PROPOSTA DE UMA ARQUITETURA PAR-A-PAR
ORGANIZADA POR ÍNDICES

Gabriel Epsztejn

DISSERTAÇÃO SUBMETIDA AO CORPO DOCENTE DA COORDENAÇÃO DOS PROGRAMAS DE PÓS-GRADUAÇÃO DE ENGENHARIA DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE MESTRE EM CIÊNCIAS EM ENGENHARIA ELÉTRICA.

Aprovada por:

Prof. Otto Carlos Muniz Bandeira Duarte, Dr.Ing.

Prof. Bruno Richard Schulze, D.Sc.

Prof. Luís Henrique Maciel Kosmalski Costa, Dr.

Prof. Marcelo Gonçalves Rubinstein, D.Sc.

RIO DE JANEIRO, RJ - BRASIL

MARÇO DE 2007

EPSZTEJN, GABRIEL

Proposta de uma Arquitetura Par-a-Par Organizada por Índices [Rio de Janeiro] 2007

XII, 75 p. 29,7 cm (COPPE/UFRJ, M.Sc., Engenharia Elétrica, 2007)

Dissertação - Universidade Federal do Rio de Janeiro, COPPE

1. Arquiteturas P2P

2. Computação em Grade

3. Buscas por Intervalos

I. COPPE/UFRJ II. Título (série)

À minha família.

Agradecimentos

Agradeço a Deus, por me permitir alcançar mais esse degrau em minha vida. A meus pais, David e Ruth, por todo amor, incentivo e orientação passados em todos os anos de minha vida. Aos meus irmãos, Michel e Daniela, pela amizade incondicional.

Ao professor e orientador Otto, responsável por grande parte da minha formação pessoal, acadêmica e profissional, por sua amizade, conselhos e orientação.

Aos amigos e colegas de graduação Rafael, Igor e Marco, pelos bons momentos passados tanto nas horas de trabalho quanto nas de lazer.

Resumo da Dissertação apresentada à COPPE/UFRJ como parte dos requisitos necessários para a obtenção do grau de Mestre em Ciências (M.Sc.)

PROPOSTA DE UMA ARQUITETURA PAR-A-PAR
ORGANIZADA POR ÍNDICES

Gabriel Epsztejn

Março/2007

Orientador: Otto Carlos Muniz Bandeira Duarte
Programa: Engenharia Elétrica

Observou-se, nos últimos anos, uma rápida adoção pelos usuários da Internet de serviços baseados em redes P2P, principalmente para compartilhamento de arquivos. Esse crescimento levou ao interesse pela expansão dos serviços ofertados a novas aplicações, tendo como ponto de partida as de fins amadores, como as redes de trocas de arquivos musicais, que não possuem garantias de qualidade de serviço ou otimização de desempenho. A utilização de redes P2P para fins profissionais, como grades computacionais e infraestrutura de armazenamento distribuído de dados, impõe uma série de novos requisitos à tecnologia. Dessa forma, apresenta-se como desafio a implementação de arquiteturas P2P escaláveis, que associem à característica de descentralização a possibilidade de execução de buscas complexas por recursos. Neste trabalho, é proposta uma arquitetura para redes P2P em que a organização dos nós na rede se baseia em índices de busca, tendo como objetivo viabilizar a incorporação desses novos requisitos. São apresentados ainda resultados analíticos e de simulação para a rede, bem como comparações de desempenho com propostas anteriores.

Abstract of Dissertation presented to COPPE/UFRJ as a partial fulfillment of the requirements for the degree of Master of Science (M.Sc.)

Proposal of an Index-Organized Peer-to-Peer Architecture

Gabriel Epsztejn

March/2007

Advisor: Otto Carlos Muniz Bandeira Duarte

Department: Electrical Engineering

The adoption by Internet users of services based on P2P networks has shown, in the last years, a fast expansion, mostly for file-sharing. This growth has led to the interest in expanding the offered services to new applications, from those with amateur goals, such as the sharing of music files, which do not support the assurance of quality of service nor optimal performance. The use of P2P networks for professional services, such as computational grids and distributed data storage, imposes a group of new requisites on the technology. In this context, the challenge of implementing scalable P2P architectures, that associate to the features of complete decentralization, the execution of complex queries for resources, emerges. In this work, a P2P architecture is proposed, in which the nodes organization in the network is based on search indexes, as a way to support these new requisites. Analytical and simulation results are presented, as well as performance compairisons with previous proprosals.

Sumário

Lista de Figuras

Lista de Acrônimos

CAN : *Content Addressable Network*;

DHT : *Distributed Hash Tables*;

FTP : *File Transfer Protocol*;

IP : *Internet Protocol*;

P2P : *Peer to Peer*;

PHT : *Prefix Hash Table*;

PKI : *Public Key Infrastructure*;

QoS : *Quality of Service*;

SOAP : *Simple Object Access Protocol*;

TCP : *Transmission Control Protocol*;

TTL : *Time to Live*;

XML : *Extensible Markup Language*;

Capítulo 1

Introdução

OBSERVOU-SE , nos últimos anos, um rápido crescimento na utilização de serviços baseados em redes par-a-par (*peer-to-peer* - P2P). A facilidade para iniciar e aderir a tais redes e a possibilidade de utilização de recursos ociosos [1], sem custos aparentes, levou este tipo de serviço a alterar de forma significativa os padrões de tráfego na Internet. Segundo medições de provedores de acesso nos EUA e Europa, em 2005, as aplicações P2P para compartilhamento de arquivos já eram responsáveis por 71% do tráfego na Internet [2, 3], com efeitos semelhantes em redes nacionais [4]. Redes voltadas a trocas de arquivos de áudio, como Napster, Audiogalaxy e KaZaA, foram as responsáveis pela popularização dos serviços P2P, tendo sido freqüentemente combatidas, por permitirem a quebra de direitos autorais por seus usuários. As aplicações dessas redes foram rapidamente estendidas, sendo utilizadas para a troca, muitas vezes ilegal, de quaisquer tipos de arquivos, de filmes a programas corporativos. Atualmente, estão disponíveis na Internet serviços P2P para diversas finalidades, como execução de processamento distribuído [5], comunicação instantânea [6] e armazenamento de dados em rede [7]. Entretanto, a implementação de aplicações emergentes, como bibliotecas virtuais, bancos de dados distribuídos e tratamento de informações geo-referenciadas, impõe novos requisitos às redes P2P, por exigirem a execução de buscas mais complexas por recursos, como buscas por atributos e semânticas, e garantias mínimas de qualidade de serviço (*Quality of Service* - QoS).

1.1 Motivação

Enquanto grandes máquinas servidoras de uma empresa trabalham em seu limite máximo de processamento, centenas de máquinas residenciais estão desligadas. Enquanto sítios de Internet brasileiros encontram dificuldades para atender a requisições em momentos de pico, servidores web no outro lado do mundo trabalham com ociosidade. Com a proliferação e barateamento dos recursos para processamento de dados, passou-se a dispor a todo momento, em nível global, de grande quantidade de recursos computacionais ociosos. Apresenta-se então como uma oportunidade implementar formas de compartilhar os recursos disponíveis, permitindo a disponibilização, a baixo custo, de altos níveis de serviço. Neste sentido, duas importantes linhas de pesquisa se desenvolveram: grades computacionais (*grids*) e redes P2P. Embora com muitas similaridades, as propostas baseiam-se em premissas distintas.

Os estudos na área de computação em grade tiveram como motivação inicial requisitos de comunidades profissionais, para acesso a recursos remotos para processamento em larga escala e bases de dados distribuídas. Assim, foram tratadas principalmente questões de localização e distribuição otimizada de recursos, bem como paralelismo de tarefas. Entretanto, não houve como premissa a descentralização completa destas funções na rede, podendo haver nós com responsabilidades únicas, como organizações hierárquicas para localização de recursos. Outra preocupação presente foram questões de segurança, tendo sido implementados mecanismos de autenticação, que consideram um grupo restrito de usuários e de nós confiáveis participando da rede. De forma geral, as aplicações de grades computacionais consideram a participação de nós de alta persistência na rede [8].

Em contrapartida, o desenvolvimento de redes P2P foi impulsionado pela popularização de serviços de finalidades amadoras específicas, como compartilhamento de arquivos musicais, que antecederam os estudos mais aprofundados. Assim, tiveram como premissa a inexistência de monitoração e controle centralizados. Outro aspecto importante foi a consideração da participação de um volume muito maior de nós, mais voláteis e de baixa capacidade, como as máquinas pessoais tipicamente utilizadas por usuários da Internet. Por outro lado, questões referentes a otimização na utilização de recursos da rede, balanceamento de carga, qualidade de serviço ou suporte a diversos tipos de recursos,

foram abordadas apenas em segundo plano [9].

Ao associar a localização dos nós na rede à responsabilidade por um conjunto de recursos, a abordagem por tabelas *hash* distribuídas (*Distributed Hash Tables* - DHTs) representou uma importante evolução para redes P2P em comparação às arquiteturas não-estruturadas anteriormente adotadas. No entanto, tal abordagem permite apenas buscas baseadas em identificadores únicos, o que dificulta a oferta de novos serviços que requeiram buscas mais complexas, como descoberta de recursos por intervalos de valor de atributos ou valores máximos e mínimos.

Torna-se claro, então, o interesse por associar a infra-estrutura oferecida por redes P2P a serviços tipicamente implementados por arquiteturas em grade. Para tanto, tornam-se necessárias adaptações às arquiteturas P2P existentes de forma a permitir a oferta de serviços mais avançados e confiáveis, utilizando recursos dispersos e pouco poderosos. Ainda que algumas propostas, como a feita por Chunqiang Tang *et al.* [10], tenham buscado enriquecer as funcionalidades das redes baseadas em DHT, ainda não há uma arquitetura definitiva que forneça todos os requisitos demandados.

1.2 Objetivo

Este trabalho propõe e avalia, por análises teóricas e simulações, uma arquitetura para redes P2P que permite a execução de buscas complexas por recursos e flexibilidade para suportar a convivência de diversos tipos de recurso na mesma rede, abrindo ainda espaço para a implementação de garantias de qualidade de serviço. Isto deve ser feito mantendo a descentralização completa de todas as tarefas e a escalabilidade com relação ao número de nós participantes. Para tanto, é necessário associar diversos conceitos: a utilização de metadados para descrição de recursos, organização dos nós na rede de forma adequada à execução de buscas e o uso de algoritmos de replicação e balanceamento de carga. Neste trabalho, são associadas diversas propostas isoladas anteriores a uma nova forma de organização dos nós.

1.3 Contribuições

Neste trabalho, é apresentada uma nova proposta de arquitetura P2P, visando a expansão de seu uso a novos serviços. A organização dos nós na rede se baseia na criação de um espaço d-dimensional, conforme proposto por Ratnasamy *et al.* [11]. Entretanto, para permitir buscas complexas, as dimensões são associadas a índices com relevância para buscas, em substituição a utilização de funções *hash* para mapeamento de recursos na rede. Desta forma, torna-se possível a execução de buscas por intervalos de valor, bem como por máximos e mínimos.

São ainda considerados os diversos impactos ocasionados por essa alteração no desempenho da rede, principalmente no que tange a eficiência das buscas e balanceamento de carga. Desta forma, são feitas propostas para contornar os impactos negativos identificados. Além da análise de questões fundamentais é feita ainda a proposta de técnica para permitir a utilização de uma mesma rede para compartilhamento de diversos tipos de recurso, questão geralmente não abordada em propostas anteriores de arquiteturas.

Para verificação do desempenho da rede, são deduzidas as expressões matemáticas para o número médio de nós envolvidos nas buscas, em diversas configurações. São ainda apresentados os resultados de simulação para a rede, sendo comparados às previsões teóricas.

É feita também uma revisão do estado da arte em redes P2P, que permite a classificação e identificação de vantagens e limitações individuais. Este levantamento permite a comparação da nova proposta com as arquiteturas anteriores, bem como a identificação de pontos que permanecem em aberto na viabilização de uma arquitetura definitiva.

1.4 Organização

Este trabalho está organizado da seguinte forma. No Capítulo 2, são apresentadas as diversas arquiteturas P2P atualmente disponíveis, com uma análise baseada em critérios pré-definidos e posicionamento histórico das propostas. São levantados os parâmetros de busca suportados, completude da busca e garantias de nível de serviço em cada um

dos tipos de arquitetura existentes. Em seguida, no Capítulo 3, é apresentada a arquitetura P2P proposta, denominada Rede P2P Organizada por Índices. É descrita a forma de organização dos nós na rede, bem como os diversos algoritmos de busca suportados. Também são abordados os desafios identificados para implementação da proposta e sugeridas técnicas para sobrepujá-los. No Capítulo 4, são apresentadas as análises teóricas de desempenho, bem como o resultado das simulações realizadas. As simulações buscam explorar principalmente a alteração do desempenho da rede mediante mudanças em diversas configurações, e comparar os resultados obtidos com propostas anteriores. Por fim, no Capítulo 5, são descritas as conclusões alcançadas e pontos abertos para trabalhos futuros.

Capítulo 2

Conceitos e Arquiteturas

NESTE capítulo, é feita inicialmente uma revisão do conceito de redes par-a-par (P2P), sendo apresentados os requisitos gerais dos serviços que se baseiam nesta tecnologia. São apresentadas e avaliadas as diversas arquiteturas disponíveis atualmente.

2.1 Definição

O termo par-a-par (*peer-to-peer* - P2P) tem sido associado a um amplo conjunto de aplicações, não havendo uma definição formal única para o termo. Segundo definição de Miller [12], corresponde a sistemas e aplicações distribuídos onde não há nenhum controle centralizado ou organização hierárquica, onde a aplicação sendo executada em cada um dos nós é equivalente em funcionalidade, e a comunicação é feita sem intermediários. Entretanto, essa definição exclui diversos sistemas popularmente aceitos como P2P, por possuírem nós com funcionalidades especiais. Por exemplo, a primeira aplicação de grande popularidade a ser rotulada como P2P, o sistema de troca de arquivos musicais Napster, não se enquadra perfeitamente nessa classificação. Seu funcionamento se baseia na existência de um servidor central contendo o catálogo de todos os arquivos disponibilizados para compartilhamento e responsável pela execução do serviço de busca.

De fato, o fundamento por trás da tecnologia P2P é prover uma alternativa ao modelo cliente-servidor, que compreende uma clara divisão de funções e centraliza a responsabi-

lidade pela disponibilidade dos serviços. Assim, a característica fundamental e comum aos sistemas atualmente classificados como P2P é proverem aplicações cooperativas, em que os nós participantes podem atuar tanto como clientes como provedores de recursos.

A sigla P2P tornou-se popular pela utilização desta tecnologia em aplicações de compartilhamento de arquivos na Internet, muitas vezes de forma ilegal. Entretanto, esta aplicação é apenas um caso específico de utilização de tecnologia P2P, que tem sido utilizada para fornecimento de serviços com diversas finalidades, podendo ser classificadas em três grandes grupos:

- Compartilhamento de Conteúdo - Esta é a aplicação mais comum para redes P2P, incluindo os serviços P2P mais populares de troca de arquivos, como Napster [13, 14] e Gnutella [15];
- Computação Distribuída - Nesta categoria, o objetivo é permitir que os nós da rede se sirvam da capacidade de processamento disponível em outros nós para execução de funções que, de outra forma, teriam que ser executadas localmente. O exemplo mais conhecido de tal aplicação é o sistema SETI@home [5];
- Comunicação e Colaboração - A utilização de serviços P2P para aprimorar a colaboração entre pessoas é mais recente, e inclui serviços de mensageria instantânea e comunicação por voz, como no sistema Skype [16].

Considerando-se a diversidade de sistemas P2P existentes, são inicialmente classificados em relação ao nível de descentralização dos serviços, podendo ser divididos em topologias P2P pura e híbrida. Há ainda diversas formas de organização das redes de sobreposição, sendo classificadas em redes não-estruturadas e estruturadas.

2.1.1 Topologias P2P Pura e Híbrida

A primeira classificação aplicável a arquiteturas P2P se refere ao nível de distribuição dos serviços e igualdade de funções entre os nós. Conforme mencionado na seção anterior, a rede Napster continha um servidor central para catalogação, ainda que os arquivos compartilhados estivessem armazenados de forma distribuída entre milhares de

nós na Internet. Este tipo de topologia foi posteriormente classificada como P2P híbrida, possuindo também diversas variantes [17]. Embora fosse claramente dependente de um servidor centralizado, este serviço já se mostrava muito mais distribuído e ágil na oferta de conteúdo que os tradicionais serviços cliente-servidor baseados em FTP disponíveis à época. A mesma topologia é utilizada pelo sistema SETI@home [5], que possui um servidor centralizado para a distribuição da execução de complexos algoritmos de tratamento de sinais de rádio e imagens espaciais entre milhares de computadores residenciais, em busca de inteligência extraterrestre. É apresentada na Figura 2.1 uma ilustração do funcionamento de uma rede de troca de arquivos baseada em topologia híbrida. Todos os nós da rede mantêm uma conexão constante com o servidor central, que armazena suas informações de conectividade, como endereço IP e largura de banda da conexão. Além disso, o servidor central mantém uma lista atualizada de todos os arquivos compartilhados por cada nó da rede. O processo de obtenção do arquivo desejado se inicia pela solicitação de busca feita por um nó ao servidor, que passa ao solicitante os endereços de diversos nós que disponibilizam o arquivo desejado, tendo como base o seu catálogo de arquivos compartilhados. A partir desse momento, a comunicação se dá diretamente entre o nó solicitante e o provedor do arquivo, para a transferência do arquivo desejado.

A execução de buscas de forma centralizada traz eficiência e permite buscas por diversos critérios mas, por apresentar um ponto único de falha, facilita o desligamento dos serviços, seja por questões técnicas ou, como no caso do Napster, jurídicas. Uma série de processos judiciais por quebra de direitos autorais culminaram no desligamento do serviço [18] e sua posterior aquisição pela iniciativa privada.

Como resultado dessa fragilidade, foi observada a emergência de serviços de compartilhamento de arquivos onde não apenas o armazenamento como também a execução das buscas é feita de forma distribuída. Isto é conseguido pela criação de catálogos distribuídos entre os nós ou pelo simples encaminhamento sucessivo da busca até o alcance de um nó com o conteúdo desejado. A essa topologia se denominou P2P pura, exatamente por não prever a existência de nenhum serviço centralizado. Exemplos claros de aplicações que adotam esta topologia são as redes Gnutella [15, 19] e KaZaA [20]. Vale ressaltar que, mesmo nessas redes, pode haver nós com funções especiais de catalogação e busca, também chamados de super-nós, como ocorre na rede KaZaA. Entretanto, ainda assim,

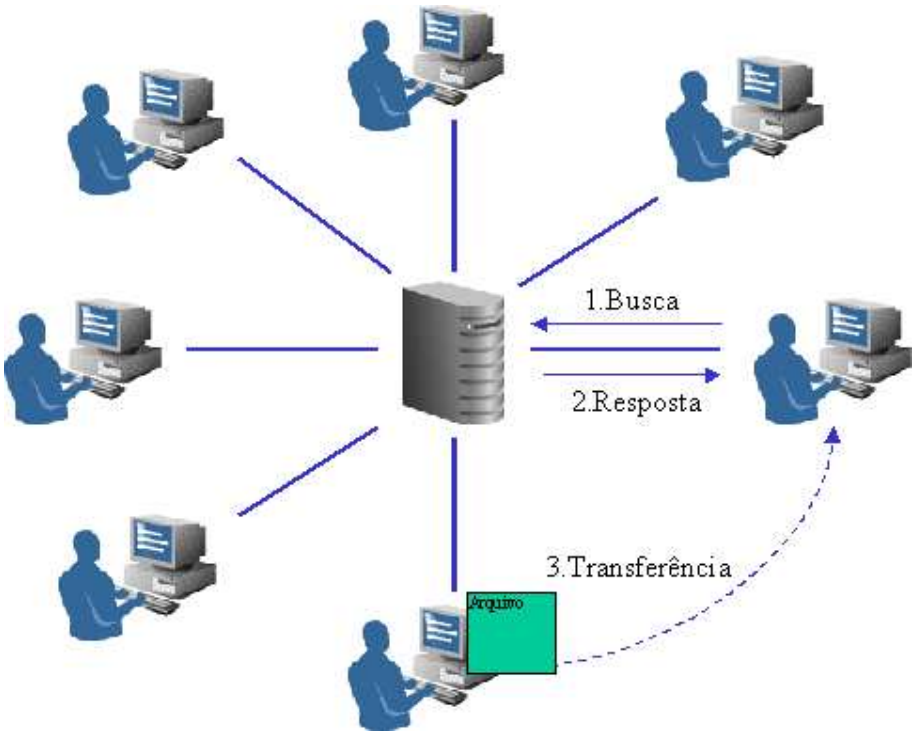

Figura 2.1: Rede de troca de arquivos, com topologia P2P híbrida.

estes nós compõem entre si uma rede que garante a inexistência de pontos únicos de falha. Graças a essas características, torna-se virtualmente impossível o desligamento de um serviço baseado em topologia P2P pura. Apesar de sucessivas tentativas de desligamento destes serviços por ações judiciais, tendo como fato motivador a quebra de direitos autorais, estas redes continuam em funcionamento. A Figura 2.2 representa uma rede P2P pura composta por 12 nós. Pode-se perceber que cada nó possui conexão apenas com seus vizinhos, sem necessitar de conhecimento sobre o restante da rede. A busca por um recurso é encaminhada sucessivamente até alcançar um nó que responda positivamente à requisição.

Atualmente, a maioria dos autores entende como aplicações P2P aquelas baseadas em topologia pura, validando a definição mais recente para P2P [21]: “Sistemas peer-to-peer são sistemas distribuídos consistindo de nós interconectados capazes de se auto-organizarem em topologias de rede com o objetivo de compartilhar recursos como conteúdo, ciclos de CPU, armazenamento e banda, capazes de adaptação a falhas e de acomodar populações transientes de nós enquanto mantêm níveis aceitáveis de conectividade e desempenho, sem ser necessária a intermediação ou suporte de um servidor ou autori-

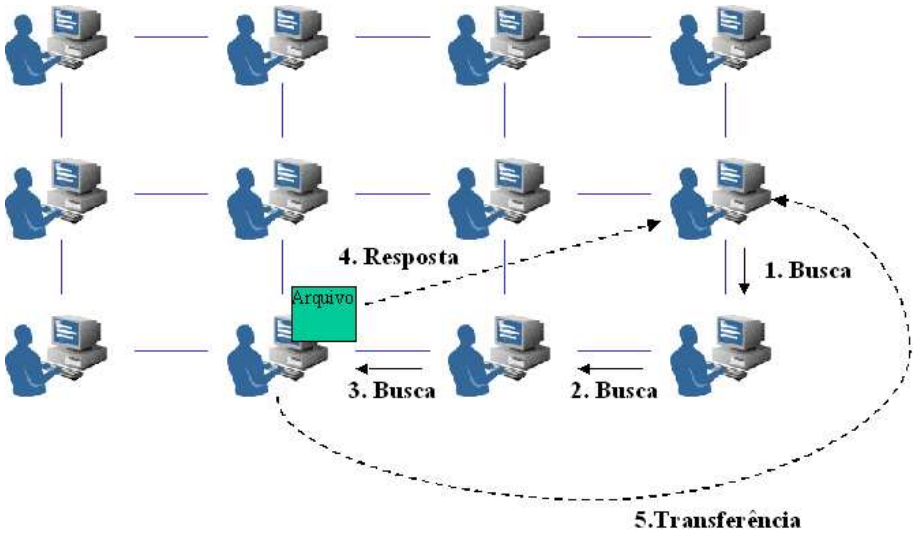

Figura 2.2: Rede de troca de arquivos, com topologia P2P pura.

dade global centralizada". Neste trabalho, é seguida esta linha de classificação, abordando apenas este grupo de arquiteturas.

2.2 Características Analisadas

Com a expansão na utilização de serviços baseados em redes P2P, o conjunto de requisitos desejáveis em tais redes tem sido elevado. Consideradas a necessidade de manter uma total descentralização e a possibilidade de falha dos nós, a implementação de diversas funcionalidades relativamente simples e já exploradas em outras topologias se torna um desafio no caso de redes P2P. A seguir, estão descritos alguns destes requisitos, que têm sido considerados nos trabalhos mais atuais em arquiteturas P2P.

Escalabilidade

A primeira característica fundamental para redes P2P é ter suas funcionalidades pouco afetadas pelo grande número de nós participantes. Uma premissa para tais redes é a participação de números elevados de nós, chegando a milhões em alguns casos [22]. De forma geral, a função mais afetada pelo aumento no número de nós é a busca por recursos na rede. Assim, o desenvolvimento de algoritmos mais eficientes para encaminhamento

das solicitações se tornou uma constante em diversos trabalhos.

Balanceamento de Carga

De forma geral, os nós participantes de uma rede P2P oferecem e utilizam recursos da rede, sendo essa constante troca o que viabiliza seus serviços. Entretanto, se um grupo muito grande de nós adota atividades exclusivamente de consumo, concentrando a carga de trabalho em um grupo reduzido de nós, é esperada uma queda no desempenho dos serviços. Para garantir que isso não ocorra, torna-se importante a implementação de mecanismos de balanceamento de carga na rede. Ainda nesse sentido, deve-se atentar que os nós possuem capacidades diferentes de processamento, conexão à rede e armazenamento. Assim, distribuir igualmente a carga entre eles não garante necessariamente um desempenho ótimo.

Disponibilidade

O nível de utilização de qualquer serviço está diretamente relacionado à confiança em sua disponibilidade. Em serviços centralizados, para garantia de alta disponibilidade, é necessária a utilização de componentes que possuam individualmente baixa probabilidade de falha. Já as redes P2P, ainda que sejam formadas por nós com altas chances de falha, não possuem ponto único de falha, podendo oferecer assim um nível de disponibilidade relativamente elevado. Para tanto, deve-se garantir que a recuperação à saída ou falha de um nó ocorra de forma rápida, sendo clara para o restante dos nós a divisão da responsabilidade em assumir as funções do nó indisponível.

Flexibilidade das Buscas

A localização de recursos em uma grande rede pode ser uma tarefa árdua, como observamos hoje na *World Wide Web*. A forma mais simples de fazê-lo em redes P2P é o conhecimento prévio de um identificador único do recurso desejado. No caso de uma rede de troca de arquivos musicais, esse identificador poderia ser o nome do arquivo. En-

tretanto, esse conhecimento prévio nem sempre existe, e torna-se necessária a realização de buscas por outras características conhecidas do recurso, podendo-se obter vários itens como resposta. Neste exemplo, poderia-se buscar por nome do intérprete, ou data da gravação. Muitas vezes, é interessante executar esse tipo de busca por intervalos de valor, expandindo ainda mais a facilidade em se localizar os recursos desejados. Dependendo da arquitetura utilizada para oferecer os serviços, essas buscas podem ser dificultadas ou facilitadas.

Segurança

Certamente, uma das questões mais controversas em redes P2P diz respeito à segurança. Os requisitos de segurança podem variar e se mostrar até mesmo antagônicos dependendo da aplicação. Em alguns casos há a preocupação com anonimato, de forma a garantir que não haja invasão da privacidade dos usuários. Esse é o exemplo da rede Freenet [7], em que se pretende permitir que os usuários compartilhem áreas de disco distribuídas sem acessos indevidos ou ligação entre autor e obra. Por outro lado, em redes de compartilhamento de arquivos, torna-se essencial a rastreabilidade, de forma a evitar a quebra de direitos autorais ou compartilhamento de outros tipos ilegais de conteúdo. Há ainda a necessidade de garantir a autenticidade dos recursos disponibilizados, de forma a evitar a inundação da rede por arquivos maliciosos ou alterados. Todos esses requisitos de segurança se apresentam como desafios ainda a serem vencidos por completo. Uma vez que não há uma entidade central de autenticação ou certificação, não é possível a adoção de uma estrutura de segurança baseada em certificados digitais (PKI). Há a necessidade de criação de mecanismos descentralizados que assegurem a redução do efeito de nós nocivos à rede. Nesse sentido, surgiram uma série de algoritmos, como o proposto em [23], baseados no conceito de reputação, de forma a criar modelos distribuídos de confiança. Entretanto, são algoritmos probabilísticos e não estão livres de manipulação, o que deixa esta questão ainda como um desafio a ser vencido.

2.3 Arquiteturas Não-Estruturadas

Denominam-se redes P2P não-estruturadas aquelas que se estruturam de maneira menos ordenada, não-determinística, não havendo associação entre o posicionamento de cada nó na rede e o conteúdo por ele disponibilizado. A maior parte dos serviços P2P de compartilhamento de arquivos oferecidos na Internet se encontram nesse grupo. Nesses serviços, cada nó trabalha de maneira independente na oferta dos recursos, sem um esforço coordenado por balanceamento de carga ou garantia de disponibilidade. Dessa forma, medições em redes não-estruturadas indicam um altíssimo nível de concentração de carga em um grupo muito pequeno de nós. Na rede Gnutella, por exemplo, foi identificado que 70% de todos os arquivos são providos por apenas 5% dos nós [24]. A representação gráfica desse tipo de arquitetura pode ser vista na Figura 2.2.

Também não há separação entre as camadas de controle e armazenamento de dados. Isto quer dizer que cada nó tem conhecimento exclusivo dos recursos por ele disponibilizados, a menos da utilização de algum mecanismo de *caching*. Como o posicionamento de cada nó na rede é independente do conteúdo por ele disponibilizado, não há nenhuma relação entre proximidade entre nós na rede e afinidade de interesses. Além disso, cada uma das implementações nessa arquitetura define parâmetros diferentes para as buscas, como nome de arquivo, autor e data de publicação, que estão relacionados à sua aplicação-fim.

Devido a essa organização, as buscas nas redes não-estruturadas são feitas por inundação. O nó que solicita uma busca consulta seus vizinhos quanto à disponibilidade de recursos que atendam a determinados parâmetros. Estes re-encaminham a solicitação aos seus vizinhos e assim sucessivamente. Essa busca pode ser interrompida tendo como base um TTL (*Time to Live*) ou por outros eventos, como a localização do recurso. Este algoritmo de busca impede a obtenção de respostas completas, uma vez que nem toda a rede pode ser consultada, e possui uma baixa escalabilidade pelo rápido crescimento no número de nós e mensagens envolvidas na busca. Entretanto, para usos amadores como busca de músicas e filmes, essa arquitetura tem se mostrado viável. Isso ocorre pois há uma rápida replicação dos arquivos mais solicitados, fazendo com que em poucos saltos seja possível encontrar o recurso desejado.

Buscando otimizar o desempenho dessas redes, em alguns casos são criadas hierarquias entre os nós, criando super-nós que servem como catálogo para um grupo de nós da rede e limitam a inundação a um sub-conjunto de nós. Vê-se na Figura 2.3 a representação de tal organização, presente na rede KaZaA, onde encontra-se super-nós, representados pela sigla SN, e nós ordinários, representados pela sigla ON. Também nessas redes não há suporte à implementação de garantias de nível de serviço para os recursos ou balanceamento de carga.

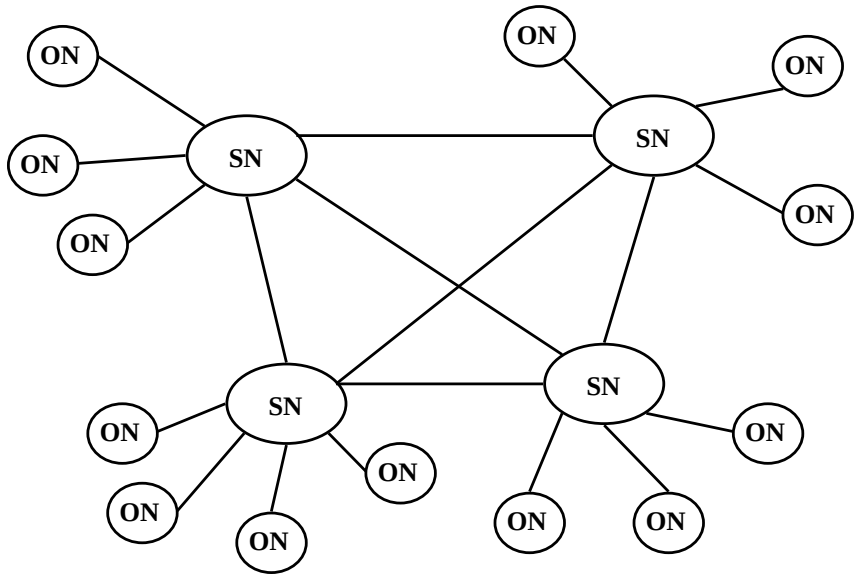

Figura 2.3: Representação da rede KaZaA. Arquitetura não-estruturada, com utilização de super-nós.

Nesse modelo, cada um dos nós ordinários mantém uma conexão TCP constante com um super-nó e transfere a esse nó os metadados de todos os seus arquivos compartilhados. Quando deseja realizar uma busca, um nó a envia exclusivamente ao SN a que está associado, que então verifica em seus registros quais os outros ONs oferecendo o arquivo solicitado. A busca pode ser encaminhada a outros SNs, para permitir uma maior abrangência. Cada nó mantém uma lista de outros SNs disponíveis na rede, de forma a poder restabelecer sua conexão em caso de falha do nó a que está conectado. Entretanto, esse tipo de arquitetura se torna inviável para fins que requeiram buscas completas na rede, como a implementação de bancos de dados distribuídos, e exige uso excessivo de recursos e replicação dos dados, para garantir que as buscas sejam adequadamente atendidas.

2.3.1 Redes de Sobreposição Semântica

Dada a grande quantidade de mensagens em circulação em redes de arquitetura não-estruturada, surgiram uma série de mecanismos para direcionamento mais assertivo das buscas na rede. Estes mecanismos se baseiam na criação de redes de sobreposição, em que os nós se conectam por afinidade de interesses e conteúdo disponibilizado. Assim, qualquer busca é inicialmente encaminhada a nós com maior chance de acerto. Na Figura 2.4 são apresentados exemplos de redes de sobreposição para um serviço de compartilhamento de arquivos musicais.

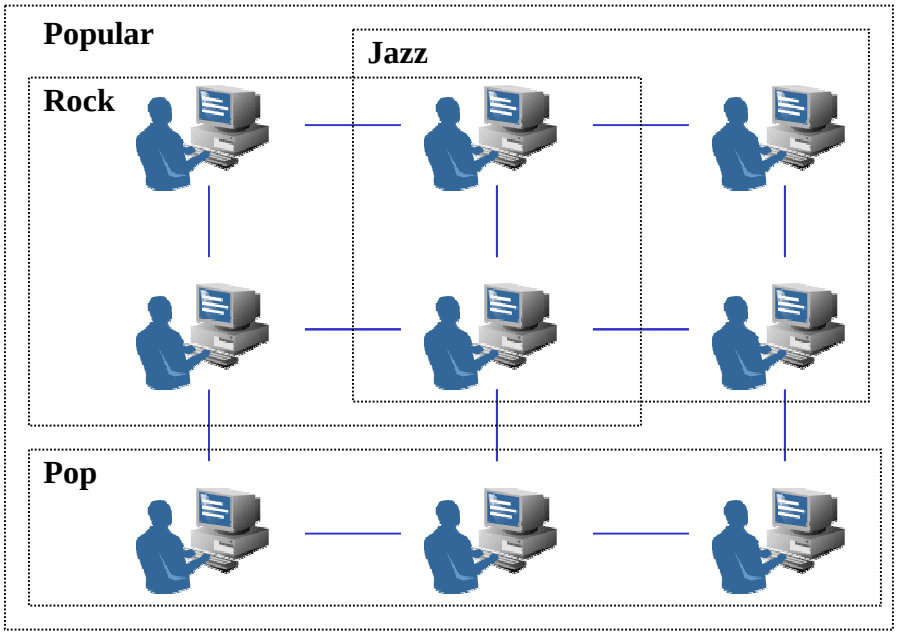

Figura 2.4: Redes de sobreposição semântica.

Um exemplo de tal mecanismo é proposto em [25]. A proposta apresentada se baseia na definição de um critério de classificação hierárquico e estático do conteúdo, onde cada item disponibilizado pode estar associado a uma ou mais folhas desta hierarquia. A cada uma das folhas da hierarquia, bem como a cada vértice, está associada uma rede de sobreposição. Fazem parte de uma rede de sobreposição os nós que declarem explicitamente possuir conteúdo classificado na sua categoria. Vemos na Figura 2.5 um exemplo de classificação possível.

Para que tal técnica apresente um real impacto positivo no desempenho da rede, é es-

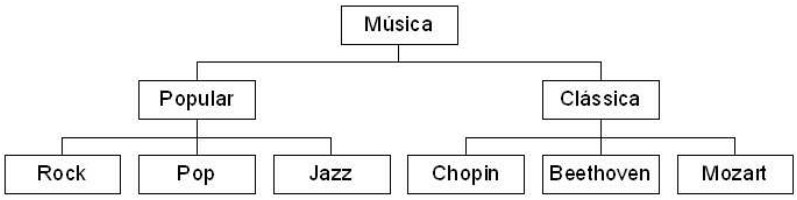

Figura 2.5: Classificação hierárquica de arquivos musicais.

sencial uma escolha adequada da classificação a ser utilizada. Caso contrário, a realização de uma única busca pode envolver um grande número de redes de sobreposição ou cada nó pode necessitar participar de muitas redes.

Outro exemplo de mecanismo com finalidade semelhante é a proposta de criação de *clusters* para a rede eDonkey, feita em [26]. Esta proposta possui um caráter muito mais dinâmico para a construção das redes de sobreposição. Ao invés de utilizar uma definição explícita de categorias de conteúdo, a proposta se baseia na captura de correlações semânticas entre nós através de heurísticas. Neste trabalho, são analisados diversos algoritmos que se baseiam no direcionamento inicial das buscas aos nós que se mostraram mais úteis no passado. As simulações, utilizando conteúdo similar ao disponível em redes reais, demonstram um incremento na taxa de acerto.

Embora as redes de sobreposição semântica permitam uma redução considerável no número médio de nós envolvidos nas buscas, elas não alteram as características fundamentais das redes não-estruturadas. Isto quer dizer que as buscas continuam baseadas em algoritmos de inundação, ainda que limitados a uma fração dos nós, sem permitir a realização eficiente de buscas completas na rede. Também continuam inexistentes mecanismos para balanceamento de carga entre os nós ou garantia de qualidade de serviço.

2.3.2 Agnus: Adaptações aos nós Gnutella

Uma outra proposta para resolução de algumas das dificuldades das redes P2P não estruturadas é feita em [27], o servidor altruísta para a rede Gnutella (Agnus - *Altruistic Gnutella Server*). Esta proposta se baseia na criação de um novo tipo especializado de nó para a rede Gnutella, com a função de melhorar o nível de serviço da comunidade que o cerca, ao permitir uma localização mais ampla de recursos na rede, impedir a alocação

total de banda de conexão por pedidos entrantes, melhorar o balanceamento de carga e prevenir a transferência de arquivos maliciosos.

Este nó mantém um número maior de conexões que os nós regulares da rede, 8 ao invés de 4, de forma a influenciá-la de forma mais forte. Para melhorar o balanceamento de carga, este nó especializado limita o encaminhamento de mensagens a nós que já estejam sobrecarregados. Para permitir uma taxa de acerto maior, o nó Agnus mantém uma lista de nós com maior probabilidade de responderem a buscas por determinados tipos de recurso. Também é criado neste nó um repositório mais extenso de arquivos em *cache*, tendo como base as buscas mais freqüentes na rede, reduzindo assim o número de saltos para estas buscas.

Ainda que os resultados de tal proposta tenham se mostrado interessantes, reduzindo a variação de carga nos nós e a chance de queda por alto volume de requisições, não há alterações na organização da rede. Desta forma, as características fundamentais das arquiteturas não-estruturadas, que impedem uma escalabilidade desejada e a realização de buscas completas na rede, são mantidas.

2.4 Arquiteturas Estruturadas Baseadas em Tabelas *Hash* Distribuídas

Consideradas as limitações de escalabilidade e balanceamento de carga existentes nas redes P2P não-estruturadas, tornou-se essencial o desenvolvimento de arquiteturas diferenciadas para a expansão da utilização destes serviços. Nesse sentido, as propostas de redes P2P estruturadas de maior impacto se baseiam na utilização de tabelas *hash* distribuídas (*Distrubuted Hash Table* - DHT).

Uma tabela *hash* é uma estrutura de dados que mapeia de forma uniformemente distribuída as chaves de identificação em valores de tamanho fixo. A chave pode ser qualquer seqüência de dados, como um nome de identificação ou um número inteiro, de qualquer comprimento. Este mapeamento é feito pela aplicação de uma função do tipo *hash* criptográfica, também chamada de função de espalhamento, à chave de entrada. Tais funções

são unidirecionais, sendo impossível determinar a entrada a partir do resultado, e não mantêm qualquer ordenamento na transformação [28]. Quando aplicada a um número suficientemente grande de chaves, pode-se assegurar ainda que os resultados estão uniformemente distribuídos pelo espaço de valores da saída. Dependendo do tamanho definido para os valores de saída, várias chaves de entrada podem resultar em uma mesma saída; tais eventos são chamados de colisão.

Utilizando esse conceito, pode-se criar uma tabela *hash* possuindo como chaves de entrada identificadores dos recursos disponibilizados na rede, como nomes de arquivos. Esta tabela pode estar dividida entre os vários nós da rede, ficando cada nó responsável pelos recursos que resultaram em saídas da função *hash* dentro de um determinado intervalo de valores. Como a função *hash* utilizada gera saídas uniformemente distribuídas, se os intervalos de valor de responsabilidade dos nós forem iguais, eles devem ter responsabilidade por igual número de recursos na rede, resultando em um balanceamento de carga natural. Este modelo de funcionamento é utilizado em duas importantes propostas: as redes Chord [29] e CAN [11].

2.4.1 A Rede Chord

A rede Chord utiliza o conceito de DHT com organização circular dos nós. A definição do posicionamento do nó na rede é feita durante o seu ingresso, pela aplicação de uma função *hash* de *m* bits ao seu endereço IP. Os nós formam uma rede circular, ordenada segundo os resultados dessa função. A Figura 2.6 ilustra um exemplo de rede Chord onde *m* é igual a 3, e fazem parte da rede, no presente momento, apenas 3 nós. Toda vez que um novo recurso é disponibilizado na rede, a mesma função *hash* é aplicada a sua chave de identificação única, gerando uma nova chave. O nó responsável por armazenar esta chave e o ponteiro para este recurso é aquele cujo posicionamento na rede é igual ou imediatamente posterior a essa nova chave. No exemplo da Figura 2.6 foi disponibilizado na rede um recurso com chave associada à posição 2, tendo ficado sob responsabilidade do nó com posicionamento 4.

Fica claro ser suficiente que cada nó da rede conheça apenas seu antecessor e suces-

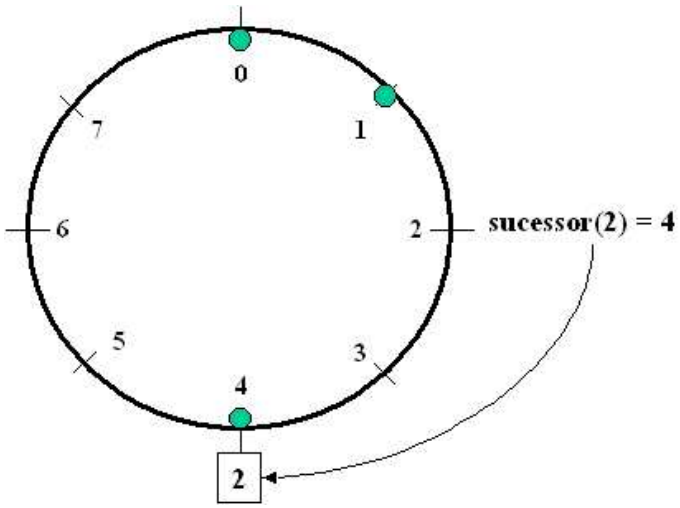

Figura 2.6: Rede Chord com *m* = 3.

sor. Essa conexão já é suficiente para realizar o registro de qualquer recurso na rede ou busca a partir da chave de identificação, bastando encaminhamentos sucessivos no sentido adequado. Entretanto, apenas com essa informação, as buscas teriam ordem de execução igual a N, onde N é o número de nós na rede. Assim, é provado em [29] ser possível atingir ordem $\log(N)$, altamente escalável, se cada nó armazenar uma quantidade de ponteiros para outros nós de ordem $\log(N)$.

Tabela de Ponteiros

Cada nó da rede mantém uma tabela com, no máximo, *m* registros. O i-ésimo elemento na tabela de cada nó *n* contém a identidade e endereço IP do primeiro nó *s* que suceda *n* em, ao menos, 2^{i-1} no círculo de identificação. Cada nó armazena informação de apenas um grupo restrito de outros nós, possuindo maior conhecimento dos nós próximos a ele no círculo de identificação. Assim, uma tabela de ponteiros de um nó não possui necessariamente a informação necessária para determinar o nó sucessor a uma chave arbitrária *k*.

Execução de Buscas

A rede Chord possui uma única operação para a localização de recursos. O nó solicitante deve conhecer previamente uma chave de identificação única do recurso desejado. Aplica a esta chave a função *hash*, obtendo como saída uma nova chave *k*, de *m* bits. O próximo passo é a localização do nó da rede cujo identificador suceda imediatamente a *k*. A primeira etapa é procurar em sua própria tabela de apontamento. Quando o nó solicitante não é capaz de determinar tal nó baseando-se unicamente em sua tabela, busca em sua tabela o nó *j* cujo identificador preceda *k* de forma mais próxima, solicitando a *j* a informação do nó com identificador mais próximo de *k* em sua tabela de ponteiros. Dessa forma, *n* toma conhecimento de nós com identificadores cada vez mais próximos de *k*. Esse processo é repetido até alcançar-se o nó imediatamente anterior ao desejado, que é aquele cujo primeiro sucessor na tabela de ponteiros possui identificador maior ou igual a *k*. O nó desejado é o sucessor direto deste nó.

Balanceamento de Carga

Havendo um número suficientemente grande de nós e recursos publicados na rede, o balanceamento de carga é atingido de maneira quase que imediata. Uma vez que o identificador de cada nó na rede é resultado da aplicação de uma função *hash* ao seu endereço IP, havendo um número suficientemente grande de nós da rede, eles deverão estar igualmente espaçados no círculo de identificação. Assim, a tendência é que os nós tenham regiões de responsabilidade de mesmo tamanho. Como também ocorre a aplicação de função *hash* nas chaves de identificação dos recursos, seu posicionamento na rede também será uniforme, podendo-se assegurar o balanceamento de carga no que se refere ao número de recursos por nó.

2.4.2 A Rede CAN

A proposta para uma rede escalável, com endereçamento por conteúdo (*Content Addressable Network* - CAN) feita em [11] prevê a organização dos nós em um espaço cartesiano de *d* dimensões. Em qualquer momento, todo o espaço de coordenadas está

dividido entre os nós da rede, de forma que cada nó possui uma região de responsabilidade. A Figura 2.7 mostra um espaço de coordenadas bi-dimensional dividido entre 5 nós. Para a disponibilização de um recurso na rede, são aplicadas *d* funções *hash* à sua chave de identificação, gerando como saída *d* ordenadas, representando um ponto no espaço d-dimensional. Assim, o nó responsável pelo armazenamento dessa chave é aquele responsável pela região contendo o ponto encontrado. Cada nó possui, dessa forma, conhecimento exclusivo de 2.*d* nós, seus vizinhos superior e inferior em cada uma das dimensões.

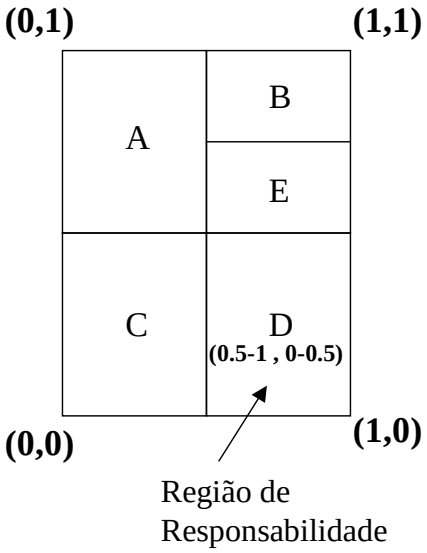

Figura 2.7: Uma rede CAN com *d* = 2.

Execução de Buscas

Assim como na rede Chord, a única operação de busca suportada pela arquitetura CAN é baseada na chave de identificação única do recurso desejado. Para realizar a busca por um recurso na rede, um nó deve aplicar *d* funções *hash* à chave de identificação do recurso buscado, descobrindo assim as *d* ordenadas que indicam o "posicionamento" do recurso na rede. Após essa definição, inicia um processo de encaminhamento sucessivo da busca para um de seus nós vizinhos, em direção àquele com região de responsabilidade mais próxima ao ponto desejado. Este processo é repetido até o alcance do nó que possui

o ponto associado à chave de identificação do recurso dentro de sua região de responsabilidade. Este algoritmo de busca alcança ordem de $d.\sqrt[d]{N}$, *d*é o número de dimensões e *N* o número de nós da rede.

Balanceamento de Carga

Ao ingressar na rede, um dado nó *n* gera, aleatoriamente, um conjunto de *d* ordenadas. Envia, então, uma mensagem de tipo específico ao nó responsável por aquela posição na rede. Este nó divide, então, sua própria região de responsabilidade em duas menores, passando o controle de uma delas ao nó entrante. Uma vez que a escolha das ordenadas seja aleatória, e com distribuição uniforme, pode-se esperar que, para grandes quantidades de nós, a divisão de regiões também seja igualitária. Uma vez que o posicionamento de cada recurso na rede segue a aplicação de funções *hash*, também com distribuição uniforme, pode-se esperar um balanceamento de carga em relação à quantidade de recursos sob responsabilidade de cada nó. Entretanto, já é sugerida em [11] uma forma de permitir um balanceamento de forma mais rápida, alterando a forma de entrada de nós na rede. No lugar de cada nó contatado se dividir imediatamente, ele compara o tamanho de sua região de responsabilidade com a de seus vizinhos, encaminhando o pedido de entrada na rede para o nó com a maior região.

2.5 Desafios e Adaptações a Redes Baseadas em DHT

Mesmo tendo atingido o objetivo de prover escalabilidade para as redes P2P, as propostas baseadas em DHT ainda apresentam sérias limitações para aplicações mais abrangentes.

No tocante à localização de recursos, estas redes suportam apenas buscas por chaves de identificação. Assim, o nó solicitante deve conhecer de antemão o identificador único do recurso solicitado, não podendo executar buscas por atributos diversos. Em uma rede para troca de arquivos musicais, por exemplo, o nó solicitante deve conhecer o nome do arquivo buscado, não podendo realizar buscas pelo nome do intérprete ou data de

lançamento. Outra grave limitação é a impossibilidade de buscar por intervalos de valor, ou por máximos e mínimos, de forma eficiente. Uma vez que nessas redes a proximidade entre nós não representa proximidade semântica de recursos, seria necessário executar uma busca independente para cada valor no intervalo desejado. Estes tipos de busca têm relevância para permitir a utilização de redes P2P no compartilhamento de recursos computacionais, onde são procurados recursos que atendam a requisitos mínimos, como velocidade de processamento, área disponível em disco e memória. Estas buscas também são necessárias para realizar pesquisas em informação catalogada, como em bibliotecas virtuais, em que é usual buscar documentos pelo período da publicação, associado a outros filtros, como autor. A criação de bases de dados distribuídas sobre redes P2P também tem como requisito a implementação destas formas de busca, uma vez que são intensamente utilizadas em sistemas transacionais, para geração de relatórios e consolidação de dados.

As análises da utilização de tais redes também indicam que o atendimento do objetivo de balanceamento de carga só ocorre em condições de quantidades muito grandes de nós e recursos na rede. Assim, é necessária a criação de nós virtuais, o que prejudica o desempenho da rede, conforme demonstrado em [30]. Além disso, considerando que os nós participantes de uma rede possuem capacidades de processamento, armazenamento e largura de banda distintas, a distribuição igualitária de responsabilidade por chaves não resulta necessariamente em desempenho conjunto ótimo. No que tange a garantias de qualidade de serviço, estas propostas não tratam da possibilidade de existência de recursos com requisitos distintos. Há, nessas redes, preocupação com a disponibilidade das chaves, sendo prevista a replicação das mesmas. Em [11] é proposta a criação de diversas redes distintas, chamadas de realidades, para compartilhamento dos mesmos recursos. Entretanto, não é abordada a questão da disponibilidade dos recursos em si, e a possibilidade de implementar mecanismos de garantia de qualidade de serviço diferenciada para tipos distintos de recursos compartilhados na rede.

Algumas importantes propostas foram apresentadas visando adaptar estas redes de forma a cobrir estes pontos, principalmente com foco na flexibilização das buscas.

2.5.1 Busca por Proximidade Semântica: pSearch

Conforme mencionado anteriormente, os sistemas baseados em DHT suportam apenas buscas simples baseadas em chaves. Não permitem, por exemplo, buscas semânticas aproximadas. Em [10], é pretendida uma forma de aplicar em redes DHT os avanços na área de recuperação eficiente de informação, utilizando algoritmos de pontuação e ordenamento. Estes algoritmos, como o modelo de espaço de vetores (*Vector Space Model* - VSM) e a indexação semântica latente (*Latent Semantic Indexing* - LSI) [31], representam documentos e buscas como vetores, e medem a afinidade entre eles como o cosseno do ângulo entre os vetores. É selecionado um conjunto de termos (palavras), cada um associado a uma componente do vetor de representação. Para cada documento, é criado o vetor que indica a freqüência de cada termo no mesmo. No caso do algoritmo LSI, este vetor se converte em uma matriz, resolvendo problemas de sinônimos. A realização de uma busca semântica é, então, a busca por um documento que possua um vetor de representação próximo ao desejado. Fica claro que este procedimento se aplica exclusivamente a documentos de texto. São também propostas várias formas para permitir tais buscas sobre uma rede CAN. Primeiramente, é sugerido o mapeamento dos termos mais freqüentes em cada documento em um ponto do espaço d-dimensional, através da aplicação de funções *hash*. Assim, os termos passam a servir como chaves de identificação do documento. Cada nó responsável pela chave armazena o valor que indica a incidência daquele termo em cada documento registrado. A busca por um conjunto de termos, com determinado valor de incidência, passa a ser um conjunto de buscas por chaves no espaço, cada uma retornando todos os documentos a ela associados. Entretanto, fica clara a ineficiência desse método, por exigir a execução de muitas buscas e muita informação potencialmente inútil como retorno. A outra forma proposta é a utilização de cada dimensão do espaço para representar um termo do vetor, ficando cada documento registrado unicamente no ponto associado ao seu vetor de representação. Isso gera a necessidade de uma grande quantidade de dimensões, o que prejudica o desempenho da rede. Em ambos os métodos, ainda que haja uma importante extensão para o uso de redes DHT, a proposta se limita a permitir buscas por documentos de texto, não sendo aplicável a outros tipos de recursos. Outra questão relevante é a necessidade do conhecimento, por todos os participantes da rede, dos termos usados como índices, que devem ser estáticos. Também há necessidade de conhecimento

global da freqüência média de ocorrência dos termos, de forma a permitir a normalização necessária à execução das buscas.

2.5.2 Busca por Intervalos

A limitação tida como mais séria nas redes baseadas em DHT é o fato de não permitirem buscas por intervalos de valor, e sim apenas por identificador único. Assim, há várias propostas visando adicionar esta capacidade a tais redes, mas mantendo suas características fundamentais. Estas propostas são classificadas em duas categorias: arquiteturas de sobreposição a redes DHT e arquiteturas DHT adaptadas.

Árvore de Prefixos

As arquiteturas de sobreposição a redes DHT são aquelas onde se utiliza uma infraestrutura DHT simples (como Chord ou CAN), sendo criado um mecanismo externo para prover buscas mais flexíveis. Um exemplo de tal arquitetura é a proposta de tabelas *hash* de prefixo *Prefix Hash Table* - PHT) feita em [32]. Nesta proposta, é criada uma estrutura de árvore binária sobre uma rede DHT. Cada vértice da árvore possui um rótulo de prefixo R, e seus nós-filhos direito e esquerdo possuem, respectivamente, os rótulos $R0$ e $R1$. Este prefixo está associado aos valores de um determinado atributo dos recursos sendo compartilhados na rede. Assim, cada item da rede é armazenado no nó que possua o prefixo mais longo coincidente com o valor de seu atributo. A determinação de qual nó deve assumir o papel de vértice do prefixo R é feita pela aplicação de função *hash* a este prefixo, e associação ao nó da rede DHT responsável por aquele item. A busca por itens que possuem um atributo dentro de um intervalo de valores se dá pela identificação do nó da árvore com o prefixo mais longo que ainda englobe todo o intervalo desejado. A identificação desse nó pode exigir mais de uma busca, começando-se por um prefixo mais restritivo que pode não existir e, progressivamente, diminuindo o tamanho do prefixo buscado. Em seguida, é executada uma busca transversal em todas as sub-árvores daquele nó. A construção da árvore se dá de forma progressiva, sendo iniciada com um vértice-raiz, com um prefixo vazio, que é inicialmente responsável por todos os recursos da rede.

Quando este nó atinge um valor de disparo *D*, cria vértices-filhos, que passam a dividir a responsabilidade pelos itens. Esse processo se dá continuamente na rede, conforme são adicionados novos itens. Fica clara, então, a necessidade de reorganização periódica da estrutura, de forma a manter a busca otimizada. Ainda que esta arquitetura consiga prover buscas por intervalos de valor, não é possível aplicar filtros a vários atributos para uma mesma busca, uma vez que os prefixos da árvore representam um único atributo. Além disso, não está clara a possibilidade de executar outras funções como ordenamento e localização de máximos e mínimos.

Buscas por Múltiplos Atributos

Na categoria de arquiteturas DHT adaptadas, encontra-se a proposta Mercury [33]. A arquitetura Mercury se assemelha à rede DHT Chord, no aspecto que os nós se organizam de forma circular. Entretanto, o intervalo de valores de responsabilidade de cada nó possui um significado, sendo um atributo dos itens disponíveis na rede. Isto quer dizer que uma busca por itens que possuam um atributo dentro de um intervalo de valores pode ser mapeada em um conjunto de nós vizinhos nessa rede circular. Para permitir o suporte a buscas por diversos atributos, devem ser construídas várias dessas redes circulares, cada uma associada a um atributo. A Figura 2.8 ilustra um exemplo de rede com 8 nós, onde os itens possuem dois atributos, ano de publicação e tamanho do arquivo, suportados para buscas por intervalo.

Deve-se ressaltar que cada item a ser disponibilizado deve ser registrado em as todas as redes circulares (anéis), de forma a permitir sua busca por qualquer um dos atributos. Isto que dizer que, quanto maior o número de atributos, maior a quantidade necessária de réplicas das informações na rede e maior a quantidade de mensagens necessárias para a adição de novos itens. Outra opção seria o registro de cada item em apenas um dos anéis. Entretanto, isso exigiria que toda busca solicitada fosse encaminhada a todos os anéis, para garantir uma resposta completa, o que degradaria em grande medida o desempenho da rede.

Além de manter ponteiros para os nós sucessor e antecessor em seu próprio anel, cada nó deve possuir também ponteiros para os outros anéis, de forma a poder solicitar buscas

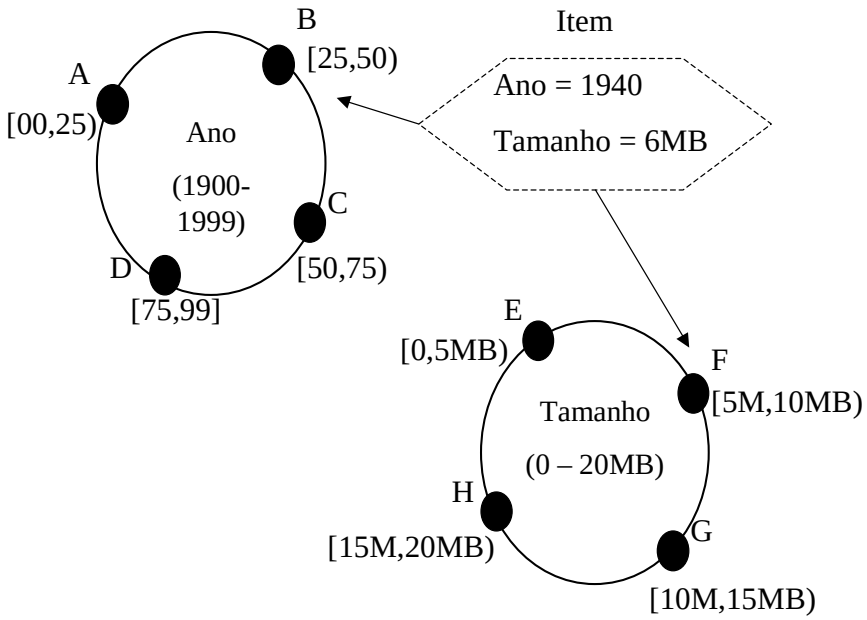

Figura 2.8: Rede Mercury para dois atributos de busca, com oito nós.

baseadas em outros parâmetros. A execução de uma busca é sempre executada em um único anel, mesmo que seja uma busca que aplique filtros em múltiplos atributos, uma vez que cada nó conhece todos os atributos dos itens de sua responsabilidade. Assim, antes de executar uma busca, o solicitante deve identificar qual o melhor anel a ser utilizado, que está associado ao atributo que implica maior restrição à busca. No exemplo da Figura 2.8, a realização de uma busca por arquivos publicados entre 1980 e 1981, com qualquer tamanho, implica o envolvimento de apenas um nó se executada pelo anel A mas, se executada no anel B, envolverá 4 nós.

Outra questão importante se refere ao balanceamento de carga. Uma vez que não há a aplicação de funções *hash* para determinação do posicionamento de cada item na rede, dividir o espaço de valores igualmente entre os nós da rede não implica necessariamente uma quantidade igual de itens de responsabilidade. Não há nenhuma garantia de que os itens da rede possuirão valores uniformemente distribuídos para um atributo. Assim, qualquer mecanismo que seja utilizado para reorganizar as responsabilidades dos nós e obter balanceamento de carga implicará regiões de responsabilidade com tamanho diferente. O

resultado dessa irregularidade é que torna-se impossível prever com exatidão o melhor anel para execução de uma busca, uma vez que uma maior restrição nominal da busca em um atributo não significa menor número de nós a serem pesquisados.

Um ponto em aberto nessa arquitetura é quanto à utilização da rede para compartilhamento de diversos tipos de recursos, que possuam atributos distintos. Uma vez definidos os anéis criados e os atributos com os quais estão associados, não há como utilizar a rede para compartilhamento de recursos que não possuam exatamente esses atributos. Isso quer dizer que as redes Mercury se tornam especialistas em uma aplicação e não genéricas, para compartilhamento de qualquer tipo de recurso.

2.6 Diversidade de Recursos

Conforme visto anteriormente, a maioria das redes P2P tem como objetivo o atendimento a uma aplicação específica, envolvendo o compartilhamento de um tipo específico de recurso. Nestes casos, qualquer recurso disponibilizado na rede pode ser descrito segundo um conjunto único de atributos, que possuam relevância para as buscas. Além disso, a interface para acessar qualquer recurso é a mesma. No exemplo de transferência de arquivos, se utiliza um mesmo protocolo de transferência para qualquer arquivo da rede.

Entretanto, pode-se desejar utilizar uma mesma rede P2P para compartilhamento de diversos tipos de recursos, como espaço em disco, capacidade de processamento e arquivos. Neste caso, torna-se necessário permitir o uso de atributos distintos para cada busca, dependendo do tipo de recurso buscado. Além disso, o protocolo de acesso ao recurso deve também ser distinto. Para tanto, fica clara a necessidade de utilização de metadados para a descrição dos diversos tipos recursos da rede, como proposto em [34], pela utilização de linguagem XML. Assim, é criada uma hierarquia de tipos de recursos, a partir de tipos básicos de atributos. Esta proposta é feita de forma independente em relação à arquitetura P2P utilizada, atendendo exclusivamente à questão da multiplicidade de tipos de recursos. A plataforma JXTA [35], que tem como objetivo se tornar um padrão em serviços distribuídos, também prevê o uso de arquivos XML para descrição dos recursos.

Já na questão da utilização de um protocolo único para acesso aos diversos tipos de recurso, tornou-se claro nos últimos anos que o protocolo SOAP, para acesso a *Web-Services*, se tornará o padrão para acesso a quaisquer recursos. Diversos trabalhos já apontam para a convergência entre redes P2P e Web-Services, como em [36], onde é proposta a implementação de serviços de descoberta de *Web-Services* utilizando redes P2P.

Capítulo 3

Arquitetura Proposta

Tendo em vista a análise das vantagens e limitações existentes nas arquiteturas P2P atuais, é apresentada neste capítulo uma nova arquitetura, visando compatibilizar as diversas características desejadas. Conforme observado no Capítulo 2, não foi ainda identificada uma única rede capaz de associar de forma definitiva as características de execução de buscas eficientes e flexíveis, balanceamento de carga, diversidade de recursos e garantias de qualidade de serviço. A arquitetura proposta neste capítulo tem como objetivo elevar a possibilidade de conciliação entre estes diversos requisitos, ao permitir a execução de buscas por atributos, intervalos de valor, máximos e mínimos, para diversos tipos de recurso. Nas seções a seguir, são apresentados os diversos aspectos da definição da arquitetura, incluindo a organização dos nós, algoritmos de execução de buscas, técnicas para assegurar a disponibilidade dos recursos, algoritmos de balanceamento de carga e descrição dos recursos.

A proposta tem como fundamento a divisão da rede em duas camadas: metadados e recursos. A primeira, é uma rede de sobreposição responsável pela execução das buscas, armazenando, para esse fim, descrições dos recursos disponíveis na rede. Já a camada de recursos é onde se encontram, de fato, os recursos disponibilizados. Dessa forma, é papel da camada de metadados apontar a localização dos recursos desejados na camada de recursos. De fato, as redes baseadas em DHT já prevêem intrinsecamente essa divisão, ao definirem regiões de responsabilidade de forma independente dos recursos disponibilizados por cada nó. Entretanto, não consideram a utilização de metadados, por não preverem

a execução de buscas por parâmetros semânticos. A criação de uma camada de metadados é fundamental para possibilitar buscas complexas, baseadas em semântica, conforme apresentado em [37].

Para que essas buscas sejam executadas de forma eficiente, é sugerida também, na arquitetura proposta, uma organização dos nós que reflete as informações dos metadados associados aos recursos. Isto quer dizer que a proximidade entre nós na rede representa uma proximidade entre os valores dos atributos de seus recursos, tornando mais eficientes as buscas por intervalos de valor.

A Figura 3.1 mostra a representação gráfica dos nós físicos, e da rede de sobreposição criada. Cada nó é representado por um número e pode fazer parte tanto da camada de metadados como de recursos. Cada recurso é representado por uma letra, possuindo seus metadados publicados na camada de metadados e estando disponível para consumo na camada de recurso. Na camada de metadados, cada nó é posicionado de forma a permitir um encaminhamento otimizado das buscas, de acordo com os metadados que armazena, possuindo ponteiros para os nós que contêm os recursos descritos. No exemplo apresentado, embora sete nós façam parte da camada de metadados, apenas três (1, 5 e 6) disponibilizam recursos para a rede.

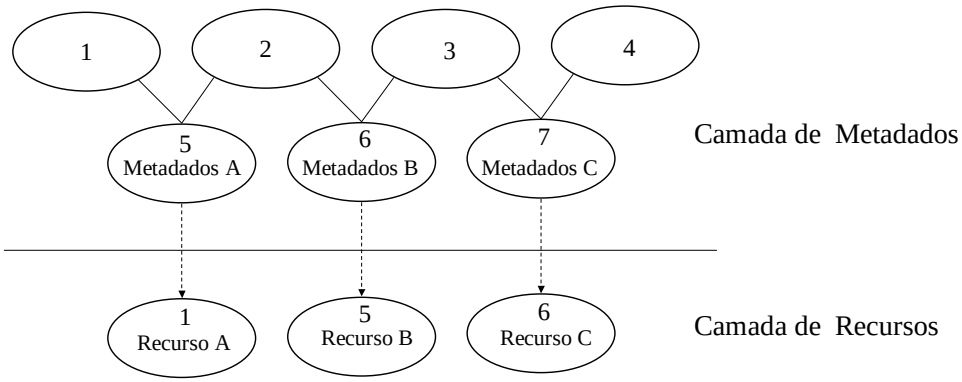

Figura 3.1: Representação das camadas de metadados e recursos na rede.

3.1 Organização dos Nós

A questão fundamental no projeto de uma arquitetura P2P é a definição das regras para criação de conexões entre os nós. Esta organização impacta diretamente na eficiência das buscas, medida pelo número de nós participantes em cada operação, definindo assim a escalabilidade da rede. Ao observar as redes não-estruturadas, percebe-se que as ligações entre nós são aleatórias, não estando associada aos recursos disponibilizados por cada participante. Por isso, não há como definir regras de encaminhamento baseadas nos parâmetros da busca, sendo implementados apenas mecanismos de inundação. Por outro lado, em redes estruturadas, baseadas em DHT, o posicionamento dos nós indica sua região de responsabilidade, sendo possível o encaminhamento baseado no ponto desejado pela busca. Entretanto, conforme demonstrado no Capítulo 2, essa organização impõe fortes restrições ao tipo de busca suportada, não sendo possível realizar buscas por intervalos de valor.

A organização proposta é baseada na arquitetura CAN [11], que se estrutura em relações de vizinhança entre os nós, em d dimensões. Nesta estrutura, cada nó possui a responsabilidade por armazenar os metadados dos recursos mapeados dentro de sua região de cobertura. Além disso, é suficiente que conheça exclusivamente seus vizinhos e suas respectivas regiões de responsabilidade, devendo manter ativas as conexões e armazenar informação sobre $2.d$ nós (vizinhos inferior e superior em cada direção). O ponto de diferenciação entre a arquitetura proposta e a rede CAN é que, na arquitetura proposta, não são utilizadas funções *hash* para definir a posição do espaço d-dimensional associada a cada recurso na rede. É exatamente esse mapeamento que quebra, na arquitetura CAN, a relação entre proximidade de nós na rede e proximidade semântica. Ao invés disso, cada dimensão do espaço está associada a um parâmetro de descrição dos recursos, com relevância para buscas. Esses parâmetros são denominados índices. Dessa forma, qualquer busca por intervalo de valores dos índices pode ser mapeada em um hipercubo no espaço d-dimensional, o que permite a execução de buscas por intervalos de valor de forma muito mais eficiente.

Na Figura 3.2, a organização dos nós é exemplificada para uma rede organizada com d=2, formando um espaço bi-dimensional, voltada ao compartilhamento de arquivos mu-

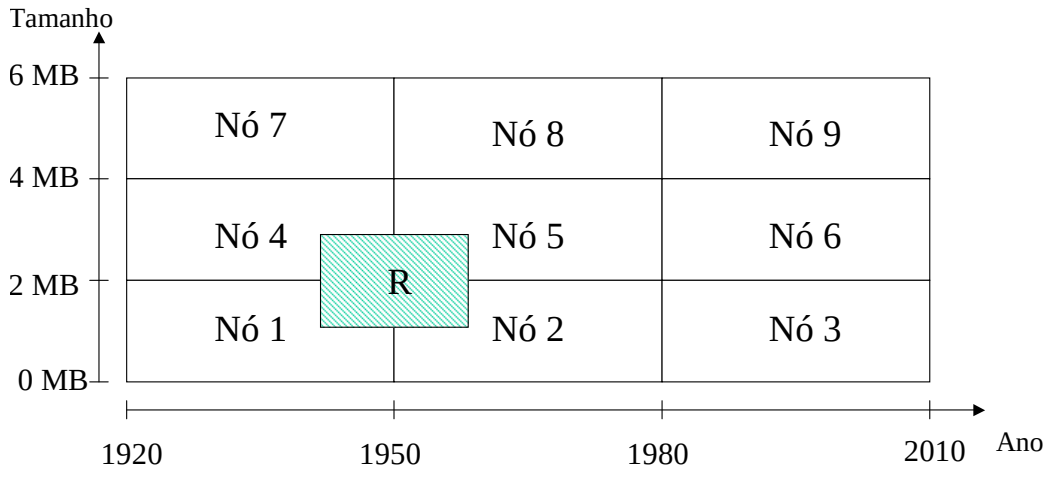

Figura 3.2: Organização dos nós e regiões de responsabilidade.

sicais. Compõem a rede 9 nós, que dividem entre si a responsabilidade por todo o espaço. A dimensão horizontal está associada ao ano de gravação da música, enquanto a dimensão vertical está associada ao tamanho do arquivo, sendo esses os dois índices disponíveis para busca eficiente. A região R representa a aplicação do filtro “todas as trilhas musicais gravadas entre 1935 e 1965, e com tamanho entre 1MB e 3 MB” ao universo. Para cada recurso disponibilizado na rede, o nó responsável por armazenar seus metadados é aquele em cuja região de responsabilidade recai o cruzamento dos valores dos seus índices.

3.2 Algoritmos de Funcionamento

O funcionamento da rede depende de uma série de algoritmos, executados de forma totalmente descentralizada, de forma a assegurar as características fundamentais de redes P2P. Nesta seção, são descritos os algoritmos para busca por recursos, publicação de recursos e balanceamento de carga entre os nós.

3.2.1 Busca por recursos

Um dos principais objetivos da rede proposta é permitir a execução de buscas mais flexíveis, baseadas em intervalos de valor, de forma a permitir uma aplicação mais abran-

gente da rede e sem prejuízo da escalabilidade apresentada na rede CAN. Nesse sentido, as buscas previstas para a rede são de três tipos: por item único, por intervalos de valor e por máximos ou mínimos.

Busca por Item Único

Neste tipo de busca o nó solicitante já conhece um identificador único do recurso desejado, ou um conjunto específico de atributos a ele associados, com valores únicos. Neste caso, a busca é mapeada em um ponto único do espaço d-dimensional. O encaminhamento da solicitação é relativamente simples. O nó solicitante verifica se o ponto solicitado está em sua região de responsabilidade, em caso negativo, identifica qual de seus vizinhos possui região de responsabilidade geometricamente mais próxima do ponto associado ao recurso. A solicitação é, então, encaminhada a esse nó. Esta operação é repetida por esse vizinho e assim sucessivamente, até o alcance do nó em cuja região se encontra o recurso desejado. Conforme será apresentado no Capítulo 4, o desempenho dessa busca na rede organizada por índices é diferente daquele obtido na CAN, uma vez que não se pode garantir uma relação entre menor distância geométrica e menor quantidade de saltos.

```
inicia_busca_recurso(parâmetros_do_recurso, nó_solicitante)
    ponto = mapeia_localização(parâmetros_do_recurso)
    busca_ponto(ponto, nó_solicitante)
fim

busca_ponto(ponto, nó_solicitante)
    Se (dentro_minha_região(ponto)) então
        responde(nó_solicitante)
        fim
    vizinho = verifica_vizinho_mais_próximo(ponto)
    vizinho.busca_recurso(parâmetros_do_recurso, nó_solicitante)
fim
```

Há duas opções para o envio da resposta ao nó solicitante. A primeira é que a resposta seja passada nó a nó, no sentido contrário ao da execução da busca, até o alcance do nó solicitante. Essa opção evita a necessidade de se transmitir, junto aos parâmetros de busca, as informações sobre o solicitante. Embora diminua o volume de dados por interação, essa forma de resposta exige a mesma quantidade de interações para envio da busca e da resposta, o que gera um volume excessivo de tráfego. A opção mais eficiente é que sejam repassadas, durante a execução da busca, em conjunto as informações sobre o nó solicitante, para que o respondedor possa contatá-lo diretamente.

Busca por Intervalos de Valor

A execução de buscas por intervalos de valor ocorre quando o solicitante não possui de antemão um identificador único do recurso desejado, podendo nem mesmo saber se existe tal recurso. Neste caso, define limites para os parâmetros de qualificação do recurso e realiza buscas por essas informações. Esse tipo de busca é fundamental para permitir a utilização de redes P2P em diversas aplicações emergentes, como a criação de bibliotecas virtuais, armazenamento e busca de informações geo-referenciadas. O compartilhamento eficaz de recursos computacionais, como capacidade de processamento, memória ou armazenamento, também depende da execução deste tipo de busca, uma vez que as buscas por estes tipos de recursos se baseiam em requisitos mínimos desejados, podendo ser atendidas por vários dos recursos da rede, e não na localização de um recurso específico.

Na rede proposta, a execução dessas buscas é relativamente simples, ao contrário das adaptações necessárias às redes DHT, e se divide em duas etapas: alcance e varredura. Durante a etapa de alcance, é calculado o ponto da região desejada mais próximo geometricamente do nó solicitante e a busca é encaminhada sucessivamente, pelo critério de menor distância geométrica, até alcançar o nó que contenha esse ponto. Caso o nó solicitante já faça parte da região desejada, essa etapa não é executada. Uma vez alcançado, esse nó inicia um processo de varredura por inundação limitada dentro da região desejada. Essa inundação fica restrita aos nós envolvidos na busca, uma vez que, baseados em sua região de responsabilidade, os nós de fronteira não encaminham a busca para fora desse perímetro. Apresenta-se a seguir o algoritmo executado para realizar a varredura dentro

da região por intervalos de valor.

```
inicia_varredura(R, outros_filtros, identificador_busca)
    Se já_realizou_varredura(identificador_busca) então fim
    recursos = consulta_recursos(R, outros_filtros)
    informa_recursos(recursos, identificador_busca)
    registra_execução(identificador_busca)
    vizinho_na_região[1 a i] = verifica_vizinhos_na_região(R)
    t = 1
    enquanto t≤ i
        vizinho_na_região[t].inicia_varredura
        t ++
    fim loop
fim
```

Como apresentado, os parâmetros de entrada são a região buscada (limites para os índices), outros filtros por atributos que não estejam associados a índices e um identificador único da busca. Este identificador é gerado pelo solicitante e contém seu endereço, associado a uma numeração aleatória, servindo como informação para envio das respostas e como garantia de que um mesmo nó não executará repetidamente as operações para uma mesma busca. Durante a fase de varredura, o nó verifica quais os recursos que possui seguindo os filtros aplicados, informando-os ao solicitante. Posteriormente, verifica quais dos seus vizinhos estão na região buscada e lhes encaminha a solicitação. Esse algoritmo é executado por todos os nós da região buscada. Outra característica interessante deste algoritmo é que o solicitante pode verificar se a busca foi executada de forma completa, bastando para isso que todos os nós incluam em sua resposta a sua própria região de responsabilidade.

Em algumas aplicações, como a implementação de bancos de dados distribuídos, a verificação da completude da busca é requisito fundamental para um funcionamento correto da aplicação [38]. Buscas por intervalos não-contínuos podem ser convertidas em diversas buscas contínuas.

Busca por Máximos ou Mínimos

Outro tipo de busca que provê grande flexibilidade para o compartilhamento de recursos é a localização de recursos com valores máximos ou mínimos para determinado parâmetro. Um exemplo de necessidade é no compartilhamento de espaço de armazenamento. A busca por tal recurso é descrita como “espaço para gravação de, no mínimo, 10MB”. Buscas similares a essa ocorrem por outros tipos de recursos computacionais, como capacidade de processamento. Sua correta execução permite prover, ao solicitante, recursos com desempenho alinhado à sua demanda, evitando o desperdício de recursos em operações de menor criticidade. Buscas por máximos e mínimos também são comuns em bases relacionais, utilizadas em aplicações transacionais, como a identificação da “maior compra realizada no mês de outubro”, em um sistema de varejo. Este tipo de busca também não é, atualmente, suportado por redes com arquitetura baseada em DHT.

Na arquitetura proposta, esta modalidade de busca é executada de forma relativamente simples. Uma opção é executar a busca sem considerar a condição de máximo ou mínimo, obtendo assim todos os recursos dentro das outras condições de filtragem e ficando a cargo do solicitante o ordenamento. Entretanto, esta opção leva ao envolvimento de muitos nós na busca e, possivelmente, ao retorno de um número muito grande de recursos de forma desnecessária. Na rede proposta, este tipo de busca pode ser realizada de forma mais eficiente, dividindo-a em duas etapas: busca por região limite e propagação na direção de ordenamento. Na primeira etapa, é realizada uma busca aplicando todos os filtros aplicáveis e um filtro com a condição de limite para o parâmetro de ordenamento (valor máximo ou mínimo para o parâmetro). Cada um dos nós alcançados responderá, informando os recursos que se enquadram na busca e o nó solicitante faz a classificação exclusivamente desse conjunto. Caso algum dos nós alcançados não possua recursos que obedeçam às condições, ele encaminha a busca para seu vizinho superior (busca por mínimo) ou inferior (busca por máximo) na direção de ordenamento. Esse encaminhamento é repetido até que o nó alcançado possua um recurso aplicável e pare a busca.

3.2.2 Publicação de Recursos

Para a publicação de um novo recurso na rede, devem ser seguidos os seguintes passos:

1. O nó que está incluindo o recurso na rede deve montar seu arquivo de descrição de metadados contendo, obrigatoriamente, os valores de todos os atributos associados a índices da rede e a localização do recurso. Essa atividade pode envolver interação humana ou utilizar algoritmos que analisam o recurso de forma automática para extração das informações necessárias;

2. A partir da descrição criada, o nó deve solicitar à rede uma busca pelo ponto associado àqueles valores de índice. O nó da rede responsável pela região onde o ponto se encontra responderá à busca;

3. O nó transfere o arquivo de descrição dos metadados para o nó de resposta.

3.2.3 Balanceamento de Carga

Em ambientes descentralizados de uso profissional, é fundamental que a carga de trabalho seja distribuída adequadamente entre os nós, de forma a evitar gargalos no serviço enquanto há capacidade disponível em outros nós da rede. Arquiteturas P2P não-estruturadas não prevêem balanceamento de carga coordenado. Nessas arquiteturas, o mecanismo que mais se aproxima desse objetivo é, em redes de compartilhamento de arquivos, a utilização de mecanismos de *caching* e replicação dos recursos para eliminação de pontos críticos da rede (*hot spots*).

No caso da rede proposta, é importante fazer inicialmente uma distinção entre o balanceamento de carga nas camadas de metadados e de recursos. O atendimento de uma distribuição adequada da carga na camada de metadados se dá quando os diversos nós da rede realizam o mesmo "esforço" para execução das buscas. Para esta camada, buscam-se algoritmos distribuídos que atinjam esse objetivo, de forma independente dos tipos de recursos sendo compartilhados. Já na camada de recursos, não há como definir mecanismos *a priori*, uma vez que muitos tipos de recurso, como capacidade de processamento, não

podem ser transferidos de uma máquina para outra. Assim, esse assunto será detalhado na Seção 3.4.

As propostas iniciais de redes baseadas em DHT, como CAN e Chord, entendem que a distribuição uniforme da responsabilidade por recursos (metadados) entre os nós implica um balanceamento de carga perfeito para a execução de buscas. Além disso, uma vez que nessas arquiteturas os recursos são mapeados em pontos geométricos de forma aleatória e seguindo uma distribuição uniforme (funções *hash*), se os nós tiverem regiões de responsabilidade de mesmo tamanho, terão responsabilidade pelo mesmo número de recursos. Assim, o balanceamento de carga nessas redes é baseado na execução de algoritmos que corrigem constantemente as regiões de responsabilidade dos nós visando igualá-las.

Entretanto, essas técnicas apresentam graves problemas. Primeiramente, o pressuposto de distribuição uniforme só se confirma, na prática, para uma grande quantidade de recursos e nós na rede. Assim, há a necessidade de criação de nós virtuais, conforme proposto em [11]. Entretanto, esta solução resulta num aumento artificial do número de nós da rede, causando graves impactos em seu desempenho para buscas, conforme demonstrado em [30]. Outra questão é que uma distribuição uniforme da carga não implica em desempenho ótimo ou nível idêntico de esforço entre os nós. Uma vez que os participantes da rede possuem diferentes capacidades de processamento, armazenamento e banda de conexão, a distribuição uniforme da carga pode gerar gargalos em nós de menor capacidade. A proposta de servidores virtuais, feita em [39], procura adequar o tamanho da região de responsabilidade à capacidade de cada nó. Isto é feito pela associação de cada nó físico a um número de nós virtuais proporcional a sua capacidade de processamento. Entretanto, o problema do aumento do número de nós da rede persiste, prejudicando o desempenho da rede. Por fim, as análises de tráfego em diversos serviços P2P indicam uma grande concentração das buscas a um número reduzido de recursos, o que gera forte desequilíbrio na carga dos nós da rede, que não é resolvido por uma distribuição uniforme de responsabilidade por recursos.

Consideradas essas questões, torna-se necessária a utilização de algoritmos dinâmicos de balanceamento de carga, que adaptem periodicamente a divisão das regiões de responsabilidade. Tais algoritmos devem possuir como parâmetro o nível de capacidade livre

em cada nó, e não o tamanho das regiões de responsabilidade. Este tipo de algoritmo foi inicialmente estudado e desenvolvido para resolução de gargalos em sistemas de armazenamento, conforme descrito por Ganesan *et al.* [40]. Assim como no caso de redes P2P, além da preocupação com o espaço de disco disponível em cada unidade, em tais ambientes há também a criticidade da distribuição do volume de acesso, de forma a evitar gargalos de leitura. Assim, a idéia fundamental é a criação de um parâmetro de carga *L*, que agrega à avaliação de uso e disponibilidade todas as capacidades, como processamento, banda de conexão e armazenamento, de cada nó físico. Esse parâmetro é um valor percentual, normalizado de acordo com as capacidades individuais. Cada nó calcula periodicamente o seu parâmetro de carga *L* e compara com um de seus vizinhos, escolhido aleatoriamente. Caso a diferença ultrapasse um valor de disparo, haverá uma redefinição de fronteiras na direção e apropriada, que resulte numa previsão de carga *L* igual para os nós. Quanto menor for o valor de disparo, maior é o nível de equilíbrio entre os nós. Entretanto, maior será o gasto para a transferência periódica de metadados.

Cabe ressaltar que a aplicação desta técnica de balanceamento de carga causa um impacto direto na organização da rede. Uma vez que o tamanho das regiões de responsabilidade de cada nó passa a ser função da carga, regiões do espaço com maior concentração de recursos estarão divididas entre mais nós. Desta forma, haverá regiões de maior "densidade" que outras. Assim, não se pode mais garantir, como ocorre em divisões uniformes, que o caminho de menor distância geométrica entre dois pontos corresponde ao caminho de menor número de saltos. O resultado disso é uma deterioração na eficiência das buscas por elementos únicos na rede, conforme apresentado no Capítulo 4.

3.2.4 Disponibilidade

Assim como na questão de balanceamento de carga, a garantia de disponibilidade pode ser dividida entre a camada de metadados e de recursos. Garantir a disponibilidade na camada de metadados significa garantir que as buscas serão executadas sem falhas, independentemente das possíveis saídas ou falhas dos nós da rede. Já a forma de garantir disponibilidade dos recursos depende do tipo de recurso em questão. Por exemplo, para garantir a disponibilidade de um arquivo, deve-se implementar replicação em dois ou mais

nós da rede. Já no caso de capacidade de armazenamento, esta técnica não é possível. O objetivo, nesta seção, é tratar da disponibilidade na camada de metadados, sendo a camada de recursos discutida na Seção 3.4.

Diversas pesquisas em redes P2P em funcionamento buscaram identificar os padrões de disponibilidade dos nós. Dada a natureza da maioria dos nós, são esperadas saídas e entradas voluntárias e repetidas, além das dificuldades causadas por falhas em seus componentes. Saroiu *et al.* [41] fez uma análise dos perfis de presença e largura de banda de nós participantes das redes Napster e Gnutella, resultando numa classificação dos nós. Um dos resultados relevantes deste estudo é que, na rede Gnutella, apenas 20% dos nós permaneciam conectados por mais de 45% do tempo. Mesmo corrigindo os erros de leitura devido a mudanças de endereço IP, o estudo feito por Bhagwan *et al.* [42] aponta que 50% dos nós possuem disponibilidade menor que 30%.

No tocante a saídas voluntárias dos nós, uma simples transferência de sua região de responsabilidade para algum de seus vizinhos, incluindo a transferência de metadados, já garantiria a manutenção da execução adequada das buscas. Entretanto, para garantir o funcionamento da rede mesmo em situações de saída não planejada, é necessário manter constantemente réplicas dos metadados na rede. Na rede CAN, propõe-se a utilização de múltiplas “realidades”, que nada mais são que estruturas d-dimensionais independentes. Cada nó físico está presente em ambas as redes, e cada recurso é sempre publicado duplamente. Esta forma de organização permite, ainda, uma otimização das buscas. Ainda que tal organização proveja disponibilidade, ela não teria impacto positivo na eficiência de execução de buscas por intervalos na rede proposta. Assim, é sugerida a replicação dos metadados sempre de um nó para o seu vizinho em uma direção previamente definida. Desta forma, além de não ser necessária a manutenção de mais conexões, há ainda a otimização na execução de buscas por intervalo de valor (ver Capítulo 4). A técnica de replicação utilizada é apresentada na Figura 3.3. Em caso de falha de um nó, o vizinho que contém seus metadados replicados assume a sua região de responsabilidade.

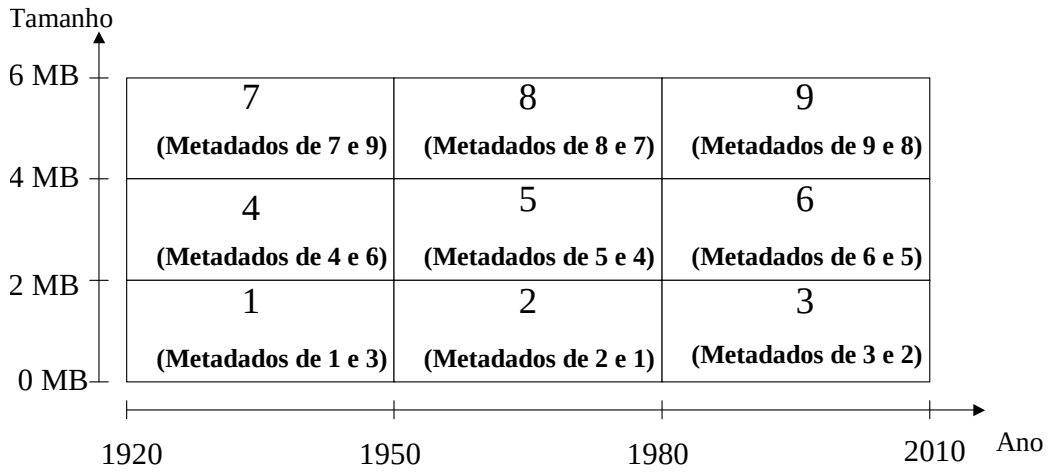

Figura 3.3: Rede P2P organizada por índices com replicação na direção 1 (ano), sentido crescente.

3.3 Flexibilidade e Metadados

Conforme mencionado anteriormente, um dos objetivos da arquitetura proposta é permitir a construção de redes com suporte simultâneo ao compartilhamento de diversos tipos de recurso. Uma das características da maioria dos serviços P2P em funcionamento é a especificidade a um ou a um grupo restrito de tipos de recurso para compartilhamento. O serviço Napster, por exemplo, permitia apenas o compartilhamento de arquivos do tipo MP3. Os sistemas P2P que o sucederam já previam o compartilhamento de qualquer tipo de arquivo. Há, agora, o interesse em que uma única rede possibilite o compartilhamento de tipos de recurso diversos, conforme a evolução da demanda.

A proposta de organização dos nós por índices feita na Seção 3.1 atende às necessidades de um único tipo de recurso, uma vez que cada direção do espaço está associada a um parâmetro com relevância para busca. Uma vez que se deseja compartilhar recursos classificados por parâmetros distintos, surgem dificuldades. Por exemplo, uma rede utilizada para compartilhamento de arquivos teria como um dos índices definidos o tamanho do arquivo. Haveria dificuldade em utilizar a mesma rede para o compartilhamento de capacidade de processamento, uma vez que esse atributo não faz sentido para buscas por esse tipo de recurso. Para tornar a rede flexível nesse aspecto, é necessária uma implementação

diferenciada, em que cada direção não esteja associada a um tipo restrito de dados, como tamanho de arquivo, espaço em disco ou data de publicação. Ao contrário, deve ser de um tipo genérico, um intervalo de números inteiros positivos, podendo representar valores de qualquer tipo de atributo. Assim, antes da execução de qualquer busca ou publicação de recursos na rede, deve ser executado um algoritmo de mapeamento, que converte cada um dos valores dos parâmetros de tipo restrito em valores do tipo genérico. Esse mapeamento deve conservar o ordenamento original, garantindo a viabilidade das buscas por intervalos. O resultado de tal alteração é que uma busca antes descrita como "todos os itens publicados entre 1940 e 1960, e com tamanho entre 1MB e 3MB" é transformada, antes de sua execução, em "todos os arquivos em que 60>D1>40 e 40>D2>30", onde 1940 é mapeado no número 40 e 1960 no número 60 da dimensão D1 (ano), e 1MB é mapeado no número 30 e 3MB no número 40 da dimensão D2 (tamanho). Como pode-se perceber, há a necessidade de informar, agora, qual o tipo de recurso sendo procurado. Embora essa informação não precise ser analisada em cada nó para o encaminhamento da busca na rede, ela é essencial para restringir a resposta aos recursos desejados. É apresentado na Figura 3.4 um exemplo de mapeamento de valores de tipos diferentes em uma mesma dimensão.

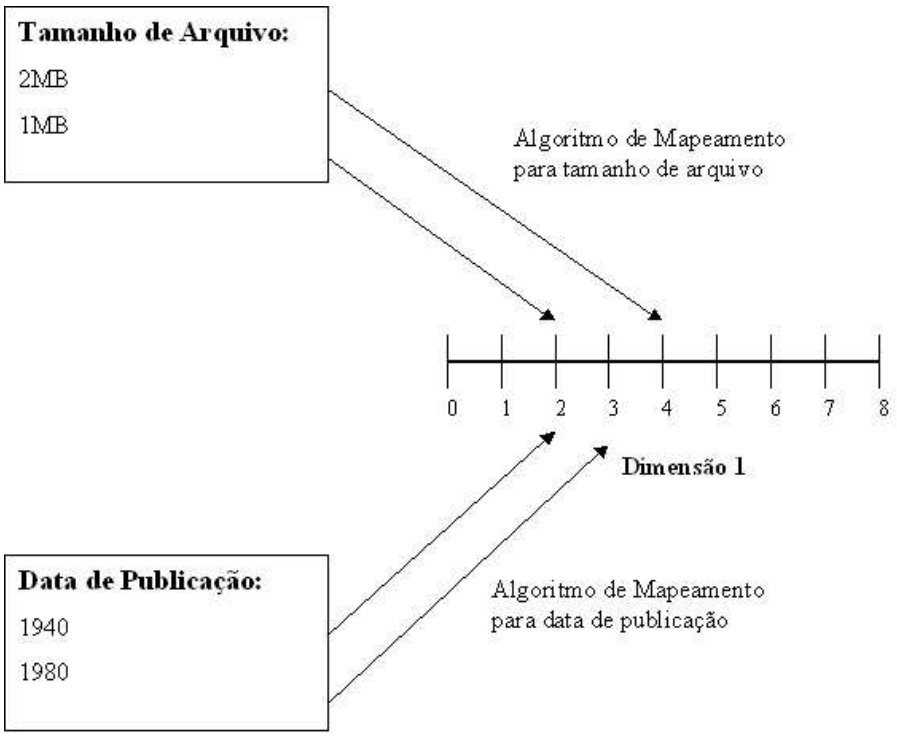

Figura 3.4: Mapeamento de vários atributos em uma mesma dimensão.

O algoritmo de mapeamento utilizado é, evidentemente, particular a cada tipo de parâmetro. Em alguns casos, como em parâmetros numéricos, a aplicação de um simples fator de multiplicação e o arredondamento do resultado são suficientes. A conversão de datas também é relativamente simples e bem conhecida, sendo implementada na biblioteca padrão da maioria das linguagens de programação, como a função *getTime* na linguagem Java [43]. Essa conversão é geralmente baseada no número de segundos, minutos ou dias transcorridos desde um instante inicial até a data considerada. Assim, para adaptá-las a um intervalo de datas válidas e ao intervalo de valores da dimensão, basta a aplicação de um *offset* e um fator de multiplicação. Caso o atributo tenha como valores possíveis uma lista discreta de valores, o mapeamento também é simples, bastando associar um intervalo dos valores naturais da dimensão a cada uma das possibilidades da lista.

Provavelmente, o mapeamento mais complexo diz respeito a parâmetros do tipo texto. A associação de uma seqüência de caracteres sem limite de tamanho a um número inteiro dentro de um intervalo não é uma tarefa trivial. Esse desafio já é bem conhecido no campo de bancos de dados relacionais. A criação de estruturas de índices para acelerar buscas baseadas em textos ou sub-textos já foi abordada em diversos trabalhos. Um algoritmo interessante é apresentado em [44], em que se trabalha com pesos diferentes para cada letra, sendo esse valor elevado à sua posição na seqüência. Dessa forma, o ordenamento resultante do mapeamento segue a ordem alfabética e permite buscas por prefixos. Operações anteriores aplicadas ao texto, como mudança de ordem das palavras pela relevância, podem ainda permitir buscas mais complexas sem alteração no algoritmo de mapeamento.

Outra questão que deve ser tratada é a possibilidade de haver na rede tipos de recurso com quantidades diferentes de índices. O número de dimensões da rede é, necessariamente, o limitante superior para o número de índices suportado numa busca. Entretanto, pode haver recursos que não utilizem todas as dimensões para buscas. Nesse caso, pode-se utilizar um algoritmo de mapeamento por funções *hash*, aplicado a um parâmetro específico ou à chave de identificação do recurso, para as dimensões restantes, de forma similar ao que é feito na arquitetura CAN. Como essas dimensões não serão usadas para buscas por intervalos de valor, não haverá impacto nas buscas suportadas.

3.3.1 Metadados

Em redes P2P de uso específico, os recursos compartilhados podem ser descritos de forma relativamente simples. Uma vez que todos os recursos disponíveis na rede são classificados pelos mesmos critérios, basta a utilização de arquivos muito simples, contendo apenas um conjunto limitado de informações pré-definidas. Em redes não estruturadas, focadas em compartilhamento de arquivos, não é nem mesmo necessária a existência de arquivos separados para descrição dos atributos, uma vez que são utilizadas apenas características inerentes ao recurso, como nome e tamanho, e não há separação entre as camadas de recurso e descrição. Por outro lado, a utilização de metadados, que nada mais são que informação sobre os próprios dados [45], é fundamental para possibilitar buscas efetivas em meio a imensos volumes de material, como os disponíveis na rede mundial de computadores *World Wide Web*, por permitir restrições por categorias e análises previamente computadas. Algumas dessas informações podem ser geradas automaticamente, pela análise do próprio conteúdo, enquanto outras necessitam ser explicitadas manualmente. Para uma rede que visa suportar buscas complexas por recursos de naturezas distintas, com critérios mutáveis, a utilização de metadados se torna indispensável. Há três formas de associar metadados aos recursos [46]:

- Embutida: a descrição do recurso é incluída no próprio arquivo que descreve, geralmente na forma de cabeçalho, e gerada no momento da criação do mesmo. Esta forma de associação é válida apenas para descrição de arquivos, não sendo possível para outros tipos de recurso;

- Associada: a descrição do recurso é armazenada em um arquivo independente, mas intimamente relacionado ao recurso e armazenado pela mesma entidade. Tem como vantagem a possibilidade de alteração dos metadados sem a necessidade de alteração no recurso;

- Metadados de terceiros: a descrição do recurso é armazenada de forma independente do recurso em si. O agente responsável por sua manutenção não possui, necessariamente, acesso ao recurso.

Na arquitetura proposta, são utilizados metadados de terceiros, uma vez que há uma divisão clara entre a camada de metadados e a camada de recursos, indicando não haver nenhuma relação entra a localização da descrição e o recurso em si. O fato de diversos tipos de recurso serem compartilhados na mesma rede torna necessária a inclusão de informações adicionais aos metadados, que indiquem qual o tipo de recurso sendo disponibilizado. Além disso, é necessária também a disponibilização de arquivos que descrevam os tipos de recurso, indicando os parâmetros válidos e suas associações às dimensões da rede.

Descrição de Recursos

Cada recurso disponibilizado na rede deve ter seu arquivo de descrição registrado na camada de metadados. Este arquivo deve estar armazenado no nó a cuja região de responsabilidade está associado o recurso. Toda vez em que ocorrer alguma mudança nas características do recurso, um novo arquivo de descrição deve ser publicado em substituição ao anterior. Como padrão de mercado, utiliza-se a linguagem de formatação extensível [47] (XML - *Extensible Markup Language)* para a representação de dados e metadados em um mesmo documento. Na rede proposta, as descrições de recurso devem possuir as seguintes informações:

- Tipo de Recurso: uma vez que a rede suporta vários tipos de recurso, deve fazer parte da descrição do recurso o identificador único do tipo a que está associado, de forma a limitar as respostas às buscas aos recursos do tipo desejado. Dependem do valor desse campo todos os outros atributos que deverão constar da descrição do recurso e os algoritmos de mapeamento que devem ser utilizados para converter os valores dos parâmetros usados a posições na rede;
- Valores dos Atributos Indexados: os valores dos parâmetros de classificação do recurso que estão associados a dimensões da rede. O formato e intervalo de valores possíveis variam para cada tipo de recurso. Estes valores definirão, após a aplicação dos algoritmos de mapeamento, o nó responsável pelo armazenamento da descrição na camada de metadados;

- Valores dos Atributos Não-Indexados: outros atributos possíveis para aquele tipo de recurso, mas que não estão associados a dimensões da rede. Embora não otimizem a execução das buscas, podem ser utilizados para limitar o número de respostas.

Apresenta-se na Figura 3.5 um exemplo de arquivo de descrição para um recurso do tipo “artigo científico”.

```
<descrição de recurso>
    <tipo do recurso>Artigo Cientifico </tipo do recurso>
    <valores de atributos indexados>
        <data>10-setembro-2006</data>
        <área>Engenharia</área>
    </valores de atributos indexados>
    <valores de outros atributos>
        <autor>João da Silva</autor>
        <tamanho>1000</tamanho>
        <título>Título de Exemplo</título>
    </valores de outros atributos >
</descrição de recurso>
```

Figura 3.5: Arquivo de descrição de recurso do tipo “Artigo Científico”.

Descrição de Tipo de Recurso

A utilização de arquivos de descrição de recurso seria suficiente para permitir o funcionamento da rede se todos os nós, ao ingressarem, já possuíssem a informação do significado de cada tipo de recurso. Isto é, quais os parâmetros suportados para aquele tipo de recurso, suas associações às dimensões da rede e os algoritmos de mapeamento a serem utilizados. Entretanto, essa premissa seria um obstáculo à utilização da rede, não permitindo que os tipos de recursos compartilhados fossem expandidos com o tempo e exigindo uma grande quantidade de informação prévia para o ingresso de um nó na rede. Para contornar esse problema, é proposta a criação de uma estrutura hierárquica de metadados, seguindo o modelo sugerido por Rusitschka e Southall [34]. A idéia central é que, além de existirem arquivos de descrição dos recursos, publicados na camada de metadados, existam também disponíveis na rede arquivos de descrição dos tipos de recurso. Para

cada tipo diferente de recurso apontado nas descrições de recurso, haverá disponível na rede um arquivo de descrição daquele tipo de recurso, contendo as seguintes informações:

- Identificador de Tipo: nome de identificação único para aquele tipo de recurso. Este nome é utilizado para referenciar, a partir dos arquivos de descrição de recurso, o tipo de recurso ao qual está associado;
- Atributos Indexados: atributos utilizados para classificar os recursos deste tipo. São descritos, nesse campo, os nomes e tipos de dados dos atributos que estão associados às dimensões da rede. Os tipos de dados definem o algoritmo de mapeamento a ser utilizado em cada dimensão;
- Atributos Não-Indexados: outros atributos que podem ser utilizados para classificação dos recursos, mas que não estão associados a dimensões do espaço.

É apresentado na Figura 3.6 um exemplo de arquivo de descrição de tipo de recurso, e como se associa ao arquivo de descrição de recurso.

Como pode ser observado, a descrição dos atributos se baseia na existência de "tipos básicos", como lista de valores, número inteiro, ou texto, que podem estar associados a alguns parâmetros de configuração, como os valores máximo e mínimo. Assim, o conhecimento prévio que cada nó deve possuir ao ingressar na rede fica restrito aos algoritmos de mapeamento associados aos tipos básicos, em função dos parâmetros de configuração. Os arquivos de descrição de tipo de recurso são disponibilizados na rede como recursos de um tipo específico: arquivo de descrição de tipo de recurso. Cada nó deve possuir, ao ingressar na rede, a descrição deste tipo de recurso, de forma a ser capaz de realizar buscas pelos descritores de todos os tipos de recurso que desejar.

3.4 Qualidade de Serviço

O termo qualidade de serviço em redes P2P geralmente se refere a garantias de desempenho e disponibilidade da camada de sobreposição. Desta forma, deseja-se garantir que as buscas são devidamente encaminhadas dentro da rede, e reduzir a possibilidade

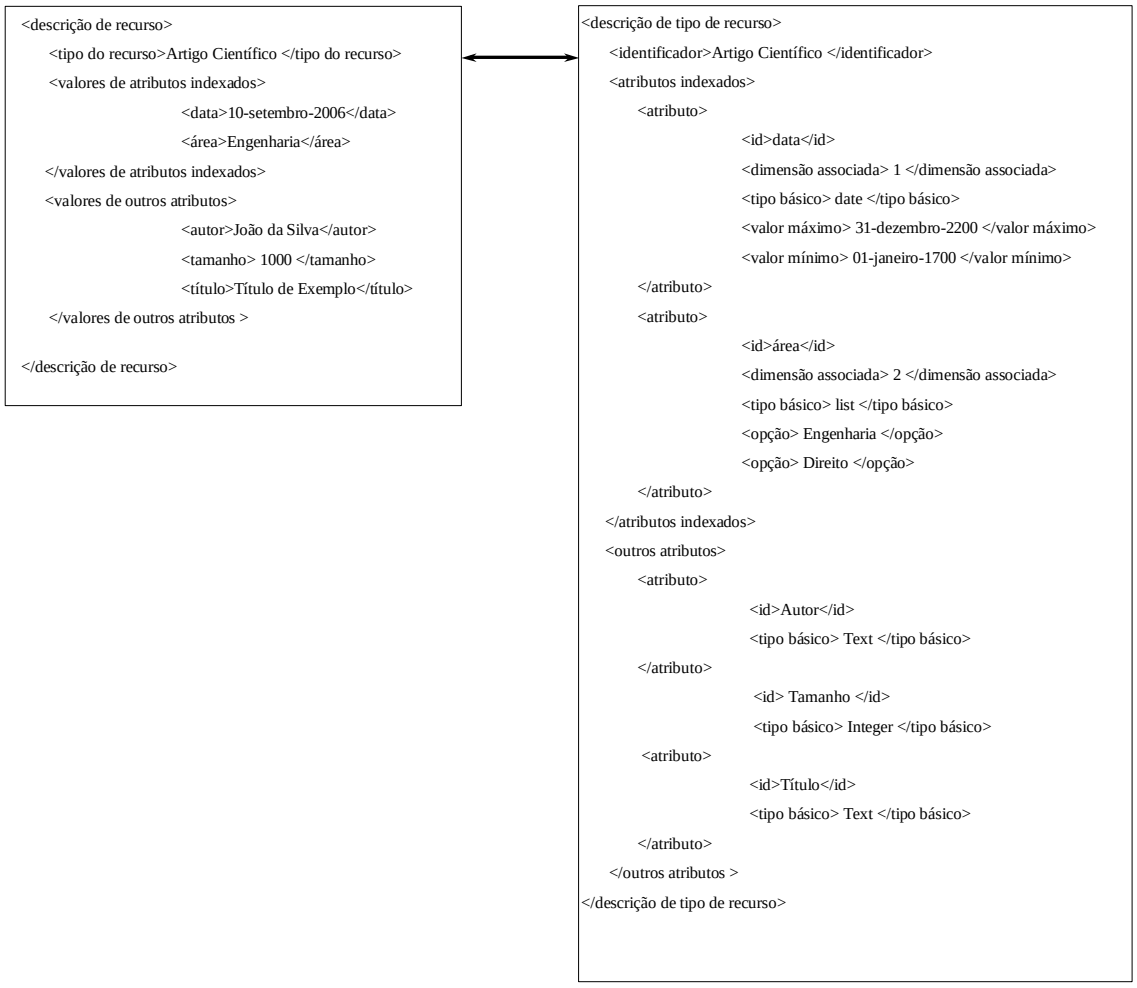

Figura 3.6: Exemplo de arquivo de descrição de tipo de recurso e de relacionamento hierárquico.

de existência de nós como gargalos para essas operações. Estas questões foram abordadas, para a rede proposta, nas Seções 3.2.4 e 3.2.3. Entretanto, o serviço entregue ao usuário não termina com a correta localização do recurso, e sim com o fornecimento do recurso solicitado. Nesse sentido, a arquitetura proposta atribui à camada de metadados também o papel de controle do nível de qualidade de serviço dos recursos sendo ofertados. Esse controle não pode ser feito de forma idêntica para todos os tipos de recurso. No caso de arquivos compartilhados, por exemplo, significa garantir a sua disponibilidade mesmo em caso de falha do nó ofertante, sendo necessário implementar algoritmos de replicação. Além disso, pode significar também armazenar e prover aos nós solicitantes

informações de qualidade de serviço experimentadas anteriormente no acesso ao recurso, evitando repassar ponteiros para recursos não mais disponíveis na rede ou a nós no limite de utilização de banda.

Já no caso de compartilhamento de capacidade de processamento, não há como implementar a garantia de disponibilidade por replicação. Entretanto, continua sendo interessante o armazenamento de informações históricas de uso, de forma a realizar balanceamento de carga também na camada de recursos. Também é possível a implementação de reserva antecipada de recursos, de forma a garantir, durante um período de tempo, que não haverá concorrência. É interessante passar ao nó responsável pelos metadados essa função de controle e armazenamento de histórico, uma vez que a utilização dos recursos é sempre precedida pela descoberta de sua descrição na camada de metadados. Assim, pode-se adicionar, nos arquivos de descrição de recurso, os níveis de qualidade de serviço desejados, para que sejam implementados pelo nó de controle. Pode-se verificar, na Figura 3.7, o novo arquivo de descrição de recurso contendo parâmetros de qualidade de serviço.

```
<descrição de recurso>
    <tipo do recurso>Artigo Científico </tipo do recurso>
    <valores de atributos indexados>
            <data>10-setembro-2006</data>
            <área>Engenharia</área>
    </valores de atributos indexados>
    <valores de outros atributos>
            <autor>João da Silva</autor>
            <tamanho>1000</tamanho>
            <título>Título de Exemplo</título>
    </valores de outros atributos >
    <valores de QoS requisitados>
            <replicações iniciais> 3 </replicações iniciais>
            <replicações mantidas> 2 </replicações mantidas>
    </valores de Qos requisitados>
</descrição de recurso>
```

Figura 3.7: Exemplo de arquivo de descrição recurso com parâmetros de qualidade de serviço.

Fica claro que o arquivo de descrição de tipo de recurso também deverá ser alterado,

de forma a indicar os parâmetros de qualidade de serviço suportados para aquele grupo de recursos. Estes parâmetros devem estar associados a algoritmos, descritos em uma linguagem compreendida universalmente. Tais algoritmos podem ser disponibilizados na rede como recursos, de forma a permitir que os nós os aprendam conforme a necessidade.

Capítulo 4

Análise de Desempenho

Neste capítulo são apresentadas as expressões analíticas de desempenho desenvolvidas para a rede e os resultados de simulação, considerando diversas configurações para a rede.

4.1 Resultados Analíticos

Dada a rede proposta, são feitas nesta seção as previsões teóricas para os custos envolvidos na execução dos principais algoritmos de busca, executados na camada de metadados.

4.1.1 Busca por Recurso Único

A tarefa mais simples a ser executada na rede é a busca por um único recurso. Esta busca é traduzida em atingir o nó que possui em sua região de responsabilidade um determinado ponto do espaço, a partir de um outro ponto aleatório (o nó solicitante). Neste caso, não há nenhuma diferença entre o algoritmo em execução e aquele de encaminhamentos por aproximação sucessiva utilizado na arquitetura CAN. Primeiramente, considera-se uma rede com n nós, organizada igualmente em d dimensões, e com divisão uniforme do espaço entre os nós participantes. Ressalta-se ainda que cada busca pode ser

encaminhada em qualquer um dos sentidos (crescente ou decrescente), para cada uma das dimensões, uma vez que os nós máximos de uma dimensão são também vizinhos dos nós mínimos. Assim, o número médio de nós no menor caminho entre dois nós aleatórios da rede é

$$N_{busca}(n) = (d/4)(n^{1/d}). \tag{4.1}$$

Esta expressão indica a necessidade de percorrer, em média, 1/4 dos nós em cada uma das dimensões da rede. Entretanto, isso é válido apenas para redes com divisão uniforme do espaço. Caso seja aplicado um algoritmo de balanceamento de carga, haverá nós com diferentes tamanhos de região sob sua responsabilidade, sendo essa variação de tamanho dependente do nível de concentração dos recursos no espaço. Dessa forma, a decisão que é tomada por encaminhamento no sentido crescente ou decrescente de cada dimensão, tendo como base a distância geométrica entre os pontos, deixa de indicar necessariamente o menor número de saltos. Essa diferença pode ser visualizada na Figura 4.1, em que o encaminhamento baseado no menor percurso geométrico levaria a 4 nós participando na busca, enquanto o caminho de maior percurso geométrico exigiria apenas 3.

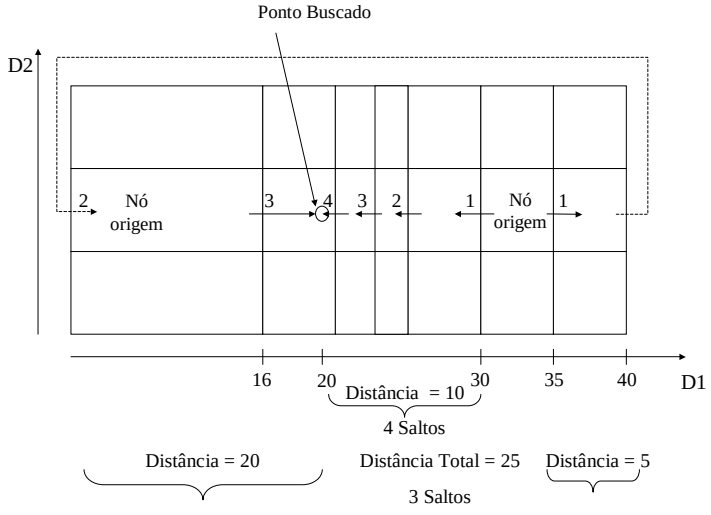

Figura 4.1: Organização de nós com regiões desiguais de responsabilidade.

Dessa forma, no pior caso, quando assume-se que a área de responsabilidade de cada nó é aleatória, não havendo um tamanho mínimo para este valor, o número médio de nós é dado por

$$N_{busca}(n) = (d/2)(n^{1/d}). \tag{4.2}$$

4.1.2 Inclusão e Exclusão de Recurso

Os algoritmos para inclusão e exclusão de recursos na rede são idênticos ao de busca por um recurso único. Dessa forma, o número médio de saltos para execução destas tarefas é, a princípio, o mesmo.

A diferença ocorre quando se utiliza alguma técnica de replicação de metadados, de forma a garantir disponibilidade em casos de falha. No caso da técnica sugerida na Seção 3.2.4, haverá a necessidade de um salto a mais, que é feito do nó responsável pelo ponto associado ao recurso para um de seus vizinhos, que contém a replicação. Dessa forma, a previsão para o número médio de nós envolvidos na inclusão ou exclusão de um recurso, para uma rede com nós de mesmo tamanho de região, é expressa por

$$N_{inclusao}(n) = (d/4)(n^{1/d}) + 1. \tag{4.3}$$

4.1.3 Busca por Intervalos de Valor

Conforme descrito na Seção3.2.1, a execução de uma busca por intervalos de valor pode ser dividida em duas etapas: alcance da região desejada e varredura dentro desta região. Dessa forma, o número médio de nós envolvidos neste tipo de busca pode ser decomposto em duas parcelas

$$N_{intervalo} = N_{alcance} + N_{varredura}. \tag{4.4}$$

Este valor médio pode ser definido em função dos seguintes parâmetros:

- n: número total de nós participantes da rede;
- d: número de dimensões da rede;
- k: número de índices utilizados para filtragem na busca;
- Razão R/U: relação entre o tamanho da região R buscada e todo o universo U.

Para todo o desenvolvimento é considerada uma rede dividida igualmente entre os nós e que a busca aplicada é igualmente restritiva nos *k* índices filtrados.

Etapa de Alcance

Na etapa de alcance, é executado o algoritmo de busca por um ponto único da rede, semelhante à busca por um recurso único. Entretanto escolhe-se, dentro da região buscada, o ponto geometricamente mais próximo ao solicitante, podendo este inclusive já fazer parte da região, o que elimina a etapa de alcance. Uma vez que a busca é igualmente restritiva nos *k* índices, a relação entre o comprimento do intervalo buscado em uma dimensão e o tamanho total da dimensão é $(R/U)^{1/k}$. Deve-se considerar também a probabilidade de que nó solicitante já faça parte da região buscada, em uma ou mais dimensões, fazendo com que o número de nós envolvidos na etapa de alcance seja zero naquela dimensão. Caso isso não ocorra, basta que seja atingida uma das extremidades da região buscada. Assim, o comprimento restante, em número de nós, possível de ser percorrido em cada dimensão é $n^{1/d}[1-(R/U)^{1/k}]$, e não toda a dimensão. Há ainda a possibilidade de o que nó solicitante já estaja localizado na região buscada, para cada uma das dimensões. Nesse caso, o número de saltos naquela dimensão vai a zero. Nas dimensões em que não há filtro aplicado, não há necessidade de realizar salto algum. Dessa forma, a expressão do número médio de nós envolvidos na etapa de alcance é dada por

$$N_{alcance} = \frac{k}{4}(n^{\frac{1}{d}}).[1-(R/U)^{\frac{1}{k}}].[1-\frac{n^{\frac{1}{d}}.(R/U)^{\frac{1}{k}}}{n^{\frac{1}{d}}-1}]. \tag{4.5}$$

Etapa de Varredura

Na etapa de varredura, estão sendo acessados apenas os nós que fazem parte da região buscada. Dessa forma, o número ideal de nós a ser acessado é $n(R/U)$, seguindo a proporção entre a região buscada e todo universo. Entretanto, essa proporção não é seguida perfeitamente, uma vez que, de forma geral, há um descasamento entre as fronteiras da região buscada e as regiões de responsabilidade dos nós. Esse efeito pode ser verificado claramente na Figura 3.2. Nela, embora a região R buscada represente apenas 1/9 do espaço, participariam da busca 4/9 dos nós, uma relação muito maior. Para definir o número médio de nós nessa etapa, faz-se uma análise do efeito desse descasamento em cada dimensão.

Dada uma dimensão utilizada para restringir a busca, com comprimento total igual

a 1, sua fração associada a busca tem tamanho igual a $(R/U)^{\frac{1}{k}}$. Assim, dependendo da defasagem x_0 entre as fronteiras das regiões buscada e dos nós, podem participar da busca, naquela dimensão, um ou dois nós a mais, conforme Figura 4.2.

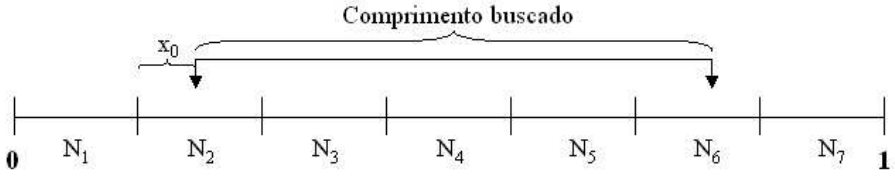

Figura 4.2: Defasagem entre a região buscada e as fronteiras dos nós em uma dimensão.

Considerando P a probabilidade desta defasagem causar a participação de 2 nós a mais em uma dimensão da rede, e $W = (R/U)^{\frac{1}{k}}$, a expressão desenvolvida para o valor de $N_{varredura}$ é

$$N_{varredura} = [(Int(W.n^{\frac{1}{d}}) + 2)P + (Int(W.n^{\frac{1}{d}}) + 1)(1 - P)]^k.(n^{\frac{1}{d}})^{d-k}, \quad (4.6)$$

Sendo $Int(x)$ a função que retorna a parte inteira do número x. A probabilidade P é expressa por

$$P = W.n^{\frac{1}{d}} - Int(W.n^{\frac{1}{d}}). \quad (4.7)$$

4.2 Resultados de Simulação

Para realizar a simulação da rede proposta, foi desenvolvido um simulador em linguagem C++. Devido ao grande número de nós envolvidos na simulação (centenas de milhares), a utilização de ferramentas previamente construídas, como o Omnetpp [48] ou o Network Simulator [49] mostrou-se inviável. Já as ferramentas construídas especificamente para simulação de redes P2P, como a PeerSim [50], permitem a simulação de um grupo restrito de arquiteturas, exigindo grandes alterações para a simulação desejada. Dessa forma, foi criado um código-fonte especificamente construído para as simulações apresentadas. Os nós da rede foram codificados em uma classe que possui variáveis para definição de sua região de responsabilidade, ponteiros para nós vizinhos e métodos que implementam os algoritmos de encaminhamento de buscas. Para cada rede de simulação, foi criada uma matriz de nós com tamanho diferente, refletindo a variação no número de

nós e de dimensões. As redes criadas para as simulações possuem sempre nós com regiões de responsabilidade de mesmo tamanho, em todas as dimensões, estando as redes, assim, divididas uniformemente em todas as dimensões.

4.2.1 Busca por Recurso Único

Primeiramente, foram feitas simulações para medição do número de nós envolvidos nas buscas por recursos únicos na rede. Para cada ponto de simulação, foi gerada uma rede organizada segundo o número de dimensões definido e com universo uniformemente dividido em todas as dimensões, estando todos os nós responsáveis por regiões de mesmo tamanho. Os nós solicitantes também foram definidos de forma aleatória. Foi obtida a média da execução de 1000 buscas por recursos com posicionamento aleatório na rede. Para todos os resultados de simulação, foram observados intervalos de confiança de 95% menores que 2% do valor dos resultados, não sendo apropriada sua representação no gráfico. A simulação foi repetida para redes com diversas quantidades de nós, de forma a verificar a escalabilidade das buscas. Também foi variada a quantidade de dimensões em que a rede se organiza. O resultado é apresentado na Figura 4.3.

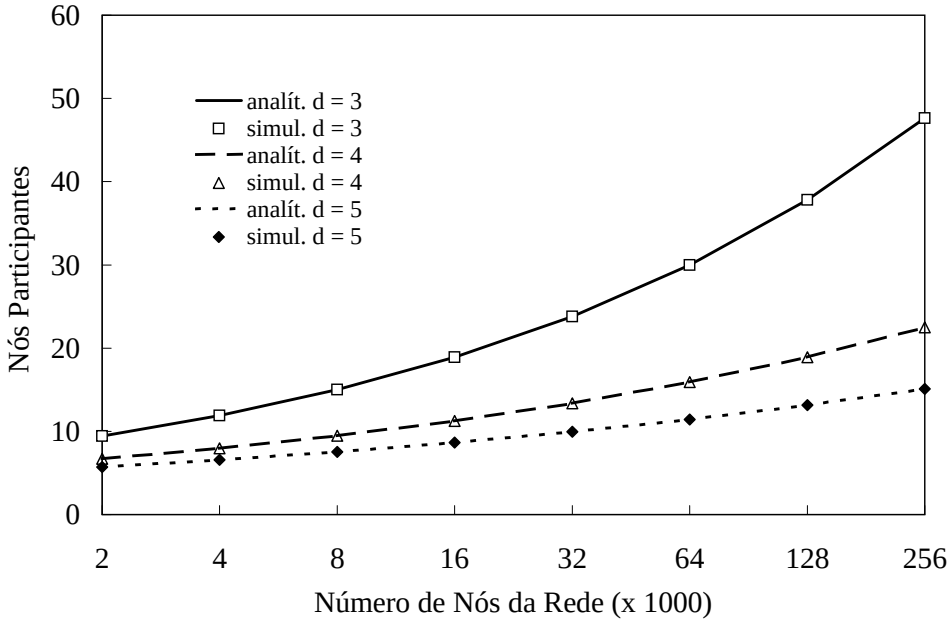

Figura 4.3: Número médio de nós participantes em busca por recurso único.

Como pode-se observar, a rede apresenta escalabilidade para suportar crescimento acentuado no número de nós participantes. Enquanto há crescimento exponencial no número de nós da rede, o número de nós envolvidos nas buscas cresce de forma próxima à linear durante boa parte do gráfico. O número de nós envolvidos na busca escala com ordem $d(n^{1/d})$, conforme previsto na análise teórica. Também observa-se que, para buscas por recursos únicos, o aumento no número de dimensões aumenta a eficiência da busca e garante escalabilidade maior.

4.2.2 Busca por Intervalos de Valor

Para a verificação da eficiência da rede em buscas por intervalos de valor, foram feitas simulações em função de três variáveis: relação R/U entre o tamanho da região buscada e o espaço total da rede, número de nós e número de dimensões da rede. Os resultados são apresentados de forma normalizada em relação ao resultado ideal, que corresponde a uma relação n/N, entre os nós que participam da busca e o total de nós da rede, igual à relação R/U entre o tamanho da região buscada e o tamanho de todo o universo. Para cada caso de simulação, foi gerada uma rede organizada no número de dimensões apropriado, possuindo nós com regiões de responsabilidade de mesmo tamanho. As buscas são solicitadas por nós definidos aleatoriamente.

Primeiramente, foi utilizada uma rede de 4 dimensões, em que os 4 índices são utilizados para restrição às buscas. As buscas executadas são igualmente restritivas em todas as dimensões da rede. São variadas a quantidade de nós na rede e a relação R/U das buscas executadas, o resultado é apresentado na Figura 4.4. Para cada ponto simulado, foram executadas 1000 buscas por regiões aleatórias, respeitando a relação R/U definida, solicitadas por nós aleatórios na rede. Para todos os resultados de simulação, foram observados intervalos de confiança de 95% menores que 1% do valor dos resultados, não sendo apropriada sua representação no gráfico.

O gráfico obtido comprova algumas características interessantes da rede, previstas na expressão matemática deduzida. A primeira característica clara é que o desempenho da rede melhora conforme cresce a região buscada. Este comportamento é resultado da

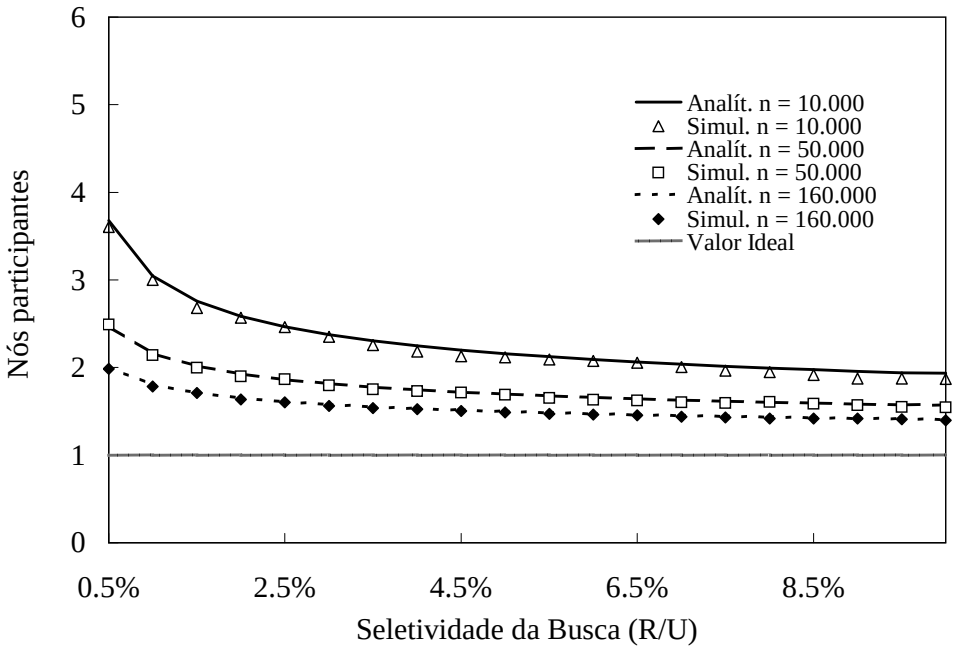

Figura 4.4: Número médio de nós participantes em busca por intervalos de valor, normalizado em relação ao valor ideal. Em função da seletividade R/U da busca, em rede com 4 dimensões.

redução do impacto da defasagem entre os limites das regiões dos nós e os limites da região buscada. Esse efeito é mostrado na Figura 4.2. Uma vez que esse descasamento tem impacto fixo, adicionando um máximo de 2 nós em cada dimensão, quanto maior é a região buscada, menor é o impacto percentual desse efeito. Esse é o mesmo motivo pelo qual pode-se perceber também a obtenção de melhores resultados para redes com maior número de nós. Quanto maior o número de nós necessários para cobrir uma região buscada menor o impacto percentual da adição de um ou dois nós em cada dimensão.

Na simulação a seguir, é mantido o tamanho da rede fixo em 160.000 nós, para verificação do efeito da variação no número de dimensões. Para cada caso de simulação, foi gerada uma nova rede, organizada no número de dimensões apropriado. Para esta simulação, também foi extraída para cada caso a média do número de nós envolvidos em 1000 buscas aleatórias, mantendo a relação R/U definida, e sendo solicitadas por nós aleatórios. Todos as dimensões da rede são restringidas em igual proporção.

Pode-se perceber que, para buscas por intervalos, a rede possui desempenho melhor quando está organizada em menos dimensões. Esse efeito é compatível com a expres-

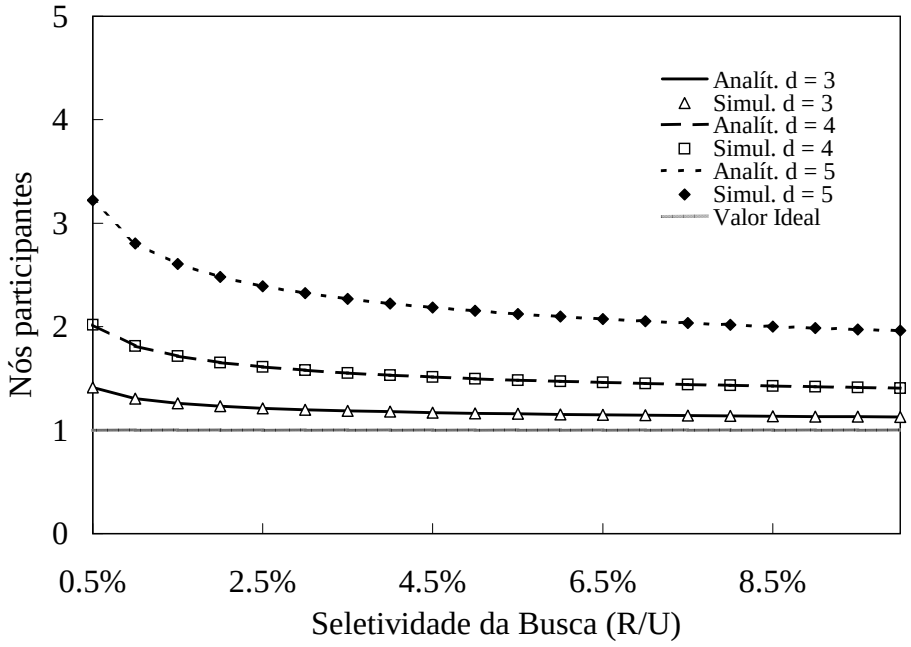

Figura 4.5: Número médio de nós participantes em busca por intervalos de valor, normalizado em relação ao valor ideal, em rede com 160.000 nós.

são matemática obtida. Isso ocorre pois, quanto maior o número de dimensões para um mesmo número de nós, menor é o número de divisões em cada uma delas, ficando cada um dos nós responsável por comprimentos maiores. Dessa forma, o efeito da defasagem, adicionando um ou dois nós em cada dimensão, tem um impacto percentual maior no número de nós envolvidos na busca. Essa característica pode representar um limitador para a utilização da arquitetura, uma vez que o número de dimensões define o número máximo de índices para buscas. Caso seja desejado, para algum tipo de recurso, a execução de buscas eficientes utilizando muitos parâmetros, seria necessário utilizar uma rede com grande dimensionalidade, o que acarretaria em perda acentuada de desempenho. Dessa forma, torna-se necessário fazer a escolha de um valor de compromisso para o número de dimensões da rede.

4.2.3 Buscas por Tipos Diferentes de Recursos

Conforme detalhado no Capítulo 3, a rede tem como objetivo permitir o compartilhamento de diversos tipos de recursos. Na prática, isso pode resultar em tipos de recurso

com uma quantidade de índices menor que o total de dimensões na rede. Dessa forma, a distribuição dos recursos em uma ou mais dimensões é feita utilizando uma função *hash*, aplicada a um ou mais parâmetros do recurso, de forma a facilitar o balanceamento de carga nesta dimensão. Para a execução de buscas, isto se reflete na aplicação de filtros em menos dimensões. Este mesmo efeito acontece na busca por recursos quando se aceita qualquer valor para um ou mais índices. Assim, é apresentado na Figura 4.6 o número médio de nós envolvidos em buscas, em relação ao valor ideal $n(R/U)$, em simulações em que o número de índices *k* filtrados é menor que o número de dimensões da rede. É utilizada uma rede de 4 dimensões, com 160.000 nós. Para cada ponto simulado, foi calculada a média da execução de 1000 buscas por regiões aleatórias, respeitando a relação R/U definida, solicitadas por nós aleatórios na rede. Para todos os resultados de simulação, foram observados intervalos de confiança de 95% menores que 1% do valor dos resultados, não sendo apropriada sua representação no gráfico.

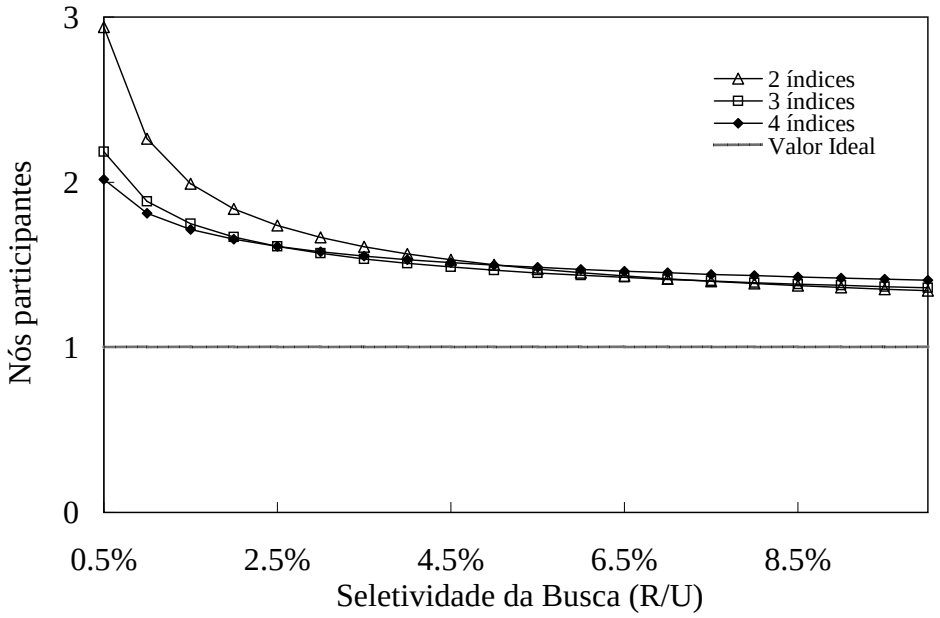

Figura 4.6: Número médio de nós participantes em busca por intervalos de valor, normalizado em relação ao valor ideal, em rede com 160.000 nós e 4 dimensões.

O efeito da utilização de menos índices para filtragem da busca afeta o desempenho da rede apenas em buscas por regiões reduzidas. Para as buscas em que há maior participação de nós da rede, em que o efeito degradante seria mais prejudicial, não há perda de desempenho e percebe-se até mesmo melhorias a partir de certo ponto. Assim, a execução

de buscas com quantidades diferentes de parâmetros não se apresenta como entrave para a utilização da rede.

4.2.4 Replicação de Dados

Para garantir a disponibilidade do serviço de busca, é implementada a replicação de metadados na rede. O método proposto na Seção 3.2.4 determina a cópia dos metadados de cada nó para seu vizinho superior em uma ou mais dimensões, dependendo do número de replicações desejadas. O resultado de simulação apresentado na Figura 4.7 demonstra o efeito da aplicação dessa técnica, com replicação em uma dimensão, sobre o desempenho das buscas por intervalos. É utilizada uma rede de 4 dimensões, composta por 160.000 nós. O impacto da replicação na eficiência das buscas deve ser maior em redes com mais dimensões e menos nós, uma vez que a técnica utilizada minimiza a degradação de desempenho decorrente da defasagem entre os limites da região buscada e das regiões dos nós, mais grave nesses casos. Para cada ponto da simulação foi extraída a média do resultado de 1000 buscas aleatórias, respeitando a relação R/U e com filtros aplicados aos 4 índices, de forma igualmente restritiva. Para todos os resultados de simulação, foram observados intervalos de confiança de 95% menores que 1% do valor dos resultados, não sendo apropriada sua representação no gráfico.

Conforme esperado, a utilização dessa técnica de replicação também traz benefício para a eficiência da busca. Em buscas por regiões que representam um menor percentual do universo da rede, o impacto chega a 20% de redução no número de nós envolvidos. Conforme este número cresce, o ganho de desempenho se estabiliza próximo a 10% de redução. A utilização dessa técnica de replicação demonstra ser mais vantajosa para a busca por intervalos que a utilização de múltiplas realidades, conforme proposto para rede CAN [11], que não traria ganhos de desempenho para esse tipo de busca.

4.2.5 Comparação de Desempenho

A comparação da rede organizada por índices com propostas anteriores não é simples. Conforme detalhado no Capítulo 2, cada uma das arquiteturas possui fatores limitantes em

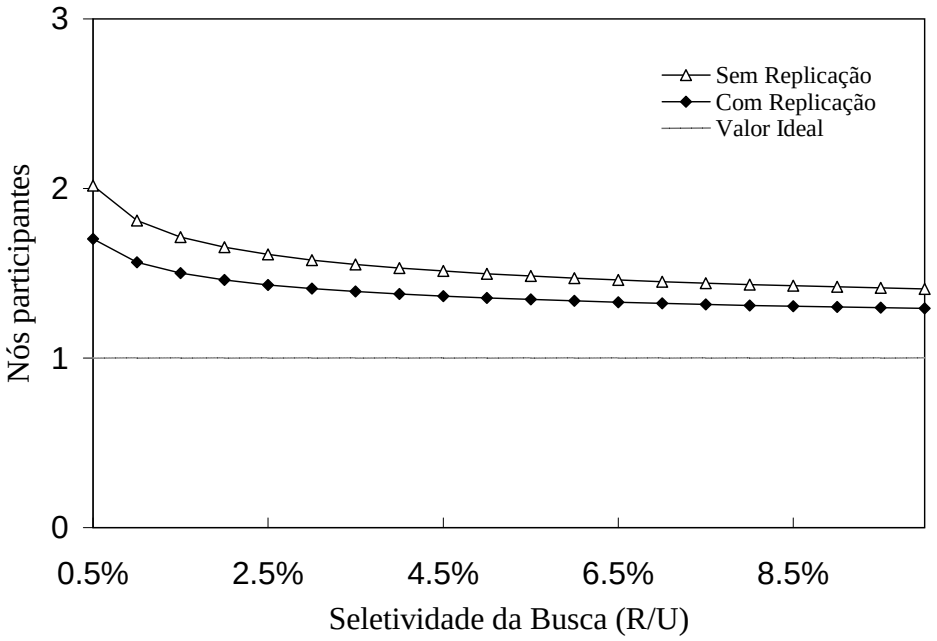

Figura 4.7: Número médio de nós participantes em busca por intervalos de valor, normalizado em relação ao valor ideal, em rede com 160.000 e 4 dimensões.

alguma das características desejadas, tornando uma simples comparação de desempenho das buscas insuficiente para definir a escolha da arquitetura. A arquitetura com suporte a tipos de busca mais próxima à rede organizada por índices é a rede Mercury [33], detalhada na Seção 2.5.2. Uma vez que também suporta busca por intervalos de valor com diversos parâmetros, é possível fazer uma comparação direta entre as duas redes. A rede Mercury pode estar configurada para trabalhar de duas formas: sem replicação de metadados ou com replicação total. Na primeira modalidade, o objetivo é otimizar o custo de armazenamento e do processo de publicação de recursos na rede, havendo na rede apenas uma cópia do arquivo de descrição de cada recurso. Já na segunda modalidade, faz-se otimização da busca, havendo para cada dimensão da rede uma cópia do arquivo.

Quando operando em otimização para armazenamento, a rede Mercury apresenta desempenho muito ruim para buscas, exigindo consultas a todos os seus anéis [33]. Por isso, para a comparação entre o desempenho das redes em buscas por intervalos de valor, a rede Mercury é utilizada na modalidade de otimização de busca, havendo d cópias de cada arquivo de metadados na rede. A rede Mercury foi simulada pela criação de código-fonte C++, com a utilização de uma classe para a descrição de nós da rede e a geração

de várias listas encadeadas circulares de objetos, formando os diversos anéis da rede. As redes utilizadas são compostas de 160.000 nós, e possuem 4 índices de busca, refletidos em dimensões na rede organizada por índices e em anéis na rede Mercury, uniformemente divididos entre os nós. A quantidade de nós utilizada na comparação foi definida considerando o fato de que ambas as redes têm desempenho otimizado para maiores quantidades de nós. A rede Mercury, ao contrário da rede organizada por índices, melhora sua eficiência com o aumento no número de dimensões, mas exige um maior número de réplicas de metadados. Dessa forma, foi fixado em 4 o número de cópias dos metadados de cada recurso nas redes. É esperado que para redes com maior número de dimensões, o desempenho da rede Mercury melhore, com efeito inverso na rede organizada por índices. No caso de aumento no número de nós, é esperada melhora de desempenho para ambas as redes. Assim como nas simulações anteriores, os filtros aplicados são igualmente restritivos em todas as dimensões. Uma vez que a rede Mercury provê várias técnicas para criação de ponteiros de salto, diminuindo o número de nós na etapa de alcance, consideramos para essa rede apenas os nós envolvidos na etapa de varredura. Para igualar o espaço em disco ocupado por metadados nas redes, também foi aplicada na rede organizada por índices a replicação de arquivos em 3 dimensões, totalizando 4 cópias. Foram executadas 1000 buscas aleatórias em cada rede, respeitando a relação R/U, para cada caso de simulação. Para todos os resultados de simulação, foram observados intervalos de confiança de 95% menores que 1% do valor dos resultados, não sendo apropriada sua representação no gráfico. A Figura 4.8 mostra o resultado obtido em cada uma das redes, normalizado em relação ao valor ideal $n(R/U)$, em função da seletividade da busca.

Conforme resultado da simulação, a rede proposta possui, principalmente para buscas mais restritivas, desempenho muito superior à rede Mercury. A diferença de desempenho se torna pequena quando a região buscada representa um percentual alto do espaço. Isso ocorre pois, na rede Mercury, os índices de busca estão completamente separados entre os anéis, não sendo possível a aplicação de diversos filtros simultaneamente. De fato, escolhe-se apenas o parâmetro mais restritivo para ser usado com filtro. No caso de buscas executadas com parâmetros mais restritivos que outros, a diferença de desempenho esperada é menor. Para verificar essa previsão, foram realizadas simulações em que apenas dois dos quatro parâmetros são utilizados para restrição da busca. Foram executadas

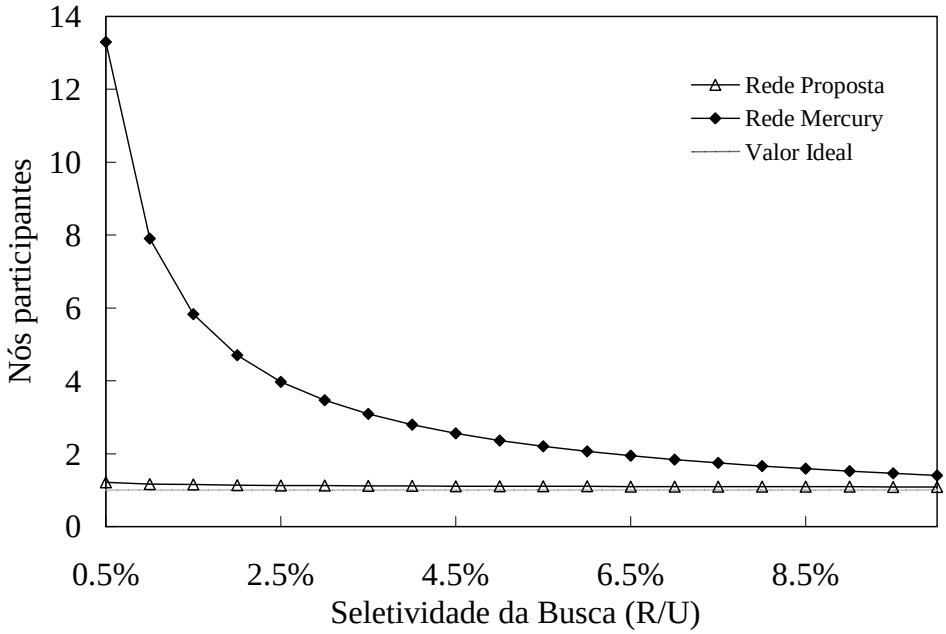

Figura 4.8: Comparação do desempenho da rede proposta, em relação à rede Mercury, em número médio de nós participantes em buscas por intervalos de valor.

1000 buscas aleatórias em cada rede, respeitando a relação R/U, para cada caso de simulação. Para todos os resultados de simulação, foram observados intervalos de confiança de 95% menores que 1% do valor dos resultados, não sendo apropriada sua representação no gráfico. Os resultados são apresentados na Figura 4.9.

Pelo gráfico fica claro que, quando a busca executada apresenta grande disparidade de seletividade entre os parâmetros, a rede Mercury poderá apresentar, para buscas muito extensas, resultado similar ou até melhor que a rede proposta.

Entretanto, deve-se ressaltar que todas as simulações foram realizadas para a utilização de rede Mercury em modo otimizado para buscas, que exige grande grau de replicação dos metadados. Caso fosse necessário restringir a replicação de dados, para trabalhar em modo otimizado de espaço de armazenamento ou para permitir maior freqüência de entrada e saída de nós, a rede Mercury apresentaria resultados *d* vezes maiores para execução das buscas. Assim, a rede proposta permite um ajuste mais fino para a definição do número de replicações desejadas.

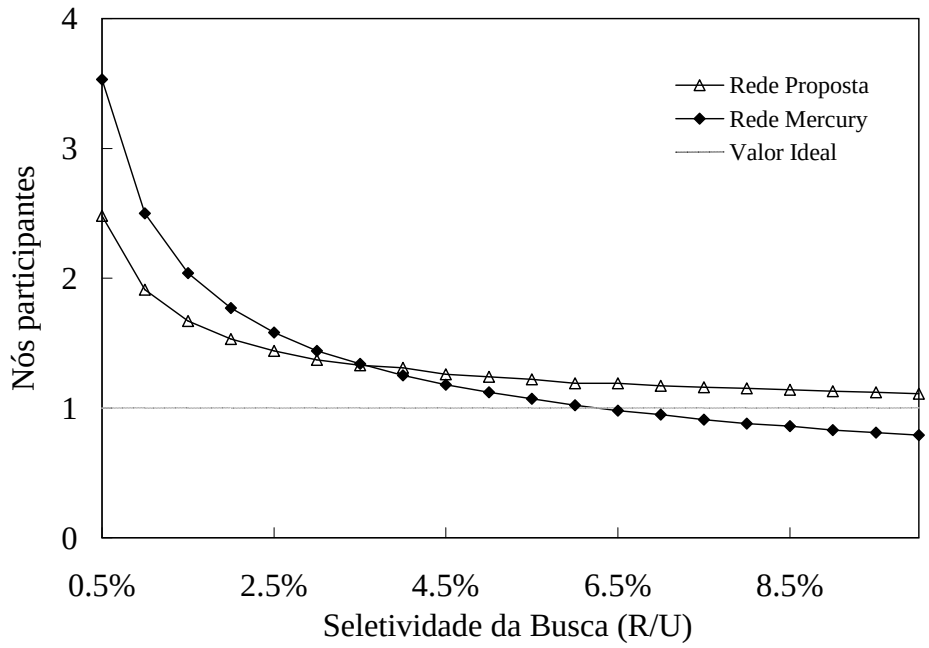

Figura 4.9: Comparação do desempenho da rede proposta, em relação à rede Mercury, utilizando-se apenas 2 parâmetros para filtragem da busca.

Capítulo 5

Conclusões

A Utilização de serviços baseados em P2P teve uma velocidade de adoção sem precedentes. Nos últimos anos, verificou-se o surgimento, popularização e desaparecimento de centenas de sistemas para compartilhamento de diversos tipos de recursos, focados nos mais distintos públicos. Um olhar mais crítico sobre o sucesso dos serviços P2P aponta que a facilidade de desrespeito a leis de restrição à distribuição de conteúdo, como as leis de direitos autorais, foi em grande parte a responsável pela adesão tão intensa à tecnologia. Muito mais que explorar de forma eficiente recursos ociosos da rede, as aplicações P2P mais populares se utilizam das características técnicas da topologia descentralizada como meio de garantir anonimato e impedir a monitoração de conteúdo, o que é comumente chamado de “democratização da rede”.

Por outro lado, este enorme sucesso desperta o interesse na exploração de forma legítima dos benefícios da arquitetura descentralizada. As aplicações amadoras deixaram clara a alta robustez e otimização de recursos possíveis, a custos muito reduzidos e utilizando infra-estrutura simples. Dessa forma há, atualmente, muitas aplicações com fins educacionais e científicos baseadas em redes P2P. Até mesmo serviços pagos de comunicação por voz sobre IP (VoIP) já fazem uso de P2P.

É exatamente na incorporação de funções que permitam a aplicação de redes P2P a novas áreas, incluindo fins corporativos, que se concentram os esforços das pesquisas. As adaptações feitas às redes iniciais, que possuíam organização aleatória dos nós, alcan-

çou seu limite. Ainda que mecanismos de *caching* e conexões por interesse melhorem a eficiência das buscas por recursos, não alteram as características fundamentais da rede, impossibilitando buscas completas ou garantias de qualidade de serviço. O desenvolvimento de arquiteturas estruturadas, baseadas em DHT, representou um importante avanço, garantindo alto nível de eficiência nas buscas e conseqüente escalabilidade. Entretanto, junto a essa eficiência, foi eliminada a possibilidade de realização de buscas flexíveis, sem conhecimento prévio específico do identificador do recurso desejado.

Nesse trabalho, é apresentada uma nova arquitetura para redes P2P, visando uma maior aderência aos diversos requisitos de flexibilidade de buscas por recursos. A idéia fundamental da proposta é alinhar a organização dos nós na rede às necessidades de buscas, fazendo com que o próprio posicionamento do nó na rede contenha informação sobre seu conteúdo. É utilizado o conceito de índices, que são parâmetros de classificação dos recursos com relevância para buscas. Cada um desses índices está refletido em uma dimensão em que a rede se organiza, e cada nó possui dois vizinhos, superior e inferior, em cada dimensão. A arquitetura proposta é capaz de executar de forma eficiente buscas por recursos únicos, bem como por intervalos de valor de parâmetros, graças à correspondência entre proximidade dos nós na rede e afinidade semântica. Esta organização também permite a localização de recursos com atributos máximos e mínimos. Essa capacidade é fundamental para o compartilhamento de diversos tipos de recurso e a utilização dos mesmos de forma otimizada. Além de prover escalabilidade, a rede permite ainda a implementação de garantia de disponibilidade na execução das buscas, por meio de replicação de metadados. É previsto ainda o suporte ao compartilhamento de diversos tipos de recursos em uma mesma rede, sem perda de eficiência, questão geralmente desconsiderada nas atuais redes em funcionamento.

Foi desenvolvida inicialmente uma análise teórica do desempenho esperado para a rede, tanto na execução de buscas por recursos únicos como por intervalos de parâmetro. Foram realizadas também simulações de custo de execução para as buscas, utilizando diversas configurações, com variações no número de nós, dimensões e índices utilizados. Os resultados demonstraram desempenho superior da rede para buscas mais abrangentes e para redes com maior quantidade de nós. Estes resultados apontaram escalabilidade adequada. Foi identificado um compromisso entre o desempenho da rede e a quantidade

de dimensões em que se organiza. Assim, ainda que a utilização de muitas dimensões permita buscas por maior quantidade de parâmetros, a eficiência da rede é degradada. A flexibilidade em suportar diversos tipos de recurso foi comprovada pela manutenção da eficiência mesmo em buscas que utilizam menor quantidade de índices para filtragem. Além disso, a utilização de replicação de metadados em uma ou mais dimensões se mostrou eficaz no incremento do desempenho da rede em buscas por intervalo de valor. Por fim, foi feita ainda a comparação de desempenho da rede proposta com a rede Mercury, que permite execução de buscas semelhantes. De forma a permitir uma comparação equilibrada, foram sempre igualados os números de cópias de metadados existentes nas redes. Os resultados obtidos demonstram um melhor desempenho da rede proposta para a maior parte das situações. Apenas em casos em que se utilizam muitas replicações de metadados e em que as buscas apresentam níveis de restrição muito desiguais entre os vários índices, a rede Mercury apresenta desempenho superior. Assim, a proposta de organização da rede se apresenta como opção interessante para implementação de serviços que requeiram o compartilhamento de diversos tipos de recursos, com buscas flexíveis.

Uma questão que não foi abordada neste trabalho e que se apresenta como um ponto importante para trabalho futuro é a implementação de mecanismos de segurança para a rede. Esse aspecto tomou grande importância em trabalhos recentes e apresenta-se como um grande desafio para serviços totalmente descentralizados. A existência de nós mal intencionados pode afetar consideravelmente o desempenho da rede, apontando para recursos inexistentes, ocupando a rede com pedidos desnecessários ou bloqueando a execução de buscas. Outra questão relevante é a otimização de desempenho considerando os tempos de resposta e banda de conexão existentes entre os nós. Diversas técnicas propostas em outros trabalhos podem ser aplicadas também à rede proposta, de forma a garantir que nós com melhor conexão entre si se mantenham próximos na rede. Foi analisado neste trabalho o desempenho da rede baseado apenas em número de nós para execução das operações, sem se aprofundar nas questões das camadas inferiores de rede. A análise considerando todos esses efeitos pode ser conseguida numa implementação futura, com disponibilização para uso prático de aplicações baseadas na rede proposta.

Referências Bibliográficas

[1] BALAKRISHNAN, H., KAASHOEK, M. F., KARGER, D., MORRIS, R., E STOICA, I. Looking Up Data in P2P Systems. *Communications of the ACM 46*, 2 (2003), 43–48.

[2] TAN, T. *Edonkey - Still King of P2P in France and Germany*, 2005. http://www.sandvine.com/news/pr_detail.asp?ID=88. Acessado em 20 de fevereiro de 2007.

[3] BitTorrent: The "one third of all Internet traffic" Myth. http://torrentfreak.com/bittorrent-the-one-third-of-all-internet-traffic-myth. Acessado em 27 de fevereiro de 2007.

[4] SILVESTRE, G., KAMIENSKI, C., FERNANDES, S., MARIZ, D., E SADOK, D. Análise de Tráfego Peer-to-Peer Baseada na Carga Útil dos Pacotes. Em *XXIV Simpósio Brasileiro de Redes de Computadores - II Workshop de Peer-to-Peer* (Curitiba, PR, Brasil, maio de 2006), pág. 91–102.

[5] DAVID P. ANDERSON, JEFF COBB, ERIC KORPELA, MATT LEBOFSKY E DAN WERTHIMER. SETI@home: An Experiment in Public-Resource Computing. *Communications of the ACM 45*, 11 (2002), 56–61.

[6] ICQ.com community, people search and messaging service. http://www.icq.com. Acessado em 22 de fevereiro de 2007.

[7] CLARKE, I., SANDBERG, O., WILEY, B., E HONG, T. W. Freenet: A Distributed Anonymous Information Storage and Retrieval System. Em *Workshop on Design*

Issues in Anonymity and Unobservability (Berkeley, CA, EUA, agosto de 2000), pág. 46–66.

[8] TALIA, D., E TRUNFIO, P. Toward a Synergy Between P2P and Grids. *IEEE Internet Computing 7*, 4 (julho de 2002), 95–96.

[9] FOSTER, I., E IAMNITCHI, A. On Death, Taxes, and the Convergence of Peer-to-Peer and Grid Computing. Em *2nd International Workshop on Peer-to-Peer Systems - IPTPS'03* (Berkeley, CA, EUA, fevereiro de 2003), pág. 118–128.

[10] TANG, C., XU, Z., E MAHALINGAM, M. pSearch: information retrieval in structured overlays. *SIGCOMM Comput. Commun. Rev. 33*, 1 (2003), 89–94.

[11] RATNASAMY, S., FRANCIS, P., HANDLEY, M., KARP, R., E SCHENKER, S. A scalable content-addressable network. Em *SIGCOMM '01: Proceedings of the 2001 conference on Applications, technologies, architectures, and protocols for computer communications* (San Diego, CA, EUA, agosto de 2001), pág. 161–172.

[12] MILLER, M. *Discovering P2P*. Sybex, 2001.

[13] Napster Web Site. http://www.napster.com. Acessado em 20 de fevereiro de 2007.

[14] TYSON, J. *How the Old Napster Worked*, 2005. http://computer.howstuffworks.com/napster.htm. Acessado em 20 de fevereiro de 2007.

[15] *The Annotated Gnutella Protocol Specification v0.4*, 2002. http://rfc-gnutella.sourceforge.net/developer/stable/index.html. Acessado em 21 de fevereiro de 2007.

[16] BASET, S. A., E SCHULZRINNE, H. An Analysis of the Skype Peer-to-Peer Internet Telephony Protocol. Em *Workshop on Design Issues in Anonymity and Unobservability* (Berkeley, CA, EUA, agosto de 2000), pág. 46–66.

[17] YANG, B., E GARCIA-MOLINA, H. Comparing Hybrid Peer-to-Peer Systems. Em *Proceedings of the 27th International Conference on Very Large Databases (VLDB)* (Roma, Itália, setembro de 2001), pág. 1–25.

[18] NEWS, B. *Napster Must Stay Shut Down*, março de 2002. http://news.bbc.co.uk/1/hi/entertainment/new_media/1893904.stm. Acessado em 20 de fevereiro de 2007.

[19] RIPEANU, M. Peer-to-Peer Architecture Case Study: Gnutella Network. Em *Proceedings of the First International Conference on Peer-to-Peer Computing - P2P'01* (Linköping, Suécia, agosto de 2001), pág. 99–109.

[20] LIANG, J., KUMAR, R., E ROSS, K. W. Understanding KaZaA. Relatório técnico, Polytechnic University of Brooklyn, 2004.

[21] ANDROUTSELLIS-THEOTOKIS, S., E SPINELLIS, D. A survey of peer-to-peer content distribution technologies. *ACM Comput. Surv. 36*, 4 (2004), 335–371.

[22] MENNECKE, T. eDonkey/Overnet Rapidly Approaching FastTrack, maio de 2004. http://www.slyck.com/news.php?story=477. Acessado em 21 de fevereiro de 2007.

[23] ABERER, K., E DESPOTOVIC, Z. Managing trust in a peer-2-peer information system. Em *CIKM '01: Proceedings of the tenth international conference on Information and knowledge management* (Atlanta, Georgia, EUA, 2001), pág. 310–317.

[24] ADAR, E., E HUBERMAN, B. A. Free Riding on Gnutella. Relatório técnico, Xerox Palo Alto Research Center, setembro de 2000.

[25] CRESPO, A., E GARCIA-MOLINA, H. Semantic Overlay Networks for P2P Systems. Em *Third International Workshop on Agents and Peer-to-Peer Computing - AP2PC 2004* (Nova York, NY, EUA, julho de 2004), pág. 1–13.

[26] HANDURUKANDE, S. B., KERMARREC, A.-M., FESSANT, F. L., E MASSOULIÉ, L. Exploiting semantic clustering in the eDonkey P2P network. Em *Proceedings of the 11th workshop on ACM SIGOPS European workshop: beyond the PC* (Leuven, Bélgica, 2004), pág. 20.

[27] HUGHES, D., WARREN, I., E COULSON, G. AGnuS: The Altruistic Gnutella Server. Em *Proceedings of the 3rd International Conference on Peer-to-Peer Computing - P2P'03* (Linköping, Suécia, setembro de 2003), pág. 202–203.

[28] NAOR, M., E YUNG, M. Universal one-way hash functions and their cryptographic applications. Em *STOC '89: Proceedings of the twenty-first annual ACM symposium on Theory of computing* (Seattle, WA, EUA, 1989), pág. 33–43.

[29] STOICA, I., MORRIS, R., KARGER, D., KAASHOEK, M. F., E BALAKRISHNAN, H. Chord: A Scalable Peer-to-Peer Lookup Service for Internet Applications. Em *SIGCOMM '01: Proceedings of the 2001 conference on Applications, technologies, architectures, and protocols for computer communications* (San Diego, CA, EUA, agosto de 2001), pág. 149–160.

[30] KARGER, D. R., E RUHL, M. Simple efficient load balancing algorithms for peer-to-peer systems. Em *Proceedings of the sixteenth annual ACM symposium on Parallelism in algorithms and architectures* (Barcelona, Espanha, 2004), pág. 36–43.

[31] BERRYY, M. W., DRMA, Z., E JESSUPZ, E. R. Matrices, Vector Spaces, and Information Retrieval. *SIAM REVIEW 41*, 2 (1999), 335–362.

[32] RATNASAMY, S., HELLERSTEIN, J. M., E SHENKER, S. Range queries over dhts. Relatório Técnico IRB-TR-03-009, Intel Research, julho de 2003.

[33] BHARAMBE, A. R., AGRAWAL, M., E SESHAN, S. Mercury: supporting scalable multi-attribute range queries. *SIGCOMM Comput. Commun. Rev. 34*, 4 (2004), 353–366.

[34] RUSITSCHKA, S., E SOUTHALL, A. The Resource Management Framework: A System for Managing Metadata in Decentralized Networks Using Peer-to-Peer Technology. Em *First International Workshop on Agents and Peer-to-Peer Computing - AP2PC 2002* (Bolonha, Itália, julho de 2002), pág. 144–149.

[35] *Project JXTA: A Technology Overview*, 2004. http://www.jxta.org/project/www/docs/jxtaview_01nov02.pdf. Acessado em 20 de fevereiro de 2007.

[36] BANAEI-KASHANI, F., CHEN, C.-C., E SHAHABI, C. WSPDS: WebServices Peer-to-Peer Discovery Service. Em *Proceedings of the International Conference on Internet Computing* (Las Vegas, NE, EUA, junho de 2004), pág. 733–143.

[37] BROEKSTRA, J., EHRIG, M., HAASE, P., VAN HARMELEN, F., KAMPMAN, A., SABOU, M., SIEBES, R., STAAB, S., STUCKENSCHMIDT, H., E TEMPICH, C. A Metadata Model for Semantics-Based Peer-to-Peer Systems. Em *Proceedings of the WWW'03 Workshop on Semantics in Peer-to-Peer and Grid Computing* (Budapeste, Hungria, maio de 2003).

[38] SARTIANI, C. On the Correctness of Query Results in XML P2P Databases. Em *Proceedings of the Fourth International Conference on Peer-to-Peer Computing* (Zurique, Suíça, 2004), pág. 18–25.

[39] RAO, A., LAKSHMINARAYANAN, K., SURANA, S., KARP, R., E STOICA, I. Load Balancing in Structured P2P Systems. Em *2nd International Workshop on Peer-to-Peer Systems - IPTPS'03* (Berkeley, CA, EUA, fevereiro de 2003), pág. 68–79.

[40] GANESAN, P., BAWA, M., E GARCIA-MOLINA, H. Online Balancing of Range-Partitioned Data with Applications to Peer-to-Peer Systems. Em *Proceedings of the 30th VLDB Conference* (Toronto, Canadá, setembro de 2004), pág. 434–445.

[41] SAROIU, S., GUMMADI, P. K., E GRIBBLE, S. D. SA Measurement Study of Peer-to-Peer File Sharing Systems. Em *Proceedings of the Multimedia Computing and Networking (MMCN)* (San Jose, CA, EUA, janeiro de 2002).

[42] BHAGWAN, R., SAVAGE, S., E VOELKER, G. M. Understanding Availability. Em *2nd International Workshop on Peer-to-Peer Systems - IPTPS'03* (Berkeley, CA, EUA, fevereiro de 2003), pág. 256–267.

[43] JavaTM 2 Platform Std. Ed. v1.4.2 Documentation. http://java.sun.com/j2se/1.4.2/docs/api/java/util/Date.html. Acessado em 21 de fevereiro de 2007.

[44] JAGADISH, H. V., KOUDAS, N., E SRIVASTAVA, D. On effective multi-dimensional indexing for strings. Em *SIGMOD '00: Proceedings of the 2000 ACM SIGMOD international conference on Management of data* (Dallas, TX, EUA, maio de 2000), pág. 403–414.

[45] ORAM, A. *Peer-to-Peer: Harnessing the Power of Disruptive Technologies*. O'Reilly, 2001.

[46] DUVAL, E., HODGINS, W., SUTTON, S., E WEIBEL, S. L. Metadata Principles and Practicalities. *D-Lib Magazine 8*, 4 (abril de 2002).

[47] BRAY, T., PAOLI, J., SPERBERG-MCQUEEN, C. M., MALER, E., E YERGEAU, F. Extensible Markup Language (XML) 1.0 (Fourth Edition). *W3C Recommendation* (setembro de 2006). http://www.w3.org/TR/REC-xml/. Acessado em 20 de fevereiro de 2007.

[48] OMNeT ++ Discrete Event Simulation System . http://www.omnetpp.org. Acessado em 20 de fevereiro de 2007.

[49] The Network Simulator Community Portal . http://nsnam.isi.edu/nsnam. Acessado em 20 de fevereiro de 2007.

[50] PeerSim: A Peer-to-Peer Simulator. http://peersim.sourceforge.net/#intro. Acessado em 20 de fevereiro de 2007.