Capítulo 1

Peer-to-Peer: Computação Colaborativa na Internet

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

Abstract

Peer-to-peer (P2P) computing has been promoting a substantial change in the usage patterns of the Internet in the last years. Its most important advantage, compared to client/server computing, is making the direct collaboration among users possible, with no need for intervenient third-party administered servers. P2P networks allow users to share computing resources through the Internet, even for those hosts hidden behind firewalls and NATs. Actually, this apparently new term is the reintroduction of a model used at the very beginning of the Internet, which brings certain benefits but also various problems. This document surveys the main issues related to P2P technology, using the experience that has been gained by the P2P Work Group of the Brazilian Research Network (RNP).

Resumo

A computação peer-to-peer (P2P) tem promovido uma grande modificação nos padrões de uso da Internet nos últimos anos. Sua grande vantagem, em relação à computação cliente/servidor, é possibilitar a colaboração direta entre os usuários, sem depender de servidores administrados por terceiros. Redes P2P permitem que recursos computacionais sejam compartilhados pelos usuários da Internet, mesmo que as máquinas estejam escondidas atrás de firewalls e NATs. Na verdade, esse termo aparentemente inovador consiste na re-introdução de um modelo usado no início da Internet, que possui benefícios, mas também uma série de problemas. Este documento faz um levantamento das principais questões relacionadas à tecnologia P2P, utilizando como base as experiências vivenciadas pelo GT-P2P da RNP.

1.1. Introdução

A *World Wide Web* (Web) sempre teve como proposta principal promover a liberdade, que deve se traduzir no acesso irrestrito a todos os recursos da rede, de qualquer lugar e a qualquer hora. Apesar disso, a Web ainda está presa ao modelo cliente-servidor no qual servidores centralizados executam tarefas para clientes distribuídos, como PCs, *laptops* e telefones celulares. Ou seja, a maior parte das máquinas participam da Web apenas como coadjuvantes acessando recursos providos pela minoria.

A tecnologia *peer-to-peer* (P2P) surge para mudar o paradigma existente, à medida que não depende de uma organização central ou hierárquica, além de dispor aos seus integrantes as mesmas capacidades e responsabilidades [38]. Através dessa tecnologia qualquer dispositivo pode acessar diretamente os recursos de outro, sem nenhum controle centralizado.

A inexistência de um servidor central significa que é possível cooperar para a formação de uma rede P2P sem qualquer investimento adicional em *hardware* de alto desempenho para coordená-la. Outra vantagem é a possibilidade de agregar e utilizar a capacidade de processamento e armazenamento que fica subutilizada em máquinas ociosas. Além disso, a natureza descentralizada e distribuída dos sistemas P2P torna-os inerentemente robustos a certos tipos de falhas muito comuns em sistemas centralizados. Finalmente, o modelo P2P apresenta o benefício da escalabilidade, para tratar de crescimentos incontroláveis no número de usuários e equipamentos conectados, capacidade de rede, aplicações e capacidade de processamento.

A tecnologia P2P estimula as pessoas no momento que elas percebem que podem participar e fazer a diferença. Isso explica o sucesso de algumas aplicações como o ICQ, KaZaA e o Napster. Um efeito contrário, no entanto, pode ser percebido em provedores de serviço e gravadoras que se sentem extremamente prejudicados quando perdem o controle. Nesse contexto, o ambiente acadêmico se mostra ideal para realizar experimentos e se beneficiar dessa tecnologia, pois pode utilizar novas ferramentas que auxiliam a troca de informações sem despender recursos adicionais.

O objetivo principal deste documento é fornecer subsídios para a compreensão das arquiteturas de redes *peer-to-peer* (P2P), elucidar aspectos de aplicações já utilizadas, bem como analisar as ferramentas mais importantes de suporte à construção de aplicações colaborativas. Um aspecto importante é a utilização da experiência prática dos membros da equipe do Grupo de Trabalho e Computação Colaborativa da RNP (GT-P2P) [19], cujo objetivo é avaliar os benefícios da implantação de suporte a sistemas P2P na RNP e nas instituições conectadas, bem como o impacto da utilização de tais sistemas no desempenho da rede. As instituições conectadas à RNP poderão utilizar esse suporte para, por exemplo, disponibilizar aplicações educacionais e de pesquisa.

1.1.1. A Evolução dos Modelos de Rede na Internet

Desde o princípio da Internet, nos anos 60 e 70, à época denominada ARPANET, os computadores em rede utilizam um modelo de comunicação baseado na idéia de prestação de

serviço entre as máquinas, onde uma máquina presta um serviço e as outras máquinas podem ter acesso ao mesmo. Embora essa definição se assemelhe com o modelo cliente/servidor predominante hoje na Internet, no início não havia restrições para a prestação de serviços entre máquinas. Todas exerciam ao mesmo tempo a função de cliente e servidor. Em outras palavras, adotavam o modelo P2P. Aliás, ainda hoje a rede formada somente pelos roteadores no interior da Internet segue o modelo P2P. Todos transmitem e recebem informações, existindo uma igualdade de funções e características.

Aos poucos, a Internet foi perdendo essa característica, por motivos como especialização de funções, necessidade de robustez e alta disponibilidade, problemas de segurança e limitação de recursos computacionais ou capacidade dos enlaces de comunicação. Neste momento, o modelo predominante é aquele onde alguns poucos servidores superdimensionados prestam serviços a uma infinidade de clientes nas bordas da rede. Isso ocorre, mesmo que a grande maioria dos clientes tenha uma grande capacidade de processamento e armazenamento.

Ainda, no fim dos anos 70 e nos anos 80, muita pesquisa foi feita sobre o uso de sistemas distribuídos para aplicações específicas. As idéias de autonomia, transparência, execução concorrente, tolerância a falhas e heterogeneidade de plataformas foram organizadas no modelo distribuído e postas em prática, uso que foi consolidado na década seguinte.

No fim dos anos 90 apareceu o Napster (seção 1.2.2), aplicação de compartilhamento de arquivos que popularizou o uso de redes P2P, trazendo de volta o modelo original da Internet. No modelo P2P, as máquinas têm mais autonomia, e toda máquina faz o papel de cliente e o papel de servidor.

1.1.2. Definições

Há inúmeras definições sobre redes *peer-to-peer* na literatura e tentaremos condensar as mais aceitas.

Redes *peer-to-peer* são redes virtuais que funcionam na Internet com o objetivo de compartilhar recursos entre os participantes, sendo que por princípio não há diferenciação entre os participantes. O grupo de pesquisa sobre redes P2P da IRTF [22] as define como “o compartilhamento de recursos e serviços computacionais diretamente entre sistemas”. Em geral, é aceito pela comunidade que sistemas P2P devem suportar os seguintes requisitos [37]:

- Nós podem estar localizados nas bordas da rede;
- Nós com conectividade variável ou temporária e endereços também variáveis;
- A capacidade de lidar com diferentes taxas de transmissão entre nós;
- Nós com autonomia parcial ou total em relação a um servidor centralizado;
- Assegurar que os nós possuem capacidades iguais de fornecer e consumir recursos de seus *peers*;
- A rede deve ser escalável;

- A capacidade dos nós se comunicarem diretamente uns com os outros.

Tendo todas estas características, uma rede pode ser dita *peer-to-peer*, mesmo que algumas das funções de controle da rede estejam localizadas em um servidor central (ponto de falha).

1.1.3. Redes P2P versus Redes Overlay

Uma rede *overlay* é uma rede "virtual" criada sobre uma rede existente, por exemplo: a Internet, com infra-estrutura IP. A rede *overlay* cria uma arquitetura com nível mais alto de abstração, de modo a poder solucionar vários problemas que, em geral, são difíceis de serem tratados ao nível dos roteadores da rede subjacente [1]. A própria Internet pratica o paradigma *overlay* quando usa o protocolo IP como solução de *interworking* sobre tecnologias de redes diversas como ATM, Frame Relay, PSTN, LANs, etc.

Numa perspectiva de alto nível, uma rede P2P pode ser considerada uma rede *overlay*, uma vez que funciona como uma rede virtual, formada pela interconexão dos nós (*peers*), executando sobre a infra-estrutura de uma rede física (Figura 1).

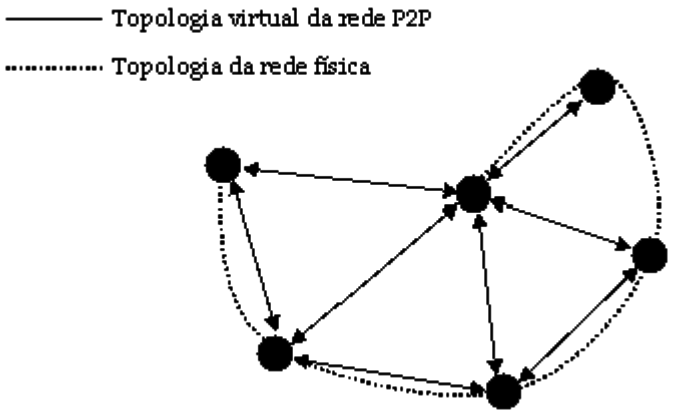

Figura 1: Rede *overlay*.

A característica básica de uma rede P2P é que existe um grupo de nós com interesses comuns que estão conectados através do mesmo sistema de comunicação. Outras características dessa rede são:

- Os nós são conectados de forma aleatória, não há restrição sobre o número de nós que participam da rede.
- A conexão de um nó à rede se estabelece através de outro nó que já pertença à rede.
- Os nós podem se unir e sair da rede a qualquer momento sem prévio conhecimento dos demais membros.

1.1.4. Modelos de Arquitetura P2P

Existem diferentes categorizações dos modelos de arquitetura P2P indicados na literatura, das quais três são apresentadas nesta seção. Na primeira categorização [37] são considerados dois modelos de rede P2P:

- Descentralizado – uma rede onde não há um ponto central e cada nó tem o mesmo nível. Neste modelo todos os nós são autônomos e todos os nós são responsáveis por troca de recursos e por controle (gerenciamento) de recursos. Os nós podem se comunicar de maneira direta ou através de vizinhos comuns (melhor escala de informações de controle, pior escala de tráfego de rede);
- Semicentralizado – uma rede onde há diferença de relevância entre os nós, havendo um nó central para informações de controle (e, possivelmente, para tráfego de dados) ou um conjunto de supernós que assume tais funções (onde a queda de um supernó afeta apenas os nós inferiores ligados a ele). Os nós inferiores podem ter maior ou menor nível de autonomia, mas são equivalentes entre si.

Na segunda categorização [49], também muito utilizada, existem três tipos de arquitetura de rede P2P:

- Busca centralizada – rede com um ponto central (possivelmente espelhado para outros pontos, dando a impressão de serem vários) de busca e nós que consultam o ponto central para trocar informações diretamente entre os *peers*;
- Busca por inundação – rede com nós totalmente independentes, onde normalmente a busca é limitada à vizinhança mais próxima do nó que fez a busca (assim, a busca é escalável, mas não é completa);
- Busca por tabela *hash* distribuída (DHT) – rede onde os nós têm autonomia e usam uma tabela *hash* para separar o espaço de busca entre eles.

Uma terceira classificação [29], que está sendo utilizada principalmente no mundo acadêmico nos últimos anos, divide as arquiteturas P2P em:

- Centralizada - rede que mantém um índice central com informações atualizadas (similar à busca centralizada da classificação anterior). Sistemas de compartilhamento de arquivos como Napster (seção 1.2.2.1) e de troca de mensagens (seção 1.2.1) utilizam esta arquitetura.
- Descentralizada e Estruturada: rede que não possui um servidor centralizado de diretório de informações, mas que tem uma estruturação significativa entre os nós. A topologia da rede é controlada e os documentos/arquivos são posicionados em locais que posteriormente tornam fácil a sua localização. Esse tipo de arquitetura em geral utiliza busca baseada em DHT. Essa arquitetura é utilizada pelos sistemas Chord, Pastry, Tapestry e CAN (seção 1.4.1).
- Descentralizada e não Estruturada: rede que não possui servidor centralizado, nem controle preciso sobre a topologia e localização/busca de documentos. Compreende os dois tipos (descentralizado e semicentralizado) da primeira classificação e a busca por inundação da segunda. Exemplos da utilização desta arquitetura são as redes Gnutella e KaZaA (seção 1.2.2.2).

1.2. Aplicações

Esta seção descreve em maiores detalhes as principais classes de aplicações P2P: mensagem instantânea, compartilhamento de arquivo, computação distribuída e trabalhos colaborativos. Enquanto essa seção apresenta as categorias de sistemas P2P, a subseção 1.4.3 mostra o funcionamento das principais redes que comportam tais aplicações.

1.2.1. Troca de Mensagens

A possibilidade de poder observar as pessoas entrando na rede e enviar uma mensagem em tempo real, tem tornado as aplicações de mensagem instantânea (IM – *Instant Messaging*) uma das mais populares da Internet.

Diferentemente do correio eletrônico, em que uma mensagem é armazenada em uma caixa postal e posteriormente entregue ao usuário que verificou a caixa postal no seu servidor, os sistemas IM fornecem entrega imediata ao usuário. Se o usuário não está disponível, a mensagem pode ser armazenada até que o mesmo se torne *"on-line"*, ou ela pode ser simplesmente descartada. Para evitar esta incerteza na entrega, os sistemas IM fornecem uma lista de contatos com um mecanismo capaz de identificar um usuário e determinar o seu estado, por exemplo, ativo, inativo ou ocupado.

Devido à popularidade das mensagens instantâneas, não é surpresa que exista uma variedade de aplicações IM. Algumas dessas soluções são apresentadas a seguir:

1.2.1.1. AIM

O AIM (AOL Instant Messenger - http://www.aim.com) é um dos mais populares programas de mensagem instantânea. Além das características comuns a todas aplicações IM, o AIM possui um recurso que notifica o usuário das últimas notícias, cotações de bolsas, esporte e outras informações. Um outro recurso interessante é a integração do AIM com o correio eletrônico. Esta integração permite que o usuário tenha acesso através do seu navegador a qualquer uma de suas contas da AOL.

1.2.1.2. MSN Messenger

O MSN Messenger da Microsoft (http://messenger.msn.com) possui a funcionalidade de sincronização de vídeo e voz para oferecer comunicação em tempo real. Além disso, os usuários podem tirar vantagem da funcionalidade de uma câmera WEB integrada, que permite mostrar imagens ao vivo para a lista de amigos. As características de voz também estão disponíveis, possibilitando aos usuários conversar entre si usando o microfone em seu computador.

1.2.1.3. Yahoo! Messenger

Uma característica interessante do Yahoo! Messenger (http://messenger.yahoo.com) é a sua boa integração com os serviços e conteúdos do Yahoo!. Além disso, o usuário pode receber e enviar mensagens através do seu telefone celular, ser alertado instantaneamente quando há novas mensagens em seu Yahoo! Mail ou quando está na hora de um de seus compromissos na Yahoo! Agenda. Um exemplo da interface do usuário do Yahoo! Messenger é apresentado na Figura 2.

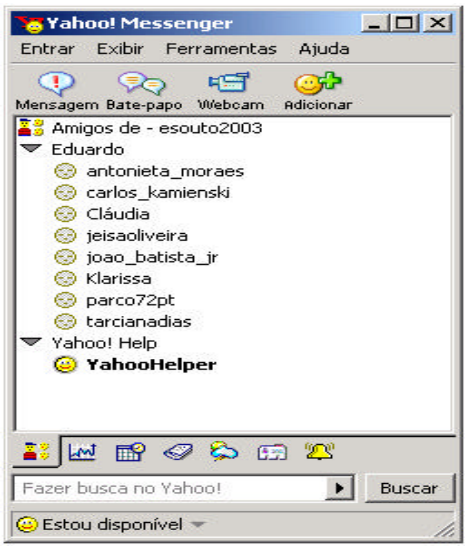

Figura 2: Cliente IM Yahoo! Messenger

Outras aplicações IM são:

- ICQ – (http://web.icq.com)
- JabberIM – (http://www.jabber.com)
- Messaging Architects – (http://www.gwtools.com)
- Omnipod – (http://www.omnipod.com)
- Sametime – (http://www.lotus.com/products/lotussametime.nsf/wdocs/homepage)
- Trillian Messenger – (http://www.trilian.cc)
- Vayusphere – (http://www.vayusphere.com)

1.2.2. Compartilhamento de Arquivos

A troca e o armazenamento de conteúdo é uma das áreas onde a tecnologia P2P tem alcançado maior sucesso. Várias Aplicações têm sido usadas por usuários da Internet para burlar as limitações de largura de banda dos servidores, que em geral, impedem a transferência de arquivos grandes. Os sistemas de armazenamento distribuído baseados na tecnologia P2P utilizam as vantagens da infra-estrutura existente para oferecer as seguintes características:

- Áreas de armazenamento – sistemas como Freenet (http://freenet.sourceforge.net) fornecem ao usuário uma área potencialmente ilimitada para o armazenamento de arquivos;
- Alta disponibilidade do conteúdo armazenado – para garantir alta disponibilidade dos arquivos armazenados, alguns sistemas adotam uma política de replicação múltipla do conteúdo, ou seja, um arquivo pode ser armazenado em mais de um nó na rede, ex: Chord [46];
- Anonimato – alguns sistemas P2P como Publius [52] garantem matematicamente que o documento publicado será mantido no anonimato de seus autores e publicitários;

- Gerenciamento – a maioria dos sistemas P2P fornece mecanismos eficientes de localização e recuperação dos conteúdos armazenados na rede.

As principais questões técnicas em sistemas de compartilhamento de arquivo são o consumo de largura de banda da rede, a segurança e sua capacidade de pesquisa [33].

1.2.2.1. Napster

O Napster (http://www.napster.com) é basicamente uma máquina de busca dedicada a encontrar arquivos MP3. Um servidor central é utilizado para armazenar uma lista com as músicas disponibilizadas pelos usuários e onde elas estão localizadas. O programa cliente Napster, instalado no computador dos usuários, faz uma consulta ao servidor Napster para obter informações sobre o arquivo desejado. O servidor Napster responde se existe o arquivo desejado e onde ele está localizado. Caso exista, uma conexão direta é estabelecida com o computador onde o arquivo está armazenado para que seja efetuado o *download*. Essa arquitetura é ilustrada pela Figura 3.

Existe uma variedade de aplicações P2P semelhantes ao Napster no que se refere a funcionalidade e algumas das mais populares são: Dmusic (http://www.dmusic.com), Audiogalaxy (http://www.audiogalaxy.com) MyNapster (http://www.mynapster.com) e Wippit (www.wippit.com).

Figura 3: Arquitetura do Napster

1.2.2.2. Gnutella

O Gnutella (http://www.gnutella.com) é considerado a primeira solução puramente P2P. Diferentemente do Napster, não existe um servidor central que armazena informações sobre os arquivos que estão disponíveis na rede. Em vez disso, todas as máquinas da rede informam sobre os seus arquivos disponíveis usando um mecanismo de inundação e utilizam uma abordagem de busca distribuída para recuperá-los. A seção 1.4.3 apresenta informações mais detalhadas sobre o Gnutella.

Existem muitas aplicações cliente disponíveis para acessar a rede Gnutella, algumas das mais populares são: BearShare (http://www.bearshare.com), Gnucleus (http://www.gnucleus.com) LimeWire (http://www.limewire.com), WinMX (http://winmx.com) e XoloX (http://www.holox.com).

1.2.2.3. Freenet

O Freenet é um software livre que permite que seus usuários publiquem e obtenham informações na Internet. Para alcançar esta liberdade, a rede é completamente descentralizada e os usuários são mantidos no anonimato. As comunicações dos usuários são sempre cifradas, o que torna extremamente difícil determinar quem está solicitando a informação e qual é o seu conteúdo.

Os usuários contribuem com a rede fornecendo largura de banda e espaço para armazenamento dos dados. Ao contrário das outras soluções, o Freenet não deixa o usuário controlar o que é armazenado no espaço de dados. Ao invés disso, os arquivos são mantidos ou excluídos de acordo com a freqüência de acessos, os arquivos acessados com menor freqüência são descartados. Isso ajuda a manter o conteúdo do espaço de dados sempre atualizado. A seção 1.4.3 apresenta informações mais detalhadas sobre o Freenet.

1.2.2.4. KaZaA

O KaZaA (http://www.kazaa.com) é o sistema de compartilhamento de arquivos mais utilizado na Internet. Ele utiliza o conceito de supernós para melhorar o desempenho da rede. Um supernó é uma máquina que participa da rede (nó) e que possui um alto poder computacional e rápidas conexões com a Internet. Os supernós mantêm uma lista contendo os arquivos disponibilizados por outros usuários e o local onde eles estão armazenados. Quando uma busca é executada, a aplicação KaZaA procura primeiro no supernó mais próximo do usuário que iniciou a consulta, retorna um conjunto de respostas para o usuário e encaminha a consulta para outros supernós. Uma vez localizado o usuário que possui o arquivo, uma conexão é estabelecida diretamente entre os *peers* para que seja efetuado o *download*.

1.2.2.5. Outras Aplicações

Outras aplicações P2P de compartilhamento de arquivos são:

- AudioGnome (www.audiognome.com)
- Badblue (www.badblue.com)
- eDonkey (www.edonkey.com)
- eMule (www.emule-project.net)
- Filetopia (www.filetopia.org)
- Frost (jtcfrost.sourceforge.net)
- Grokster (www.grokster.com)
- Imesh (www.imesh.com)
- Jungle Monkey (www.junglemonkey.net)
- Napigator (www.napigator.com)
- Project Elf (www.projectelf.com)
- Shareaza (www.shareaza.com)
- Snarfzilla (snarfzilla.sourceforge.net)
- Spinfrenzy (www.spinfrenzy.com).
- Splooge (www.splooge.com)

1.2.3. Computação Distribuída

A idéia de usar os recursos computacionais ociosos não é nova. O projeto *Beowulf* [4] da NASA mostrou que o alto desempenho computacional pode ser obtido usando um conjunto de máquinas. Outros esforços como o MOSIX [2] e Condor [27] também têm feito uso da computação distribuída delegando as tarefas computacionais entre as máquinas ociosas.

Grade computacional (*grid computing*) é um outro conceito que foi primeiramente explorado em 1995 pelo projeto I-WAY [14], em que foram utilizadas redes de alta velocidade para conectar recursos computacionais de diferentes localidades nos Estados Unidos. No final da década de 90, percebeu-se que a Internet com sua flexibilidade e crescente largura de banda fornece uma infra-estrutura que preenche bem os requisitos de computação distribuída. Um dos primeiros eventos visíveis de computação distribuída ocorreu em janeiro de 1999, quando o projeto *distributed.net* [http://www.distributed.net], contando com a ajuda de várias dezenas de milhares de computadores na internet, quebrou o algoritmo RSA em menos de 24 horas.

Recentes projetos têm estimulado o interesse de muitos usuários dentro da comunidade Internet. O projeto *SETI@home* (http://setiathome.ssl.berkeley.edu), por exemplo, possui um poder computacional de aproximadamente 25 TFlops/s (trilhões de operações de ponto flutuante por segundo), coletado de mais de três milhões de computadores conectados à Internet. Esse projeto visa utilizar essa grande capacidade de processamento para analisar os sinais obtidos a partir do rádio-telescópio do observatório de Arecibo a procura de algum sinal inteligente. A Figura 4 mostra a tela de proteção SETI@home que é ativada quando o seu computador está inativo.

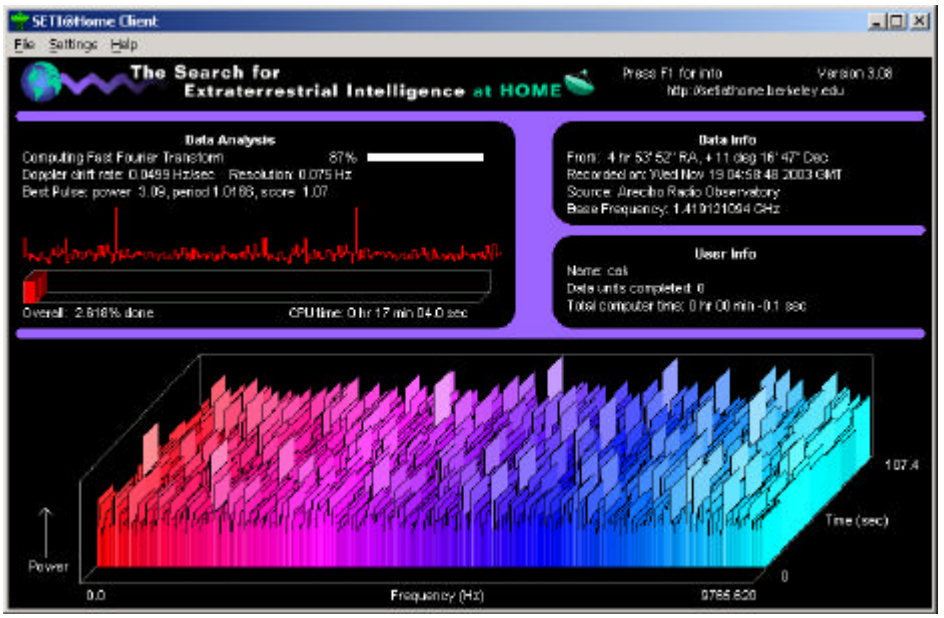

Figura 4: A aplicação SETI@home

De modo geral, o SETI@home trabalha da seguinte forma: o problema computacional a resolver é dividido em partes pequenas e independentes. O processamento de cada uma é feito em um computador individual na rede e os resultados são coletados por um servidor central. Este servidor central é responsável por distribuir as tarefas entre os computadores na Internet. Em cada computador registrado é instalado um software cliente que executa alguma computação requisitada pelo servidor. O processamento é realizado toda vez que o computador entra em período de inatividade, freqüentemente caracterizado pela ativação da proteção de tela. Após o término da computação, o resultado é retornado para o servidor, e uma nova tarefa é alocada para o cliente.

1.2.4. Trabalho Colaborativo

Os programas de trabalho colaborativo ou *groupware* são projetados para melhorar a produtividade de indivíduos com metas e interesses comuns. A expressão "*groupware*" é definida como um software que suporta colaboração, a comunicação e coordenação de vários usuários em uma rede. Isto inclui a integração de características como *e-mail*, calendário, espaços de trabalho, listas de discussão, sistemas de gerenciamento de documentos, vídeo conferência, entre outras.

Os sistemas *groupware* tradicionais como o Lotus Notes foram projetados para suportar o trabalho colaborativo em redes locais. A combinação das soluções *groupware* com a tecnologia P2P trouxe novas oportunidades. As atuais aplicações possibilitam a criação e extensão de grupos de trabalho espontaneamente, sem levar em consideração a localização. Algumas aplicações bem conhecidas são Microsoft NetMeeting (http://www.microsoft.com/windows/netmeeting/) e Groove (http://www.groove.net). Uma ilustração da interface do Groove é apresentada na Figura 5.

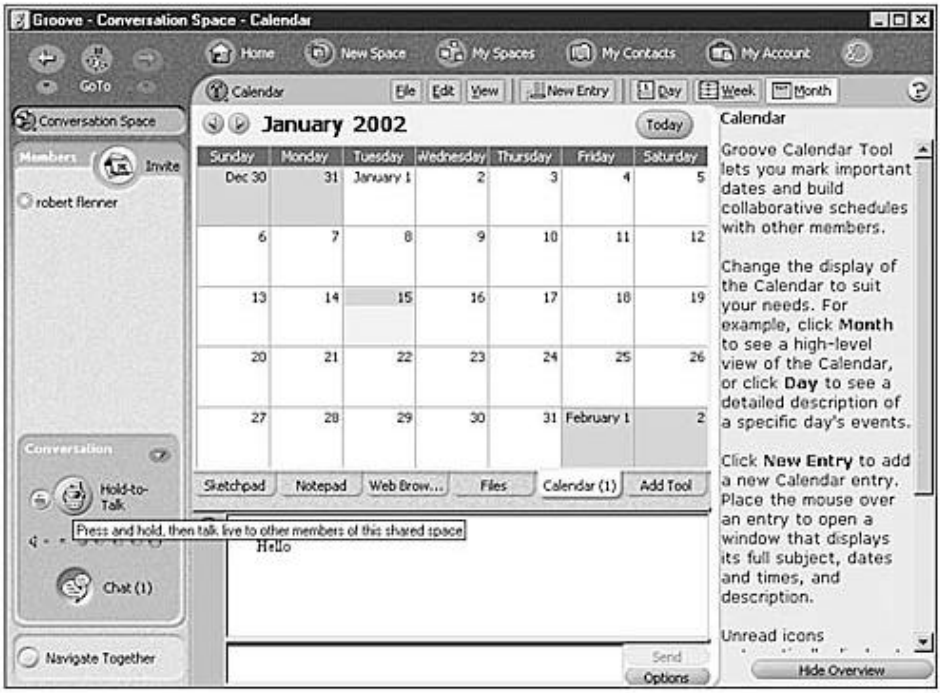

Figura 5: Colaboração P2P usando a rede Groove

O Groove usa espaços de trabalho que são criados e compartilhados entre os participantes. Ele usa um mecanismo cliente-servidor para encontrar os *peers* e iniciar os serviços. Após isso, toda a comunicação acontece diretamente entre os *peers* participantes.

Outras soluções colaborativas que estão se popularizando são:

- Ikimbo (http://www.ikimbo.com) - fornece mecanismos de mensagens instantâneas, conferência, detecção de presença, identificação, caixa de mensagem e transferência de arquivos.

- Consilient (http://www.consilient.com) - fornece um arcabouço para o desenvolvimento e gerenciamento do fluxo de processos de uma empresa.

1.2.5. Outras Aplicações

Conforme observamos nas subseções acima, as implementações P2P oferecem um melhor suporte às aplicações onde a centralização não é possível, a escalabilidade é desejável, o relacionamento entre os usuários é transiente e os recursos são altamente distribuídos. Assim, sua infra-estrutura descentralizada abre a possibilidade para o desenvolvimento de outras aplicações como as descritas abaixo:

- Blogs – algumas aplicações P2P permitem com que o usuário possa escrever diretamente em páginas WEB denominadas de "*blog*". Os usuários podem criar seus *blogs* de acordo com um tema. Por exemplo, um *blog* privativo para uma equipe de trabalho discutir projetos e apresentar soluções. Um sistema *blogger* é encontrado em (www.blogger.com.br).
- Jogos – A infra-estrutura P2P é comumente usada como base para as inúmeras comunidades de jogos *on-line*.
- Interconexão de dispositivos de rede – a união de tecnologias como Bluetooth, Jini e JXTA com o paradigma de comunicação ponto-a-ponto possibilita a formação de grupos de *peers* com diferentes dispositivos de rede, sejam eles fixos, móveis ou mesmo embutidos.

1.3. Questões de Projeto

Essa seção apresenta as principais questões que devem ser consideras no projeto de sistemas P2P.

1.3.1. Endereçamento

Aplicações inerentemente P2P necessitam de conexões diretas entre os *peers*. Entretanto um problema preponderante a ser tratado no projeto desses sistemas são as barreiras de proteção dispostas nas redes, que impedem a comunicação direta entre os *peers*. Os tipos de proteção mais utilizados são os *firewalls, servidores proxy e NAT – Network Address Translator* [3].

- *Firewalls* – são utilizados para filtragem de protocolos e portas específicas, geralmente permitem o fluxo de mensagens do protocolo http. Atualmente, além de proteger grandes empresas, os *firewalls* também estão disponíveis para usuários de computadores pessoais;
- Servidores *Proxy* – o serviço *proxy* é configurado em um equipamento entre a Internet pública e a rede local e provê serviços de filtragem de *sites* indesejáveis, *caching* e monitoramento do tráfego;
- NAT – são empregados para permitir que usuários utilizem poucos endereços IP válidos para uma grande rede de computadores. O NAT é configurado entre a Internet pública e rede local, sua principal função é reescrever um endereço IP válido

e o número das portas nos cabeçalhos IP que ele manipula de tal maneira que os pacotes pareçam ter sido enviados pelo equipamento que implementa o NAT. Também constrói uma tabela relativa a cada solicitação das máquinas da rede interna para realizar o encaminhamento de mensagens que saem e que entram na rede. Freqüentemente, por razões de segurança, os NATs permitem apenas tráfego a partir de um IP externo para máquinas da rede interna, se já tiver sido gerado tráfego a partir de uma máquina na rede interna para o endereço IP externo específico.

Uma solução para transpor essas barreiras pode ser implementada utilizando o protocolo HTTP: *HTTP tunneling*. Esse protocolo permite que um *web browser* inicie um pedido em um servidor WEB através da porta de comunicação 80. Isso efetivamente não se traduz em ameaça, uma vez que aplicações externas não podem iniciar ações danosas nas máquinas da rede interna, pois em nenhum momento o servidor WEB inicia uma conexão com o *web browser* [33]. Por isso, engenheiros de segurança normalmente permitem que essa porta não seja barrada pelos *firewalls*. Essa solução também é aplicada para redes que utilizam servidores *proxy* e NAT.

Um problema com a estratégia *HTTP tunneling* é que a comunicação não mantém estado. As respostas podem ser associadas aos pedidos, mas o contexto entre cada ciclo pedido/resposta anterior é perdido. Outro problema com *HTTP tunneling* está relacionado com a necessidade de iniciar uma mensagem ou passar dados para um cliente "protegido" por *firewall*. Como foi dito, o mundo exterior não pode iniciar uma comunicação para fornecer uma mensagem, a única alternativa é esperar para que o cliente entre em contato com o servidor e retire a mensagem. Isso pode aumentar o número de mensagens e conseqüentemente a latência. Portanto, clientes em redes cobertas por *firewalls* têm a desvantagem de não poderem atuar com *peers* servidores.

A utilização de *peers* para retransmissão agindo como servidores http, com razoável capacidade de enlace e posicionados em lugares estratégicos na rede, pode isolar o fluxo de mensagens provenientes dos *peers* "protegidos", melhorando o desempenho do *HTTP tunneling* [37].

1.3.2. Conectividade

A natureza das conexões dos *peers* de uma rede tem grande relevância no projeto de sistemas P2P. A questão tem dois lados: As conexões da maioria dos *peers* têm baixa capacidade de enlace, e ao mesmo tempo existe uma variedade muito grande de tipos de conexões – *dial-up*, banda larga (*cable* modem, isdn, etc.), redes acadêmicas e corporativas. Outra importante característica dos enlaces é a assimetria no provisionamento das bandas para *upload* e *download*. Essas particularidades requerem que aplicações considerem a heterogeneidade das conexões dos *peers* um requisito muito importante no projeto de aplicações P2P.

Os sistemas P2P normalmente crescem em quantidade de recursos disponíveis à medida que o número de usuários da rede também cresce. Assim, a escalabilidade de uma rede P2P é pelo menos linear. Porém, algumas redes são projetadas levando em consideração as baixas capacidades dos enlaces de grande parte dos usuários, o que conduz a um comportamento tipicamente cliente/servidor em detrimento do tráfego P2P. Assim, há

indícios de que as redes P2P apresentam características *small world* com comportamentos de distribuições de Lei de Potência (*Power Law*) [24]. Em geral, existe uma preferência por *peers* bem estabelecidos, isto é, com alta capacidade de processamento e armazenamento, e conectados a enlaces dedicados de alta capacidade.

1.3.2.1. Small World

A primeira referência sobre o tema "small world" foi relatada antes mesmo do advento da Internet. Stanley Milgram buscava determinar se a maioria dos indivíduos em sociedade estava ligada por alguma rede de conhecimentos. Assim, foram recrutados voluntários para tentarem encaminhar uma carta para uma determinada pessoa "alvo", através de pessoas que eles conheciam baseadas apenas no primeiro nome. A conclusão da pesquisa mostrou que grande parte de pares de indivíduos estão unidas por um número médio de seis passos, o chamado *princípio dos seis graus de separação* [24].

Grande parte das técnicas relacionadas com sistemas P2P considera este fenômeno. Em redes como a Gnutella, selecionando os *peers* vizinhos que retornam o maior número de resultados, é assumido uma alta similaridade entre os vizinhos, de modo que uma pesquisa em assuntos de interesse comum pode ser atendida com maior eficiência. Isso se assemelha à sociedade moderna onde pessoas têm maior interesse em associar-se a grupos com interesses comuns. Como já foi dito anteriormente, a topologia da rede P2P tem características de uma distribuição *Power Law*, e buscas em redes com essa natureza são mais eficazes. Essa estrutura também reporta a vida em sociedade, onde as pessoas direcionam suas necessidades para pessoas com conexões sociais bem estabelecidas [54].

Nesse sentido, a capacidade de comunicação das redes é um dos principais recursos para compartilhamento em redes P2P. Freqüentemente, para os usuários finais, a preocupação com capacidade de enlace acontece apenas na conexão com o *gateway* de acesso ao ISP (*Internet Service Provider*). Contudo, a eficiência da infra-estrutura da rede é uma questão com grande importância para aplicações P2P, que normalmente movimentam um volume de dados muito grande. No Groove (seção 1.4.2.1), somente as modificações nos arquivos são enviadas, ou seja, a complexidade envolvida no registro das partes e arquivos modificados desde o último acesso não é visível ao usuário. O método para enviar essas modificações leva em consideração as características da rede e dependendo do tamanho das mensagens e da capacidade dos enlaces disponíveis, as modificações podem ser enviadas diretamente.

1.3.3. Escalabilidade

Um dos benefícios imediatos da descentralização das redes P2P é a melhor escalabilidade dos sistemas. A escalabilidade é limitada por fatores relacionados à quantidade de operações centralizadas, como coordenação e sincronização, os estados que precisam ser mantidos, o paralelismo inerente das aplicações, o modelo de programação utilizado na construção do sistema.

Em geral, sistemas P2P tendem a ter maior escalabilidade que sistemas que utilizam o modelo cliente-servidor. Em um sistema cliente-servidor, os servidores são os únicos responsáveis por toda a carga do sistema. O que ocorre muitas vezes é que em horários de

pico os servidores ficam sobrecarregados e o sistema como um todo tende a oferecer um serviço com baixa qualidade. Em um sistema P2P, quando o número de clientes na rede aumenta, cresce também o número de servidores, uma vez que todos podem atuar como clientes e servidores, aumentando na mesma proporção a quantidade de recursos compartilhados. Dessa forma, o sistema aumenta de tamanho sem perder escalabilidade [6].

O Napster trata o problema da escalabilidade permitindo que os *peers* realizem *download* de arquivos de músicas diretamente dos *peers* que possuem o arquivo solicitado. Desse modo, o Napster, chegou a suportar até 6 milhões de usuários no seu auge. Em comparação com o Napster, o sistema SETI@home, uma aplicação que utiliza o tempo ocioso das máquinas na rede para realização de um *grid* computacional para analisar dados coletados a partir de telescópios com o objetivo de encontrar formas de vida extraterrestre, tem como foco principal o paralelismo da computação distribuída, e já atingiu 3,5 milhões de usuários. Obviamente o apelo do objeto tratado pelas aplicações é um fator a ser considerado na análise do volume de usuários.

A escalabilidade também depende do raio de comunicação (grau de conectividade) para computação entre os *peers* de um sistema P2P. Aplicações para quebra de código e para buscas de grandes números primos têm melhor desempenho quando o raio de comunicação é mais próximo de zero. Ainda citando o exemplo do SETI@home, o grande gargalo para escalabilidade está na transferência de dados.

Os sistemas P2P mais antigos como Gnutella e Freenet são *ad-hoc* em essência. Nesses sistemas, um *peer* simplesmente envia suas consultas para outros *peers* e estes repassam adiante para outros *peers*. Esse comportamento faz com que o tempo de resposta da consulta tenha características não-determinísticas.

Algoritmos mais recentes, dentre eles, CAN, Chord, Oceanstore e Pastry (ver item 1.4.1) implementam mapeamento consistente entre a chave do objeto e o *peer* hospedeiro, conseqüentemente um objeto pode ser recuperado tão logo os *peers* hospedeiros possam ser alcançados. *Peers* nesses sistemas compõem uma rede *overlay*. Cada *peer* mantém somente informações acerca de um pequeno número de outros nós no sistema. Essa característica limita a quantidade de estados do sistema que precisam ser mantidos, melhorando dessa forma a escalabilidade e proporcionando também balanceamento de carga. Por exemplo, Oceanstore foi projetado para permitir bilhões de usuários, milhões de *peers* servidores e mais de 10^{10} arquivos disponíveis na rede.

Portanto, gerar uma topologia estruturada e escalável é importante para sistemas P2P para poder prover latências menores e melhor desempenho na comunicação entre os *peers*.

1.3.4. Roteamento

Localizar informação numa rede volátil, de grande escala e altamente distribuída não é tarefa simples. Nessa seção serão descritos os principais mecanismos de roteamento existentes, ressaltando suas vantagens e desvantagens. Na seção 1.4.1, os principais algoritmos de roteamento são apresentados.

Algoritmos de busca e roteamento geralmente tentam otimizar o encaminhamento de uma mensagem de um *peer* para outro. Os três modelos mais comuns são (seção 1.1.4):

centralizado, inundação e tabela de *hash* distribuída (DHT). O modelo de inundação é também classificado como modelo "descentralizado não-estruturado", enquanto que o modelo DHT é chamado de modelo "descentralizado estruturado". Isso se deve ao mecanismo (estruturado ou não) de entrada de um nó na rede e de busca de informações.

1.3.4.1. Modelo Centralizado

Popularizado pelo Napster, esse modelo caracteriza-se pela conexão dos *peers* a um diretório central onde eles publicam informações sobre o conteúdo que oferecem para compartilhamento. Como ilustra a Figura 6, ao receber uma requisição, o índice (diretório) central escolhe o *peer* no diretório que for mais adequado. Esse *peer* pode ser o mais rápido e disponível, dependendo das necessidades do usuário. Então, a troca de arquivos será realizada diretamente entre os dois *peers*. Esse modelo requer uma infra-estrutura de gerenciamento (o servidor de diretórios), que armazena informações sobre todos os participantes da comunidade. Tal aspecto pode gerar limites de escalabilidade ao modelo, uma vez que requer servidores maiores quando o número de requisições aumenta, e mais espaço para armazenamento à medida que a quantidade de usuários cresce. Contudo, a experiência do Napster mostrou que, exceto por questões legais, esse modelo era bastante robusto e eficiente.

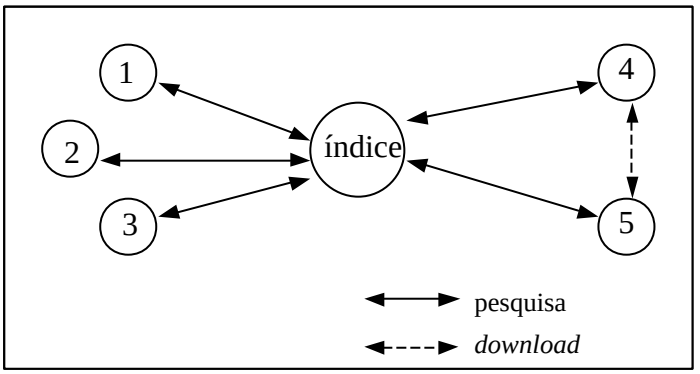

Figura 6: Modelo centralizado

1.3.4.2. Modelo de Inundação

O modelo de inundação de requisições é diferente do modelo de índice central, pois não se baseia na publicação dos recursos compartilhados. Ao invés disso, cada requisição de um *peer* é enviada para todos os *peers* diretamente conectados, os quais enviam para os *peers* diretamente conectados a eles, e assim sucessivamente até que a requisição seja respondida ou que ocorra o número máximo de encaminhamentos (tipicamente 5 a 9). Esse modelo (Figura 7) é utilizado pelo Gnutella e requer alta capacidade dos enlaces de comunicação para proporcionar desempenho razoável, apresentando problemas de escalabilidade quando o objetivo é alcançar todos os *peers* em uma rede. Contudo, o modelo é eficiente em comunidades limitadas e em redes corporativas.

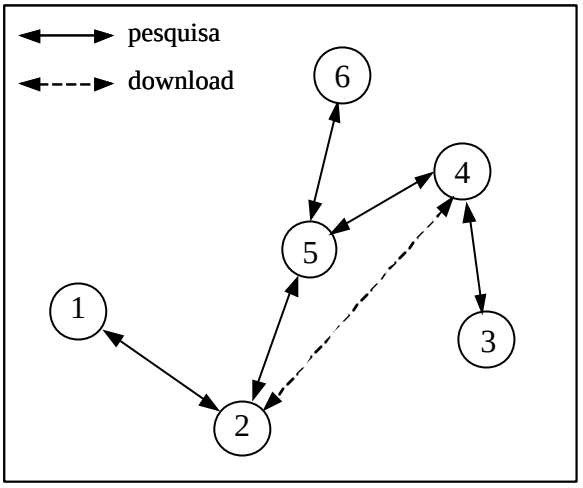

Figura 7: Modelo de inundação

Manipulando o número de conexões de cada nó e configurando apropriadamente o valor do parâmetro TTL (*time to live*) das mensagens de requisição, o modelo de inundação pode ser utilizado por centenas de milhares de nós. Além disso, algumas empresas têm desenvolvido softwares para clientes "supernós", os quais concentram várias requisições, alcançando bom desempenho com enlaces com capacidades de transmissão mais baixas a custo de alto consumo de CPU. O armazenamento de pesquisas recentes (*caching*) também é usado para melhorar a escalabilidade.

1.3.4.3. Modelo DHT

O modelo de tabelas de *hash* distribuídas (DHT) é o mais recente. Nesse modelo, um ID randômico é associado a cada *peer* da rede que conhecem um determinado número de *peers*, como pode ser visualizado na Figura 8. Quando um documento é publicado (compartilhado) em tal sistema, um ID é associado ao documento baseado em uma *hash* de conteúdo dos documentos e no seu nome. Cada *peer* então encaminha o documento ao *peer* cujo ID é mais próximo do ID do documento. Esse processo é repetido até que o ID do *peer* atual seja o mais próximo do ID do documento. Cada operação de roteamento também garante que uma cópia local do documento seja mantida. Quando um *peer* solicita o documento de um sistema P2P, a requisição irá até o *peer* com ID mais semelhante ao ID do documento. Esse processo continua até que uma cópia do documento seja encontrada. Então o documento é transferido ao *peer* que originou a requisição, enquanto cada *peer* que participou do roteamento permanecerá com uma cópia local do documento [46].

Apesar do modelo DHT ser eficiente para comunidades grandes e globais, ele apresenta um problema relacionado ao ID do documento. Este precisa ser conhecido antes que uma requisição do documento seja realizada. Assim, é mais difícil implementar uma pesquisa nesse modelo que no modelo de inundação. Além disso, pode ocorrer a formação de "ilhas", onde a comunidade se divide em sub-comunidades que não possuem nenhuma ligação (*links*) entre si.

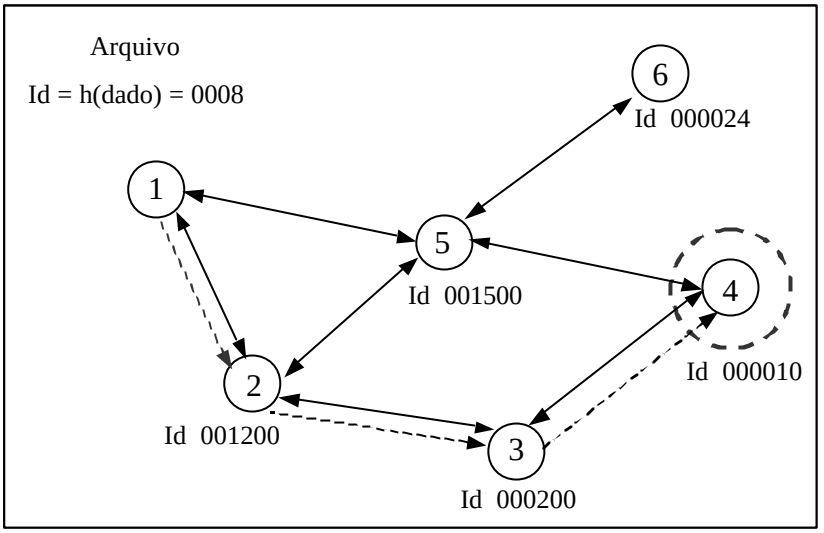

Figura 8: Modelo DHT

Existem quatro algoritmos principais que implementam esse modelo: Chord, CAN, Tapestry e Pastry, que são apresentados na seção 1.4.1.

1.4. Tecnologias e Soluções

Esta seção trata das principais soluções e tecnologias utilizadas na implementação de sistemas P2P. São abordadas as principais soluções de busca (os algoritmos de roteamento), as plataformas de desenvolvimento e as redes públicas mais utilizadas na Internet.

1.4.1. Algoritmos de Roteamento

A busca de informações em sistemas P2P depende dos algoritmos de roteamento. Em todos eles uma chave é fornecida como entrada e, como resposta, uma mensagem é roteada para o nó responsável pela chave. Cada nó mantém uma tabela de roteamento consistindo de um pequeno subconjunto de nós do sistema. Quando um nó recebe uma consulta de uma chave pela qual ele não é responsável, o nó roteia a consulta para algum nó vizinho que continua resolvendo a consulta.

1.4.1.1. Chord

O sistema Chord [46] consiste de *n* usuários participantes, cada um dos quais pode ter alguns registros armazenados e está preparado para armazenar *bits* e itens do índice para a utilização por outros usuários. O endereço IP de cada usuário pode ser mapeado para um número de *m* bits através de uma função *hash* consistente , como SHA-1 [5]. Desse modo, é possível converter qualquer endereço IP em um número de 160 *bits* chamado de identificador do nó.

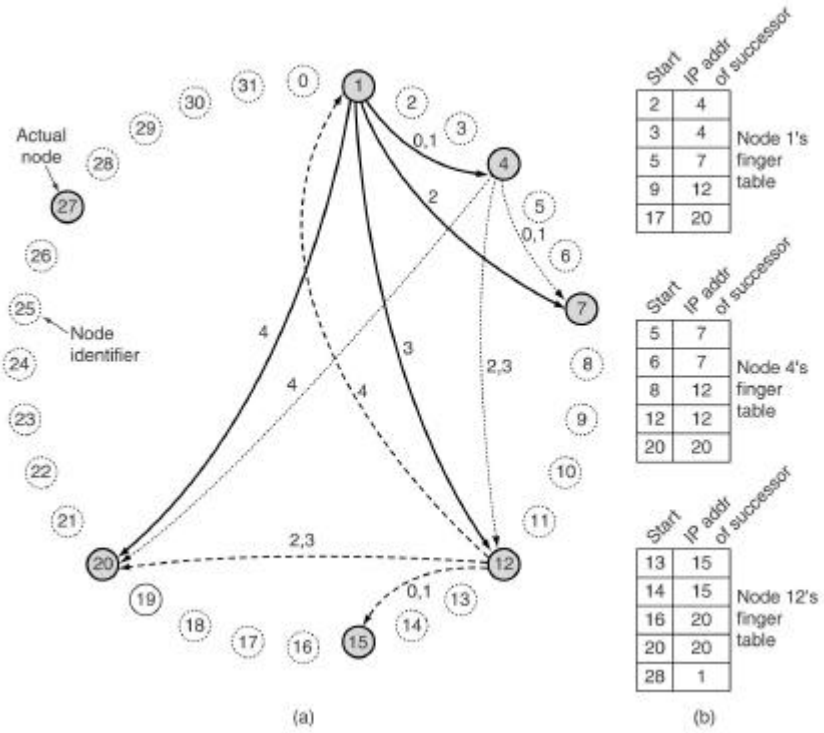

Figura 9: (a) Um conjunto de 32 identificadores de nós organizados em um círculo. (b) Exemplos de tabelas *finger*.

Conceitualmente, todos os 2^{160} identificadores estão organizados em um espaço de chave circular. A Figura 9 mostra o círculo de identificadores de nós para $m = 5$, ou seja, 2^5 identificadores. Observe que os nós 1, 4, 7, 12, 15, 20 e 27 correspondem a nós reais e estão sombreados na Figura 9, apenas esses nós fazem parte da rede.

Os identificadores são representados como um círculo de números de 0 a $2^m - 1$, então o sucessor(k) é definido como sendo o identificador de nó do primeiro nó real seguinte a k, no sentido horário. Por exemplo, o sucessor(5) = 7, sucessor(9) = 12 e sucessor(20) = 27. Os nomes de registro também são mapeados para números através da função *hash* (SHA-1) para gerar um número de 160 *bits*, denominados chaves. Dessa forma, para converter o nome do registro (arquivo procurado) em sua chave usamos chave = *hash* (nome). Se uma pessoa deseja tornar disponíveis os seus registros, primeiro ela deve criar uma tupla do tipo (nome, IP) e depois pedirá ao sucessor (*hash*(nome)) para armazenar a tupla. Se mais tarde algum usuário quiser procurar por nome, ele utilizará o *hash* do nome para obter chave, e depois usará sucessor (chave) para encontrar o endereço do IP do nó que armazena suas tuplas de índice.

Localização de Chave Simples

De um modo geral, o algoritmo Chord funciona da seguinte forma: o nó solicitante envia um pacote a seu sucessor contendo seu endereço IP e a chave que está procurando. O pacote se propaga pelo anel até localizar o sucessor para o identificador que está sendo procurado. Quando o sucessor (aquele que possui a chave solicitada) é encontrado, ele devolve as informações diretamente ao solicitante, cujo endereço IP ele dispõe. Cada nó contém o

endereço IP do seu sucessor e de seu predecessor, de forma que as consultas possam ser enviadas no sentido horário ou anti-horário, dependendo do percurso mais curto.

Mesmo com duas opções de sentido, a pesquisa linear de todos os nós é muito ineficiente em um sistema P2P de grande porte, pois o número médio de nós exigidos por pesquisa é N/2.

Localização de Chave Escalável

Para tornar a busca mais rápida, o Chord mantém informações adicionais de outros nós. Estas informações são armazenadas em uma tabela de *m* entradas, denominada de tabela *finger*. Cada uma das entradas tem dois campos: início e o endereço IP de seu sucessor, como mostra os exemplos apresentados na Figura 9b. Os valores dos campos correspondentes à entrada i no nó k são:

inicio = $k + 2^i$ (módulo 2^m)
endereco IP de sucessor(inicio[i])

Este esquema tem duas características importantes. Primeiramente, cada nó armazena informações de um número relativamente pequeno de nós, em que a maior parte desses nós tem identificadores de nós bastante próximos. Segundo, a tabela *finger* de um nó não contém informações suficientes para determinar diretamente o sucessor de uma chave *k* arbitrária. Por exemplo, o nó 1 na Figura 9a não pode determinar o sucessor da chave 14, uma vez que o sucessor (nó 15) não aparece em sua tabela *finger*. Utilizando a tabela *finger*, a pesquisa da chave no nó *k* prossegue da seguinte maneira:

Se a chave estiver entre *k* e o sucessor(*k*), então o nó que contém a informação sobre a chave é o sucessor(*k*), e a pesquisa se encerra. Caso contrário, a tabela *finger* é consultada para determinar a entrada cujo campo início é o predecessor mais próximo da chave. A seguir, uma solicitação é enviada diretamente ao endereço IP contido nessa entrada da tabela *finger*, solicitando que ele continue a pesquisa. Tendo em vista que cada pesquisa reduz à metade a distância restante até o destino, é possível mostrar que o número médio de pesquisa é $log_2\ n$. Para melhorar o entendimento, considere a pesquisa de chave = 14 no nó 1. Como 14 não está entre 1 e 4, a tabela *finger* é consultada. O predecessor mais próximo a 14 é 9, e assim a solicitação é encaminhada ao endereço IP da entrada de 9, isto é, ao nó 12. O nó 12 verifica que o nó 14 está entre ele e o seu sucessor, no caso 15, e assim retorna o endereço IP do nó 15.

O Chord ainda traz um procedimento que visa corrigir a tabela *finger*, constantemente desatualizada pela entrada e saída dos nós. Maiores detalhes podem ser obtidos em [46].

1.4.1.2. Pastry

No Pastry [41], a cada nó da rede é atribuído um identificador (*nodeId*) de 128 *bits* que pode ser gerado a partir de uma função *hash* aplicada ao seu endereço IP ou a sua chave pública. O *nodeId* é usado para indicar a posição do nó em um espaço de chave circular com faixa de 0 à $2^{128} - 1$. Uma chave é mapeada para um nó cujo *nodeId* está numericamente

mais próximo da identificação da chave. Assumindo que uma rede consiste de *N* nós, o Pastry pode rotear qualquer mensagem em O($log_2^b N$) *hops* (*b* é um parâmetro de configuração).

Para o propósito de roteamento, o *nodeId* e as chaves são pensados como uma seqüência de dígitos com base 2^b. O Pastry roteia as mensagens para o nó cujo *nodeId* está numericamente mais próximo da chave pesquisada. Isto é feito da seguinte forma: a cada etapa do roteamento, um nó normalmente encaminha as mensagens para outro nó cujo *nodeId* compartilhe com a chave pelo menos um digito (ou *b bits*) a mais do que é compartilhado com o nó atual. Se nenhum nó é conhecido, a mensagem é encaminhada para um nó cujo *nodeId* compartilha um prefixo com a chave e está numericamente mais próximo da chave do que o presente nó. Para suportar este procedimento de roteamento, cada nó mantém uma tabela de roteamento, um conjunto de vizinhanças e um conjunto de folhas.

Uma tabela de roteamento é formada por $\lfloor log_2^b N \rfloor$ linhas, cada uma com $2^b - 1$ entradas. Cada entrada na tabela de roteamento contém o endereço IP de potenciais nós cujo *nodeId* tem um prefixo apropriado.

O conjunto de vizinhança *M* contém os *nodeIds* de |*M* | nós que estão mais próximos (de acordo com uma métrica de proximidade) do nó local. O conjunto de vizinhança é normalmente utilizado no roteamento das mensagens.

O conjunto de folhas, *L*, é formado pelos |*L*|/ 2 nós sucessores e |L|/ 2 nós predecessores mais próximos de um dado nó. A Figura 10 apresenta um estado hipotético de um nó Pastry com *nodeId* igual a 10233102 (base 4), em um sistema de 16 *bits* para identificação e um valor de b = 2.

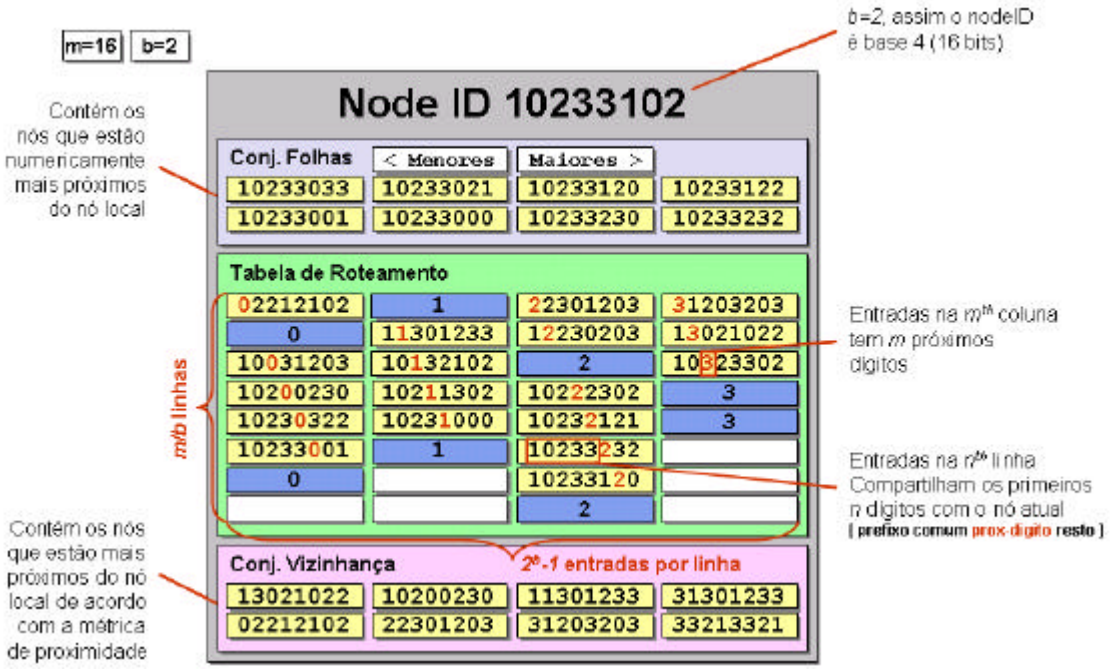

Figura 10: Tabela de roteamento do Nó 10233102

O procedimento de roteamento funciona da seguinte maneira: dado uma mensagem, o nó primeiro checa se a chave está dentro da faixa de endereços coberto pelo conjunto de folhas. Em caso afirmativo, a mensagem é encaminhada diretamente para o nó destinatário.

Isto quer dizer que existe um nó no conjunto de folhas que está mais próximo da chave pesquisada (possivelmente no presente nó). Se a chave não é encontrada no conjunto de folhas, então a tabela de roteamento é usada e a mensagem é encaminhada para o nó que compartilha um prefixo comum com a chave (pelo menos um ou mais dígitos). Em certos casos, é possível que o nó associado não esteja alcançável. Neste caso, a mensagem é encaminhada para um nó que compartilhe um prefixo com a chave e está numericamente mais próximo da chave do que o nó atual.

O sistema Pastry ainda fornece alguns procedimentos para lidar com as questões de entrada e saída de nós da rede. Além de aplicar uma heurística que garante que as entradas na tabela de roteamento são escolhidas de maneira a fornecer uma boa localização.

1.4.1.3. Tapestry

O Tapestry [47] é muito similar ao Pastry. Entre as similaridades estão o uso do prefixo/sufixo no roteamento, o algoritmo de entrada e saída dos nós e o custo de armazenamento do conteúdo. Por estes motivos, abordaremos neste texto somente as principais diferenças.

No Tapestry não existem os conjuntos de nós folhas e de vizinhos. Quando a tabela de roteamento de um nó não possui uma entrada para um nó que compartilhe um sufixo comum com a chave (pelo menos um ou mais dígitos), a mensagem é encaminhada para um nó que está numericamente mais próximo da chave do que o nó atual. O número de saltos esperados é $log_{16}N$.

1.4.1.4. CAN

O CAN (*Content Addressable Network*) [40] é baseado em um espaço cartesiano de *d* dimensões sobre uma toróide. Este espaço de coordenada é completamente lógico e não possui nenhuma relação com as coordenadas físicas do sistema. Este plano virtual é dinamicamente dividido entre todos os nós de maneira que cada nó possua sua própria zona. Por exemplo, a Figura 11 mostra um espaço de coordenada 2d [0,1] x [0,1] dividido entre 5 nós.

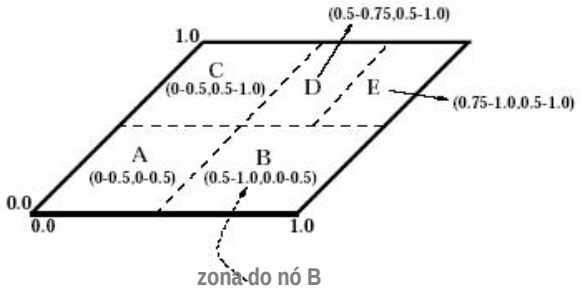

Figura 11: Exemplo de um espaço 2d com 5 nós

O espaço virtual é usado para armazenar todos os pares (chave, valor) entre os nós participantes da rede. Uma chave *K* é deterministicamente mapeada em um ponto *P* do espaço cartesiano utilizando uma função *hash* uniforme. O par (chave, valor) é então armazenado no nó responsável pela zona onde o ponto *P* se encontra. Quando um nó deseja

encontrar o valor correspondente à chave *K*, ele deve aplicar a mesma função no valor da chave para encontrar o ponto para o qual irá direcionar sua busca. Se o ponto *P* não é de responsabilidade do nó, ou de seus vizinhos, a busca deverá ser roteada pela rede CAN até encontrar o nó que possua a zona onde o ponto *P* se encontra.

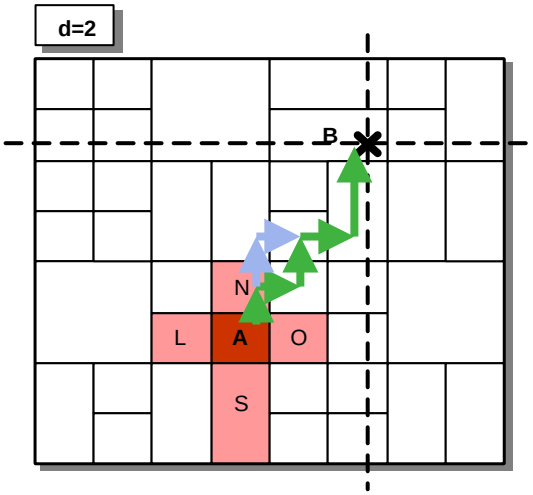

Figura 12: Roteamento no espaço de coordenadas 2d

Cada nó mantém uma tabela de roteamento contendo o endereço IP e a coordenada da zona de cada um dos seus vizinhos imediatos. Uma mensagem CAN inclui as coordenadas do destinatário. Usando o conjunto de coordenadas dos seus vizinhos, um nó faz o roteamento das mensagens para o vizinho que mais se aproxima das coordenadas do destinatário, conforme mostra a Figura 12.

Para um espaço com *d* dimensões dividido em *n* zonas iguais, em média são necessários $(d/4)(n^{1/d})$ saltos e cada nó mantém 2d vizinhos.

1.4.2. Plataformas de Desenvolvimento

Muitas aplicações P2P atuais não utilizam nenhum arcabouço (*framework*) para desenvolver suas aplicações. Entretanto, já é possível utilizar algumas plataformas de desenvolvimento para alavancar a construção de novas soluções P2P, como JXTA, .Net e Groove.

1.4.2.1. JXTA

Até recentemente, a tecnologia P2P esteve sendo utilizada em aplicações de função única, tal como comunicadores instantâneos. Levando o conceito de P2P mais adiante, a Sun Microsystems concebeu a idéia do Projeto JXTA como um meio de integrar a tecnologia P2P ao núcleo da arquitetura de rede [23].

O Projeto JXTA é um conjunto de protocolos P2P simples e abertos que habilitam os dispositivos na rede a se comunicarem, colaborarem e compartilharem recursos. Os *peers* JXTA criam uma rede virtual *ad hoc* no topo de redes existentes, mascarando a complexidade existente nas camadas de baixo. Na rede virtual JXTA, qualquer *peer* pode interagir com outros *peers*, independente de sua localização, tipo de serviço ou ambiente operacional – mesmo quando alguns *peers* e recursos estão posicionados atrás de *firewalls* ou estão em diferentes tecnologias de transporte de rede. Assim, o acesso aos recursos da

rede não é limitado por incompatibilidades de plataforma ou restrições da arquitetura cliente-servidor.

A tecnologia do Projeto JXTA adota como objetivos: interoperabilidade (entre diferentes sistemas e comunidades P2P), independência de plataforma (diversas linguagens, sistemas e redes), generalidade (qualquer tipo de dispositivo digital) e segurança. A tecnologia JXTA funciona em qualquer dispositivo, incluindo aparelhos celulares, PDAs, sensores eletrônicos, estações de trabalho e servidores. Baseado em tecnologias aprovadas e padronizadas como HTTP, TCP/IP e XML, JXTA não é dependente de nenhuma linguagem de programação particular, sistema de rede, ou plataforma de sistema e pode trabalhar com uma combinação dos mesmos.

JXTA está posicionada como uma pilha P2P, uma camada localizada acima do sistema operacional ou máquina virtual e abaixo das aplicações e dos serviços P2P. Pode ser descrita simplesmente como uma tecnologia que permite a comunicação entre *peers*. Cada *peer* é associado a um identificador único, um "peer ID", e pertence a um ou mais *peergroups*. Dentro dos *peergroups* os *peers* cooperam e têm funções similares sob um conjunto unificado de capacidades e restrições. JXTA provê protocolos para as funções básicas: criar e encontrar grupos, entrar e sair de grupos, monitorar os grupos, conversar com outros grupos e *peers*, compartilhar conteúdo e serviços – tudo isso é realizado através da publicação e troca de anúncios XML e mensagens entre os *peers*.

Conceitualmente, cada *peer* no JXTA abstrai três camadas: o núcleo, a camada de serviços e a camada de aplicação. O núcleo é responsável por gerenciar o protocolo JXTA; ele encapsula o conhecimento de todas as operações P2P básicas. Ou seja, ele contém as funcionalidades e a infra-estrutura suficientes para o desenvolvimento de qualquer aplicação P2P. A camada de serviço, por sua vez, armazena as funcionalidades comuns que mais de um programa P2P poderia utilizar. Ela provê funcionalidades similares a de uma biblioteca que pode ser controlada pelas aplicações JXTA através de lógica na camada de aplicação. A camada de aplicação, finalmente, é onde a aplicação P2P realmente reside. Ela pode permitir que o usuário controle diferentes serviços, ou pode ser onde a lógica de uma aplicação autônoma opera. Por exemplo, um simples programa de bate-papo pode ser construído nessa camada, fazendo uso tanto do serviço quanto do núcleo para permitir que os *peers* troquem mensagens.

Para desenvolvedores, o Projeto JXTA provê um conjunto de blocos que permitem uma sólida fundação para aplicações computacionais distribuídas e dão suporte a funções comuns requeridas por qualquer sistema P2P. Utilizando esses recursos, os desenvolvedores podem elaborar suas aplicações P2P mais facilmente. Já estão disponíveis APIs em Java, C++ e outras linguagens. Os protocolos JXTA estão especificados em alto-nível e, portanto, podem ser implementados teoricamente em qualquer linguagem.

A arquitetura JXTA está atualmente em sua segunda versão. Modificações significativas foram realizadas a fim de criar redes P2P de melhor desempenho, maior escalabilidade e facilidade de manutenção.

1.4.2.2. .NET

A plataforma .Net [31] disponibiliza um conjunto de alternativas que podem ser utilizadas para construir aplicações P2P. Entretanto, é importante conhecer como as funcionalidades podem ser utilizadas, de modo a tornar mais fácil a escolha do modelo mais adequado a cada aplicação. A plataforma .NET disponibiliza quatro modelos (ou tipos) de aplicação para P2P [32]:

- Web Services: Essa tecnologia provê mecanismos para fazer registro, descoberta e recuperação (*download*) para aplicações P2P.
- Windows Forms: Essa é a solução fornecida pela plataforma .NET para facilitar a construção de interfaces gráficas sofisticadas e pode ser utilizada para construir aplicações P2P mais estimulantes.
- Service Process: Essa tecnologia permite construir aplicações P2P capazes de compartilhar e localizar serviços de processamento. Esse tipo de solução é muito útil para construir soluções que requerem processos distribuídos, como o Seti@home (http://setiathome.ssl.berkeley.edu).

1.4.2.3. Groove Development Kit (GDK)

O GDK [18] é uma plataforma para desenvolvimento de aplicações P2P que pode ser utilizada gratuitamente. No entanto, deve-se obter uma licença para os produtos criados com ele. O Groove foi desenvolvido utilizando *Microsof Component Object Model* (COM). Sendo assim, além de simplesmente utilizar as funcionalidades fornecidas pelo Groove, que permitem a prototipação rápida de aplicações P2P utilizando linguagens *script* como VBSCript ou JavaScript, qualquer extensão deve ser implementada utilizando objetos COM, o que prende os programas à plataforma de desenvolvimento Microsoft.

O Groove utiliza uma abordagem híbrida, o que permite o emprego de solução centralizada ou descentralizada. Além disso, ele disponibiliza os seguintes serviços:

- Armazenamento de mensagens enviadas para clientes *off-line*;
- Serviços de interface gráfica;
- Gerenciamento de dados, permitindo manipular facilmente dados sincronizados;
- Compartilhamento de espaço, possibilitando acessar ferramentas (de um grupo) em tempo de execução;
- Serviço de identificação, que permite recuperar (localizar) usuário que estão *on-line*.
- Ferramentas de publicação, que permitem disponibilizar, criar, refinar e testar as ferramentas desenvolvidas.

Tudo que é preciso para desenvolver aplicações baseadas nessa arquitetura é o kit de desenvolvimento (GDK) e conhecimento em linguagens COM como, Visual C++, Visual Basic, Delphi, etc. Informações adicionais sobre o desenvolvimento de aplicações P2P com Groove podem ser encontradas em [28] e [39].

1.4.3. Redes Públicas

Redes P2P tornaram-se populares com o surgimento do Napster em 1999 e continuaram em evidência com os serviços providos pelo KaZaA e Morpheus. Em poucos anos, aplicações P2P passaram a ser utilizadas por milhares de usuários para troca de mensagens, texto, áudio e vídeos, armazenados em milhares de computadores pessoais espalhados em todo o mundo. Atualmente existem várias aplicações P2P disponíveis na Internet. Esse texto descreve em resumo o funcionamento de algumas dessas aplicações.

1.4.3.1. Gnutella

Pode-se dizer que o Gnutella (http://www.gnutella.com) é um dos primeiros protocolos e aplicações verdadeiramente P2P. Difere do Napster por não possuir um servidor centralizado. O Gnutella é baseado em um método de descoberta dinâmico para encontrar outros nós Gnutella na Internet. Tanto o cliente quanto o servidor são empacotados em uma única distribuição binária. A rede que é formada pela união de todos os nós que rodam o protocolo Gnutella chama-se Gnutella Network ou gNet. O protocolo Gnutella está em constante evolução e a última versão é a 0.6 [26].

Gnutella é um popular protocolo P2P, e foi inicialmente desenvolvido por programadores da NullSoft com a finalidade de compartilhar arquivos, era disponibilizado sobre GNU Public License (GPL) por volta de março de 2000. A AOL adquiriu a NullSoft após o surgimento do WinAmp, e devido a problemas com gravadoras e o Napster, a AOL suspendeu qualquer desenvolvimento formal do Gnutella. Porém, ocorreu uma reação por parte da comunidade internet, que uniu esforços individuais e corporativos para inovar o Gnutella, mantê-lo funcionando e trabalhando melhor.

O alvo primário do Gnutella é o compartilhamento de músicas, dando ênfase à computação P2P. Há clientes disponíveis para diferentes plataformas, tais como: Windows, Unix, Macintosh e Linux.

Arquitetura

A rede Gnutella consiste em um sistema descentralizado onde os nós estão conectados via TCP/IP e executando um software que implementa o protocolo Gnutella. Para se conectar à rede, um nó precisa saber previamente um endereço de um outro nó que já participa da rede.

A rede Gnutella é do tipo *broadcast*, no qual consultas são duplicadas e repetidas para os demais nós. Para melhorar o desempenho e a escalabilidade da rede o Gnutella categoriza os nós em "supernó" e "nó cliente". O supernó nada mais é que uma máquina confiável e que possui alta taxa de transmissão de dados agindo como um servidor *proxy*, servindo vários "nós clientes". Dessa forma, o "supernó" remove uma carga excessiva da rede roteando mensagens dos clientes que possuem baixa taxa de transmissão de dados.

Nesse caso, um usuário que acessa a rede via modem conecta-se diretamente ao "supernó" melhorando bastante o seu desempenho. Quando utiliza esse conceito, a rede Gnutella imita a própria Internet: nós que possuem baixa taxa de transmissão de dados se conectam a potentes roteadores que transmitem a maioria dos dados utilizando altas taxas de transmissão.

Procedimento de Busca

O nó onde a busca foi realizada envia uma mensagem com a consulta para todos os nós que estão diretamente ligados a ele. Esses nós por sua vez, repassam a consulta para todos os nós vizinhos. Esse processo se repete atingindo todo os nós pertencentes a um determinado raio.

Como as consultas consomem muita largura de banda da rede Gnutella, elas são limitadas a 256 *bytes*. Além disso, o Gnutella utiliza um parâmetro (TTL) para especificar o número máximo de vezes que a consulta pode se propagar para outros nós. Os nós são responsáveis por ignorar uma determinada consulta quando o TTL expira.

As consultas no Gnutella contêm uma *string* com as palavras chaves da busca. Para atingir esse critério o nome do arquivo deve conter todos os termos procurados. Sendo assim, uma consulta por "Caetano mp3" retornaria o arquivo "musicas_caetano.mp3" e não retornaria "caetano.doc". As consultas possuem um campo que determina a velocidade mínima que um *nó* deve possuir para que ele esteja apto a retornar uma consulta, isso evita que máquinas que tenham uma baixa taxa de transmissão forneçam arquivos. Pode existir ainda um tipo de consulta que requisite todos os arquivos de um determinado nó.

Um nó responde a uma consulta quando o seu conteúdo satisfaz a consulta. A resposta do nó contém o seu endereço IP e uma porta onde o nó pode ser acessado para transferir o arquivo. A resposta percorre o mesmo caminho da consulta.

Transferência de arquivos

Uma vez que o nó que realizou a busca e recebe uma resposta positiva, ele sabe o endereço IP do nó, ou dos nós, que atendeu, ou atenderam, a sua consulta. Sendo assim, ele abre uma conexão TCP/IP diretamente para o nó que possui os arquivos. Se o nó que contém os arquivos está embaixo de um *firewall*, o nó envia uma consulta *push*, esse tipo de consulta diz para o nó que contém os arquivos para ele enviar determinados arquivos para o solicitante. Caso as duas máquinas estejam em baixo de um *firewall*, outras técnicas mais elaboradas devem ser utilizadas [33].

1.4.3.2. Freenet

Freenet [10] (http://freenet.sourceforge.net/) nasceu de um projeto iniciado em 1997 por Ian Clarkena na divisão de informática da Universidade de Edinburgh. Os objetivos do Gnutella e Freenet são diferentes. Enquanto o Gnutella concentra-se em ser um mero aplicativo, o Freenet tem objetivos sócio-políticos, tais quais:

- Permitir que pessoas distribuam material anonimamente;
- Permitir que esse material seja consultado também de forma anônima;
- Garantir que seja praticamente impossível a retirada completa do material na rede;
- Operar sem controle central.

Tecnicamente, o Freenet possui um mecanismo dinâmico de publicação que cria réplicas dos arquivos solicitados nos *peers* entre o solicitante e o fornecedor. Isso possibilita otimizar buscas dinamicamente, diminuindo assim o número de saltos, uma vez que os *peers*

de interesse comum tendem a estar conectados de forma direta ou próxima, conforme a Figura 13.

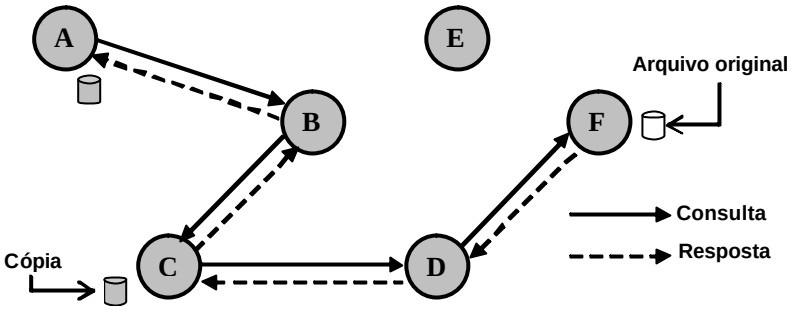

Figura 13: Mecanismo de replicação de arquivos do Freenet

O Freenet também possui um controle do tempo de publicação de cada arquivo, o material que passa muito tempo sem ser consultado é excluído da rede. Durante a busca, quando o cliente Freenet recebe uma requisição e não pode atendê-la, ele encaminha para apenas um *site* (*unicast*) onde a busca é feita em profundidade. Ao contrário do Gnutella que a repassa a todos os seus vizinhos (*multicast*). Por isso o Freenet apresenta maior escalabilidade e menor demanda de tráfego.

Arquitetura

O Freenet possui uma topologia de rede similar a do Gnutella, exceto pelo fato de que o Freenet tem como objetivo principal criar um sistema global de armazenamento de informações. O foco principal é no armazenamento de arquivos. A rede Freenet não apenas roteia mensagens, ela também armazena vários arquivos em máquinas que não os pertence. Sendo assim, o Freenet constrói um imenso sistema de armazenamento distribuído e tolerante a falhas.

Os nós estão conectados via TCP/IP. Quando um novo nó entra na rede ele tem que conhecer o endereço de algum outro nó que pertença à rede. Além disso, o cliente deve especificar a quantidade de disco disponível para armazenar dados de outros nós. O usuário não é capaz de acessar o conteúdo que está armazenado na sua máquina local, pois esses dados ficam codificados.

Todo arquivo armazenado no Freenet é identificado por uma chave única, dessa forma é necessário saber a chave do arquivo antes de recuperá-lo. Quando o *download* do arquivo é realizado, a aplicação re-aplica o algoritmo sobre o arquivo para regerar a chave e comparar com a chave original, esse procedimento permite verificar se o arquivo foi adulterado pela rede.

Procedimento de Busca

A busca no Freenet evita encher a rede de consultas, assim como faz a rede Gnutella. Cada nó do Freenet envia uma consulta para nós próximos ao que eles acham que possui a informação. Essa decisão é tomada levando em consideração uma tabela de roteamento que cada nó possui. Essa tabela armazena endereços de outros nós e chaves que eles devem possuir.

Quando um nó atende a uma consulta, eles armazenam a direção (caminho) dos nós que possuem a consulta, otimizando futuras solicitações desse arquivo. Os nós que repassam a resposta também podem armazenar o arquivo localmente, passando a ser fonte da informação para consultas futuras.

Transferência de arquivos

Um dos objetivos do Freenet é manter anônimos os nós que possuem um arquivo. Diferentemente do Gnutella, a recuperação de um arquivo é realizada através de vários nós, o que dificulta a identificação do nó que forneceu o arquivo, tornando anônimas as máquinas que compartilham informações.

1.4.3.3. FastTrack

O FastTrack é a rede pública dos aplicativos: KaZaA, KaZaA Lite, iMesh e Grokster. Essa rede utiliza uma arquitetura em duas camadas, onde a primeira é composta por supernós (máquinas mais potentes e que possuem conexão rápida) e a segunda é formada por máquinas de usuários comuns.

Os supernós assumem um papel principal na rede, uma vez que todo usuário precisa se conectar a um supernó para ter acesso ao sistema. Além disso, eles são responsáveis pela busca que é realizada da seguinte forma:

1. Ao se conectar a um supernó as máquinas avisam quais os nomes dos arquivos que elas dispõem. Quando chega uma consulta, a busca é feita apenas nesse sub-conjunto dos dados.
2. O supernó retorna o resultado da busca feita localmente. Esse processo é rápido e o efeito causado no usuário final é excelente, uma vez que ele percebe que o sistema está respondendo rapidamente.
3. O supernó, então, submete a consulta a outros supernós, para encontrar respostas mais precisas para a consulta do usuário.

O FastTrack é a maior rede pública existente e mantém em média 4 milhões de usuários conectados a todo o momento. Esses usuários disponibilizam diferentes tipos de arquivos, como vídeo, áudio, softwares, etc.

1.4.3.4. Edonkey2000 e OverNet

Edonkey2000 (http://www.edonkey2000.com) é a rede dos aplicativos: eDonkey, eMule (http://www.project-emule.com). Trata-se de rede centralizada, entretanto a empresa que desenvolveu essa rede, a MetaMachine, distribuiu o software servidor. Atualmente, qualquer usuário está apto a criar o seu próprio servidor.

Essa rede se destaca das demais pela quantidade de arquivos de vídeo que elas contém. A maior parte dos arquivos disponíveis são de boa qualidade. Isso se deve ao fato de que cada arquivo armazenado no sistema possui um código *hash* que o identifica unicamente. Esses arquivos são indexados e selecionados em *sites* especializados, que excluem os arquivos de baixa qualidade ou danificados.

Um problema acarretado pela distribuição do software servidor é que o próprio usuário fica responsável por manter a lista de servidores atuais. Para facilitar essa tarefa, existem alguns *sites* que mantém a lista de servidores mais atuais, portanto o que o usuário

precisa fazer é ir ao *site* e recuperar a lista. No entanto, estes servidores podem (e ficam) facilmente congestionados, prejudicando o desempenho da rede eDonkey.

Overnet (http://www.overnet.com) foi desenvolvido para resolver os problemas de servidores centralizados do eDonkey. Ele publica as informações e efetua as buscas de uma maneira completamente descentralizada, usando um modelo DHT. Ambos (eDonkey2000 e Overnet) usam o mesmo protocolo para transferência rápida de arquivos, o Multisource FTP (MFTP). A tendência é que os usuários do eDonkey2000 migrem para a rede Overnet.

1.5. Atividades do Grupo de Trabalho P2P da RNP

Esta seção tem por objetivo apresentar o Grupo de Trabalho de Computação Colaborativa da RNP [19], seus objetivos e atividades, bem como a experiência da equipe com algumas questões relacionadas a sistemas P2P na Internet e particularmente no *backbone* da RNP.

1.5.1. Objetivos

O Grupo de Trabalho em Computação Colaborativa (GT P2P) tem o objetivo de avaliar os benefícios da implantação de suporte a sistemas P2P na RNP e nas instituições conectadas, bem como o impacto da utilização de tais sistemas no desempenho da rede. Alguns dos objetivos específicos que definem o escopo do projeto dentro das possíveis atividades na área de P2P, são:

- Implantação de um projeto piloto para suporte a aplicações P2P;
- Análise dos efeitos do tráfego gerados por aplicações P2P no *backbone* da RNP;
- Avaliação do impacto de aplicações P2P em diversas plataformas e tecnologias de comunicação.

Nesta seção são descritos dois principais objetivos do GT-P2P: projeto piloto e avaliação de tráfego. O projeto piloto compreende uma infra-estrutura para o desenvolvimento de aplicações P2P e um conjunto de aplicações que utilizam essa infra-estrutura. A análise de tráfego apresenta um estudo feito pelo GT-P2P sobre o tráfego de aplicações P2P na RNP. Finalmente são abordados aspectos de simulação relacionados à redes P2P.

Informações adicionais sobre o GT-P2P (e formas de colaborar com o desenvolvimento) estão disponíveis em http://www.cin.ufpe.br/~gprt/gtp2p. A página também disponibiliza relatórios técnicos mais detalhados sobre o projeto e soluções que estão sendo desenvolvidas.

1.5.2. Projeto Piloto

O Projeto Piloto que está sendo desenvolvido pelo GT-P2P pode ser dividido em duas partes: infra-estrutura e aplicações. A infra-estrutura abrange uma série de serviços que serão fornecidos por máquinas executando no *backbone* da RNP para auxiliar e facilitar a construção de aplicações P2P. As aplicações compreendem o desenvolvimento de softwares para validar a infra-estrutura proposta.

1.5.2.1 Infra-estrutura

Existe atualmente uma série de ferramentas que auxiliam a construção de aplicações P2P (seção 1.4.2). A proposta do GT-P2P não é construir uma nova plataforma de desenvolvimento, mas utilizar os recursos da RNP para prover um conjunto de serviços que atendam às principais necessidades das aplicações P2P, como identificação de nomes e localização de recursos.

A infra-estrutura, denominada de XPeer é uma aplicação distribuída que permanecerá em execução no *backbone da RNP* em tempo integral. Os PoPs irão executar instâncias dessa aplicação formando uma rede P2P de altíssima velocidade dentro do *backbone* da RNP. No futuro, outras máquinas poderão executar instâncias dessa aplicação P2P, o que irão permitir acesso a um maior número de usuários, além de aumentar a quantidade de informações compartilhadas. Dessa forma, a infra-estrutura XPeer pode alcançar maior escalabilidade sem perder desempenho.

Cada nó da rede XPeer mantém um conjunto de usuários (aplicações que utilizam o serviço da rede XPeer) da RNP conectados a ele. As informações referentes a cada usuário ficam armazenadas no PoP no qual o usuário tiver sido registrado e replicadas em outros PoPs vizinhos. Essas informações podem ser acessadas por qualquer outro PoP da rede XPeer e servem para localizar um usuário ou mesmo customizar aplicações.

Através do uso desta arquitetura, diferente de uma arquitetura cliente-servidor, qualquer nó (PoP) da rede pode deixar de funcionar e mesmo assim a aplicação continuará executando. Outra característica inerente às aplicações P2P presente nessa infra-estrutura é a capacidade de um nó acessar qualquer outro nó da rede. Todo nó tem papel de cliente e de servidor, operando como servidor quando compartilham informações e como cliente quando recuperam informações na rede.

A arquitetura do XPeer é dividida em três camadas principais (Figura 14) que são:

- **Camada de Serviços** – Contém os serviços disponibilizados pelo XPeer e que serão utilizados pelas diversas aplicações.
- **Aplicação** – Consiste em uma aplicação distribuída que utiliza soluções P2P existentes como JXTA (seção 1.4.2) e Chord (seção 1.4.1).
- **Módulos úteis** – Contém os módulos que dão suporte à aplicação distribuída. Existem três módulos principais que serão utilizados por esta camada: o JXTA, o Chord e Storage. O JXTA contém uma série de funcionalidades que auxiliam na construção de aplicações P2P genéricas. O Chord provê soluções de busca e o Storage é um sistema de armazenamento de dados proprietário que poderá conter informações da aplicação e do usuário.

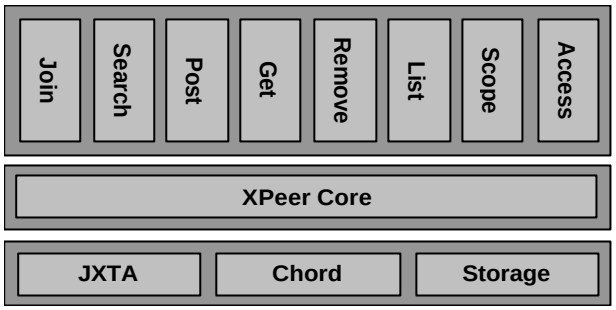

Figura 14: A arquitetura da XPeer.

A parte mais importante da infra-estrutura XPeer são os serviços que ela disponibiliza para auxiliar outras aplicações.

- **Join** – Através desse serviço o usuário solicita a sua participação na rede. Para que um usuário seja aceito pela rede, ele deve possuir um identificador único e uma senha. O usuário "publica" na rede um conjunto de informações que servem para que outros usuários possam encontrá-lo. Esse serviço também é responsável por autenticar e garantir a identidade de um usuário, podendo ser utilizado a partir de qualquer dispositivo capaz de se conectar a rede como, *laptops* e PDAs.
- **Leave** – Esse serviço faz com que o usuário deixe a rede de forma elegante, requisitando à XPeer que remova as informações publicadas pelo usuário. Para saber o que acontece quando o usuário desconecta da rede sem utilizar esse serviço veja a descrição do serviço Post abaixo.
- **Scope** – Especifica a rede visível por um usuário, ou seja, define os usuários que são acessíveis durante uma busca.
- **Access** – Especifica o acesso que um determinado usuário tem às informações compartilhadas por um determinado usuário, ou seja, esse serviço serve para restringir o acesso às informações compartilhadas por um usuário.
- **Post** – Cada usuário pertencente à rede XPeer mantém uma estrutura de dados com informações públicas, que servem para localizar o usuário na rede. O Post é o serviço que permite adicionar informações nessa estrutura, sendo cada informação composta por uma tupla contendo o nome do campo e o valor do campo. Uma aplicação de Xadrez, por exemplo, poderá utilizar essa estrutura para armazenar o ranking do usuário.
- **Get** – Recupera informações públicas de um usuário cadastradas pelo serviço Post.
- **Remove** – Remove um campo publicado através do Post.
- **List** – Lista todos os campos armazenados na estrutura de dados da aplicação.
- **Search** – Este serviço permite localizar um usuário na rede XPeer. Essa localização pode ser feita através do nome do usuário ou de uma informação publicada por ele. O nome do campo, publicado através do Post, também pode ser utilizado para recuperar um usuário.

1.5.2.2 Aplicações

A infra-estrutura citada na seção anterior permite a construção de diversas aplicações P2P. Para validá-la, o GT-P2P está propondo a construção de duas aplicações: uma mais simples de troca de mensagens e outra mais elaborada com o objetivo de promover uma maior integração educacional entre os usuários da RNP.

Nessa sub-seção será descrita apenas a aplicação educacional chamada de XBrain. Seu objetivo é fazer com que pessoas que estão dispostas a doar um pouco do seu tempo possam ajudar pessoas que tem sede por conhecimento. Para esclarecer o funcionamento dessa aplicação imagine o seguinte cenário: Eduardo tem 27 anos, é formado em Ciências da Computação e se acha capaz de auxiliar pessoas interessadas em matemática básica. Assim, Eduardo entra na aplicação XBrain e se cadastra como uma pessoa capaz de ajudar alguém nesse assunto. Feito isso, ele deixa a aplicação e vai navegar na Internet. Enquanto isso, um garoto chamado Marquinhos, entra na rede XBrain à procura de alguém capaz de lhe ajudar a resolver funções de primeiro grau. Fazendo uma busca pela palavra matemática, ele recebe como resposta um conjunto de pessoas capazes de ajudá-lo nesse assunto, entre elas, Eduardo. Marquinhos seleciona Eduardo e, a partir daí, os dois passam a se comunicar trocando mensagens ponto-a-ponto (P2P).

Essa aplicação pode ter um impacto importante na Internet, uma vez que irá permitir que pessoas que tenham conhecimento possam ajudar tantas outras que possuem deficiência. Além disso, ela tem um grande apelo social, tendo em vista que o conjunto menor de pessoas que tiveram acesso a educação de boa qualidade estarão "doando" um pouco do seu tempo para levar o conhecimento a um conjunto maior de pessoas que não tiveram a mesma oportunidade.

Isto é chamado de "Computação Colaborativa e Solidária". É colaborativa porque os computadores interagem entre si de maneira direta (P2P), todos se comportando como clientes e servidores e compartilhando informações. É solidária porque permite que as pessoas compartilhem o seu conhecimento uns com os outros. Um outro aspecto importante a ressaltar é que, da mesma forma que os computadores, os usuários têm um comportamento P2P, onde qualquer um pode ser professor em um assunto e aluno em outro. A grande idéia é alterar o paradigma de educação à distância, deixando a tradicional interação professor/aluno com hora marcada, para uma forma de complementação pedagógica que tem um enorme poder sinérgico. Existe uma grande oportunidade para oferecer às pessoas a possibilidade de fazerem um *trabalho voluntário sem saírem da frente dos seus computadores*.

O GT-P2P irá disponibilizar essa aplicação para ser executada nas máquinas dos usuários da RNP utilizando os serviços da rede XPeer. A Figura 15 mostra a aplicação XBrain solicitando serviços da infra-estrutura XPeer, instalada no *backbone* da RNP. Para utilizar os serviços XPeer cada aplicação XBrain precisa se conectar a um PoP que disponibiliza os serviços. Cada PoP vai manter um registro dos usuários de forma a possibilitar a autenticação dos usuários na rede, bem como fornecer subsídios que possibilitem a localização desse usuário.

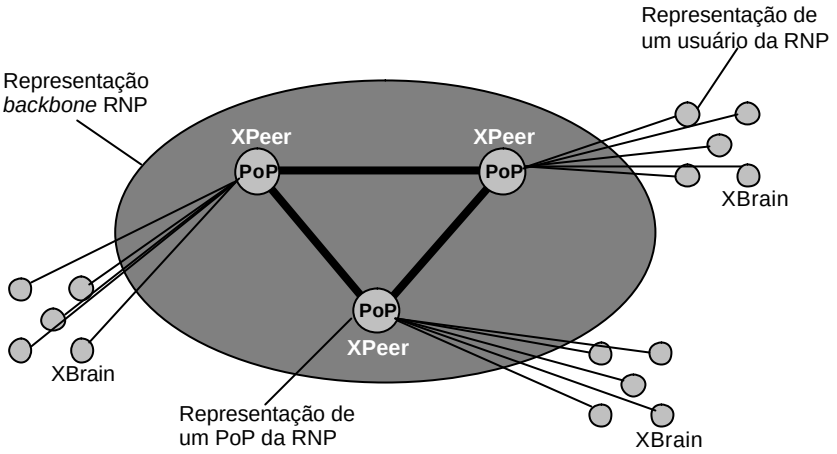

Figura 15: Aplicação XBrain utiliza os serviços da rede XPeer instalada no backbone da RNP.

As buscas também serão realizadas nos PoPs, uma vez que eles contêm todas as informações públicas referentes aos usuários. Sendo assim, a busca por um usuário ou pela informação que ele disponibiliza somente precisará ser realizada na rede XPeer dentro do *backbone* da RNP. O papel da aplicação XPeer termina quando localiza um usuário na rede, a partir daí os usuários devem se conectar diretamente, ou seja, P2P.

Uma vez que utiliza os serviços da XPeer para realizar as principais tarefas, a aplicação XBrain se torna muito simples. Os principais componentes (Figura 16) são:

- **Interface gráfica** – Esse módulo contém as telas da aplicação e acessa a aplicação XBrain através de uma "fachada". A fachada é responsável por disponibilizar as principais funcionalidades que podem ser exibidas e manipuladas. Estima-se que a aplicação será composta por 5 telas principais que irão permitir: a entrada do usuário na rede Xpeer, cadastrar as habilidades e conhecimentos desejados, localizar um "professor" na rede, trocar conhecimentos (mensagens) e publicar questões (dúvidas).
- **Fachada** – Esse módulo tem o papel de isolar a interface gráfica da aplicação. Isso é extremamente importante para uma aplicação P2P, porque ela poderá possuir várias interfaces diferentes. Além disso, como um dos objetivos do GT-P2P é disponibilizar essa aplicação em diferentes dispositivos como, celulares, *notebooks*, *desktops* e PDAs. Com o uso da fachada cada um desses dispositivos poderá utilizar interfaces diferentes para acessar as mesmas funcionalidades da aplicação.
- **XBrain Core** – Esse módulo contém a aplicação propriamente dita. Nele serão programadas as regras de negócio da aplicação. Esse módulo faz uso de dois componentes de comunicação para executar as principais tarefas referentes à conexão P2P e também à conexão com a rede XPeer.
- **Módulos para Comunicação** – A comunicação P2P e a utilização dos serviços XPeer são tarefas bastante específicas e devem ser isoladas em componentes. Esses componentes terão como tarefa principal fazer a conexão entre dois computadores e possibilitar a utilização dos serviços oferecidos pela rede XPeer.

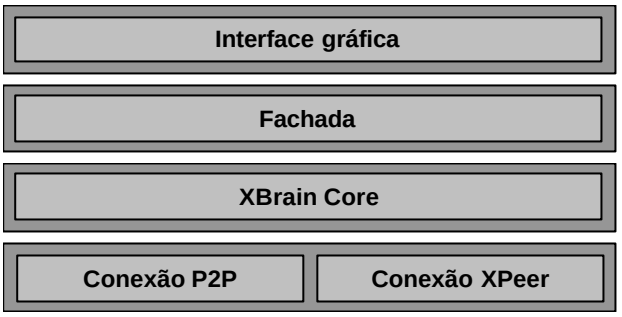

Figura 16: Arquitetura da aplicação XBrain

1.5.3. Avaliação de Tráfego

Essa seção apresenta os resultados obtidos na avaliação da utilização dos recursos da rede RNP (os recursos computacionais e os enlaces de comunicação) pelos sistemas P2P existentes na Internet, principalmente para transferência de arquivos (ex: KaZaA).

O objetivo é avaliar o impacto gerado pelo uso destas aplicações no *backbone* da RNP, com base em dados coletados sobre fluxos durante uma semana. Os resultados da análise mostram indícios de um grande impacto do tráfego das aplicações P2P no tráfego global da rede [42].

1.5.3.1. Metodologia

Os dados analisados foram obtidos através de arquivos coletados pelo grupo de trabalho GT-QoS (http://www.nuperc.unifacs.br/gtqos/), responsável pelo projeto-piloto de medição do tráfego no *backbone* da RNP. O tráfego analisado compreende a um conjunto de arquivos no formato binário gerado pelo NetFlow [9] que foram coletados no período de 02 a 08 de novembro de 2003.

As métricas adotadas para este estudo visam descrever o comportamento básico do tráfego em volume, vazão e duração dos fluxos. Entre as ferramentas utilizadas para a obtenção de resultados, destacam-se: Flow-Tools, AWK e c-shell script.

Para otimizar o uso das ferramentas acima descritas, foram definidas técnicas para manipulação e exibição dos resultados dos relatórios. Os principais critérios de filtragem foram em relação às portas P2P, interfaces do POP-SP com os demais POPs da RNP, aplicações mais utilizadas a serem comparadas às aplicações P2P, tempo mínimo do fluxo e unidades de tempo dos relatórios finais.

1.5.3.2. Resultados

A Figura 17 mostra o perfil de tráfego relacionado ao volume de dados passante no POP-SP durante o período de medição. Para apresentação, cada amostra no gráfico corresponde ao volume de tráfego num intervalo de 5 minutos. A primeira impressão que se tem deste resultado é que o perfil de tráfego não aparenta substanciais alterações em relação ao perfil conhecido do tráfego Internet [7]. Ou seja, tem-se a percepção de que a contribuição de tráfego P2P não é relevante, visto que seu volume gerado é substancialmente menor do que o tráfego gerado por aplicações *Web*.

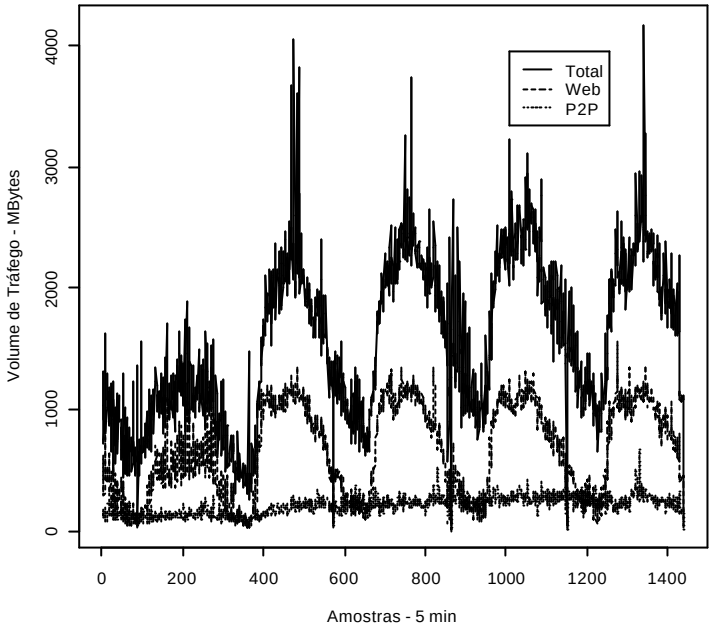

Figura 17: Perfil de Tráfego em Volume Transferido (Mbytes) no POP-SP, 02 a 08/Nov, Média de 5 minutos

Uma outra avaliação que pode ser feita é através da observação da função de distribuição acumulada empírica ECDF (*Empirical Cumulated Distribution Function*) da vazão medida por tipo de tráfego no intervalo considerado. Na Figura 18 pode ser visto que a distribuição da vazão de tráfego difere entre eles de forma clara. Observa-se que 50% do tráfego *Web* contribui com até 20Mbps, enquanto que 50% do tráfego P2P contribui com até 6Mbps.

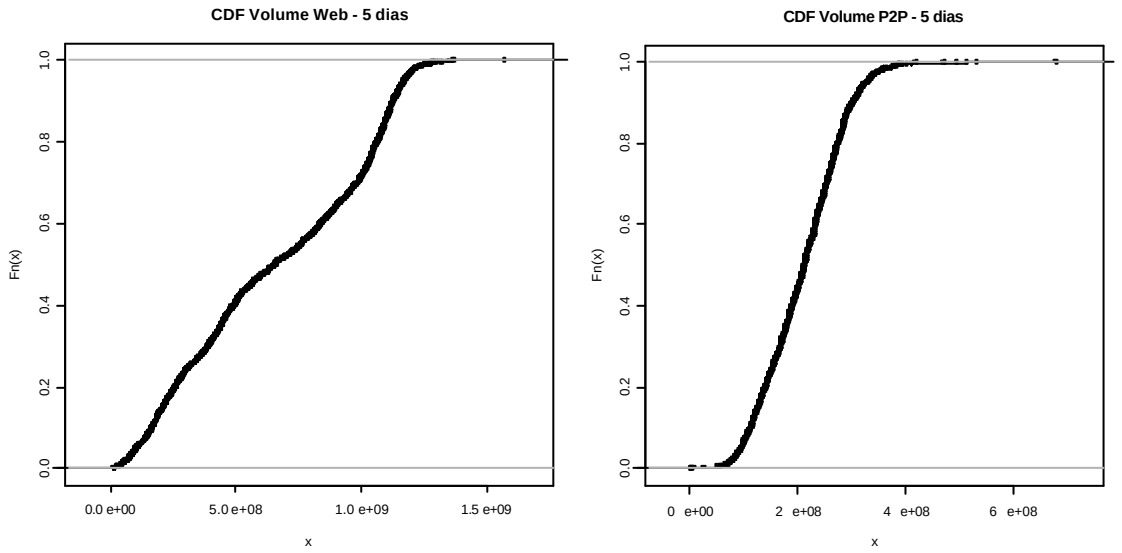

Figura 18: O gráfico da esquerda representa um ECDF do Volume de Tráfego Web, enquanto que o da direita representa um ECDF do Tráfego P2P

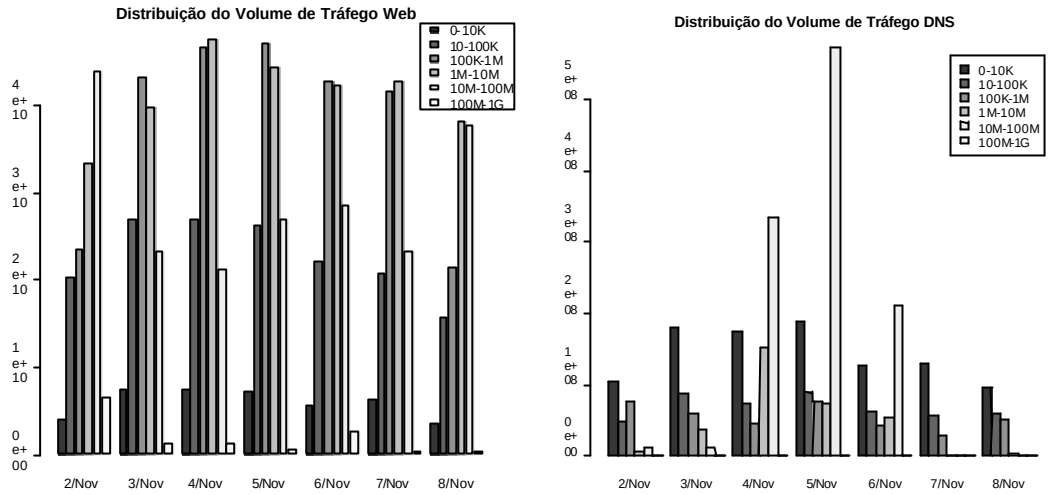

Figura 19: O gráfico da esquerda representa a distribuição do volume de tráfego na porta 80 (Web), enquanto que o da direita representa a distribuição do volume de tráfego na porta 53 (DNS)

Na Figura 19 é apresentada a análise da contribuição do volume de tráfego gerado por fluxos de aplicações na porta 80 (tipicamente tráfego WEB) de acordo com sua classificação nas faixas de 0 a 10KB, de 10KB a 100KB, de 100KB a 1MB, de 1MB a 10MB, de 10MB a 100MB e de 100MB a 1GB [20]. O perfil esperado do tráfego nesta porta, de acordo com [8], é de uma maior contribuição no volume de tráfego associada às duas primeiras faixas de classificação. Porém, como está bem claro neste resultado, o maior volume de tráfego gerado por dia durante toda a semana está associado às faixas de 100KB a 100MB. Na região de ocorrência de fluxos de tamanho entre 100KB a 10MB existe uma contribuição significativa ao volume de tráfego nesta porta. Como este comportamento é inesperado, isto é um forte indicativo de que as aplicações P2P estejam também utilizando esta porta para transferência de dados.

Além da porta 80, esta análise foi estendida para outras portas de serviços conhecidos, a saber nas portas 53, 22 e 443. Esses resultados estão descritos em detalhes em [42].

1.5.4. Simulação

Devido a natureza descentralizada e crescimento rápido das redes P2P, realizar testes em um ambiente real que utiliza essa tecnologia torna-se uma tarefa quase impossível. Entretanto é possível utilizar a simulação de uma rede P2P para avaliar aplicações e protocolos, em um ambiente controlado. A simulação também pode ser usada para fornecer um melhor entendimento de como as redes P2P funcionarão em um ambiente real.

Construir um simulador capaz de retratar fielmente o comportamento de uma rede P2P é uma tarefa bastante complicada. Existe, atualmente, uma série de simuladores para redes P2P, mas em geral cada rede possui um simulador próprio. Ainda não se chegou a um consenso sobre um sistema de simulação genérico para redes P2P, apesar de já ter sido proposto por [48].

Entretanto existe um consenso a cerca das principais características que precisam ser avaliadas ao se construir ou utilizar um simulador para redes P2P. Entre elas estão: distribuição de conteúdo, comportamento dos nós da rede e configuração da rede física.

1.5.4.1. Distribuição de conteúdo

A distribuição de conteúdo define que a dinâmica de uma rede P2P depende diretamente do volume e variedade dos arquivos compartilhados pelos nós. Logo, os simuladores devem utilizar esses atributos para retratar com maior fidelidade uma rede *peer-to-peer* real.

Para uma rede que possui pequeno volume de dados, ou seja, poucos nós compartilhando arquivos, as consultas tendem ser mais demoradas uma vez que precisam ser roteadas por um maior número de nós. Além disso, a carga no sistema fica desbalanceada sobrecarregando uma pequena parte da rede que dispõe de conteúdo [45].

O outro parâmetro que deve ser considerado é a variedade dos dados. Em geral as redes P2P obedecem a distribuições de probabilidade conhecida como leis de potência [30], que indicam que a maior parte dos nós está interessada em uma pequena parte do conteúdo disponível na rede. Portanto, para simular essa característica é preciso configurar diferentes tipos de arquivos atribuindo relevância específica para cada um deles.

1.5.4.2. Comportamento dos nós

O comportamento dos nós pode afetar a dinâmica de um sistema de diversas maneiras. Por exemplo, nós que se conectam e saem rapidamente da rede podem afetar significativamente o desempenho do sistema, uma vez que é necessário uma série de procedimentos para restaurar a rede, após a entrada ou saída de um nó [21].

Outro comportamento que pode afetar o sistema P2P é o número de *downloads* efetuados na rede, quando são realizados muitos *downloads* o desempenho da rede tende a ser prejudicado. Sendo assim, o simulador deve permitir a configuração da probabilidade de um dado arquivo ser recuperado da rede [21].

1.5.4.3. Configuração da rede física

Embora muitas demonstrações tenham mostrado que o desempenho de uma rede P2P depende diretamente das características da rede física, muitos simuladores existentes insistem em desconsiderar detalhes de configuração da rede física [21]. Isso se deve, em geral, ao fato de que é mais complexo avaliar um sistema P2P executando sobre uma rede que incluem detalhes de pacotes.

A topologia da rede é um dos parâmetros que precisam ser configurados pelo simulador. Ao se conectarem a rede P2P os nós tendem a seguir um padrão onde alguns nós possuem uma maior probabilidade de serem contatados por nós que acabam de entrar na rede. Além disso, dependendo do modelo de rede a ser utilizado (seção 1.1.4.) alguns nós podem assumir papéis diferentes na rede.

Outro parâmetro importante para as redes físicas é a largura de banda. Alguns nós possuem maior largura de banda e, devido a essa característica, possuem papéis e atribuições diferentes na rede. Em algumas arquiteturas de rede, nós que possuem baixa largura de banda não podem compartilhar arquivos, por exemplo.

1.5.4.4. O simulador GnutellaSim

GnutellaSim [17] é um simulador P2P que trabalha ao nível de pacotes, permitindo a avaliação completa da rede Gnutella (e outras rede P2P), com um modelo de rede detalhado. O simulador é baseado em um arcabouço que permite o isolamento de funções e uma estrutura centrada em protocolos. O arcabouço foi projetado para permitir a incorporação de diferentes alternativas de implementação para um determinado sistema P2P e pode ser usado em conjunto com diferentes simuladores de redes. Na verdade, GnutellaSim é apenas uma instância desse arcabouço, a primeira implementada usando as suas funcionalidades.

Em sua versão atual, o simulador GnutellaSim pode ser executado em conjunto com o simulador ns-2 [34], mas está sendo portado para outros simuladores de rede. Ele implementa as versões 0.4 e 0.6 do protocolo Gnutella [26].

O que pode ser extraído dessa seção é que a simulação de um sistema P2P antes dele ser avaliado na prática é fundamental para garantir o seu desempenho. Entretanto, para retratar com maior fidelidade as redes P2P, é necessário considerar todos os atributos que influenciam na configuração do sistema.

1.6. Considerações Finais

Esta seção explica alguns pontos sobre a questão social do uso de redes *peer-to-peer*, detalha os próximos passos da tecnologia e tece algumas considerações acerca da necessidade e importância dessas redes.

1.6.1. Questões Jurídicas e Impacto Social

Muito se tem falado sobre a ilegalidade de sistemas *peer-to-peer*, em especial os de compartilhamento de arquivos, e alguns sistemas já sofreram intervenções judiciais. Por exemplo, o Napster teve seu funcionamento interrompido e restaurado várias vezes.

Entretanto, os esforços de empresas e mesmo indústrias contra os sistemas de compartilhamento de arquivos se dá devido à facilidade de compartilhar arquivos ilegalmente, realizada por usuários que disponibilizam arquivos indevidamente (i.e., programas, músicas, filmes, etc., com direitos de uso, execução ou exibição restritos), e não por ilegalidade nos sistemas de compartilhamento em si, o que não ocorre. Os sistemas *peer-to-peer* de compartilhamento de arquivos não têm como controlar a origem dos arquivos compartilhados e dessa forma deixam para o usuário esse controle. Assim, o uso de tais sistemas dificulta a descoberta e mesmo a coleta de provas contra usuários que disponibilizam arquivos indevidamente.

Um dos debates ainda em aberto é sobre o anonimato dos usuários de sistemas *peer-to-peer*. O conhecimento da identidade do usuário traz problemas de segurança para o mesmo, sua máquina e sua rede; mas o anonimato facilita o uso dos recursos de maneira ilegal. Muitos dos sistemas implementados vêem o anonimato como algo positivo para os usuários, mas não o implementam completamente, tornando possível a *hackers* identificar a origem dos usuários.

Apesar desses problemas, o uso de redes *peer-to-peer* representa um passo importante para a democratização de conhecimento e a socialização de recursos. Qualquer

que seja o foco da rede em questão (compartilhamento de arquivos, troca de mensagens, conferências de áudio ou vídeo, computação colaborativa, compartilhamento de processador, etc), como na rede as máquinas podem ter o mesmo nível de importância e prioridade, qualquer usuário da rede *peer-to-peer* pode ter acesso aos recursos compartilhados por ela. Isso permite, por exemplo:

- A uma pessoa "publicar" um documento ou um programa que não seria possível nos meios tradicionais sem recursos para essa publicação e sem ser passível de censura;
- A uma junta médica se reunir para conversar sobre casos difíceis de pacientes sem precisar pagar o acesso caro a uma rede de videoconferência;
- A grupos de interesse em escolas trocar experiências em matérias específicas;
- A uma instituição de pesquisa realizar simulações de grande porte com o acesso a processadores ociosos compartilhados.

Como país industrializado em desenvolvimento, o Brasil pode aproveitar bastante o uso de redes *peer-to-peer* para os casos citados e muitos outros, auxiliando sua população a trocar idéias e informações e acessar recursos que não seriam possíveis de outra maneira, em especial pessoas que não têm acesso fácil a computador (por exemplo, só na escola ou no trabalho). Muitas instituições de pesquisa com poucas verbas podem se beneficiar do poder de processamento de redes com processadores compartilhados.

1.6.2. Direções Futuras

Alguns problemas em redes *peer-to-peer* que não foram resolvidos pela comunidade acadêmica nem pela indústria estão sendo estudados no momento e são descritos aqui como assuntos relevantes de pesquisa.

A grande maioria das redes P2P em utilização não foi simulada em cenários de grande escala (de tamanho próximo à Internet) nem em condições que previam muitos defeitos (alta perda de conectividade, travamento da máquina, baixíssima taxa de transmissão, etc). Isso fez com que algumas aplicações tivessem seu desempenho prático muito inferior ao previsto antes de sua implantação [7], gerando a necessidade de alterações em programas e até mesmo em sua arquitetura e de publicação de várias correções para os programas.

A maioria dos sistemas P2P requer um comportamento colaborativo dos participantes, onde os benefícios gerados pelos participantes que colaboram (por exemplo, participante que compartilha o acesso a seu processador quando ocioso) são divididos entre os participantes que os requisitam (por exemplo, o participante que joga um processo na rede para execução no número máximo de máquinas possível). Assim, se a quantidade de recursos compartilhados for muito menor que a demanda, haverá muita demanda reprimida, o que acarretará em um funcionamento lento (por exemplo, para conseguir alocar um processo em um processador ocioso) da rede e, conseqüentemente, na sua perda de importância. Dessa forma, alguns trabalhos estudam formas de compensação para usuários que contribuem com recursos para a rede. Essa compensação pode ser financeira, em tempo de acesso a recurso, em prioridade de acesso a recurso ou em quantidade de recurso acessível simultaneamente Algumas aplicações que implementam redes P2P (como eDonkey2000, seção 1.4.3) usam

esquemas próprios de pontuação para os usuários que mais compartilham recursos, dando a estes uma maior prioridade no acesso a outros recursos compartilhados.

Alguns problemas de segurança no uso de redes P2P ainda não estão resolvidos, embora haja várias propostas para resolvê-los. O anonimato, quando devidamente implementado, traz o problema de confiança no recurso compartilhado. Esse problema pode ser parcialmente resolvido com qualificação (reputação) de uns usuários pelos outros. Entretanto, como usuários novos não têm qualificação, um usuário malicioso pode simplesmente entrar na rede sempre como um novo usuário (o que é possível pelo anonimato). Assim, a detecção de usuários maliciosos torna-se difícil e muito dependente do tipo de recurso compartilhado. No caso de troca de mensagens ou de computação colaborativa, o anonimato perde o sentido, pois impossibilita a não-repudiação, por exemplo, não há como provar que as mensagens enviadas por um usuário foram realmente enviadas por ele, exceto se houver um esquema de autenticação. Também é muito difícil detectar identidades múltiplas (por exemplo, a mesma pessoa acessando a rede fazendo-se passar por diferentes usuários) sem relacionar a identidade virtual a uma identificação real do usuário, através de um documento físico ou biometria.

A maior parte das redes públicas P2P são baseadas em arquiteturas descentralizadas, onde a publicação e busca de conteúdo não é feita em todos os nós, porque não há uma organização estruturada entre eles. Entre essas redes se destacam Gnutella, KaZaA e eDonkey2000. As redes estruturadas são representadas por modelos baseados em DHT, onde a busca e inserção sempre levam em consideração todos os nós da rede. Modelos DHT apresentam muitas vantagens teóricas (por isso são mais apreciados pela comunidade acadêmica), mas também têm desvantagens. Entre elas, a principal é a falta de localidade no armazenamento das informações. Uma tabela *hash* pode escolher qualquer nó da rede para armazenar a informação, independente de sua localização geográfica. Essa característica leva fatalmente a problemas de desempenho, pois as transferências de documentos e informações percorrem um maior número de roteadores, potencialmente gerando maiores congestionamentos, assim como sendo influenciadas negativamente por eles.

Algumas redes públicas, no entanto, já estão optando por soluções estruturadas, baseadas em DHT. Este é o caso da rede Overnet (seção 1.4.3) que está substituindo aos poucos a rede eDonkey. No exato momento em que este documento está sendo escrito, o Overnet está com aproximadamente 1,6 milhões de usuários conectados, enquanto que o KaZaA conta com mais de 4 milhões. Uma avaliação simples (não científica) mostra que o resultado das buscas é mais rápido no Overnet, mas a transferência de arquivos mais lenta. Além disso, o KaZaA retorna um maior número de respostas, embora o mecanismo de busca não pesquise em todos os nós conectados.

1.6.3. Conclusões

As redes P2P não são novidade para os usuários mais antigos da Internet. A principal diferença das atuais redes P2P e do modelo original da Internet (seção 1.1.1) é a possibilidade de qualquer computador, inclusive (e principalmente) aqueles instalados nas residências dos usuários, participarem ativamente da rede. Por isso, é esperado que com o amadurecimento da tecnologia, muitos tipos novos de redes P2P devam surgir. Resta

descobrir a resposta para a seguinte pergunta: as atuais redes públicas P2P são um sucesso devido à tecnologia ou ao conteúdo grátis (freqüentemente infringindo os direitos autorais)?

A grande vantagem das redes P2P está na inerente escalabilidade e disponibilidade de informações, servidas por uma grande quantidade de *peers* em vez de alguns servidores específicos. No entanto, não se pode dizer que o modelo P2P vai suplantar o modelo tradicional cliente/servidor. Aplicações críticas ainda não podem ser implementadas com o modelo P2P, uma vez que ele apresenta problemas de confiabilidade, devido à conectividade variável das máquinas participantes e à administração descentralizada. O modelo cliente/servidor permite que servidores de alta disponibilidade e segurança forneçam informações confiáveis, característica indispensável para o mundo corporativo. Para o usuário doméstico, ao contrário, freqüentemente uma grande quantidade de informações (documentos, software, músicas, filmes, etc.) é mais importante do que uma informação específica.

Referências

[1] ANDERSEN, D., BALAKRISHNAN, H., KAASHOEK, F. & MORRIS, R., "Resilient Overlay Networks", 18th ACM Symposium on Operating Systems Principles, Banff/Canada, Outubro de 2001.

[2] BARAK, A. et WHEELER, R., "MOSIX: An Integrated Multiprocessor UNIX", Winter USENIX Conference, Fevereiro de 1989.

[3] BARKAY, D., "Peer-to-Peer Computing: Technologies for Sharing and Collaborating on the Net". Intel Press, Agosto de 2001.

[4] BECKER, D., et al., "Beowulf: A ParallelWorkstation for Scientific Computation", 24º International Conference on Parallel Processing, 1995.

[5] BOSSELAERS, A., et al., "SHA: a design for parallel architectures?", Proceedings Eurocrypt´97, 1997.

[6] BROOKSHIER, D., GOVONI, D., KRISHNAN, N. "JXTA: Java P2P Programming". Sams. Março de 2002.

[7] CHAWATHE, Y., RATNASAMY, S., BRESLAU, L., LANHAM, N., SHENKER, S., "Making Gnutella-like P2P Systems Scalable", ACM SIGCOMM 2003, Karlsruhe, Alemanha, Agosto de 2003.

[8] CHOI, Hyoung-Kee, LIMB, John O., "A Behavioral Model of Web Traffic", 7th Annual IEEE International Conference on Network Protocols (ICNP), Canada, Outubro de 1999.

[9] CISCO, "NetFlow Services and Applications", White Paper, http://www.cisco.com/warp/public/cc/pd/iosw/ioft/neflct/tech/napps_wp.htm, acessado em Novembro 2003.

[10] CLARKE, I., et al, "Protecting Free Expression Online with Freenet", IEEE Internet Computing, Janeiro de 2002.

[11] CRESPO, A. et GARCIA-MOLINA, H., "Routing indices for peer-to-peer systems". In Proceedings of the 22nd International Conference on Distributed Computing Systems, pages 23-30, Vienna, Austria, Julho de 2002.

[12] CROWCROFT, Jon, MORETON, Tim, PRATT, Ian, TWIGG, Andrew. "Peer-to-Peer Systems and the Grid", draft, University of Cambridge Computer Laboratory, 2003.

[13] DABEK, F., KAASHOEK, M. F., KARGER, D., MORRIS, R. et STOICA, "Wide-area cooperative storage with CFS". In 18th ACM Symposium on Operating Systems Principles, Outubro de 2001.

[14] FOSTER, I., "Internet Computing and the emerging Grid", Nature, Junho de 2000.

[15] FOX, Geoffrey. "Peer-To-Peer Networks", IEEE Computing in Science & Engineering Magazine, Maio-Junho 2001.

[16] GE, Z., FIGUEIREDO, D., JAISWAL, S., KUROSE, J. et Towsley, D., "Modeling Peer-Peer File Sharing Systems", IEEE INFOCOM 2003, São Francisco, CA, USA, Março de 2003.

[17] GNUTELLASIM, "Packet-level Peer-to-Peer Simulation Framework and GnutellaSim", http://www.cc.gatech.edu/computing/compass/gnutella/.

[18] Groove Networks, "Groove Platform Development Kit (GDK)", http://www.groove.net/devzone/default.cfm?pagename=Platform_GDK.

[19] GTP2P, Grupo de Trabalho em Computação Colaborativa da RNP. http://www.cin.ufpe.br/~gprt/gtp2p. Fevereiro de 2004.

[20] GUMMADI, K., DUNN, R., SAROIU, S., GRIBBLE, S., LEVY, H., ZAHORJAN, J., "Measurement, Modeling, and Analysis of a Peer-to-Peer File-Sharing Workload", 19th ACM Symposium on Operating Systems Principles (SOSP-19), Bolton Landing, NY, USA, Outubro de 2003.

[21] HE, Qi, AMMAR, Mostafa, RILEY, George, RAJ, Himanshu, FUJIMOTO, Richard. "Mapping Peer Behavior to Packet-Level Details: A Framework for Packet-Level Simulation of Peer-to-Peer Systems". In 11th IEEE/ACM International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS), Outubro de 2003.

[22] IRTF, "Peer-to-Peer Research Group", http://www.irtf.org/charters/p2prg.html.

[23] JXTA Project, http://www.jxta.org, Fevereiro de 2004.

[24] JOVANOVIC, M., ANNEXSTEIN, F. et BERMAN, K., "Modeling Peer-to-Peer Network Topologies through 'Small-World' Models and Power Laws", IX Telecommunications Forum, Dezembro de 2001.

[25] KALOGERAKI, V., GUNOPULOS, D. et ZEINALIPOUR-YAZTI, D., "A local search mechanism for peer-to-peer networks". In Proceedings of the eleventh international conference on Information and knowledge management, pages 300--307. ACM Press, Fevereiro de 2002.

[26] KLINGBERG, T. et MANFREDI, R., “Gnutella 0.6”, Draft, http://rfc-gnutella.sourceforge.net/src/rfc-0_6-draft.html, Junho de 2002.

[27] LITZKOW, M., LIVNY, M., et MUTKA, M. “Condor – A Hunter of Idle Workstations”, 8th International Conference on Distributed Computing Systems, Junho de 1988.

[28] LUDVIGSON, T., “Groove Networks: Making P2P a Reality”, Case #6-0008, Tuck School of Business at Dartmouth, http://mba.tuck.dartmouth.edu/pdf/2002-6-0008.pdf, Janeiro de 2002.

[29] LV, Q., et al., “Search and Replication in Unstructured Peer-to-Peer Networks”, 16º ACM International Conference on Supercomputing(ICS'02), Junho de 2002.

[30] MEDINA, A., MATTA, I., BYERS, J. “On the origin of power laws in internet topologies”, technical report, Computer Science Department, Boston University, Abril de 2000.

[31] Microsoft .NET Homepage. http://www.microsoft.com/net/, Outubro de 2003.

[32] Microsoft, “.NET P2P: Writing Peer-to-Peer Networked Apps with the Microsoft .NET Framework”, MSDN Magazine, http://msdn.microsoft.com/msdnmag/issues/01/02/netpeers/, Fevereiro de 2001.

[33] MILOJICIC, Dejan S., KALOGERAKI, Vana, LUKOSE, Rajan, NAGARAJA, Kiran, PRUYNE, Jim, RICHARD, Bruno, ROLLINS, Sami, XU, Zhichen, “Peer-to-Peer Computing”, technical report, HP Laboratories Palo Alto, Março de 2002.

[34] Network Simulator (ns-2, versão 2.27), http://www.isi.edu/nsnam/ns/.

[35] ORAM, A., “Peer-to-Peer for Academia”. Published on The O’Reilly Peer-to-Peer & Web Services Conference, Outubro de 2001.

[36] ORAM, A., “Peer-to-Peer: Harnessing the Power of Disruptive Technologies”, O’Reilly, 2001.

[37] P2P Architect Project, ““Ensuring dependability of P2P applications at architectural level”, Deliverable D1, Abril de 2002, http://www.atc.gr/p2p_architect/results/0101F05_P2P Survey.pdf.

[38] PARAMESWARAN, M., SUSARLA, A., WHINSTON, A., “P2P Networking: An Information-Sharing Alternative”. IEEE Computer, Julho de 2001.

[39] PITZER, Bill, “P2P with Groove”, WebTechniques, http://www.catdancers.com/webmags/webtech/2001/05/stratrevu, Fevereiro de 2004.

[40] RATNASAMY, S., FRANCIS, P., HANDLEY, M., KARP, R., SHENKER, S., “A Scalable Content-Addressable Network”. ACM SIGCOMM 2001, Agosto de 2001.

[41] ROWSTRON, A. et DRUSCHEL, P., "Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems", in Proc. 18th IFIP/ACM International Conference on Distributed Systems Platforms (Middleware), Heidelberg, Germany, pages 329-350, Outubro de 2001.

[42] SADOK, Djamel, KAMIENSKI, Carlos, FERNANDES, Stênio, SILVESTRE, Guthemberg, DIAS, Kelvin, ROCHA, João. "Análise de Tráfego P2P no Backbone da RNP", 22º Simpósio Brasileiro de Redes de Computadores (SBRC 2004) , Maio 2004.

[43] SCHOLLMEIR, R., "Peer-to-Peer Networking. Applications for and Impacts on Future IP-Based Networks". 3. ITG Fachtagung Netze und Anwengungen, Duisburg, Germany, Março de 2002.

[44] SCHLOSSER, Mario T., KAMVAR, Sepandar D., "Modeling Interactions in a P2P Network", First Workshop on Semantics in P2P and Grid Computing, Dezembro de 2002.

[45] SCHLOSSER, M. T., CONDIE, T. E., KAMVAR, S. D., "Simulating a P2P File-Sharing Network"; First Workshop on Semantics in P2P and Grid Computing, USA, Dezembro de 2002.

[46] STOICA, I., MORRIS, R., KARGER, D. R., KAASHOCK, M. Frans et BALAKRISHMAN, H., "Chord: A scalable peer-to-peer lookup protocol for internet applications". In Proceedings of the ACM SIGCOMM, pages 149--160, San Diego, California, Agosto de 2001.

[47] STRIBLING, J., RHEA, S., JOSEPH, A. et KUBIATOWICZ, J. "Tapestry: A Resilient Global-scale Overlay for Service Deployment", IEEE Journal on Selected Areas in Communications, Janeiro 2004.

[48] TING, Nyik San, DETERS, Ralph. "A Generic Peer-to-Peer Network Simulator", technical report, Department of Computer Science, University of Saskatchewan, Junho de 2003.

[49] TOWSLEY, D., "Peer-peer Networking", Tutorial no SBRC2003, http://www.cin.ufpe.br/~gprt/gtp2p/tutoriais/p2p03-tutorial_towsley.pdf, Maio de 2003.

[50] TRAN, D., HUA, Kien, DO, Tai, "ZIGZAG: An Efficient Peer-to-Peer Scheme for Media Streaming", IEEE INFOCOM 2003, São Francisco, CA, USA, Março de 2003.

[51] TRIANTAFILLOU, P., XIRUHAKI, C., KOUBARAKIS, M. et NTARMOS, N. "Towards High Performance Peer-to-Peer Content and Resource Sharing Systems", Proceedings of CIDR, Janeiro de 2003.

[52] WALDMAN, M., RUBIN, A. et CRANOR, L., "Publius: A Robust, Tamper-Evident, Censorship-Resistant Web Publishing System", 9ª USENIX Security Symposium, Agosto de 2000.

[53] YANG, B. et GARCIA-MOLINA, H. "Efficient Search in Peer-to-Peer Networks". I Proc. Intl. Conf. On Distributed Computing Systems (ICDCS), Vienna, Austria, Julho de 2002.

[54] ZHANG H., GOEL, A. et GOVINDAN, R., "Using the small-world model to improve freenet performance". In Proceedings of the IEEE INFOCOM, New York, NY, USA, 2002.