UNIVERSIDADE ESTADUAL DO NORTE DO PARANÁ - CAMPUS LUIZ MENEGHEL

RONALDO PITOLI

ARMAZENAMENTO E COMPARTILHAMENTO DE DADOS UTILIZANDO ARQUITETURA PEER-TO-PEER PARA BIBLIOTECA DIGITAL

RONALDO PITOLI

ARMAZENAMENTO E COMPARTILHAMENTO DE DADOS UTILIZANDO ARQUITETURA PEER-TO-PEER PARA BIBLIOTECA DIGITAL

Trabalho de Conclusão de Curso submetido às Universidade Estadual do Norte do Paraná Campus Luiz Meneghel, como requisito parcial para a obtenção do grau de Bacharel em Sistemas de Informação.

COMISSÃO EXAMINADORA

_____

Prof. Msc Ricardo Gonçalves Coelho
Uenp - Campus Luiz Meneghel

_____

Prof. Luiz Fernando Legore do Nascimento
Uenp - Campus Luiz Meneghel

_____

Prof. Msc Ederson Marcos Sgarbi
Uenp - Campus Luiz Meneghel

Bandeirantes, __ de_____ de 2010

A minha esposa e meus filhos.

Agradecimentos

De forma especial agradeço a minha família e minha esposa Ângela por ter me dado força em todos os momentos, ser compreensível na minha ausência, peço desculpa a ela e meus filhos por ter me abdicado deste tempo com eles. Obrigado pelo amor de vocês carinho e compreensão!

Agradeço a mesmos amigos companheiros de viagem, e especialmente a estes que tiveram coragem de ir junto comigo de carro, Vanessa, Bismark, Luiz e Renata, desculpa, obrigado, e aqueles que de perto ou longe, torceram por mim em mais este desafio. A vocês, os meus sinceros agradecimentos!

E por fim agradeço ao meu orientador Ricardo, por ter tido sabedoria e paciência e por me ajudar nos problemas encontrados para desenvolvimento deste trabalho.

"Para realizar grandes conquistas, devemos não apenas agir, mas também sonhar; não apenas planejar, mas também acreditar."

(Anatole France)

Conteúdo

Lista de Figuras

Lista de Siglas

P2P - Peer-to-peer

DHT - (*Distributed Hashing Table*), tabela de resumo distribuída

ID - Identificador do nó

IP - Internet Protocol

Resumo

Com o uso da Internet e os meios de comunicações atuais cada vez mais são gerados dados em meios digitais, e a grande dificuldade esta em armazenar e distribuir estes dados, para isto foi desenvolvido uma aplicação de biblioteca digital utilizando a arquitetura P2P. Esta aplicação foi desenvolvida na liguagem Java em conjunto com a biblioteca FreePastry, a qual visa criar dois anéis de comunicação unindo redes diferentes. Para gerenciamento e comunicação entre estas redes os nós precisam estar conectados a um super-peer, o qual gerencia esta comunicação, ficando este responsável por responder as buscas dos nós e ainda publicar os arquivos enviados pelos nós. O controle dos arquivos é feito por um catálogo de dados sendo este um arquivo XML que contém informações sobre os arquivos publicados no sistema. Para o gerenciamento deste catálogo o definimos de forma centralizada em um único ponto sendo este o super-peer, desta forma, em uma busca não se exige um auto-custo com a largura de banda da rede, pois, como o catálogo dos arquivos esta publicado em um único ponto, o nó envia a busca para somente um ponto na rede.

Abstract

Using the Internet and modern communication means more data are generated in digital media, and the great difficulty is in storing and distributing these data, this was developed for a digital library application using the P2P architecture. This application was developed in Java language-along with the library FreePastry, which aims to create two rings of communication linking different networks. For management and communication between these network nodes must be connected to a super-peer, which manages the communication, this being responsible for answering queries of nodes and even publish the files uploaded by nodes. The control of the files is done by a catalog of data which is an XML file that contains information about the published files on the system. For management of the catalog set centrally at one point being the super-peer in this way in a search does not require a self-cost of the bandwidth of the network because, as the catalog of this file is published in a single point, the node sends the search to just one point in the network.

1 Introdução

Com o grande aumento de informações arquivadas em meios digitais, cada vez mais se necessita de meios para preservar e garantir a disponibilidade destas informações.

Como os sistemas digitais estão suscetíveis a diversas vulnerabilidades tais como: ataques de hackers, bugs, vírus e ainda uma pane no sistema, estes fatores motivam a criação de um sistema de preservação e distribuição de dados, que visa garantir a integridade dos dados armazenados nele e que garanta a alta disponibilidade do sistema e arquivamento em longo prazo.

Segundo (Vignatti, 2009), uma das preocupações dos sistemas de preservação digital esta em como estes dados estarão armazenados e este requisito é de responsabilidade do sistema de preservação, o sistema de preservação deve ter baixo custo, garantir alta disponibilidade e segurança nos dados contidos nele.

Neste contexto, a arquitetura *peer-to*-peer (P2P), segundo (Rocha, et al, 2004), vem para quebrar o paradigma existente hoje, pois, todas as aplicações WEB são construídas na arquitetura cliente servidor, a qual existe um servidor central que disponibiliza os serviços e os clientes que utilizam estes serviços.

Na arquitetura P2P não tem um servidor central, o que a difere da cliente servidor, pois, os pontos podem fornecer ou utilizar recursos disponíveis por outros pontos, nesta arquitetura os pontos ficam ligados entre si.

Esta arquitetura pode ser divida em duas categorias: Descentralizado: Nesta categoria não há um ponto central e todos os nós têm o mesmo nível, os quais ficam responsáveis por gerenciar e controlar a troca de mensagens e recursos. Os nós podem se comunicar de maneira direta com outros nós da rede. Semicentralizado: Nesta há diferença entre os nós havendo um nó central para gerenciamento e controle de tráfego de dados chamado de supernó, os nós inferiores possuem um mesmo nível, e em caso de queda do supernó afeta somente os nós conectados a ele.

A forma de comunicação do sistema P2P pode ser definida numa perspectiva de alto nível de rede *overlay* (Rede sobreposta), uma vez que

funciona como uma rede virtual, interligando os nós em cima de uma rede física.

Fator importante na disponibilidade do sistema é que esta arquitetura é heterogênea, ou seja, trabalha com diferentes plataformas ou sistemas operacionais, o que a torna menos vulnerável, pois mesmo que ocorra problema em um sistema especifico, como, vírus ou um próprio bug a aplicação permanece ativa, garantindo assim alta disponibilidade. Nesta arquitetura a qual utiliza a tecnologia de redes sobrepostas, fornecem um subsidio maior para a disponibilidade do sistema, em que a rede sobreposta, segundo (Obelheiro e Fraga, 2007), é uma rede lógica construída em cima da rede física, que fornecem serviços seguros e possuem tolerância a falhas.

Também é importante ser levado em conta o processo de persistência que é a garantia de segurança e recuperação dos arquivos armazenados, como no sistema P2P, os pontos podem ser voláteis, conectar ou desconectar a qualquer momento, faz se necessário um controle que garanta a disponibilidade dos arquivos contidos, uma solução encontrada por, (Rowstron, Druschel, 2001), é o uso de réplica de arquivos, o qual pode se distribuir os arquivos por diversos pontos da rede P2P, garantindo assim persistência nos dados armazenados.

Neste contexto, a arquitetura P2P fornecem abordagens promissoras para o sistema de biblioteca digital, a qual garante alta disponibilidade, pois, o sistema estará disponibilizado por diversos pontos, também garante maior segurança nos dados contidos, pois, com a distribuição e das réplicas dos arquivos, mesmo que ocorra problema em um dos pontos o arquivo pode ser acessado através de outro ponto, além um baixo custo, pois, de acordo com esta arquitetura não é necessário investir um alto valor em um único equipamento com alto poder de armazenamento e processamento, pois o sistema irá consumir uma pequena parte do poder de armazenamento e processamento de cada ponto.

1.1 Objetivos

Neste contexto o objetivo deste trabalho vem para elucidar as principais características das aplicações P2P, e desenvolver uma aplicação para Biblioteca Digital, interligando colégios universidades ou instituições publicas, o intuito é criar com o uso desta arquitetura dois anéis de comunicações entre os pontos, conforme apresentado na Figura 1, o primeiro anel ficará responsável por disponibilizar e gerenciar as réplicas dos arquivos na rede interna e o segundo anel ficará responsável por disponibilizar e gerenciar as réplicas dos arquivos e a comunicação com os demais pontos externos.

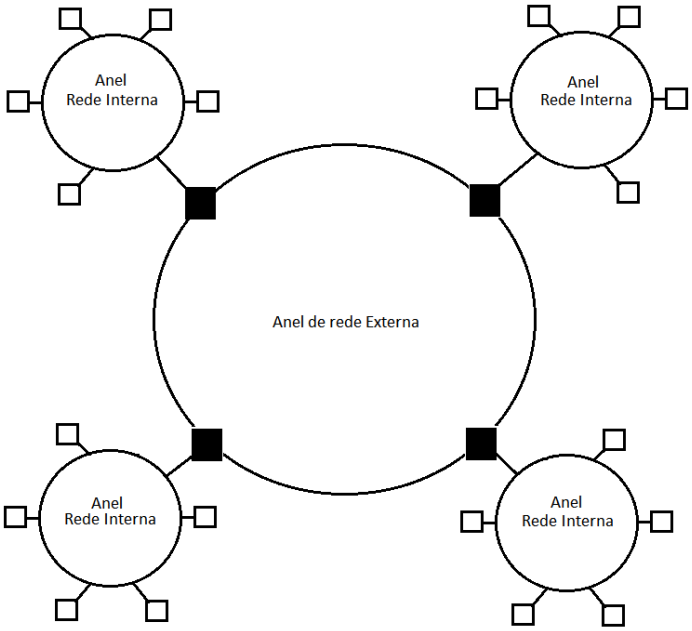

Figura 1 - Anel de comunicação - A figura ilustra a forma de comunicação que será usado para interligar as redes internas e externas.

Com a utilização destes dois anéis de comunicação o sistema terá maior disponibilidade e persistência, pois, mesmo que ocorra problema com a comunicação externa da instituição o sistema permanece ativo, pois o segundo anel de comunicação garante o acesso ao sistema e as réplicas dos arquivos.

Por meio da arquitetura P2P, com ajuda o sistema Past conforme descrito em (Rowstron, Druschel, 2001), podemos definir um número X de réplicas dos arquivos armazenados neste sistema, garantindo maior segurança nos dados contidos nele. Assim quando um usuário tentar buscar um arquivo e o ponto que contém este arquivo apresentou problema ou uma falha, o sistema busca automaticamente outro ponto que ponto que contenha a réplica do arquivo em questão.

1.2 Justificativa

Cada vez mais alunos e professores têm a necessidade de acesso a informações e conteúdos acadêmicos e a maioria destes conteúdos está em papel impresso como, livros, revistas, os quais ficam na maioria das vezes arquivados em bibliotecas, o que restringe seu acesso ao horário de funcionamento da mesma.

Com a utilização de um sistema de distribuição e de dados por meios digitais garantimos a disponibilização destes documentos à qualquer momento, e ainda estes podem ser acessados de qualquer lugar, pois, com este sistema pode se publicar ou requisitar arquivos, bastando o usuário ter acesso a internet.

Em (Eternal, 2005) é descrito, que a Biblioteca de Washington nos Estados Unidos demorou cerca de duzentos anos para conseguir um acervo de 29 milhões de exemplares de livros, com o avanço das tecnologias e os meios de comunicação, a internet hoje produz conteúdo novo equivalente em cerca de quinze minutos. O problema não está em como se produzir conteúdo em meios digitais e sim como garantir o acesso e a persistência destes dados.

Um fato ao qual convivemos hoje é que empresas e instituições publicas, precisam preservar dados como: informações fiscais, dados de funcionários e outros por um determinado período de tempo, que em alguns casos estes dados devem ser preservados por um período mínimo de cinco anos.

Com isso a utilização da arquitetura P2P para desenvolvimento da aplicação de biblioteca digital é a mais adequada, pois neste sistema existe uma forte ênfase em controle descentralizado, adaptação e escalabilidade. Além disso, com o sistema de controle de réplicas agregado ao sistema P2P e distribuição dos arquivos por diversos pontos na rede conseguimos maior persistência nos dados, pois, mesmo que ocorra problema em um ponto os arquivos presentes nele poderão ser acessados através de outro ponto.

Este trabalho está organizado da seguinte maneira. Capítulo 2 apresenta introdução aos sistemas distribuídos e as características do sistema Peer-to-peer, Capítulo 3 apresenta o funcionamento do sistema Pastry e seu funcionamento e a maneira que o sistema trata os dados em sua DHT, o sistema Juxmen e suas particularidades. No capítulo 4 apresenta os materiais e métodos aplicados para o desenvolvimento do trabalho e o desenvolvimento do sistema de biblioteca digital, e os testes realizados. No capítulo 5 é apresentado a conclusão do trabalho e os trabalhos futuros.

2 Fundamentação Teórica.

Neste capítulo são relatados os conceitos a qual o trabalho foi embasado.

2.1 Sistemas Distribuídos

Com o avanço das tecnologias e os meios de comunicação cada vez mais aumenta as redes de computadores, seguindo este fato a Internet a qual faz parte deste sistema, vem cada vez mais ganhando espaço e notoriedade. Sendo assim a Internet é composta por diversas redes como, redes domésticas, telefones móveis, redes corporativas, redes em campus e até em veículos.

Em (Coulouris et al., 2007), defini Sistema distribuído como, aquele no qual os componentes de *hardware* e *software* interligados em rede, compartilham informações coordenam suas ações e trocam mensagens entre si, estes sistemas podem estar separados por continentes, ou ainda na mesma sala. Estes sistemas têm as seguintes características:

Concorrência: Neste tipo de sistema existe a concorrênvcia entre os processos, como exemplo: posso estar trabalhando em meu computador, em quanto outro usuário utiliza seu computador, neste mesmo tempo estes computadores podem estar trocando informações entre si, ou compartilhando páginas web.

Inexistência de relógio global: Neste sistema como os programas precisam cooperar entre si não existe um relógio global, e sim um controle de tempo para a troca de mensagens.

Falhas Independentes: Todos os sistemas de computadores podem falhar só que em sistemas distribuídos estas falhas são diferentes, falhas na rede resultam de isolamento dos computadores conectados a ela. Mesmo assim em caso de parada de um dos recursos do sistema distribuídos o sistema deve permanecer operante.

Escalabilidade: O sistema pode ser dizer escalável quando há um aumento significativo no número de recursos ou em número de usuários, estas escalas podem ter medidas diferentes como, por exemplo, uma intranet ou a Internet.

Dentro de Sistemas Distribuídos existem os tipos de arquiteturas Cliente Servidor e *Peer-to-Peer*: na arquitetura cliente servidor existe um servidor central responsável por fornecer serviços aos clientes, por ser um ponto central, este é o seu ponto fraco, pois em caso de parada deste servidor os clientes perdem os serviços disponibilizados pelo servidor. Na arquitetura P2P todos os clientes têm papeis iguais, podem enviar ou fazer solicitações a outros pontos na rede não contendo um elo central de conexão, tornando assim a rede mais segura. Para desenvolvimento do sistema de biblioteca digital a arquitetura P2P é a mais adequada, por fornecer controle descentralizado, maior escalabilidade e auto organização, a qual descrevemos na seção 2.2.

2.2 Arquitetura Peer-to-Peer

Em (Kamienski et al., 2005), diz que na literatura se encontra diversas definições sobre sistemas P2P, a encontrada em (Coulouris et al., 2007), diz que este sistema não se faz distinção de função dos envolvidos todos os usuários podem desempenhar papel de cliente ou de servidor. Embora o sistema cliente servidor forneça uma estratégia direta e relativamente simples para o compartilhamento de dados ou recursos, ele não é flexível em termos de escalabilidade, pois, o fornecimento dos serviços está a cargo de um único servidor o que limita sua capacidade e a largura de banda de suas comunicações.

Objetivo da arquitetura P2P, de acordo com (B. Yang, H. G. Molina, 2002), é explorar recursos de um grande número de computadores para uma dada tarefa ou atividade, foram construídos aplicativos com sucesso que permitem que dezenas, centenas ou milhares de computadores se comuniquem e compartilhem recursos ou dados entre si. O exemplo mais

antigo desta arquitetura conhecido foi o Napster empregado para o compartilhamento de música digital.

Segundo (Rowstron, Druschel, 2001), os sistemas P2P tiveram seu início com o compartilhamento de arquivos, mas ainda estes sistemas tem aspectos técnicos muito interessantes e que podem ser explorados como, controle descentralizado, auto organização, adaptação e escalabilidade.

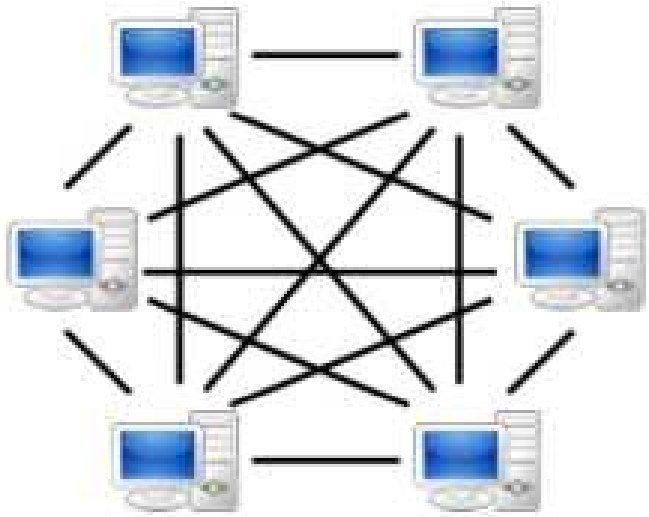

Figura 2 - Sistema Peer-to-peer.

A Figura 2 ilustra o formato de um aplicativo P2P, os quais são compostos de um grande número de *peers*, executados em computadores distintos e o padrão de comunicação dependem do aplicativo utilizado.

Um dos primeiros sistemas a utilizarem a arquitetura P2P e ganhar escalas globais foi o Napster, sistema de compartilhamento de arquivos de músicas digitais. Por ser um sistema que utiliza um controle central com índices dos arquivos, é o que mais se assemelha com a aplicação de biblioteca digital, pois, no sistema de biblioteca digital mantem-se um catálogo de documentos, semelhante ao Napster o qual descrevemos com maiores detalhes na próxima seção.

2.3 Napster

Segundo (MV Neves, 2006), a primeira versão de um sistema P2P a ser desenvolvida para armazenamento e recuperação de dados em escala global, foi com o intuído de compartilhar arquivos de música digital, e estas características foram encontradas pela primeira vez no sistema Naspter, o qual tornou este aplicativo muito popular e logo após seu lançamento em 1999, atingiu seu auge, com milhares de usuários registrados os quais trocavam milhares de arquivos de música simultaneamente.

A arquitetura do Napster segundo incluía índices centralizados, o qual armazenava a lista de usuários e arquivos disponibilizados pelos mesmos, mas eram os usuários que forneciam os arquivos, que eram armazenados e acessados em seus computadores pessoais, conforme demonstrado na Figura3.

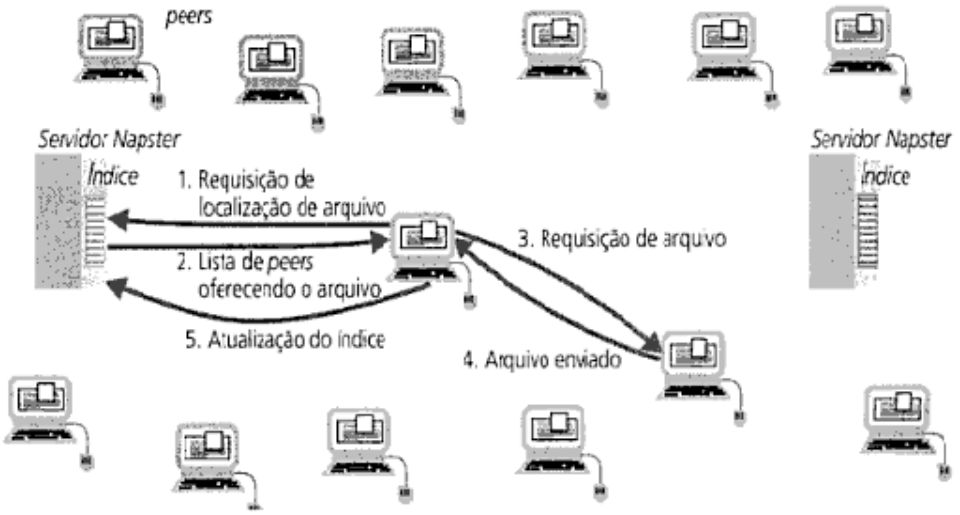

Figura 3 - Funcionamento do Naspter.

O Naspter foi desativado devido ao grande número de ações judiciais sofrida de proprietários de direitos autorais, dos arquivos de música digital disponibilizado por ele. Mesmo os desenvolvedores do Naspter alegando que

não eram responsáveis pela violação dos direitos autorais, pois, não participavam do processo de troca dos arquivos, o qual era realizado inteiramente por seus usuários, esta alegação fracassou, porque os servidores de índice foram considerados uma parte essencial do processo, como os servidores estavam localizados em endereços conhecidos, seus operadores eram incapazes de manter anonimato, e, portanto poderiam ser alvos de ações judiciais.

Uma das lições aprendidas com o Napster é de manter o anonimato dos receptores e dos provedores de dados, existem justificativas sociais e política legitima para o anonimato dos clientes e servidores, a justificativa mais persuasiva surge quando o anonimato é usado para vencer a censura e mantiver a liberdade de expressão dos indivíduos, em sociedade e organizações opressivas.

Algumas das limitações do Naspter como, consistência das réplicas e garantia de disponibilidade dos dados não eram requisitos fundamentais para o funcionamento da aplicação, pois, as réplicas dos arquivos de músicas nunca são atualizadas, e quanto a disponibilidade mesmo que o arquivo não esteja disponível naquele momento, ele pode ser carregado posteriormente.

2.4 Middleware para peer-to-peer

Segundo (Coulouris et al., 2007), a primeira geração dos sistemas P2P surgiu com o Naspter, o qual tinha intuito o compartilhamento de arquivos de músicas, a segunda geração surgiu com os aplicativos de compartilhamento de arquivos, fornecendo maior escalabilidade, anonimato e tolerância a falhas como: Freenet, Gnutella, Kazaa e Bit-Torrent. Na terceira geração é caracterizada pelo aparecimento da camada de *middleware*, para gerenciamento dos recursos, dentro desta terceira geração incluem os sistemas Pastry, Tepastry, Can e Chord.

A função do *middleware* P2P é simplificar a construção e utilização de serviços em muitos hosts distribuídos na rede, o qual deve permitir que os hosts se localizem e se comunique trocando informações, outra função importante é que os sistemas devem garantir que os hosts possam se conectar e desconectar a vontade, adicionando e removendo serviços.

Para o bom funcionamento do *middleware* P2P, o sistema deve garantir:

Escalabilidade Global: O sistema deve permitir que e suportar a conexão de milhares de hosts e o permitir o acesso a centenas ou milhares de objetos.

Balanceamento de Carga: O desempenho deve ser garantir com a distribuição de carga entre os hosts e ainda trabalhar em conjunto com o uso de réplicas.

Otimização das interações locais entre os peers vizinhos: Para garantir melhor controle e consumo da rede o sistema deve garantir a conexão entre os pontos, e assim conectando os hosts mais próximos uns dos outros, evitando assim um gasto desnecessário com a rede.

2.5 Sobreposição de Roteamento

Segundo (Coulouris et al., 2007), o algoritmo conhecido como sobreposição de roteamento (*routing overlay*) assume a responsabilidade por localizar nós e objetos. Este processo garante que os objetos podem ser alocados e movidos a qualquer nó da rede sem o envolvimento do cliente, este mecanismo é chamado de sobreposição de roteamento o qual é implementado na camada de aplicação que é totalmente separado dos demais mecanismos de roteamento implementados em nível da rede, como o roteamento IP.

Como os sistemas P2P, têm diversos nós espalhados pela rede o sistema de sobreposição de roteamento garante que o requisitante de um serviço ou arquivo acessará a réplica mais próxima a ele garantindo assim menor consumo com o acesso a rede. Todo o objeto disponível no sistema é controlado por um número único chamado de GUID, quando o sistema tenta

localizar um objeto ele procura o seu GUID, na DHT (*Distributed Hashing Table*), tabela de resumo distribuída, facilitando assim a busca por um objeto.

Em sistemas como Chord segundo (Vignatti, 2009), para se localizar as chaves de maneira eficiente o sistema executa uma busca de forma circular com ajuda da tabela de roteamento, a qual é armazenada em cada nó e esta contém informações sobre seus vizinhos como: identificadores e endereço IP.

No sistema CAN a utiliza em sua arquitetura o espaço de coordenadas cartesianas para organizar seus nós, sendo que cada nó é responsável por uma área, suas buscas são encaminhadas para o nó vizinho mais próximo geograficamente.

O sistema Pastry mantém sua tabela de roteamento estruturada em forma de árvore mantendo um valor X para o número de nós folhas, a qual mantém informações sobre seus vizinhos mais próximos, contendo o ID do vizinho e seu endereço IP, a DHT Pastry é apresentando com maiores detalhes na seção 3.1.1.

3 Trabalhos Relacionados

3.1 Sistema Pastry

O sistema Pastry é um sistema de sobreposição de roteamento, utilizando a tecnologia P2P*,* que segundo (Rowstron, Druschel, 2001), este sistema possui características de armazenamento global o qual proporciona forte persistência, alta disponibilidade, escalabilidade global, segurança, resistência a falhas e de auto organização.

De acordo com (Rowstron, Druschel, 2001), quando um usuário salva um arquivo no sistema Pastry, este cria um número K de réplicas para garantir a disponibilidade e persistência deste arquivo, utilizando esta técnica de armazenamento parece que o sistema não terá balanceamento eficiente, mas o sistema resolve isto de duas maneiras:

Primeiro o sistema permite que um dos nós GUID que não é mais próximo do usuário salve uma cópia do arquivo para garantir sua persistência.

Segundo desvio de arquivo, este é executado quando um nó folha atinge seu limite de capacidade, enviando este arquivo para o nó mais próximo.

Armazenamento por nó, o sistema Pastry faz controle dos nós representando como uma árvore, o qual cada conexão com o sistema é representada por um nó.

O sistema faz um controle de capacidade de armazenamento, se for muito grande a capacidade de um nó é convidado a se dividir em múltiplos nós, e se ele é muito pequeno é rejeitado, e o nó só anuncia uma fração de seu verdadeiro espaço, para não ter uma sobrecarga.

Para garantir a persistência dos dados armazenados este sistema de gerenciamento faz alguns controles conforme exemplificado abaixo:

Desvio de réplica, quando o sistema tenta salvar um arquivo no nó A e este não tem espaço suficiente o nó grava o arquivo no nó mais próximo, B,

salvando um ponteiro em sua base e apontando para o B, posteriormente o nó emiti o recibo de que o arquivo foi salvo.

Desvio de arquivo: Quando o usuário tenta salvar o arquivo grande o sistema faz um controle de equilíbrio salvando o arquivo no nó com maior disponibilidade, e se nenhum dos nós suportar o arquivo este tenta salvar o arquivo para um número de réplicas menores ou tenta compactar o arquivo.

Réplicas de manutenção: Para saber se os nós ainda permanecem ativos o sistema envia mensagens de verificação periodicamente, e quando um nó apresenta problema o sistema não o remove da lista de nós ele simplesmente aponta para o nó mais próximo, e assim que faz o apontamento, o sistema exige que o nó faça uma réplica de cada arquivo presente no nó que falhou. E quando um novo nó se adere ao sistema este entra como nó folha.

3.1.1 DHT Pastry

Para gerenciamento dos nós ativos no sistema Pastry este faz um controle através de uma tabela de roteamento, o qual cada nó mantém sua tabela estruturada em forma de árvore, mantendo nesta tabela um valor L para o tamanho de nos nós folhas, as quais mantêm informações sobre seus vizinhos mais próximos, fornecendo os GUIDs e endereços IP, destes vizinhos.

p =	Prefixos de GUID e manipuladores de nós n correspondentes															
0	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
	n	n	n	n	n	n		n	n	n	n	n	n	n	n	n
1	60	61	62	63	64	65	66	67	68	69	6A	6B	6C	6F	6E	6F
	n	n	n	n	n		n	n	n	n	n	n	n	n	n	n
2	650	651	652	653	654	655	656	657	658	659	65A	65B	65C	65D	65E	65F
	n	n	n	n	n	n	n	n	n	n		n	n	n	n	n
3	65A0	65A1	65A2	65A3	65A4	65A5	65A6	65A7	65A8	65A9	65AA	65AB	65AC	65AD	65AE	65AF
	n		n	n	n	n	n	n	n	n	n	n	n	n	n	n

Figura 4 - Tabela de Roteamento - A tabela de roteamento está localizada no nó que tem seu GUID 65A1, os dígitos estão em hexadecimal e as letras n representam (GUID, endereço IP), facilitando assim o sistema de busca.

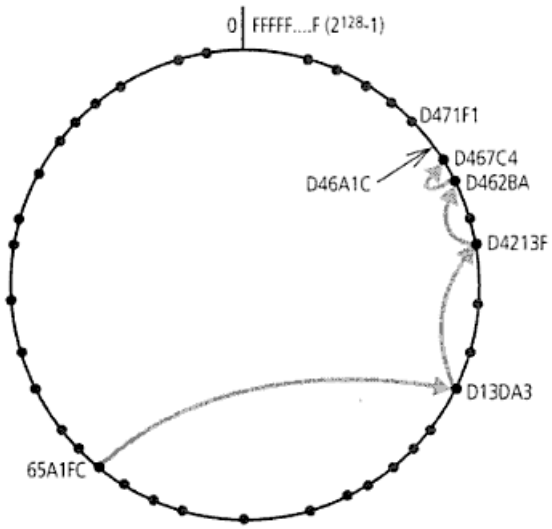

Figura 5 - Roteamento Mensagem - Os pontos representam nós ativos no sistema, a imagem ilustra o roteamento de uma mensagem do nó 65A1FC para D46A1C, com a ajuda da tabela de roteamento a mensagem pode ser enviada com um pequeno número de saltos.

Quando um novo se adere ao sistema, este utiliza um algoritmo de associação para preencher sua tabela de roteamento, primeiramente ele

calcula seu GUID, supondo que o nó X ingressou na rede e o nó mais próximo a ele seja A, o sistema encaminha uma mensagem de *join* dos pontos A a Z, solicitando a estes os conteúdos de suas tabelas de roteamento e dados dos nós folhas.

Falha ou saída de um nó, neste sistema os nós podem falhar ou sair sem aviso, por isso o sistema utiliza um método de *keep-alive*, pulsação em um intervalo de tempo determinado, para informar se o nó está vivo. Um nó é considerado defeituoso quando seus nós não conseguem mais comunicar com ele, e quando isto ocorre é necessário reparar o conjunto de nós folhas, o nó que descobre a falha solicita ao nó mais próximo do nó defeituoso e sua tabela de roteamento, sobrepondo esta tabela e avisa seus vizinhos da alteração, os quais executam procedimento semelhante.

De acordo com (Rowstron, Druschel, 2001), o qualquer máquina conectada na internet pode atuar como um nó no Sistema Pastry bastando ter instalando o software apropriado, o sistema Pastry utiliza o seguinte conjunto de operações para seus clientes:

Fileid: Ao inserir um arquivo o sistema faz o controle deste arquivo pelo: (nome, proprietário credenciais, k, arquivo), o qual k é o número de réplicas do arquivo, o fileid tem um identificador de 160 bits, que pode ser usado para identificar o arquivo.

Busca de arquivos: O sistema recupera uma cópia do arquivo através do fileid e busca sempre o nó mais próximo a ele.

3.2 Juxmen

Segundo (Antoniu et al., 2003), o Juxmen é uma plataforma de compartilhamento de dados, tem arquitetura hierárquica. Este é um software de compartilhamento de nível físico que consiste em fornecer memória para armazenamento de dados. Um grupo de clusters é um conjunto de nós, que são gerenciados por um nó chamado de gerente de cluster, este gerencia a

memória disponibilizada por seu grupo de folhas. Quando um nó simplesmente acessa os conteúdos disponibilizados é chamado de cliente.

Outro ponto importante do Juxmen é o dinamismo pois os grupos podem ser criados em tempo de execução, o mapeamento dos clientes é por id, o qual pode ser bloquear ou liberar um cliente por este campo.

3.2.1 Gerir os recursos de memória

Os gerentes de cluster são responsáveis por disponibilizar e informar a quantidade de memória disponível. Quando um cliente solicita uma quantidade de memória 1280 bytes o sistema aloca uma quantidade de 2048 potência de 2, para garantir que o arquivo possa ser salvo, sem apresentar problemas com a capacidade de armazenamento, para controlar a volatilidade da rede Juxmem o sistema está sempre republicando suas capacidades.

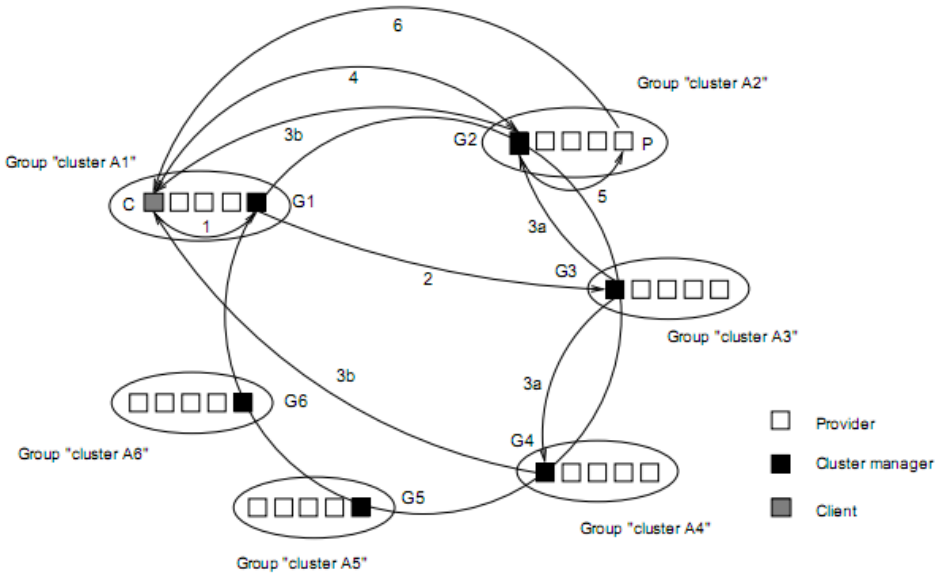

Figura 6 - Utilização do Juxmem - (Antoniu et al., 2003)

3.2.2 Manipulação de Prestadores Volátil.

A fim de garantir a persistência dos dados no sistema é parametrizado um número X de cópias, e o gerente de cluster é responsável por garantir a

redundância dos blocos de dados, se o número de cópias está abaixo do determinado o gerente de cluster procura um provedor para hospedar os dados. Para garantir a consistência entre os dados, quando é necessário fazer uma replica, o gerente de cluster bloqueia os dados, os quais só podem ser desbloqueados pelo provedor que foi escolhido para a cópia, para garantir que o arquivo seja salvo com sucesso o sistema utiliza um mecanismo de tempo limite (*timeout*) seguido por um teste ping que é responsável por verificar se o provedor ficou indisponível, se for o caso o gerente de cluster desbloquea os dados, para nova cópia.

No controle de Gestores de Clusters Voláteis o qual o sistema a fim de garantir a disponibilidade de um cluster, os gestores de cluster devem fazer sua replicação em gestores secundários garantindo assim a sobrevivência do cluster mesmo após o gestor cair. Esta comunição entre o gestor de cluster e o secundário é feita de forma periódica.

4 Materiais e Métodos

Este capítulo tem como objetivo apresentar a estrutura geral da pesquisa e do desenvolvimento do sistema de biblioteca digital.

4.1 Modelo da Pesquisa

Os materiais e métodos aplicados neste trabalho são fundamentados no estudo das arquiteturas P2P, que envolve: compartilhamento de arquivos, busca de arquivos, comunicação entre os pontos da rede, distribuição de réplicas de arquivos e persistência nos dados armazenados.

Este trabalho é definido como uma pesquisa aplicada, pois objetiva gerar conhecimentos, dirigidos à solução de problemas (Lakatos, 1993).

Será utilizada a pesquisa exploratória, pois, junto com os objetivos tende a proporcionar maior familiaridade ao tema, o qual visa apresentar as melhores soluções disponíveis para o sistema de biblioteca digital.

Também se realiza a pesquisa bibliográfica que do ponto de vista técnico tem como finalidade a pesquisa através de fontes primárias impressas ou digitais.

4.2 Processo de Desenvolvimento

As principais etapas deste trabalho foram:

Escolha do Tema: O tema é fruto do avanço das tecnologias e os meios digitais, com o aumento de informações armazenadas em conteúdos digitais, viu-se a necessidade de criar métodos que preservem estes conteúdos conforme descrito em (Eternal, 2005).

Revisão de Literatura: Na revisão de literatura foram abordadas as arquiteturas de sistemas distribuídos com maior ênfase em sistemas P2P,

clusters em sistemas P2P, compartilhamento de arquivos e distribuição de réplicas de dados em sistemas P2P.

Construção do Sistema: Nesta etapa foram avaliadas as arquiteturas para desenvolvimento do sistema de Biblioteca Digital, e consequentemente o desenvolvimento da aplicação.

4.3 Desenvolvimento do Sistema

Para o desenvolvimento do sistema de biblioteca Digital utilizou-se a linguagem Java, que é uma linguagem de programação orientada a objetos, desenvolvida na década de 90.

No desenvolvimento foram utilizadas algumas funções disponibilizadas pela linguagem Java como: Socket, Threads, XML além do Swing. Utilizou-se também a biblioteca FreePastry, que é uma biblioteca Open-Source (FreePastry, 2010), utilizada para desenvolvimento de aplicações P2P.

No processo de desenvolvimento foram utilizadas funções da linguagem Java com as funções da biblioteca FreePastry, unindo as duas para o desenvolvimento da aplicação de biblioteca digital.

A Figura 7 apresenta a tela do sistema, a qual demonstra as funções disponíveis no sistema como se conectar a rede, buscar um arquivo, baixar um ou enviar um arquivo para o sistema.

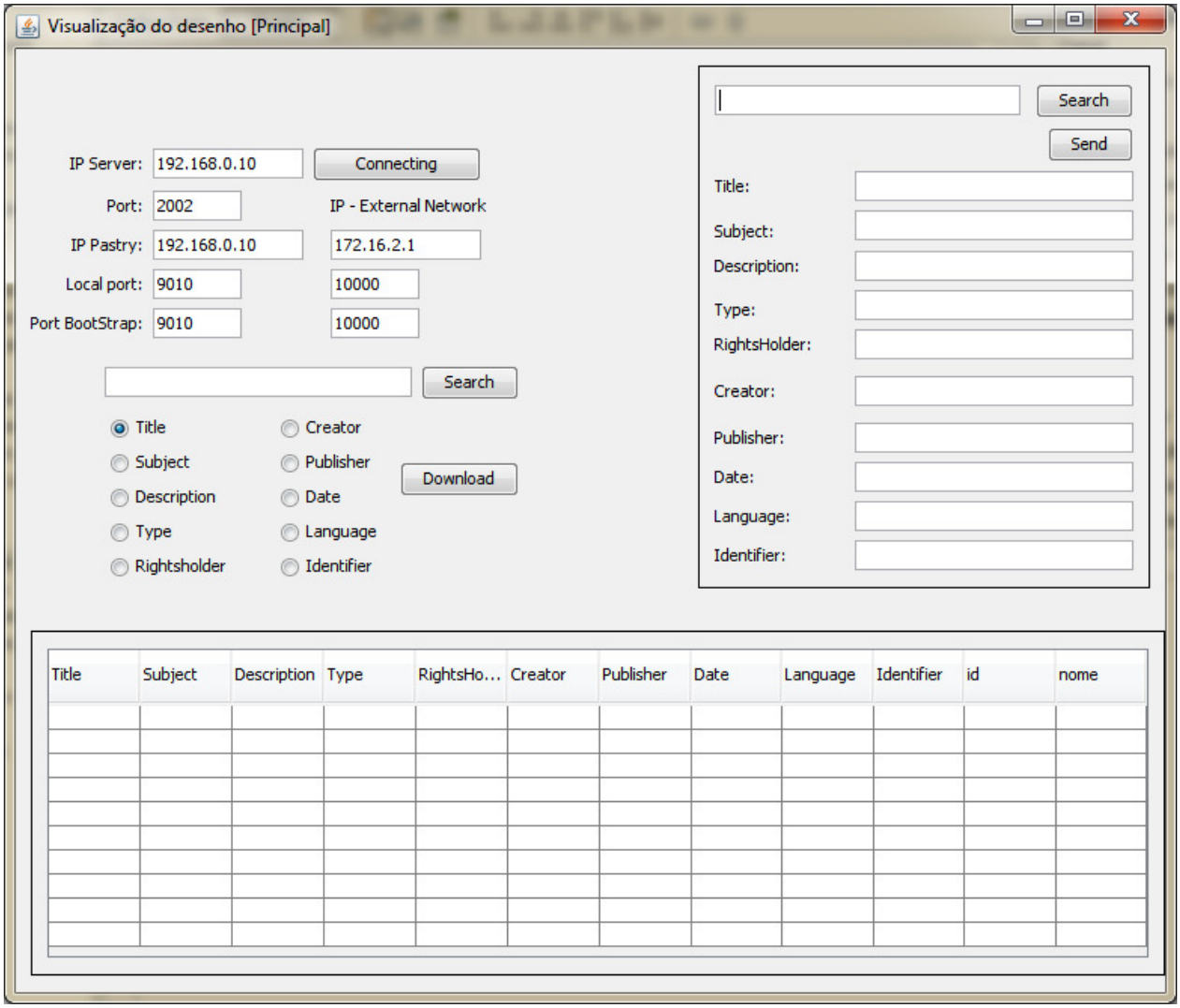

Figura 7 - Tela do Sistema

O presente programa visa solucionar alguns problemas encontrados com o armazenamento de dados em meios digitais como:

Distribuição dos Dados, neste caso encontrou como solução mais adequada a utilização da arquitetura P2P que fornece melhor abordagem para o sistema de biblioteca digital, pois, com este tipo de arquitetura a aplicação fica distribuída por diversos pontos na rede dando assim uma alta disponibilidade ao sistema.

Acesso aos dados, com a utilização da arquitetura P2P em conjunto com o super-peer, mostrou-se a mais adequada para a interligação dos nós com a

rede externa, pois, torna possível a comunicação dos nós registrados em uma rede localizar e baixar dados publicados em outra rede.

Persistência dos dados, com a ajuda da biblioteca FreePastry o qual é uma aplicação *Open-Source*, conseguimos solucionar o problema da persistência aos dados, pois, com esta biblioteca conseguimos disponibilizar réplicas dos arquivos distribuídas na DHT, mantendo assim uma grande persistência aos dados armazenados no sistema.

4.4 Catálogo de dados

O sistema possui um catálogo de dados, sendo este um arquivo XML contendo os dados dos arquivos publicados no sistema, conforme represendo no diagrama de classe na Figura 8. No momento da busca o sistema faz a consulta no catálogo retornando ao nó uma lista dos conteúdos encontrados, na inserção de um documento, é feito atualização no mantendo este sempre atualizado.

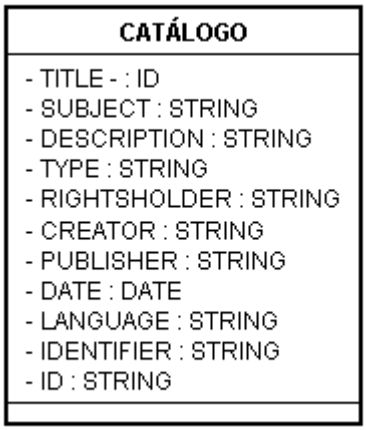

Figura 8 - Diagrama de Classe do Catálogo XML

O código abaixo representa a inserção de dados no catálogo de dados o qual é armazenado no super-peer.

```
public InserirXML (String nom,String tit,String subj, String descr,String typ,

String right,String creat, String publi, String dat, String lang, String ident, String nump,String idd ) throws ParserConfigurationException, SAXException, IOException, TransformerConfigurationException, TransformerException{
```

```
nome        = nom;
title       = tit;
subject     = subj;
description = descr;
type        = typ;
rightsholder=right;
creator     = creat;
publisher   = publi;
date        = dat;
language    = lang;
identifier  = ident;
numpartes   = nump;
id          = idd;
        DocumentBuilderFactory dbf =  DocumentBuilderFactory.newInstance();
        DocumentBuilder docBuilder = dbf.newDocumentBuilder();;
        Document doc = (Document) docBuilder.parse(new File("catalogo.xml"));
        Element raiz           = doc.getDocumentElement();
        Node nodo_artigo       = doc.createElement("artigo");
        Node nodo_nome         = doc.createElement("nome");
        Node nodo_title        = doc.createElement("title");
        Node nodo_subject      = doc.createElement("subject");
        Node nodo_description  = doc.createElement("description");
        Node nodo_type         = doc.createElement("type");
        Node nodo_rightsholder = doc.createElement("rightsholder");
        Node nodo_creator      = doc.createElement("creator");
        Node nodo_publisher    = doc.createElement("publisher");
        Node nodo_date         = doc.createElement("date");
        Node nodo_language     = doc.createElement("language");
        Node nodo_identifier   = doc.createElement("identifier");
        Node nodo_numpartes    = doc.createElement("numpartes");
        Node nodo_id           = doc.createElement("id");
        nodo_nome.setTextContent(nome);
        nodo_title.setTextContent(title);
        nodo_subject.setTextContent(subject);
```

```
        nodo_description.setTextContent(description);
        nodo_type.setTextContent(type);
        nodo_rightsholder.setTextContent(rightsholder);
        nodo_creator.setTextContent(creator);
        nodo_publisher.setTextContent(publisher);
        nodo_date.setTextContent(date);
        nodo_language.setTextContent(language);
        nodo_identifier.setTextContent(identifier);
        nodo_numpartes.setTextContent(numpartes);
        nodo_id.setTextContent(id);
        raiz.appendChild(nodo_artigo);
        nodo_artigo.appendChild(nodo_nome);
        nodo_artigo.appendChild(nodo_title);
        nodo_artigo.appendChild(nodo_subject);
        nodo_artigo.appendChild(nodo_description);
        nodo_artigo.appendChild(nodo_type);
        nodo_artigo.appendChild(nodo_rightsholder);
        nodo_artigo.appendChild(nodo_creator);
        nodo_artigo.appendChild(nodo_publisher);
        nodo_artigo.appendChild(nodo_date);
        nodo_artigo.appendChild(nodo_language);
        nodo_artigo.appendChild(nodo_identifier);
        nodo_artigo.appendChild(nodo_numpartes);
        nodo_artigo.appendChild(nodo_id);

        //grava o documento XML editado
        DOMSource source = new DOMSource(doc);
        StreamResult result = new StreamResult(new FileOutputStream("catalogo.xml"));
        TransformerFactory transFactory = TransformerFactory.newInstance();
        Transformer transformer = transFactory.newTransformer();
        transformer.transform(source, result);
}
```

4.5 Conexão do sistema.

A Figura 8 apresenta o diagrama de atividade que é proposto para o desenvolvimento da aplicação de biblioteca digital, no momento em que o nó inicia na rede, o sistema cria um ID do FreePastry, posteriormente executa uma busca por um super-peer para se conectar. No caso de existência deste super-peer o nó se junta a ele, em caso de não existência deste super-peer a aplicação o convida este nó a se tornar o super-peer.

No momento em que assumir o papel de super-peer ele deve criar dois anéis de comunicação um anel interno que é utilizado para a ligação de seus clientes e outro anel externo que é utilizado para a comunicação com outros super-peers e ligação com as demais redes externas.

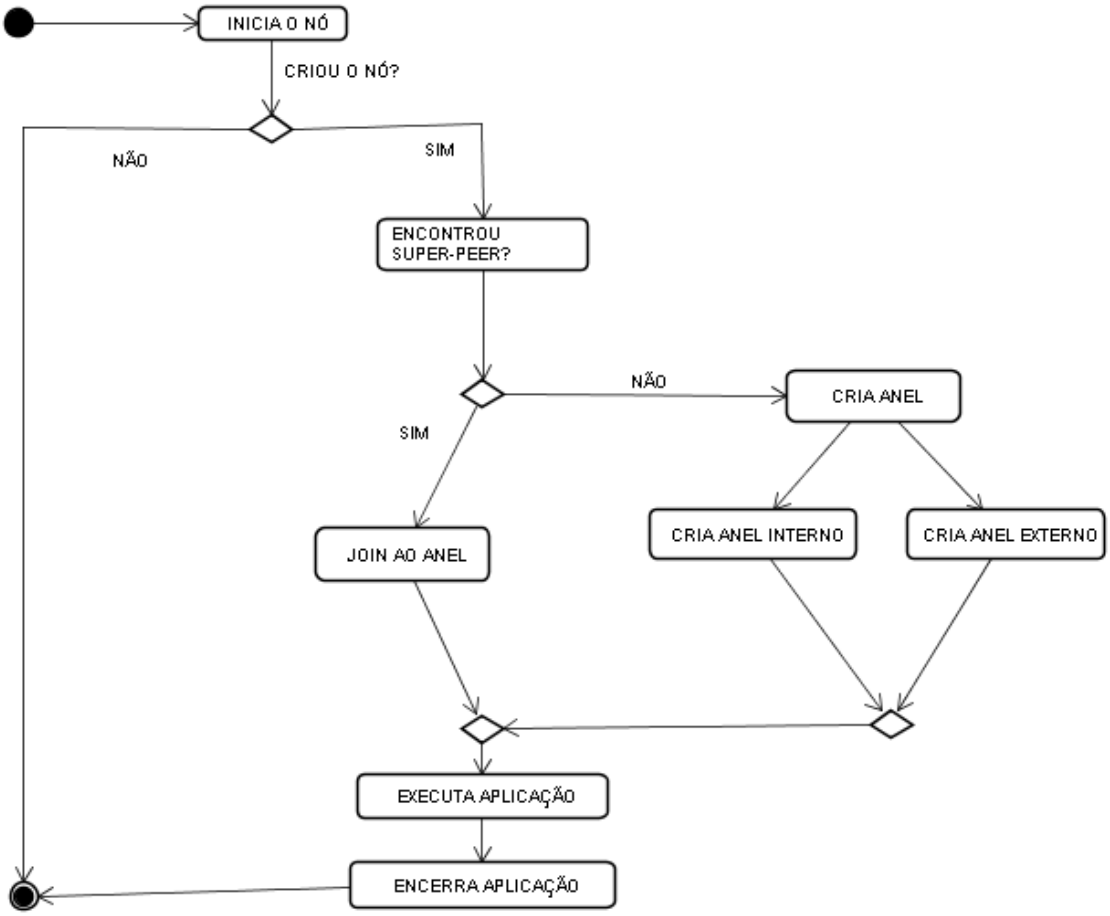

Figura 9 - Diagrama de atividade para conexão de um nó.

O código abaixo apresenta o código fonte de conexão do nó no sistema e a criação do nó Pastry.

```
public void Conectar(int bindport, InetSocketAddress bootaddress, final Environment env,InetAddress bootaddr,InetSocketAddress bootadd)throws Exception{

    localAddress = bootaddr;

    NodeIdFactory nidFactory = new RandomNodeIdFactory(env);

    PastryNodeFactory factory = new SocketPastryNodeFactory(nidFactory,localAddress,bindport, env);

    PastryNode node = factory.newNode();

    String storageDirectory = "c:/temp/storage/envio";

    Storage stor = new PersistentStorage(idf, storageDirectory, 4 * 1024 * 1024, node.getEnvironment());

    Past app = new PastImpl(node, new StorageManagerImpl(idf, stor, new LRUCache(

       new MemoryStorage(idf), 512 * 1024, node.getEnvironment())), 0, "");


    apps.add(app);

    node.boot(bootadd);

     synchronized(node) {

       while(!node.isReady() && !node.joinFailed()) {

         node.wait(500);

         if (node.joinFailed()) {

           throw new IOException("Could not join the FreePastry ring.  Reason:"+node.joinFailedReason());

         }

       }

     }

    env.getTimeSource().sleep(500);

    PastryIdFactory localFactory = new rice.pastry.commonapi.PastryIdFactory(env);

    no = node;

    Past p = (Past)apps.get(env.getRandomSource().nextInt(numNodes));

    app1 = p;

  }
```

4.6 Sistema de Busca

Para o sistema de busca de arquivo o qual é demonstrado no diagrama de atividade na Figura 9, o cliente solicita a busca por um arquivo, após confirmação o sistema envia a solicitação ao super-peer que irá executar a

pesquisa em seu nome, o super-peer por sua vez executa a busca em seu catálogo XML, caso encontre o arquivo retorna a busca ao usuário uma lista de documentos, caso não encontre o arquivo em seu catálogo irá pesquisar sobre o arquivo na rede externa, ou seja, em outros super-peers conectados a rede.

Caso encontre o arquivo em outro super-peer o sistema solicitará um *GET* do arquivo, assim que receba o arquivo executará um *PUT* do arquivo em sua DHT interna, após isto retornará a busca ao usuário, desta forma o arquivo ficará somente a um salto do nó que o solicitou proporcionando assim uma melhor utilização da rede e menor consumo da largura de banda. Caso não encontre o arquivo irá retornar *NULL*, para o cliente que fez a solicitação.

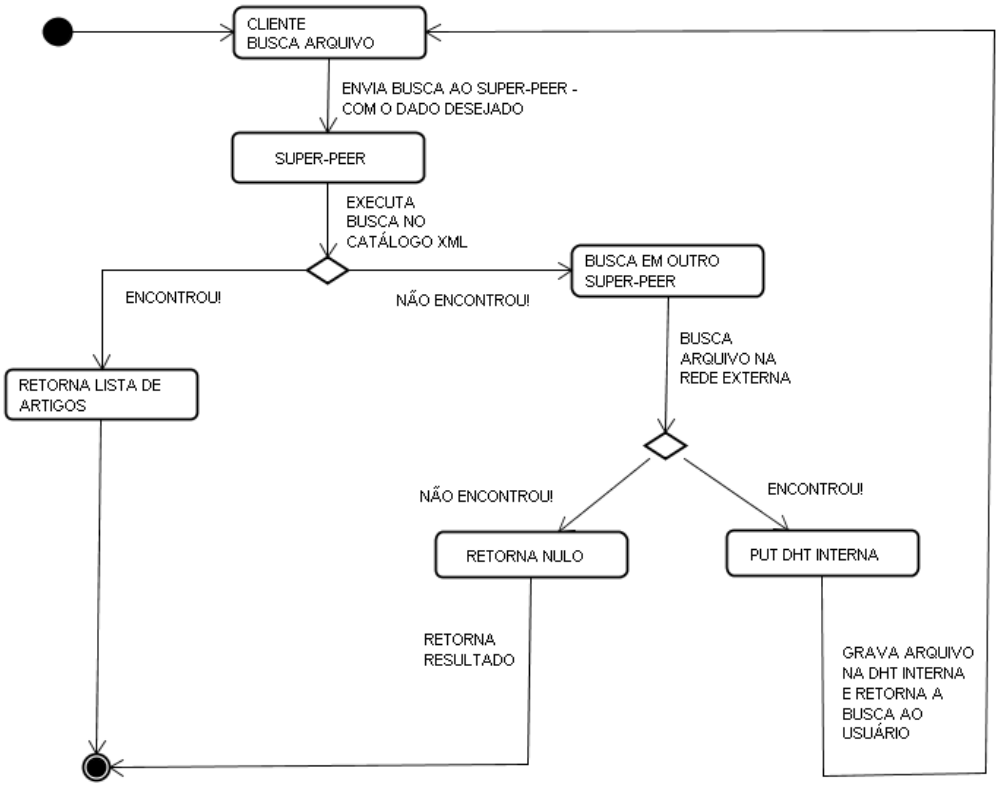

Figura 10 - Diagrama de atividade na busca de arquivo.

O código abaixo apresenta o sistema de busca de arquivos e a forma que o sistema executa a busca nas diferentes redes.

```
final Id idd = idf.buildIdFromToString(vetor[1]);
```

```
                System.out.println(idd.toStringFull());
                System.out.println("get");
                p1.getApp1i().lookup(idf.buildId(vetor[1]), new Continuation<PastContent, Exception>() {
                public void receiveResult(PastContent r) {
                   System.out.println("Baixado com Sucesso :  "+r+"    chave:  "+r.getId());
                   mMessage = senderIP + ":" + senderPort + " : " +"Baixado com sucesso!";
                        System.out.println("Baixado com Sucesso Rede Externa 2 :    chave:  "+r.getId());
                        PastryIdFactory localFactory = new rice.pastry.commonapi.PastryIdFactory(env);
                        final PastContent conteudo = new MyPastContent(idd,retorno.toString());
                        p1.getApp1e().insert(conteudo, (Continuation<Boolean[], Exception>) new
Continuation<Boolean[], Exception>() {
                        public void receiveResult(Boolean[] results) {
                           int numSuccessfulStores = 2;
                           for (int ctr = 0; ctr < results.length; ctr++) {
                           if (results[ctr].booleanValue())
                              numSuccessfulStores++;
                           }
                        System.out.println(" Salvo com sucesso no Servidor Rede EXterna " + numSuccessfulStores
+ " locations.");
                        try {
                           // wait 5 seconds
                           env.getTimeSource().sleep(5000);
                        } catch (InterruptedException ex) {
                           Logger.getLogger(Principal.class.getName()).log(Level.SEVERE, null, ex);
                        }

                          System.out.println("Enviando ao servidor 2!!!");
                        }//fim inserção
                        public void receiveException(Exception result) {
                           System.out.println("Error storing "+conteudo);
                           result.printStackTrace();
                        }
                     });
                }
```

```
public void receiveException(Exception e) {
   System.out.println("Arquivo não encontrado rede interna");
   p1.getApp1e().lookup(idf.buildId(vetor[1]), new Continuation<PastContent, Exception>() {
   public void receiveResult(PastContent r) {
      System.out.println("Baixado com Sucesso :  "+r+"    chave:  "+r.getId());
      mMessage = senderIP + ":" + senderPort + " : " +"Baixado com sucesso!";
            System.out.println("Baixado com Sucesso Rede Externa 2 :    chave:  "+r.getId());
            PastryIdFactory localFactory = new rice.pastry.commonapi.PastryIdFactory(env);
            final PastContent conteudo = new MyPastContent(idd,retorno.toString());

            p1.getApp1i().insert(conteudo, (Continuation<Boolean[], Exception>) new
Continuation<Boolean[], Exception>() {
            public void receiveResult(Boolean[] results) {
               int numSuccessfulStores = 2;
               for (int ctr = 0; ctr < results.length; ctr++) {
               if (results[ctr].booleanValue())
                  numSuccessfulStores++;
               }
            try {
               // wait 5 seconds
               env.getTimeSource().sleep(5000);
            } catch (InterruptedException ex) {
               Logger.getLogger(Principal.class.getName()).log(Level.SEVERE, null, ex);
            }
              System.out.println("Salvo com sucesso Rede Interna!!!");
            }//fim inserção
            public void receiveException(Exception result) {
               System.out.println("Error storing "+conteudo);
               result.printStackTrace();
            }
         });
   }
   public void receiveException(Exception e) {
      System.out.println("Arquivo não encontrado rede externa");
   }
```

```
        });
        }
    });
```

4.7 Envio de arquivos

A Figura 10 apresenta o diagrama de atividade para o envio de arquivos, no momento que o usuário salva um arquivo na aplicação, este executa um *PUT* na DHT interna e envia ao super-peer o ID do arquivo e dados do documento para que este salve os dados do documento em seu catálogo XML.

Em seguida o sistema executa um *GET* para busca do arquivo e em seguida executa um *PUT*, para salvar o arquivo na DHT externa, o qual pode definir parâmetro X que representa o número de réplicas que o arquivo deve conter a modo de garantir a persistência dos dados, se o arquivo for salvo com sucesso finaliza a aplicação e em caso de insucesso retorna ao super-peer para executar novamente.

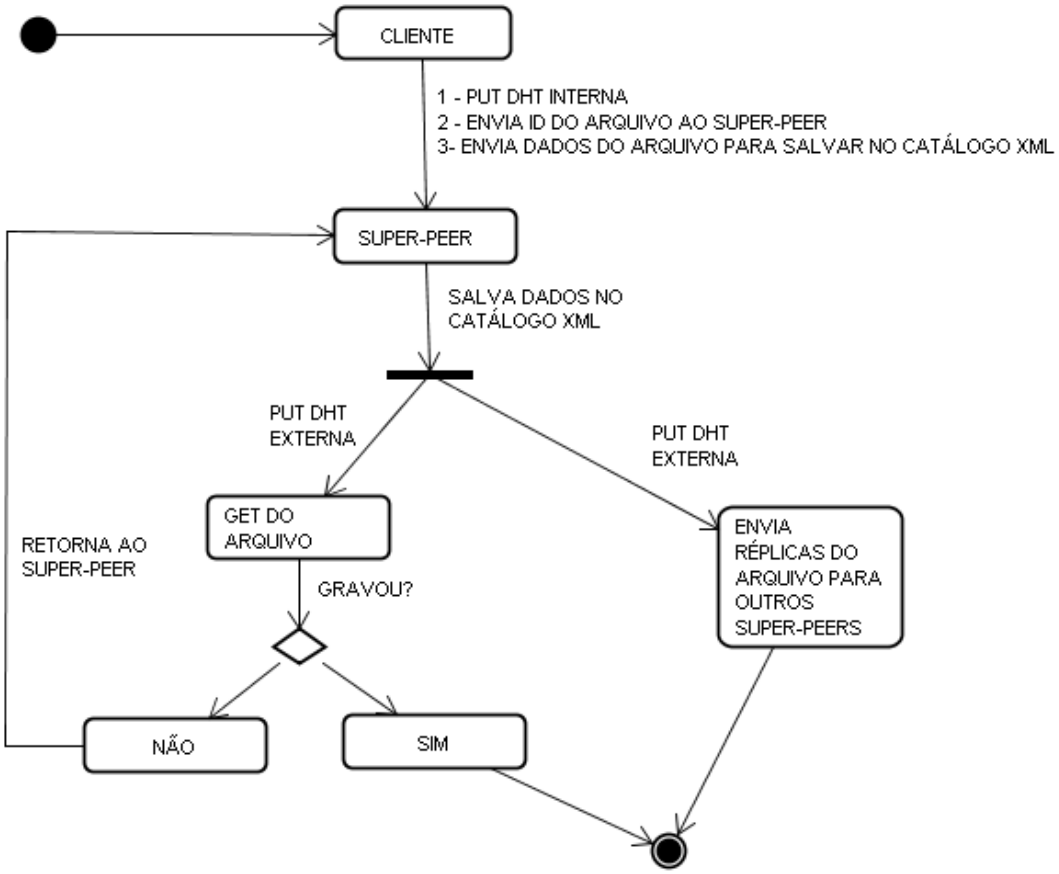

Figura 11 - Diagrama de atividade para o envio de arquivos.

O código fonte abaixo apresenta o envio de arquivo pelo usuário.

```
System.out.println(arquivo.toString());
File f = new File(arquivo.toString());
String fileContent=null;
InputStream is = new FileInputStream(f);
long length = f.length();
byte[] bytes = new byte[(int)length];
int offset = 0;
int numRead = 0;
while (offset < bytes.length && (numRead=is.read(bytes, offset, bytes.length-offset)) >= 0) {
    offset += numRead;
}
if (offset < bytes.length) {
    throw new IOException("Could not completely read file "+f.getName());
```

```
}
is.close();
fileContent = Base64.encodeBytes(bytes);

final PastContent myContent = new MyPastContent(idf.buildId(fileContent),fileContent);
chave = myContent.getId();
app1.insert(myContent, (Continuation<Boolean[], Exception>) new Continuation<Boolean[], Exception>() {
   public void receiveResult(Boolean[] results) {
      int numSuccessfulStores = 0;
      for (int ctr = 0; ctr < results.length; ctr++) {
      if (results[ctr].booleanValue())
         numSuccessfulStores++;
      }

      System.out.println(" Salvo com sucesso no cliente " + numSuccessfulStores + " locations.");
   try {
      // wait 5 seconds
      env.getTimeSource().sleep(5000);
   } catch (InterruptedException ex) {
      Logger.getLogger(Principal.class.getName()).log(Level.SEVERE, null, ex);
   }

     System.out.println("Enviando ao servidor!!!");
      try {
         out = new PrintWriter(new OutputStreamWriter(socket.getOutputStream()));
      } catch (IOException ex) {
         Logger.getLogger(Principal.class.getName()).log(Level.SEVERE, null, ex);
      }
      Envia sender = new Envia(out);
      String message = "1#"+arquivo.getName()+"$"+jTextField2.getText()+"$"+jTextField3.getText()+
            "$"+jTextField4.getText()+"$"+jTextField5.getText()+"$"+jTextField10.getText()+
            "$"+jTextField11.getText()+"$"+jTextField12.getText()+"$"+jTextField13.getText()+
            "$"+jTextField14.getText()+"$"+jTextField15.getText()+"$0$"+chave.toStringFull();
      sender.getmOut().println(message);
```

```
            sender.getmOut().flush();
            cmensa.setText("");
            System.out.println("Terminou!");
        }//fim inserção
        public void receiveException(Exception result) {
            System.out.println("Error storing "+myContent);
            result.printStackTrace();
        }
    });
```

4.8 Testes

Para comprovar os resultados da aplicação foram executados testes em três computadores, dois destes conectados em uma rede com faixa de IP 192.168.0.x, e outro conectado na rede 172.16.2.x.

Nos testes foram publicados diversos arquivos através dos computadores registrados nos IP's 192.168.0.100 e 192.168.0.10, e posteriormente foram publicados arquivos através do nó conectado com a o IP 172.16.2.1, para comprovar o funcionamento da aplicação, e a ligação entre as duas redes diferentes ou os dois anéis de comunicação.

Em seguida foi executado busca no sistema sobre os arquivos publicados pelos usuários conectados nas faixas de redes diferentes, o qual foi possível baixar os arquivos publicados entre estes usuários mesmo estes estando conectados em redes diferentes.

O super-peer por sua vez conseguiu retornar corretamente a lista de artigos a seus requisitantes, que conseqüentemente conseguiram baixar os arquivos.

E por fim para testar a disponibilidade dos dados contidos no sistema, foi desconectado os nós registrados em uma das redes, e executado a busca pelos arquivos publicados por estes, e posteriormente o *download* o que funcionou perfeitamente.

5 Conclusão

Devido à facilidade de se publicar dados em meios digitais, garantir a disponibilidade e a integridade destes dados se torna tarefa de grande prioridade.

Para isso a utilização do sistema de biblioteca digital e com seu método de distribuição de dados utilizando a arquitetura P2P e a biblioteca Freepastry, se torna um sistema de grande valia, pois, com ele garantimos maior disponibilidade dos dados armazenados no sistema, mantendo estes dados publicados na DHT.

Conforme teste executados conforme seção 4.8, constatou que o usuário que postou um dado no sistema, saia do sistema ou ocorra algum problema em sua conexão, o dado pode ser acessado por outros usuários, mantendo assim maior persistência e integridade nos dados contidos no sistema.

Como trabalho futuro propõe avaliar a criação de um cluster de super-peer dando assim segurança maior ao sistema, organizando estes de forma síncrona, mantendo os dois pontos sempre atualizados na troca dos catálogos de dados.

6 Referências

Antony Rowstron, Peter Druschel, (2001); Storage management and caching in PAST, a large-scale, persistent peer-to-peer storage utility; Microsoft Research, Rice University.

Arturo Crespo, Hector Garcia Molin, (2002); Routing Indices for peer-to-peer systems; Computer Science Department Stanford University Stanford.

Beverly Yang, Hector Garcia Molina (2002); Improving Search; Coputer Science Department, Stanford University.

Eternal Bits, IEEE Spectrum Julho de (2005).

FreePastryHomePage. https://trac.freepastry.org/; acessado em 01/10/2010.

G.Antoniu & Bougé & M.Jan, Juxmem (2003); An adaptive supportive Platform for data Sharing on the grid.

Gnutellawebsite. http://www.gnutella.com ; acessado em 10/10/2010.

George Coulouris, Jean Dollimore, Tim Kindberg; (2007) Sistemas Distribuídos, Conceitos e Projetos 4ª Edição; Editora ARTMED S/A.

Kamienski et al., 2005; Carlos Kamienski, Eduardo Souto, João Rocha, Marco Domingues, Arthur Callado, Djamel Sadok; Colaboração na Internet e a Tecnologia Peer-to-Peer.

Lakatos, Eva Maria; MARCONI, Marina de Andrade. Fundamentos de metodologia científica. São Paulo: Atlas, 1993.

MV Neves, 2006; Marcelo Veiga Neves; Utilizando JXTA para Distribuição e Compartilhamento de Arquivos em Redes Peer-to-Peer; Instituto de Informática Universidade Federal do Rio Grande do Sul (UFRGS).

NapsterHomePage. http://www.napster.com/; acessado em 10/10/2010.

Obelheiro e Fraga, 2007; Rafael R. Obelheiro e Joni da Silva Fraga; Uma Rede Overlay Tolerante a Intrusões: Arquitetura e Análise.

Rocha et al, 2004; João Rocha, Marco Domingues, Arthur Callado, Eduardo Souto, Guthemberg Silvestre, Carlos Kamienski, Djamel Sadok.

Vignatti 2009, Thiago Vignatti, Arquivamento Digital a Longo Prazo Baseado em Seleção de Repositórios em redes Peer-to-peer.

Yang e Garcia-Molina, 2001; BeverlyYang e HectorGarcia-Molina; Comparing Hybrid Peer-to-Peer Systems; Stanford University Stanford, CA, USA.

Yang e Garcia-Molina, 2003; BeverlyYang e HectorGarcia-Molina; Designinga Super-Peer Network; Computer Science Department, Stanford University.