexão a um *peer relay* JXTA para permitir a colaboração de um médico na rede virtual ("Gerenciar PeerRelay"). Para tanto, é necessário recuperar do armazenamento persistente algumas informações sobre um *peer relay* registrado: a *URL*, uma porta IP estática e seu nome. A partir destas informações o processo de conexão é ativado por uma classe Java que obedece a uma lógica de instruções de acordo com o pseudo código a seguir.

public JxtaMedConnect(){

[37] Perfil MIDP J2ME define APIs de componentes focados na criação de interfaces gráficas para usuário de um *MIDlet*, incluindo gerenciamento de eventos, armazenamento persistente de dados e comunicação, considerando às limitações de memória e tela dos dispositivos (MUCHOW, 2002).

```
        //definir nome do inputPipe de conexão, obedecendo o formato -
                //"JXTA:JXTAMED-CONECTOR:" + nome_peer_Médico;
        //definir nome do outputPipe de conexão, obedecendo o formato –
                //"JXTA:JXTAMED-CONECTOR:" + nome_relay;
        //criar um novo peer JXME com o nome especificado pelo médico
        //estabelecer tentativa de conexão com o peer relay (relayURL obedece
                //o formato: "http://xxx.xxx.xxx.xxx:Porta_IP")
        //Criar um inputPipe para receber dados vindo da rede JXTA (usar nome
                //definido anteriormente)
         //procurar pelo outputPipe de conexão a partir do nome definido
                //(procura pelo advertisement do nome definido)
         //guardar o id de identificação do outputPipe localizado
         //envia o nome do inputPipe criado para que o peer JXTA
                //estabeleça conexão com o peer JXME (parâmetro "MPIPE")
         //criar e disponibilizar ao usuário um novo menu de opções para
                //executar em cooperação com o JMED Controle
}
```

De acordo com o pseudocódigo anterior, um novo "menu de opções" é criado para que novas funcionalidades sejam disponibilizadas aos usuários do JMED Móvel, com o propósito de possibilitar a colaboração entre os médicos pertencentes à rede virtual JMED. Neste novo contexto funcional, os médicos interagem entre si, a partir do envio de mensagens textuais instantâneas, de imagens médicas capturadas pela câmera acoplada ao aparelho celular e através do pedido de consulta ao sistema especialista. Todas estas funções fazem uso do *outputPipe*, criado no processo de conexão, e de mensagens JXTA que são usadas para transportar os elementos de dados que compõem a informação a ser transmitida na rede. Note que, para haver interação entre o JMED Controle e o JMED Móvel, estas informações devem ser manipuladas de acordo com os parâmetros de filtro usado pelo escalonador de atividades do JMED Controle – consultar pseudocódigo da secção 5.4.3.

Considerando que o JMED Móvel é implementado a partir da API JXME baseada em *proxy* (consultar apêndice C para maiores informações), o processo de cooperação entre as funcionalidades do JMED Móvel e do JMED Controle obedecem a um mecanismo de comunicação síncrona. Desta forma, durante o intervalo de períodos de tempo determinado, o *peer* JXME consulta seu representante na rede JXTA (que é o

[38] Gerenciados no sentido de permitir consulta, alteração e em alguns casos exclusão de dados.

peer relay) para verificar se há alguma mensagem destinada a ele. Caso o retorno desta consulta seja afirmativo, o escalonador de mensagens do JMED Móvel entra em atividade de forma similar a executada no JMED Controle (filtrando mensagens JXTA a partir de parâmetros existentes nos elementos da mensagem, como por exemplo: “SIN”, “RESP”, “MSG” e outros). O pseudocódigo a seguir fornece a lógica utilizada para classificar os parâmetros que indicam uma nova ação a ser executada pelo JMED Móvel.

```
public void recvMessage(){
   //novas mensagens recebidas
   Para(I = 0) até (I < Numero_Mensagem)Faça{
          //guardar elemento da mensagem
          Element e = msg.getElement(i);
          //verificar cabeçalho do elemento
          Se(e.getNameSpace == "JXTAMED"){
       Se(e.getName() == "SIN"){
                    //indica solicitação do JMED Controle de pedido de envio
                    //de todos os atendimentos de urgência registrados
             }Se(e.getName() == "RESP"){
                    //resposta recebida do sistema especialista para consulta
             }Se(e.getName() == "MSG"){
                    //indica mensagem de texto recebida para visualização no chat
             }
          }SeNão Se(e.getNameSpace == ""){
      Se(e.getName() == "SAVEFILE"){
                    //salvar arquivo enviado pelo JMED Controle
            }
          }
   }
}
```

5.5 Testes e resultados obtidos

Partindo da necessidade de avaliar tanto as funcionalidades do sistema JMED quanto aos recursos da plataforma JXTA (como tecnologia de apoio ao desenvolvimento de aplicações P2P), busca-se a implementação de um ambiente computacional para aplicação de um plano de teste focado na análise da colaboração P2P. Para isto, constata-se a necessidade tanto de equipamentos pertencentes a redes

computacionais distintas quanto de dispositivos pertencentes a redes de telefonia móvel, para a formação de um ambiente variado de comunicação para análise de interações entre esses equipamentos, interligados virtualmente numa comunidade P2P (TELES, 2004).

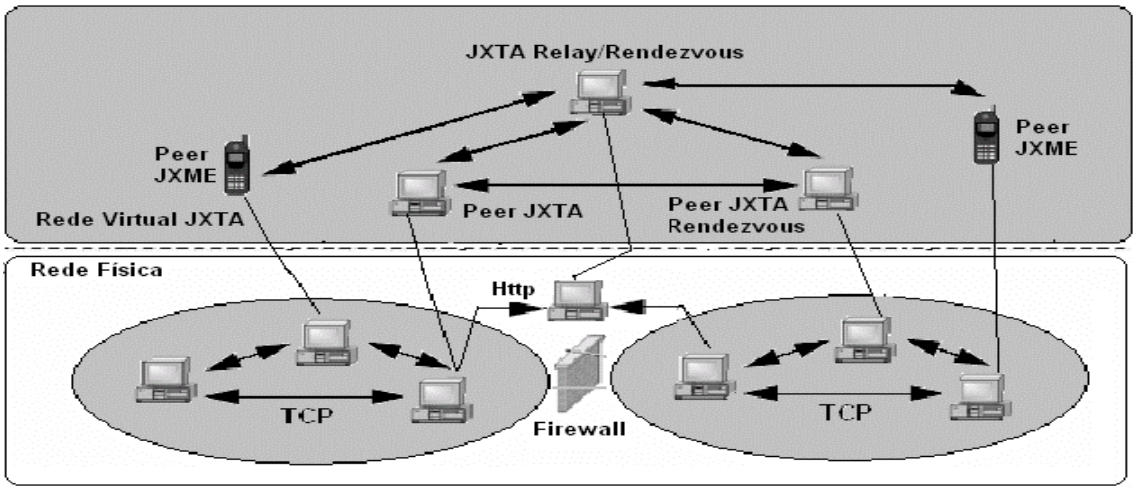


Figura 5.5: arquitetura de testes do sistema JMED.

Conforme está ilustrado na Figura 5.5, a formação do ambiente de testes se dá a partir do uso de duas redes IP distintas, isoladas por meio de *firewalls*. Os equipamentos destas redes são utilizados para formar a comunidade virtual de colaboração P2P, sustentada sob a infra-estrutura da *Internet*, a fim de permitir a troca de contexto entre *peers* JXTA e, entre *peers* JXTA e *peers* JXME. Para isto, é necessário que alguns desses *peers* assumam a função de *rendezvous* e/ou *relays* JXTA.

A partir do ambiente construído acima, foi definido o plano de teste baseado na verificação da ocorrência e validação das seguintes funcionalidades:

- conectar *peers* JXTA pertencente a redes diferentes;
- estabelecer um serviço de comunicação, por meio da troca de mensagens instantâneas, entre os *peers* JXTA referidos anteriormente;
- manter o mesmo serviço de comunicação anterior, sendo que seja possível realizar um estilo de comunicação um para muitos, ou seja, uma mensagem a ser enviada por um *peer* para múltiplos *peers* simultaneamente;
- compartilhar arquivos com outros *peers* pertencentes a um mesmo grupo;
- distribuir um arquivo para um determinado *peer* indicado;
- conectar um *peer* JXTA com um *peer* JXME;

- estabelecer serviço de comunicação entre *peers* JXTA e JXME (baseado na troca de mensagens JXTA *com peers relays*);
- distribuir um arquivo específico entre *peers* JXTA e JXME.

Com o plano de teste especificado, iniciou-se o experimento por meio da verificação do estabelecimento de conexão entre três *peers* que implementam o JMED Controle (um atuando como *relay* e *rendezvous*). Conforme o requisito apresentado na secção 5.4.2, constatatou-se a conexão automática destes *peers* ao fazerem uso do *pipe* de propagação. Isso dá condições favoráveis para validar o processo de inicialização dos *peers* na comunidade médica, através do estabelecimento de *pipes* de comunicação entre eles.

Dando-se por finalizada e validada o teste de conexão entre *peers*, iniciou-se o processo de verificação do funcionamento do sistema de comunicação, mantido a partir de canais de conexão estabelecidos. Constatou-se que durante todo o período de conexão, os *peers* se comunicaram diretamente uns com os outros, através da troca de mensagens instantâneas encapsuladas em mensagens JXTA. Esta troca de contexto é validada com sucesso tanto através do serviço de mensagem "um para um" quanto de "um para muitos", ou seja, usando o serviço por meio do *unicast pipe e propagate pipe* respectivamente.

Para finalizar, a verificação de comunicação entre *peers* JXTA, o serviço de compartilhamento e distribuição de arquivos (oferecido pelo sistema JMED Controle) é validado obtendo os seguintes resultados:

- é possível distribuir arquivos, de até 2MB, diretamente entre *peers* específicos, a partir do uso de mensagens JXTA binária;
- e para o compartilhamento de arquivos dentro do grupo, foi validada a transferência de arquivos, de tamanhos maiores que o especificado no item anterior, por meio do serviço baseado na publicação de anuncio de acordo com a API CMS.

As funcionalidades propostas ao sistema JMED foram validadas após a realização sucessiva de testes nas atividades descritas anteriormente. A comunicação P2P entre *desktops*, por meio da tecnologia JXTA, deu-se então por finalizada e validada. Em seguida, iniciou-se a segunda bateria de testes que inclui o uso de aparelhos celulares

(simulados em *desktops*) na rede virtual de colaboração P2P. Para isto, inicialmente desenvolveu-se uma aplicação simples, com o objetivo de verificar a inscrição de dois *peers* JXME junto a um *peer relay* (operado por meio do *sheel* JXTA), a fim de verificar a conexão e validar o tratamento de requisições individuais realizada pelos *peers* JXME ao *peer relay* (TELES, 2004).

Com a realização do último teste descrito acima, constatamos que um *peer relay* ao receber um pedido de conexão de um *peer* JXME, registra um ID de identificação deste *peer*, e comunica ao mesmo o estabelecimento da conexão por meio de uma mensagem, possibilitando que o mesmo passe a participar na rede JXTA. Após esta constatação, utilizou-se o JMED Móvel na busca de validar as funcionalidades descritas na secção 5.3.

O processo de integração do JMED Móvel com o JMED Controle ocorreu com sucesso através da troca de mensagens com o *peer relay* (consultar apêndice C - JXME), o que torna possível a validação do *chat* de conversação, consulta ao sistema especialista do JMED Controle e distribuição de imagens capturadas pela câmera do celular. Com todas estas funcionalidades validadas no ambiente de simulação, partiu-se para o teste final, realizando as mesmas atividades através dos seguintes aparelhos reais: Motorola V400, V600 e V3, Nokia 6230, Nokia 8250, Nokia N-90, Samsung SGH-D500 e Siemens SL-65. Destes aparelhos, apenas no Nokia N-90 e no Siemens SL-65, obteve-se sucesso em todas as funcionalidades fornecidas pelo JMED Móvel. Este fato nos levou a identificar a possibilidade da falta de compatibilidade entre o sistema operacional (SO) do aparelho real e de seu SDK (*Software Development Kit*) disponibilizado por fabricantes, visto que obteve-se problemas ao se utilizar a câmera VGA acoplada aos dispositivos e estabelecer uma conectividade de rede constante.

Para exemplificar o problema acima, tome por base dois modelos da Motorola, o V400 e V600. Por meio destes modelos, não é possível capturar imagens, pois no momento em que é solicitado o recurso da câmera pela aplicação, a mesma encerra-se com um erro originado pelo SO do aparelho. Isto traz questionamentos e leva a realização de diversas revisões na implementação, pois, durante a simulação com os mesmos modelos, este erro não aprece. Um problema semelhante ocorre durante o estabelecimento de conectividade de rede, pois, no teste realizado através do aparelho

real, o JMED Móvel não consegue estabelecer conexão com o *peer relay*, enquanto que no ambiente simulado a conexão ocorre normalmente.

CAPÍTULO 6
Conclusão e Contribuições

Durante a realização dos testes desta arquitetura, buscou-se avaliar a aplicabilidade real da tecnologia P2P em prover um ambiente favorável para colaboração médica distribuída sobre a infra-estrutura da *Internet*. Este ambiente foi alcançado através do desenvolvimento da arquitetura JMED, que visa ofertar a mobilidade da informação entre médicos a partir de qualquer dispositivo computacional, capaz de estabelecer conexão com uma rede de dados. Tal sistema é implementado utilizando-se de recursos propostos pela tecnologia JXTA, a qual objetiva fornecer um ambiente padronizado de programação para computação de rede P2P, visando a interoperabilidade de sistemas, independência de plataforma (sistema operacional, transporte de rede, linguagem de programação) e ubiqüidade na disponibilidade de seus serviços. Até este momento, na literatura por nós conhecida, não há evidências de uma arquitetura de sistemas P2P que se proponha a ser aplicado na área de Temedicina. Desta forma, este trabalho contribui de maneira significativa nesta área através da arquitetura JMED.

Por meio das funcionalidades do sistema JMED, é comprovada a eficácia da plataforma JXTA em propor um ambiente de comunicação P2P genérico, que suporta a interconectividade entre equipamentos computacionais de diferentes plataformas de *hardware* e *software*, e que a solução JXTA/JXME constituem tecnologias viáveis para implementação de aplicações P2P entre dispositivos móveis e portáteis J2ME, devido à simplicidade e facilidade de utilização da API JXME. Além disso, pôde-se conhecer a falta de informações fornecidas por fabricantes de aparelhos celulares, no que diz respeito aos recursos do perfil MIDP disponíveis para uso por parte de consumidores projetistas de aplicações J2ME. Por meio da experiência, é constatado que não basta informar qual versão do MIDP o aparelho suporta e qual JSR[39] o mesmo implementa. Faz-se necessário também informar quais recursos estão disponíveis para uso em aplicações desenvolvidas por consumidores.

[39] JSR (*Java Specification Requests*) são especificações definidas pela JCP (*Java Community Process*) relacionadas ao uso da tecnologia Java (http://www.jcp.org/en/whatsnew/jcp; http://www.jcp.org/en/jsr/all).

Um exemplo claro do problema descrito acima foi enfrentado durante a realização dos testes de conectividade de rede e na captura de imagens por parte da câmera VGA acoplada no aparelho. No caso desse último exemplo, o requisito necessário é que o dispositivo implemente a JSR 135 (conhecida por *Mobile Media API* usada para manipular recursos multimídia – vídeo e áudio). Diversos aparelhos que disponibilizam tal API são facilmente encontrados no mercado, porém em alguns não é possível alcançar o sucesso de usar esta API para capturar imagem. Os aparelhos utilizados foram: Motorola V400, V600 e V3, Nokia 6230, Nokia 8250, Nokia N-90, Samsung SGH-D500 e Siemens SL-65. Destes aparelhos, obteve-se sucesso apenas com o Nokia N-90 e Siemens SL-65; os demais apresentam problemas na execução. Com relação à conectividade, tanto o MIDP 1.0 e 2.0 permitem o uso de requisições HTTP por parte de aplicações J2ME (requisito este suficiente para a API JXME estabelecer comunicação com um *peer Relay* JXTA). Todos os aparelhos descritos suportam o MIDP 2.0, porém isto não é suficiente para permitir a conectividade com o *peer relay*, em todos eles (com nenhum Motorola utilizado conseguiu-se conexão e, neste caso, não foi realizado teste com o Nokia 8250).

Diante dos problemas enfrentados, conclui-se que determinadas JSRs suportadas pelos aparelhos são utilizáveis apenas por aplicações nativas do fabricante, e isto é entendido como uma estratégia de proteção imposta pelo sistema operacional dos mesmos.

Por fim, conclui-se que a tecnologia P2P pode ser considerada como uma tecnologia importante a ser explorada no desenvolvimento de aplicações computacionais tanto na área de Telemedicina, a fim de se inovar na oferta de serviços médicos especializados à distância, quanto em qualquer outra área que necessite interconectar equipamentos computacionais que busquem a troca e a distribuição de informações eletrônicas.

6.1 Trabalhos futuros

Para trabalhos futuros relacionados a esta dissertação, destaca-se a importância de se realizar os seguintes estudos e projetos:

- estudo na área de processamento de sinais focado no filtro de imagens capturadas a partir da câmera VGA de aparelhos celulares. O objetivo desse estudo é avaliar a

possibilidade de se realizar diagnóstico à distância através do uso de imagens de pacientes e com o uso de um telefone celular como meio de transmissão. De posse dessas imagens, o especialista pode executar filtros em imagens de exames de Raio-X, Ressonância Magnética, Tomografia Computadorizada e Ultra-sonografia;

- realizar testes de desempenho na rede JXTA, avaliando o impacto causado na performance da rede física implementada que fornece suporte ao desenvolvimento do ambiente de comunicação P2P;
- realizar estudo comparativo entre o uso da API JXME versão *proxyless* e *proxied*, destacando suas principais vantagens e desvantagens co-relacionadas;
- expandir as funcionalidades do sistema JMED, para que o mesmo proporcione a captura automática de informações sobre pacientes, a partir da tecnologia *Bluetooth*. Nesse contexto, destacam-se a importância da coleta automática dos sinais vitais de pacientes, por meio de sensores *Bluetooth* que se comunicam com celulares que suportam tal tecnologia;
- avaliar o desenvolvimento de um ambiente de comunicação P2P, usando a tecnologia JXTA para prover uma colaboração descentralizada entre equipamentos que suportem *Bluetooth*;
- fazer estudo comparativo de plataformas de programação P2P existentes na atualidade;
- desenvolver uma aplicação que objetive analisar as JSR suportadas por um telefone celular, a fim de ser utilizada como uma ferramenta de auxilio ao desenvolvimento de aplicações J2ME. Tal aplicação deverá fornecer informações sobre quais recursos estariam disponíveis para uso em aplicativos desenvolvidos por usuários finais.

Referências Bibliográficas

(ANDROUTSELLIS, 2005) Androutsellis, S.; Spinellis, D. "A Survey of Peer-to-Peer Content Distribution Technologies". In: Jornal ACM Computing Surveys, vol.36, Nº.4,December 2004, pp.335-371. Disponível em: http://www.dmst.aueb.gr/dds/pubs/jrnl/2004-ACMCS-p2p/html/AS04.pdf. Última data de acesso: 18/05/06.

(ARORA, 2002) Arora, A.; Haywood, C.; Pabla, K. S. "JXTA for J2ME – Extending the Reach of Wireless with JXTA Technology". In: Technical Report Sun Microsystems Inc, Março 2002. Disponível em: http://www.jxta.org/project/www/docs/JXTA4J2ME.pdf. Última data de acesso: 18/05/06.

(BARBOSA, 2004) Barbosa, A. K. P.; Novaes, M. A.; Araújo, G. P. M.; Sarmento, L. C. M.; Lima, A. L. S. "Criação de uma rede de cooperação em saúde baseada em Web Services". In: Anais do IX CBIS – IX Congresso Brasileiro de Informática em Saúde, Novembro 2004. Disponível em: http://www.hu.ufsc.br/IX_CIBS/trabalhos/arquivos/360.pdf. Última data de acesso: 18/05/06.

(BARDRAM, 2003) Bardram, J. E. "Hospitals of the future – Ubiquitous computing support for Medical Work in Hospitals". In: Proceedings of UbiHealth 2003: The 2nd International Workshop on Ubiquitous Computing for Pervasive Healthcare Applications. Disponível em: http://www.healthcare.pervasive.dk/ubicomp2003/papers/Final_Papers/13.pdf. Última data de acesso: 18/05/2006.

(BROOKSHIER, 2002) Brookshier, D.; Govoni, D.; Krishnan, N. "JXTA: Java P2P Programming". 1º.ed., 2002 Indianapolis: SAMS, ISBN: 0672323664.

(CABAÇO, 2003) Cabaço, S. Sítio sobre segurança em sistemas de informação distribuídos. Disponível em: http://clientes.netvisao.pt/sucabaco/trabalhos/SSID/SSID-TS.pdf. Última data de acesso: 15/04/2004.

(CARRARE, 2002) Carrare, A. P. G. D.; Amaral, L. H. ; Moura, L. A. R. "Biblioteca Digital de Imagens Médicas – Uma Proposta de Democratização e Conservação de Imagens". In: Anais do VIII CBIS – VIII Congresso Brasileiro de Informática em Saúde, Setembro 2002. Disponível em: http://www.avesta.com.br/posters/ps03_carrare.pdf. Última data de acesso: 15/04/2004.

(CFM, 2002) Brasil, Resolução do Conselho Federal de Medicina – CFM nº 1.643/2002. Disponível em: http://www.unifesp.br/dis/set/law/resolucaocfm16382002%20.html. Última data de acesso: 18/05/2006.

(COULOURIS, 2001) Coulouris, G.; Dollimore, J.; Kindberg, T. "Distributed Systems: Concepts and Design". 3º ed., 2001. Boston, USA: Addison-Wesley Longman Publishing Co., ISBN 0201-619-180.

(DING, 2003) Ding, C. H.; Nutanong, R. Buyya. "Peer-to-Peer Networks for Content Sharing". In: Technical Report Grid Computing and Distributed Systems Laboratory, University of Melbourne, Australia, Dezembro 2003. Disponível em: http://www.gridbus.org/papers/P2PbasedContentSharing.pdf. Última data de acesso: 18/05/2006.

(DTSOUMA, 2003) Dtsouma, D. T. "A Comparison of Peer-to-Peer Search Methods". In: Proceedings of International Workshop on the Web and Databases (WebDB). June 12-13 2003, San Diego, California. Disponível em: http://www.cse.ogi.edu/webdb03/papers/11.pdf. Última data de acesso: 18/05/2006.

(FRANÇA, 2001) França, G. V. "Telemedicina: breves considerações ético-legais". In: REDI – Revista Electrónica de Derecho Informático, nº.39, Out. 2001. Disponível em: http://www.portalmedico.org.br/revista/bio1v8/simpo6.pdf. Última data de acesso: 18/05/2006.

(GONG, 2001) Gong, L. "JXTA: A Network programming environment". In: Technical Report Sun Microsystems Inc, Jun. 2001. Disponível em: http://www.jxta.org/project/www/docs/JXTAnetworkProgEnv.pdf. Última data de acesso: 18/05/2006.

(GRADECKI, 2002) Gradecki, J. D. "Mastering JXTA. Building Java Peer-to-Peer Applications". 1º ed. Indiana: Wiley Publishing Inc., ISBN: 0471250848. Disponível em: http://www.win.tue.nl/~ymazuryk/books/m_jxta.pdf. Última data de acesso: 18/05/2006.

(HAJAMOHIDEEN, 2003) Hajamohideen, S. H. "A Model for Web Service Discovery and Invocation in JXTA". In: Project work, Hamburg University of Technology, Hamburg, Germany, Abr. 2003. Disponível em: http://www.ti5.tu-harburg.de/publication/2003/Thesis/haja03/haja03.pdf. Última data de acesso: 18/05/2006.

(HERTZOG, 2005) Hertzog, H.; Silveira, J. G. "Desenvolvimento de Ferramentas Aplicadas à Telemonitoração e Teleconsultas Remotas". In: Revista CCEI – URCAMP, v.9, n.15, p.34-44, Março 2005. Disponível em: http://www.urcamp.tche.br/ccei/revista15.pdf. Última data de acesso: 18/05/2006.

(HIRA, 2002) Hira, A. Y.; Bacic, A. S.; Zuffo, M. K.; Lopes, R. D. "A Telemedicina sob o Paradigma de Componentes e Objetos Distribuídos. Um Estudo de Caso: Protocolos Cooperativos em OncoPediatria". In: Anais do VIII CBIS - Congresso Brasileiro de Informática em Saúde, Setembro 2002. Disponível em: http://www.avesta.com.br/anais/dados/trabalhos/314.pdf. Última data de acesso: 15/04/2004.

(ISHIKAWA, 2003) Ishikawa, N.; Kato, T.; Sumino, H.; Hjelm, J.; Yu, Y.; Zhu, Z. "A Platform for Peer-to-Peer Communications and its Relation to Semantic Web Applications". In: Proceedings SemPGRID Workshop - 2nd Workshop on Semantics in Peer-to-Peer and Grid Computing, 17-18 May 2003, New York USA. Disponível em: http://citeseer.ist.psu.edu/cache/papers/cs/27541/http:zSzzSzwww.isi.eduzSz~stefanzSzSemPGRIDzSzproceedingszSz1.pdf/a-platform-for-peer.pdf. Última data de acesso: 18/05/2006.

(JAIN, 1998) Jain, P.; Widoff, S.; Schmidt, D. C. "The Design and Performance of MedJava. Experience Developing Performance-Sensitive Distributed Applications with Java". In: EE/BCS Distributed Systems Engineering Journal, 1998 vol.5, nº.4, pp141-155. Disponível em: http://www.cs.wustl.edu/~schmidt/PDF/MedJava.pdf. Última data de acesso: 18/05/2006.

(JI, 2003) JI, L. "Computation in Peer-to-Peer Networks". In: Proceedings of the 2002-2003 Grad Symposium, CS Dept, University of Saskatchewan, 10 Abril 2003. Disponível em: http://bistrica.usask.ca/madmuc/Pubs/lichun880.pdf. Última data de acesso: 18/05/2006.

(JXTA SPEC, 2004) Sun Microsystems. "JXTA v2.0 Protocols Specification". In: White Paper Sun Microsystems Inc. Disponível em: http://spec.jxta.org/nonav/v1.0/docbook/JXTAProtocols.pdf. Última data de acesso: 18/05/2006.

(JXTA, 2005) Sitio oficial do Projeto JXTA. http://www.jxta.org . Última data de acesso: 18/05/2006.

(JXTA_CMS, 2005) Sítio oficial do Projeto JXTA CMS. http://cms.jxta.org/ . Última data de acesso: 18/05/2006.

(JXME, 2005) Sítio oficial do Projeto JXTA-J2ME: JXME. http://jxme.jxta.org/. Última data de acesso: 18/05/2006.

(JXTA PROGUIDE, 2003) Sun Microsystems. "Project JXTA: Java Programmer's Guide". In: White Paper Sun Microsystems Inc. Disponível em: http://www.jxta.org/docs/JxtaProgGuide_v2.pdf. Última data de acesso: 07/08/2004.

(KATO, 2003) Kato, T. "A Platform and Applications for Mobile Peer-to-Peer Communications". In: Technical Report Ericsson Research Japan, Abril 2003. Disponível em: http://www.research.att.com/~rjana/Takeshi_Kato.pdf. Última data de acesso: 18/05/2006.

(KHLUDOV, 2000) Khludov, S.; Vowerk, L. "Internet-Oriented Medical Information System for DICOM-Data Tranfer, Visualization and Revision". In: Proceedings of CBMS 2000, Houston Texas, USA, pp293-296. Disponível em: http://www.informatik.uni-trier.de/TI/Projekte/Telemedizin/cbms2000hlu.pdf. Última data de acesso: 18/05/2006.

(KLEINROCK, 2003) KLEINROCK, L. "An Internet Vision: The invisible global infrastructure". In: Technical Report Computer Science Department UCLA, Los

Angeles. Disponível em: http://www.lk.cs.ucla.edu/PS/paper223.pdf. Última data de acesso: 18/05/2006.

(KORHONEN, 2005) Korhonen, I. "Pervasive Health Technologies Research at VTT". In: Proceedings of Connections Workshops in Berkeley and in Tampere, Abr. 2005. Disponível em: http://websrv2.tekes.fi/opencms/opencms/OhjelmaPortaali/Kaynnissa/FinnWell/fi/Dokumenttiarkisto/Viestinta_ja_aktivointi/Seminaarit/Berkeley/Pervasive_health_xVTT.pdf. Última data de acesso: 22/05/2006.

(KURMANOWYTSCH, 2003) Kurmanowytsh, R.; Kirda, E.; Kerer, C.; Dustdar, S. "OMNIX: A Topology-independent P2P middleware". In: Proceedings of UMICS 2003 - Ubiquitous Mobile Information and Collaboration Systems, Klagenfurt, Austria, Jun. 2003. Disponível em: http://www.infosys.tuwien.ac.at/Staff/ck/files/umics2003.pdf. Última data de acesso: 18/05/2006.

(LAMMINEN, 2001) Lamminen, H. "Medical Applications and Technical Standardization of Teleconferencing". In: Doctor Dissertation, Faculty of Medicine of the University of Tampere. Disponível em: http://acta.uta.fi/pdf/951-44-5096-5.pdf. Última data de acesso: 18/05/2006.

(LOPES, 2004) Lopes, T.T.; Trautenmuller, P.; Hira, Y. A.; Zuffo, M. K. "Sistema de Teleradiologia para Diagnóstico de Imagem em Oncologia Pediátrica". In: Anais do IX CBIS – IX Congresso Brasileiro de Informática em Saúde, Novembro 2004. Disponível em: http://www.hu.ufsc.br/IX_CIBS/trabalhos/arquivos/409.pdf. Última data de acesso: 18/05/2006.

(MACIEL, 2005) Maciel, J. N.; Machado, R. B.; Wu, F. C.; Lee, H. D.; Fagundes, J. J.; Góes, J. R. V. "Proposta de um Modelo de Conferência Multimídia e Transmissão de Dados de Experimentos Médicos em Tempo Real pela Web". In: Anais da III JORNADA CIENTÍFICA DA UNIOESTE, 15 a 17 de junho de 2005 – Unioeste, PRPPG - Campus de Marechal Cândido Rondon – PR.. Disponível em: http://www.foz.unioeste.br/labi/documentos/III%20Jornada%20Cientifica/Proposta_Joylan.pdf. Última data de acesso: 18/05/2006.

(MANSSOUR, 1998) Manssour, I. H. "Visualização Colaborativa de Dados Científicos com Ênfase na Área Médica". In: Dissertação de Mestrado em Ciências da Computação, UFRGS – Porto Alegre. Disponível em: http://www.inf.pucrs.br/~manssour/Publicacoes/qualify.pdf. Última data de acesso: 18/05/2006.

(MUCHOW, 2002) Muchow, J.W. "Core J2ME Technology and MIDP". PT R Prentice-Hall, Englewood Cliffs, NJ 07632, USA, 2002, ISBN 0-13-066911-3.

(NOVAES, 2001) Novaes, M.; Barbosa, A. K.; Stamford, P.; Queiroz, A. E.; Morais, G.; Barros, D.; Belian, R.; Hedayioglu, F. "HealthNet: um Sistema Integrado de Telediagnóstico e Segunda Opinião Médica". In: Anais do III Workshop RNP2, 21 a 22 Maio de 2001. Disponível em: http://www.rnp.br/wrnp2/2001/palestras_aplicacao/res_aplic_17.pdf. Última data de acesso: 18/05/2006.

(ORAM, 2001) Oram, A. “Peer-to-Peer: Harnessing the power of distributive techologes”. By O´Relly, First Edition, Fev. 2001. ISBN: 0-596-00110-X.

(PATTICHIS, 2002) Pattichis, C.S.; Kyriacou, E.; Voskarides, S.; Pattichis, M.S.; Istepanian, R.; Schizas, C.N. “Wireless Telemedicine Systems: An Overview”, In: Jornal IEEE Antennas and Propagation Magazine, vol. 44, Nº.2, p143-153. Disponível em: http://technology.kingston.ac.uk/momed/papers/IEEEWirelessreview02.pdf. Última data de acesso: 29/03/2005.

(RICCI, 2002) Ricci, R. J. “Future of healthcare: 2012”. In: Technical Report IBM Healthcare Industry, p1-27. Disponível em: http://www.kana.com/pdf/Healthcare2012_F2.pdf. Última data de acesso: 07/08/2004.

(ROCHA, 2004) Rocha, J.; Domingues, M.; Callado, A.; Souto, E.; Silvestre, G.; Kamienski, C.; Sadok, D. “Peer-to-Peer: Computação Colaborativa na Internet”. In: Mini-curso XXII Simpósio Brasileiro de Redes de Computadores, SBRC2004 Gramado, RS, Maio 2004. Disponível em: http://www.cin.ufpe.br/~cak/publications/sbrc2004_minicurso_p2p.pdf. Última data de acesso: 22/05/2006.

(RUBEL, 2001) Rubel, P.; Gouaux, F.; Fayn, J.; Assanelli, D.; Cuce, A.; Edenbrandt, L.; Malossi, C. “Towards Intelligent and Mobile Systems for Early Detection and Interpretation of Cardiological Syndromes”. In: Jornal Computers in Cardiology, 2001, vol.28 pp193-196. Disponível em: http://tie.telemed.org/europe/citations2.asp?citation=11967&key=1487158608&page=1&pagecount=1. Última data de acesso: 18/05/2006.

(SABBATINI, 1999) Sabbatini, R. M. E. “Utilizando Teleconferência em Medicina”. In: Revista Médico Repórter, Outubro de 1999. Disponível em: http://www.sabbatini.com/renato/papers/reporter-medico-15.htm. Última data de acesso: 22/05/2006.

(SALEMA, 2003) Salema, C. “Infra-estruturas de Telecomunicações”. Chapter in A Casa do Futuro Interactiva , Fundação Portuguesa das Comunicações , Lisboa , 2003, Instituto Superior Técnico e Instituto de Telecomunicações. Disponível em: http://www.casadofuturointeractiva.com.pt/publico/ID39.pdf. Última data de acesso: 17/05/2004.

(SANTOS, 2004) Santos, C. A.; Neto, A. N. R. “Uma abordagem para Anotação em Vídeos Digitais com Aplicações em Telemedicina”. In: Anais do SBQS 2004 – Simpósio Brasileiro de Qualidade de Software. Disponível em: http://almerindo.devin.com.br/staticfiles/publicacoes/WIM2004-FINAL.pdf. Última data de acesso: 22/05/2006.

(SEABRA, 2003) Seabra, A. L. R. “Telemedicina”. Capítulo em Angiologia e Cirurgiua Vascular: guia ilustrado. Maceió: UNCISAL/ECMAL & LAVA; p1-13. Disponível em: http://www.lava.med.br/livro/pdf/seabra_telemedicina.PDF. Última data de acesso: 22/05/06.

(STOICA, 2001) Stoica, I.; Morris, R.; Karger, D. R.; Kaashock, M. F.; Balakrishman, H. "Chord: A scalable peer-to-peer lookup protocol for internet applications". In: Proceedings of the ACM SIGCOMM, pages 149-160, San Diego, California, Agosto de 2001.
Disponível em: http://delivery.acm.org/10.1145/390000/383071/p149-stoica.pdf?key1=383071&key2=8589083411&coll=GUIDE&dl=ACM&CFID=68489483&CFTOKEN=83024060. Última data de acesso: 22/05/2006.

(TEL AVIV, 1999) Declaração de Tel Aviv: http://www.dhnet.org.br/direitos/codetica/medica/27telaviv.html. Última data de acesso: 22/05/2006.

(TELES, 2004) Teles, C.P.; Castro, H. S. "Um Suporte de Transferência de Informações para Prontuários Eletrônicos: Experiência com JXTA". In: IX Congresso Brasileiro de Informática em Saúde (CBIS2004), Ribeirão Preto, São Paulo, Novembro de 2004. Disponível em: http://www.hu.ufsc.br/IX_CIBS/trabalhos/arquivos/720.pdf. Última data de acesso: 22/05/2006.

(TELERADIOLOGIA, 2005) Free DICOM Viewers and Radiology PACS Software Programs. Disponível em: http://www.rtstudents.com/pacs/free-dicom-viewers.htm. Última data de acesso: 22/05/2006.

(TSOUMAKOS, 2003) Tsoumakos, D.; Roussopoulos, N. "A Comparasion of Peer-to-Peer Search Methods". In: Sixth International Workshop on the Web and Databases (WEBD 2003), San Diego, California, June 2003. Disponível em: http://www.cs.umd.edu/~dtsouma/objects/webdb03.pdf. Última data de acesso: 22/05/2006.

(TRAVERSAT, 2003) Traversat, B.; Arora, A.; Abdelaziz, M.; Duigou, M.; Haywood, C.; Hugly, J. C.; Pouyoul, E.; Yeager, B. "Project JXTA 2.0 Super–Peer Virtual Network". In: White Paper Sun Microsystems. Disponível em: http://www.jxta.org/project/www/docs/JXTA2.0protocols1.pdf. Última data de acesso: 22/05/2006.

(TULU, 2005) Tulu, B.; Chatterjee, S. "A Taxonomy of Telemedicine Efforts with respct to Applications, Infrastructure, Delivery Tools, Type od Setting and Purpose". In: Proceedings of the 38th Hawaii Internacional Conference on System Sciences – 2005. Disponível em: http://csdl2.computer.org/comp/proceedings/hicss/2005/2268/06/22680147b.pdf.
Última data de acesso: 22/05/2006.

(WALDMAN, 2000) Waldman, M.; Rubin, A.; Cranor, L. "Publius: A Robust, Tamper-Evident, Censorship-Resistant Web Publishing System". In: 9a USENIX Security Symposium, 16th August 2000. Disponível em: http://www.cs.nyu.edu/~waldman/publius/publius.pdf. Última data de acesso: 22/05/2006.

(WILSON, 2001) Wilson, B. “JXTA”. By New Riders Publishing, ISBN: 0735712344, First Edition., Indianapolis – Indiana. (Book) Disponível em: http://www.brendonwilson.com/projects/jxta-book/. Última data de acesso: 22/05/2006.

(YAMIN, 2004) Yamin, A. C.; Barbosa, J. L. V.; Augustin, I.; Silva, L. C.; Real, R. A.; Geyer, C. F. R. ISAM: Uma Arquitetura de *Software* para *Pervasive Computing*. Artigo técnico, In: Anais do SBPL 2004 – 8º Simpósio Brasileiro de Linguagens de Programação, 26-28 de Maio 2004, Niterói RJ. Disponível em: http://sblp2004.ic.uff.br/papers/augustin-yamin-silva-real-geyer.pdf. Última data de acesso: 22/05/2006.

Apêndice A – Detalhamento das camadas da arquitetura de sistema JXTA

1. JXTA *core*

Todos os aspectos relacionados aos serviços e aplicações JXTA estão localizados nesta camada, com o propósito de tornar possível a interoperabilidade entre sistemas P2P JXTA pois, cada *peer* possui em comum o núcleo das funcionalidades de seus protocolos aqui implementados (WILSON, 2001; Arora, 2002). É importante saber também, que as principais funcionalidades desta camada estão relacionadas com quatro elementos-chave para comunicação P2P JXTA, que são:

- Canais de *peers* (*Peer Pipes*): são "túneis" de comunicação que torna possível a conexão de um *peer* a outro, bem como o compartilhamento distribuído de informações na rede;
- Grupos de *peers* (*Peer Groups*): organização dinâmica e flexível de grupos de *peers*, destinados aos trabalhos colaborativos entre seus componentes;
- Monitoramento de *peers* (*Peer Monitoring*): consiste da obtenção de informações sobre as interações entre *peers*, além do estabelecimento de normas de controle entre eles;
- Segurança (*Security*): sigilo da identidade de *peers* e do controle de acesso aos serviços oferecidos por um grupo.

Focando-se nos elementos relacionados acima, os protocolos JXTA disponibilizam os seguintes serviços à camada de Serviço (WILSON, 2001):

- Serviços de acesso (*Access Service*) – controle de acesso aos serviços e recursos disponíveis em grupos de *peers* (requisição de credenciais);
- Serviços de descoberta (*Discovery Service*) – permite a descoberta de *peers* existentes em grupos, além de *pipes* e serviços oferecidos por *peers*;
- Serviços de membro (*Membership Service*) – valida a entrada e saída de *peers* em grupos;
- Serviços de *pipe* (*Pipe Services*) – permitem a transferência de dados de forma síncrona e assíncrona;
- Serviços de Informação (*Resolver Service*) – usam anúncios (*advertisements*) como referências para unir *peers* a grupos, *pipes* e serviços.

2. JXTA *service*

A camada de serviços JXTA está posicionada acima da camada de núcleo, a fim de possibilitar a implementação de funcionalidades específicas a serem incorporadas em aplicações P2P JXTA. Segundo Wilson (2001), essa camada possui serviços que não são fundamentais para operação da rede JXTA, porém comuns ou desejáveis em qualquer ambiente desse tipo de rede. Dentro desse contexto, podemos exemplificar como serviços:

- Localização dos recursos disponíveis de um *peer*;
- Compartilhamento de documentos entre *peers*;
- Autenticação de *peer*;
- Sistemas de armazenamento e distribuição de arquivos.

3. JXTA Application

A camada de aplicação JXTA é construída sob a capacidade da camada de serviço com o propósito de ser suprida por funcionalidades que auxiliam a implementação de soluções P2P interoperáveis, como por exemplo: aplicações de mensagens instantâneas, compartilhamento de recursos, sistemas de processamento distribuídos e muitos outros.

Segundo Wilson (WILSON, 2001), o limite entre serviços e aplicações JXTA não é rigidamente definido, pois uma aplicação pode ser visualizada como fornecedora de serviço a outra. Como exemplo disso temos o JXTA *Shell*, que é uma aplicação desenvolvida inicialmente pela equipe da plataforma JXTA com o intuito de alcançar uma solução semelhante ao *Shell* original *Unix*. Ela é baseada em comandos não hierárquicos, especificados na camada de serviço, que permitem a operação e o controle de *peers* na rede por meio de um console. Isso possibilita sua utilização por outras aplicações, pois JXTA *Shell* são simples argumentos de comando que manipulam serviços a serem executados na rede P2P JXTA.

Apêndice B – Detalhamento da pilha de protocolos JXTA

1. Peer Discovery Protocol

Peer Discovery Protocol (PDP) é um protocolo desenvolvido para permitir o descobrimento de anúncios publicados por *peers* dentro de um *peer group*. Ele opera associado a um *peer group* e é implementado como um serviço de descoberta (*discovery service*). Seu trabalho é dividido em dois níveis: Local (dentro da *cache* dos *peers* requisitantes) e remoto (através de *peers* da rede), e seu método de chamada deve possuir os seguintes parâmetros (GRADECKI, 2002):

- Nome do elemento (usado como palavra chave durante a busca);
- O valor da palavra chave procurada;
- Um valor indicando o número máximo retornado de respostas desejadas.

O serviço de descoberta que implementa o PDP encapsula seus parâmetros dentro de um formato próprio de mensagem de descoberta (*PDP query message*). Os elementos para compor esse formato de mensagem são (GRADECKI, 2002):

- *Type* – o tipo do anúncio que será explicitamente procurado durante a requisição. Seus valores podem ser: *PEER*, *GROUP* e ADV (anúncio qualquer);
- *Threshold* – o número máximo de anúncios retornado por *peer*;
- *PeerAdv* – o anúncio do *peer* que solicitou a performance da requisição de descoberta;
- *Attr* – o nome do elemento que deve ser procurado nos anúncios;
- *Value* – o valor associado ao atributo *Attr*. Ele pode conter um caractere *wildcard* (*) que pode estar no início, no final ou em abas extremidades da string, para generalizar uma busca por anúncios que tenham partes de valores conhecidos.

Quando o PDP trabalha em nível remoto, o *peer* requisitante envia uma mensagem de requisição a outros *peers* da rede (inclusive ao *peer rendezvous*,) por meio do *Peer Resolver Protocol* (responsável pelo serviço de comunicação entre *peers* – esse protocolo será abordado posteriormente). Cada *peer* que recebe a *PDP query message* deverá examinar seus anúncios armazenados, comparando a palavra chave (*Attr*) com o valor procurado (*Value*). Caso a comparação se iguale com o valor procurado, uma outra mensagem (a PDP *Response Message*) será retornada ao *peer* requisitante com as informações procuradas. Os elementos dessa mensagem são (GRADECKI, 2002):

- *Type* – o tipo de anúncio retornado;
- *Count* – o número total de elementos da mensagem de resposta;
- *PeerAdv* – o anúncio do *peer* que retornou a resposta.
    - *Expiration* – um atributo do elemento que indica o número total de milisegundos úteis de expiração do anúncio retornado;
- *Attr* – elemento usado na busca do anúncio;
- *Value* – o valor procurado no elemento *Attr*;
- *Response* – um elemento contador, que indica o número total de anúncios que será enviado pelo *peer* que originou essa mensagem de resposta.
    - *Expiration* – um atributo do elemento de resposta que indica o total de número de milesegundos úteis de expiração do anúncio retornado.

Com relação ao trabalho do PDP em nível local, o serviço de descoberta irá checar na própria *cache* do *peer* por seus anúncios armazenados, no intuito de encontrar algum anúncio que corresponda à sua requisição. O *cache* local contém tanto anúncios enviados por outros *peers*, como anúncios criados pelo próprio *peer*. Dessa forma, o serviço de descoberta não precisa criar uma *query message* quando estiver trabalhando localmente na *cahe*; para isso, seria suficiente se realizar uma procura pelo anúncio desejado no *cache* local. Geralmente, o *cache* é mantido num diretório chamado "cm" localizado dentro do diretório "*root*", onde as aplicações dos *peers* JXTA são iniciadas (tal estrutura é criada em tempo de execução). Em sumo, o serviço que implementa o PDP deve realizar as seguintes atividades (GRADECKI, 2002):

1. Receber uma requisição de descoberta de uma aplicação, tanto local como remota;
2. Para a descoberta local, procurar na *cache* local do *peer* pelo *advertisement* que possui palavra chave e valor desejado;
3. Para descoberta remota, construir uma *query message*, e repassá-la para todos os *peers* conhecidos;
4. Armazenar as respostas da *query message* no *cache* local baseado no tipo de *advertisement*, e subseqüentemente armazena-las num diretório apropriado no disco rígido do *peer*.

Na Figura B.1 e B.2 abaixo estão ilustrados o modelo do formato XML usado para representar o *PDP Query Message* e o *PDP Response Message* , respectivamente.

```
<xs:element name="DiscoveryQuery" type="jxta:DiscoveryQuery"/>

<xs:complexType name="DiscoveryQuery">
    <!-- this should be an enumeration -->
    <xs:element name="Type" type="xs:string"/>
    <xs:element name="Threshold" type="xs:unsignedInt" minOccurs="0"/>
    <xs:element name="PeerAdv" type="jxta:PA" minOccurs="0"/>
    <xs:element name="Attr" type="xs:string" minOccurs="0"/>
    <xs:element name=vValue" type="xs:string" minOccurs="0"/>
</xs:complexType>
```

Figura B.1: formato XML do PDP *query message*.

```
<xs:element name="DiscoveryResponse" type="jxta:DiscoveryResponse"/>

<xs:complexType name="DiscoveryResponse">
    <!-- this should be an enumeration -->
    <xs:element name="Type" type="xs:string"/>
    <xs:element name="Count" type="xs:unsignedInt" minOccurs="0"/>
    <xs:element name="PeerAdv" type="xs:anyType" minOccurs="0">
        <xs:attribute name="Expiration" type="xs:unsignedLong"/>
    </xs:element>
    <xs:element name="Attr" type="xs:string" minOccurs="0"/>
    <xs:element name="Value" type="xs:string" minOccurs="0"/>
    <xs:element name="Response" type="xs:anyType" maxOccurs="unbounded">
        <xs:attribute name=vExpiration" type="xs:unsignedLong"/>
</xs:element>
</xs:complexType>
```

Figura B.2: formato XML do PDP *response message*.

2. *Peer Resolver Protocol*

Quando um *peer* necessita instanciar uma requisição à outro *peer* da rede, a especificação JXTA define um protocolo chamado *Peer Resolver Protocol* (PRP), com o propósito de um *framework* de camada de saída que manipula mensagens genéricas (requisição e resposta) de comunicação entre *peers*. Tal especificação não define a forma de implementar qualquer tipo de serviço de descoberta ou localização no *peer*, porém espera que tais serviços sejam construídos usando esse *framework* (o serviço disponível deve ser feito dentro de um grupo de peers) (GRADECKI, 2002).

Dentro do contexto acima, o PRP é um conjunto genérico de mensagens, perguntas e respostas, projetadas para facilitar o desenvolvimento de um sistema comum de comunicação entre *peers* na rede JXTA. Ele é fundamental para suportar requisições genéricas, pois permite que *peers* às definam para procura de suas informações de serviço. Tanto o *Peer Information Protocol* (PIP), quanto *Peer*

Discovery Protocol (PDP) usam o PRP para definir suas requisições específicas (o PIP é usado para uma requisição de informação de status e o PDP, como já foi visto, é usado para descobrir recursos de elementos da rede) (JXTA PROGUIDE, 2003).

A fim de reduzir o número de serviços que deve ser realizado durante o processo de comunicação, as mensagens são assinadas e entregues para um *handler* (escalonador) específico no peer. O *handler* é um nome assinado por uma definição que especifica o formato de uma mensagem, assim como, a resposta que pode ocorrer quando a mensagem desse tipo for recebida (GRADECKI, 2002). Por exemplo, um *handler* pode usar o serviço de *rendezvous* para propagar uma mensagem para múltiplos *peers*, ou para enviar a mensagem a um *peer* específico (depende de parâmetros definidos que devem ser filtrados pelo *handler*). Como se deve esperar, o serviço de descoberta possuirá a assinatura especificada para ele, ou seja, projetada para um *handler* processar requisições e mensagens de respostas específicas.

O PRP é implementado através de um *resolver service*, o qual deve encapsular uma mensagem em seu próprio formato definido, para que posteriormente seja enviada a outro *peer*. O *peer* remoto, ao receber a mensagem, despachará a mesma ao *handler* apropriado, o qual se encarregará de retornar com uma mensagem de resposta. Segue abaixo os elementos que compõem o formato da mensagem de requisição criada pelo resolver *service*, conhecida por *Resolver query message* (GRADECKI, 2002):

- *Credential* – credencial da pergunta do *peer*;
- *SrcPeerID* – identificador do *peer* que fez a pergunta;
- *HandlerName* – uma string que identifica o *handler*. Ela ao ser recebida pelo *resolver service* será usada para determinar qual processamento a ser aplicado;
- *QueryID* – o ID JXTA da requisição, que deve ser usado na mensagem de resposta para identificação da mesma;
- *Query* – uma string que representa a requisição feita pelo *peer*.

A seguir temos os elementos da mensagem de resposta processada pelo *resolver service* do *peer* requisitado:

- *Crendital* – credencial da pergunta do *peer*;
- *HandlerName* – a string que identifica o *handler*;
- *QueryID* – o ID JXTA da pergunta;

- *Response* – uma string que representa a resposta da requisição.

Através das considerações acima, todos os *peers* dentro de um mesmo *peer group* deverão usar o mesmo *resolver service* para processar um sistema comum de mensagens. Segundo a especificação JXTA é possível criar um grupo que não use a implementação do *resolver service* padrão, porém o estilo das requisições e respostas deverá ser o mesmo. Nas Figuras B.3 e B.4 a seguir, estão ilustradas a estrutura XML usada para representar o *Resolver Service query message* e *Resolver Service Response message*, respectivamente.

```
<xs:element name="ResolverQuery" type="jxta:ResolverQuery"/>

<xs:complexType name="ResolverQuery">
    <xs:element name="Credential" type="xs:anyType" minOccurs="0"/>
    <xs:element name="SrcPeerID" type="xs:anyURI"/>
    <!-- This could be extended with a pattern restriction -->
    <xs:element name="HandlerName" type="xs:string"/>
    <xs:element name="QueryID" type="xs:string" minOccurs="0"/>
    <xs:element name="Query" type="xs:anyType"/>
</xs:complexType>
```

Figura B.3: formato XML do *Resolver Service query message*.

```
<xs:element name="ResolverResponse" type="ResolverResponse"/>

<xs:complexType name="ResolverResponse">
    <xs:element name="Credential" type="xs:anyType" minOccurs="0"/>
    <xs:element name="HandlerName" type="xs:string"/>
    <xs:element name="QueryID" type="xs:string" minOccurs="0"/>
    <xs:element name="Response" type="xs:anyType"/>
</xs:complexType>
```

Figura B.4: formato XML do *Resolver Service response message*.

3. Peer Information Protocol

Em muitas aplicações voltadas para equipamentos interconectados numa rede, existe o desejo comum de um "nó" conhecer informações sobre os demais conectados, como por exemplo: o nome do *peer*, o tempo que o mesmo está ativo, quando enviou sua última mensagem e quantos dados processados têm sido transferido por ele. Nesse

contexto, a especificação JXTA inclui um protocolo chamado de *Peer Information Protocol* (PIP), responsável em manusear requisições de informação sobre *peers* remotos.

De forma semelhante aos outros protocolos JXTA, o PIP é implementado usando uma série de mensagens transmitidas entre dois ou mais *peers*. Dessa forma, para requisitar uma informação de um *peer* remoto, o PIP faz uso de um formato de mensagem por PIP *query message*. Os elementos dessa mensagem são(GRADECKI, 2002):

- *SourcePid* – o ID do *peer* requisitante;
- *TargetPid* – o ID do *peer* alvo ou de destino;
- *Request* – uma requisição opcional.

O elemento *request* fornece a forma base da mensagem, caso esse elemento seja vazio (*empty*), o *peer* de destino assumirá que o *peer* requisitante precisa somente da informação padrão, a qual é retornada baseada no formato da mensagem de resposta PIP (PIP *response message*). Caso o *peer* requisitante esteja interessado numa informação adicional, a mensagem de requisição poderá ser feita como sendo parte do elemento *request* (lembrando que a implementação do PIP deve ser capaz de processar a requisição da informação apropriadamente) (GRADECKI, 2002). Dessa forma, quando um *peer* requisitante recebe um PIP *response message*, o PRP repassa a mensagem ao *handler* apropriado, o qual se encarregará em processar a requisição. Os elementos desse formato de mensagem são (GRADECKI, 2002):

- *SourcePid* – o ID do *peer* requisitante;
- *TargetPid* – o ID do *peer* alvo ou de destino;
- *Uptime* – o tempo decorrido, em milesegundos, desde de que o serviço PIP iniciou a execução no *peer* remoto;
- *Timestamp* – tempo decorrido, em milesegundos, desde quando a mensagem de resposta foi criada;
- *Traffic* – um elemento adicional de informação sobre o tráfego gerado na rede por esse *peer*;
    - *LastIncomingMessageAt* – tempo decorrido, em milesegundos, desde que o *peer* remoto recebeu a mensagem passada;

- *LastOutgoingMessageAt* – tempo decorrido, em milesegundos, desde que o *peer* remoto enviou a última mensagem;
- *In* – um elemento adicional de informação de tráfego de chegada;
    1. *transport* – o número de bytes recebidos de um endereço de *endpoint* específico;
- *Out* – um elemento adicional de informação de tráfego de saída
    1. *transport* – o número de bytes enviados para um endereço de *endpoint* específico;
- *Response* – o elemento que contem opcionalmente informação baseada no elemento *request* correspondente da mensagem de pergunta.

4. Peer Endpoint Protocol

Todos os protocolos descritos até esse ponto são responsáveis por lidar com a manipulação e o ecapsulamento de mensagens, com o propósito de transportá-las para outros *peers*. Sendo que nenhum desses protocolos é capaz de executar o transporte dessas mensagens.

O *Peer Endpoint protocol* (PEP) é responsável em determinar a rota entre *peers* numa rede JXTA. Ele fornece uma visão entre dois *peers* que não estão ligados diretamente um com o outro, como se eles possuíssem uma conexão direta. Isso faz com que a rede JXTA pareça ser uma topologia de rede muitos-para-muitos. Também conhecido por *Endpoint Routing Protocol* (ERP), o PEP define um conjunto de mensagens de requisições e respostas usadas para encontrar informações de rota. Elas são (GRADECKI, 2002; JXTA PROGUIDE, 2003):

- *Ping query* – mensagem para determinar se existe uma rota entre dois *peers*;
- *Ping response* – mensagem de resposta usada quando um *peer* recebe um *ping query*. Ela retorna ao *peer* de origem a rota conhecida entre os *peers*;
- *Route query* – mensagem usada quando um *peer* necessita se comunicar com outro *peer*, cuja rota é desconhecida. Nesse caso, ele envia a mensagem de *route query* para os *peers* que estão conectados diretamente com ele procurando por uma rota para o *peer* desejado;

- *Route response* – mensagem usada para quando um *peer* possui a rota de um *peer* desejado. Em outras palavras, é uma mensagem de rota de resposta contento à solução da mensagem de requisição de rota.

Cada uma das mensagens de requisição e resposta contém os seguintes elementos (GRADECKI, 2002):

- *Source* – o endereço de *endpoint* do *peer* de origem da mensagem;
- *Destination* – o endereço de *endpoint* de um potencial *peer* de destino;
- *LastHop* – o endereço de *endpoint* do último *router* de contato de mensagem;
- *NbOfHop* – o número total de *peers* que a mensagem teve contato;
- *ForwardRoute* – uma lista opcional de *strings* de endereço de *endpoints* que o *peer* pode precisar enviar de modo que uma mensagem chegue ao *peer* de destino;
- *ReverseRoute* – uma lista opcional de *strings* de endereço de *endpoints* que o *peer* pode usar para responder a mensagem enviada por um *peer* que a originou.

Partindo das mensagens descritas acima, quando um *peer* é requisitado para identificar um *endpoint* de um *peer*, ele primeiro procura no seu *cache* local, verificando se possui a rota do *peer* alvo; caso contrário enviará uma requisição de rota (*resolver query*) aos *peers relays*, por ele conhecido. Quando o *peer relay* recebe a *route query*, ele checa se conhece a rota para o *peer* procurado, e retorna a informação de rota como uma enumeração de nós (caso saiba a rota) (JXTA ProgGuide, 2001). Neste contexto, quando uma mensagem estiver pronta para ser enviada de um *peer* a outro, o *endpoint service* precisará de uma origem e um destino em forma de um ID; o ID é construído na forma a seguir (GRADECKI, 2002):

```
protocol://address_as_per_protocol/unique_name_of_recip
ient/unique_name_in_recipient_context .
```

Baseado no valor da string “protocol”, o *endpoint service* invoca um *endpoint protocol*. O valor do campo acima, “address_as_per_protocol”, representa o endereço da máquina que a mensagem está sendo enviada (para o protocolo TCP esse valor pode ser tipicamente um endereço IP). Quando a mensagem alcança a máquina de destino, um *handler* é invocado baseado na concatenação do valor “unique_name_of_recipient” e do “unique_name_in_recipient_context” (GRADECKI, 2002).

O *endpoint service* ou serviço de *endpoint* é o principal serviço da rede JXTA, pois fornece funcionalidades de transporte entre *peers* usando os protocolos de transporte da rede física, ou seja, ele é responsável em implementar o PEP. Os protocolos de transporte JXTA atuais são os seguintes (GRADECKI, 2002): *HTTP*, *TCP, TLS, Beep* e *ServletHttp*.

Tais protocolos de transporte descritos acima são responsáveis pela comunicação de baixo nível sobre canais específicos. O PEP é projetado para fornecer um canal de comunicação entre um *peer* e possíveis diversos outros *peers*. Dessa forma, ele é localizado na camada mais inferior dos protocolos JXTA, acima da camada de transporte da rede (GRADECKI, 2002).

5. Peer Binding Protocol

Peer Binding Protocol (PBP) é um dos protocolos JXTA mais utilizados, pois é responsável em criar e administrar *pipes* ou canais de comunicação virtual entre *peers* (um *pipe* é uma abstração de um canal de comunicação construído no topo dos protocolos de transporte da Internet, como por exemplo: HTTP ou TCP).

O PBP define um *pipe*, e especificações de como esse *pipe* pode ser usado para estabelecer comunicação com outros *peers* dentro de um *peer group*. Para facilitar a transferência de informações para *peers*, o PBP é construído no topo do *Peer Endpoint Protocol* (PEP). Ele usa o *Peer Resolver Protocol* para obter o endereço de um peer para outro. O PRP usará o *Rendezvous protocol*, se caso ele necessite propagar a mensagem para um *peer* remoto, e usará o *Peer Endpoint Protocol* para estabelecer comunicação mais direta com outro *peer* (GRADECKI, 2002).

Dentro da especificação JXTA, o PBP é considerado um serviço de alto nível, e existe com o propósito de permitir que mensagens sejam passadas de um *peer* para outro. Ele é implementado como um serviço de *pipe* (*pipe service*), e é obtido através de um *peer group*, do qual o *peer* for membro naquele momento. Esse serviço torna possível a criação de *input* e *output pipes*, além de objetos de mensagens a serem enviados através de um output *pipe*. Para usar um *pipe*, um anúncio deve ser publicado informando todas as características necessárias que descreva o mesmo para que outros *peers* possam utiliza-lo. Os elementos desse anúncio são (GRADECKI, 2002):

- *Name* – um nome opcional usado pelo *pipe*;
- *ID* – um JXTA ID do *pipe*;
- *Type* – tipo do *pipe*, que pode ser: *JxtaUnicast* (para *pipe Unicast*), *JxtaUnicastSecure* (para *pipe SecureUnicast*), ou *JxtaPropagate* (para *pipe propagate*).

O PBP tem que conectar dois *peers* usando um *pipe*. Nesse contexto, temos um *peer* que deverá criar um input *pipe*, incluindo seu anúncio, e outro *peer* que deverá descobrir esse anúncio para estabelecer conexão. Isso porque esse último *peer* não pode assumir que o primeiro já possui um *pipe* construído pronto para comunicação. Dessa forma, ele precisa enviar um *pipe* resolver *message* ao *peer host* (*peer* que criou o *input pipe*). Os elementos do formato dessa mensagem são (GRADECKI, 2002):

- *MsgType* – um valor para identificar requisição ou resposta. O valor usado para identificar uma requisição ocorre quando um *peer* remoto inicialmente envia uma mensagem ao *peer host*, e o valor referente a uma resposta ocorre quando o *peer host* responde a pergunta do *peer* remoto;
- *PiepID* – o JXTA ID do *pipe*;
- *Type* – o tipo do *pipe*: *JxtaUnicast, JxtaUnicastSecure,* ou *JxtaPropagate*;
- *Cached* – elemento assume valor verdade ou falso. O valor falso referencia uma resposta que não deve vir da *cache* do *peer* remoto, ou seja, deve ser provida baseado na captura de dados;
- *Peer* – o *peer ID* do *peer host*, o qual deve responder a requisição. Caso o valor não esteja presente e o valor do elemento *Cached* seja verdade, um *peer* intermediário pode responder a pergunta;
    - *Found* – valor verdade indica que o *pipe* foi encontrado no *peer* remoto;
    - *PeerAdv* – o *advertisement* do *peer* do *pipe*.

Uma vez o *input pipe* criado e seu anúncio publicado na rede, *peers* remotos poderão descobrir esse anúncio para construir um *output pipe* através do *pipe service*, o qual é responsável enviar mensagens de requisição a um *peer hopst*, que pode ter ou não um input *pipe* disponível. O *pipe service* do input *pipe* do *peer host* irá processar essas requisições e enviar uma mensagem de resposta ao *peer* requisitante. Caso a resposta reportada sobre input *pipe* confirme sua existência, então um *output pipe* será criado

para esse input *pipe* e as informações poderão ser transmitidas entre os *peers* diretamente (GRADECKI, 2002). Na Figura B.5 a seguir, está ilustrado a estrutura XML do anúncio de um *pipe*.

```
<?xml version="1.0"?>

<!DOCTYPE jxta:PipeAdvertisement>

<jxta:PipeAdvertisement xmlns:jxta="http://jxta.org">
      <Id>
            urn:jxta:uuid-
59616261646162614E5047205032503331BE4F9EC1D941BBBDA18B3273791A9D04
      </Id>
      <Type>
            JxtaPropagate
      </Type>
      <Name>
            bipipe2.end1
      </Name>
</jxta:PipeAdvertisement>
```

Figura B.5: formato XML do *Advertisement* de um *pipe*.

6. Peer Rendezvous Protocol

O *Peer Rendezvous Protocol* (RVP) é responsável em propagar mensagens dentro de um grupo de *peers*. Ele foi criado para facilitar propagação de mensagens para múltiplos *peers* dentro da rede JXTA. Seus mecanismos de transporte, utilizados como referência na especificação JXTA, são: HTTP e TCP/IP, ou seja, correspondem à mecanismos que não foram projetados para enviar informações de uma máquina para diversas outras. Dessa forma, o transporte é basicamente um-para-um (com exceção do mecanismo *multcast* TCP/IP) (GRADECKI, 2002).

O RVP é usado pelo *Peer Resolver Protocol* e pelo *Peer Discovery Protocol*, na ordem de propagar mensagens. Quando um *peer* usa o *discovery service* para encontrar um anúncio na rede JXTA, o serviço utiliza o resolver *service* implementado com base no *Peer* Resolver *Protocol*. O *resolver service* terá que escolher usar entre o *Peer Endpoint Protocol* ou *Rendezvous Protocol*, ou ambos. Ao usar o RVP, o *discovery service* irá ganhar uma audiência muito mais ampla para publicar ou descobrir eventos**,** pois se reportará à peers *rendezvous* da rede, que são *peers* com a função de propagar mensagens por eles recebidas. A seguir temos os parâmetros definidos para compor o anuncio de um *peer* configurado como Rendezvous (GRADECKI, 2002):

- *Name* – um nome opcional para o *peer rendezvous*;
- *RdvGroupID* – o *peer group id* que o *rendezvous* está associado;
- *RdvPeerID* – o *peer id* do *peer rendezvous*.

Os parâmetros usados para compor uma mensagem de propagação a ser enviado por um rendezvous são:

- *MessageID* – um JXTA ID associado com a mensagem;
- *DestSName* – uma string representando o nome do destino;
- *DestSParam* – um string representando qualquer parâmetro associado ao destino;
- *TTL* – um inteiro representando o tempo de vida da mensagem (*time-to-live integer*);
- *Path* – a *URI* onde a mensagem já foi visitada.

Apêndice C – Principais elementos da rede JXTA

1. Peers JXTA

Segundo Wilson (Wilson, 2001), um *peer* é uma entidade da rede P2P que realiza algum trabalho útil e que é capaz de comunicar os resultados desse trabalho à outras entidades da rede de forma direta ou indireta. Dessa forma, cada *peer* da rede deve operar independentemente e assincronamente de todos os outros *peers*, além de ser identificado unicamente por um *peerID*, ou um identificador que o distingue dos demais *peer* na rede.

Um *peer* JXTA se diferencia da idéia anterior, ao ter-se adicionado no conceito de um *peer* a necessidade de implementar uma ou mais funcionalidades do núcleo de protocolos JXTA. E dependendo dessas funcionalidades acrescidas, eles podem ser classificados em três tipos de *peers* (JXTA ProGuide, 2003):

- *Peers Edge* – são *peers* geralmente projetados para atender as necessidades de usuários finais da rede, permitindo o consumo de serviços e recursos disponibilizados por outros *peers*, ou seja, são *peers* que utilizam serviços fornecidos por outros, e quando ofertam serviços os fazem através da troca de mensagens. Na rede P2P essa classe de peers é caracterizada como instável, por abandonarem e entrarem na rede de forma aleatória, e na rede física estão normalmente posicionados por trás de *firewalls*, e dessa forma não participam no roteamento das mensagens, tarefa essa de responsabilidade de outra categoria de *peers*;
- *Peers Rendezvous* – são peers que possuem as mesmas funcionalidades básicas de um *peer edge*, porém participam em atividades adicionais, como por exemplo: roteamento de mensagens. *Peer rendezvous* são considerados mais estáveis numa rede P2P, pois são utilizados para realizar a descoberta de outros *peers* e recursos disponíveis na rede, além de serem responsáveis em propagar mensagens e armazenar informações de outros *peers* para uso futuro, ou para requisições de outros *peers Rendezvous*. Essa categoria de *peers* encontra-se posicionada em redes internas, e podem servir de ponte de comunicação com outras redes, mas não possui a capacidade de atravessar *firewalls*, ficando isso a cargo da categoria de *peers Relay*;

- *Peers Relay* – são *peers* que operam na rede JXTA com a função de rotear mensagens entre peers de diferentes redes protegidas por firewalls e NAT, ou seja, são responsáveis por entregar mensagens a *peers* pertencentes à diferentes redes separadas por barreiras de proteção.

2. Peers Group

Uma das características importantes pertencentes a redes P2P, diz respeito à habilidade de seus elementos interligarem-se dinamicamente formando uniões cooperativas, ou seja, interligarem-se para trabalhar cooperativamente em grupos de colaboração específicos, conhecidos por *Peers Group* (JXTA ProGuide, 2003). Por meio da partição lógica de uma rede física, a tecnologia JXTA permite a criação desses grupos de *peers*, a fim de seus componentes trabalharem em prol de interesses comuns.

Na rede JXTA um *Peer Group* é uma coleção de *peers* que compartilham um conjunto comum de serviços e recursos, estabelecendo suas próprias políticas de segurança e acesso, disponibilizando serviços específicos para membros do grupo (JXTA ProGuide, 2003). Dessa forma, não se limitam a união de *peers* pertencentes a uma única rede física, mas podem ser compostos por elementos de redes distintas, possibilitando o compartilhamento de serviços entre eles através de procedimentos acordados pelo grupo. Os protocolos JXTA descrevem a maneira com que os *peers* devem publicar, descobrir, entrar e monitorar grupos de *peers*; eles não ditam quando, onde, e por quê razão grupos de *peers* devem ser criados (Brookshier, 2002). A especificação JXTA ilustra três situações comuns para que esses grupos sejam formados, são elas (Wilson, 2001):

- para criar um ambiente que envolva aplicações especializadas – um grupo de *peers* é formado para trocar serviços e informações através de aplicações que possuem as mesmas características de funcionamento (baseados num domínio lógico de especialização de serviços). Como exemplo, podemos citar: aplicações para troca de mensagens ou que implementem uma rede de compartilhamento de documentos;
- para criar um ambiente de Segurança – um grupo pode exigir autenticação dos usuários para restringir o acesso e a utilização de seus serviços, ou seja, objetiva formar uma região lógica para limitar o acesso à conteúdos protegidos por meio da aplicação de políticas de segurança. Essas políticas podem ser simples, tal como, a

troca de textos verificando a senha e o *login*, ou mais sofisticados envolvendo a criptografia de chaves públicas;

- para criar um ambiente de monitoração – um grupo pode se organize a fim de se executar um propósito específico, porém para isso, necessitam do monitoramento de seus componentes. Por exemplo, no projeto SETI@Home, (rever secção 2.5.3.1) a informação de *status* de um *peer* é usada para manter um nível mínimo de serviço a ser processado.

Os grupos de *peers* JXTA são formados dentro de sua rede virtual e são identificados unicamente por um *PeerGroup Id*, para que seja possível distinguí-los uns dos outros. Nesse contexto, Os peers JXTA se auto-organizam nesses grupos, podendo eles pertencer a mais de um grupo simultaneamente, porém por padrão, todo *peer JXTA* deve ser instanciado primeiramente no *NetPeerGroup*. Isso se deve ao fato da plataforma JXTA definir dois tipos de grupos de *peers* (JXTA ProGuide, 2003):

- *NetPeerGroup* – é o grupo padrão, onde todo *peer* automaticamente deve ser adicionado após ter sido iniciado na rede JXTA (também conhecido por *World PeerGroup*). Esse grupo é o *root* de todos os outros grupos, ele define o escopo inicial de operações e serviços padrões de que *peers* JXTA necessitam;]
- *User PeerGroups* ou grupos de usuários – são os novos grupos criados por *peers* com seus próprios conjuntos de serviços e regras de segurança. Eles são criados através de uma das duas formas: estendendo os serviços do *NetPeerGroup*, ou por grupos gerados a partir do *NetPeerGroup*. Dessa forma, todos esses grupos são sub-grupos do *NetPeerGroup*, e isso nos fornece a idéia de que cada grupo é criado hierarquicamente dentro de outros grupos.

3. Endpoint

Segundo Brookshier (Brookshier, 2002), *endpoint* é o método de endereçamento básico usado para comunicação entre *peers* JXTA. Um *endpoint* é o endereço de um *peer* que implementa um mecanismo de transporte de rede específico, e dessa forma, cada *peer* pode possuir múltiplos *enpoints*, além de estabelecer comunicação através de diferentes protocolos.

Dentro do contexto acima, o endereço de um *peer* deve ser definido não necessariamente por um endereço físico, pois esses podem ser alterados. Eles estão

diretamente relacionados com canais de comunicação virtual estabelecido entre *peers* da rede JXTA. Um exemplo simples de um *endpoint* é um endereço IP associado com uma porta.

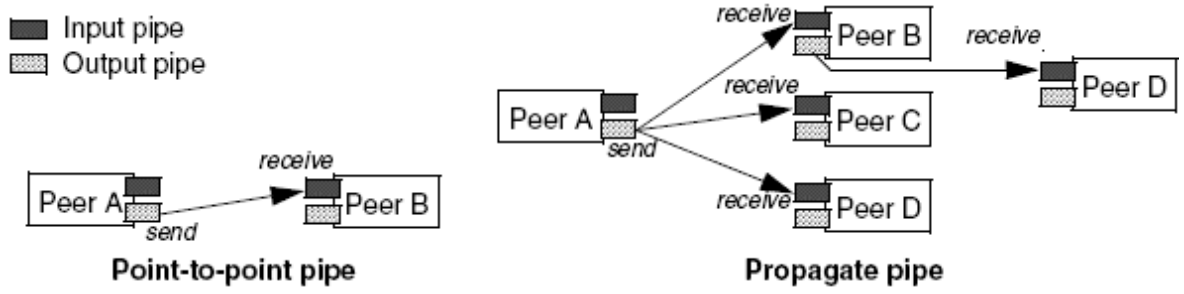
4. Pipes

Pipes definem um recurso básico e importante da tecnologia JXTA, pois possibilitam a comunicação entre *peers* desprezando a existência de *firewalls* ou de outras barreiras existentes na topologia da rede física. Normalmente, pensamos que a comunicação P2P é estabelecida por meio de uma simples conexão, e esquecemos da existência do problema de que muitos *peers* não podem se conectar diretamente uns aos outros por se tratarem de equipamentos pertencentes à redes protegidas.

Segundo a especificação JXTA, *pipe* é um canal de conexão virtual utilizado para troca de contexto entre serviços ou aplicações. Eles fornecem ao usuário de um *peer* a ilustração de uma entrada e saída virtual para uma caixa de correio eletrônico. Eles são utilizados também para conectar *peers* que não possuem link físico direto; nesse caso, um ou mais *endpoints* intermediários são usados para rotear mensagens entre dois *pipes*. Dessa forma, os *pipes* fornecem uma abstração da rede sobre o *endpoint* de transporte do *peer*, sendo designados como uma camada sobre múltiplos protocolos de comunicação de rede (Brookshier, 2002).

De acordo com a especificação JXTA, existem dois tipos de *pipes* (JXTA SPEC, 2004):

- *Point-to-Point* – conecta exatamente dois *pipes endpoints*, um *Input Pipe* (por onde um *peer* recebe dados) e um *Output Pipe* (por onde um *peer* envia dados);
- *Propagate* – conecta um *Output pipe* à múltiplos *Input pipes*.



FiguraC.1: p*ipes* de comunicação *point-to-point* e *propagation.*

Conforme está ilustrado na Figura C.1, cada *peer* da rede JXTA, para estabelecer comunicação com outros *peers* pertencentes ao mesmo grupo, deve possuir pelo menos dois *pipes endpoints*: um para receber informações da rede (*input pipe*), e um outro para enviar informações à elementos da rede (output pipe). Geralmente, os anúncios de *input pipe* de um *peer* devem ser publicados na rede para conhecimento dos demais, pois tal anúncio é necessário para que um *output pipe* de um outro *peer* seja criado para estabelecer um canal de comunicação entre eles; isso requer a existência de um emissor e pelo menos um ouvinte.

Além dos tipos de *pipes* descritos anteriormente, a tecnologia JXTA também provê o *secure unicast pipe*, uma variação segura do *point-to-point pipe* que fornece um canal de comunicação seguro. A partir da existência desses diferentes *pipes*, tipos adicionais podem ser construídos a partir do núcleo básico desses *pipes* fornecidos pela plataforma *JXTA*, como por exemplo: *bidirectional pipes* e *bidirectional/reliable pipes* (JXTA ProGuide, 2003).

5. Message

As informações transmitidas através dos *pipes* de comunicação são encapsuladas como *messages* ou mensagens JXTA. Elas são objetos enviados entre *peers JXTA* para transferir qualquer tipo de dado. Dessa forma, elas constituem a unidade básica de troca de dados entre *peers* (JXTA ProGuide, 2003).

Existem duas formas de representar uma mensagem: pode ser através de estruturas XML, ou por meio de uma representação binária. A representação binária foi adotada pela plataforma JXTA para encapsular as mensagens, mas nada impede que se utilizem arquivos XML para facilitar a comunicação com diversas aplicações. Normalmente, as mensagens XML são usadas por mecanismos de transporte que suportam apenas texto. Seu formato consiste de uma *tag* de mensagem que encapsula um dado ou uma mensagem. Cada elemento possui um nome, um tipo de dado, e um parâmetro de codificação opcional (Brookshier, 2002)**.**

6. Advertisement

Todos os recursos da rede JXTA tais como, *peers*, *peers groups*, *pipes* e *services*, são representados por um anúncio JXTA, que são documentos XML utilizados para

abstrair informações sobre um determinado recurso disponível na rede (Gradecki, 2002). Cada anúncio é publicado com um tempo de vida útil, que especifica a disponibilidade de seu recurso associado. Esse tempo de vida possibilita a exclusão de recursos obsoletos na rede JXTA, sem a necessidade de qualquer controle centralizado. Os principais *advertisements* descritos pela especificação JXTA são (JXTA ProGuide, 2003):

- *Peer Advertisement* – descreve os recursos de um *peer*. O uso principal desse anuncio objetiva fornecer informações sobre um *peer*. Por exemplo: seu nome, *peer ID*, *endpoints* disponíveis, e qualquer atributo de tempo real, que deve ser publicado individualmente para um serviço do grupo (exemplo: a existência de um *peer rendezvous* para um grupo);
- *Peer Group Advertisement* – descreve recursos específicos de um *peer group*, tais como, nome, *peer group ID*, descrição, especificação e parâmetros de serviço;
- *Pipe Advertisement* – descreve canais de comunicação JXTA. Eles são utilizados para criar *pipes endpoints* (*input pipe* e *output pipe*). Cada núncio de *pipe* possui um nome, um *ID* simbólico que identifica o tipo de *pipe* (*point-to-point*, *propagate*, *secure* e outros) e um *pipe ID*.

Outros tipos de *advertisements* estão associados a *modules*. Segundo Brookshier (Brookshier, 2002), *modules* são definições de serviços e aplicações disponíveis por um *peer* ou *peer group*. Ele é definido em três anúncios:

- *Module Class* – define comportamentos ou procedimentos propostos principalmente para documentar formalmente a existência de um module. *Peer groups*, *peer* e outros *advertisements* referenciam um module *class ID*, que é definido por um module *class advertisement*. Esse anúncio possui os seguintes parâmetros: module *class ID (MCID)*, nome e descrição usada para localização e identificação do module;
- *Module Specification* – define a especificação de um *module class*. Esse anúncio contém informações sobre uma implementação classificada por um module *specification id*. Ela provê referencia a documentações necessárias na ordem de criar implementações conforme sua especificação. Seus *tags* são: nome, descrição, module *spec id (SID)*, *pipe advertisement*, e campos de parâmetros contendo informações a serem interpretadas por cada implementação;

- *Module Implementation* – define uma implementação dada por um module *specification*. Ele inclui nome, *ModuleSpecID* associado, como também o código, pacote e campos de parâmetros que possibilitam um *peer* recuperar dados necessários para executar uma implementação. Esse anúncio é usado para carregar o código.

7. Service

Os serviços fornecem funcionalidades para realização de qualquer trabalho útil dentro de uma rede *P2P*. Esses trabalhos úteis podem ser desde transferência de arquivos, prover informações de status, executar cálculos ou realizar qualquer tarefa que seja de proveito para outro *peer* (Wilson, 2001). Eles podem ser pré-instalados em *peers*, ou carregados a partir da rede. Nesse caso, o processo de localização, *donwloading* e instalação do serviço da rede, é similar à execução de procura de uma página *web* na *Internet*.

Segundo Brookshier (Brookshier, 2002), os protocolos JXTA organizam dois níveis de serviço de rede, são eles:

- *Peer Services* – um serviço oferecido por um *peer* da rede aos outros *peers*. A funcionalidade deste tipo de serviço é provida por um único *peer*, e dessa maneira só estará disponível quando esse estiver conectado na rede. Múltiplas instancias de um serviço pode ser executada em diferentes *peers*, mas cada instância deve ser anunciada através de seu próprio *advertisement*;
- *Peer Group Services* – funcionalidade oferecida por um *peer group* aos seus *peers* participantes. Neste caso, quem oferece o serviço são vários *peers* de um grupo e não somente um, formando uma coleção de instâncias do serviço sendo executado em múltiplos *peers* membros do *peer group*. Esses serviços são redundantes, cooperando potencialmente uns com os outros para garantir a funcionalidade do serviço no grupo. Dessa forma, mesmo que alguns *peers* deixem a rede o serviço não será afetado.

A plataforma JXTA fornece uma gama de serviços básicos associados à grupos de *peers* (Gradecki, 2002):

- *Discovery service* – usado por membros de um grupo para realizar pesquisa dos recursos disponíveis no grupo, tais como, *peers*, *peer groups*, *pipes* (serão abordados no próximo tópico) e serviços;
- *Membership service* – usados pelos membros correntes do grupo para aceitar ou rejeitar a solicitação da entrada de um novo membro ao grupo. Esse serviço pode fornecer o voto de *peers* ou eleger um representativo grupo designado para aceitar ou rejeitar um novo membro;
- *Access Service* – usado para validar requisições feitas de um *peer* para outro. Um *peer* ao receber uma requisição, verifica os direitos de acesso do *peer* sobre a requisição, objetivando determinar se o acesso é permitido. Nem todas as ações em grupos de *peers* necessitam ser checadas com o *Access service*;
- *Pipe service* – usado para criar e gerenciar conexões entre os *peers* membros de um grupo;
- *Resolver service* – usado para enviar requisições genéricos para outros *peers*. *Peers* podem definir e trocar *querys* para encontrar qualquer informação que possa ser necessário. Por exemplo: o status de um serviço ou o estado de um *pipe endpoint*;
- *Monitoring service* – usado para permitir que um *peer* monitore outro membro de um mesmo *peer group*.

Os serviços citados acima não precisam ser obrigatoriamente implementados por todos os *peer groups*.

8. ID JXTA

Num sistema P2P todos os recursos devem ser referenciados de alguma maneira diferente de um simples nome, pois dois ou mais recursos podem ter nomes idênticos. Para resolver esse problema, a plataforma JXTA considerou a necessidade de um JXTA ID, que são utilizados para distinguir entidades da rede JXTA. Eles são expressos através de URNs (*Uniform Resource Names*), sendo representado na forma de um arquivo texto em XML (Wilson, 2001).

Um JXTA ID consiste de três partes (Gradecki, 2002):

- Namespace Identifier – jxta (not case-sensitive);

- Format specifier – urn (not case-sensitive);
- ID – valor único (*case-sensitive*).

A Figura C.2 abaixo ilustra ABNF (*Augmented Backus-Naur Form*) que pode ser usado para especificar um JXTA ID.

```
<JXTAURN>     ::= "urn:" <JXTANS> ":" <JXTAIDVAL>
<JXTANS>      ::= "jxta"
<JXTAIDVAL>   ::= <JXTAFMT> "-" <JXTAIDUNIQ>
<JXTAFMT>     ::= 1 * <URN chars>
<JXTAIDUNIQ>  ::= 1 * <URN chars>
<URN chars>   ::= <trans> | "%" <hex> <hex>
<trans>       ::= <upper> | <lower> | <number> | <other> |
                  <reserved>
<upper>       ::= "A" | "B" | "C" | "D" | "E" | "F" | "G" | "H" |
                  "I" | "J" | "K" | "L" | "M" | "N" | "O" | "P" |
                  "Q" | "R" | "S" | "T" | "U" | "V" | "W" | "X" |
                  "Y" | "Z"
<lower>       ::= "a" | "b" | "c" | "d" | "e" | "f" | "g" | "h" |
                  "i" | "j" | "k" | "l" | "m" | "n" | "o" | "p" |
                  "q" | "r" | "s" | "t" | "u" | "v" | "w" | "x" |
                  "y" | "z"
<hex>         ::= <number> | "A" | "B" | "C" | "D" | "E" | "F" |
                  "a" | "b" | "c" | "d" | "e" | "f"
<number>      ::= "0" | "1" | "2" | "3" | "4" | "5" | "6" | "7" |
                  "8" | "9"
<other>       ::= "(" | ")" | "+" | "," | "-" | "." |
                  ":" | "=" | "@" | ";" | "$" |
                  "_" | "!" | "*" | "'"
<reserved>    ::= "%" | "/" | "?" | "#"
```

Figrua C.2: ABNF para especificar JXTA ID.

Segundo Gradeck (2002), existem três IDs reservados na especificação JXTA. São eles (ABNF para esses IDs estão ilustradas na figura C.3):

- NULL ID;
- World Peer Group ID;
- Peer Group ID ou *NetGroup*;

```
<JXTAJXTAURN>     ::= "urn:" <JXTANS> ":" <JXTAJXTAFMT> "-"
                      <JXTAJXTAFMTID>

<JXTAJXTAFMT>     ::= "jxta"

<JXTAJXTAFMTID>   ::= <JXTANULL> | <JXTAWORLDGROUP> | <JXTANETGROUP>

<JXTANULL>        ::= "Null"

<JXTAWORLDGROUP>  ::= "WorldGroup"

<JXTANETGROUP>    ::= "NetGroup"
```

Figura C.3: ABNF para especificar JXTA ID Reservados.

9. JXTA para J2ME – JXME

JXME é um projeto mantido pela comunidade JXTA, com o propósito de fornecer funcionalidades compatíveis com a tecnologia JXTA a dispositivos com recursos limitados, que utilizam o CLDC (*Connected Limited Device Configuration*) e MIDP (*Mobile Information Device Profile*). Usando JXME, qualquer dispositivo MIDP será capaz de participar em atividades P2P com outros *peers* da rede JXTA (JXTA-J2SE, JXTA-C e JXTA-J2ME).

As principais características que direcionam o desenvolvimento da tecnologia JXME são (JXME, 2005):

- Criar uma infra-estrutura P2P adequada aos dispositivos móveis;
- Proporcionar interoperabilidade com JXTA em plataformas como estações de trabalho e servidores;
- Ser simples e fácil de utilizar por desenvolvedores;
- Ser pequeno o suficiente para ser utilizado nos dispositivos que implementam J2ME (a grande maioria deles sendo telefones celulares e *PDAs* simples);
- Ser compatível com o MIDP (perfil mais utilizado hoje em dia em telefones celulares).

A primeira implementação JXME, baseada em *Proxy* foi desenvolvida sobre a plataforma CLDC e MIDP 1.0 (J2ME). Devido às características desta plataforma serem direcionadas para dispositivos com recursos limitados (processadores de 16bits com 160KB de memória, com suporte à bibliotecas *Java* limitadas, inexistência de um *parser* XML nativo, inexistência de suporte HTTP *listener*, e inexistência de suporte a segurança), a implementação JXME permite que *peers* na plataforma J2ME atuem apenas como *edge peers* (consultar Apêndice B). A estratégia atual do projeto é atualizar a implementação, removendo a dependência da comunicação com um *peer relay*, e atualizá-lo para ser totalmente compatível com os protocolos JXTA 2.x (versões em C e J2SE) (JXME,2005).

Devido às limitações descritas anteriormente, um *peer* JXME para CLDC/MIDP não pode executar as mesmas tarefas que *peers* mais sofisticados, ou seja, eles não podem oferecer serviços para outros membros do grupo, ou mesmo executar tarefas que

exigem um processamento mais intensivo, como a busca por recursos na rede. Dessa forma, para esse tipo de *peer* participar na rede JXTA é necessário utilizar serviços de outros *peers* que atuam como *relays*.

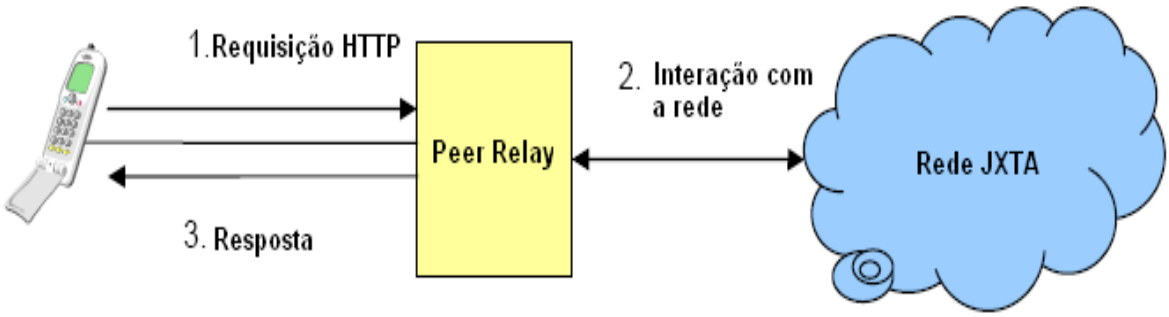


Figura C.4: interação entre *peer relay* e *peer* JXME.

Conforme está ilustrado na Figura C.4, os *peers* JXME se comunicam com os *relays* JXTA através de requisições HTTP. Quando um peer JXME faz uma requisição de busca por algum elemento da rede JXTA (como por exemplo: um *pipe*), a requisição é enviada à algum *relay*. Esse *relay* propaga a requisição na rede JXTA, e as respostas são por ele coletadas e armazenadas temporariamente, fazendo otimização das mesmas. Finalmente, o *peer* JXME contata o *relay* para buscar as mensagens a ele endereçadas. Um *peer* que atua como *relay* JXTA oferece uma série de serviços aos *peers* JXME, entre eles destacamos:

- Filtrar e otimizar anúncios propagados pela rede a fim de otimizar o processamento e consumo de memória dos *peers* JXME;
- Rotear as mensagens vindas de, e para *peers* JXME;
- Converter as mensagens XML para o formato binário e vice versa, a fim de proporcionar interoperabilidade entre os *peers* JXME e os *peers* convencionais;
- Atuar como um *proxy* para os *peers* JXME, realizando todo o processamento intensivo de descoberta de *peers* e grupos, criação de grupos e *pipes*, e comunicação.

Um *peer* JXME não necessita estar sempre atrelado a um único *peer relay*, podendo ele dinamicamente, mudar seu *relayi*, ou conversar com vários *relays*. Portanto, este arranjo não compromete a visão da arquitetura P2P. A API JXME fornece funcionalidades necessárias para integração dos *peers* JXME numa rede JXTA. Ela foi projetada de forma a mascarar a complexidade das trocas de mensagens e requisições de

serviços na rede JXTA. A API JXME possui três classes, conforme está ilustrado na Figura C.5.

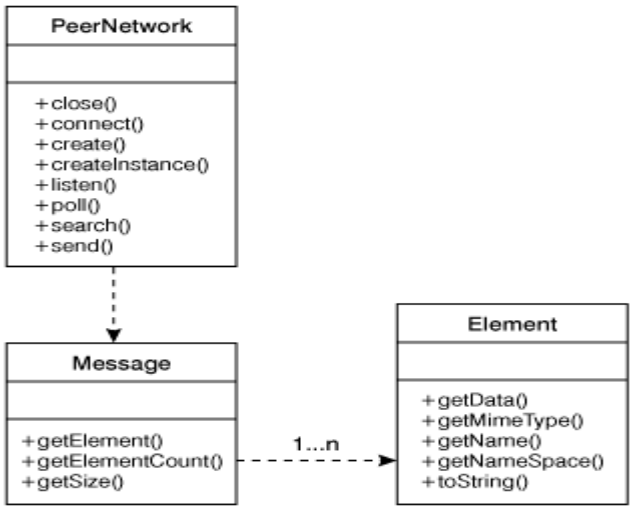


Figura C.5: diagrama de classes da API JXME.

Conforme a Figura C.5, as classes que compõem a versão API JXME que implementa *proxy* são:

- *Element* - representa um elemento à ser anexado a uma mensagem JXTA. Um elemento contém um nome, um *namespace*, um tipo MIME e um *array* binário de dados;
- *Message* - representa uma mensagem JXTA composta por um ou mais elementos;
- *P*eerNetwork - é a classe utilizada para especificar o ambiente JXME e executar tarefas na rede JXTA. Ela manipula a comunicação com um *peer relay*.

A Figura C.9 a seguir, representa a estrutura da mensagem JXME definida pela API para estabelecer comunicação a um *peer relay*.

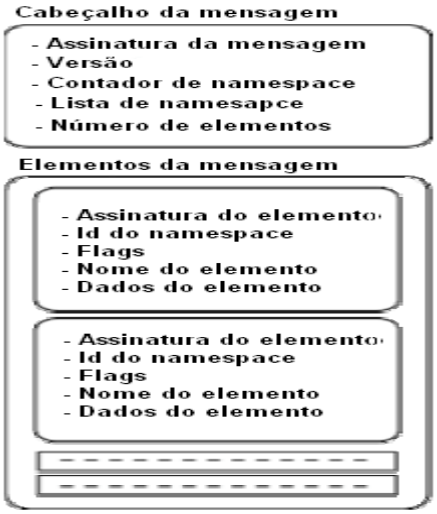


Figura C.6: estrutura da mensagem JXME.

Conforme está ilustrado na Figura C.6, a estrutura da mensagem de comunicação com um peer relay é dividida em duas partes: o cabeçalho (informa parâmetros para comunicação com o *relay*) e elementos da mensagem (que definem os dados à serem enviados na rede JXTA).