UNIVERSIDADE REGIONAL DE BLUMENAU

CENTRO DE CIÊNCIAS EXATAS E NATURAIS

CURSO DE CIÊNCIAS DA COMPUTAÇÃO – BACHARELADO

PROTÓTIPO DE SOFTWARE PARA COMPARTILHAR INFORMAÇÕES ENTRE COMPUTADORES ATRAVÉS DA TECNOLOGIA PEER-TO-PEER (P2P), USANDO A PLATAFORMA JXTA

JOSÉ VOSS JUNIOR

BLUMENAU
2004

2004/1-18

JOSÉ VOSS JUNIOR

PROTÓTIPO DE SOFTWARE PARA COMPARTILHAR INFORMAÇÕES ENTRE COMPUTADORES ATRAVÉS DA TECNOLOGIA PEER-TO-PEER (P2P), USANDO A PLATAFORMA JXTA

Trabalho de Conclusão de Curso submetido à Universidade Regional de Blumenau para a obtenção dos créditos na disciplina Trabalho de Conclusão de Curso II do curso de Ciência da Computação — Bacharelado.

Prof. Francisco Adell Péricas - Orientador

BLUMENAU
2004

2003/2-18

PROTÓTIPO DE SOFTWARE PARA COMPARTILHAR INFORMAÇÕES ENTRE COMPUTADORES ATRAVÉS DA TECNOLOGIA PEER-TO-PEER (P2P), USANDO A PLATAFORMA JXTA

Por

JOSÉ VOSS JUNIOR

Trabalho aprovado para obtenção dos créditos na disciplina de Trabalho de Conclusão de Curso II, pela banca examinadora formada por:

Presidente:	Prof. Francisco Adell Péricas, MSc. – Orientador, FURB
Membro:	Prof. Sérgio Stringari, MSc. – FURB
Membro:	Prof. Mauro Marcelo Mattos, Dr. – FURB

Blumenau, 01 de julho de 2004

Dedico este trabalho aos meus pais e aos meus dois irmãos que sempre incentivaram e deram a força necessária para que com muita luta e força de vontade pudesse concluir este sonho que parecia impossível.

Sempre tenha na sua vida sonhos, objetivos e metas, pois com a conquista de metas você realiza um objetivo e o objetivo é o primeiro passo para conquista de um sonho. Assim se você tiver muitas metas terá muitos objetivos e sonhos alcançados.

José Voss Junior

AGRADECIMENTOS

Agradeço primeiramente a Deus que me deu forças para poder vencer mais este degrau da vida.

Agradeço em especial meus pais Álida e José Voss que sempre tiveram um grande poder no incentivo as minhas conquistas e que sempre me apoiaram nas horas que mais precisei.

Aos meus dois irmãos Rafael e Paulo que sempre acreditaram no desafio de vencer.

À minha namorada Lissiana que teve um papel muito importante nesta última fase desse desafio.

Ao meu orientador, Francisco Adell Péricas, por acreditar na minha capacidade de concluir este trabalho.

A todos os professores que repassaram parte de seus conhecimentos e experiência para contribuir na melhor formação profissional.

RESUMO

Este trabalho apresenta a especificação e implementação de um protótipo de software para compartilhar arquivos e mensagens entre computadores ligados a uma rede local ou Internet. São apresentados conceitos pertinentes à tecnologia *peer-to-peer* (P2P) e como a mesma se comporta. Para implementação dos recursos disponibilizados pela tecnologia é utilizada uma plataforma chamada JXTA que, além de especificar um conjunto de protocolos, bibliotecas e serviços para redes *peer-to-peer*, possui uma implementação para linguagem de programação Java, com a qual o protótipo foi desenvolvido.

Palavras chaves: *Peer-to-Peer*; JXTA; Anúncio de *peers* e conteúdos.

ABSTRACT

This work presents a specification and implementation of a software prototype for sharing files and messages among computers linked together in a local area network or through the Internet. Concepts regarding a peer-to-peer (P2P) technology and its behavior are presented. To implement the resources provided by the technology, it is used a development platform called JXTA. This platform specifies a set of protocols and libraries for peer-to-peer networks. There is an implementation of this platform using the Java programming language, which was used to implement this prototype.

Key-Words: Peer-to-Peer; JXTA; Peers and content announcement.

LISTA DE ILUSTRAÇÕES

LISTA DE SIGLAS

FTP – *File Tranfer Protocol*

P2P – *Peer-to-Peer*

IP – Internet *Protocol*

TCP – *Transmission Control Protocol*

NAT – *Network Address Translation*

UDP – *User Datagram Protocol*

HTTP – *Hypertext Transfer Protocol*

SMTP – *Simple Mail Transfer Protocol*

TTL – *Time To Live*

XML – *eXtensible Markup Language*

DNS – *Dynamic Name Service*

URN – *Uniform Resource Names*

RAM – *Random Acces Memory*

UML – *Unified Modeling Language*

PDP – *Peer Discovery Protocol*

PIP – *Peer Information Protocol*

PRP – *Peer Resolver Protocol*

PBP – *Pipe Binding Protocol*

ERP – *EndPoint Routing Protocol*

RVP – *Rendezvous Protocol*

JAR – Java *Archives*

SUMÁRIO

1 INTRODUÇÃO

Com o crescimento exponencial do número de computadores ligados à Internet, vem crescendo também a necessidade de redes de computadores mais eficazes, não somente em relação a tecnologias de transmissão, mas também na forma que elas ocorrem. A arquitetura mais utilizada na Internet para comunicação, sem dúvida é a cliente/servidor, onde o cliente pode ser, desde um celular, *pocket*, *desktop* ou qualquer dispositivo que possui processador. Neste caso, todos clientes comunicam-se com um servidor central, o qual é responsável pela comunicação com todos os outros clientes envolvidos. Isso muitas vezes pode provocar uma sobrecarga excessiva no servidor, já que todas mensagens são propagadas pelo mesmo. Uma maneira de diminuir a sobrecarga do servidor é fazer com que cada cliente ligado à rede faça o papel tanto de servidor quanto cliente.

Em novembro de 1996, é liberada a primeira versão do ICQ (segundo definição encontrada (ICQ, 2004) significa '*I Seek You*') um aplicativo para compartilhamento de mensagens instantâneas, usando uma tecnologia conhecida como *Peer-to-Peer* (P2P), onde cada cliente conectado pode comunicar-se diretamente com outro cliente sem haver necessidade direta de um servidor, pois o cliente também faz este papel. Em 1999, surge o Napster (NAPSTER, 2004), uma aplicação para compartilhamento de arquivos de música entre computadores ligados à Internet, também usando P2P. Desde então, P2P vem ganhando um grande espaço.

Hoje existem diversas aplicações usando redes P2P. Alguns exemplos são o MSN Messenger (MICROSOFT CORPORATION, 2004) e AOL Internet Messenger (AOL, 2004) para compartilhar mensagens instantâneas e o projeto chamado Gnutella (GNUTELLA, 2004) para compartilhar arquivos. Cada uma destas aplicações implementa um conjunto de protocolos proprietários, onde não há compatibilidade entre diversas aplicações do mesmo segmento. Por exemplo, o MSN Messenger não troca mensagens com o AOL Internet Messenger e vice versa.

Em abril de 2001, foi proposto pela Sun Microsystems um projeto chamado de JXTA (JXTA, 2004) que tem o objetivo de especificar um conjunto de protocolos para redes P2P.

Este trabalho propõe-se a desenvolver um protótipo de software em redes P2P para compartilhar arquivos entre os computadores envolvidos na rede, baseado inteiramente no projeto JXTA e usando a linguagem de programação Java. Aspectos detalhados de como será

estabelecida comunicação entre *peers* sem a intervenção direta do servidor, serão esclarecidos no decorer do projeto.

1.1 OBJETIVOS DO TRABALHO

O objetivo principal deste trabalho foi desenvolver um protótipo de software para compartilhar arquivos e mensagens entre computadores ligados à Internet ou a redes internas.

Os objetivos específicos do trabalho são:

a) compartilhar arquivos de qualquer formato;
b) enviar mensagens para computadores;
c) utilizar bibliotecas da plataforma JXTA.

1.2 ESTRUTURA DO TRABALHO

O trabalho está organizado em cinco capítulos:

O capítulo inicial apresenta uma visão geral sobre o assunto que está sendo abordado neste trabalho, e sua importância para redes P2P.

O segundo capítulo expõe alguns conceitos das arquiteturas cliente/servidor e P2P apresentando uma breve descrição das mesmas. Em destaque a contextualização dos principais componentes da arquitetura P2P.

O terceiro capítulo apresenta as características da plataforma JXTA bem como descreve os conceitos utilizados na arquitetura P2P.

O quarto capítulo apresenta o protótipo de software desenvolvido, descrevendo especificação, características da implementação e seu funcionamento através de algumas de suas telas.

O quinto e último capítulo apresentam as conclusões obtidas no desenvolvimento deste trabalho e algumas sugestões para a sua continuidade.

2 INTERNET

Desde o surgimento da Internet em meados da década de 70 até os dias atuais, o número de computadores conectados à mesma cresce de forma exponencial. O fato é que a Internet se tornou uma rede gigantesca, ligando computadores espalhados por todo o mundo.

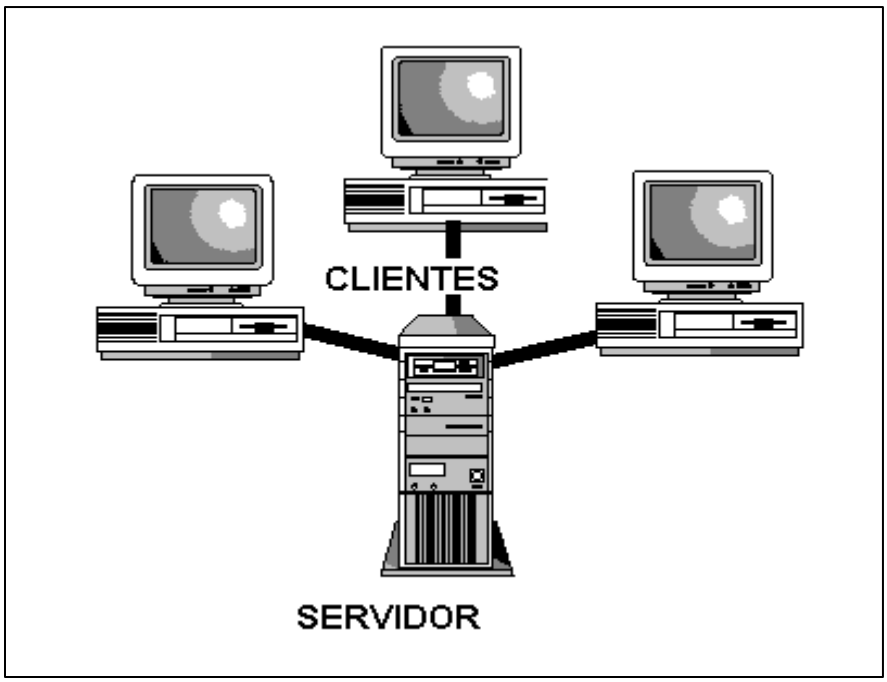
O surgimento de novas tecnologias tanto de software quanto de hardware influenciam diretamente na performance da comunicação entre os computadores, Também há influência direta dos meios de comunicação, cuja comunicação muitas vezes é realizado de forma precária. Redes mais eficazes podem ser conseguidas com meios de transmissão mais elaborados, também se deve levar em conta hardwares e softwares e como eles interagem.

Quando se fala em Internet logo já se pensa na arquitetura cliente/servidor, que atualmente é a mais utilizada. Nessa arquitetura cada cliente, que pode ser desde um celular, *pocket*, *desktop* ou qualquer dispositivo que possua processador está ligado direta ou indiretamente a um servidor, responsável por manter o cliente conectado. Essa é uma forma de interação onde todas as mensagens trocadas pelos clientes conectadas à rede passam obrigatoriamente pelo gerenciamento do servidor.

Há uma arquitetura conhecida como *Peer-to-Peer* (P2P), que propõe outra forma de interação, onde cada cliente conectado a rede pode comunicar-se diretamente com outro cliente sem haver a intervenção direta do servidor, pois o cliente faz os dois papéis.

2.1 ARQUITETURA CLIENTE/SERVIDOR

A maior parte dos serviços distribuídos na Internet usa a tradicional arquitetura cliente/servidor representada na Figura 1. Nessa arquitetura os clientes conectam-se ao servidor usando protocolos específicos, como o protocolo de transferência de arquivos (FTP), utilizado para obter arquivos do servidor.



Fonte: adaptado de Wilson (2004, p. 4)

Figura 1 – Arquitetura cliente/servidor.

Uma das vantagens dessa arquitetura, é que os clientes não necessitam de tanto recurso computacional, pois estão ligadas a um servidor responsável pelo processamento das requisições. O cliente de uma arquitetura cliente/servidor tem um papel passivo, capaz de utilizar um serviço oferecido pelo servidor, mas incapaz de oferecer serviços a outros clientes.

Este modelo segundo Wilson (2004, p. 15), foi desenvolvido quando a maioria dos computadores ligados à Internet tinha um endereço de IP (Protocolo de Internet) estático, ou seja, não havia troca de IPs a cada vez que um computador conectava-se à rede. Como a Internet cresceu de uma forma muito rápida, houve a necessidade de haver provedores de IPs dinâmicos, onde era alocado novo endereço a cada cliente conectado, pois muitos dos clientes conectavam-se através de redes *dial-up*.

Cada cliente conectado a um servidor forma uma extremidade da Internet, ou seja, não fornecem qualquer serviço a outros clientes. Por essa razão a maioria dos serviços estão centralizadas em um servidor com IP fixo, facilitando a execução por parte do cliente. Outra razão para o cliente executar serviços no servidor é por eles fazerem parte de uma rede privada localizada no próprio departamento da empresa.

2.2 ARQUITETURA TCP/IP

Segundo Comer (1999, p. 38) a arquitetura TCP/IP (*Transmission Control Protocol*/*Protocol* Internet) surgiu por causa do departamento de defesa do governo dos Estados Unidos da América (*Departament of Defense)* com o objetivo principal de manterem conectados órgãos do governo e universidades.

A ARPANET, um projeto de interligação de computadores criado pelo departamento de defesa dos Estados Unidos da América, surgiu como uma rede que permaneceria intacta caso um dos servidores perdesse a conexão, e para isso, ela necessitava de protocolos que assegurassem tais funcionalidades trazendo confiabilidade, flexibilidade e que fosse fácil de implementar. Foi desenvolvida então a arquitetura TCP/IP.

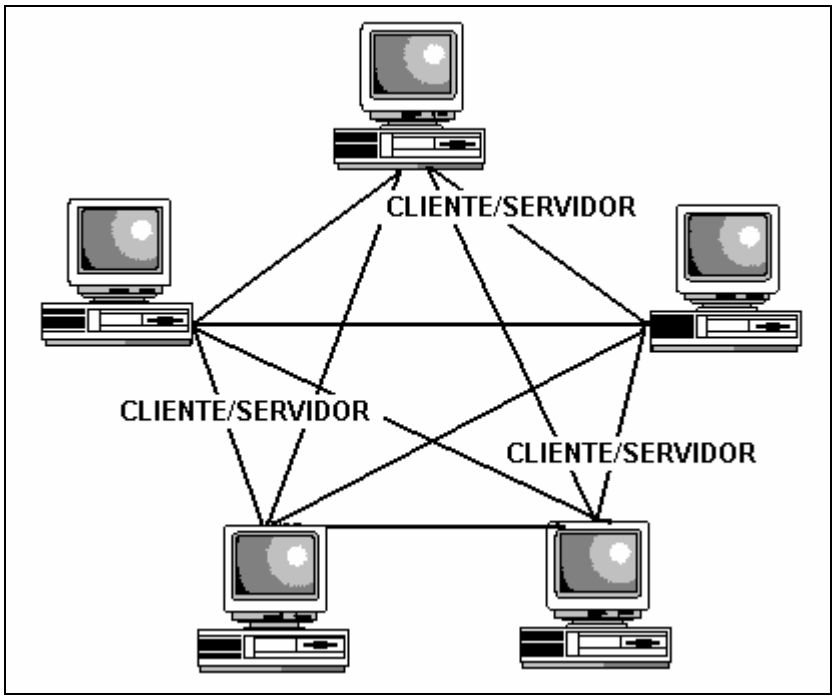
2.3 ARQUITETURA PEER-TO-PEER E CLIENTE/SERVIDOR

Segundo Silva (1998), a principal diferença entre arquiteturas P2P e cliente/servidor é o conceito de entidades, onde nessa última existe uma entidade que faz o papel de servidor e outras entidades que fazem o papel de clientes. Já na arquitetura P2P as entidades podem atuar ao mesmo tempo como servidores e como clientes.

Segundo Kellerer (apud SILVA, 2003):

> "Uma arquitetura de rede distribuída pode ser chamada P2P se os participantes compartilharem parte de seus próprios recursos de hardware. Esses recursos são necessários para prover os serviços e conteúdo oferecidos pela rede (ex. compartilhamento de arquivos ou espaços de trabalho para colaboração mútua). Os serviços e recursos são acessíveis por todos os pares sem necessidade de passar por nenhuma entidade intermediária. Os participantes dessa rede são tanto provedores de recursos como demandadores desses mesmos recursos".

Segundo Wilson (2004, p. 5), *peer-to-peer* é a chave para realização potencial de processamento intensivo, pois cada *peer* da rede possui mecanismo individual para prover serviço para cada um dos outros *peers*. Ao contrário da arquitetura cliente/servidor, redes P2P não dependem única e exclusivamente de um servidor centralizado para prover acesso a serviços. A Figura 2 mostra a representação de uma rede P2P, onde não há distinção de clientes e servidores.



Fonte: adaptado de Wilson (2004, p. 6)

Figura 2 – Arquitetura Peer-to-Peer.

A principal vantagem de uma rede P2P, segundo Wilson (2004, p. 5), é que ela permite a distribuição de responsabilidades em prover serviços para os *peers* da rede, eliminando o processamento excessivo em um único ponto, já no caso de uma arquitetura cliente/servidor é o próprio servidor que processa todas as requisições. Além disso, redes P2P podem ter uma melhor exploração de largura de banda da rede, usando para extremidades da Internet uma maior variedade de canais de comunicação. Ao contrário da comunicação da tradicional arquitetura cliente/servidor, que nas quais podem ser sobrecarregadas por rotas específicas para destinos muito acessados (por exemplo, a rota para livraria Amazon.com), P2P habilita comunicação por uma variedade de rotas de rede, reduzindo assim o congestionamento da mesma.

P2P tem a capacidade de servir recursos com alta disponibilidade por um menor custo, enquanto maximiza o uso dos recursos de todos os *peers* conectados à rede. Considerando que soluções cliente/servidor necessitam de equipamentos de rede e instalações diferenciadas para conseguir soluções robustas, P2P pode oferecer um nível de robustez semelhante, distribuindo as necessidades por toda rede.

Infelizmente, P2P oferece algumas desvantagens devido à natureza redundante da estrutura da rede que lhe faz necessária. A forma distribuída de canais dessa rede resulta em pedidos de serviços de natureza não-determinística. Por exemplo, clientes requisitam um mesmo arquivo da rede P2P e podem conectar-se em máquinas completamente diferentes por rotas de comunicação diferentes e com resultados diferentes. Pedidos enviados podem não ter resultados imediatos e em muitos casos pode não resultar em qualquer resposta.

Porém P2P pode superar todas essas limitações. Embora recursos possam muitas vezes desaparecer na rede, P2P pode implementar funcionalidades para replicar os recursos mais acessados em outros *peers* da rede, promovendo assim acesso redundante a este recurso. Quanto maior o número de *peers* que possui recursos replicados, menor será a probabilidade de um *peer* receber um pedido sem resposta. Em resumo, a mesma estrutura que pode causar problemas pode ser a solução para resolvê-los.

2.4 CONCEITOS PEER-TO-PEER

Será necessária uma introdução a terminologias e conceitos comuns a uma rede P2P, bem como esclarecer alguns aspectos inerentes a esse tipo de arquitetura.

2.4.1 PRINCIPAIS ELEMENTOS DE UMA REDE P2P

P2P é a resposta estratégica para as seguintes perguntas: como fazer a conexão entre diversos dispositivos ligados a uma rede e como fazer com que compartilhem informações, recursos e serviços entre eles. Segundo Wilson (2004, p. 15), estas simples perguntas somente poderão ser respondidas com o conhecimento de outras questões importantes:

a) como reconhecer um outro dispositivo presente na rede;
b) como organizar os dados compartilhados;
c) como publicar suas informações para os demais dispositivos;
d) como identificar unicamente um dispositivo;
e) como compartilhar dados na rede P2P.

Todas redes P2P possuem no mínimo os elementos necessários fundamentais para responder todas as perguntas acima. Infelizmente a maioria destes elementos é de difícil implementação, resultando em um código de programa inflexível.

2.4.1.1 PEER

Peer é um nodo na arquitetura P2P, é uma unidade fundamental de processamento de qualquer solução P2P. Pode ser um simples computador, um dispositivo móvel tal como um celular ou uma aplicação distribuída em vários computadores.

Segundo definição de Wilson (2004, p. 16), *peer* é qualquer entidade capaz de executar qualquer trabalho útil e comunicar os resultados desse trabalho para outras entidades da rede P2P, podendo ser direta ou indiretamente. A definição de trabalho útil depende do tipo do *peer*. Existem três possibilidades de tipos de *peer*, sendo que cada uma tem responsabilidade específica. Os tipos são:

a) *peer* simples: designado unicamente para servir o usuário, permite prover e consumir serviços providos por outros usuários da rede;
b) *peer* de encontro: provê o encontro de *peers* de um determinado ponto de rede, encontra outros *peers* e seus respectivos serviços. Resolve questões de descoberta de serviços. Propaga mensagens entre *peers* da rede. Pode armazenar informações de *peers* para uso futuro ou para requisições de outros *peers* de encontro. Pode usualmente ser encontrado em redes internas e pode servir de ponte para comunicação com outras redes internas, mas não possui a capacidade de atravessar *firewalls*, isso fica a cargo do *peer relay*;
c) *peer relay*: possui mecanismo para comunicação com outros *peers* separados por *firewalls* de rede ou um equipamento NAT (tradução de endereço de rede). Este mecanismo será visto com mais detalhes no tópico “Atravessando *firewalls* e NAT”. Para um *peer* enviar mensagem para outros, primeiro deve ser determinada a rota com o destino do *peer*. O *peer* de rota possui mecanismo para encontrar dispositivos que possuem IP dinâmico.

2.4.1.2 GRUPO DE PEERS

São grupos de *peers* formados para servir interesses comuns. Podem prover serviços para outros membros do grupo e limitar o acesso a *peers* de outras redes. Segundo Wilson (2004, p. 18), grupo de *peers* são divididos de acordo com os seguintes itens:

a) tipo de aplicação: um grupo de *peers* é formado para trocar informações com aplicações que possuem as mesmas características de funcionamento. Um exemplo disso é um grupo de aplicações de troca de mensagens;

b) segurança dos *peers*: um grupo de *peers* pode exigir autenticação dos usuários participantes, pois questões de sigilo podem estar envolvidas;
c) informação de status de membros do grupo: membros de um grupo de *peers* podem monitorar outros. Informação de *status* pode ser usada para manter um nível mínimo de serviço para outros *peers*.

2.4.1.3 REDE DE TRANSPORTE

Para troca de dados, *peers* devem possuir uma série de mecanismos para transmitir os dados pela rede. Essa camada se chama rede de transporte, e é responsável por todos aspectos de transmissão dos dados, incluindo a segmentação de pacotes e a adição de cabeçalhos apropriados para controle de destino do pacote. A rede de transporte pode ser, transporte em baixo nível como UDP ou TCP, ou transporte em alto nível como HTTP ou SMTP.

O conceito de rede de transporte em P2P pode ser dividida em três partes:

a) pontos de fim (*endpoint*): uma fonte de destino inicial ou final para qualquer informação transmitida pela rede. Estes pontos de fim correspondem a interfaces usadas para enviar e receber dados da rede;
b) duto (*pipe*): canal virtual, unidirecional e assíncrono para comunicação de dois ou mais pontos de fim (*endpoint*);
c) mensagens: contêm dados para serem transmitidos sobre um duto de um ponto para outro.

Para comunicação usando dutos, o *peer* primeiro necessita achar os pontos finais (*endpoint*), um ponto para ser a fonte da mensagem e outro para cada destino da mensagem um ponto para ser a fonte da mensagem e outro para cada destino da mensagem, assim se forma um canal virtual entre cada um dos pontos. Quando este canal virtual estiver formado as mensagens podem ser transmitidas de um ponto a outro.

Para enviar dados de um *peer* para outro, deve se utilizar mensagens, pois estas contêm os dados a serem transmitidos, *peers* utilizam *output pipe* para enviar dados e o *peer* que recebe a mensagem utiliza *input pipe* para extrair os dados da mensagem recebida.

Um duto (*pipe*) de comunicação entre *peers* pode ser somente unidirecional, sendo assim para haver comunicação entre dois *peers* deve haver dois dutos de comunicação entre

eles. Embora uma comunicação bidirecional faça parte de qualquer rede moderna existente, uma comunicação unidirecional pode ser modelada em qualquer ambiente de rede.

2.4.1.4 SERVIÇOS

Serviços provêem funcionalidades para *peers*, ou seja é a realização de qualquer trabalho útil dentro de uma rede P2P. Este trabalho útil pode ser transferir arquivos, prover informações de *status*, executar cálculos ou fazer qualquer coisa que seja de proveito para outros *peers*. Os serviços são a motivação da junção de dispositivos dentro de uma rede, sem serviços não haveria rede P2P.

Serviços podem ser divididos em duas categorias:

a) serviços de *peer*: serviço oferecido por um *peer* para outros *peers* da rede. A capacidade deste serviço é provida por um único *peer*, e está disponível somente quando este *peer* estiver conectado a rede. Quando o *peer* não estiver conectado, o serviço estará indisponível;
b) serviços de grupos de *peer*: serviço oferecido por um grupo de *peers*. A capacidade deste serviço é aumentada em relação a um serviço de *peer*, pois quem oferece o serviço são vários *peers* de um grupo e não somente um. Os serviços serão redundantes garantindo assim que os mesmos estarão disponíveis mesmo que alguns *peers* não estejam conectados.

2.4.1.5 ANÚNCIOS

A estrutura de representação de uma entidade, serviço, ou recurso deve estar disponível para um *peer* ou para um grupo de *peers* como parte de uma rede P2P. Todo serviço disponível deve ser anunciado para que outros *peers* possam utilizá-lo.

Os anúncios podem ser de *peers*, grupos de *peers*, dutos, pontos de fim (*endpoint*). Também podem ser anunciados todos serviços oferecidos por um *peer* ou grupo de *peers*.

2.4.1.6 PROTOCOLOS

Toda troca de dados utiliza protocolos que padronizam a forma de enviar ou receber qualquer informação. Protocolo é um modo de estruturar a troca de informações entre duas ou mais partes, usando regras previamente estabelecidas e de conhecimento de todas as partes envolvidas.

Em P2P, protocolos são necessários para definir todos os tipos de interações entre *peers*. A organização de informações em anúncios simplifica a representação dos dados, simplificando assim a definição dos protocolos.

Aplicações P2P utilizam um conjunto de protocolos previamente estruturados para comunicação, deve haver um mínimo de funcionalidades especificadas como:

a) procura de *peers* em uma rede;
b) procura de serviços providos por um *peer* ou grupo de *peers*;
c) obtenção de *status* de um *peer*;
d) invocar serviço de um *peer*;
e) criação, união e desvinculação de um *peer* de um grupo;
f) rotas de mensagens para outros *peers*.

2.4.1.7 NOMEANDO ENTIDADES

A maioria dos componentes de uma rede P2P necessita de partes de informações unicamente identificadas, a seguir a descrição desses componentes:

a) *peers*: *peers* necessitam uma identificação única para que outros *peers* possam localizá-lo. Essa identificação particular do *peer* pode ser necessária para permitir que uma mensagem seja roteada de maneira correta;
b) grupo de *peers*: o *peer* necessita de um modo para identificar o grupo e realizar ações. Uma ação pode ser a união do *peer* ao grupo, obtenção de *status* do grupo ou desvinculação de alguns *peers* do grupo;
c) dutos: para permitir comunicação, *peers* necessitam de algum modo para identificar o duto no qual os dados irão ser transmitidos;
d) conteúdos: as partes de um conteúdo necessitam ser univocamente identificadas para serem reconhecidas em qualquer ponto da rede P2P, embora possa existir partes redundantes.

Em um sistema de identificação única de nomes consegue-se uma maior flexibilidade, facilitando a comunicação entre toda estrutura da rede.

.

2.4.2 COMUNICAÇÃO P2P

Um dos problemas fundamentais de uma rede P2P é como ativar a troca de serviços entre dispositivos a ela ligados. Para resolver este problema é necessário responder duas questões importantes:

a) como um dispositivo encontra *peers* e serviços em uma rede P2P;
b) como dispositivos participantes em redes locais participam de redes P2P.

A primeira questão é importante porque, sem o conhecimento da existência de um *peer* ou serviço de rede, não é possível o dispositivo fazer parte de rede. A segunda questão é importante porque muitos dispositivos em uma rede P2P serão separados da rede com equipamentos designados para prevenir ou restringir conexões diretas entre dois dispositivos em diferentes redes internas.

2.4.2.1 LOCALIZAÇÃO DE ANÚNCIOS

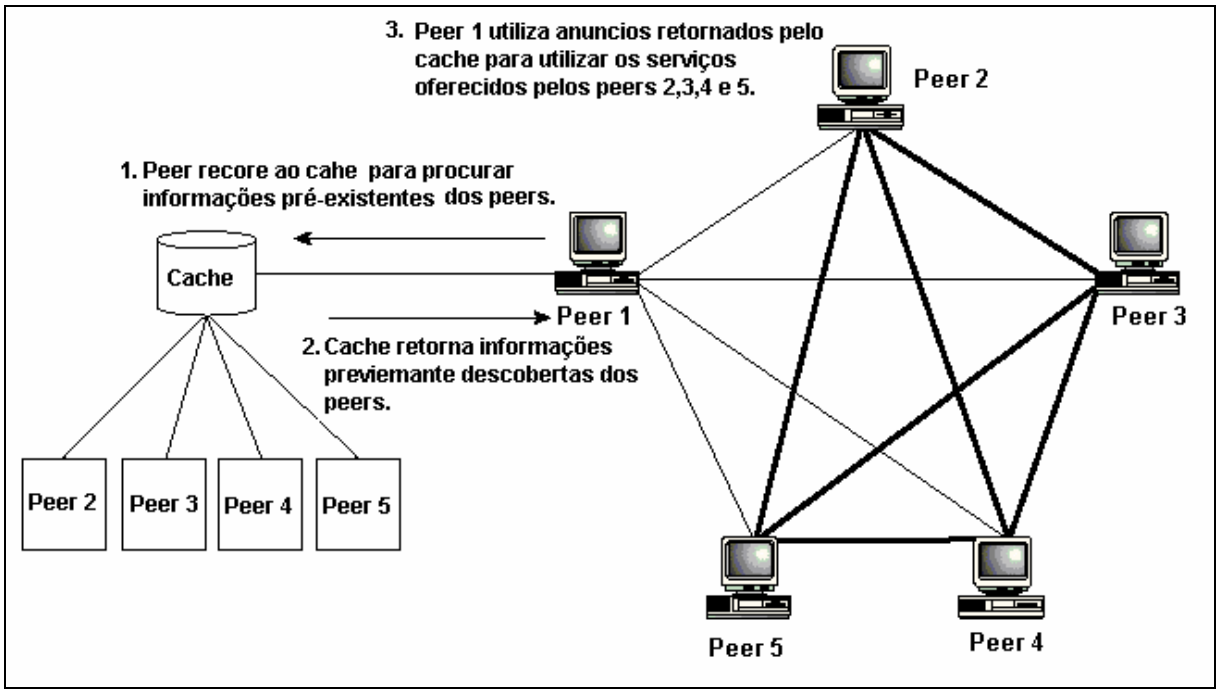
Conforme citado em tópicos anteriores, anúncios podem ser *peers*, grupos de *peers*, dutos, pontos de fim (*endpoint*) ou qualquer serviço oferecido por *peer* ou grupo de *peer*. Um *peer* pode descobrir um anúncio de três maneiras:

a) descoberta em *cache*;
b) descoberta direta;
c) descoberta indireta.

A primeira técnica não envolve nenhuma conectividade de rede e pode ser considerada uma técnica de descoberta passiva. A outras duas técnicas que envolvem conectividade de rede e são consideradas descobertas ativas.

2.4.2.1.1 DESCOBERTA EM *CACHE*

A maneira mais fácil para um *peer* descobrir um anúncio, é eliminar completamente o processo de descoberta. Ao invés de procurar ativamente um anúncio em uma rede P2P, o *peer* pode ter um anúncio previamente descoberto, pois pode ter armazenado em seu *cache* informações de anúncio de *peers* anteriormente descobertos, como mostrado na Figura 3. Embora este método possa ser trivial, ele pode reduzir a quantia de tráfego gerado por *peers* e permite resultados quase instantâneos, ao contrário do que acontece com a descoberta ativa.



Fonte: adaptado de Wilson (2004, p. 23)

Figura 3 – Peer de descoberta usando *cache* de anúncios.

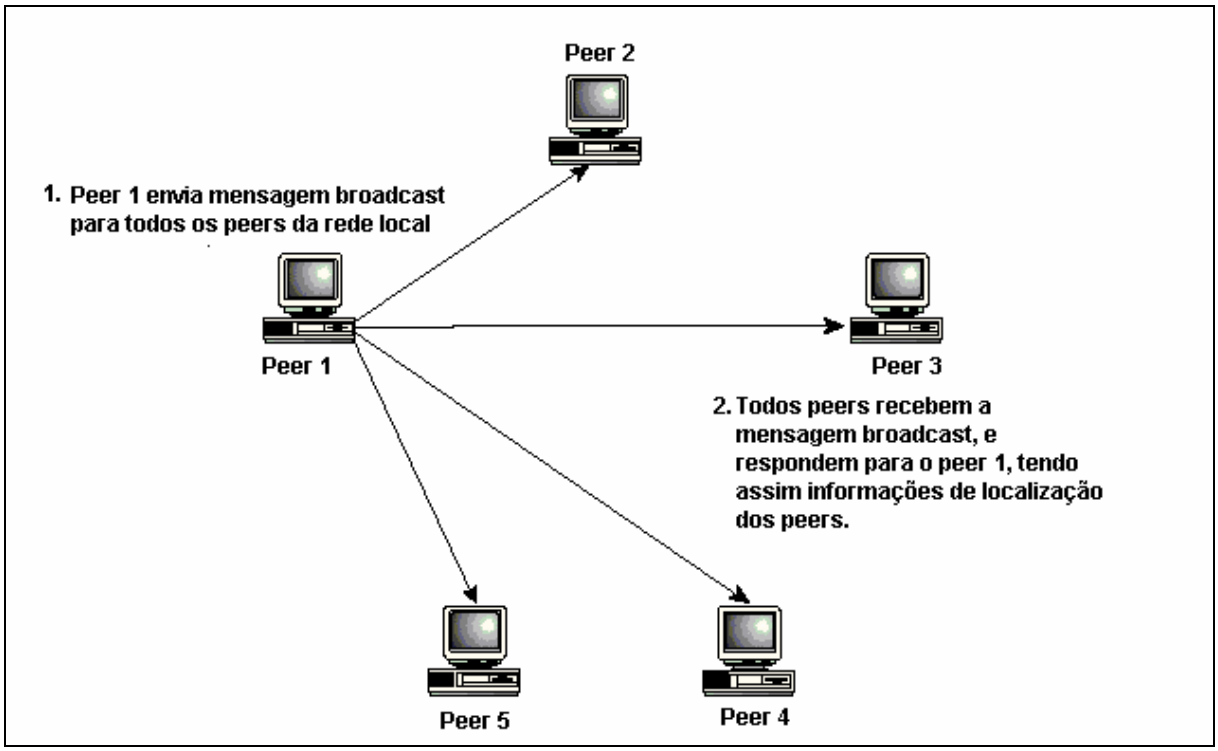
O *cache* em sua forma mais simples pode ser apenas um arquivo texto contendo uma lista de endereços de IP e portas previamente descobertas por *peers* de encontro, providenciando assim um ponto de partida para descoberta de serviços. Outro tipo de *cache* pode ser uma base de dados contendo todos anúncios descobertos pelos *peers*.

Embora o uso de *cache* tem a vantagem de diminuir o tráfego de rede, tem a desvantagem de guardar informações de *peers* ou serviços que podem não mais existir. Sendo assim quando um *peer* encontra no *cache* uma informação que não está mais disponível, terá que usar um processo de descoberta direta ou indireta, além de ter que remover as informações não mais válidas do *cache.* Um modo que melhora o processo de armazenamento em *cache é* determinar um tempo para expiração de cada informação de serviço armazenado. Antes de usar um anúncio é verificado a data de sua expiração: caso tenha seu prazo de validade vencido as informações deste serviço são removidas do *cache.*

2.4.2.1.2 DESCOBERTA DIRETA

Peers existentes dentro de uma rede local podem ser capazes de descobrir cada um dos outros *peers* diretamente, sem haver intermediação de *peers* de encontro no processo de

descoberta. Descoberta direta pode utilizar *broadcast* ou *multicasting*, que são capacidades nativas de uma rede, como mostrado na Figura 4.



Fonte: adaptado de Wilson (2004, p. 25)

Figura 4 – Descoberta direta de peers.

Quando outro *peer* descoberto utiliza este mecanismo, o *peer* pode descobrir outros anúncios comunicando-se diretamente com este peer sem utilizar *broadcast* ou *multicasting.*

Contudo essa técnica de descoberta de anúncios só é válida para redes locais. Para descobrir anúncios em outras redes como a Internet, por exemplo, terá que se fazer uso de peers de rotas, que são peers com capacidades específicas para este fim.

2.4.2.1.3 DESCOBERTA INDIRETA

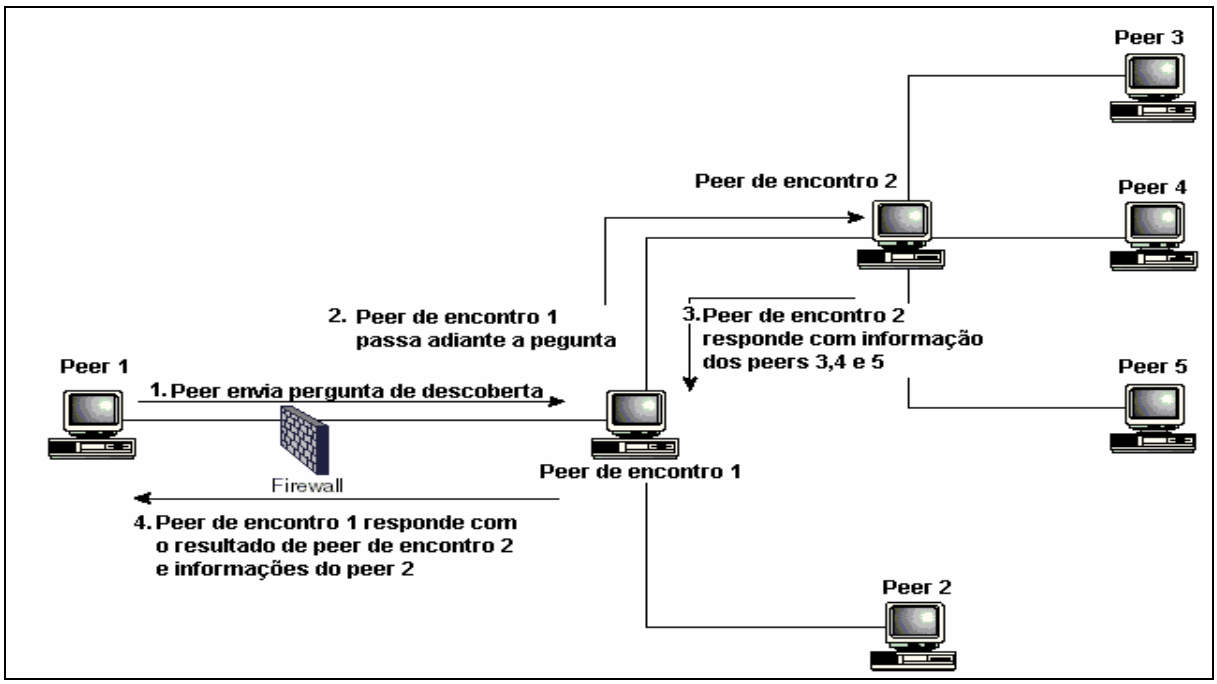
Descoberta indireta necessita do uso de peers de encontro para atuar como fonte de conhecimento em anúncios de peers. Essa técnica pode ser utilizada tanto em redes locais como em outras redes externas, como por exemplo, a Internet.

Peers de encontro provêem dois modos para localização de peers e anúncios:

a) propagação: o peer de encontro passa a requisição de descoberta para outros peers de seu conhecimento, incluindo outros peers de encontro, que podem continuar a propagar a requisição;

b) anúncios em *cache*: a mesma maneira que peers simples podem utilizar *cache* de anúncio para reduzir tráfego na rede, peers de encontro podem utilizar *cache* de anúncio para responder requisições de outros peers.

A utilização dos modos de localização por propagação e por anúncios em *cache* é representada na Figura 5. Este tipo de propagação provê uma efetiva solução para procura de anúncios em redes que possuem peers simples e peers de encontro.



Fonte: adaptado de Wilson (2004, p. 26)

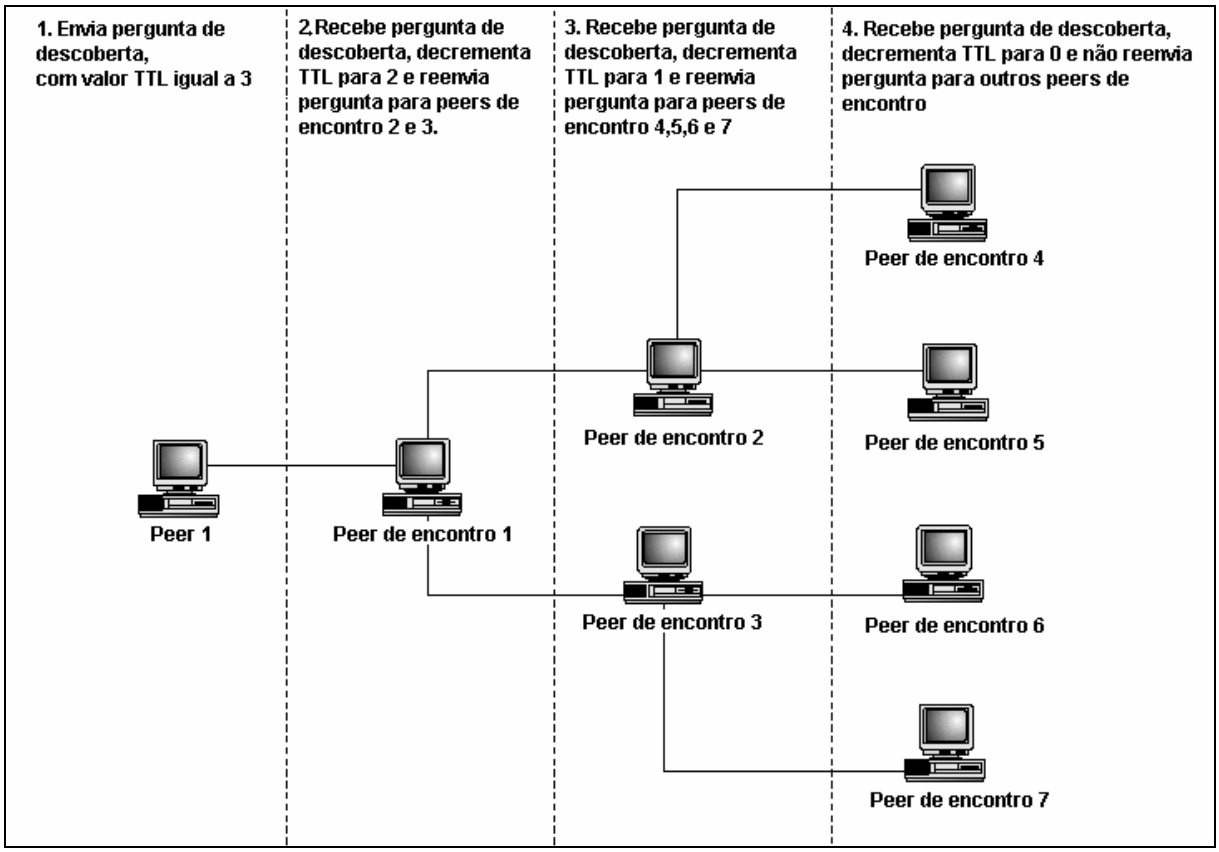
Figura 5 – Descoberta indireta com peers de encontro.

Embora o uso de *cache* reduz o tráfego de rede, as perguntas realizadas por outros peers propagam-se para outros peers de encontro sem restrição alguma, podendo levar a um sério problema de congestionamento na rede.

Quando um peer de encontro recebe uma requisição de descoberta ele repassa para outros peers de encontro conhecidos, ou seja, uma requisição de entrada é repassada para várias outras saídas. Este tipo de retransmissão amplifica a descoberta de um anúncio, mas pode acarretar em sobrecarga excessiva de rede.

Para prevenir uma excessiva propagação de pedidos, mensagens incorporam um atributo que contém um tempo de vida (TTL). Este tempo de vida é expresso em um número máximo de perguntas a serem propagadas entre peers da rede. Como mostrado na Figura 6,

quando um peer de encontro recebe uma mensagem que contém pergunta de descoberta, a propriedade TTL é decrementada em uma unidade, e caso o valor de TTL for zero a mensagem é descartada. Caso contrário, a mensagem é propagada para outros peers utilizando o novo valor de TTL.



Fonte: adaptado de Wilson (2004, p. 28)

Figura 6 – TTL em propagação de descobertas.

2.4.2.2 ROTAS E PEERS DE ENCONTRO

Para muitos peers existentes em redes locais, a procura de rotas é algo bem crítico, pois a rede pode estar protegida por algum tipo de *firewall.* Estes peers não podem utilizar descobertas diretas, pois precisam de peers de encontro para comunicação externa e de peers de rotas para buscar rotas externas.

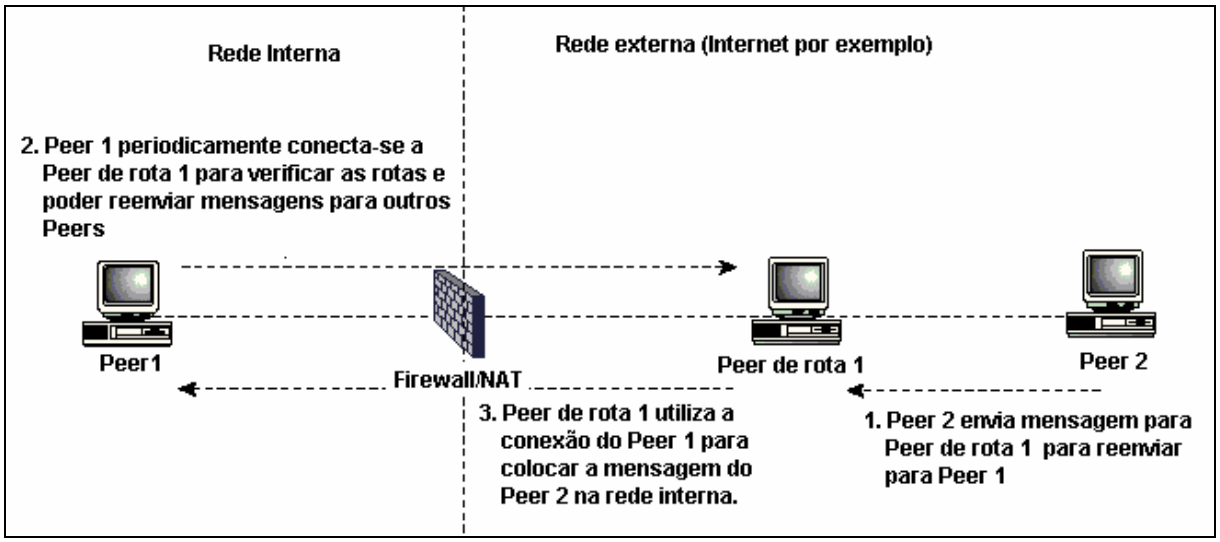
Segundo Wilson (2004, p. 34), em algumas aplicações P2P, como exemplo o projeto Gnutella (GNUTELLA, 2004) a procura de anúncios é realizada usando endereços de IP estáticos, ou seja, peers internos protegidos por *firewalls* tomam conhecimento de endereços de peers de encontro externos, e estes são responsáveis pela procura de rotas e outros peers.

2.4.2.3 ATRAVESSANDO *FIREWALLS* E NAT

O uso combinado de *firewalls* e NAT resulta em uma especial dificuldade para a comunicação P2P. Peers externos não podem se conectar a redes protegidas por NAT, a menos que o peer localizado internamente a rede inicia a comunicação. Além disso, conexões podem ser bloqueadas por *firewalls* que se baseiam no protocolo de conexão ou em destinos de endereço IP. A única ferramenta que um peer possui a disposição para resolver alguns destes problemas, é sua capacidade para criar conexões de rede em peers fora da proteção de *firewalls*. Peers podem usar protocolos permitidos pelo *firewall* para criar um tunelamento para a rede externa. Porém, se um *firewall* é configurado para negar todas as conexões que partem da rede interna, a comunicação do peer se torna impossível.

Segundo Wilson (2004, p. 32), na maioria das redes internas, o protocolo HTTP (*Hypertext Transfer Protocol*) é geralmente habilitado para iniciar conexões. Infelizmente, cada conexão HTTP envia um pedido e aguarda a resposta, sendo assim quando uma conexão faz um pedido inicial de comunicação terá que ficar aberto até receber uma resposta. Embora HTTP possa prover mecanismos para enviar pedidos da rede interna, não provê a capacidade para peers externos cruzarem o limite dos *firewalls*.

Para solucionar este problema, um peer de rota dentro de uma rede interna protegido por um *firewall* utiliza um peer localizado fora da proteção do mesmo, como mostrado na Figura 7. O peer tentando se comunicar com outro peer protegido por um *firewall* conecta-se a um peer de rota. Quando o peer interno estabelece a conexão, qualquer mensagem pode entrar, assim todas mensagens direcionadas para o peer interno são liberadas para comunicação.



Fonte: adaptado de Wilson (2004, p. 33)

Figura 7 – Travessia em *Firewall*/NAT

2.5 COMPARAÇÃO ENTRE SOLUÇÕES P2P

Utilizando alguns conceitos definidos na seção anterior é possível comparar soluções P2P proprietárias.

2.5.1 NAPSTER

O Napster (NAPSTER, 2004) foi à aplicação propulsora da plataforma P2P. Consistia em uma aplicação de rede P2P híbrida, pois havia um servidor centralizado onde eram realizadas todas as funcionalidades de procura de anúncios e de peers. Podia ser modelado como um único peer de encontro e diversos peers simples, todos utilizando TCP como rede de transporte. O peer de encontro provê a todos os peers simples a capacidade de localizar arquivos de música anunciados em um único arquivo, endereço de IP e porta de comunicação. Todos os peers simples utilizam essas informações para se conectarem diretamente com outros peers, e assim realizar cópias dos arquivos compartilhados.

Napster não prove uma solução completa para evitar *firewalls*, pois é capaz de atravessar somente um. Cada peer atua com uma simples rota, não possui capacidade para habilitar outros peers como peers de rotas.

2.5.2 GNUTELLA

Na rede Gnulella (GNUTELLA, 2004) cada peer atua como peer simples, peer de encontro e peer de rota, utilizando TCP como rede de transporte e HTTP para transferência de arquivos. As procuras realizadas na rede são propagadas por todos os peers conhecidos deste peer, e estes propagam para os seus peers conhecidos. Anúncio de conteúdo consiste em endereço IP, número da porta e um índice que identifica o arquivo no peer hospedeiro. Também são anunciados detalhes do arquivo como nome e tamanho. Como acontece com o Napster o Gnutella, pode atravessar apenas um *firewall.*

2.5.3 KAZAA

É uma aplicação P2P criada recentemente, e, segundo Good (2004), possui algumas inovações, tal como a facilidade para compartilhar arquivos com outros usuários, pois permite que seja feita a cópia de múltiplos arquivos simultaneamente. Permite também que sejam compartilhados diversos tipos de formatos, tal como formatos para músicas, vídeos, documentos, planilhas, fotos entre muitos outros. Permite que seja efetuada a cópia particionada de arquivo, onde um mesmo arquivo é copiado de diversos usuários ao mesmo tempo.

Kazaa possui um tipo de arquitetura puramente P2P, ou seja, não tem necessidade gerenciamento explícito por um servidor central.

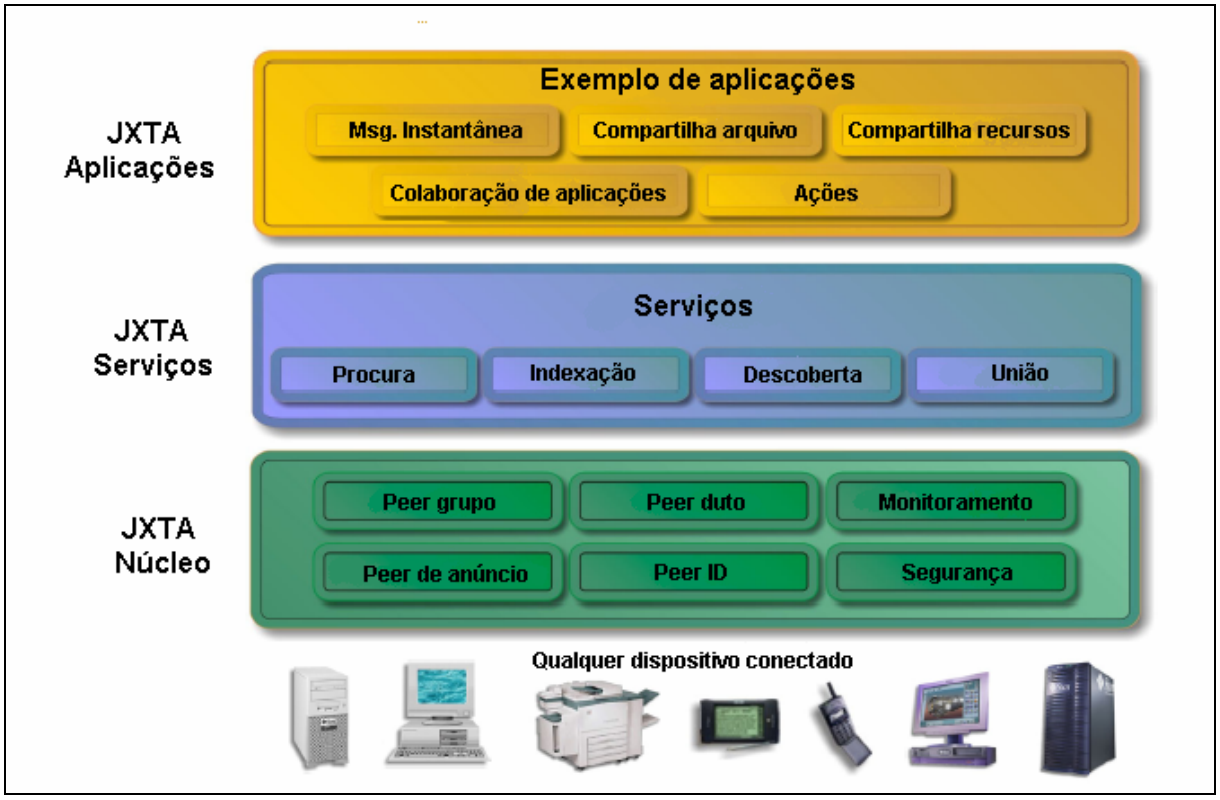
3 PLATAFORMA JXTA

Conforme JXTA PROGRAMMER GUIDE (2004), JXTA é uma plataforma aberta, uma solução genérica para plataforma P2P. Especifica um conjunto de protocolos para conexão de qualquer dispositivo de rede, tal como telefone celular, *desktop* e servidores. Os protocolos JXTA são independentes de linguagem de programação e de sistema operacional. Esta plataforma chamada de *Project* JXTA (JXTA, 2004) foi proposta em Abril de 2001, por Bill Joy, cientista chefe da Sun Microsystems. Além de manter os protocolos JXTA, que agora estão na sua versão 2.2, o projeto JXTA é responsável pela implementação de referencia dessa plataforma, tendo também que controlar o código fonte de uma variedade de projetos disponíveis no *site* do projeto.

Atualmente o projeto JXTA tem uma implementação de referência disponível na linguagem de programação Java, e com implementações na linguagem C, Objective-C, Ruby e Perl 5.0. Neste momento o projeto JXTA é utilizado por uma variedade de projetos nas mais diversas áreas de conhecimento, como gerenciamento de conteúdo, inteligência artificial e segurança de sistemas.

3.1 ARQUITETURA JXTA

A arquitetura da plataforma JXTA está dividida em três camadas: núcleo JXTA, serviços JXTA e aplicações JXTA. A Figura 8 mostra uma visão dessa arquitetura.



Fonte: adaptado de JXTA PROGRAMMER GUIDE (2003, p. 5)

Figura 8 – Arquitetura de camadas de JXTA

As camadas são as seguintes:

a) núcleo: essa camada, também conhecida como JXTA *core*, encapsula funcionalidades básicas comuns a uma rede P2P. Possui mecanismos chaves, onde se destaca a função para algoritmos de encriptação, entrada baseada em senha/usuário e mecanismos para criação de *peers* e grupo de *peers*;

b) serviços: essa camada possui serviços de rede que podem não ser absolutamente necessários para que uma rede P2P possa operar, mas é comum e desejável que a possua. Alguns exemplos de serviços de rede podem ser indexação e procura de arquivos, sistema de armazenamento, compartilhamento de arquivos, serviços de autenticação e serviços de chave pública;

c) aplicações: essa camada possui de fato a integração às aplicações, tal como mensagens instantâneas, compartilhamento de documentos e recursos, gerenciamento de conteúdos e descobertas, aplicações de correio eletrônico, sistemas distribuídos e muitos outros.

O limite entre serviços e aplicações não é rígido, ou seja, uma aplicação para um cliente pode ser visto como um serviço para outro. O projeto JXTA é projetado para ser modular, permitindo ao desenvolvedor poder escolher entre uma coleção de serviços e aplicações tal como é de sua necessidade;

Segundo JXTA PROGRAMMER GUIDE (2004), existem três aspectos chaves na arquitetura JXTA que se destacam entre as demais, conforme descritas:

a) o uso de documentos XML para descrever anúncios de recursos em uma rede P2P;
b) abstração de dutos para os *peers*, sendo que os mesmos não precisam ser ligados explicitamente a uma servidor central de nomes, tal como DNS;
c) um esquema uniforme de endereçamento de *peers* (*peer* ID).

3.2 CONCEITOS JXTA

Todos os conceitos descritos em tópicos anteriores sobre P2P são equivalentes para toda plataforma JXTA, sendo que os principais são *peers*, grupos de *peers*, dutos e anúncios. A seguir são apresentados alguns dos conceitos pertinentes à plataforma JXTA.

3.2.1 MENSAGENS

Mensagens são objetos enviados entre *peers* JXTA, ou seja é a unidade básica de troca de dados entre *peers*. Mensagens são enviadas e recebidas por serviços de dutos e por pontos de fim (*endpoint*). Tipicamente aplicações utilizam serviço de dutos para criar, enviar e receber mensagens. Em geral as aplicações não necessitam do uso direto de serviços de pontos de fim (*endpoint*).

Mensagem é essencialmente um conjunto chave/valor, seu conteúdo pode ser de tipos arbitrários e cada plataforma de software descreve como uma mensagem é convertida de uma estrutura de dados nativa (como exemplo objetos da linguagem Java ou C) para uma representação conhecida por toda arquitetura JXTA.

Existem duas representações para mensagens: XML e binária. A plataforma JXTA utiliza um formato binário para encapsular as mensagens, mas serviços podem utilizar um formato mais apropriado para o transporte de mensagens. Um serviço pode necessitar de uma representação mais compacta, e para isso utiliza representação binária, enquanto outro serviço

pode utilizar arquivos XML, para facilitar a comunicação com as diversas aplicações envolvidas.

3.2.2 SEGURANÇA

Redes P2P dinâmicas como JXTA necessitam um suporte para diferentes níveis de segurança de acesso aos recursos. *Peers* JXTA operam em modelos baseados em papéis, no qual diferentes *peer* se integram na rede e cada um possui níveis de privilégio diferentes, o qual podem executar diferentes tarefas. Cinco são os princípios básicos de segurança necessários:

a) sigilo: garante que conteúdos de mensagens não são abertos sem autorização;
b) autenticação: garante de que o remetente que solicita informação realmente é quem diz ser;
c) autorização: garante que o remetente é autorizado para enviar uma mensagem;
d) integridade dos dados: garante que a mensagem não foi alterada acidentalmente ou deliberadamente no caminho ao destino;
e) refutabilidade: garante de que a mensagem foi transmitida por um remetente corretamente identificado e não é reenvio de uma mensagem previamente transmitida.

3.2.3 IDENTIFICADORES (IDs)

Peers, grupos de *peers*, dutos e outros recursos JXTA precisam ser univocamente identificados. O JXTA ID é um identificador único de uma entidade. Algumas entidades que precisam ser unicamente identificadas são: *peers*, grupos de *peers*, dutos e conteúdos. URNs (*Uniform Resource Names*) são utilizados para expressar JXTA IDs, sendo que estes servem como identificadores independentes. Estes identificadores são representados em arquivo texto no formato XML reconhecido por toda plataforma JXTA, como mostrado no Quadro 1.

```
Representação de ID de peer JXTA:
 urn:jxta:uuid-59616261646162614A787461503250033F3BC76FF13C2414CBC0AB663666D

Representação de ID de duto JXTA :
 urn:jxta:uuid-59616261646162614E50472050325O338E3E786229EA460DADC1A176B69
```

Fonte: adaptado de JXTA PROGRAMMER GUIDE (2003, p. 14)

Quadro 1 – Identificadores da plataforma JXTA

Identificadores são gerados aleatoriamente pela plataforma JXTA. Existe um identificador especial chamado de *NetPeerGroup ID*, que representa o identificador do *Net peer group*, que é o primeiro grupo padrão da plataforma JXTA, sendo que todos os *peer* são criados e inicializados com ele.

3.3 PROTOCOLOS JXTA

JXTA define uma série de formatos de mensagens em XML ou protocolos, para comunicação entre *peers*. Estes protocolos são utilizados por *peers* para comunicação com outros, podendo ser utilizados nos diversos tipos de anúncios.

3.3.1 PROTOCOLO DE DESCOBERTA DE PEER (PDP)

Este protocolo é utilizado pelo *peer* para anunciar seus próprios recursos e descobri-los em outros *peers*. Cada recurso do *peer* é descrito e publicado usando anúncio. Recursos podem ser um *peer*, grupos de *peer*, dutos, serviços ou qualquer recurso que pode ser anunciado. Este protocolo é padrão para todos os *peers* definidos inicialmente por um grupo.

O atual projeto da plataforma JXTA utiliza a combinação de IP *multicasting* para uma rede local e utiliza um *peer* de encontro para procurar em outras redes. Como apresentado em seção anterior o *peer* de encontro possui um mecanismo para enviar requisições para os *peers* conhecidos. Um *peer* pode ser pré-configurado como *peer* de encontro, pode também ser o *bootstrap*, que é o primeiro *peer* de encontro a conhecer endereços de *peers* externos. Todos *peers* necessitam ter conhecimento de ao menos um *peer* de encontro, pois assim poderão ser capazes de executar todos os recursos da plataforma JXTA.

Um *peer* pode receber zero, um, ou mais respostas de seus pedidos de requisições. A mensagem de resposta pode conter um ou mais anúncios.

3.3.2 PROTOCOLO DE INFORMAÇÕES DE PEER (PIP)

Utilizado por *peers* para obterem informações de outros *peers*. Essas informações podem ser: tempo de vida das mensagens, informações do tráfego de rede, entre outras coisas. Essas informações obtidas podem ser utilizadas comercialmente ou para desenvolvimento interno de aplicações JXTA. Por exemplo, em aplicações comerciais as informações podem ser utilizadas para determinar o tempo de uso de um serviço de *peer*. Em um desenvolvimento interno, as informações podem ser utilizadas para monitorar o comportamento de um nó de

rede e quando detecta congestionamento pode reencaminhar o tráfico para outros pontos a fim de melhorar o desempenho global da rede.

3.3.3 PROTOCOLO DE RESOLUÇÃO DE PEER (PRP)

É um mecanismo responsável por enviar e receber perguntas a outros *peers*. As perguntas são direcionadas para todos os *peers*, para um grupo de *peers* ou para um *peer* específico dentro de um grupo. O PRP propaga mensagens via serviços de *peers* de encontro. Utiliza o serviço de encontro para disseminar uma mensagem para múltiplos *peers* e utiliza mensagens *unicast* para enviar pergunta para um *peer* específico.

O PRP é um protocolo que dá suporte para pedidos de questões genéricas. Tanto PIP (protocolo de informações de *peer*) como PDP (protocolo de descoberta de *peers*) são construídos utilizando este protocolo, pois necessitam de mecanismos pergunta/resposta.

Uma mensagem de resolução é utilizada para enviar um pedido de requisição para outro *peer* do mesmo grupo. A mensagem de resolução contém informações do remetente, um identificador único, um manipulador (referência de quem utiliza este serviço) e a própria mensagem. Cada serviço pode registrar um manipulador dentro de um *peer* de grupo. O manipulador é quem de fato utiliza o serviço, por isso é referenciado em todas as mensagens enviadas por ele.

3.3.4 PROTOCOLO DE LIGAÇÃO DE DUTO (PBP)

Utilizado para estabelecer um canal de comunicação virtual entre dois ou mais *peers*, fazendo a ligação começo/fim de uma mensagem. Um duto de ligação virtual pode possuir qualquer ligação de rede física de transporte, como o TCP/IP. Cada final de duto trabalha para manter a ligação virtual e restabelecê-la quando necessário.

O PBP responde uma mensagem mandando de volta ao *peer* que a solicitou. Cada mensagem que trafega por este canal virtual de comunicação leva com ele um identificador, chamado de PBP ID, isso serve para o *peer* que envia as mensagens definir informações referentes ao PBP.

3.3.5 PROTOCOLO DE ROTAS DE PONTOS DE FIM(ERP)

Este protocolo define um conjunto de mensagens de pergunta/resposta que são utilizadas para encontrar informações de rotas. Essas informações de rotas são necessárias para o envio de mensagens de um *peer* (fonte) para outro *peer* (destino). Quando um *peer* é requisitado para enviar uma mensagem para outro *peer*, primeiro ele consulta seu *cache* local para determinar se a rota deste *peer* encontra-se armazenada. Se não encontrar uma rota para o *peer*, é enviada uma pergunta de resolução de rota para o *peer relay* disponível. Quando o *peer relay* recebe a pergunta de resolução de rota, ele faz uma verificação se conhece a rota, caso conheça é retornada a informação dessa solicitação, e caso contrário o *peer relay* reenvia para outros *peer relay*.

Qualquer *peer* pode requisitar a um *peer relay* informações de rotas, e qualquer *peer* dentro de um grupo de *peers* pode se tornar um *peer relay*. *Peers relay* armazenam rotas de *peer* em seus *caches* locais. Quando um *peer relay* é dinâmico (*peer* ligado à Internet com IP dinâmico), os armazenamentos em *cache* serão constantemente alterados.

3.3.6 PROTOCOLO DE PEER DE ENCONTRO (RVP)

É um mecanismo que um *peer* pode conter para propagação de serviços. Permite um *peer* enviar mensagens para todos os *peers* que aguardam por um serviço. Este mecanismo é usado pelo protocolo de resolução de pares (PRP) e pelo protocolo de ligação de pares (PBP) para propagação de mensagens tanto em rede interna quanto em rede externa.

4 DESENVOLVIMENTO DO TRABALHO

Este trabalho teve como objetivo desenvolver um protótipo de software para compartilhar informações entre diversos computadores ligados entre si, através de rede local ou Internet. Desempenha um papel tanto de cliente como de servidor baseado no paradigma P2P. Para sua realização foram executados procedimentos de análise, elaboração e implementação do projeto.

O protótipo de software utiliza a plataforma JXTA, implementada com a linguagem de programação Java, para ser multiplataforma, mas, no entanto foi testado somente no sistema operacional Windows.

4.1 REQUISITOS PRINCIPAIS DO PROBLEMA A SER TRABALHADO

Foram levantados alguns dos principais requisitos do protótipo de software desenvolvido. Estes requisitos são parte essencial para que seja alcançado o objetivo esperado.

Os requisitos principais do protótipo de software são:

a) anunciar a existência do *peer* (RF);
b) encontrar outros *peers* localizados em qualquer ponto de uma rede local;
c) encontrar outros *peers* localizados na Internet, desde que tenham ao menos um *peer* comum que faça a comunicação inicial (RF);
d) compartilhar arquivos com todos os *peers* envolvidos na rede P2P (RF);
e) possuir ambiente visual onde são mostrados todos os *peers* e seus anúncios de arquivos compartilhados (RNF);
f) enviar mensagens para os *peers* envolvidos na rede (RF);
g) permitir a cópia de um arquivo anunciado por outro *peer* apresentado no resultado das pesquisas dos *peers* (RF);
h) computador deve possuir no mínimo 64 Mbytes de memória RAM, necessária para que seja instalada a máquina virtual Java (RNF);
i) usuário do protótipo deve ser cadastrado previamente com nome e senha (RNF);
j) possuir ambiente amigável e intuitivo de operação (RNF).

4.2 ESPECIFICAÇÃO

Para fazer a especificação deste protótipo foi utilizada uma metodologia orientada a objetos, representada em diagramas da *Unified Modeling Language* (UML), utilizando como ferramenta o *Power Designer* da SYBASE (2004).

O primeiro diagrama utilizado na especificação é o de caso de uso; na seqüência, o protótipo é representado em forma de diagrama de classes; e por último, o diagrama de seqüência para detalhar o funcionamento de protótipo.

4.2.1 DIAGRAMAS DE CASO DE USO

Caso de uso têm como propósito principal descrever de forma conceitual a estrutura do protótipo (FURLAN, 1998, p. 169).

A Figura 9 representa os principais casos de uso deste protótipo de software.

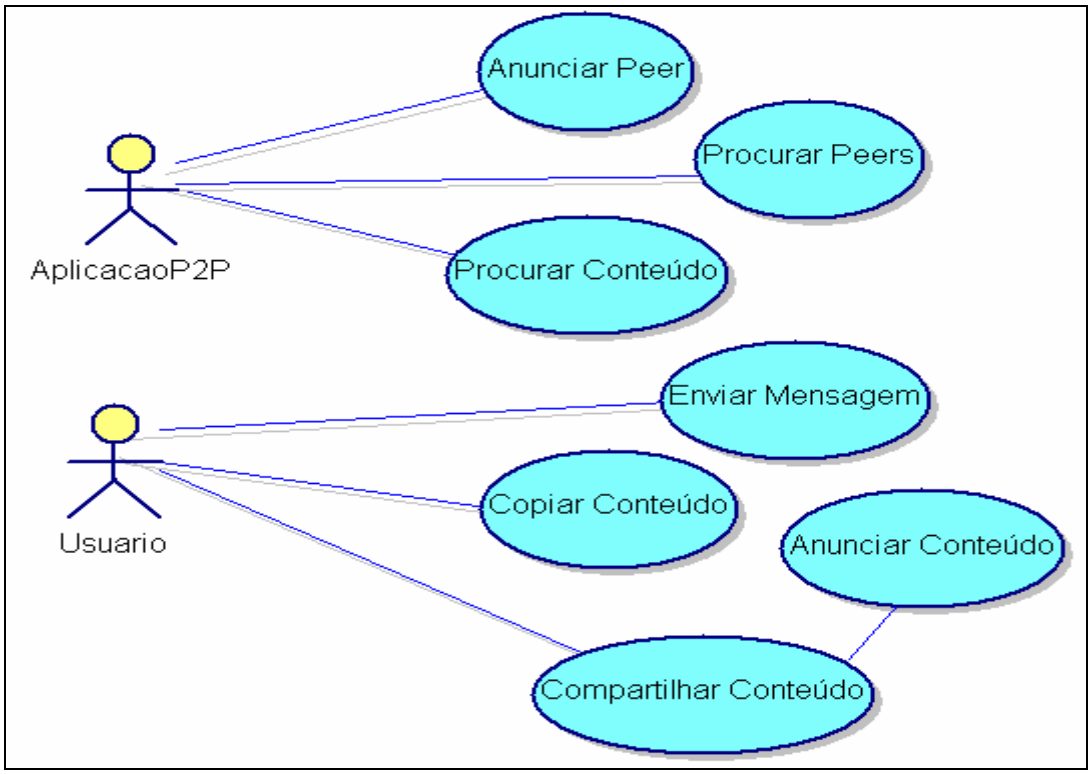


Figura 9 – Diagrama de Caso de Uso

O Quadro 2 descreve cada um dos casos de uso do protótipo, indicando o nome do caso de uso, o respectivo ator e uma descrição resumida de cada um deles.

Caso de Uso	Ator	Descrição
Anunciar *Peer*	AplicacaoP2P	Quando o protótipo é iniciado, o *peer* deve ser anunciado na rede P2P. Isso é necessário para que o mesmo possa ser encontrado por outros *peers* da mesma rede. Este processo ocorre sem interferência do usuário.
Procurar *Peers*	AplicacaoP2P	A cada 10 segundos o protótipo dispara uma rotina de procura de *peers*. A procura é realizada tanto em rede local quanto na Internet.
Procurar Conteúdo	AplicacaoP2P	O processo de procura de conteúdos ocorre a cada 10 segundos sem interferência do usuário. A procura é realizada sem vínculo direto do *peer* o qual o conteúdo pertence, ou seja, cada anúncio de conteúdo descoberto na procura é independente e tem apenas uma referência para o *peer* a qual este anúncio pertence. Cada conteúdo possui um anúncio, e é este anúncio que é procurado nas pesquisas.
Enviar Mensagem	Usuário	A mensagem é enviada por um usuário do protótipo. Cada usuário pode ser tratado com um *peer* da rede P2P. A mensagem corresponde a um texto simples que é enviada para todos os *peer* da rede.
Copiar Conteúdo	Usuário	Em cada anúncio encontrado pode ser recuperado o conteúdo e copiado para uma pasta local. Para copiar um conteúdo basta recuperá-lo do anúncio.
Compartilhar Conteúdo	Usuário	Existe uma pasta local compartilhada com outros *peers*. Quando se necessita compartilhar algum arquivo este deve ser adicionado nesta pasta. Podem ser compartilhados somente arquivos e não diretórios.
Anunciar Conteúdo	Usuário	A cada 10 segundos é verificada, na área compartilhada, a existência de novos arquivos. Caso seja encontrado, é gerado um anuncio que é publicado na rede P2P.

Quadro 2 – Descrição dos Casos de Uso

4.2.2 DIAGRAMA DE CLASSES

Segundo Furlan (1998, p. 91), o diagrama de classes é utilizado para representar a estrutura lógica de um sistema detalhando cada elemento como: classes, tipos, atributos e métodos. A Figura 10 representa a estrutura básica do protótipo, onde são mostradas somente as classes com seus relacionamentos e multiplicidades.

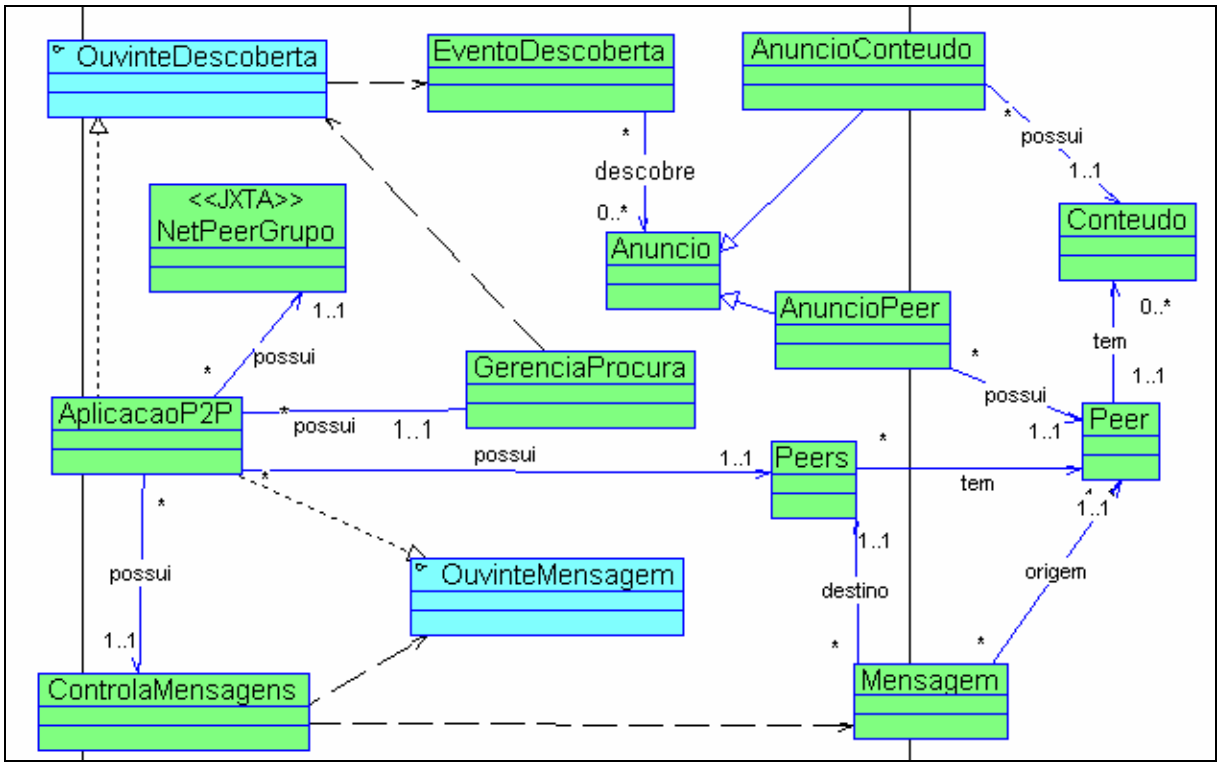


Figura 10 – Diagrama de Classes

O protótipo é composto de classes que fazem chamadas a bibliotecas da plataforma JXTA. Nas classes representadas na Figura 10 não foram apresentados detalhes como métodos e propriedades, para simplificar a figura tornando-a mais legível. A seguir está um resumo do funcionamento de cada classe:

a) AplicaçãoP2P: responsável pela execução do protótipo. Faz a chamada das principais classes envolvidas, implementa funcionalidades de um ouvinte de descoberta e a cada descoberta de *peer* ou conteúdo faz a chamada de um processo que faz o tratamento necessário;

b) *NetPeerGroup*: uma classe da plataforma JXTA utilizada para representar um grupo de *peers*. Todos os *peers* envolvidos na rede P2P estão ligados a este grupo, pois o mesmo é quem faz a autenticação necessária para um *peer* juntar-se à rede. Ao criar uma instância dessa classe, a plataforma JXTA apresenta uma janela de configuração das informações mínimas que um *peer* precisa informar para juntar-se ao grupo;

c) OuvinteDescoberta: é um ouvinte de todas as descobertas realizadas na rede P2P. Quando um *peer* ou conteúdo são descobertos pela pesquisa realizada a cada 10

segundos, a aplicação representada pela classe AplicacaoP2P recebe uma mensagem. Através dessa mensagem consegue-se recuperar o *peer* ou o anúncio de conteúdo descoberto;

d) EventoDescoberta: a cada descoberta realizada é disparada uma mensagem, que tem como parâmetro uma classe chamada EventoDescoberta que possui uma lista de todos anúncios descobertos na pesquisa;
e) GerenciaProcura: quando a aplicação P2P é iniciada, inicia-se também uma instância dessa classe, que é responsável pela gerência de todas as procuras realizadas. Essa classe faz referencia para a biblioteca de procura da plataforma JXTA, e tem como função disparar as operações de procura de *peers* e de conteúdos. A cada descoberta de um novo conteúdo é enviada uma mensagem para todas as classes que registraram uma instância de OuvinteDescoberta;
f) Anuncio: é a classe base dos anúncios de *peers* e de conteúdos. Cada classe EventoDescoberta pode possuir uma lista desses anúncios;
g) AnuncioPeer: cada *peer* encontrado nas pesquisas é representado por esta classe a qual possui ligação direta com a classe Peer;
h) AnuncioConteudo: representa o anúncio de um conteúdo. Tem uma ligação com um conteúdo;
i) *Peer*: representa um *peer* da rede P2P. Cada *peer* corresponde a uma aplicação JXTA ativa, tanto em rede local quanto na Internet;
j) Conteudo: é o conteúdo que pode ser copiado para uma pasta local. Conteúdo é o mesmo que arquivo, e possui nome, tamanho e um tipo;
k) *Peers*: é uma classe que contém todos os *peers* encontrados na rede P2P. Responsável pela atualização da lista de *peer* encontrados, tem operações para adicionar e remover *peers* da lista;
l) OuvinteMensagem: é ouvinte das mensagens enviadas para rede P2P, pois cada mensagem sempre é enviada para todos os *peers* da rede. A classe AplicacaoP2P implementa as operações dessa interface e a cada evento de mensagem é disparado um método específico. Os métodos são: de saída de *peer*, entrada de *peer*, ou uma mensagem escrita por um usuário da rede P2P;
m) Mensagem: uma mensagem de texto para comunicação entre os *peers* da rede. Cada mensagem tem origem em um *peer* específico e destino todos os *peers* da rede;

n) ControlaMensagens: essa classe faz o controle de todas as mensagens enviadas entre os *peers* da rede. Faz referência à biblioteca da plataforma JXTA. Ao receber uma mensagem é disparado um método correspondente na interface OuvinteMensagem. Conseqüentemente, todas as classes que a implementam recebem mensagens. Neste protótipo a classe que recebe as mensagens é a AplicacaoP2P.

4.2.3 DIAGRAMAS DE SEQUÊNCIA

Os digramas de seqüência representam a seqüência das ações ocorridas em um conjunto de classes, demonstram como ocorre a troca de mensagens entre as classes. Para cada caso especificado de uso, há um diagrama de seqüência, conforme detalhamento a seguir.

4.2.3.1 ANUNCIAR PEER

Este diagrama de seqüência representado na Figura 11 mostra como ocorre um anúncio de *peer*, ocorrido a cada vez que o protótipo e iniciado. Cada vez que a aplicação é iniciada é realizado um anúncio remoto do *peer* com a duração de 2 horas. A tempo duração do anúncio pode ser alterado por parâmetros passados para a plataforma JXTA.

Quando o protótipo está executando mais que 2 horas, ou seja, mais que o tempo do anúncio do *peer*, a própria plataforma JXTA faz um novo anúncio remoto. Todo anúncio também sempre é publicado em *cache* local.

O processo de anúncio ocorre sem intervenção do usuário. Ao iniciar o protótipo é criada uma instância da classe AplicacaoP2P que se encarrega do processo de anúncio.

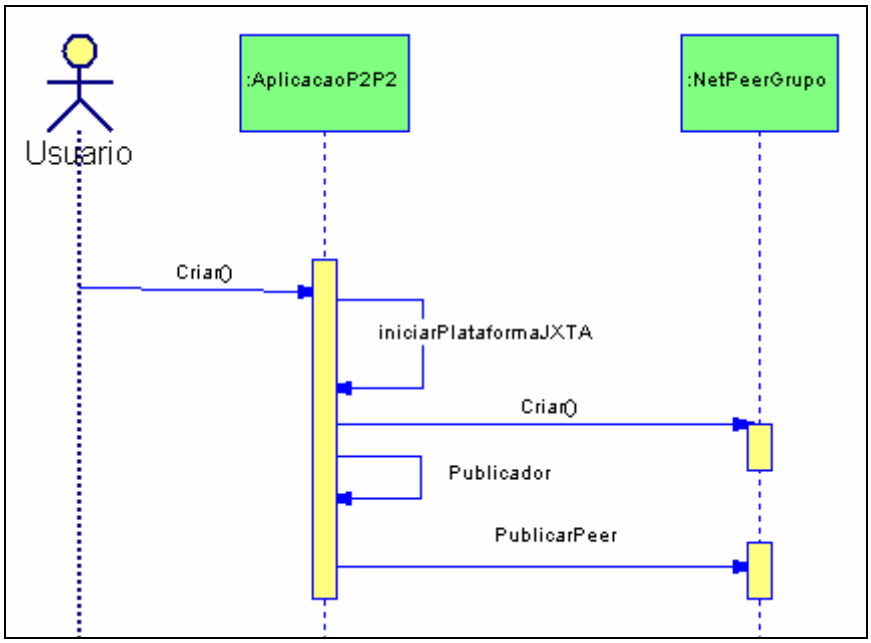


Figura 11 – Diagrama de seqüência: “Anunciar Peer”

4.2.3.2 PROCURAR PEERS

Este diagrama representado na Figura 12 mostra como um *peer* procura outros. A cada 10 segundos é realizada uma busca na rede P2P. Caso encontre novos *peers*, os mesmo são adicionados em uma lista. Para o processo de procura de *peers* é chamado um serviço de descoberta da plataforma JXTA. As pesquisas são realizadas em até cinco níveis de *peers* de encontro, que são *peers* responsáveis por propagar anúncios e mensagens e contém informações de outros *peers*. As Figuras 4 e Figura 5 representam descobertas diretas e indiretas de *peers* por *peers* de encontro.

A Figura 12 representa desde o processo de registro do ouvinte de descobertas até quando o anúncio de um *peer* é encontrado e recuperado. A cada requisição de descoberta enviada à rede P2P, pode haver ou não uma resposta. Todas as requisições de procura são enviadas por um duto (*pipe*) fornecido pela plataforma JXTA, e a cada anúncio de *peer* encontrado é disparado um método da interface OuvinteDescoberta que reflete na AplicacaoP2P, pois a mesma tem um registro de todas as descobertas.

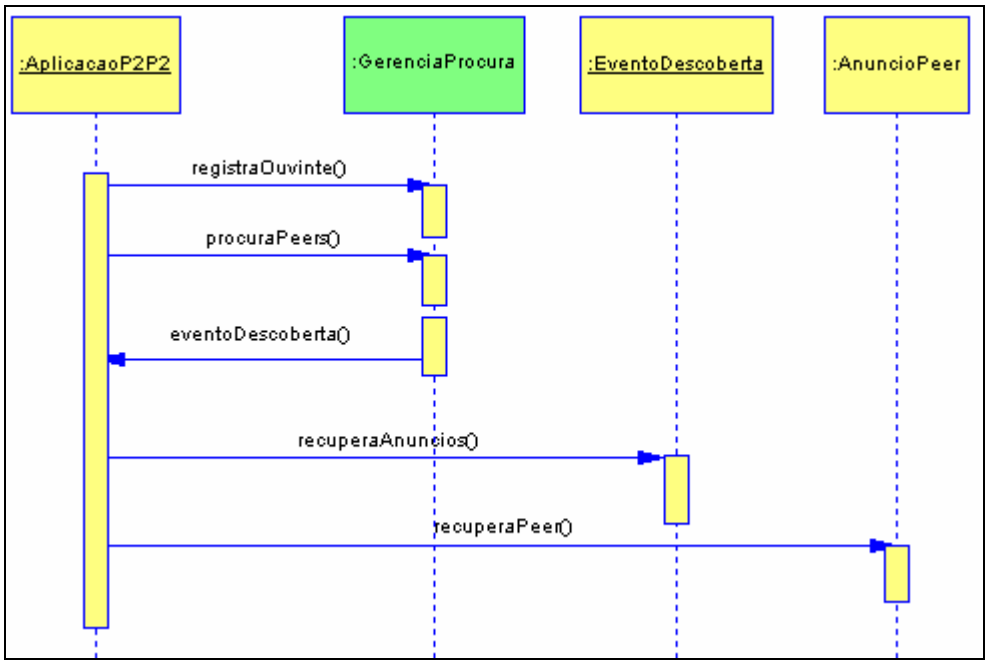


Figura 12 – Diagrama de seqüência: “Procurar Peers”

4.2.3.3 PROCURAR CONTEÚDO

O diagrama de seqüência mostrado na Figura 13 representa a procura de conteúdos na rede P2P. A cada 10 segundos é disparado um método da plataforma JXTA que efetua a procura de conteúdos. Quando qualquer anúncio de conteúdo é encontrado, o mesmo é adicionado na instância da classe Peer que anunciou o conteúdo.

Como acontece na procura de *peers*, conteúdos também são procurados em até cinco níveis de *peers* de encontro, como mostradas na Figura 4 e Figura 5, que representam descobertas diretas e indiretas de *peers*. A diferença que conteúdos são anunciados por *peers* e possuem referência aos *peers* que os anunciam.

O processo de procura de conteúdo é semelhante ao da procura de *peers*, envolvendo as classes GerenciaProcura e EventoDescoberta, e cada descoberta de anúncio de conteúdo é refletido na classe AplicacaoP2P, pois a mesma registra-se com ouvinte de descobertas. Tanto as requisições de descobertas como os eventos de anúncio são conduzidos por meio de dutos (*pipes*) representado pela plataforma JXTA.

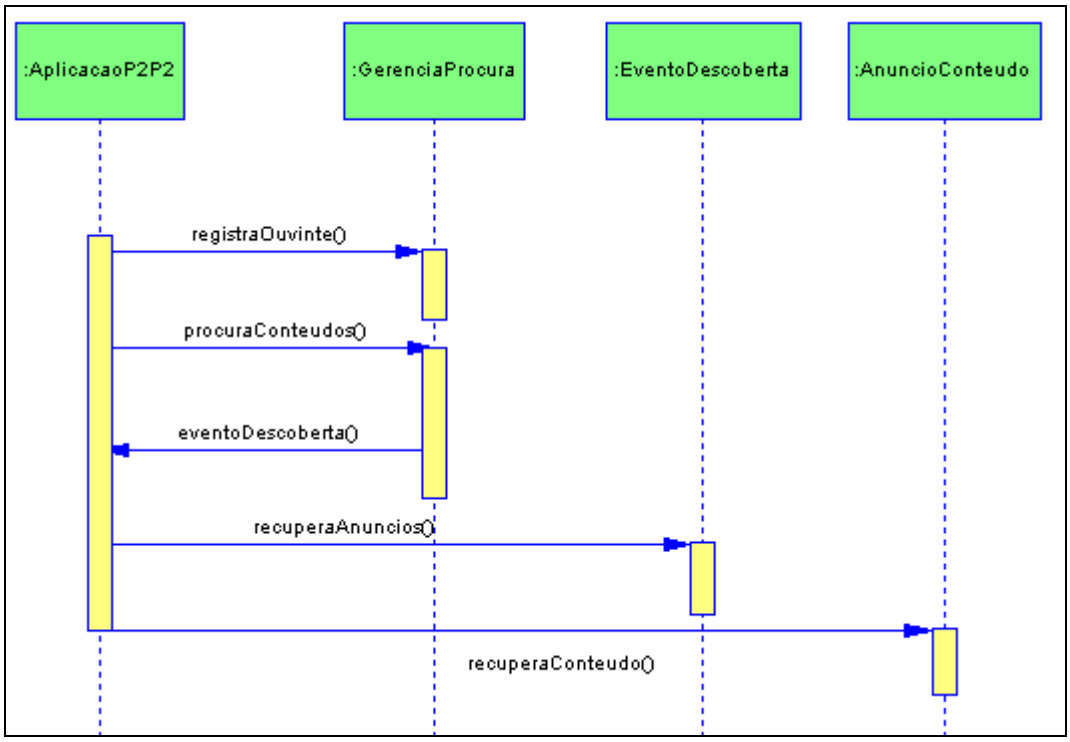


Figura 13 – Diagrama de seqüência: “Procurar Conteúdo”

4.2.3.4 ENVIAR MENSAGEM

O diagrama de seqüência mostrado na Figura 14 representa o processo de envio de mensagem de um *peer* para outros. As mensagens enviadas pelos *peers* são sempre direcionadas para toda rede P2P, o que foi implementado no protótipo devido à complexidade de gerenciamento das mensagens entre os *peers*. Para envio de mensagens deve ser utilizado um tipo especial de *pipe,* chamado *output pipe* que é fornecido pela plataforma JXTA.

A classe AplicacaoP2P é registrada como ouvinte de todas as mensagens enviadas pelos *peers*, e a classe ControlaMensagens recupera todos os *peers* atualmente ativos na rede P2P para que possa enviar mensagens para os mesmos. Enviar uma nova mensagem requer a criação de uma instância da classe Mensagem e quando a mensagem está devidamente redigida pelo usuário, a mesma é enviada para o *pipe* por uma instância da classe ControlaMensagens.

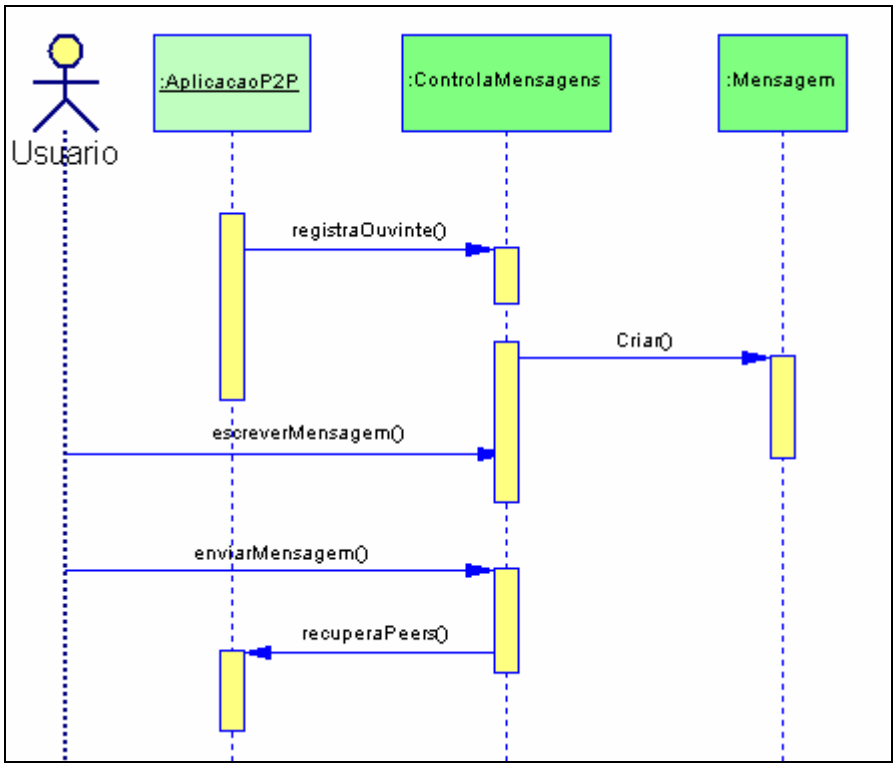


Figura 14 – Diagrama de seqüência: “Enviar Mensagem”

4.2.3.5 COPIAR CONTEÚDO

O diagrama de seqüência da Figura 15 representa a maneira que um usuário faz a cópia de um conteúdo remoto para uma pasta local.

Inicialmente o usuário faz a seleção de um *peer*, no qual também efetua a seleção de um anúncio de conteúdo daquele *peer*. Cada anúncio de conteúdo tem uma referência indireta para o seu conteúdo. Por padrão o conteúdo será copiado para o mesmo diretório da aplicação P2P, mas tem opção para direcionar para outro diretório do computador local.

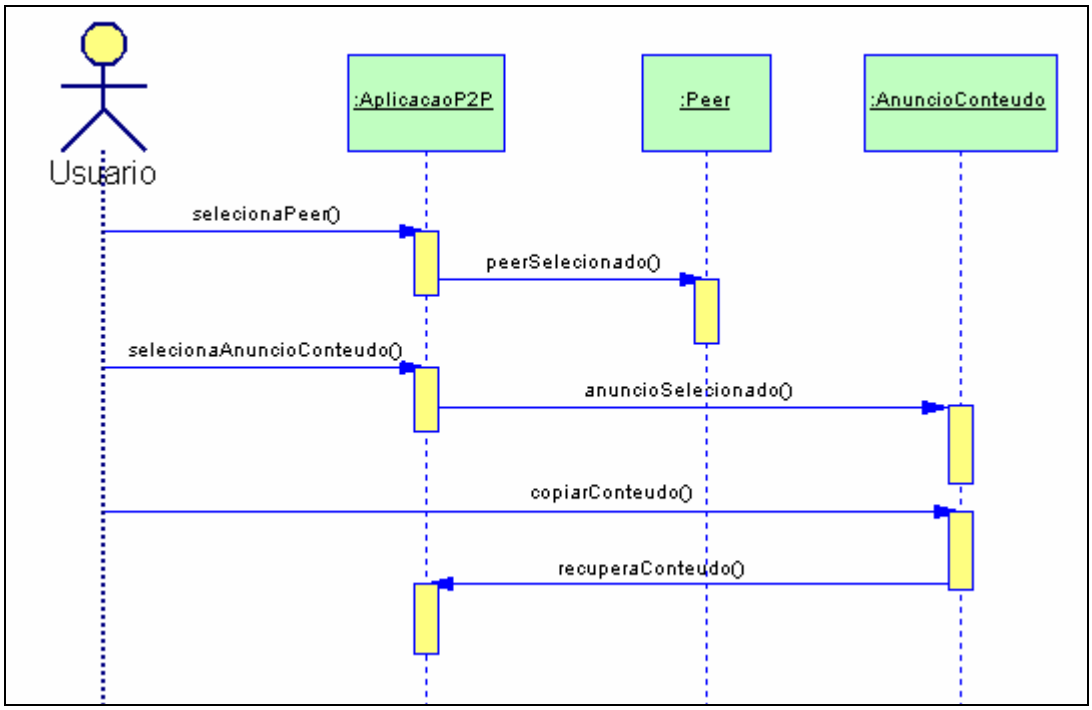


Figura 15 – Diagrama de seqüência: “Copiar Conteúdo”

4.2.3.6 COMPARTILHAR CONTEÚDO

O diagrama de seqüência da Figura 16 representa a maneira que conteúdos podem ser compartilhados.

Para compartilhar conteúdo com outros *peers*, o usuário precisa adicionar o mesmo na pasta compartilhada do protótipo. O processo de anúncio está descrito no diagrama de seqüência: Compartilhar Conteúdo.

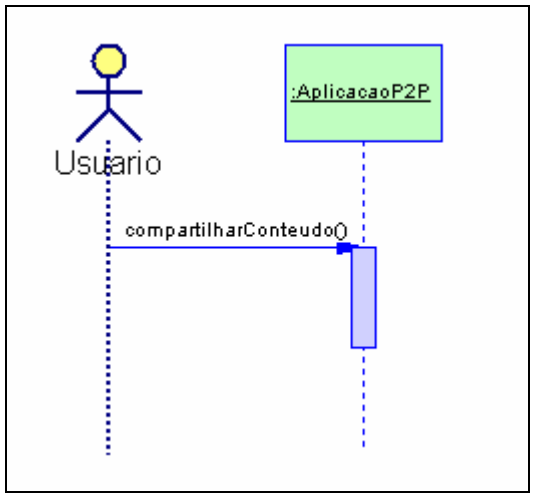


Figura 16 – Diagrama de seqüência: “Compartilhar Conteúdo”

4.2.3.7 ANUNCIAR CONTEÚDO

O diagrama de seqüência da Figura 17 representa como ocorre o processo de anúncio de conteúdo adicionado na pasta compartilhada.

A cada 10 segundos é realizada uma leitura da pasta local compartilhada, e caso exista novos conteúdos adicionados, os mesmo são compartilhados através da plataforma JXTA.

Para cada conteúdo é criado um anúncio correspondente. Junto a esse anúncio é adicionada uma referência do *peer* que o representa.

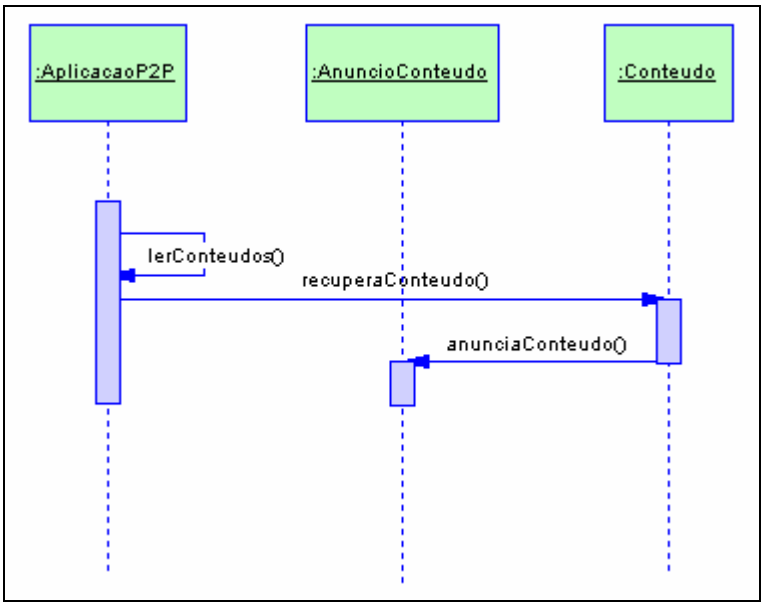


Figura 17 – Diagrama de seqüência: “Anunciar Conteúdo”

4.3 IMPLEMENTAÇÃO

Neste capítulo serão apresentados tópicos referentes à implementaçãodo modelo proposto neste trabalho.

4.3.1 TÉCNICAS E FERRAMENTAS UTILIZADAS

Nos itens a seguir serão apresentadas algumas das principais técnicas e ferramentas utilizadas no desenvolvimento deste trabalho. Em destaque alguns dos pontos relevantes da implementação com a plataforma JXTA e como a mesma representa as principais funcionalidades de redes P2P.

4.3.1.1 PLATAFORMA ECLIPSE

Eclipse é uma ferramenta para desenvolvimento de programas em linguagens de programação como o C++ e Java, possui diversas extensões para facilitar a implementação de códigos. É mantido atualmente por um projeto chamado de Eclipse (2004) que detem todos direitos reservados para IBM Corp.

4.3.1.2 PLATAFORMA E BIBLIOTECAS JXTA

A plataforma JXTA é composta de um conjunto de bibliotecas que oferece recursos para implementação de redes P2P. Desenvolvidas na linguagem de programação Java e compactadas em pacotes conhecidos como Java arquivos (JAR). Estes arquivos contêm classes, arquivos fonte, arquivos de imagem ou qualquer arquivo utilizado no sistema.

Alem das bibliotecas utilizadas especificamente para a plataforma JXTA existem outras desenvolvidas para extensão de alguns dos principais recursos. As principais são de: segurança, sistema de administração de conteúdo, leitura dos protocolos em formato XML e extensão da configuração básica. Conforma apresentado a seguir:

a) básica: as principais bibliotecas utilizadas na plataforma JXTA são: ‘jxta.jar’, ‘jxtaptls.jar’ e ‘jxtasecurity.jar’ utilizadas na implementação de qualquer aplicação P2P;
b) segurança: existe três bibliotecas utilizadas na segurança em aplicações JXTA. O arquivo Java chamado de ‘bcprov-jdk14.jar’ provê implementações criptográficas de comunicação. Os arquivos chamados de ‘cryptix32.jar’ e ‘cryptix-asn1.jar’ desenvolvidas provêem implementações de código aberto para bibliotecas criptográficas;
c) sistema de administração de conteúdo: é uma biblioteca de extensão criada pelo projeto JXTA para controle de compartilhamento remoto de conteúdos. Está no arquivo Java chamado ‘jxtacms.jar’;
d) leitura do protocolos em formato XML: é um conjunto de bibliotecas utilizada para manipulação de arquivos com formato XML, utilizada pelos seis protocolos da plataforma JXTA. É possível criar, excluir ou alterar qualquer arquivo XML utilizado na plataforma JXTA. As bibliotecas estão implementadas nos arquivos ‘jaxen-core.jar’, ‘jaxen-jdom.jar’, ‘jdom.jar’ e ‘saxpath.jar’;
e) extensão de configuração básica: existe uma configuração básica necessária para todos os *peer* participantes da rede P2P implementadas com a plataforma JXTA. Na criação do *netPeerGroup*, um grupo obrigatório para todos os *peers*, á aberta uma janela contendo uma série de campos que necessitam ser preenchidos, essa janela estão com todas as informações escritas na língua inglesa. Para isso existem duas bibliotecas ‘ jxtaext.jar’ e ‘jxtaSwing.jar’ que oferecem ao desenvolvedor o recurso

necessário para criar sua própria janela de configuração, fazendo com que o usuário necessite preencher somente dados básicos.

4.3.1.3 ESTRUTURA DE ARQUIVOS E DIRETÓRIOS DA PLATAFORMA JXTA

Quando uma aplicação JXTA for iniciada pela primeira vez é criada uma estrutura básica de arquivos e diretórios na máquina local.

A estrutura é criada a partir do diretório raiz onde a aplicação está sendo executada. O diretório onde estão armazenados todos os outros subdiretórios é chamado de '.jxta'.

Para armazenar as informações básicas de cada *peer* a plataforma JXTA cria um arquivo de formato XML chamado de '.jxta/PlatformConfig' onde estão armazenadas alem do *peer* ID (identificador único do *peer*), seu nome e outras informações como:

a) configuração de IP do *peer*: um *peer* sempre é identificado por um IP e pelo menos uma porta. A porta para Internet pode ser diferente para rede local, por padrão a porta da Internet é 9700 e para rede local é 9701;
b) *peers* de encontro e *peers relay*: cada *peer* obrigatoriamente deve conter pelo menos um *peer* de encontro ou um *peer relay*, necessários para iniciar a comunicação tanto em rede local quanto na Internet;

As informações de segurança de cada *peer*, como nome, senha e certificados de segurança são armazenados em subdiretórios de '.jxta/pse'. Opcionalmente cada *peer* é autenticado, mas sempre que aplicação JXTA foi iniciada pela primeira vez essa senha deve ser cadastrada, mesmo que não utilizado nas próximas execuções.

Um diretório chamado de '.jxta/cm' é criado para armazenar anúncios de todos os grupos e *peer* encontrados. Está dividido em dois subdiretórios '.jxta/cm/jxta-NetGroup' e '.jxta/cm/jxta-WorldGroup', o primeiro armazena anúncios do *NetPeerGroup*, grupo comum a todos os *peers* encontrados, o segundo armazena anúncios de todos os grupos encontrados na rede P2P. Cada um desses subdiretórios contém arquivos que representam anúncios de *peers*, grupos de *peers*, dutos (*pipes*) além de armazenar arquivos indexados pela plataforma JXTA.

Um arquivo chamado de '.jxta\jxta.properties' é criado pela plataforma JXTA e tem a finalidade de armazenar algumas informações básicas como o número mínimo e o máximo de *Threads* que o servidor HTTP (*Hypertext Transfer Protocol*) do *peer* deve possuir.

Toda estrutura criada pela plataforma JXTA também é mantida pela mesma, ficando a cargo do desenvolvedor somente anúncios referentes há a arquivos locais. A Figura 18 mostra a estrutura de arquivos e diretórios criados pela plataforma JXTA.

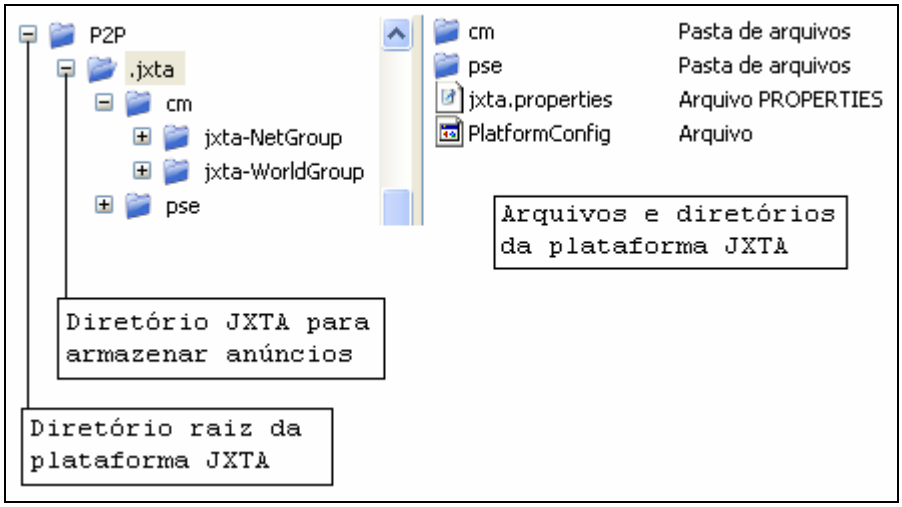


Figura 18 – Estrutura de arquivos e diretórios da plataforma JXTA

4.3.1.4 INICIALIZAÇÃO DO PROTÓTIPO

As configurações mínimas necessárias para inicializar o protótipo são criados a partir das bibliotecas de extensão 'jxtaext.jar' e 'jxtaSwing.jar'. A classe principal dessa biblioteca é chamada de *net.jxta.ext.config.Configurator* e possui métodos para configuração básica, como apresentados no Quadro 3.

```
//Configuração básica
Configurator config = new Configurator();
config.setName("Nome_Peer");
config.setSecurity("Nome_Usuario","Senha_Usuário");
config.addRendezVous("http://200.25.0.18:9701");
config.addRelay("tcp://200.25.0.18:9700");
config.save();
```

Quadro 3 – Configuração básica inicial do protótipo.

Essa configuração é apresentada ao usuário através de uma janela como mostrado na Figura 19. O protótipo já oferece as configurações básicas, não havendo necessidade de preencher cada um dos campos. A seguir a descrição dos principais campos da Figura 19:

a) configuração local: tem-se a opção de ativar ou desativar o *peer* para uma rede local, caso seja desativado, o *peer* não pode ser encontrado por outros *peers*. Cada *peer* deve ser identificado por um IP (protocolo de Internet) e uma porta. O protocolo TCP (*Transmission Control Protocol*) sempre é utilizado pela plataforma JXTA para redes locais;

b) configuração Internet: tem-se a opção de ativar ou desativar o *peer* para a Internet, caso desativado, o *peer* não pode ser encontrado por outros *peers*. Este *peer* por padrão tem uma porta diferente da utilizada na rede local. O protocolo HTTP (*Hypertext Transfer Protocol*) é utilizado pela plataforma JXTA para comunicação com outros *peers* na Internet;

c) segurança: a plataforma JXTA exige que cada *peer* seja autenticado com nome e senha para entrar no *netPeerGroup*. Mesmo que não utilizado a autenticação, sempre que a aplicação é iniciada aparece uma janela da plataforma JXTA exigindo que nome e senha sejam informados.

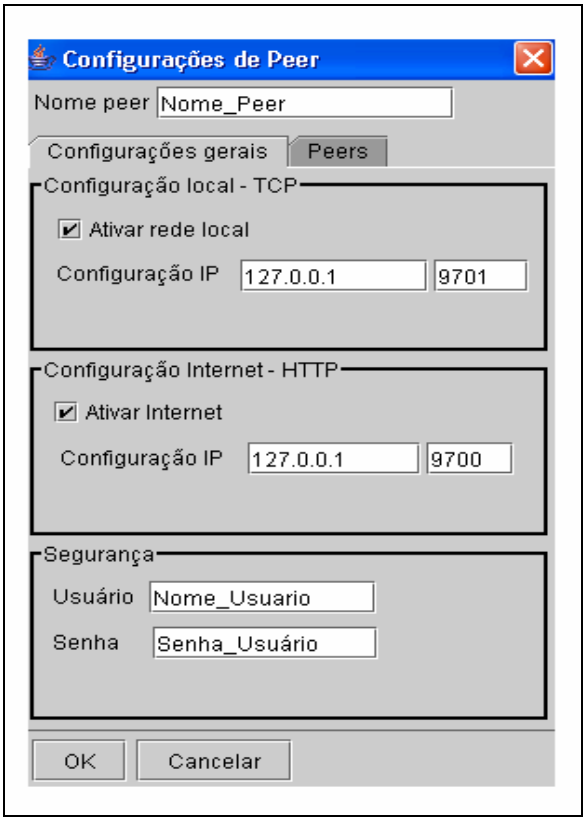


Figura 19 – Configuração mínima exigida na plataforma JXTA – parte 1

A plataforma JXTA exige também que sejam informadas configurações de *peers* de encontro e *peers relay*. Para isso devem ser configuradas informações de *peers* conforme apresentados na Figura 20. A seguir a descrição dos campos da Figura 20:

a) *peer* de encontro: para que um *peer* possa entrar na rede P2P o mesmo deve ter ao menos um *peer* de encontro conhecido. Sendo que para rede local a porta é 9701 e para Internet 9700. A plataforma JXTA oferece o recurso de permitir que um *peer* seja configurado como *peer* de encontro automaticamente;
b) *peer relay*: como acontece com o *peer* de encontro, a plataforma JXTA exige que seja cadastrado ao menos um *peer relay* (*peer* que recupera informações de rotas de outros *peers* e é capaz de atravessar firewalls). Para permitir que o *peer* seja configurado como *peer relay* o campo 'Atuar como *peer relay'* deve ser checado;

Depois de complementado o preenchimento dos dados dessa janela, será apresentada outra janela ativada pela própria plataforma JXTA. Portanto não pode ser omitida na inicialização desse protótipo.

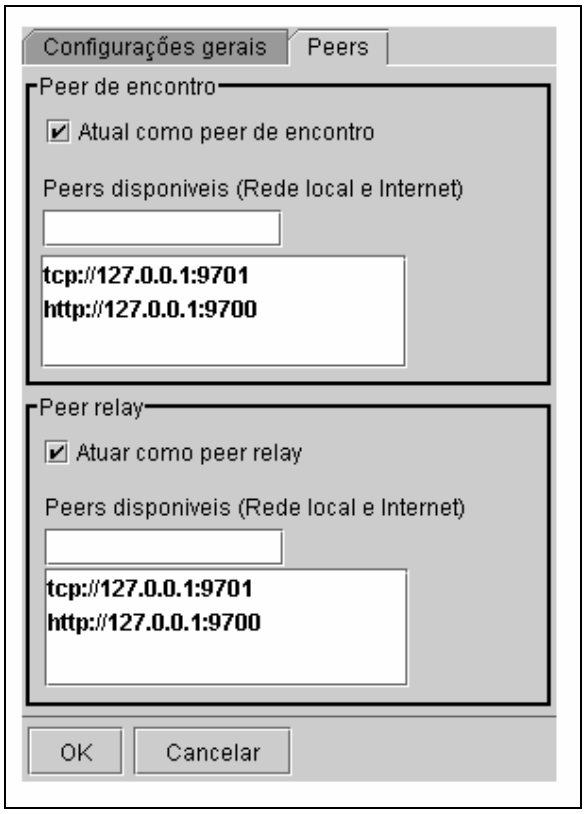


Figura 20 – Configuração mínima exigida na plataforma JXTA – parte 2

4.3.1.5 PRINCIPAIS ROTINAS DO PROTÓTIPO

Nesta seção serão apresentados alguns dos principais processos desenvolvidos para o protótipo.

O digrama de classes do protótipo representado na Figura 10 mostra somente as classes do protótipo e não as classes da plataforma JXTA. A representação dessas classes foi omitida devido à quantidade. Cada classe do protótipo encapsula as principais funcionalidades oferecidas pela plataforma JXTA.

Alguns dos códigos fonte apresentados nessa seção fazem menção a classes específicas da plataforma JXTA, quando isso acontece será acompanhada de uma breve descrição.

4.3.1.5.1 LOCALIZAÇÃO DE ANÚNCIOS DE PEERS E CONTEÚDOS

Existe uma *Thread (*sub-programa*)* verificando a existência de anúncios a cada 10 segundos, mas antes disso a aplicação deve ser registrada para receber mensagens que contém informações de descoberta.

A interface utilizada para registro chama-se OuvinteDescoberta que contém um método eventoDescoberta, chamado cada vez que um ou mais anúncios são encontrados. O Quadro 4 mostra o código Java executado a cada descoberta de anúncio.

Inicialmente é recuperado do EventoDescoberta uma lista de anúncios, após isso é realizado um *loop* para todos estes anúncios verificando se é AnuncioPeer ou AnuncioConteudo. Para cada tipo de anúncio é chamado um método que faz o tratamento específico como: atualizar a lista de *peers* e conteúdo.

```
public void eventoDescoberto(EventoDescoberta evento) {
   List lista = evento.recuperaAnuncios();
   Anuncio anuncio;
   for (int i = 0; i < lista.size(); i++) {
      anuncio = (Anuncio)lista.get(i);
      //
      // Anúncio de Peers encontrados
      //
      if (anuncio instanceof AnuncioPeer) {
         AnuncioPeer anPeer = (AnuncioPeer)anuncio;

         // peers é instância da classe Peers, e contém a lista de todos o peers
         boolean novoPeer = peers.atualizaPeer(anPeer.getNome(), anPeer);
         if (novoPeer) {
            areaMensagem.append("Peer encontrado: "+anPeer.getNome());

            //Método que trata o novo peer encontrado
           tratamentoPeer(anPeer);
         }
      }
      //
      // Anúncio de arquivos descobertos
      //
      else {
         AnuncioConteudo anConte = (AnuncioConteudo)anuncio;

         boolean novoArq = peers.atualizaAnuncioConteudo(anConte.getNome(),anConte.getPeer());
         if (novoArq){
             areaMensagem.append("Arquivo encontrado : "+anPeer.getNome());

            //Método que trata o novo peer encontrado
            tratamentoAnuncioConteudo(anConte);
         }
      }
   } // fim do for
} // fim do método
```

Quadro 4 – Evento de descoberta de anúncios de peers e conteúdos.

4.3.1.5.2 ENVIO DE MENSAGENS

O envio de mensagens sempre á realizado pela classe ControlaMensagens que encapsula funcionalidades de escrita de mensagem. Para enviar mensagem para rede P2P é necessário criar uma instância da classe *OutputPipe* adicionando elementos do tipo *StringMessageElement*.

O Quadro 5 representa a maneira de enviar mensagem para o *output pipe.*

```
public void enviarMensagem(String mensagem) {

    Mensagem msg = new Mensagem();

    // Nome do peer que envia a mensagem
    msg.addMessageElement(null,new StringMessageElement("NomePeer", nomePeer, null));

    //Mensagem adicionada no outputPipe
    msg.addMessageElement(null,new StringMessageElement("MensagemPeer", mensagem, null));

    //
    // outputPipe é instância da classe OutputPipe da plataforma JXTA
    //
    outputPipe.send(msg);
}
```

Quadro 5 – Envio de mensagens para rede P2P

4.3.1.5.3 ANÚNCIO DE CONTEÚDOS

O anúncio de conteúdo é realizado pela plataforma JXTA utilizando a classe *net.jxta.share.ContentManager*. Para isso foi criado uma variável de instância responsável pelo compartilhamento de cada um dos conteúdos. O Quadro 6 apresenta de forma compacta o código responsável pela publicação dos conteúdos.

Para inicialização da classe *ContentManager* é utilizada uma instância da classe *net.jxta.share.CMS*. A classe CMS (*Content Management Service*) faz a inicialização dos anúncios criando uma pasta no diretório raiz da aplicação chamada de anuncioArquivos. Nesta pasta há um arquivo chamado de shares.ser que contém uma referência para cada anúncio de conteúdo realizados pelo *peer*.

```
private void lerConteudoLocais() {
             //
             // Recupera a lista de arquivos locais
             //
             File[] files = diretorioLocal.listFiles(new FilenameFilter() {
                public boolean accept(File dir, String name) {
                    if (hashArquivosLocais.get(name) == null) {
                         return true;
                    }
                    return false;
                }
             });

              //
              //Publicação de arquivos
              //
              for (int i = 0; i < files.length; i++) {
                  File file = files[i];
                  if (file.isFile()) {
                       //
                       // contentManager é instância da classe
                       // net.jxta.share.ContentManager
                       //
                       contentManager.share(file, manager.getMyPeerName());
                  }
              }

}
```

Quadro 6 – Compartilhamento de conteúdos locais

4.3.2 OPERACIONALIDADE DA IMPLEMENTAÇÃO

Para utilização desse protótipo é necessário que haja dois ou mais computadores ligados em rede local ou a Internet. Para o funcionamento do protótipo em redes locais não há necessidade de acrescentar nenhuma informação adicional. O mesmo não acontece na execução dos protótipos em computadores ligados a Internet, pois é necessário que seja informado pelo menos o endereço de IP de um *peer* de encontro e um *peer relay*.

A Figura 21 apresenta a tela principal do protótipo, sendo que na mesma possui todos recursos visuais necessários para demonstração de uma rede P2P. A seguir será descrita cada uma das partes do protótipo:

a) lista de *peers*: na parte esquerda da tela estão todos os *peers* encontrados na rede P2P. Sendo que o primeiro *peer* é o *peer* local. O restante da lista são *peers* remotos. *Peers* remotos podem ser considerados tanto os *peers* da mesma rede local, quanto os *peers* localizados na Internet;

b) lista de anúncios de conteúdos: na parte direita da tela estão todos os anúncios de conteúdos encontrados. Conteúdos são sempre arquivos com um nome e tamanho em Kbytes. Quando o *peer* local está selecionado (sempre o primeiro da lista) é listado os anúncios feitos pelo mesmo. Ao selecionar qualquer outro *peer* são listados todos anúncios publicados pelo mesmo;

c) diretório compartilhado: na parte superior do protótipo está descrito o caminho do diretório compartilhado. Cada arquivo adicionado neste diretório é anunciado na rede P2P;

d) informações de *peers* e arquivos: na parte inferior da tela encontra-se uma área contendo diversas informações dos *peers* participantes da rede, e cada um de seus anúncios publicados;

e) envio de mensagens: ainda na parte inferior encontra-se um campo onde são escritas as mensagens que são enviadas para todos os *peers* da rede. Para o envio de mensagens deve ser pressionada a tecla Enter.

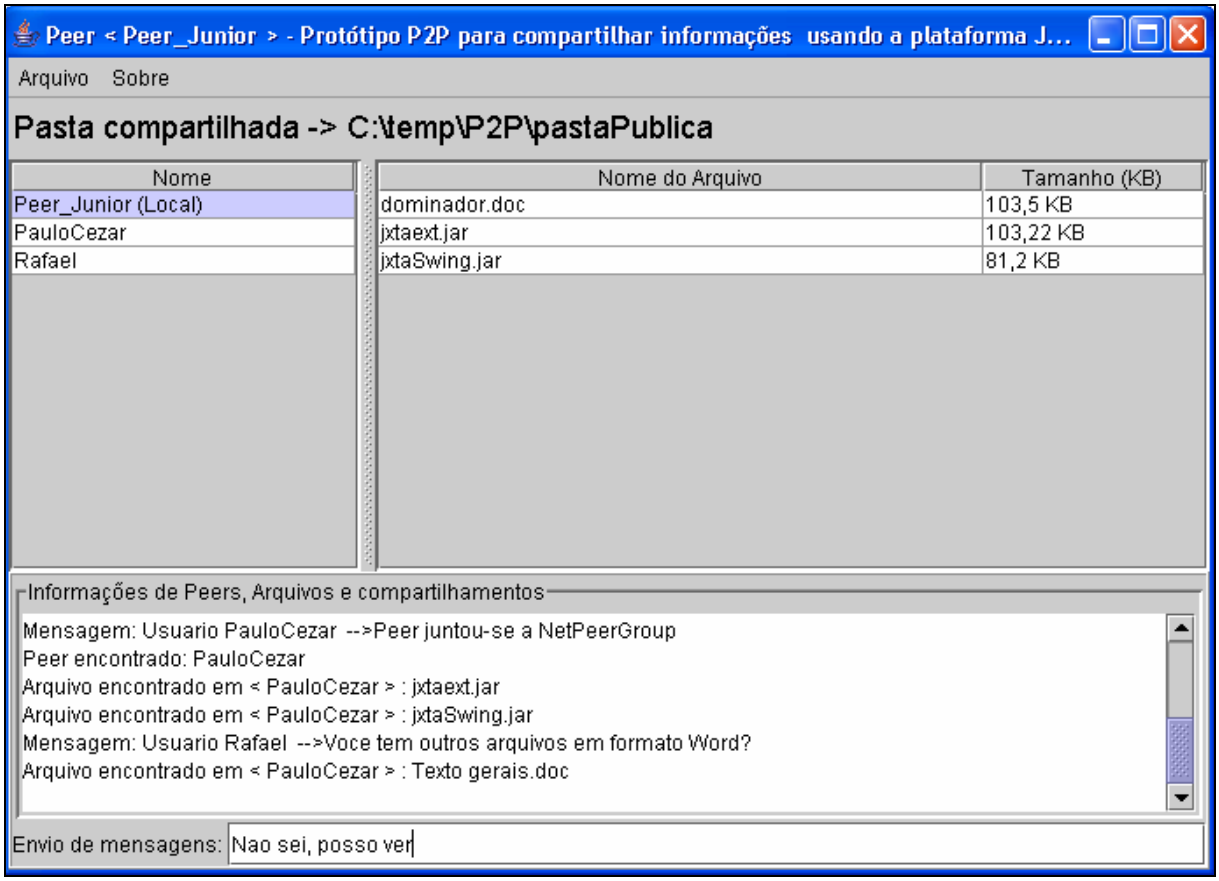


Figura 21 – Tela principal do protótipo P2P

Cada operação ou evento que acontece na rede P2P gera uma linha de texto na área informações de *peers* e arquivos como mostrados na Figura 22. A seguir são apresentadas algumas das informações presentes nessa área.

a) *peer logado* e juntou-se ao *netPeerGroup*: são mensagens que indicam que o *peer* conseguiu criar um serviço de descoberta e escrever uma mensagem no *output pipe* através de uma instância do *netPeerGroup*, criado na inicialização do protótipo;
b) *peer* encontrado: cada anúncio de *peer* encontrado na rede P2P gera uma linha que contém seu nome;
c) *peer* de encontro: como visto anteriormente cada *peer* pode ou não ser um *peer* de encontro (configuração mostrada na Figura 20), por padrão todos os *peers* são tanto de encontro quanto de *relay*. Quando um *peer* de encontro é encontrado na rede, o mesmo fica armazenado em *cache* local isso também gera uma linha de texto com o *peer* ID deste *peer*;
d) anúncio de conteúdo encontrado: cada anúncio de conteúdo encontrado, gera uma linha contendo nome do *peer* que o possui e o nome do próprio anúncio;
e) envio de mensagens: todas as mensagens enviadas por qualquer *peer* da rede geram uma linha contendo o nome do *peer* que a envia, e o texto da própria mensagem.

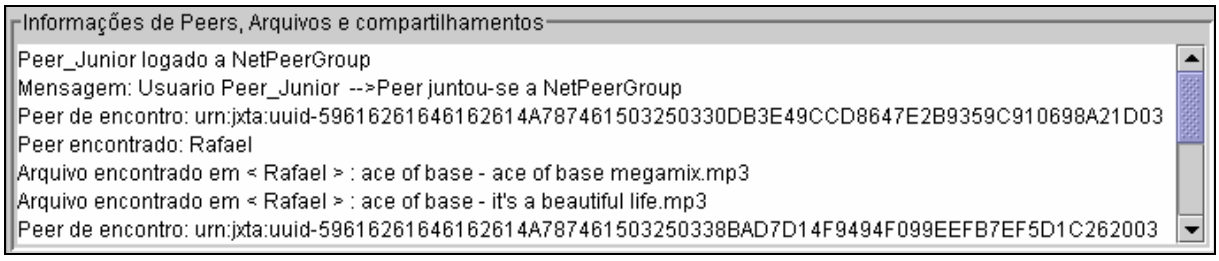


Figura 22 – Informações de peer e anúncio de arquivos

4.3.2.1 COPIAR CONTEÚDO

O protótipo oferece uma maneira muito simples para efetuar cópia de conteúdos dos anúncios descobertos. Para efetuar a cópia de um arquivo, deve-se inicialmente selecionar um *peer* (que não seja o local) e em seguida selecionar um anúncio de conteúdo disponível na parte direita do protótipo, conforme apresentado na Figura 23. Ao ser pressionado o botão direito do mouse sobre o conteúdo selecionado, será mostrado um *pop-up,* que ao ser pressionado oferece opção de selecionar um diretório para destino do conteúdo.

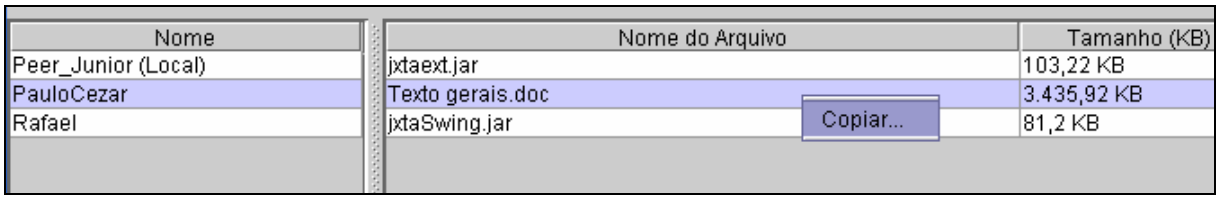


Figura 23 – Cópia de conteúdo

4.3.2.2 COMPARTILHAR CONTEÚDO

Para compartilhar conteúdo com a rede P2P é necessário movê-lo ou copiá-lo para pasta compartilhada, conforme indicado na Figura 24. Ao fazer isso o protótipo se encarrega de fazer a leitura dessa área e gerar um anúncio para cada arquivo ainda não anunciado.

O protótipo não permite que seja compartilhada estrutura de diretório por completo, mas somente arquivos de qualquer formato.

Quando o protótipo é executado pela primeira vez é criado a partir do diretório raiz da execução, uma pasta chamada de pastaPublica, que conterá todos os conteúdos compartilhados desse *peer*. Será criada também uma pasta chamada de copiados, criada como padrão de cópia de todos os conteúdos remotos.

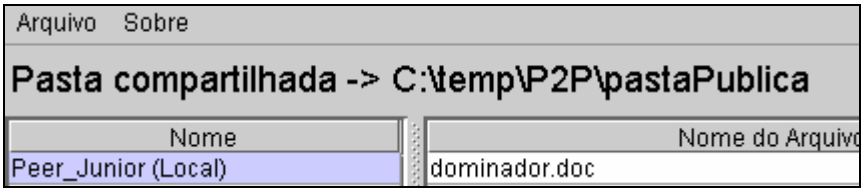


Figura 24 – Pasta compartilhada

4.4 RESULTADOS E DISCUSSÃO

Observou-se que no funcionamento do protótipo em redes locais, a procura de anúncios, tanto de *peers* quanto de conteúdos acontece de forma eficiente. Em uma rede de aproximadamente duzentos computadores, onde cada computador representa um *peer*, foram selecionados quatro deles para execução do protótipo. Cada *peer* reconheceu em cerca de um minuto todos os outros três *peers* envolvidos, bem como encontrou todos seus anúncios de conteúdos, que não passou de cinco anúncios por *peer*.

Foi possível também, enviar mensagens para todos os *peers* envolvidos na rede. Quando uma mensagem é enviada todos os *peer* a recebem.

Em redes locais a plataforma JXTA faz descobertas diretas de anúncios, utilizando recursos *multicast ou broadcast*. Já na Internet, as descobertas podem ser feitas utilizando recursos de descoberta indireta. Com isso nota-se que a comunicação entre *peers* via Internet é mais lenta que na rede local.

Ao utilizar o protótipo via Internet, notou-se que a comunicação se tornou mais eficiente quando adicionado um número maior de *peers* de encontro para o *peer*. Pois cada *peer* procura anúncios disponíveis em cada um dos *peers* de encontro conhecidos.

5 CONCLUSÕES

No decorrer do desenvolvimento desse trabalho verificou-se a complexidade de cada componente que faz parte da rede P2P. Teve-se uma idéia geral de como é realizada cada uma das etapas da comunicação, e como de fato os *peers* se comunicam sem haver um servidor central processando todas as requisições.

Ao utilizar a plataforma JXTA para implementação de uma rede P2P, observou-se o quão direto ficou implementar uma comunicação entre computadores sem que haja um servidor, pois a mesma oferece uma biblioteca que resolve as questões complexas de comunicação P2P. O desenvolvedor sequer precisa conhecer em detalhes todos os conceitos de como é realizada a comunicação entre os *peers*.

Em relação aos objetivos propostos no inicio do desenvolvimento deste projeto, todos foram alcançados com sucesso, já que o mesmo demonstra a capacidade de comunicação de redes P2P construídas com a plataforma JXTA.

Tem-se agora uma noção das possibilidades de futuros desenvolvimentos utilizando-se essa tecnologia ainda pouco explorada, mas que vem ganhando bastante força de mercado. O que pode impulsionar a utilização de softwares P2P é justamente a plataforma JXTA, que apresenta grande facilidade nas implementações P2P.

A vantagem na utilização deste protótipo de compartilhar arquivos e mensagens com outros computadores é que em redes locais não há necessidade de compartilhar pastas na rede, pois tudo que é colocado na chamada pasta pública do protótipo é compartilhado automaticamente com todos os *peers*. Na Internet todos *peers* da rede podem copiar os arquivos anunciados, sem precisar haver um servidor WEB instalado ou haja disponibilidade em outro servidor qualquer.

Uma limitação do protótipo é que não há compartilhamento de estrutura de pastas e sim somente de arquivos. Outra desvantagem é o excesso de congestionamento que uma rede pode sofrer, pois a cada dez segundos são enviadas requisições, tanto de pesquisa quanto de anúncio (caso houver).

5.1 EXTENSÕES

Como extensão ao protótipo pode-se implementar um algoritmo que não faça pesquisas de anúncios a cada dez segundos, e sim somente uma pesquisa manual por nome de *peer* ou arquivo, isso pode evitar o excesso de congestionamento que uma rede pode ter.

Pode-se também implementar um software que faça compartilhamento de estruturas de pastas e não somente de arquivos.

Outra possibilidade é implementar um programa que faça a criação de grupos de *peers* seguros, já que neste protótipo há somente o *netPeerGroup* com autenticação padrão. Com grupos seguros pode se fazer compartilhamento de dados entre empresas e seus clientes.

REFERÊNCIAS BIBLIOGRÁFICAS

AOL. **AOL instant Messenger**, Atlanta, USA, 2004. Disponível em: <http://americaonline.com.br/aim >. Acesso em: 30 maio 2004.

COMER, Douglas; DAVID, L. Stevens. **Interligação em rede com TCP/IP**. Tradução: Ana Maria Neto Guz. Rio de Janeiro: Campos, 1999. 594 p.

ECLIPSE. **Main page of Project**, Canada, 2004. Disponível em: <http://www.eclipse.org >. Acesso em: 05 jun. 2004.

FURLAN, Jose Davi. **Modelagem de objetos através da UML-The Unified Modeling Language**. São Paulo: Makron Books, 1998. 329 p.

GNUTELLA. **Presentation**, Philadelphia, USA, 2004. Disponível em: <http://www.gnutella.com >. Acesso em: 20 abr. 2004.

GOOD, Nathaniel S. **Usability and privacy: a study of Kazaa P2P file-sharing**, Palo Alto, USA, [2003?]. Disponível em: <http://www.hpl.hp.com/shl/papers/kazaa/KazaaUsability.pdf>. Acesso em: 30 de mar. 2004.

ICQ INSTANT MESSENGER. **The Community**, Atlanta, USA, 2004. Disponível em: <http://company.icq.com/info/icqstory.html>. Acesso em: 30 maio 2004.

JXTA. **Project JXTA**, San Jose, USA, 2004. Disponível em: <http://www.jxta.org >. Acesso em: 01 jul. 2004.

JXTA PROGRAMMER GUIDE. **Project JXTA v2.0: Java Programmer´s Guide**, San Jose, USA, 2004. Disponível em: < http://www.jxta.org/docs/JxtaProgGuide_v2.pdf >. Acesso em: 01 jul. 2004.

KELLERER, Wolfgang. **Dienstarchitekturen in der Telekommunikation Evolution, Methoden und Vergleich**, Muenchen, Alemanha, 1998. Disponível em: <http://www.lkn.ei.tum.de/lkn/publications/1998/wk_TechReport_9801.pdf >. Acesso em: 25 mar. 2004.

MICROSOFT CORPORATION. **Msn Messenger**, São Paulo, 2004. Disponível em: <http://messenger.msn.com>. Acesso em: 30 maio 2004.

NAPSTER. **Share of music**, San Diego, USA, 2004. Disponível em: <http://www.napster.com >. Acesso em: 20 abr. 2004.

SILVA, William Roger Salabert. **Introdução às redes *Peer-to-Peer*(P2P)**. Rio de Janeiro, [2003?]. Disponível em: < http://www.gta.ufrj.br/seminarios/semin2003_1/william/index.htm >. Acesso em: 15 mar. 2004.

SYBASE. **e-Business software integration with databases**, San Francisco, USA, 2004. Disponível em: <http://www.sybase.com>. Acesso em: 23 maio 2004.

WILSON, Brendon J. **Project Jxta book**, Vancouver, Canada, [2004?]. Disponível em: <http://www.brendonwilson.com/projects/jxta/pdf/JXTA.pdf>. Acesso em: 27 maio 2004.

ANEXO A – Bibliotecas do Projeto JXTA utilizadas no protótipo implementadas com a linguagem de programação Java

Todas as bibliotecas necessárias para o desenvolvimento do protótipo estão disponíveis na seção *downloads* do *site* do projeto JXTA em <http://www.jxta.org>.

As bibliotecas estão divididas conforme quadro abaixo:

Nome	*Site*	Biblioteca
platform	<http://platform.jxta.org>	jxta.jar * jxtaext.zip ** jxtalib.zip ** jxta-doc.zip platform-changelog.zip platform-src.zip
security	<http://security.jxta.org>	jxtasecurity.jar * security-src.zip
cms	<http://cms.jxta.org>	jxtacms.jar * jxtacms-doc.zip cms-src.zip
shell	<http://shell.jxta.org>	jxtashell.zip jxtashell-doc.zip shell-src.zip
instantp2p	<http://instantp2p.jxta.org>	instantp2p.zip * instantp2p-doc.zip instantp2p-src.zip

* - utilizado no protótipo
** - utilizado no protótipo em modo descompactado

ANEXO B – Bibliotecas do Projeto JXTA implementadas em linguagens de programação C, Perl 5, Python, Ruby e Smalltalk

Bibliotecas implementadas em outras linguagens de programação, tais como C, Perl 5, Python, Ruby e Smalltalk podem ser encontradas nos endereços eletrônicos mostradas no quadro abaixo:

Nome	*Site*
jxta-c	<http://jxta-c.jxta.org/servlets/ProjectHome>
jxtaperl	<http://jxtaperl.jxta.org/servlets/ProjectHome>
jxtapy	<http://jxtapy.jxta.org/servlets/ProjectHome>
jxtaruby	<http://jxtaruby.jxta.org/servlets/ProjectHome>
smalltalkjxta	<http://smalltalkjxta.jxta.org/servlets/ProjectHome>