UNIVERSIDADE FEDERAL DE SANTA MARIA
CENTRO DE TECNOLOGIA
CURSO DE CIÊNCIA DA COMPUTAÇÃO

DESENVOLVIMENTO DE UMA APLICAÇÃO P2P UTILIZANDO A BASE DE DADOS PERVASIVA GDATA

TRABALHO DE GRADUAÇÃO

Alexander Fiabane do Rego

Santa Maria, RS, Brasil.
2008

DESENVOLVIMENTO DE UMA APLICAÇÃO P2P UTILIZANDO A BASE DE DADOS PERVASIVA GDATA

por

Alexander Fiabane do Rego

Trabalho de Graduação apresentado ao Curso de Ciência da Computação da Universidade Federal de Santa Maria (UFSM, RS), como requisito parcial para a obtenção do grau de
Bacharel em Ciência da Computação

Orientador: Profª Drª Iara Augustin

Trabalho de Graduação N. 265
Santa Maria, RS, Brasil.

2008

Universidade Federal de Santa Maria
Centro de Tecnologia
Curso de Ciência da Computação

A Comissão Examinadora, abaixo assinada,
aprova o Trabalho de Graduação

DESENVOLVIMENTO DE UMA APLICAÇÃO P2P UTILIZANDO A BASE DE DADOS PERVASIVA GDATA

elaborado por
Alexander Fiabane do Rego

como requisito parcial para obtenção do grau de
Bacharel em Ciência da Computação

COMISSÃO EXAMINADORA:

_____
Profª Iara Augustin
(Presidente/Orientador)

_____
Profª Márcia Pasin (UFSM)

_____
Profª Oni Reasilvia de Almeida Oliveira Sichonany (UFSM)

Santa Maria, 17 de Dezembro de 2008.

RESUMO

Trabalho de Graduação
Curso de Ciência da Computação
Universidade Federal de Santa Maria

DESENVOLVIMENTO DE UMA APLICAÇÃO UTILIZANDO A BASE DE DADOS PERVASIVA GDATA

Autor: Alexander Fiabane do Rego

Orientador: Profª Drª Iara Augustin

Local e data da defesa: Santa Maria, 17 de Dezembro de 2008.

O aumento no uso de dispositivos tecnológicos e a necessidade de um melhor fluxo da informação entre esses dispositivos requerem uma maneira diferente de acesso a essas informações. É então que a Computação Pervasiva, considerada a terceira onda da computação, traz a proposta do acesso a recursos de qualquer dispositivo tecnológico (notebooks, PDA's, celulares,...), a qualquer tempo e em qualquer lugar. Nesse mesmo sentido, também caminha a Computação nas Nuvens (*Cloud Computing*), visando à promoção do software como serviço através da Internet. Sendo assim, este trabalho visa à criação de uma aplicação que permita o acesso pervasivo a uma base de dados também pervasiva através de uma rede P2P. Esse propósito baseia-se na visão da Computação Pervasiva aliada à Computação nas Nuvens, considerando: (a) a construção de uma rede P2P, segundo a tecnologia *Peer-to-Peer,* implementada através da plataforma JXTA e (b) o acesso aos serviços da base de dados pervasiva - GDATA, criando assim um ambiente pervasivo, no qual é possível acessar os serviços oferecidos pela base de dados pervasiva de qualquer participante da rede de maneira simples.

Palavras-chave: Acesso pervasivo; P2P; JXTA; GDATA, Computação nas Nuvens; Base de Dados Pervasiva.

ABSTRACT

Trabalho de Graduação
Curso de Ciência da Computação
Universidade Federal de Santa Maria

DEVELOPMENT OF AN APPLICATION USING THE PERVASIVE DATABASE GDATA

Author: Alexander Fiabane do Rego
Advisor: Profa Iara Augustin

The increase in the use of technological devices and the need for a better information flow between these devices requires a different access way those information. So then, the Pervasive Computing, considered the third wave of computing, brings the proposal of access to resources of any technological device (notebooks, PDAs, cell phones ,...), anytime, anywhere. In the same way it is the Cloud Computing, which aims to promote software as a service through the Internet. Thus, this work relates the creation of a pervasive access using both a pervasive database and a P2P network. This purpose is based on the combined vision of Pervasive Computing and Cloud Computing, considering (a) the construction of a P2P network, according to technology Peer-to-Peer, it implemented by the JXTA platform and (b) the access to services from a pervasive database - GDATA, creating a pervasive environment, where any participant of the network can access the services offered by the pervasive database in a simple way.

Keywords: Pervasive Access; P2P; JXTA; GDATA, Cloud Computing; Pervasive Database.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS

API - Interface de Programação de Aplicativos (*Application Programming Interface*)

HTTP - Protocolo de Transferência e Hipertexto (*HyperText Transfer Protocol*)

IP - Protocolo de Internet (*Internet Protocol*)

ID - Identificador (*Identifier*)

JXTA - Juxtapose

P2P - Par-a-Par (*Peer-to-Peer*)

PDA - Assistente Pessoal Digital (Personal Digital Assistants)

RSS - Protocolo Realmente Simples (Really Simple Syndication)

TCP - Protocolo de Controle de Transmissão (*Transmission Control Protocol*)

XML - Linguagem de Marcação Extensível (*Extensible Markup Language*)

ADSL – Linha de assinatura digital assimétrica (*Asymmetric Digital Subscriber Line*)

IBM - International Business Machines

GLOSSÁRIO DE TERMOS

Peer - Dispositivo que participa de uma rede P2P, habilitado a comunicar-se com a rede [RIZZETTI, 2006].

Rendezvous - *Peer* especial, também atua como mecanismo de cache de mensagens para proporcionar funcionalidade e eficiência à rede P2P [RIZZETTI, 2006].

Atom - Protocolo ao nível da aplicação para publicar e editar fontes web [http://pt.wikipedia.org/wiki/Atom].

SUMÁRIO

1 INTRODUÇÃO

1.1 Motivação

Computação pervasiva é um paradigma computacional que nasceu a partir da idéia de Computação Ubíqua proposta por Mark Weiser [WEISER, 1991]. Esse novo paradigma do século XXI [SAHA ; MUKHERJEE, 2003] [SATYANARAYANAN, 2001] é considerado a terceira onda da computação [JANSEN et al, 2005], tendo como uma das suas propostas o acesso pervasivo, de forma que qualquer serviço/informação possa ser acessado/compartilhado entre os diversos dispositivos tecnológicos, sendo que o usuário não necessite ter conhecimento de como esse serviço/informação está sendo acessado/compartilhado.

O acesso pervasivo, que pode ser definido como o acesso de qualquer lugar, a qualquer tempo e de qualquer dispositivo, contribui para uma característica essencial da ubiqüidade: a invisibilidade da computação [AUGUSTIN et al, 2006]. Com isso, as informações não ficam mais restritas a um determinado tipo de tecnologia e nem a comunicação restrita a um grupo específico de dispositivos. Os diversos grupos tecnológicos (Celulares, PDA's, Notebooks,...) comunicar-se-ão entre si de forma espontânea, sem que para isso o usuário precise configurar os dispositivos, como acontece no cenário atual, para interagirem com outras tecnologias.

Os serviços, hoje oferecidos, foram projetados para atender a grupos de dispositivos tecnológicos da mesma família. Mas, o crescimento no uso de outros dispositivos como PDA's, smartphones e telefones celulares, tende a facilitar a troca de informações/dados, e surge a necessidade de interligar esses dispositivos heterogêneos, a fim de permitir um fluxo natural da informação. Uma nova forma de projetar esses serviços faz-se necessária.

Todos os serviços de acesso utilizam bases de dados para gerenciamento e armazenamento de suas informações, informações de clientes e/ou relacionadas a estes. Com isso, as bases de dados utilizadas terão que permitir o acesso de diversos dispositivos, a fim de que seja possível a comunicação entre essas diferentes tecnologias de forma automática.

Sendo assim, a maneira de projetar novos serviços utiliza o que a tecnologia disponibiliza para o auxílio da implantação desses serviços. O conceito de computação pervasiva adapta a realidade tecnológica a essas necessidades.

Nessa direção, a proposta da Google para um provável futuro da computação é a Computação nas Nuvens (*Cloud Computing*) [http://code.google.com/apis/gdata]. O termo "nuvem" [LOHRS, 2007] é uma metáfora para a Internet. A Computação nas Nuvens visualiza a Internet (nuvem) como uma provedora de serviços. Dessa forma, os *software*s são vistos como um serviço. Assim, pode-se ter acesso a *software*s de seu uso geral de qualquer dispositivo computacional, sem a preocupação de instalá-los. A pessoa os acessa através da Internet.

Tendo em vista os conceitos de Computação Pervasiva e de Computação nas Nuvens (na forma proposta atualmente pela Google para a Computação Ubíqua) este trabalho prototipa uma aplicação que visa criar um serviço de acesso pervasivo a uma base de dados, também, pervasiva, usando uma rede P2P. A aplicação P2P gerencia a comunicação entre dispositivos tecnológicos heterogêneos e a oferta de serviços a esses.

1.2 Objetivos

As tecnologias para acesso pervasivo procuram oferecer uma solução simples e eficiente, de tal forma que os diferentes dispositivos, através de uma aplicação ou serviço, obtenham acesso ao serviço sem muitas configurações.

Nesse trabalho, relata-se o desenvolvimento de um serviço de acesso a uma base de dados pervasiva na qual dispositivos como notebook, PDA's e telefones celulares, têm acesso à base de dados GDATA [http://code.google.com/apis/gdata/overview.html] e compartilham as suas informações/dados através de uma rede P2P [http://www.gta.ufrj.br/grad/04_1/p2p].

Em um primeiro momento, é construída a rede P2P entre esses dispositivos (notebook, PDA's e celulares). A comunicação é baseada na troca de mensagens, a fim de se verificar o estado de cada dispositivo na rede.

Após essa primeira etapa, é realizado o acesso à base de dados pervasiva. Esse acesso é feito através da rede P2P criada anteriormente.

Uma aplicação foi criada para a interação, de maneira mais amigável, com a rede P2P. Essa aplicação, então, acessa a base de dados pervasiva GDATA através da rede P2P, oferecendo os serviços que essa base de dados dispõe e que os dispositivos envolvidos na rede suportam.

Com isso, objetiva-se entender melhor o potencial dessa tecnologia e demonstrar como usar a computação pervasiva como paradigma no projeto de aplicações que visem à promoção de serviços que independam do tipo de tecnologia-cliente e que estejam de acordo com o avanço tecnológico.

1.3 Estrutura do Texto

O restante do texto está estruturado da seguinte forma; o capítulo 2 aborda as tecnologias de base de dados pervasiva, onde se encontra uma pequena pesquisa a respeito das empresas que oferecem esse tipo de serviço. Nesse mesmo capítulo é abordado o conceito de computação nas Nuvens e a base de dados escolhida - GDATA.

O capítulo 3 trata do serviço de acesso à base de dados pervasiva GDATA através de uma rede P2P, trazendo dessa forma, os conceitos e abordagens desse tipo de rede. Nesse capítulo, é apresentada a plataforma em que é implementada a rede P2P e a solução prevista nesse trabalho.

No capítulo sobre a arquitetura da rede, capítulo 4, detalha-se a solução proposta no capítulo 3. Assim, é abordado (i) como acessar os serviços oferecidos pela base de dados pervasiva e (ii) como funciona esse acesso, tendo como (iii) caso de uso a implementação de uma aplicação exemplo.

O capítulo 5 descreve os testes realizados com a aplicação exemplo e os resultados obtidos com esses testes.

Por fim, o capítulo 6 apresenta a conclusão desse trabalho com relação às tecnologias empregadas para a implementação da aplicação exemplo e sobre os resultados obtidos com os testes realizados, descritos no capítulo 5.

2 TECNOLOGIAS DE BASE DE DADOS PERVASIVA

Qualquer aplicação que interaja com o usuário armazena informações, seja em bancos de dados ou algum outro meio de armazenamento.

Com o crescimento da utilização e o avanço de tecnologias como telefones celulares, assistentes pessoais (PDA) e *notebook*, essas bases de dados também precisam se modificar, a fim de permitir o acesso via diferentes dispositivos. Uma base pervasiva é a solução para esse problema.

A base de dados é dita pervasiva quando o usuário não a gerencia, não tem conhecimento de sua localização e do tipo de formato de dados utilizado (MySQL, Oracle,...). Ela provê somente uma interface com os métodos a serem usados para acesso de escrita e consulta. Dessa forma, os dispositivos a acessam através de mecanismos padronizados como protocolos, que fazem à abstração da tecnologia empregada na base de dados.

Uma empresa que fornece esse tipo de base de dados é a Google através da plataforma GDATA [http://code.google.com/apis/gdata/overview.html]. Porém, a Google não é a única companhia a investir em armazenamento de dados *on-line* e Computação nas Nuvens (*Cloud Computing*). Entre outras empresas tem-se a *Amazon Web Services* [http://aws.amazon.com/] com *Simple Storage Service*, AOL [http://www.aol.com/] com *Xdrive* e Microsoft [http://www.microsoft.com/en/us/default.aspx] oferecendo o *Windows Live SkyDrive*.

Como mostra a tabela 1, a Google tem um impacto maior no desenvolvimento do paradigma de Computação nas Nuvens, pois integra os dispositivos móveis com a Computação nas Nuvens.

Tendo em vista essa integração e o domínio de mercado exercido pela Google, escolheu-se desenvolver a aplicação usando a plataforma para acesso pervasivo GDATA.

Na seqüência, aprofundam-se um pouco mais questões relativas à Computação nas Nuvens e a plataforma Google GDATA.

Tabela 1 – Comparação entre bases de dados pervasivas

Sistema	**Características**	**Suporta dispositivos móveis**	**Empresa**
Simple Storage Service (3S)	• Disponibiliza bibliotecas para o desenvolvimento de aplicações que acessem a base de dados • Disco virtual	Não	Amazon *Web Services*
Xdrive	• Repositório de arquivos com acesso via Internet. • Disponibiliza aplicação desktop para interação com o disco virtual.	Não	AOL
GDATA	• Permite o armazenamento de dados. • Acesso via Internet. • Disponibiliza biblioteca para desenvolvimento de aplicações para interação com a base de dados. • Oferece serviços (Google Calendar, Google Docs, entre outros).	Sim	Google
Windows Live SkyDrive	• Diretório virtual que pode ser acessado de qualquer computador com acesso a Internet.	Não	Microsoft

2.1 Computação nas Nuvens (*Cloud Computing*)

Nuvem, como dito anteriormente, é uma metáfora para Internet. É o desenvolvimento da computação baseada na Internet, onde os *software*s são oferecidos como serviços, permitindo usuários acessarem a tecnologia habilitada como serviço através da Internet, sem conhecimento prévio sobre aquela tecnologia ou toda a infra-estrutura que oferece suporte a ela.

Computação nas Nuvens é um conceito geral que incorpora *software* como um serviço, Web 2.0 e outras recentes tendências tecnológicas bem conhecidas, cujo propósito é satisfazer as necessidades computacionais de seus usuários através da Internet. Nessa linha, tem-se a *Google Apps* que provê aplicações comuns *on-line* que são acessadas por navegadores enquanto que o *software* em si e os dados são armazenados nos servidores [http://code.google.com/apis/gdata].

O Windows Azure [http://www.microsoft.com/azure/], versão do sistema operacional para Cloud Computing, permite que desenvolvedores criem e hospedem seus serviços usando a infra-estrutura da Microsoft.

Esse sistema (Windows Azure) compete com o *Elastic Compute Cloud* (EC2), da Amazon, como local de hospedagem, no qual os desenvolvedores podem construir e armazenar seus aplicativos. Neste mês, novembro/2008, a Microsoft está lançando o Community Technology Preview (CTP) do Windows Azure, nos Estados Unidos, que hospedará o serviço em centros de dado globais. Os desenvolvedores podem usar as ferramentas .Net para criar aplicativos no Windows Azure, as mesmas utilizadas para a criação do ambiente.

Dessa forma, com a Computação nas Nuvens, o usuário não terá que ter necessariamente o *software*, ele apenas o acessará ou irá alugá-lo, podendo assim economizar capital e consumir recursos como um serviço, pagando pelo que ele usa. Esse modelo computacional é similar aos sistemas utilitários tradicionais como a água (serviço público). Com isso, o poder computacional crescerá drasticamente, já que toda tecnologia física (*hardware*) ficará a cargo das empresas que irão prover os serviços de computação. Não só o *hardware* irá avançar, mas a velocidade da banda de Internet, a fim de que a resposta seja tão rápida como a do modelo *desktop*. A tendência é o desaparecimento dos computadores como os conhecemos hoje (CPU, monitor,...), sendo substituídos por interfaces de comunicação.

O conceito de Computação nas Nuvens não é novo, ele data do ano de 1960 quando John McCarthy[http://en.wikipedia.org/wiki/Cloud_computing] disse que a computação, um dia, seria organizada como uma utilidade pública.

Historicamente, a Amazon desempenhou um papel fundamental para a Computação nas Nuvens ao modernizar o seu *datacenter* provendo acesso aos seus sistemas através do Amazon Web Services, em 2002, com utilitários computacionais básicos.

Em 2007, aumentaram as atividades nesse sentido com a Google, IBM e universidades apoiando projetos de pesquisa na área de Computação nas Nuvens – a computação vista como um serviço público.

Destacam-se, a seguir, alguns pontos-chaves a respeito da Computação nas Nuvens [http://en.wikipedia.org/wiki/Cloud_computing].

Gasto de Capital: minimiza os custos com a aquisição de tecnologia (*hardware*) e a barreira imposta pelos *softwares*, já que essa responsabilidade é de quem provê o serviço (*software* como um serviço).

Independência de localização e aparelho: possibilita que os usuários acessem sistemas de onde estiverem e com o aparelho que possuírem (*notebook*, PDA, telefone celular, smartphone).

Escalabilidade: atende ao crescimento da demanda dos usuários sem picos de processamento.

Segurança: tendo em vista que os dados ficam centralizados, o investimento em recursos de segurança se torna mais viável, mas aumentam as considerações sobre a perda do controle do usuário-final sobre alguns dados mais sensíveis.

Sustentabilidade: melhora a utilização dos recursos, pelo fato desses serem adquiridos por quem provê os serviços, e por serem utilizados em pontos centrais melhorando a eficiência do sistema.

Assim, a Computação nas Nuvens vem a somar na Computação Pervasiva (ou Ubíqua), permitindo um melhor uso dos dispositivos tecnológicos, abstraindo toda parte de configuração de *hardware*, algumas vezes necessária, para a utilização do *software*. E, também, uma maior interação entre os diversos dispositivos.

2.2 Google Data (GDATA)

GDATA é uma abreviação para Google *Data APIs.* Essa API permite o acesso à base de dados do Google e aos serviços oferecidos por ela. O GDATA provê um protocolo padrão e simples de leitura e escrita de dados na Internet [http://code.google.com/apis/gdata/overview.html].

O GDATA utiliza dois formatos de protocolos padrões baseados em XML: Atom e RSS. Também possui um sistema de publicação baseados em *feed*[1], o qual é composto pelo protocolo Atom e mais algumas extensões (dentro do padrão Atom) para trabalhar com as consultas.

[1] Conteúdo.

Esses dois padrões de protocolos, baseados em XML, são utilizados para distribuição de *feeds* e trocas de dados na web. O RSS já existe há algum tempo e em diversas versões, já o Atom é mais novo, porém apresenta mais recursos.

O Atom oferece o *Atom Publishing Protocol* (APP). Esse protocolo é baseado em HTTP para o intercâmbio de dados/recursos da web. O padrão utiliza comandos simples como GET, POST, PUT e DELETE, o que facilita o desenvolvimento de novos serviços.

O Google *Data APIs* amplia esses protocolos, utilizando mecanismos de extensão integrados aos protocolos. O modelo de publicação do GDATA adequa-se ao *Atom Publishing Protocol.*

Para se obter informações de um serviço que suporte o GDATA, por exemplo, envia-se uma solicitação HTTP GET e o serviço retorna os resultados como um *feed Atom* ou RSS. Ressalta-se, novamente, que para requisitar alguma informação pode-se utilizar um dos dois protocolos, mas para enviar informações utiliza-se somente o padrão Atom.

Muitos serviços podem fornecer *feeds* ao GDATA, desde serviços como blogs, emails até eventos de agendas. Isso porque os modelos RSS e Atom permitem extensões, fazendo com que cada provedor de dados defina, conforme as suas necessidades, suas próprias extensões e semânticas.

Devido ao fato de o GDATA ter sido criado sobre tecnologias básicas, HTTP e modelos de distribuição padrões, pode-se enviar solicitações e processar dados de diversas maneiras, como clientes baseados em JavaScript/AJAX, aplicativos independentes ou outras abordagens. O protocolo do GDATA independe da linguagem usada na programação, desde que essa permita o envio de solicitações HTTP e análise de respostas baseadas em XML.

Levando em consideração a flexibilidade e simplicidade que o GDATA agrega, e a integração com dispositivos móveis, optou-se por usar esta base de dados para implementação do serviço de acesso a dados via rede P2P.

O GDATA disponibiliza bibliotecas [http://code.google.com/intl/pt-BR/apis/gdata/clientlibs.html] para o desenvolvimento de aplicações em diversas linguagens. Nesse trabalho, devido à linguagem de programação utilizada ser a linguagem Java, adotou-se a biblioteca do GDATA referente a essa linguagem.

Todos os serviços do GDATA são, efetivamente, acessados via HTTP. No desenvolvimento de aplicações utilizando as bibliotecas disponibilizadas pelo GDATA, basicamente, cada serviço possui classes específicas que auxiliam na interação da aplicação com ele. De modo geral, cada serviço possui uma classe *Service*, a qual interage efetivamente com ele, e uma *Entry*, que representa a informação do serviço. Dessa forma, com as

bibliotecas devidamente associadas à aplicação, basta instanciar a classe *Service* e efetuar o *login*, para interagir com o respectivo serviço.

3 PROPOSTA DO SERVIÇO DE ACESSO A DADOS VIA REDE P2P

Antes de detalhar a proposta desse trabalho, torna-se necessário o conhecimento de conceitos como *rede P2P* e de tecnologias que serão empregadas na implementação da solução, como o Juxtapose (*JXTA).*

Os conceitos descritos a seguir serão importantes, pois, aliados ao conceito de Computação Pervasiva e Computação nas Nuvens, visto anteriormente, darão uma visão mais clara sobre o processo de desenvolvimento deste trabalho.

3.1 Tecnologia P2P

A arquitetura P2P (*peer-to-peer*) tornou-se mais conhecida a partir do uso de um *software* Napster [KOVER, 2000]. Diferente da arquitetura cliente/servidor, onde se tem a centralização de serviços, a rede P2P tem como base a descentralização dos serviços. Desse modo, cada nó na rede assume papel equivalente em capacidade e responsabilidade.

O termo P2P ainda é erroneamente usado para descrever a ligação entre dois usuários para troca de informações e dados através de uma aplicação-cliente P2P comum. A rede P2P não possui apenas esse gênero, podendo-se classificar a arquitetura P2P em três categorias, segundo sua rede e aplicações [BEAL, 2005]:

Computação Colaborativa: também conhecida como computação distribuída, visa o compartilhamento de recursos dos nós (processador, memória) entre os nós participantes da rede. O uso desse gênero de P2P é freqüente em ambientes de pesquisas científicas, onde o processamento de informações é intenso;

Mensagem Instantânea: essa categoria refere-se ao uso de aplicações (MSN e afins) que permitem a troca de mensagens em tempo real;

Comunidades Afins: são grupos de redes P2P baseadas no compartilhamento de arquivos. É nessa categoria que se encontram *software*s como Napster, Limewire, entre outros.

As comunidades afins são construídas através da colaboração dos usuários, compartilhando os seus arquivos, e na pesquisa de informações e dados nos dispositivos dos usuários participantes da comunidade.

Para se construir uma rede P2P existe duas abordagens. A primeira abordagem é baseada em um servidor de indexação (*Super-peer* [TRAVERSAT, 2006]), o qual guarda informações sobre avisos[2], grupos e serviços. A comunicação em si não é afetada por esse servidor, ela permanece *peer-to-peer*. Sendo assim, quando o usuário inicia a aplicação-cliente P2P, loga-se nesse servidor de indexação.

Ao iniciar uma busca, o usuário, na verdade, envia essa busca para o servidor de indexação que procura nos usuários on-line o arquivo solicitado; ao encontrá-lo, retorna, ao usuário que efetuou a busca, o ID do usuário que possui aquele arquivo, a partir desse ponto a comunicação é efetivamente ponto-a-ponto.

Alguns pesquisadores consideram essa abordagem como sendo uma combinação entre cliente/servidor e P2P, devido à existência desse servidor de indexação (*Super-peer*) que cria uma estrutura hierarquizada.

Outra maneira de se construir uma rede baseada na arquitetura P2P é através da descoberta de recursos/serviços realizada pelo par que deseja a comunicação; essa abordagem exclui o servidor de indexação (ponto de centralização). Assim, nesse cenário, a aplicação-cliente P2P procuraria por outro usuário, que utilize o mesmo *software* e informaria que está on-line, construindo assim uma larga rede de usuários.

Em suma, a arquitetura P2P tem o propósito de trazer interação entre os dispositivos conectados a uma rede baseada nessa arquitetura, possibilitando, em tempo real, o compartilhamento direto de recursos e serviços.

3.2 Tecnologia P2P adotada: Plataforma JXTA

O JXTA [JXTA. ORG 2008] é um projeto desenvolvido pela Sun Microsystems, que auxilia a criação de redes P2P. O objetivo do JXTA é prover um conjunto simples, pequeno e flexível de mecanismos que apóie a construção de redes P2P, em qualquer plataforma, em qualquer lugar e a qualquer hora. O JXTA é a primeira generalização das funcionalidades da

[2] Nesse contexto, define-se como sendo mensagem de descoberta de recursos em uma rede P2P.

arquitetura P2P, seu foco é a criação de mecanismos básicos, deixando as políticas de rede para serem escolhidas pelos desenvolvedores.

Os mecanismos básicos que o JXTA oferece são baseados em protocolos. Isso permite que o desenvolvedor tenha flexibilidade para programar suas aplicações, não se detendo em protocolos de redes e transporte dos dados.

Os protocolos básicos oferecidos pelo JXTA são seis (6); eles foram criados para atuarem em conjunto e permitem o descobrimento, o monitoramento, a organização e a comunicação entre pares (*peers)*. Não é necessário utilizar todos os protocolos em todos os pares, mas para entrar em uma rede é requisitado o protocolo de roteamento (*endpoint routing*). Abaixo segue uma descrição dos protocolos básicos e suas funções.

- *Peer Discovery Protocol* (Protocolo de Descoberta): permite aos pares ofertarem seus recursos e buscarem recursos que estão disponíveis em outros *peers*.

- *Peer Resolver Protocol* (Protocolo de Resolução): possibilita que os pares enviem consultas para um ou mais pares e recebam a resposta da consulta em questão. Na mensagem, junto com a consulta, é enviada uma identidade única, a fim de associar a resposta com a pergunta.

- *Peer Information Protocol* (Protocolo de Informação): com esse protocolo um par pode descobrir informações do estado de outros pares, isto é, *uptime*[3] e capacidades, e também, obter informações de um par.

- *Pipe Binding Protocol* (Protocolo de Tunelamento): o par realiza um canal de comunicação virtual (*pipe*) entre outro par ou mais. Em suma, liga dois pares.

- *Endpoint Routing Protocol* (Protocolo de Roteamento): permite a descoberta de uma rota entre dois pares. Este protocolo é usado no caso de não haver uma ligação direta entre dois pares.

- *Rendezvous Protocol* (Protocolo de Encontro): possibilita que os pares se tornem membros de um serviço de propagação. Sendo membro de um serviço de propagação, um par pode receber mensagens e informações de todos os membros desse serviço.

[3] Tempo que determinado serviço ou dispositivo está ativo e funcional.

No JXTA, a comunicação entre os pares é feita por mensagens XML. Isso faz com que a plataforma possa ser implementada em uma variedade de linguagens. Além disso, pelo fato de ser baseada em protocolos, torna portável a sua implementação.

Toda essa flexibilidade que o JXTA oferece torna-o robusto, facilitando assim, o desenvolvimento de sistemas P2P.

Sendo assim, o JXTA torna-se uma ferramenta muito útil no desenvolvimento de sistemas que estejam imersos em um ambiente pervasivo, pois possibilita a comunicação entre dispositivos heterogêneos de maneira simples e eficiente, abstraindo a comunicação entre esses dispositivos.

3.3 Solução Proposta

A solução proposta nesse trabalho visa não só a aplicação dos conceitos de Computação Pervasiva, mas também o conceito e tecnologias que envolvam a comunicação *peer*-to-*peer* entre dispositivos móveis heterogêneos.

Sendo assim, inicialmente, foi desenvolvida uma aplicação responsável pela comunicação entre os pares envolvidos na rede. Essa aplicação está implementada em *Java* [Sun Developer Network].

Conforme revisão de conceitos feita em tópicos anteriores, tem-se como solução a construção de uma rede P2P, na qual dispositivos heterogêneos podem conectar-se a essa rede, possibilitando o acesso pervasivo a uma base de dados também pervasiva, como pode ser visto na figura 1, pois o acesso a base de dados pervasiva é feito via rede P2P.

Figura 1 - Proposta de Solução

Assim, inicialmente é criada uma rede P2P entre dispositivos heterogêneos (telefone celular, PDA, notebook) e a base de dados.

Segundo revisão feita sobre o JXTA, são usados os protocolos básicos para a criação dessa rede. As redes são compostas de pares, onde cada *peer* representará um dispositivo. Esses pares procuram uma rede e são indexados pelo par *rendezvous*, responsável pela estruturação da rede.

A comunicação entre os pares ocorre por meio de mensagens que contém o *peer*ID (Identificador único que representa o par na lista de pares conectados a rede, a qual o par *rendezvous* mantêm), informações sobre o par (descrição do dispositivo) e o serviço requerido com informações referentes ao serviço (Rendezvous Protocol).

Cada par provê serviços à rede, da mesma forma que requisita serviços aos outros pares da rede. Essa oferta de serviços, na realidade, é feita através de um acesso a base de dados, tendo em vista que essa base de dados oferece serviços e que alguns desses somente podem ser acessados por determinados tipos de dispositivos. Assim, com a oferta de serviços na rede, todos os pares terão acesso aos mesmos serviços, pois, se o dispositivo em questão não possui acesso direto a um determinado serviço, ele procura algum par que oferte aquele

serviço e, então, efetua a comunicação para obtenção do serviço requisitado (*Peer Discovery Protocol*).

O tratamento da comunicação, entre os pares, é feito através da plataforma JXTA, através do envio de mensagens, no formato XML, com as informações sobre o par que efetuou a requisição contida na mensagem. Dessa forma, a aplicação lê a mensagem, identifica o tipo de dispositivo que efetuou a requisição, verifica o serviço requisitado e, se esse estiver acessível, inicia o acesso à base de dados, a fim de realizar a requisição. Caso essa transação seja efetuada com sucesso, o par que requisitou o serviço recebe uma mensagem de aprovação; caso não seja possível o acesso, seja por indisponibilidade ou por não ter acesso a tal serviço, o par que recebeu a mensagem irá procurar um outro par que possa ter acesso ao serviço em questão.

Sendo assim, a mensagem de requisição de um serviço passa de par em par até que um efetue o acesso; se o par que requisitou o serviço receber a sua própria mensagem saberá que aquele serviço encontra-se indisponível naquela rede e enviará uma mensagem ao usuário de indisponibilidade do serviço.

Os diferentes tipos de serviços somente são possíveis de serem acessados devido à construção da rede P2P. Assim, o acesso aos serviços oferecidos pela base de dados é feito pelo par que tem acesso direto ao serviço. Para se tornar o acesso à base de dados pervasivo, utiliza-se o conceito de rede P2P e, então, tem-se o acesso indireto a qualquer serviço, por qualquer par. E para que não seja necessária a configuração manual sobre a informação que se deseja, uma aplicação faz o tratamento dos dados, tornando assim pervasivo o acesso ao serviço.

Dessa forma, constrói-se um ambiente pervasivo onde qualquer dispositivo acessa qualquer informação, sem que para isso seja necessária a noção detalhada de como é feito. Isso se deve a abstração que o JXTA nos possibilita, a base de dados pervasiva GDATA e o conceito de rede P2P aplicado à realidade tecnológica disponível.

4 ARQUITETURA DA SOLUÇÃO PROPOSTA

Neste capítulo é apresentado o desenvolvimento da solução proposta no capítulo anterior, abordando os principais aspectos do desenvolvimento e características da implementação realizada.

4.1 Acesso à base de dados pervasiva via rede P2P

Antes de se implementar o acesso pervasivo é necessário definir o tipo de rede P2P que será utilizada, qual o objetivo da rede. Nesse ponto, têm-se duas abordagens: *Superpeer* (híbrida) e P2P pura (*ad-hoc*).

Na construção da rede, utilizando o JXTA, é necessário que pelo menos um dos pares da rede seja um par *rendezvous*, pois este funciona como roteador por onde certas informações devem passar, tais como as redes que podem ser acessadas, os pares que estão conectados e suas redes e os recursos disponíveis a cada rede e que par os disponibilizou.

De certa forma, a rede P2P sofre certa centralização utilizando o JXTA, mesmo na abordagem *ad-hoc*. Mas a diferença entre as duas abordagens é que na rede P2P híbrida tem-se um Superpar que está sempre conectado e que busca informações (em arquivos, estruturas ou banco de dados) pré-definidas, como as redes e seus respectivos pares. Na rede P2P pura, a construção da rede ocorre espontaneamente. Se um par deseja conectar-se a uma rede P2P, ele deve procurar um par *rendezvous*. Caso não encontre, um novo par *rendezvous* deve ser criado, iniciando assim uma nova rede, onde outros pares poderão efetuar conexão.

Escolhido o tipo de rede, projeta-se o desenvolvimento da aplicação e a escolha do banco de dados pervasivo. Tal escolha depende do objetivo que se queira dar à rede.

Conforme a figura 2, toda a comunicação e o acesso a serviços/recursos ocorrem através da rede P2P.

O dispositivo executa a aplicação que irá procurar um par *rendezvous*, informando seu *status* (*on-line*) e os recursos/serviços que ele possui.

Os serviços que o par disponibilizar devem estar de acordo com os serviços oferecidos pela base de dados pervasiva.

Para solicitar determinado serviço, o par envia uma mensagem ao par *rendezvous* requisitando-o. Esse, por sua vez, verifica que par disponibilizou o serviço e, então, repassa o *peerID* para o par que iniciou a solicitação.

De posse do *peerID* do par que disponibilizou o serviço solicitado, a aplicação envia uma mensagem iniciando a requisição. Nesta primeira mensagem são enviados os dados para autenticação no serviço; em caso de sucesso a aplicação do par que iniciou o processo de requisição disponibiliza a tela de interação com o serviço. Ao concluir essa etapa, a aplicação enviará outra mensagem com esses dados para o par que disponibilizou o serviço e este efetuará a gravação desses dados na base de dados.

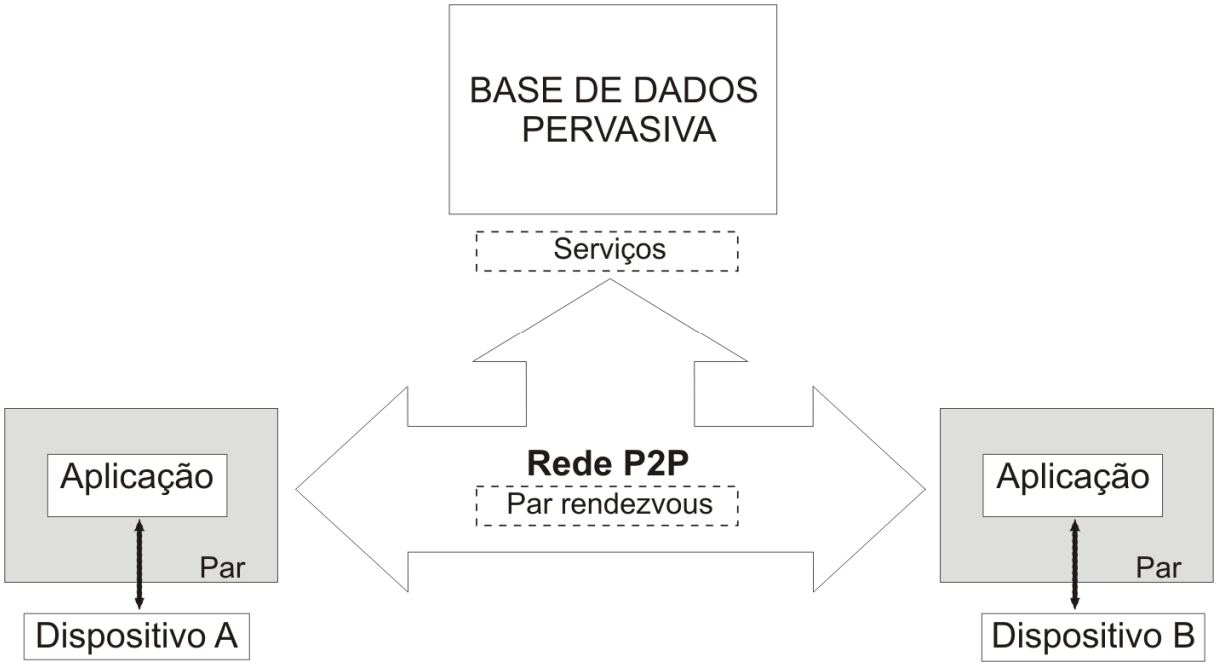

Figura 2 - Arquitetura do acesso pervasivo.

4.2 Funcionamento do acesso pervasivo

Para criar esse ambiente pervasivo, definiram-se situações principais: entrada na rede e comunicação na rede. Essas situações independem do tipo de rede escolhida, pois se tratam de apontamentos para a criação do ambiente pervasivo.

4.2.1 Entrada na Rede

Para a entrada na rede JXTA, inicialmente, é efetuada a inscrição do par na rede (através do *NetPeerGroup*) através do acesso ao par *rendezvous*, e envio de uma mensagem com informações sobre o par, *status* e serviços que compartilhará.

O par *rendezvous* verifica a que rede o par pertence e, senão pertence à rede nenhuma, cria uma. Por fim, insere o par na rede, a fim de que possa interagir com os demais usuários da rede.

4.2.2 Comunicação na Rede

A descoberta de recursos/serviços é feita pelo par *rendezvous*. Ele, através do envio de mensagens aos pares participantes da rede, pergunta, em espaços de tempos regulares, o *status* e os recursos/serviços que os pares estão ofertando. De posse das respostas, envia uma mensagem para cada par com informações a respeito de quais pares estão *on-line* e os serviços que aquela rede oferta.

A comunicação ocorre de forma efetiva, entre pares na rede JXTA, através dos *pipes*, que são mecanismos de comunicação unidirecionais.

Ao iniciar um serviço, ou mesmo uma troca de mensagens com outro par, é criado um *pipe* e este recebe um *pipeID,* como qualquer recurso e par da rede. O par que iniciou a ação fica de posse desse *pipeID* para identificação tanto no envio como no recebimento de mensagens.

Os *IDs* (*PeerID*s e *IDs* dos recursos envolvidos em comunicações) que devem ser de conhecimento dos pares encontram-se disponíveis no par *rendezvous*.

Assim, cria-se um ambiente pervasivo, onde dispositivos heterogêneos podem comunicar-se de maneira simples, propiciando um melhor fluxo de informações.

4.3 Implementação de um caso de uso

A fim de dar sentido prático para as definições propostas acima, foi implementada uma aplicação exemplo, na qual foi desenvolvida a solução do acesso pervasivo via rede P2P.

A aplicação tem por objetivo, mostrar o acesso pervasivo, via rede P2P, a uma base de dados também pervasiva, nesse caso o GDATA. Sendo assim, a aplicações disponibiliza apenas funcionalidades básicas como a troca de mensagens entre pares, a listagem dos eventos do calendário pré-definido pela aplicação, a inserção e exclusão de eventos desse calendário.

Na figura 3 tem-se um diagrama de classes, na qual se pode observar algumas das classes mais importantes na construção da aplicação.

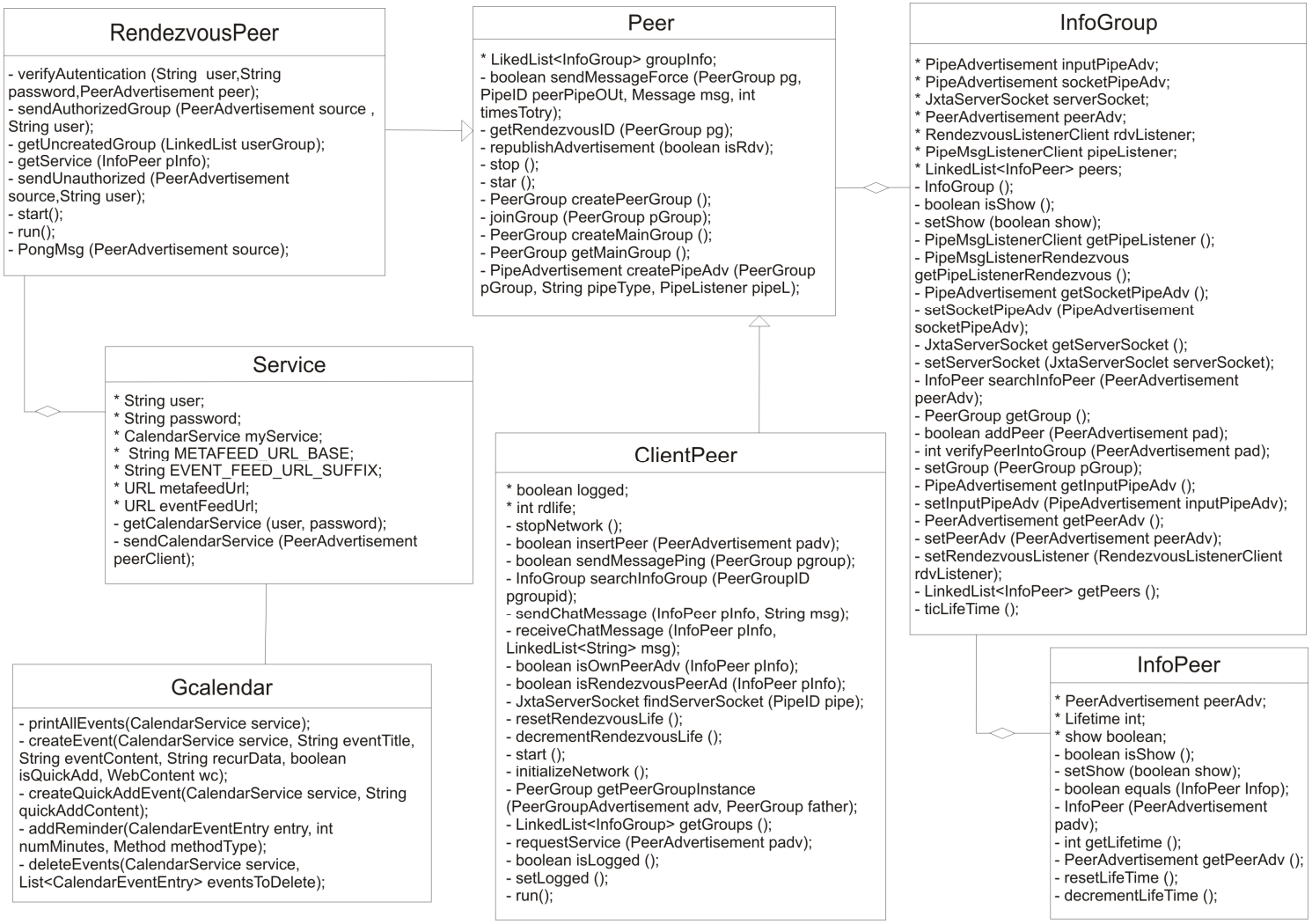

Figura 3 – Diagrama UML da aplicação exemplo.

Na aplicação, utiliza-se uma abordagem *ad-hoc*; sendo assim, ao iniciar a aplicação, conforme a figura 4, o `ClientPeer` procura por um `RendezvousPeer` e entra com a requisição e autenticação no grupo. Caso não encontre um `RendezvousPeer`, cria um e segue o processo descrito acima.

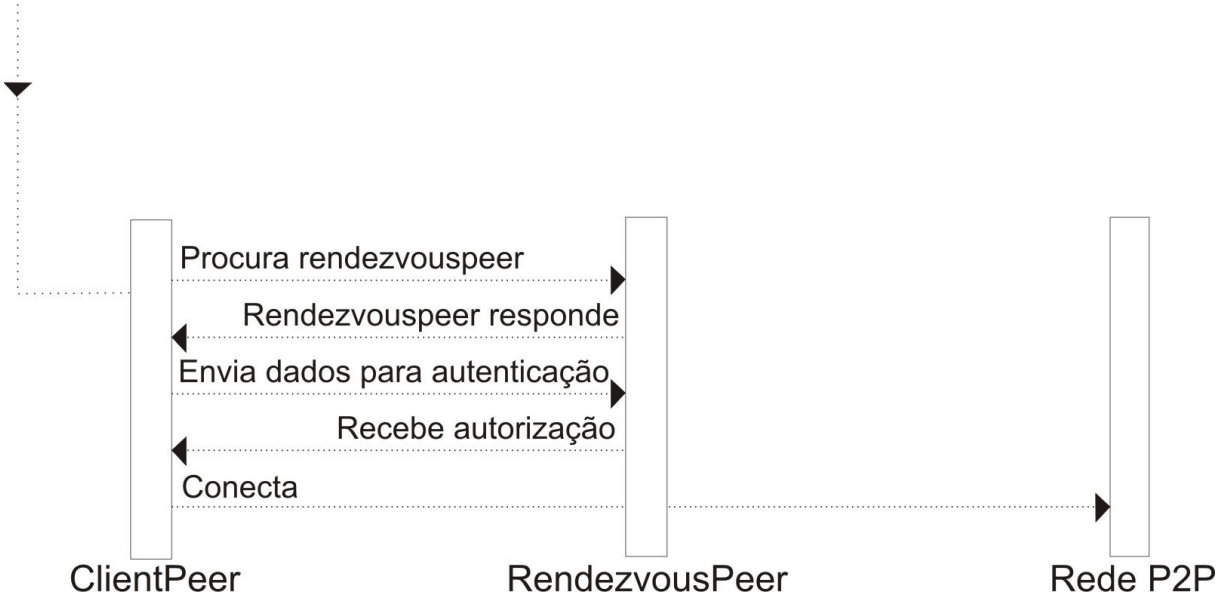

Figura 4 – Diagrama de seqüência da inscrição de um par em uma rede P2P.

A rede é sempre inicializada através da inscrição do `ClientPeer`, primeiro, no `NetPeerGroup` e depois no Grupo a que pertence.

O `RendezvousPeer` atua como uma espécie de roteador. Essa é a situação de centralização da rede JXTA, pois é necessário um par *rendezvous* para a descoberta de redes e seus respectivos pares, bem como dos recursos disponibilizados pelos pares em cada rede. É necessário pelo menos um par *rendezvous* por grupo [NASCIMENTO, 2007].

Todo controle de descoberta de redes, dos pares conectados a elas e dos recursos é feito pelo `RendezvousPeer`, através de *advertisements*, que são mensagens enviadas aos pares que buscam informações sobre *status* de conexão, *on-line* ou *off-line*, e de recursos oferecidos pelos pares.

Uma vez conectado à rede JXTA, o par troca mensagens com outros pares que pertençam às mesmas redes (grupos). Ao enviar uma mensagem para outro par, o `ClientPeer` envia um pedido ao `RendezvousPeer` que repassa o `PipeID` (Id do canal

de comunicação direto com o par desejado) e então é estabelecida a comunicação P2P propriamente dita.

O mesmo ocorre com o serviço, segundo a figura 5, o par (`ClientPeer`) entra com a requisição do serviço (`RendezvousPeer`), autentica no serviço (`Service`) e então interage com o serviço (`GCalendar`). No apêndice A, tem-se algumas das funções mais utilizadas para interação com o serviço Google Agenda.

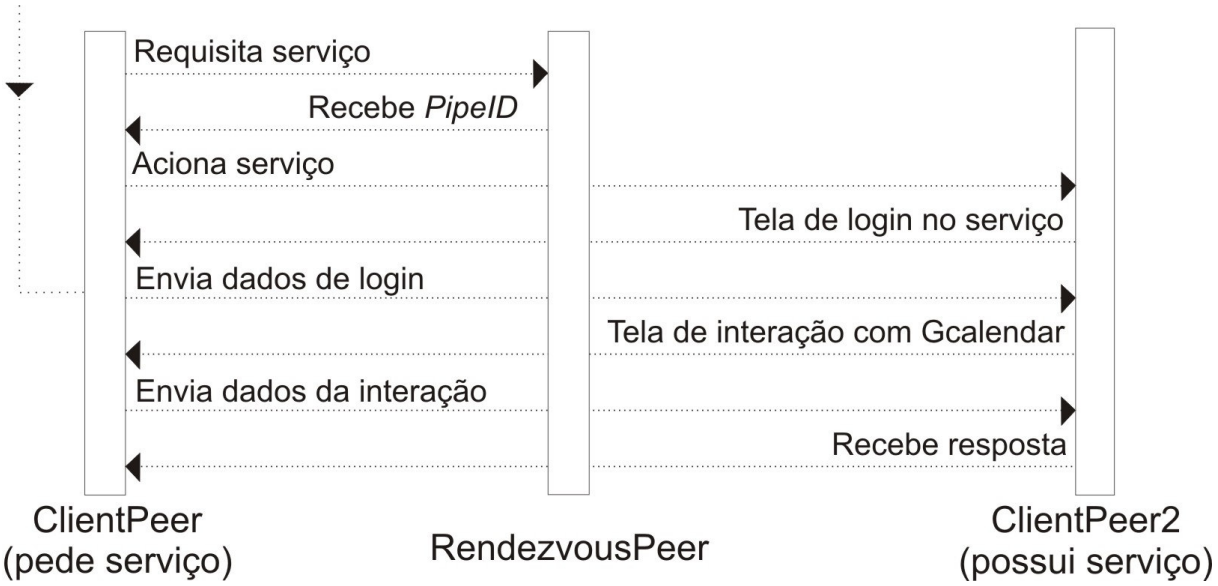

Figura 5 – Diagrama de seqüência do acesso ao serviço GCalendar.

Assim, verifica-se o acesso ao serviço (`GCalendar`) via rede JXTA (P2P), caracterizando um acesso pervasivo.

5 TESTES E RESULTADOS

Os testes foram realizados com base na aplicação desenvolvida no caso de uso, no qual foram empregadas as definições citadas no capítulo 4.

A interface, conforme figura 4, é simples, tendo em vista que o objetivo do trabalho é o acesso pervasivo via rede P2P. Ela mostra as redes e os pares conectados àquela rede (nesse caso o par `fulano` está conectado na `Rede1`), mostra, também, quem iniciou a aplicação onde o nome do par aparece ao lado do título da aplicação (como mostra a figura, o par que iniciou a aplicação foi `fiabane`). A figura também mostra os serviços disponíveis àquela rede e uma área para envio de mensagens.

A aplicação exemplo apresentou funcionamento satisfatório e seu desempenho, tanto na inscrição na rede, quanto na autenticação no serviço foram dentro do esperado, não apresentando outros erros além dos gerados na parte de autenticação dos dados de *login*.

Foram realizados testes quanto à comunicação da rede P2P e o acesso ao serviço Google Agenda via rede P2P. Todos os testes foram realizados em um computador com 512 MBytes de memória RAM, processador AMD Athlon(tm) XP 2500 + 1.83 GHz, sistema operacional Windows XP Service pack 2 e conexão ADSL 300 Kbps, com a execução de duas instâncias da aplicação.

Na parte de comunicação da rede P2P, os testes realizados envolveram a entrada e a saída da rede (conexão e desconexão) e a troca de mensagens entre duas instâncias de aplicações JXSE.

Todas as trocas de mensagens foram realizadas com sucesso, pois, como dito em capítulo anterior, a rede JXTA propicia esse ambiente heterogêneo devido à abstração que faz dos protocolos de rede e de como o dado será enviado.

Quanto à entrada e saída da rede, houve pequenos atrasos na atualização da rede para os demais pares, sendo natural esse acontecimento, pois esse processo é assíncrono. Esses pequenos atrasos não impediram o desempenho satisfatório da aplicação no que tange ao objetivo desse trabalho.

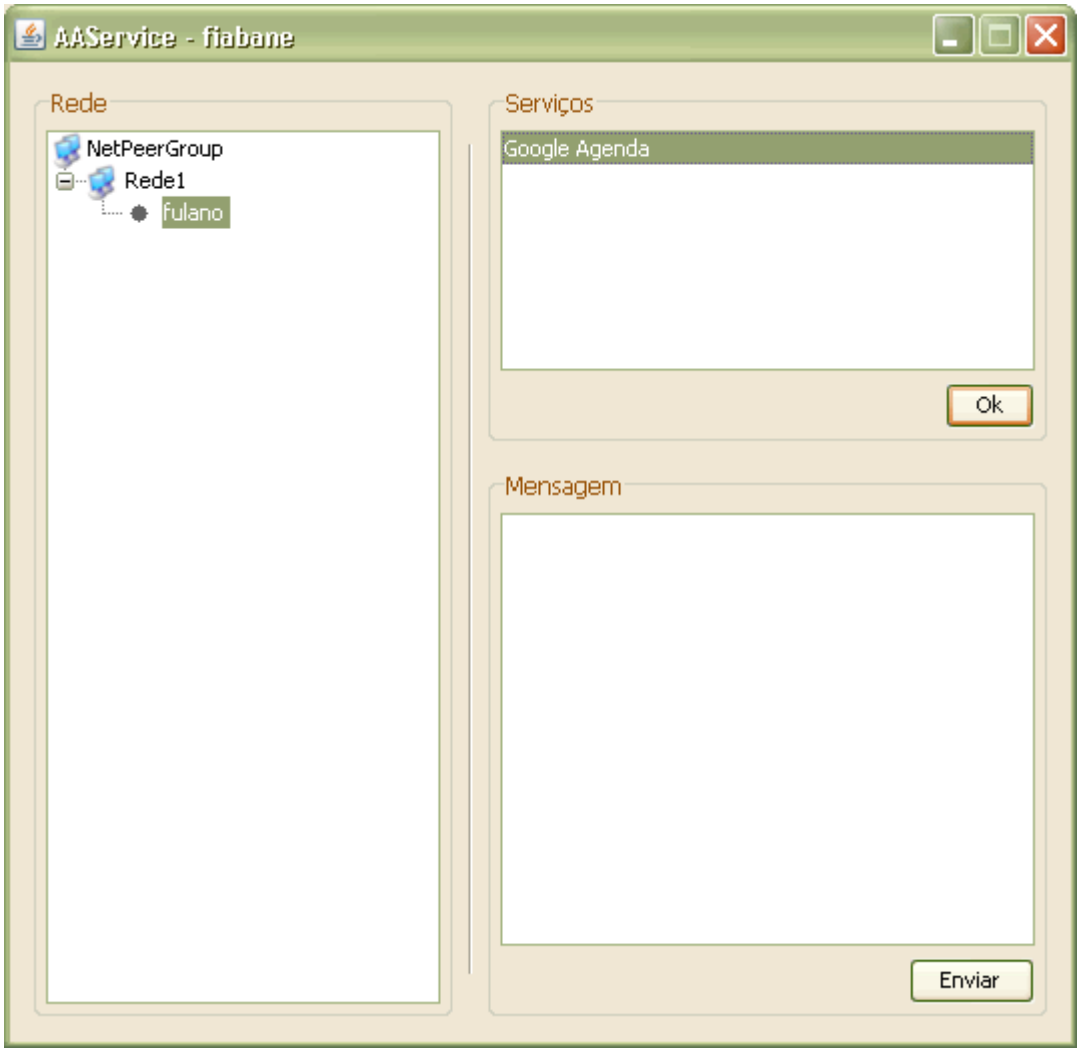

Figura 6 - Instância da aplicação exemplo exibindo a rede e os serviços disponíveis àquela rede.

No acesso ao serviço, segundo nos mostra a figura 5, tem-se, também uma interface simples e amigável, onde se podem ver os eventos referentes ao calendário que foi escolhido ao inscrever-se no serviço e campos para a inserção de novos eventos nesse calendário.

Figura 7 - Serviço Google Agenda.

Para autenticar-se no serviço, é preciso informar o usuário e a senha, que são as mesmas utilizadas para acessar o calendário via Internet.

Quanto aos testes de acesso ao serviço via rede JXTA, segundo a tabela 2, os resultados também foram satisfatórios, pois nos testes realizados de autenticação no serviço, inserção de um evento na agenda e exclusão de eventos a aplicação não apresentou erros.

Tabela 2 – Número e tamanho das mensagens trocadas na rede para entrada na rede e acesso ao serviço.

	Número de mensagens trocadas	**Mensagens Trocadas**	**Tamanho da mensagem**
Acesso rede (RendezvousPeer já existe)	5	Procura RendezvousPeer – RendezvousPeer responde – Envia dados para autenticação – Recebe autorização – Conecta.	Busca RendezvousPeer: 13 bytes. Conexão: 62 bytes. Autenticação: xx bytes (usuário + senha).
Acesso rede (RendezvousPeer ainda não existe)	8	Procura RendezvousPeer (3) – (Cria RedezvousPeer) RendezvousPeer responde – Envia dados para autenticação – Recebe autorização – Conecta.	Busca RendezvousPeer: 13 bytes. Conexão: 62 bytes. Autenticação: xx bytes (usuário + senha).
Acesso serviço	8	Requisita Serviço – Recebe PipeID – Aciona Serviço – Tela de Login no Serviço – Envia dados do Login – Tela de interação com o GCalendar – Envia dados da interação – Recebe Resposta	Autenticação: 39 bytes. Dados serviço: - Deletar evento: 17 bytes. - Inserir evento: xx bytes (título do evento + hora inicial + hora final + data atual + calendário).

Os resultados foram obtidos através da geração de *log* a cada mensagem enviada, contendo o tamanho do objeto, *advertisements* no caso de mensagens de tamanho fixo e *string* para mensagens de tamanho variável.

6 CONCLUSÃO

Com a oferta de softwares na internet como serviços e de espaços para armazenamento de dados, o conceito da Computação nas Nuvens vem adquirindo mais importância. Esse conceito, junto com o crescimento no uso de dispositivos móveis e de seus recursos, introduz a idéia de um fluxo maior e mais livre da informação, um ambiente pervasivo, onde a informação circula entre os diferentes tipos de dispositivos de maneira simples.

Junto com o crescimento do uso da Internet e da Computação nas Nuvens, vem a utilização de redes P2P. Situação facilmente compreendida pelos benefícios que ela oferece como a melhor utilização da rede, eliminando o tráfego redundante já que somente os interessados participam da comunicação. Porém, devido a esse crescimento, é preciso o desenvolvimento de ferramentas que promovam e tornem eficientes essas comunicações.

O JXTA é um grande aliado na promoção dessas ferramentas, pois ao abstrair muitos aspectos de rede para o desenvolvedor, permite que este possa se deter em outras áreas do desenvolvimento dessas ferramentas, como o desenvolvimento de serviços. Além disso, promove uma padronização e permite uma integração entre diferentes aplicações.

A eficiência da rede P2P aliada a API JXTA forma um conjunto robusto na criação de ambientes onde se tenha dispositivos heterogêneos, facilitando o fluxo de informações e de dados, possibilitando, também, o compartilhamento de recursos e serviços, criando assim um ambiente pervasivo.

Dessa forma, a utilização, neste trabalho, da plataforma JXTA com o conceito de rede P2P, possibilitou a modelagem de um acesso pervasivo a uma base de dados também pervasiva (conceito de Computação nas Nuvens) permitindo o acesso de qualquer recurso/serviço por qualquer dispositivo participante da rede, de maneira simples.

Os objetivos iniciais desse trabalho foram alcançados com êxito, pois foi possível criar um ambiente pervasivo, possibilitando assim a comunicação e o acesso a serviços por diversos tipos de dispositivos, melhorando, dessa forma, o fluxo da informação.

A aplicação exemplo requer aprimoramentos, principalmente, na parte de acesso aos serviços, sendo essa uma das partes do trabalho futuro, visando uma generalização desse acesso. Outra sugestão de aprimoramento a ser feito é o desenvolvimento da mesma aplicação só que utilizando as plataformas JXME e J2ME.

REFERÊNCIAS BIBLIOGRÁFICAS

AUGUSTIN, I.; LIMA, J.C.D.; YAMIN, A.C. **Computação Pervasiva**: como programar aplicações. SBPL, 2006.

AUGUSTIN, I.; LIMA, J.C.D; SILVA, M.L.; PEREIRA, R.P.; MIOTTO, R.R. **Explorando Adaptação Dinâmica ao Contexto no Projeto de Aplicações da Computação Pervasiva**. Santa Maria: SIRC/UNIFRA, 2004.

BEAL, V. All About Peer-to-Peer Architecture. **Webopedia, the first online encyclopedia dedicated to computer technology**, 2005. Disponível em: <http://www.webopedia.com/DidYouKnow/Internet/2005/*peer*_to_*peer*.asp>. Acesso em: 10 de Junho de 2008.

CLOUD COMPUTING. **Wikipédia, a enciclopédia livre**. Disponível em: <http://en.wikipedia.org/wiki/Cloud_computing>. Acesso em: 27 de Outubro de 2008.

DURANTE, G.B. Redes peer-to-peer. **Universidade Federal do Rio de Janeiro**. Disponível em: <http://www.gta.ufrj.br/grad/04_1/p2p/>. Acesso em: 15 de Agosto de 2008.

GOOGLE DATA APIs OVERVIEW. **Google Code**. Disponível em: <http://code.google.com/apis/gdata/overview.html>. Acesso em: 30 de Outubro de 2008.

GREENE, K. Google's Cloud Looms Large: How might expanding Google's cloud-computing service alter the digital world?. **Technology Review Published by MIT,** dez 2007. Disponível em: <http://www.technologyreview.com/Biztech/19785/page1/>. Acesso em: 28 de Outubro de 2008.

JXTA (TM) COMMUNITY PROJECTS. **JXTA connected**. Disponível em: <http://www.jxta.org>. Acesso em: 19 de Março de 2008.

KALOGERAKI, V.; PRUYNE, J.; MOORSEL, A. A Peer-to-Peer Architecture for Delivering E-Services. **Hewlett – Packard Laboratories**, jun. 2001. Disponível em: <http://www.hpl.hp.com/techreports/2001/HPL-2001-181.pdf>. Acesso em: 15 de Junho de 2008.

KOVER, A. **Napster:** The Hot Idea Of The Year. Fortune, vol.142, No. 1, página 128, jun. 2000.

KRISHNAN, N. The JXTA Solution to P2P. **Java World Solutions for Java developers**, out. 2001. Disponível em: <http://www.javaworld.com/javaworld/jw-10-2001/jw-1019-jxta.html?page=2>. Acesso em: 11 de Junho de 2008.

LOHR, S. Google and I.B.M. Join in 'Cloud Computing' Research. **The New York Times**, out 2007. Disponível em: <http://www.nytimes.com/2007/10/08/technology/08cloud.html>. Acesso em: 28 de Outubro de 2008.

NASCIMENTO, G. JXTA. **DevMedia – Java Magazine**, 2007. Disponível em: <http://www.devmedia.com.br/articles/viewcomp.asp?comp=6053 >. Acesso em: 02 de Novembro de 2008.

OLIVEIRA, R.S.; SILVA, J.; HORTA, G.T. Computação Ubíqua. **Universidade Federal do Rio de Janeiro**, Rio de Janeiro. Disponível em: <http://lens.cos.ufrj.br:8080/ESEWEB/materials/workshop1/rodrigo_ubiquidade.pdf>.Acesso em: 12 de Agosto de 2008.

P2P. **Wikipédia, a enciclopédia livre**. Disponível em: <http://pt.wikipedia.org/wiki/P2P> Acesso em: 16 de Março de 2008.

PEREIRA, R.P.; MIOTTO, R.R.; BELUSSO, R.C.; AUGUSTIN, I.; LIMA, J.C.D. **Comunicação entre Componentes da Aplicação em Ambientes Pervasivos.** Santo Ângelo: SRI/URI, 2005.

PONCE, E.E.; TAKEO, S.K. Introdução a Ambientes Inteligentes. **Universidade de São Paulo**, São Paulo. Disponível em: <http://www.pad.lsi.usp.br/humanlab/trabalhos/workshop_amb_intel.pdf>. Acesso em: 15 de Agosto de 2008.

RIZZETI, T. A. **Construção de um Ambiente Distribuído e Colaborativo para o projeto PDSCE**. 2006. 52f. Monografia (Graduação em Ciência da Computação) - Universidade Federal de Santa Maria, Santa Maria, 2006.

TRAVERSAT, B.; ARORA, A. Project JXTA 2.0 Super-Peer Virtual Network. 2006. Disponível em: <http://www.jxta.org/project/www/docs/JXTA2.0protocols1.pdf>. Acesso em: 15 de Junho de 2008.

WEISER, M. **Hot Topics:** Ubiquitous Computing. IEEE Computer, out. 1993.

WEISER, M. **Some Computer Science Problems in Ubiquitous Computing**. Communications of the ACM, jul. 1993.

WEISER, M. **The Computer for the Twenty-First Century**. Scientific American, pp. 94-10, set. 1991

WHAT ARE THE GOOGLE DATA APIs ?. **Google Code**. Disponível em: <http://code.google.com/apis/gdata/>. Acesso em: 30 de Outubro de 2008.

WILLIAN, R.S.S. Introdução às Redes Peer – to – Peer (P2P). **Universidade Federal do Rio de Janeiro**, Rio de Janeiro, jun. 2003. Disponível em: <http://www.gta.ufrj.br/seminarios/semin2003_1/william/>. Acesso em: 11 de Junho de 2008.

APÊNDICE A – Funções de interação com o serviço Google Calendar

printAllEvents

```
/**
   * Imprime todos os eventos de um calendário.
   *
   * @param service An authenticated CalendarService object.
   * @throws ServiceException If the service is unable to
handle the request.
   * @throws IOException Error communicating with the server.
   */
  private static void printAllEvents(CalendarService service)
      throws ServiceException, IOException {
    // Send the request and receive the response:
    CalendarEventFeed resultFeed =
service.getFeed(eventFeedUrl,
        CalendarEventFeed.class);
    listaeventos.removeAll(listaeventos);
    for (int i = 0; i < resultFeed.getEntries().size(); i++) {
      CalendarEventEntry entry =
resultFeed.getEntries().get(i);
      listaeventos.add(entry.getTitle().getPlainText());
      System.out.println(listaeventos.get(i));
    }
  }
```

createEvent

```
/**
   * Cria um evento.
   *
   * @param service An authenticated CalendarService object.
   * @param eventTitle Title of the event to create.
   * @param eventContent Text content of the event to create.
   * @param recurData Recurrence value for the event, or null
for
   *        single-instance events.
   * @param isQuickAdd True if eventContent should be
interpreted as the text of
   *        a quick add event.
   * @param wc A WebContent object, or null if this is not a
web content event.
   * @return The newly-created CalendarEventEntry.
   * @throws ServiceException If the service is unable to
handle the request.
   * @throws IOException Error communicating with the server.
   */
```

```
  private static CalendarEventEntry
createEvent(CalendarService service,
      String eventTitle, String eventContent, String
recurData,
      boolean isQuickAdd, WebContent wc) throws
ServiceException, IOException {
    CalendarEventEntry myEntry = new CalendarEventEntry();

    myEntry.setTitle(new PlainTextConstruct(eventTitle));
    myEntry.setContent(new PlainTextConstruct(eventContent));
    myEntry.setQuickAdd(isQuickAdd);
    myEntry.setWebContent(wc);

    // If a recurrence was requested, add it. Otherwise, set
the
    // time (the current date and time) and duration (30
minutes)
    // of the event.
    if (recurData == null) {
      Calendar calendar = new GregorianCalendar();
      DateTime startTime = new DateTime(calendar.getTime(),
TimeZone
          .getDefault());

      calendar.add(Calendar.MINUTE, 30);
      DateTime endTime = new DateTime(calendar.getTime(),
          TimeZone.getDefault());

      When eventTimes = new When();
      eventTimes.setStartTime(startTime);
      eventTimes.setEndTime(endTime);
      myEntry.addTime(eventTimes);
    } else {
      Recurrence recur = new Recurrence();
      recur.setValue(recurData);
      myEntry.setRecurrence(recur);
    }

    // Send the request and receive the response:
    return service.insert(eventFeedUrl, myEntry);
  }
```

createQuickAddEvent

```
  /**
   * Cria um evento simples (utiliza o método createEvent).
   *
   * @param service An authenticated CalendarService object.
   * @param quickAddContent The quick add text, including the
event title, date
   *        and time.
```

```
   * @return The newly-created CalendarEventEntry.
   * @throws ServiceException If the service is unable to
handle the request.
   * @throws IOException Error communicating with the server.
   */
  private static CalendarEventEntry createQuickAddEvent(
      CalendarService service, String quickAddContent) throws
ServiceException,
      IOException {
    return createEvent(service, null, quickAddContent, null,
true, null);
  }
```

addReminder

```
/**
   * Adiciona um lembrete em um evento do calendário.
   *
   * @param entry The event to update.
   * @param numMinutes Reminder time, in minutes.
   * @param methodType Method of notification (e.g. email,
alert, sms).
   * @return The updated EventEntry object.
   * @throws ServiceException If the service is unable to
handle the request.
   * @throws IOException Error communicating with the server.
   */
  private static CalendarEventEntry
addReminder(CalendarEventEntry entry,
      int numMinutes, Method methodType) throws
ServiceException, IOException {
    Reminder reminder = new Reminder();
    reminder.setMinutes(numMinutes);
    reminder.setMethod(methodType);
    entry.getReminder().add(reminder);

    return entry.update();
  }
```

deleteEvents

```
  /**
   * Deleta os eventos que estiverem na lista de eventos a
serem deletados.
   *
   * @param service An authenticated CalendarService object.
   * @param eventsToDelete A list of CalendarEventEntry
objects to delete.
   * @throws ServiceException If the service is unable to
handle the request.
```

```
   * @throws IOException Error communicating with the server.
   */
  private static void deleteEvents(CalendarService service,
      List<CalendarEventEntry> eventsToDelete) throws
ServiceException,
      IOException {

    // Add each item in eventsToDelete to the batch request.
    CalendarEventFeed batchRequest = new CalendarEventFeed();
    for (int i = 0; i < eventsToDelete.size(); i++) {
      CalendarEventEntry toDelete = eventsToDelete.get(i);
      // Modify the entry toDelete with batch ID and operation
type.
      BatchUtils.setBatchId(toDelete, String.valueOf(i));
      BatchUtils.setBatchOperationType(toDelete,
BatchOperationType.DELETE);
      batchRequest.getEntries().add(toDelete);
    }

    // Get the URL to make batch requests to
    CalendarEventFeed feed = service.getFeed(eventFeedUrl,
        CalendarEventFeed.class);
    Link batchLink = feed.getLink(Link.Rel.FEED_BATCH,
Link.Type.ATOM);
    URL batchUrl = new URL(batchLink.getHref());

    // Submit the batch request
    CalendarEventFeed batchResponse = service.batch(batchUrl,
batchRequest);

    // Ensure that all the operations were successful.
    boolean isSuccess = true;
    for (CalendarEventEntry entry :
batchResponse.getEntries()) {
      String batchId = BatchUtils.getBatchId(entry);
      if (!BatchUtils.isSuccess(entry)) {
        isSuccess = false;
        BatchStatus status = BatchUtils.getBatchStatus(entry);
        System.out.println("\n" + batchId + " failed (" +
status.getReason()
            + ") " + status.getContent());
      }
    }
    if (isSuccess) {
      System.out.println("Successfully deleted all events via
batch request.");
    }
  }
```