ANDRÉ RIBEIRO DA SILVA

TAXONOMIA PARA SOLUÇÕES DE BALANCEAMENTO DE CARGA EM SISTEMAS BASEADOS EM TABELAS DE *HASH* DISTRIBUÍDAS

Belo Horizonte
31 de julho de 2006

Universidade Federal de Minas Gerais
Instituto de Ciências Exatas
Programa de Pós-Graduação em Ciência da Computação

TAXONOMIA PARA SOLUÇÕES DE BALANCEAMENTO DE CARGA EM SISTEMAS BASEADOS EM TABELAS DE *HASH* DISTRIBUÍDAS

Dissertação apresentada ao Curso de Pós-Graduação em Ciência da Computação da Universidade Federal de Minas Gerais como requisito parcial para a obtenção do grau de Mestre em Ciência da Computação.

ANDRÉ RIBEIRO DA SILVA

Belo Horizonte
31 de julho de 2006

UNIVERSIDADE FEDERAL DE MINAS GERAIS

FOLHA DE APROVAÇÃO

Taxonomia para soluções de balanceamento de carga em sistemas baseados em tabelas de *hash* distribuídas

ANDRÉ RIBEIRO DA SILVA

Dissertação defendida e aprovada pela banca examinadora constituída por:

Ph. D. Dorgival Olavo Guedes Neto – Orientador
Universidade Federal de Minas Gerais

Ph. D. Renato Antônio Celso Ferreira
Universidade Federal de Minas Gerais

Ph. D. Wagner Meira Jr.
Universidade Federal de Minas Gerais

Belo Horizonte, 31 de julho de 2006

Resumo

Sistemas computacionais distribuídos são uma valiosa ferramenta para distribuição e processamento de informação na solução dos mais diversos problemas. Um problema em especial é a distribuição de recursos entre máquinas de uma rede. Aplicações que usam algoritmos embaraçosamente paralelos não precisam se preocupar com essa distribuição. As demais aplicações devem se preocupar com qual nó possui uma determinada informação e qual irá executar determinada tarefa. Dependendo do tipo de aplicação essa funcionalidade deve ser pensada de forma a suportar entrada e saída dos nós da rede mas, ao mesmo tempo, produzindo o mínimo de modificações no sistema quando isso ocorre. A técnica de *hash* consistente possui essa funcionalidade de flexibilidade com pouca modificação do sistema. Essa idéia é utilizada nas redes baseadas em tabelas *hash* distribuídas (DHT). As DHTs por sua vez são a base de diversas redes par-a-par que denominadas estruturadas. As redes DHTs possuem um problema inerente de desbalanceamento de carga devido à forma como são gerados os identificadores dos nós nessas redes. Esses problemas vêm recentemente sendo estudados e diversos algoritmos para lidar com o problema de balanceamento de carga nessas redes foram desenvolvidos recentemente. Com o objetivo de organizar esse conhecimento de forma a permitir a comparação entre os diversos algoritmos, bem como servir de base para o desenvolvimento de novos, neste trabalho será descrita uma taxonomia para os algoritmos de balanceamento de carga em redes baseadas em DHTs. Foram analisados diversos trabalhos da área e todos foram classificados de acordo com a taxonomia criada.

Abstract

Distributed computational systems are a valuable tool for distributing and processing information and are employed in the solution of several problems. A particular problem is the distribution of resources among peers of a network. This is not a concern for applications based on embarrassingly parallel algorithms. Many other applications, however, should be able to determine which node has a specific information or could perform a certain task. Depending on the application to be considered, this functionality should be designed to support network churn with minimum alteration in the system. This is the case of the consistent hash technique which corresponds to the main idea behind networks based on Distributed Hash Tables (DHT) that are the base of several structured peer-to-peer networks. An important issue of DHT networks is its inherent load unbalancing due to the way in which peer identifiers are created. This problem has received much attention recently and, as a consequence, several algorithms have been proposed. In order to organize all this knowledge, compare the algorithms proposed in the literature, and provide a base for the development of new strategies, we propose in this work a taxonomy for load balancing algorithms for DHT Networks. As a result of this study, we analyzed several works and classified them according to the proposed taxonomy.

em memória de José Gonçalves Silveira,
meu amado pai.

Agradecimentos

Chega o momento de conclusão da dissertação. Muita coisa foi feita, muitos artigos foram lidos, muitas páginas escritas e o nome do autor é o meu. Mas ninguém faz nada sozinho. Cada sorriso, cada palavra de apoio, cada conversa, cada interação com outra pessoa muda o trabalho. Está página então é dedicada a todos que interagiram comigo e que em maior ou menor grau contribuíram para a conclusão deste trabalho.

Gostaria de agradecer primeiramente ao meu orientador, o professor Dorgival Guedes, que sempre buscou discutir e dar sua opinião sobre o andamento do trabalho. Sempre atencioso e ao alcance para sanar as dúvidas. Gostaria de agradecer também aos colegas do DCC e da Funcesi, em especial ao Hélio Marcos, figuraça de Belém, por compartilhar diversos momentos de produção e descontração.

Boa parte deste trabalho foi realizado em minha casa. Assim, foi fundamental os inúmeros momentos de descontração vividos em minha república. Agradeço a todas as pessoas que frequentaram o apartamento 301/401 e com quem convivi durante o mestrado. Agradeço à Bárbara pelo seu sorriso contagiante. Agradeço à Greiciane por manter aquela casa em ordem (pelo menos aos sábados). Agradeço ao Eric, pelo compartilhamento do sofrimento de escrever a dissertação, por sua loucura e por matar uma cigarra com o violão. Gostaria de agradecer ao amigo Marco Cristo, pela sua sempre boa companhia, seus filmes malucos e seus ensinamentos. Gostaria também de agradecer aos amigos Daniel Moura e Túlio Guimarães pelos momentos de convivência na república. Gostaria também de agradecer aos amigos Paulo José, Denise, Mário Lobato, André Bigonha, Priscila, Liliana Isabel, Jessé, Raphael Nunes, Skizos, e todos aqueles que tomaram uma cerveja comigo.

Dedico este último parágrafo à todos de minha família. Em especial, gostaria de agradecer à minha irmã, Graziella, por sempre me transmitir tanta alegria e carinho. Agradeço muito mesmo à minha mãe, sem dúvida, o maior agradecimento desta página vai para a Sra Stella. Obrigado por todo apoio, carinho e por suas sábias palavras em todos os momentos da minha vida. Por fim, mas de forma alguma com menor importância, agradeço à minha namorada, a Kellzinha, por toda a sua paciência, por todo o seu carinho e pelo incentivo constante. Muito obrigado.

Sumário

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

Sistemas computacionais distribuídos constituem uma valiosa ferramenta de processamento de informações na solução dos mais diversos problemas. Tais sistemas oferecem a vantagem de melhorar significativamente o desempenho das aplicações por meio da paralelização dos problemas, ou seja, a entrada das aplicações é decomposta em diversos blocos que são tratados paralelamente em diferentes unidades de processamento. Além disso, sistemas distribuídos permitem que vários usuários se beneficiem simultaneamente da alta disponibilidade gerada pela distribuição de recursos, por exemplo, para o armazenamento ou processamento de dados. Entre as várias arquiteturas distribuídas destacam-se hoje os agrupamentos ou agregados (*Clusters*), as grades (*Grids*) e os ambientes par-a-par (*peer-to-peer*).

Tais sistemas podem ser de grande valia, por exemplo, para a implementação de algoritmos de mineração de dados. Com a continua queda de preço dos equipamentos de armazenagem e ao aperfeiçoamento das técnicas de coleta de dados, um volume cada vez maior de dados se torna disponível. Com isso, é cada vez maior a necessidade de extrair informações úteis dessa imensa massa de dados armazenada. A aplicação de técnicas de mineração de dados pode nos ajudar a atingir esse objetivo. Entretanto, devido ao grande volume de dados, o custo da aplicação de tais algoritmos de forma seqüencial torna-se proibitivo. Dessa forma, é importante que as aplicações responsáveis pela mineração possam ser executadas em um ambiente distribuído. Para suprir essa necessidade foi desenvolvido, no laboratório e-Speed do Departamento de Ciência da Computação da UFMG, o ambiente denominado Formigueiro [7] que define um conjunto de abstrações que permitem uma execução eficiente de algoritmos intensivos em dados e processamento (como é o caso da mineração de dados) em um ambiente distribuído heterogêneo.

1.1 O ambiente distribuído Formigueiro

O ambiente Formigueiro trabalha com a idéia *pipeline* de operações, denominada nesse caso de modelo Filtro-Fluxo (*filter stream*). Um *fluxo* é um canal de processamento por onde trafegam dados. Esse canal é dividido em estágios, denominados fluxos, e em cada estágio é realizado um conjunto de operações. Assim, para uma aplicação ser executada no Formigueiro é preciso que ela seja decomposta em diversos estágios de execução, o que já foi feito para para diversas aplicações [7].

Uma vez decomposta em estágios, uma aplicação pode, a cada etapa do *pipeline* (em cada filtro) ter seus dados divididos e processados em paralelo. Isso permite que o ambiente explore as características de paralelismo de dados e de tarefas das aplicações. Além disso, o Formigueiro foi desenvolvido de forma a permitir que estágios independentes do *pipeline* sejam executados sem a necessidade de esperar pelo término de outros, possibilitando assim a exploração das propriedades de assincronismo de diversas aplicações.

Existem diversos problemas característicos desse tipo de abstração que devem ser estudados cuidadosamente para que o sistema como um todo tenha um bom desempenho. Por exemplo, o fato de distribuir as tarefas de uma aplicação entre diversas máquinas leva ao problema natural de realizar esta distribuição de forma eficiente. Isso significa que as máquinas do ambiente distribuído devem receber aproximadamente o mesmo volume de processamento. No caso de uma reconfiguração do sistema durante a execução, idealmente isso deve ocorrer com um mínimo de reassinalamento de tarefas entre os nós do sistema.

Um fator complicador na maioria dos casos é que diversas aplicações exigem que seja possível determinar rapidamente qual nó da rede possui uma determinada informação, por exemplo, para se determinar qual nó deve realizar determinada tarefa. Apenas aplicações que utilizam algoritmos embaraçosamente paralelos não precisam se preocupar com esses problemas; as demais têm que resolver esse problema para garantir a sua execução de forma correta. Se a carga das tarefas for muito variável em relação ao volume de dados envolvidos (aplicações dependentes dos dados) o problema se torna ainda maior, pois mesmo uma distribuição equilibrada do número de tarefas não garantirá um balanceamento de carga adequado. Nesse caso é preciso levar em consideração a quantidade de carga observada em cada máquina e adequar as distribuições de tarefas a essa informação.

A forma mais simples de resolver o problema de distribuição de tarefas é de forma estática do ponto de vista do código da aplicação, ou seja, assinalar as tarefas a nós

em tempo de implementação da aplicação. Essa solução, embora simples de ser aplicada, possui uma limitação muito importante: não é possível flexibilizar a solução para adaptar a mudanças que venham a ocorrer na rede. Uma outra forma de resolver o problema é utilizar um dicionário global que faça o mapeamento. Essa técnica entretanto se torna um problema quando o número de nós da rede aumenta, pois há a necessidade de manter o dicionário atualizado em todas as máquinas da rede. Outra forma é a utilização de uma função de mapeamento fixa, como uma função *hash*, que permite que o assinalamento mude com a mudança do número de nós sem que a aplicação precise ser alterada. Esse é o mecanismo utilizado no ambiente distribuído Formigueiro para implementar a distribuição de tarefas entre máquinas. Essa abstração é denominada *labeled-stream* e consiste em associar a cada mensagem um rótulo que é utilizado por uma função *hash* para indicar qual instância de um filtro (processo) deve receber a mensagem.

A utilização de funções *hash* leva ao problema de ter que adaptar a função utilizada quando o ambiente da aplicação é dinâmico (uma função que gere um assinalamento de inteiros entre zero e sete precisa ser alterada se o número de nós do sistema aumenta para dez, por exemplo). Esses problemas podem ser resolvidos pela adoção da técnica de *hash* consistente, que possui uma grande adaptabilidade a mudanças como entrada e saída de máquinas da rede [15]. Por essa característica, essa técnica é a base de diversas soluções de endereçamento em redes par-a-par estruturadas, as quais se baseiam na distribuição da tabela de *hash* entre os nós da rede (denominadas, por isso, *distributed hash tables*, DHTs). Entretanto, a utilização de *hash* consistente leva a um desbalanceamento de carga inerente ao uso dessa técnica, o que tem um impacto sobre a distribuição de carga em DHTs. Esse fato tem levado ao desenvolvimento de diversos trabalhos em busca de soluções de balanceamento nesse ambiente.

Por ser uma área de pesquisa recente, entender e comparar as diversas tecnologias é uma tarefa difícil, pois não há ainda uma estabilização do conhecimento nessa área. Uma classificação dessas técnicas então torna-se útil para facilitar o entendimento das características próprias dessa área, bem como para permitir um posicionamento correto dos novos trabalhos em uma determinada classe de algoritmos, permitindo assim a organização do conhecimento. Além disso, a partir de uma classificação das técnicas é possível identificar pontos fracos e fortes das mesmas e com isso identificar oportunidades para o desenvolvimento de novas aplicações na área.

Sendo assim, propomo-nos neste trabalho a criar uma taxonomia da área de balanceamento de carga em redes de tabelas *hash* distribuídas (DHT), as quais são fortemente apoiadas na idéia de *hash* consistente e que são a base das redes par-a-par estrutura-

das. Pretende-se com essa taxonomia obter uma base significativa de conhecimento que permita o desenvolvimento de uma solução para o problema de distribuição de tarefas dentro do ambiente Formigueiro. Espera-se que uma solução desenvolvida com base neste trabalho possa apresentar características interessantes para ambientes reconfiguráveis ao mesmo tempo em que mantém a distribuição de carga no sistema balanceada. Além disso, a partir de comparações e analise das descrições espera-se obter novas direções de trabalhos na área das redes DHT.

1.2 Trabalhos relacionados

Devido à natureza do trabalho que nos propomos a desenvolver, podemos separar os trabalhos relacionados em duas partes. A primeira parte inclui os trabalhos relacionados à produção de taxonomias para sistemas distribuídos em geral e é apresentada a seguir. A segunda se relaciona aos artigos especificamente sobre balanceamento de carga em DHTs, que serão analisados com maior detalhes no Capítulo 3.

1.2.1 Construção de taxonomias

O desenvolvimento de taxonomias é extremamente importante para estruturar o conhecimento em novas áreas de pesquisa e auxiliar os pesquisadores a identificar os caminhos passíveis de exploração dentro na nova área. Como exemplo, podemos tomar a taxonomia de Flynn [8], que descreve uma classificação para as arquiteturas de computadores baseada no número de instruções concorrentes e no fluxo de dados disponível, de grande importância para a área de arquitetura de computadores. Já na área de gerenciamento de recursos em sistemas distribuídos foram criadas algumas taxonomias que abrangem a área como um todo [4], e outras com foco mais específico em balanceamento/distribuição de carga [31, 12, 33]. Não existe, até onde pudemos observar, nenhum estudo específico que classifique os algoritmos de balanceamento de dados em redes baseadas em DHTs.

Dessa forma, considerando uma abordagem mais geral, Casavant e Kuhl [4], apresentam uma taxonomia de escalonamento em sistemas concorrentes de propósito geral. O objetivo do trabalho foi prover uma forma de comparação qualitativa entre as pesquisas realizadas na época, bem como fornecer um instrumento de classificação para futuros trabalhos. Nessa taxonomia as várias abordagens para gerenciamento de recursos em sistemas distribuídos foram classificadas seguindo uma classificação hierárquica

e outra não hierárquica. Uma terminologia adequada também foi proposta, bem como uma série de exemplos de aplicação da taxonomia.

Wang e Morris [33], por outro lado, focam seu trabalho de classificação mais especificamente na área de balanceamento/distribuição de carga, propondo uma taxonomia de duas dimensões. Em um eixo os algoritmos são classificados quanto à iniciativa de busca por carga, que pode ser feita pelo nó de rede gerador da carga ou pelo nó de rede executor da carga. No outro eixo os algoritmos são classificados quanto ao seu nível de dependência de informações do sistema, propondo, no caso, sete níveis de dependência. Além disso é proposto no mesmo trabalho uma métrica que possibilita realizar a comparação entre os diversos algoritmos estudados.

Os trabalhos de Shivaratri *et al.* [31] e Gupta e Bepari [12] fazem classificações parecidas da área de distribuição de carga em sistemas distribuídos. Segundo esses trabalhos, os algoritmos de balanceamento/distribuição de carga podem ser classificados de acordo com as políticas adotadas para transferência de carga, localização de destino, seleção de processos (carga) e de informação (dependência de informação). Além disso, as técnicas de balanceamento são classificadas seguindo a hierarquia proposta por Casavant e Kuhl [4].

1.2.2 Balanceamento de carga em DHTs

Vários algoritmos de balanceamento de carga em DHTs foram propostos nos últimos anos. Diversos algoritmos utilizam a técnica de servidores virtuais para manter a carga da rede balanceada [11, 27, 14]. Outros algoritmos não se utilizam dessas técnicas e propõem uma redistribuição do espaço de identificadores [9, 20, 2]. Além disso foram propostos algoritmos para distribuição de carga uniforme e não uniforme, e para redes com distribuição de capacidades homogêneas e não homogêneas [16, 23, 20, 2]. Foram também propostos algoritmos para lidar com balanceamento de itens de dados [20] bem como algoritmos que levam em consideração informações de proximidade entre os nós da rede [36]. Embora mencionados resumidamente aqui, as principais características dos trabalhos da área de balanceamento de carga em DHTs serão discutidas no Capítulo 3.

1.3 Organização da dissertação

Este trabalho foi dividido em cinco capítulos. Neste capítulo foi realizada uma introdução ao trabalho descrevendo os seus conceitos básicos e objetivos. No Capítulo 2 são

apresentadas as tecnologias de balanceamento de carga, de *hash* consistente, das tabelas *hash* distribuídas, bem como demais conceitos necessários para o entendimento da taxonomia criada. No Capítulo 3 são apresentados as descrições dos trabalhos utilizados na criação da taxonomia. Em seguida, no Capítulo 4, é então mostrada e discutida a taxonomia criada e feita a classificação dos trabalhos da área segundo essa taxonomia. No Capítulo 5 são apresentadas as conclusões do trabalho e possíveis trabalhos futuros. Por fim, no Apêndice A é possível encontrar uma compilação da classificação dos trabalhos obtida junto à taxonomia.

Capítulo 2

Fundamentação teórica

Considerando-se o foco deste trabalho, é importante descrever primeiramente os principais conceitos relacionados que são, nesse caso, o uso de *hash* consistente, os sistemas par-a-par baseados em DHT e os elementos gerais do problema de balanceamento de carga.

2.1 Balanceamento de carga

Balanceamento de carga se refere ao conjunto de ações tomadas para que os recursos de um sistema não sejam mal distribuídos, ou seja, para que todos os membros de um sistema de computadores tenham igual responsabilidade, ou mesmo, num ambiente heterogêneo, responsabilidade proporcional à capacidade de seus recursos. A área de balanceamento de carga já mostra certa maturidade graças aos avanços das últimas duas décadas no desenvolvimento de técnicas para uma melhor distribuição dos recursos em sistemas distribuídos, mas o problema de balanceamento em DHTs ainda é relativamente novo.

A carga de um sistema pode ser medida de várias maneiras em relação a diferentes recursos. Pode-se levar em consideração diversos tipos de recursos como armazenamento, banda de rede, CPU, etc. Essas três principais formas de balanceamento serão brevemente abordadas a seguir:

Balanceamento de armazenamento: nesse caso a carga considerada é a quantidade de espaço em disco que uma máquina da rede dispõe para armazenamento de dados. Essa carga pode ser medida em unidades múltiplas de *bytes* ou em número de itens armazenados.

Balanceamento de banda de rede: nesse caso a carga considerada é a porcentagem de banda de rede utilizada. Balanceamento entre vizinhos e entre nós próximos fisicamente normalmente reduzem a quantidade de utilização da banda de rede por realizar transferências entre poucos nós. Um sistema balanceado nesse caso seria um sistema no qual as conexões de rede fossem bem utilizadas de forma a se evitar gargalos de comunicação.

Balanceamento de processamento (CPU): a carga considerada é a utilização da CPU das máquinas da rede. Essa carga pode ser medida pela porcentagem de CPU utilizada ou pelo número de requisições atendidas por uma máquina. Nesse caso é preciso considerar o tempo de execução das tarefas durante a distribuição.

Em relação a essa última forma de balanceamento, CPU, é importante observar que o assinalamento de uma tarefa a uma máquina pode acontecer no momento da criação da tarefa, ou seja, antes do início da sua execução, ou por meio da transferência de uma máquina para outra, interrompendo a execução na máquina de origem e continuando a execução na máquina de destino. O primeiro caso é mais fácil de ser implementado, pois é necessário apenas decidir qual a máquina de destino da tarefa e enviar a tarefa para a máquina. No segundo caso, a implementação é mais complicada devido à necessidade de interromper uma tarefa que já está sendo executada. Nesse caso, o estado da tarefa deve também ser transferido para a máquina de destino.

Um outro problema é a associação entre a tarefa e os dados que ela processa. Nesse tipo de balanceamento é preciso definir como será realizada a transferência de dados. Essa característica é importante, pois ao transferir dados lida-se com as demais formas de balanceamento, uma vez que a transferência e o armazenamento dos dados a serem processados estão diretamente relacionados com o uso da banda de rede e com o espaço disponível para armazenamento nas máquinas.

Além disso, independentemente dos recursos escolhidos para realização do balanceamento é importante definir o estado de balanceamento de carga de uma rede, ou seja, sob quais aspectos o sistema é considerado sobrecarregado ou em equilíbrio em relação a um determinado recurso. Se a métrica utilizada, por exemplo, for o tempo que um processador leva para realizar uma tarefa, seria necessário distribuir as tarefas entre os processadores de forma que cada um fique em funcionamento durante um mesmo período de tempo para que o sistema atinja um equilíbrio de carga.

Por outro lado, se a carga de uma determinada máquina está relacionada a um recurso como armazenamento de dados, por exemplo, a quantidade de arquivos que ela armazena em seu disco deve ser aproximadamente igual à quantidade de todas as

máquinas da rede, proporcionalmente à sua capacidade de armazenamento. Assim, em um sistema de carga equilibrada todas as máquinas armazenariam aproximadamente o mesmo número de arquivos ou a mesma quantidade de *bytes*. Neste texto, vamos indistintamente utilizar os termos distribuição de tarefas e distribuição de recursos, quando necessário deixaremos explícito o tipo de carga tratada.

2.1.1 Balanceamento de carga em sistemas distribuídos

De acordo com Shirazi *et alii* [30], a disponibilidade de diversas máquinas para utilização traz o natural problema de como dividir tarefas entre essas máquinas. Não só a distribuição de tarefas, mas a busca por atingir o uso justo dos recursos de acordo com as potencialidades de cada máquina constitui a meta da área de balanceamento de carga. Assim, algoritmos são desenvolvidos para que a utilização dos recursos (CPU, armazenamento, banda de rede, etc) seja maximizada e os tempos de respostas, atrasos de comunicação, etc., sejam minimizados. As técnicas de balanceamento de carga são usualmente identificadas como estáticas ou dinâmicas em função da sua forma de funcionamento.

Balanceamento estático

Algoritmos estáticos para balanceamento consideram que as tarefas são distribuídas entre os processadores do sistema de forma estática, ou seja, antes do início da execução das aplicações. O objetivo principal dessa técnica é minimizar o tempo de execução das aplicações. Para fazer isso, informações sobre as tarefas são estimadas e os processadores-alvo são então associados às tarefas em tempo de compilação da aplicação.

Embora esse mecanismo elimine a sobrecarga de escolher o destino de uma tarefa em tempo de execução, segundo Papadimitriou e Yannakakis [21] obter uma solução ótima para o mapeamento de tarefas é um problema NP-completo fazendo com que as soluções normalmente adotadas sejam baseadas em heurísticas. Além disso, mudanças nos tempos de execução de uma tarefa não são previstas por essas técnicas, o que pode levar a uma distribuição de tarefas ineficiente, gerando possivelmente sobrecarga em algumas máquinas do sistema.

Balanceamento dinâmico

As técnicas de balanceamento dinâmicas, ao contrário das estáticas, buscam adequar as tarefas às máquinas durante a execução das aplicações. Nesse caso, as máquinas são monitoradas e guardam informações sobre a taxa de utilização dos recursos consumidos. A partir dessas informações é possível transferir tarefas (carga) de máquinas sobrecarregadas para máquinas que estejam consumindo poucos recursos. Para as técnicas dinâmicas são normalmente definidos três tipos de políticas para auxiliar a tomada de decisão:

política de informação: define como e onde obter informações sobre a carga das máquinas do sistema;

política de transferência: define em que condições determinada tarefa deve ser transferida de uma máquina para outra;

política de localização: define para qual máquina/processador a tarefa escolhida deve ser transferida.

Essas políticas podem ser combinadas de diversas formas e em diversos graus, de forma centralizada ou distribuída. No caso do balanceamento de carga centralizado, uma máquina da rede fica responsável por recolher as informações de carga das outras máquinas de acordo com alguma política de informação. De posse dessas informações são tomadas então decisões de localização e transferência de carga de acordo com as políticas adotadas. O problema dessa forma de balanceamento é que, por existir apenas um responsável por esse serviço, o sistema terá dificuldades de crescimento e também ficará mais sujeito a possíveis falhas do serviço de balanceamento.

Já no caso do balanceamento distribuído, todas as máquinas da rede são responsáveis pelas tomadas de decisões que manterão o sistema balanceado. Nesse caso não há um único ponto de falha, mas há uma sobrecarga de mensagens transferidas entre as máquinas para realização do balanceamento. Se a política de informação não for bem implantada esse problema pode ser ainda maior pois o grande número de mensagens poderia impedir uma boa escalabilidade do sistema. Assim, é preciso definir, de acordo com as necessidades do ambiente distribuído, qual a melhor estratégia para gerenciar o balanceamento de carga do sistema. É possível também combinar as duas estratégias fazendo com que poucas máquinas realizem o processo de balanceamento de carga do sistema. Nesse caso cada máquina seria responsável por uma porção da rede e poderia utilizar políticas diferentes para obtenção de informação, para localização e para a transferência de tarefas entre máquinas.

Essa flexibilidade de combinações e de tomada de decisão durante a execução das aplicações é a principal vantagem das técnicas de balanceamento dinâmicas em relação às estáticas. A principal desvantagem dessas técnicas é a sobrecarga de mensagens transferidas e de processamento gasta com a gerência dessas políticas e com a tomada de decisão.

2.2 Hash consistente

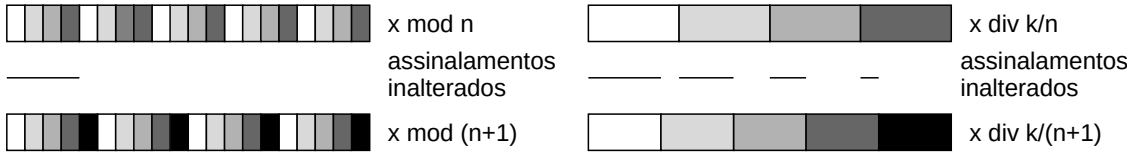
Antes de descrevermos as redes do tipo DHT, objeto último deste trabalho, precisamos entender os conceitos que servem de base para essas redes, ou seja, os conceitos de *hash* em geral e de *hash* consistente em particular.

De acordo com Ziviani [37], uma função *hash* tradicional é uma função de transformação de chaves que consiste em mapear uma dada chave i, que identifique um determinado objeto (dados, máquina, etc.), em uma outra chave i' dentro de uma faixa de números inteiros. Essa nova chave é, então, utilizada para localizar um recurso ou mesmo a máquina que suporta tal recurso, facilitando a consulta pelo objeto original, já que, para encontrar esse recurso basta aplicar a função de transformação e obter a sua localização. Uma função *hash* bastante utilizada, por exemplo, é a função de congruência linear $x \mapsto (ax+b)mod(p)$. Há, entretanto, o problema de reconfiguração: uma vez que as chaves são distribuídas na faixa da função, qualquer mudança no tamanho dessa faixa (no exemplo anterior, p) implica na necessidade de redistribuir todas as chaves para evitar inconsistências [15]. Essa operação é cara, o que torna difícil o uso desse tipo de função *hash* em sistemas dinâmicos.

Em certos ambientes distribuídos existe o que chamamos de restrições de afinidades, ou seja, certas tarefas e dados só podem ser atribuídos a determinada máquina. Isso acontece quando, por exemplo, é preciso manter algum tipo de estado entre a execução das tarefas. Assim, uma forma eficiente de definir qual máquina ficará responsável por determinada tarefa é utilizar rótulos nos dados e usar esses rótulos como valores para funções *hash* tradicionais.

Entretanto, quando esse tipo de sistema precisa sofrer reconfigurações, por exemplo, a entrada de um novo nó no sistema, a utilização de uma função tradicional seria um grande problema, pois uma grande quantidade de redistribuição de tarefas deveria ser realizada. Nesse caso o uso de uma função de *hash* consistente é mais aconselhável. Vamos por enquanto nos ater ao funcionamento das funções de *hash* tradicional.

Considere, como exemplo de funções *hash* tradicionais as funções da forma $h(x) =$



(a) Hash usando `mod` (b) Hash usando `div`

Figura 2.1: Distribuição de identificadores com duas funções *hash*.

x `mod` n e $h(x) = x$ `div` k/n onde x é o rótulo, k é o número de chaves (objetos) e n é o número de nós participantes na rede. Essas funções têm como vantagens a distribuição uniforme das chaves entre as máquinas e a sua simplicidade de implementação. Porém, para um sistema dinâmico, na inclusão ou retirada de um nó, um grande número de chaves precisaria ser redistribuída em um sistema que utilizasse essas funções. Por exemplo, para o caso da função `mod` veja a Figura 2.1 (a); nela vemos a distribuição de 20 chaves entre 4 máquinas e logo depois a redistribuição das chaves que é necessária devido à entrada de mais uma máquina na rede. Após a entrada da nova máquina, apenas 4 chaves permanecem nas máquinas originais, sendo que todas as outras são distribuídas para máquinas diferentes. Já a Figura 2.1 (b) mostra como ficaria um sistema utilizando a função `div`; vemos na figura que, embora muitas chaves ainda sejam redistribuídas, o comportamento da redistribuição é um pouco diferente. Com a utilização dessa função as chaves são trocadas apenas entre nós vizinhos, mudando assim o padrão de comunicação na troca, porém mantendo a ordem de complexidade do volume de dados migrados, que é $O(n)$ nos dois casos.

Uma função *hash* consistente tem a propriedade de mudar minimamente à medida que sua faixa é alterada [15]. Isso significa que, ao utilizar uma função *hash* para mapear os nós de uma rede ou os recursos de uma aplicação distribuída, a inclusão ou remoção de nós do sistema é feita de forma que não é necessário gerar novas chaves para todos os objetos, forçando sua redistribuição. Além disso, essas funções são construídas de forma a manter aproximadamente o mesmo número de chaves em cada máquina participante do sistema. Essa propriedade facilita o gerenciamento dos recursos de forma a garantir um melhor balanceamento da carga de processamento/armazenamento.

Consideremos, por exemplo, um sistema de *caching* distribuído que utiliza 23 máquinas como replicadoras de conteúdo Web. Podemos utilizar uma função *hash* que mapeie uma URL u em uma das máquinas (nomeadas de 0 a 22) com a seguinte função $h(u) = (7u+4)\ mod\ 23$. É possível mapear agora as URLs nos servidores de cache uniformemente, dividindo a carga total do sistema entre as 23 máquinas. Entretanto, se uma outra máquina for adicionada ao sistema, teríamos que modificar a função de *hash*

para, por exemplo, $h(u) = (7u + 4)\ mod\ 24$. O problema é que agora todas as URLs já mapeadas devem ser recalculadas com a nova função e mapeadas em outras máquinas. Uma função *hash* consistente poderia resolver o problema da seguinte forma:

- Escolha uma função hash padrão que realize o mapeamento de cadeia de caracteres em uma determinada faixa de números $[0..M-1]$.
- Dividimos o valor gerado por M de forma que o mapeamento seja feito no intervalo $[0, 1)$ que pode ser visto como o intervalo de um círculo unitário.
- Cada URL é então mapeada dessa forma para um ponto no círculo unitário. Da mesma forma, cada servidor de *cache* no sistema é também mapeado para um ponto no círculo unitário.
- Agora, associe cada URL a uma máquina, da seguinte maneira: movendo-se no círculo unitário, em um sentido pré-estabelecido a partir do ponto referente a uma URL, identifica-se a primeira máquina encontrada. Essa máquina será responsável por armazenar o documento associado àquela URL.

Assim, quando existe alguma alteração no número de *caches*, somente os objetos próximos à *cache* incluída/excluída no espaço de endereçamento devem ser movidos, mantendo-se a consistência dos sistema. A Figura 2.2 ilustra o exemplo apresentado. Os servidores representam as *caches* e os ítens são as páginas Web apontadas pelas URLs. Como pode ser visto na figura, a solução de *hash* consistente se configura assim como uma forma eficiente de se manter um mapeamento entre objetos e servidores, com um custo de reconfiguração reduzido.

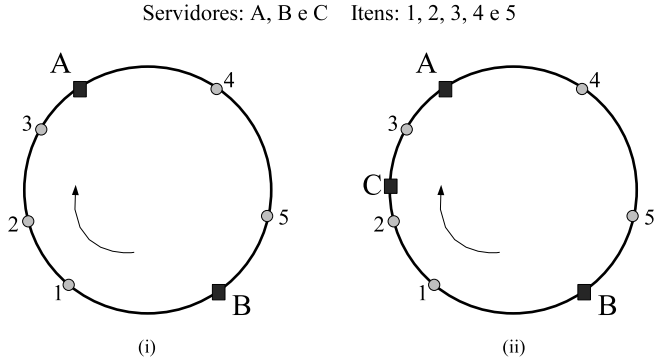


Figura 2.2: (i) Cada URL mapeada é associada com a *cache* mais próxima no sentido horário do círculo: itens 1, 2 e 3 são associados à *cache* A e os ítens 4 e 5 são associados à *cache* B. (ii) Quando uma nova *cache* C é adicionada ao sistema, apenas os ítens 1 e 2 se moverão para a nova *cache* C; nenhuma outra mudança é necessária.

2.3 Redes par-a-par

As redes par-a-par (*peer-to-peer* ou P2P) vem nos últimos anos se tornando uma tecnologia cada vez mais utilizada na estrutura de aplicações de redes populares. A popularização da tecnologia P2P se iniciou com o surgimento do Napster, aplicação que permitia compartilhamento de arquivos entre usuários da Internet. Embora fosse um sistema híbrido, que exigia a centralização de um servidor para consulta ao índice de arquivos, a transferência de informações era feita de forma independente do servidor, entre os participantes. Surgia aí a primeira de muitas aplicações distribuídas que utilizariam a idéia de comunicação par-a-par. Podemos ver na Figura 2.3 como as redes par-a-par se encaixam nos sistemas computacionais dentro da área de sistemas distribuídos segundo a taxonomia de sistemas computacionais de Milojicic *et alii* [19].

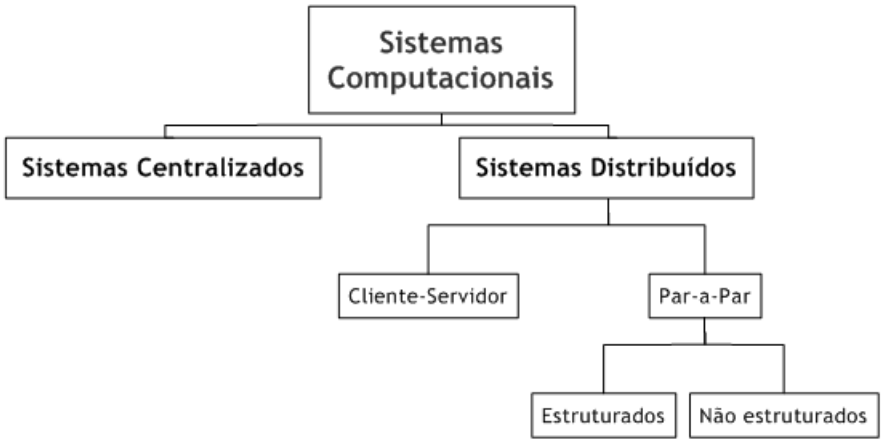


Figura 2.3: Uma taxonomia para sistemas computacionais.

Alguns trabalhos afirmam que a tecnologia P2P não traz muita coisa de novo, pois P2P não passaria de um sistema distribuído mais descentralizado em relação aos sistemas cliente-servidor [19]. De qualquer forma, vamos considerar neste trabalho que o termo *par-a-par* se refere a uma classe de sistemas onde dois ou mais pares participantes de uma rede possam trocar informações e realizar colaborações entre eles ou com demais pares, sem a necessidade de um coordenador central, como em Schoder e Fischbach [28]. Essa definição inclui todos os sistemas e as aplicações consideradas P2P pela comunidade da área.

2.3.1 Formas de organização das redes P2P

As redes P2P podem ser classificadas de acordo com as sua organização interna, ditada pelos seus algoritmos de roteamento e busca, como não-estruturadas e estruturadas. Uma rede P2P não-estruturada, também chamada de rede P2P pura, é uma rede na qual não existe estrutura pré-definida de roteamento entre os pares. Cada par, para realizar uma busca por um documento, envia essa busca para seus vizinhos por meio de inundamento (*flooding*) e os vizinhos por sua vez repassam a busca para seus vizinhos e assim sucessivamente, até que o resultado da busca seja atingido, ou seja, o documento seja encontrado, ou o número de passos de busca atinja o limite do sistema. Esses sistemas enfrentam, claramente, problemas de escalabilidade devido ao processo de *flooding*. Em sistemas pequenos, como uma rede universitária ou uma rede empresarial, esses sistemas são vantajosos devido ao baixo custo de implantação e manutenção. A principal rede P2P que representa essa categoria é a rede Gnutella [1] utilizada para compartilhamento de arquivos na Internet. Para resolver o problema de escala alguns trabalhos utilizam a idéia de super-pares, que serviriam como pontos de concentração de recursos/tarefas [35].

Já nas redes P2P estruturadas, também chamadas de orientadas a conteúdo, cada par possui um identificador, que pode ou não ser escolhido aleatoriamente, em um espaço de endereçamento. Cada objeto armazenado na rede possui também um identificador obtido do mesmo espaço de endereçamento. Os identificadores dos objetos e dos pares são então associados segundo alguma noção de proximidade. Além disso, cada nó da rede[2] possui uma tabela de roteamento com o endereço de outros nós da rede estrategicamente escolhidos de forma a tornar o processo de busca eficiente. As estruturas de endereçamento, roteamento e busca dessas redes constituem as chamadas Tabelas *Hash* Distribuídas (Distributed Hash Tables - DHT) que serão discutidas na próxima seção.

2.3.2 Tabelas *hash* distribuídas

Tabelas *hash* tradicionais são utilizadas em diversas aplicações para realização de mapeamento entre objetos. No caso de sistemas distribuídos, podem ser utilizadas, por exemplo, para mapear identificadores de objetos (tarefas, arquivos, etc.) em máquinas da rede. Nesse caso é utilizada uma tabela que associa a todo objeto um identificador que indica qual máquina é responsável pelo objeto. Essa tabela é então armazenada

[1] http://www.gnutella.com

[2] os termos *nó* ou *par* possuem o mesmo significado: participantes de uma rede.

completamente dentro de cada máquina. Assim a partir de qualquer máquina é possível saber em tempo constante qualquer mapeamento entre objetos e máquinas. O problema dessa abordagem é que quando o número de nós da rede cresce o custo de armazenar essa tabela e mantê-la atualizada em todas as máquinas se torna proibitivo.

Uma rede baseada em tabelas *hash* distribuídas resolve esse problema dividindo a faixa de valores possíveis para essa tabela em pedaços e distribuindo esses pedaços para os diversos nós da rede. Assim o custo de armazenamento é reduzido, uma vez que cada máquina armazena apenas uma parte da tabela e o custo de manutenção também é reduzido, já que não é preciso mais atualizar a tabela em todas as máquinas quando alguma modificação no número de máquinas acontece. Com essas modificações, a partir de uma máquina não é mais possível obter a resposta para um mapeamento específico já que a tabela não está mais totalmente armazenada num único local. É preciso, portanto, de uma estrutura de dados distribuída entre os nós da rede e uma forma de rotear mensagens entre esses nós, de forma que, a partir de qualquer nó, seja possível encontrar a fração da tabela para onde um objeto é mapeado.

Nessas redes, a associação de identificadores de objetos aos identificadores dos nós da rede é similar à idéia de *hash* consistente, ou seja, a associação ocorre de forma que cada objeto seja associado a um nó de acordo com alguma noção de proximidade. Assim, o algoritmo de roteamento deve enviar mensagens para o nó da rede mais próximo, entre os identificadores que ele possui, do identificador procurado. O processo se repete então até que a busca chegue ao fim, ou seja, atinja o nó da rede que é responsável pelo identificador procurado.

Essas redes funcionam como uma interface entre a infraestrutura de redes e as aplicações e disponibilizam uma primitiva de associação ou armazenamento *put(dados, identificador)* e uma primitiva de busca *get(identificador)* que retorna o objeto que tem o identificador informado, a Figura 2.4 mostra um esquema do relacionamento das redes DHT com as aplicações e os nós de uma rede.

As diversas redes P2P estruturadas se diferenciam principalmente em relação ao tipo de espaço de endereçamento utilizado e à forma de roteamento de mensagens entre os nós. A seguir apresentamos alguns exemplos de redes par-a-par baseadas em DHTs.

2.3.2.1 Chord

A rede Chord [32] é construída como uma DHT de espaço de endereçamento circular de tamanho 2^{160} posições. Todo objeto armazenado nessa rede possui um identificador

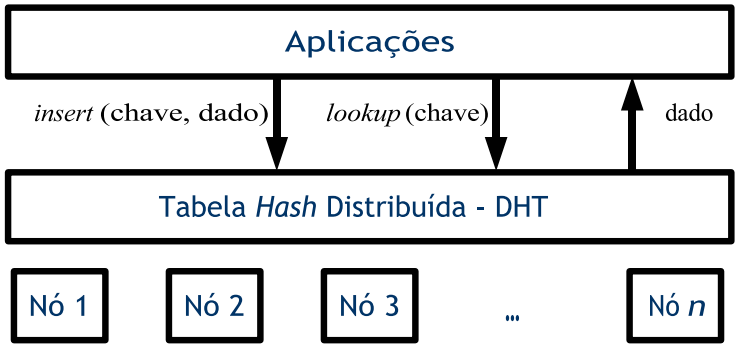


Figura 2.4: Representação das DHTs como interface entre aplicações e infraestrutura de rede.

i obtido do espaço de endereçamento. Da mesma forma, todo nó da rede possui um identificador j obtido do mesmo espaço de endereçamento. Os identificadores são gerados por uma função criptográfica SHA-1, utilizando como entrada algum valor que identifique o objeto, como nome do arquivo, uma URL, etc. Para os nós são gerados identificadores com a mesma função criptográfica utilizando-se, por questões de segurança, o endereço IP do nó. Cada objeto é então associado ao nó da rede que possui o menor identificador maior ou igual ao dele, ou seja, cada objeto é armazenado em um nó que possua o menor identificador tal que $j \geq i$, que se torna responsável pelo objeto. Esse nó é chamado nó sucessor de i (Figura 2.5).

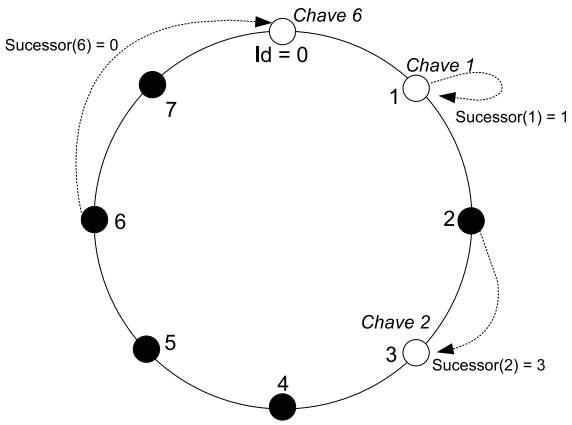


Figura 2.5: Exemplo da estrutura de anel da rede Chord. No exemplo, a rede é composta de três nós: 0, 1 e 3. Assim a chave de identificador 1 é armazenada no nó 1, a de identificador 2 fica armazenada no nó 3 e as chaves 4, 5, 6 e 7 ficam armazenadas no nó 0.

O sistema de busca é realizado por meio da *finger table* que é uma tabela de roteamento onde cada entrada aponta para outros nós da rede por meio de identificadores

especialmente calculados. Assim, dado um determinado nó, a *i-ésima* entrada da tabela desse nó é calculada para ser o primeiro nó, s, que o sucede por pelo menos 2^{i-1} pontos do espaço de endereçamento. Essa forma de calcular as entradas da tabela de roteamento permite que a cada passo de uma busca o tamanho do espaço de endereçamento seja reduzido pela metade de forma que o tempo total de busca seja $O(log(n))$, sendo n o número de pares ativos na rede.

Dessa forma, para realizar uma busca pelo identificador i, um nó k procura em sua *finger table* a entrada que possua o maior identificador menor que i. A essa entrada escolhida está associado o endereço IP da máquina mais próxima de i que o nó conhece. Esse processo se repete em cada nó que a mensagem de busca chegar até que a mensagem chegue ao seu destino, ou seja, ao nó que é responsável pelo identificador i. A Figura 2.6 ilustra o funcionamento da busca por um objeto de identificador 11.

Esse trabalho foi o primeiro a relatar o problema de desbalanceamento da distribuição de identificadores em uma rede DHT, propondo como solução o uso de servidores virtuais para lidar com esse desbalanceamento. Esse problema e a técnica de servidores virtuais, que é muito utilizada nos algoritmos de balanceamento de carga em DHTs, serão descritos com mais detalhes na seção 2.4.

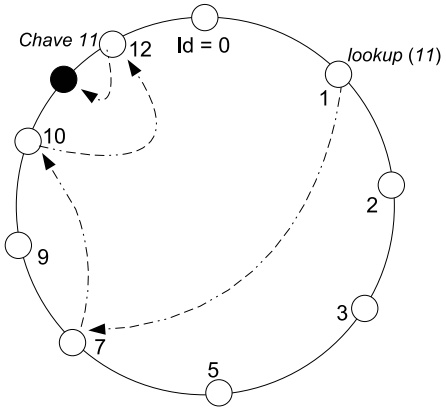
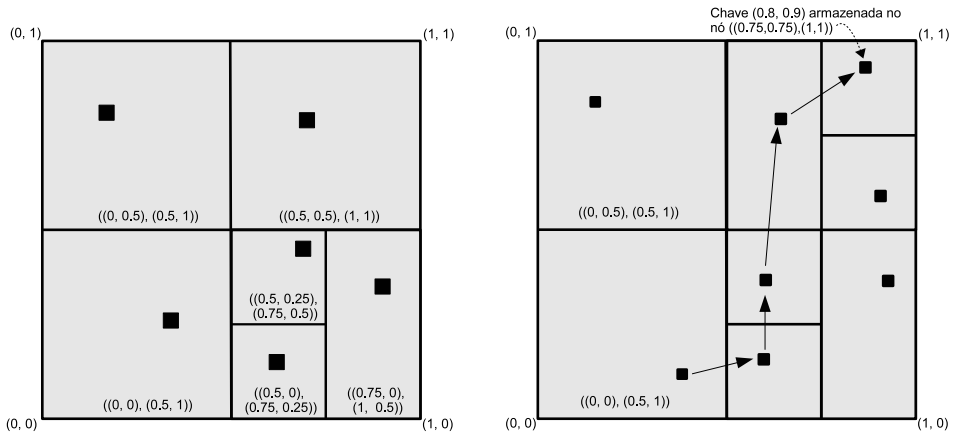


Figura 2.6: Exemplo do funcionamento da busca em uma rede Chord. No exemplo, a rede é composta de 9 nós e uma chave de identificador 11 que fica armazenada no nó de identificador 12. A busca é realiza em diversos passos. A cada passo chega-se mais próximo ao nó responsável pela chave pesquisada.

2.3.2.2 CAN

A rede CAN (*Content Addressable Network*), descrita por Ratnasamy *et alii* [24], trabalha com um espaço de endereçamento multidimensional. Cada nó é responsável por uma determinada região do espaço de endereçamento chamada zona e é identificado



(a) Estrutura da rede CAN com 6 nós em um espaço de endereçamento de duas dimensões.

(b) Pesquisa por chave na rede CAN. A mensagem é passada para o vizinho mais próximo do destino em linha reta.

Figura 2.7: Exemplo da estrutura e do funcionamento do mecanismo de busca em uma rede CAN.

pelas coordenadas das fronteiras dessa zona. Assim, um objeto a ser armazenado na rede é mapeado para um ponto P em um espaço d-dimensional e esse valor é então armazenado no nó da rede que está localizado na zona onde está o ponto P.

Para a inserção de um novo nó é necessário conhecer algum nó já presente na rede, a partir desse nó é possível encontrar o nó responsável pelo identificador do novo nó, ou seja, o nó responsável pelo ponto atribuído ao novo nó. Ao encontrar tal responsável, o espaço cartesiano é então dividido e os nós da vizinhança atualizam suas tabelas de roteamento. Por exemplo, na Figura 2.7 (a) temos uma rede CAN bidimensional em um espaço de dimensões [0,1]x[0,1] e seis nós inseridos nessa rede.

Para realizar o roteamento de mensagens, cada nó presente na rede possui informações sobre um conjunto de zonas adjacentes à sua. O roteamento é realizado com base na distância entre os pontos de origem e de destino da consulta. Assim, ao chegar uma mensagem a um nó ela é encaminhada para o nó vizinho mais próximo à região de destino. Esse processo se repete até que a região a qual pertence o objeto pesquisado é encontrada juntamente com o nó responsável pelo objeto. A noção de proximidade aqui se refere ao menor caminho entre dois pontos, ou seja, o nó escolhido é aquele que está mais próximo da reta que liga o ponto de origem ao de destino. A Figura 2.7 (b) ilustra o processo de busca por uma chave.

2.3.2.3 Tapestry

A rede Tapestry [13] é baseada na tecnologia de distribuição de objetos de Plaxton [22] que consiste em utilizar uma estrutura de dados distribuída que permite a localização de objetos e roteamento de mensagens dentro de uma rede *overlay*.

Nessa rede cada nó possui uma tabela chamada *neighbor map* que contém b entradas, onde b é a base numérica do identificador, que são contatos para os vizinhos do nó em diversos níveis. Cada nível l possui contatos para os nós que têm l dígitos do identificador conhecidos. Assim, cada entrada do mapa de vizinhos é um contato para o nó mais próximo cujo identificador casa com o número da entrada. A Tabela 2.1 ilustra uma possível tabela de vizinhos para um nó de identificador 67493. Por exemplo, a 3^a entrada para o 4^o nível aponta para o nó mais próximo que termina com ..3493.

	Nível 5	Nível 4	Nível 3	Nível 2	Nível 1
Entrada 0	07493	x0493	xx093	xxx03	xxxx0
Entrada 1	17493	x1493	xx193	xxx13	xxxx1
Entrada 2	27493	x2493	xx293	xxx23	xxxx2
Entrada 3	37493	x3493	xx393	xxx33	xxxx3
Entrada 4	47493	x4493	xx493	xxx43	xxxx4
Entrada 5	57493	x5493	xx593	xxx53	xxxx5
Entrada 6	67493	x6493	xx693	xxx63	xxxx6
Entrada 7	77493	x7493	xx793	xxx73	xxxx7
Entrada 8	87493	x8493	xx893	xxx83	xxxx8
Entrada 9	97493	x9493	xx993	xxx93	xxxx9

Tabela 2.1: Tabela de mapeamento para o nó de identificador 67493 na rede Tapestry.

A partir dessa tabela são escolhidos para cada busca o próximo nó que possui identificador de maior sufixo em comum com o objeto de destino e a mensagem é encaminhada para ele. Assim, em cada salto um novo dígito do identificador de destino é resolvido. Esse roteamento é realizado passo a passo percorrendo cada dígito do identificador da direita para a esquerda. Por exemplo, a partir do nó de identificador 67493 é possível chegar ao nó de identificador 45875 resolvendo o endereço dígito por dígito: $xxxx5 \rightarrow xxx75 \rightarrow xx875 \rightarrow x5875 \rightarrow 45875$.

A rede Tapestry é portanto uma adaptação das estruturas de roteamento e armazenamento da rede Plaxton para uma rede com as características das redes par-a-par. Foram realizadas modificações nos esquemas de atualização das tabelas para suportar entradas e saídas na rede, otimizações para melhorar o desempenho e a tolerância a falhas da rede.

Essas redes utilizam o que chamamos *nó raíz*, que é um nó responsável por guardar

a localização de um determinado objeto/recurso. Sempre que um objeto x é inserido na rede e armazenado em um nó n_i uma mensagem contendo a tupla <objeto x, nó responsável i> é enviada para o nó raíz n_r. Uma cópia da tupla é guardada em todo o caminho percorrido pela mensagem até o nó raiz para melhorar o sistema de busca por objetos e oferecer uma melhor tolerância à falhas na rede.

Outras redes DHT que utilizam um mecanismo de busca por sufixos são as redes Pastry [26], que é também baseada na rede Plaxton e é muito similar à rede Tapestry, e a rede Kademlia [18] que possui uma topologia baseada no cálculo da distância dos nós por meio de uma função XOR, além de um maior foco em desempenho com o uso de concorrência durante as atualizações e pesquisas por objetos. Uma comparação entre as redes Chord, CAN, Tapestry é apresentada na tabela 2.2.

Sistema P2P	Modelo	Parâmetros	Busca	Estado
Chord	unidimensional, endereçamento circular	N - número de pares na rede	$log(N)$	$(log(N))^2$
CAN	endereçamento multidimensional	N - número de pares na rede d-número de dimensões	$d \cdot N^{1/d}$	$2 \cdot d$
Tapestry	malha global Plaxton	N - número de pares na rede, b - base do identificador	$log_b(N)$	$log_b(N)$

Tabela 2.2: Comparação entre as redes DHTs.

2.4 Balanceamento de carga em DHTs

Em uma rede baseada em tabelas *hash* distribuídas (DHTs) os nós da rede dividem o espaço de endereçamento em função de seus identificadores. Assim, se considerarmos que a distribuição de carga é uniforme, bastaria dividir de forma justa a estrutura de endereçamento entre os pares da rede para balancear a carga do sistema corretamente. Uma vez que na rede um nó fica responsável pelos recursos que possuem identificadores próximos ao seu, cada nó ficará responsável por uma parcela igual à dos outros nós se cada um for responsável por uma parte do espaço de endereçamento de mesmo tamanho que o de todos os outros nós da rede.

Embora a geração de identificadores por funções criptográficas tenha o objetivo de balancear a distribuição de nós, a distribuição final pode ficar desequilibrada devido à escolha aleatória de identificadores, ou seja, alguns nós podem ser responsáveis por pequenas regiões do espaço de endereçamento e outros nós podem ser responsáveis por grandes regiões desse espaço. Assim, os nós que são responsáveis por uma pequena área

têm uma menor chance de ficarem sobrecarregados. Da mesma forma, nós responsáveis por grandes regiões têm maior chance de ficarem sobrecarregados [5, 32].

Se, ao contrário, considerarmos que a distribuição de carga do sistema é não-uniforme, ou seja, alguns pares da rede possam receber mais ou menos carga independente do tamanho da região do espaço de sua responsabilidade, temos o mesmo problema anterior só que visto por outro ângulo. Agora o que traz o desequilíbrio é a concentração de carga e não apenas o tamanho da região de cada nó. Veja que podemos ter as duas coisas, ou seja, podemos ter uma estrutura de identificadores mal balanceada que recebe uma carga não uniforme. Para resolver esses problemas é preciso manter informações sobre a carga de cada máquina, transferindo tarefas de máquinas sobrecarregadas para as máquinas com pouca carga.

Um outro elemento na questão do equilíbrio é levar em consideração a diferença de recursos entre as máquinas de uma rede. Normalmente, numa rede par-a-par, que é composta por muitos computadores espalhados pelo mundo, a diferença de recursos entre as máquinas é consideravelmente grande. Essa característica pode ser usada para calcular o balanceamento de carga considerando que máquinas com mais recursos podem ser responsáveis por uma maior quantidade de carga.

2.4.0.4 Desbalanceamento inerente ao *hash* consistente

No trabalho original de *hash* consistente, supunha-se a princípio que os identificadores de nós eram igualmente espaçados, o que gerava um bom balanceamento sob carga uniforme. Entretanto, o assinalamento de identificadores aleatórios para nós no espaço de endereçamento não garante um espaçamento uniforme. Stoica, no trabalho com a rede Chord [32], indicou que o desbalanceamento poderia ser significativo, chegando a um fator de $log(n)$ para uma rede com n nós.

Para entendermos melhor as tecnologias de *hash* descritas, vamos realizar a comparação de duas funções de *hash* tradicional com uma função de *hash* consistente em relação à distribuição de tarefas em uma rede que passa por um processo de reconfiguração. Utilizamos, como padrão de geração dos identificadores, o padrão seguido por algumas aplicações reais que são executadas dentro do ambiente distribuído Formigueiro (Anthill) descrito em Ferreira *et alii* [7]. O espaço de chaves simulado foi o da rede Chord, que foi estudada mais detalhadamente na seção 2.3.2.1, ou seja, um espaço de identificadores baseado no número inteiro gerado pela função de *hash* criptográfico SHA-1. O simulador da rede, desenvolvido na linguagem Java, permite a observação da distribuição de um conjunto de chaves antes e após a inclusão ou a remoção de nós

da rede. Os experimentos realizados foram executados apenas uma vez no caso do *hash* tradicional, nesse caso foram utilizadas as funções `mod` e `div`, e diversas vezes no caso do *hash* consistente, devido às suas características de geração aleatória de chaves, para obter-se uma suavização dos resultados. Foi simulado um cenário constituído por uma rede com n nós, com n variando de 10 a 100 e com a distribuição de 15.000 chaves. Foram também utilizados outros cenários para observação do comportamento da rede com o crescimento do número de nós/chaves.

A Figura 2.8 mostra o desvio médio nas distribuições (desvio padrão sobre a média). Pode-se ver pela figura que as técnicas de distribuição com *hash* tradicional geram uma distribuição mais uniforme das chaves. Essa uniformidade diminui um pouco à medida que o número de máquinas na rede aumenta, pois há menos chaves por nó nesse caso. Entre as duas funções tradicionais a melhor distribuição ficou com a função `mod` que apresentou menor variação na uniformidade da distribuição em relação ao aumento de nós da rede. O *hash* consistente apresenta uma distribuição menos uniforme, com um pequeno crescimento com o número de nós.

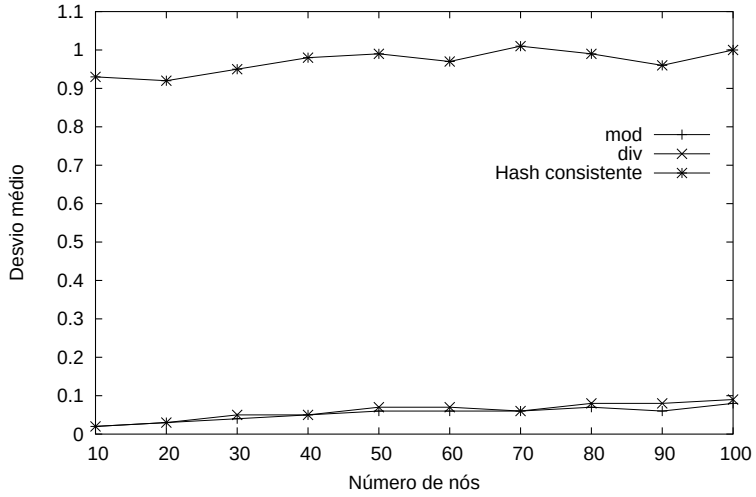


Figura 2.8: Desvio padrão da distribuição de chaves por nós (relativo à média)

Como discutido anteriormente, a maior vantagem dessa de técnica não é a uniformidade da distribuição e sim uma melhor adaptação a modificações no número de máquinas presentes na rede. Como se pode ver na Figura 2.9 a quantidade de chaves transferidas quando há uma reconfiguração da rede é muito menor para o *hash* consistente. Essa transferência de objetos é reduzida à medida que o número de nós aumenta. Já as funções de *hash* tradicional não se comportam tão bem. A função `div` transfere com a inserção de um novo nó aproximadamente metade das chaves distribuídas e a função `mod` se aproxima, à medida que o número de nós aumenta, do total de chaves

distribuídas. Outros cenários e uma análise mais detalhada do mecanismo de *hash* para distribuição de recursos pode ser encontrada em Silva *et alii* [5].

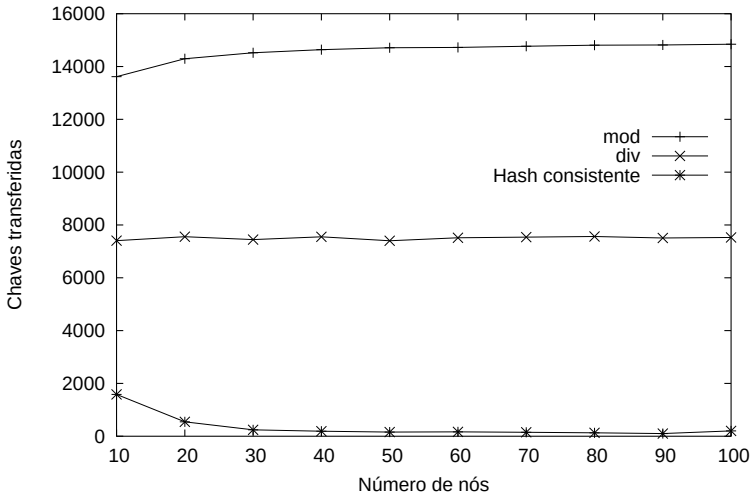


Figura 2.9: Número de chaves redistribuídas

2.4.0.5 Servidores virtuais

Conforme mencionado por Stoica [32], um problema que acontece com a escolha aleatória de identificadores por meio de uma função criptográfica é o desbalanceamento da distribuição de nós no espaço de endereçamento, que pode chegar a um fator de $O(log(n))$ para uma rede com n participantes. Esse tipo de distribuição pode gerar um desbalanceamento na carga do sistema.

Ao identificar o problema de desbalanceamento no Chord, Stoica propôs o uso de servidores virtuais para partição do anel de endereçamento entre os nós. A técnica de servidores virtuais permite uma melhor adaptação da rede à saída e entrada de novos nós graças à camada adicional de gerência do espaço de identificadores. Nessa técnica, cada nó real (ou nó físico) cria uma certa quantidade de servidores virtuais. Cada servidor virtual recebe um identificador aleatório e fica responsável por uma fração do espaço de endereçamento, normalmente escolhida de forma aleatória. Dessa forma, um nó físico fica responsável por vários pequenos pedaços desse espaço e pode atingir uma melhor distribuição de sua responsabilidade (Figura 2.10). Segundo os autores da rede Chord, com o uso de $O(log(n))$ servidores virtuais por nó físico da rede é possível atingir um fator de desbalanceamento constante.

Em comparação com o uso de *hash* tradicional e de *hash* consistente puro podemos ver na Figura 2.11 (a) que a uniformidade da distribuição, embora não seja melhor que

as funções de *hash* tradicional, é melhor que no caso do *hash* consistente com apenas um identificador por nó. Ainda assim, a quantidade de objetos redistribuídos após uma modificação na rede continua tão boa quanto ao *hash* consistente com apenas um identificador (Figura 2.11 (b)).

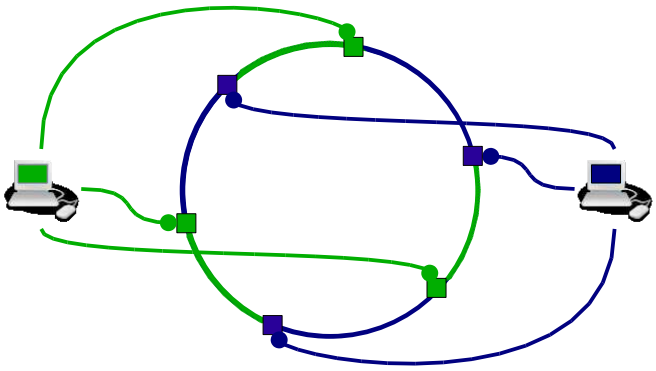
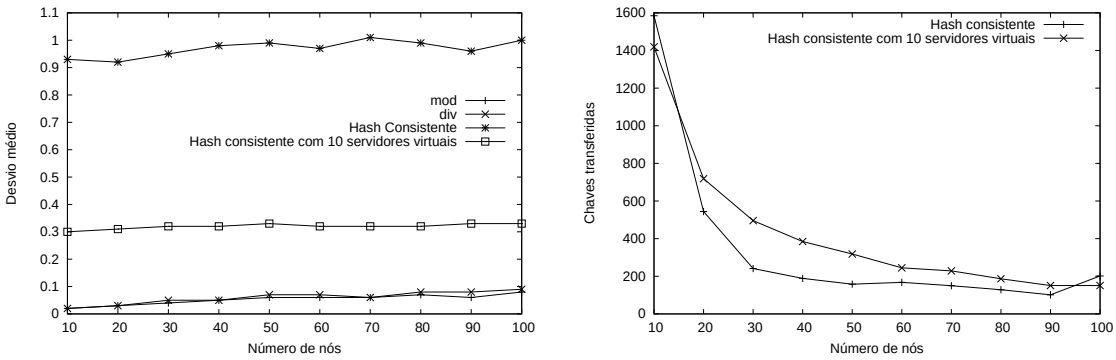


Figura 2.10: Exemplo de divisão do espaço de endereçamento com servidores virtuais. Nesse exemplo, cada nó físico possui três servidores virtuais. Dessa forma a probabilidade de um desbalanceamento diminui.

Nos algoritmos de balanceamento de carga esses servidores virtuais servem para, além de facilitar a distribuição dos segmentos do espaço de endereçamento, facilitar a transferência de carga entre os nós do sistema e a adequação da carga às capacidades de cada nó.

Alguns problemas, entretanto, surgem quando da utilização de servidores virtuais para balancear a distribuição de identificadores em uma DHT. Esses problemas têm



(a) Desvio padrão da distribuição de chaves por nós (relativo à média)

(b) Número de chaves redistribuídas

Figura 2.11: Comparação das técnicas de distribuição de recursos incluindo agora os servidores virtuais.

levado alguns pesquisadores a direcionar pesquisas de balanceamento de carga para a não utilização de servidores virtuais. O primeiro problema é que armazenar as informações de roteamento e de vizinhança aumenta proporcionalmente ao número de servidores virtuais criados. Como o número de servidores necessários para obter uma boa distribuição é logarítmico no número de pares da rede [32], o espaço a mais criado não chega a ter um grande impacto na rede. O segundo problema é manter o estado das tabelas de roteamento e vizinhança atualizada. Para realizar tal operação são necessárias mensagens de atualização enviadas de tempo em tempo para todos os ítens dessas tabelas para se certificar que os servidores estão ainda ativos. Com o aumento de servidores essas mensagens podem ocupar uma grande parte da banda disponível na rede.

Capítulo 3

Análise da literatura de balanceamento de carga em DHTs

Neste capítulo descreveremos resumidamente alguns dos trabalhos que foram estudados para obtenção da taxonomia. São trabalhos da área de balanceamento de carga em DHTs que foram publicados entre os anos 2003 e 2006 e que, acreditamos, representam bem o estado da arte dessa área. Esses trabalhos são a base para a criação da taxonomia proposta aqui.

3.1 Toward a dynamically balanced cluster oriented DHT

O trabalho de Rufino *et al.* [27] apresenta um modelo para uma DHT orientada para o uso em agrupamentos (*clusters*). O principal problema tratado é o balanceamento do espaço de endereçamento entre um conjunto de nós em um agrupamento. A solução proposta trata *clusters* homogêneos e heterogêneos onde a política de uso de um nó é exclusiva, ou seja, cada nó executa tarefas de apenas um usuário por vez. Além disso, o padrão de acesso aos nós da DHT é considerado uniforme, não considerando a popularidade dos dados.

No modelo apresentado existem os nós (chamados *snodes*) que são instâncias de software. A cada nó é associado um conjunto de servidores virtuais (chamados *vnodes*), que definem um conjunto de partições da DHT. Essas partições possuem tamanho fixo, isto é, os identificadores dos servidores virtuais são fixos e pré-definidos. A quantidade de servidores virtuais por nó pode variar dinamicamente. Os servidores virtuais são

utilizados para distribuir os recursos de acordo com a capacidade de cada nó (CPU, memória, disco, banda de rede, etc.). Se o número de partições por nó cair, devido, por exemplo, à saída de nós da rede, as partições são divididas para alcançar novamente o número mínimo de partições, similarmente, se o número de partições crescer, as partições são unidas formando partições de tamanho maior para alcançar novamente o número máximo de partições.

É utilizada uma tabela de distribuição de partições (*Partition Distributed Record* - PDR) que armazena informações globais sobre o número de partições que um determinado servidor virtual possui. Cada entrada dessa tabela possui o seguinte formato: (*snode.vnode_id*, *nparts*). São utilizados esquemas de nomes para os nós físicos, servidores virtuais e as partições da DHT. Por exemplo, *snode_id.vnode_id.partition_id*. O balanceamento do espaço de endereçamento é feito transferindo partições entre os servidores virtuais quando um deles entra ou sai da rede. A abordagem foi avaliada por meio de simulações com distribuição de carga uniforme. Algumas funcionalidades essenciais do modelo proposto são descritas a seguir:

Criação de um nó: um nó emite uma requisição de criação para todos os outros nós da DHT, constituindo um padrão de comunicação global. A requisição só estará completa quando todos chegarem a um acordo sobre a transferência de partições entre os servidores virtuais antigos e o novo. O processo de deleção de um nó da rede é semelhante e, por isso, não será descrito aqui.

Reassinalamento de partições: são definidas as seguintes ações:

- Um novo servidor virtual x coloca na sua tabela PDR uma nova entrada com o número de partições igual a zero: (snode.vnode_x, 0)
- O valor do desvio padrão entre o número de partições de cada servidor virtual e o valor da média ideal são calculados.
- Todas as entradas da tabela PDR são ordenadas pelo número de partições do espaço de endereçamento *nparts* para obtenção do servidor virtual com mais partições.
- Enquanto o desvio padrão diminuir devido à transferência de uma partição para o novo nó, essa transferência é realizada.

3.2 Heterogeneity and load Balance in distributed *hash* tables

Nesse trabalho, Godfrey e Stoica [11] propõem um conjunto de técnicas para melhorar o balanceamento de carga em sistemas distribuídos de larga escala (WANs) que utilizam DHTs. A idéia principal é conseguir tal balanceamento com um baixo *overhead* de movimentação de carga em detrimento do aumento do tamanho das tabelas de roteamento por um fator constante.

Os autores assumem que a carga é uniformemente distribuída no espaço de identificadores. Mesmo em casos onde os nós são homogêneos as redes DHT podem apresentar um fator de desequilíbrio de *O(log n)* [32]. E segundo os autores, esse fator pode aumentar à medida que a heterogeneidade no sistema cresce. Normalmente esse tipo de problema é resolvido com o uso de servidores virtuais. Nesse caso, a cada nó do sistema é associado um conjunto de identificadores coletados de forma aleatória no espaço de endereçamento. Para evitar os problemas de muitos *links* gerados pelos servidores virtuais, cada nó físico agrupa os identificadores de seus servidores virtuais em uma fração aleatória de espaço de identificadores. com esse agrupamento é possível manter o grau de cada nó físico em *O(log n)*. Porém, essa técnica depende de conhecimento global da rede, como o número total de nós. No caso, os autores argumentam que esse valor pode ser estimado.

Outra característica é que a quantidade de servidores virtuais é proporcional à capacidade do nó. As capacidades são normalizadas, e cada nó pode estimar sua capacidade (com um fator de proximidade do real) com alta probabilidade. Nas simulações foram utilizadas distribuições de capacidades de dois tipos: *power law* e distribuição real de capacidades de uma rede *Gnutella*.

3.3 Dynamic load balancing in distributed hash tables

Bienkowski *et al.* [2] considera a suavidade (*smoothness*), ou seja, a razão entre o comprimento do maior e do menor intervalos do espaço de endereçamento, um parâmetro importante da rede. Esse parâmetro informa sobre a carga de armazenamento pois grandes intervalos implicam em maior armazenamento. Isso é válido já que o trabalho considera uma distribuição uniforme de carga. Enquanto diversos outros trabalhos sugerem a utilização de servidores virtuais para balanceamento, esse trabalho sugere uma

redistribuição dos intervalos, transferindo nós com intervalos pequenos para locais de intervalos grandes, dividindo-os. A idéia é definir o tamanho dos intervalos seguindo parâmetros de sistema, ou seja, intervalos maiores que um determinado valor $T1$ são considerados longos, intervalos menores que determinado valor $T2$ ($T2 < T1$) são considerados curtos e os demais intervalos são considerados médios. Os nós que possuem intervalos curtos saem então da rede e entram novamente escolhendo um identificador no meio do maior segmento encontrado em um conjunto de nós pesquisados.

O algoritmo para balanceamento funciona da seguinte forma:

1. São definidos 3 tamanhos de intervalos (*short*, *middle* e *long*)
2. A idéia é minimizar os intervalos grandes sem, contudo, os deixar pequenos demais.
3. O algoritmo funciona em ciclos, em cada ciclo procura-se um conjunto linear de pequenos intervalos para deixar a rede. Assim cada nó sai da rede com uma certa probabilidade e volta no meio de um grande intervalo dividindo-o.
4. O algoritmo funciona diferentemente para diferentes nós. Se um nó pequeno ou médio resolve não sair da rede, ele apenas participa encaminhando as mensagens que chegam para ele.

Para realizar a migração de nós para balancear a carga o sistema precisa contatar diversos nós aleatoriamente, o que determina o padrão de comunicação do algoritmo. Entretanto, as informações de carga para tomada de decisão são todas locais, ou seja, um nó não depende de informações de carga dos outros nós para agir.

3.4 Effective load balancing of peer-to-peer systems

O trabalho de Mondal *et al.* [20] foca o balanceamento de carga envolvendo migração e replicação de itens dados em uma rede DHT de larga escala (WAN). Mais especificamente, se preocupa com redes *peer-to-peer* de compartilhamento de arquivos. Tal objetivo se diferencia dos apresentados até o momento, pois nesse caso o recurso fim é o armazenamento de dados. O sistema é visto como um conjunto de *clusters* de nós e a partir dessa visão são construídas as técnicas de balanceamento *intra-cluster* e *inter-cluster*. As máquinas presentes numa mesma rede local(LAN) são considerados participantes de um mesmo *cluster*. São analisados os compromissos entre replicação e migração de dados. A idéia é que o sistema proposto possa decidir em tempo real pela

replicação ou pela migração dos dados entre nós e *clusters*. Para tomar essa decisão são analisados dados estatísticos que todo nó da rede guarda. Por meio desses dados, chaves (itens de dados) que possuem grande procura podem então ser identificados. O trabalho não diz entretanto como uma DHT pode lidar com replicação de dados e manter o sistema de busca funcionando eficientemente. O trabalho é avaliado apenas em relação à migração de dados.

O padrão de comunicação é ditado pela comunicação entre vizinhos dentro de cada *cluster*. Os *clusters* vizinhos trocam informações de carga. Se um líder detectar que seu *cluster* possui carga superior em mais de 10% em relação à carga média de seus vizinhos ele se considera sobrecarregado. Nesse caso, o líder seleciona os itens de dados que podem estar sobrecarregando o *cluster* e envia para seus vizinhos escolhendo primeiro os que possuem menor carga.

A carga L de um nó P é medida como:

$$L_P = D \times (CPU_P \div CPU_{Total\ no\ cluster})$$

(D *é o número de megabytes retornados*)

Visão do Algoritmo: algumas ações principais do algoritmo são:

- Define a distância T entre 2 *clusters* como sendo o tempo de comunicação entre os líderes de cada *cluster*.
- *Clusters* com uma distância T menor que determinado valor são considerados vizinhos.
- Lideres são responsáveis por coordenar as ações do cluster (*Load-balancing*, *searching*, etc.).
- Lideres armazenam informações sobre seu próprio cluster e sobre seus vizinhos que são atualizadas periodicamente.

O Balanceamento de carga de armazenamento é realizado em 2 níveis:

Balanceamento Intra-cluster: nesse modo, a tomada de decisão é centralizada, ou seja, pares enviam informações para líder que resolve quando agir. Em seguida, o 1º nó — o que tem mais carga — manda carga (itens de dados) para o último — que tem menos carga, o 2º manda para o penúltimo e assim por diante.

Balanceamento Inter-cluster: nesse modo as migrações/replicações são realizadas apenas entre *clusters* vizinhos para evitar transferências de dados para nós distantes o que acarretaria uma sobrecarga de comunicação.

Nessa técnica de balanceamento não são consideradas questões de heterogeneidade dos nós. Entretanto a avaliação da técnica é feita com carga de distribuição de armazenamento não uniforme por meio de simulações.

3.5 A scheme for load balancing in heterogeneous distributed hash tables

O trabalho descrito por Giakkoupis e Hadzilacos [9] fornece mais uma técnica para balanceamento de carga em DHTs de grande escala (WANs) focada no balanceamento do espaço de endereçamento. Nesse caso não são utilizadas migrações de servidores virtuais, o balanceamento ocorre dinamicamente quando um nó entra ou sai do sistema. Questões de heterogeneidade de nós são consideradas nessa técnica.

São descritos dois algoritmos de balanceamento dos segmentos de responsabilidade de cada nó. O primeiro algoritmo é direcionado para ambientes homogêneos e o segundo, que na verdade é uma adaptação do primeiro, é direcionado para ambientes heterogêneos.

Visão do Algoritmo para ambientes homogêneos: A seguir estão algumas ações principais do algoritmo para a entrada de um nó na rede. O processo de saída do nó é semelhante e não será descrito aqui.

- São enviadas requisições para outros nós através do mecanismo de busca (*lookup*). O número de requisições enviadas, entretanto, é proporcional ao tamanho do segmento do nó de *bootstrap*.
- O maior dos segmentos retornados (cada nó é responsável por um determinado segmento do espaço de endereçamento) é dividido ao meio e uma das metades é associada ao novo nó que acabou de entrar na rede.

Visão do Algoritmo para ambientes heterogêneo: O processo é semelhante ao descrito anteriormente. Novamente, não será descrito aqui o processo de saída do nó, pelo mesmo motivo.

- Cada nó tem um peso w definido como uma potência de 2 que fica entre 1 e W (parâmetro do sistema).
- Cada nó é associado com um segmento do espaço de chaves que é proporcional ao seu peso.
- São criados *grupos virtuais* que representam um conjunto de nós. O peso de um grupo G, denominado $w(G)$ é a soma dos pesos de todos os nós do grupo.

- É necessário um mecanismo de gerência de grupos para manipular entradas e saídas de nós.
- São enviadas requisições para grupos de nós através do mecanismo de busca (*lookup*). O número de requisições enviadas é proporcional ao tamanho segmento total formado pelo grupo do nó de *bootstrap*.
- o novo nó é então associado ao grupo do maior dos segmentos retornados.

Não são comentadas na descrição do trabalho questões de distribuição de carga, de forma que foi considerada uniforme para a análise do trabalho. A avaliação da técnica foi realizada por meio de modelagem matemática, onde foram analisados os dois algoritmos propostos.

3.6 Distributed, secure load balancing with skew, heterogeneity, and churn

Ledlie e Seltzer [16] descrevem no seu trabalho uma técnica para balanceamento de carga em DHTs de larga escala que considera a heterogeneidade dos nós de uma rede, uma distribuição não uniforme de carga e alta dinamicidade (entrada e saída de nós) na rede. Em vários outros trabalhos de balanceamento o objetivo principal é manter o tamanho dos segmentos dos nós o mais parecido possível; a técnica proposta, entretanto, se preocupa com o nível de carga instantânea observada em cada nó, considerando nós heterogêneos. Dessa forma, a técnica define algumas métricas que permitem avaliar a carga de um nó. Mais especificamente, é utilizado um esquema de *limiar* para medir essa carga.

Assim, cada nó tem informações da sua própria carga, e dos parâmetros que indicam sobrecarga ou subutilização de recursos. Quem controla a quantidade da carga que cada nó será responsável, entretanto, é um conjunto de servidores virtuais associados a cada nó. Para escolher os identificadores dos servidores virtuais é feita uma requisição de k identificadores. Esses identificadores são gerados seguindo uma técnica de validação para aumentar a segurança do sistema. Os identificadores dos nós são gerados com um a partir de uma função *hash* e de um número x certificado por um centro de certificação. Dessa forma apenas nós autorizados podem gerar identificadores. Esses k identificadores são então utilizados para criação do conjunto de servidores virtuais de cada nó. Cada servidor virtual criado é posicionado no espaço de endereçamento seguindo uma função gulosa que busca uma boa localização levando em consideração como ficará sua carga e a dos seus vizinhos. A Criação de servidores virtuais continua até que a meta de carga do nó seja atingida.

Visão do Algoritmo: A seguir estão algumas ações principais do algoritmo para a entrada de um nó na rede.

- primeiro são determinados os limites de carga e obtidos os k identificadores verificáveis.
- em seguida, os nós responsáveis por esses identificadores são contactados.
- para cada nó contatado, uma função gulosa determina qual permite a inserção de um novo nó com um menor custo. Esse custo é baseado no tamanho do espaço de endereço que cada nó tem e da carga que cada um suporta, considerando que nós com mais recursos suportam mais carga que nós com menos recursos.

As informações do sistema necessárias para realizar o balanceamento são obtidas por meio de *amostragem*, uma vez que um conjunto de nós deve ser consultado antes de um novo servidor virtual ser criado. Além disso, a técnica descrita pode ser empregada de forma *passiva* - criação de servidores virtuais acontece apenas quando um nó entra na rede - ou ativa - criação de novos servidores em locais diferentes pode acontecer periodicamente. Nesse último caso é necessário realizar consultas a identificadores periodicamente. Para avaliar a técnica foram realizadas simulações que utilizavam distribuições não uniformes de carga e de capacidades dos nós.

3.7 Multitifaceted simultaneous load balancing in DHT-based P2P systems

Nesse trabalho, descrito por Aberer *et al.* [1] o principal objetivo é o balanceamento de carga de armazenamento e replicação de dados. A técnica proposta se baseia em um algoritmo descentralizado que constrói uma estrutura de dados distribuída responsável por adaptar o particionamento do espaço de chaves à distribuição de dados para alcançar balanceamento de armazenamento de dados. Esse algoritmo também é capaz de se adaptar a mudanças na distribuição dos dados, que é conseguido por meio do reparticionamento do espaço de chaves. Um mecanismo para realizar a manutenção no processo de replicação também é apresentado.

Como o algoritmo é descentralizado, para obter informações do sistema, cada nó faz uma amostragem em poucos outros nós para obter informações. A partir dessas informações são tomadas decisões locais, por exemplo, replicar uma chave em outra partição do espaço de acordo com o nível de sobrecarga. O algoritmo trabalha sobre a DHT P-Grid. Abaixo serão descritas suas principais características.

P-Grid

Utiliza uma árvore de qualquer aridade (vamos considerar aqui uma árvore binária) para representar a estrutura de buscas. Essa árvore pode ser desbalanceada para refletir inclinações na distribuição dos dados. Embora isso possa sugerir que o processo de busca de uma chave se

torne linear no pior caso, os autores garantem que na prática isso não acontece, permanecendo o processo de busca logarítmico no número de partições do espaço de chaves.

Cada nó é associado com uma folha na árvore binária e possui associado a ele um caminho, que corresponde a uma *string* binária utilizada para realização das buscas. Assim, para realização do roteamento é utilizado um mecanismo de prefixo, para isso, cada nó guarda um conjunto de referências para o nó com o mesmo prefixo com exceção do último bit, que é invertido.

Para lidar com a construção da árvore de forma a torná-la flexível e adaptável à distribuição da carga são definidas várias operações. Essas operações representam apenas possíveis interações entre os nós da rede. Elas podem ser combinadas de diversas formas para se criar estratégias de construção.

3.8 Efficient, proximity-aware load balancing for DHT-based P2P systems

Nesse trabalho, Zhu e Hu [36] procuram resolver o problema de desbalanceamento observado entre os nós de uma DHT como explicado na seção 2.4. Além disso, nesse trabalho são considerados informações de proximidade entre os nós para reduzir os custos de transferência de carga. A distribuição de carga considerada pelo algoritmo é não uniforme. A técnica proposta utiliza a idéia de servidores virtuais e a transferência desses servidores entre nós da rede para atingir o nível de balanceamento desejado.

O esquema de balanceamento proposto é realizado em quatro etapas. Durante essas quatro etapas são coletadas informações de carga/capacidade e tomadas as decisões para balanceamento. Para obtenção das informações de carga e capacidade de servidores virtuais entre os nós é construída uma estrutura de dados distribuída. Essa estrutura é uma árvore que envolve todo o espaço de endereçamento de forma que, após a construção da árvore, cada nó-folha esteja associado a algum servidor virtual. Essa árvore também é utilizada para realizar a transferências de carga entre os nós, o que permite que essa transferência fique independente do sistema de pesquisa da DHT. É assumido que durante o processo de balanceamento a carga do sistema permanece estável. As quatro etapas são descritas abaixo.

Load balancing information agregation - LBI Obtém uma visão das distribuições de carga e capacidade de todo o sistema por meio de um mecanismo de agregação de informação.

Node classification De acordo com as informações obtidas na fase anterior cada nó é classificado como sobrecarregados (heavy), subcarregados (light) e neutros.

Virtual server assignment - VSA Nessa fase, é iniciado um processo de escolha dos servidores virtuais que serão transferidos. Essa escolha é feita levando em consideração a carga e a capacidade de cada servidor virtual, além das informações de proximidade entre nós, de forma que seja possível, por meio de um algoritmo guloso, selecionar os servidores mais próximos que permitam maior equilíbrio de carga com a transferência.

Virtual server transfering - VST Nessa fase ocorrem as transferências de servidores virtuais com base nas escolhas realizadas na etapa anterior.

3.9 Simple load balancing for distributed hash table

O trabalho descrito por Byers *et al.* [3]. tenta, sob a suposição de carga uniformemente distribuída, balancear a carga de um sistema com uma técnica simples. Em vez de simplesmente utilizar uma função *hash* para mapear um item de dados no espaço de endereçamento, como é comumente realizado em redes DHTs, duas ou mais funções são utilizadas, a partir dos identificadores gerados os nós responsáveis são contactados e é escolhido para armazenar o item o nó que está com menos carga no momento.

O algoritmo funciona, então, da seguinte forma:

- Um nó utiliza uma série de funções hash, $h_1(x), h_2(x), h_3(x), \ldots$ para o item x e então inicia várias operações de busca do espaço de chaves, uma para cada ponto obtido.
- Ao contatar os nós responsáveis por esses pontos é avaliado o nível de carga de cada um e é escolhido o que possui menos carga para armazenar o item x.
- Para realizar a busca por um ponto pode-se gerar buscas em paralelo para com todas as funções *hash* utilizadas para armazenar o item. Entretanto, uma idéia menos custosa em termos de comunicação é adicionar, na hora do armazenamento, apontadores dos nós rejeitados (por estarem mais carregados) para o nó escolhido. Com esse nível de indireção é possível escolher apenas um das funções utilizadas e realizar a pesquisa.
- Para manter os apontadores, os autores assumem que os provedores do item x atualizarão a rede periodicamente.

O trabalho analisado apresenta ainda alguma outras formas de se aproveitar do esquema de indireção para melhorar o balanceamento do sistema. Mais especificamente, propõe o uso das técnicas tradicionais de balanceamento com apontadores fazendo a ligação entre os locais de origem e destino da carga.

3.10 A thermal-dissipation-based approach for balancing data load in DHTs

O trabalho de Rieche *et al.* [25] propõe um algoritmo para lidar com balanceamento de carga de número de itens de dados, ou seja, a carga de um nó é medida como a soma dos itens que ele armazena. Para um sistema com N nós o ideal é que cada nó possua uma fração $\frac{1}{N}$ dos itens de todo o sistema. A idéia do algoritmo é utilizar um mapeamento do processo natural de dissipação térmica em materiais, ou seja, considera a carga como energia térmica e a rede de nós como o material no qual a energia se espalha. Dessa forma, cada nó que percebe que sua carga aumentou compartilha-a com os seus vizinhos imediatos. Por exemplo, os sucessores e predecessores no caso de uma rede Chord, ou as regiões adjacentes no caso de uma rede CAN.

Uma modificação da estrutura da rede Chord é proposta com o objetivo de prover maior capacidade para balanceamento e maior tolerância a falhas. A modificação consiste em manter mais de um nó responsável pela gerência de um intervalo. Nesse caso, o conteúdo destinado a determinado intervalo é distribuído entre os nós responsáveis por ele, ou seja, esse conteúdo é replicado em cada um dos nós que gerenciam o intervalo. Para manter a gerência de itens de dados e de números de nós por intervalo cada nó tem um ponteiro para comunicação com cada um dos outros gerentes do intervalo. Essa nova estrutura claramente torna maior o tráfego na rede.

Devido à falta de uniformidade na geração de identificadores com funções *hash*, é comum nas DHTs a presença de intervalos menores que outros. Devido a esse fato o trabalho descrito considera que os nós podem escolher diretamente sua localização no espaço de endereços, ou seja, um nó pode deslocar-se no espaço de endereçamento para tornar o seu segmento maior ou menor. Com isso se pretende controlar o número de elementos por intervalo e assim, melhorar o balanceamento de carga. Podemos perceber, dessa forma, que os autores consideram uma distribuição de carga uniforme.

Visão geral do algoritmo.

- É definido o parâmetro f que representa o número mínimo de nós que devem gerenciar um intervalo.
- Todo nó que entra na rede é apresentado a um nó que informa a todos os nós daquele intervalo a sua presença.
- O balanceamento de carga funciona da seguinte forma:
  - $2f$ nós com carga excessiva: se $2f$ diferentes nós responsáveis por um intervalo tiverem carga (nesse caso, armazenagem de documentos), muito maior que a média, o intervalo é dividido ao meio.

- Mais que f nós em um intervalo: Se um intervalo com mais que f e menos que $2f$ nós não está sobrecarregado, é possível transferir alguns desses nós para intervalos com também mais de f nós de forma que, ao atingir $2f$ nós, o intervalo possa ser dividido. para isso nós com pouca carga devem periodicamente informar sua situação para outros nós de diferentes intervalos. Embora mencionem que isso pode ser realizado pela *finger table*, os autores não dão mais detalhes do procedimento.
- A carga de nós de um intervalo é comparada com a carga de nós localizados em intervalos vizinhos. Caso intervalos vizinhos com f nós estejam menos carregados, o intervalo vizinho pode ser redimensionado, ou seja, a borda do intervalo maior pode ser deslocada de forma a tornar mais uniforme as suas dimensões e, dessa forma, distribuir melhor a carga entre os nós vizinhos.

3.11 Load balancing in hypercubic DHTs with heterogeneous processors

O objetivo do trabalho de Liu e Adler *et al.* [17] é balancear a densidade de carga sobre os nós do sistema. A rede considerada é uma variante de DHT, chamada Hypercubic Hash Table (HHT). Essa rede utiliza uma árvore binária para dividir o espaço de endereçamento, de forma que os ramos da esquerda são rotulados com 0 e os da direita rotulados com 1. Os nós da rede são associados às folhas da árvore e cada item de dados é armazenado no nó que possui como caminho da raíz até o nó o maior prefixo em comum com o hash do item (identificador). Uma forma simples para manter o sistema balanceado quando um nó entra na rede é escolher um nó de entrada de forma que a árvore se mantenha o mais balanceada possível.

O algoritmo proposto considera um sistema de nós com distribuição de capacidades heterogênea. O objetivo do algoritmo é construir a árvore binária de forma a minimizar a diferença entre a maior e a menor partição do espaço associados a um nó. O algoritmo consiste em uma variação do algoritmo de Huffman.

Considerando que cada nó i consiga medir sua quantidade de recursos S_i (que os autores representam por 2^{S_i}), o algoritmo tenta balancear a densidade de carga, $ld_i = L\frac{1}{2^{l_i}}\frac{1}{2^{S_i}}$, onde l_i é a altura da árvore da raiz até o nó i, minimizando $\frac{max(ld_i)}{minld_i}$.

O problema é visto de forma centralizada e assume conhecer o número de nós que existirão na rede após a construção árvore de distribuição de chaves, bem como as capacidades desses nós. Por isso, é possível utilizar um algoritmo parecido com o algoritmo de Huffman, mais especificamente o algoritmo é derivado do algoritmo de Golumbic que encontra uma solução ótima para o problema de minimizar a fração máxima da árvore. A modificação consiste em tratar os casos de empate na escolha de nós segundo um conjunto de regras desenvolvidas. O algoritmo desenvolvido nem sempre atinge a solução ótima para densidade, mas atinge

soluções que são 2-aproximadas da ótima, o que é muito melhor que o algoritmo de Golumbic em alguns casos.

3.12 Simple efficient load balancing algorithms for peer-to-peer systems

Neste trabalho, Karger e Ruhl [14] apresenta duas técnicas para balanceamento de carga em DHTs, a primeira propõe uma nova forma de *hash* consistente, ou seja, tenta atingir o balanceamento da carga do sistema alterando a forma como os endereços dos nós são definidos. A segunda técnica utiliza a movimentação de itens para chegar a um estado de equilíbrio de carga para o sistema.

Na primeira técnica, cada par do sistema possui um conjunto de potenciais posições do espaço de identificadores para se manter ativo. Esse conjunto pode ser formado, por exemplo, pelo hash das tuplas <endereço IP, 1>, <endereço IP, 2>, etc. de forma a aumentar a segurança do sistema de endereçamento, evitando que nós maliciosos se ocupem de qualquer região do espaço de identificadores. Os endereços são formados respeitando um ordenamento definidos da seguinte forma: para um endereço do tipo $< a, b >$ formado por $(2b + 1)2^{-a}$, $< a, b > \prec < a', b' >$ se, e somente se, $a < a'$ ou ($a = a'$ e $b < b'$). Com esse ordenamento é provado então que cada potencial nó cobre uma determinada região entre um nó ativo e outro não ativo. Assim, basta que cada nó procure a posição que cobre a menor região segundo o ordenamento apresentado para ser ativada como seu identificador. A tomada de decisão nesse caso é local, ou seja, cada nó tenta ativar uma posição que preserve a regra de ordenação determinada, levando todo o sistema a um equilíbrio de distribuição dos identificadores.

Na segunda técnica, o objetivo não é balancear a distribuição de nós no espaço de endereçamento, mas sim balancear um sistema que trabalha com uma distribuição de dados arbitrária, ou seja, balancear as chaves entre os nós da rede. Nesse caso a idéia é transferir os itens de dados entre os nós para obter um equilíbrio para a quantidade de carga que cada nó possui. Embora o objetivo seja diferente, a estratégia utilizada é a mesma, modificar a estrutura de endereços dos pares do sistema. Nessa técnica um nó i escolhe aleatoriamente um outro nó j periodicamente e se a sua carga for diferente da observada em i por um fator ε os dois casos a seguir podem ocorrer:

1. $i = j + 1$ Isso significa que i é um vizinho lógico de j e nesse caso os identificadores de um dos dois é deslocado de forma a dividir a carga de itens entre os dois.

2. $i \neq j + 1$ Nesse caso a ação tomada depende da carga, l, do sucessor de j. Se a $l_{j+1} > l_i$ então o nó i toma o lugar do sucessor de j e o descrito no primeiro caso é executado. Caso contrário, j se coloca entre i e $i - 1$ deixando seus itens para o seu sucessor.

3.13 Load balancing in structured P2P systems

Nesse trabalho, Rao *et al.* [23] apresentam técnicas para balanceamento de carga em redes par-a-par estruturadas com a utilização de servidores virtuais. São descritas três técnicas que mudam principalmente em relação à quantidade de informação usada para decidir a origem e o destino da carga. Um nó possui como carga a soma da carga observada em todos os seus servidores virtuais. Se essa carga ultrapassa um determinado parâmetro T (razão entre a carga média do sistema e a capacidade média dos nós do sistema) o nó é considerado sobrecarregado.

As técnicas apresentadas funcionam pela transferência de servidores virtuais de um nó com muita carga para um nó com pouca carga. A quantidade de carga transferida é sempre a mínima possível que consiga deixar o nó de origem abaixo do parâmetro T. Não há transferência entre nós que estão com o mesmo estado de carga, ou seja, não há transferência entre dois nós com carga menor que T nem entre dois nós com carga maior que esse parâmetro. Isso permite que, caso a carga total do sistema seja muito alta, o sistema não entre em estado de *trashing*. Em seguida serão descritas as três técnicas:

Um-para-um: nesse caso um nó com pouca carga escolhe um outro nó aleatoriamente e se esse nó estiver sobrecarregado é iniciada então a transferência de servidores virtuais entre os nós.

Um-para-muitos: nesse caso são criados diretórios que são armazenados nos pares da rede e são responsáveis por guardar informações de carga enviada por servidores com pouca carga. Esses diretórios possuem um identificador gerado por uma função *hash* diferente da usada para mapear os objetos da rede. Um servidor com muita carga então contata um desses diretórios e escolhe o servidor virtual que melhor reduz sua carga com o mínimo de transferência.

Muitos-para-muitos: nesse caso tanto nós com pouca carga quanto nós com muita carga enviam informações para um diretório. Esse diretório então calcula qual servidor virtual deve ir para qual nó usando uma estratégia gulosa que minimiza a quantidade de carga movida. Após calcular as transferências o diretório envia de volta essa informação para os nós que o contactaram.

3.14 Load balancing in dynamic structured P2P systems

O trabalho proposto por Godfrey *et al.* [10] lida com a balanceamento de carga em redes peer-to-peer estruturadas considerando o comportamento dinâmico dessas redes, ou seja, considera

que nós entram e saem da rede muito frequentemente. Além disso, a distribuição de tamanhos ou da popularidade dos dados também podem sofrer alterações.

O algoritmo funciona como uma extensão de algoritmos estáticos previamente definidos em [23]. A idéia é centralizar informações de carga e capacidade em determinados nós denominados diretórios. Esses diretórios executam então o agendamento de transferências entre os nós mais e menos carregados. São considerados nós sobrecarregados os que estiverem com a sua taxa de utilização acima de 1. Essa taxa é calculada como a razão entre a carga observada em determinado nó e a sua capacidade. Cada nó possui $log(n)$ servidores virtuais que são utilizados para realizar a transferência de carga no sistema, ou seja, os nós físicos sobrecarregados transferem servidores virtuais para os nós que possuem pouca carga. Esse algoritmo é periódico para tratar a dinamicidade da rede. O processo de agendamento/transferência acontece periodicamente segundo um parâmetro T do sistema.

Além da ação periódica, o algoritmo prevê também ações a serem tomadas para resolver problemas emergenciais, assim, se a carga de utilização de um nó ultrapassa o limite de utilização parametrizado K_e, o nó imediatamente envia as informações de carga para um diretório e solicita a transferência de servidores virtuais até que a taxa de utilização volte a ficar abaixo de K_e.

Para realizar a transferência de servidores virtuais o algoritmo que roda no diretório utiliza uma abordagem gulosa para obter uma solução aproximada da ótima, já que, computar o reassinalamento ótimo de nós é um problema NP-Completo. Assim, ao obter informações de carga e capacidade de diversos servidores virtuais, o algoritmo realiza o processo de agendamento de transferência tentando minimizar a utilização dos servidores.

3.15 Hash-based proximity clustering for load balancing in heterogeneous DHT networks

A técnica de balanceamento apresentada por Shen e Xu [29] nesse trabalho lida com a idéia de supernós, ou seja, nós da rede que possuem maior capacidade possuem funções diferentes dos nós com menos capacidade, estes são chamados de nós regulares pelo autores. Normalmente são criados agrupamentos de nós, sendo cada agrupamento gerenciado por um supernó. Normalmente, Como supernós são escolhidos gateways estáticos ou roteadores, de forma que os agrupamentos sejam formados pelos nós nas redes locais relacionadas a esses supernós. Para evitar o custo de descobrir gateways (custo do uso de traceroute é alto) os autores usam uma técnica de pontos de referência para gerar informação de proximidade. Tal informação é conseguida considerando a intuição de que nós próximos entre si possuem distância semelhante a um determinado ponto de referencia. São espalhados então m pontos de referência pela rede e cada nó mede a sua distância para cada um dos pontos formando um vetor de

distâncias $< d_1, d_2, ..., d_m >$. Esse vetor que representa então a localização do nó no espaço m-dimensional é mapeado para um número real chamado número de Hilbert que é utilizado para identificar os nós. Assim, baseado nessas informações é formada uma rede entre os supernós para transferência de informações de carga.

Essa rede auxiliar formada para realizar o balanceamento de carga na DHT pode ser formada por nós que estão próximos fisicamente, nesse caso é dado o nome de pCluster, ou pode ser formada por nós que estão próximos logicamente, para esse tipo de rede é dado o nome de vCluster.

Em um pCluster cada nó regular possui uma ligação com o supernó mais próximo a ele fisicamente, esse supernó se liga a outros supernós por sua vez formando uma DHT auxiliar. Nessa rede os supernós são vistos como nós e os nós regulares como chaves que são mapeadas para cada supernó. Esse mapeamento, entretanto é feito por proximidade do identificador obtido pelo método de pontos de referência e não gerado por uma função criptográfica, como é comum.

Um vCluster é, por sua vez, uma estrutura onde cada nó regular possui uma ligação ao supernó mais próximo em relação ao espaço de identificadores. Nesse caso o número de Hilbert de cada do nó é usado como chave para a função de inserção de objetos na rede. A informação é então roteada seguindo os critérios de armazenamento da DHT. O algoritmo de balanceamento de carga funciona da mesma forma para as duas redes pCluster e vCluster e será descrito resumidamente abaixo.

Visão geral do algoritmo:

- Os nós chamados regulares enviam informações sobre sua carga para certos supernós.
- Os supernós então organizam as informações de carga separando-as em duas listas, uma destinada aos nós com pouca carga (Sorted Donating List - DSL) e outra destinada aos nós com muita carga (Sorted Starving List - SSL).
- Os supernós trocam então as cargas utilizando as suas próprias listas DSL e SSL. Esse procedimento é chamado de balanceamento local.
- Os supernós trocam então carga, por meio de suas listas DSL e SSL, com outros supernós e, para isso, os contactam da seguinte forma:
  - primeiro são contactados supernós vizinhos a ele escolhidos aleatoriamente da tabela de roteamento. Isso garante que o supernó escolhido esta fisicamente próximo (no caso de pClusters) ou logicamente próximo (no caso de vClusters).
  - Em seguida são escolhidos supernós aleatoriamente dentro de espaço de endereçamento.

Relacionamento entre os trabalhos analisados

Pode-se ter ainda, uma idéia do relacionamento entre tais trabalhos por meio de suas citações, ou seja, qual trabalho é referenciado por outro trabalho da mesma área. Uma descrição desse relacionamento pode ser vista na Figura 3.1. Podemos notar, nessa figura, que o trabalho de [27] não possui ligação com os demais. Isso se deve à natureza do trabalho apresentado, que se aplica a um ambiente centralizado de computadores, como o de um *cluster*.

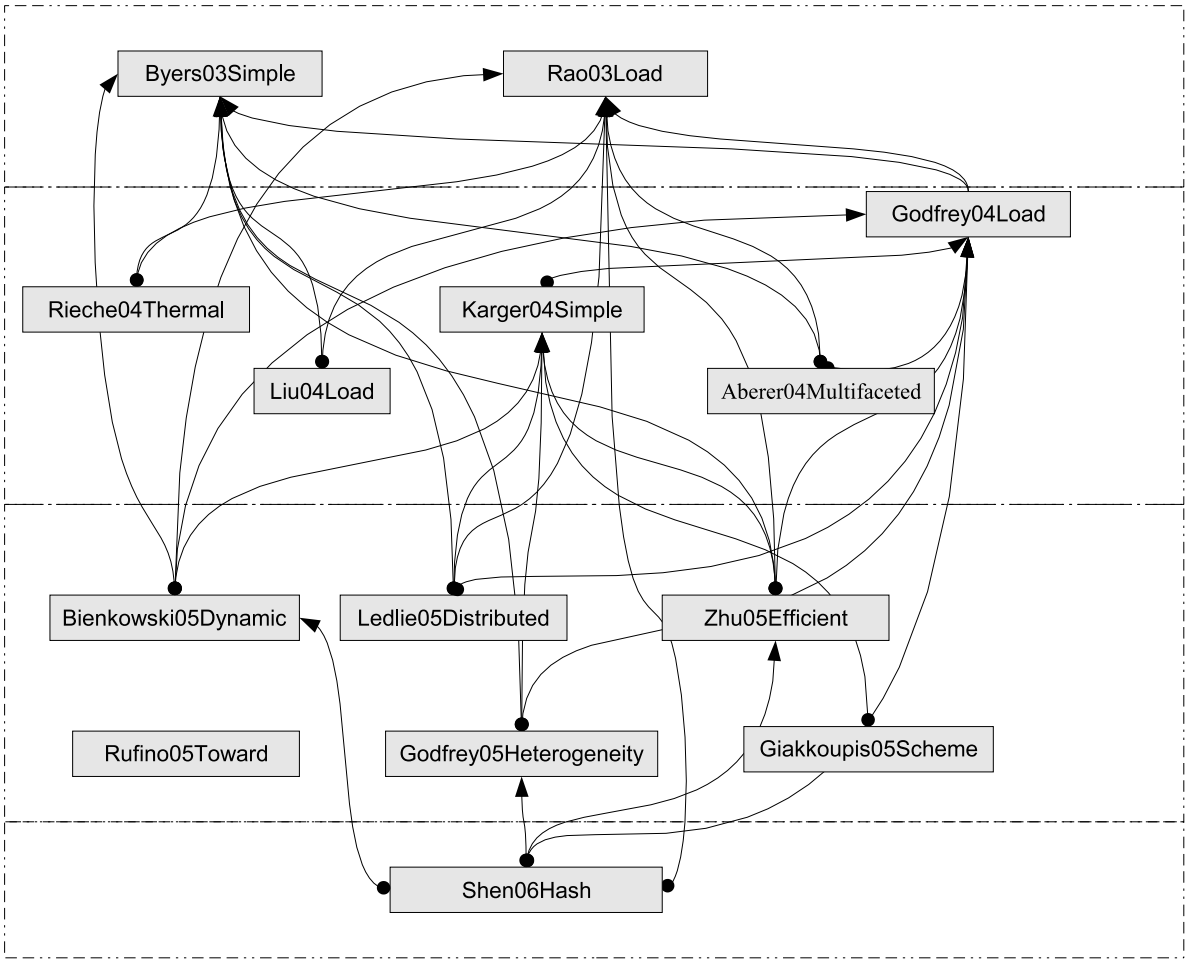


Figura 3.1: Relacionamento entre os trabalhos em termos de citação.

Capítulo 4

Taxonomia para balanceamento de carga em DHTs

Taxonomia é a *ciência responsável pela caracterização e designação dos seres vivos* [34]. Por extensão, podemos dizer que taxonomia atualmente é a caracterização e designação de qualquer área do conhecimento. De acordo com Bicudo [6] taxonomia é a ciência da identificação, ou seja, a ciência que dá nomes aos objetos e conceitos das áreas de conhecimento. Essa ciência teve início com Aristóteles no século IV a.C. quando o mesmo classificou os animais pelo tipo de reprodução e pela cor do sangue. A partir daí, diversos estudos foram desenvolvidos no processo de classificação até a obtenção da taxonomia mais conhecida hoje que é a taxonomia dos seres vivos.

4.1 Metodologia para obtenção da taxonomia

Para realizar a construção de uma taxonomia é preciso reunir um conjunto de informações significativas sobre á área escolhida. Para realizar tal tarefa foi necessária a leitura e análise de diversos trabalhos relevantes da área de balanceamento de carga em DHTs. Assim, para que fizesse parte do conjunto de trabalhos analisados, um determinado artigo tinha que ter sido publicado em algum evento ou congresso. Isso garante que os trabalhos analisados tenham passado por uma avaliação dos revisores de um congresso e que, dessa forma, é relevante para o trabalho em questão.

Uma vez definido o conjunto de artigos foram identificadas várias características de cada um deles. As características observadas foram então classificadas como relevante ou não pra identificação dos trabalhos. Uma característica deixa de ser relevante se for a descrição de um detalhe de implementação ou da definição da técnica de balanceamento de carga. Além disso, se uma determinada característica já é automaticamente representada por outra, não

precisamos utilizá-la, pois o objetivo final é conseguir colocar os trabalhos em classes que possuam as mesmas características. Por exemplo, poderíamos classificar os algoritmos em relação ao nível de informação de proximidade física utilizada, entretanto, já existe uma dimensão da taxonomia que engloba tal propriedade. Nessa caso, uma segunda classificação seria redundante.

Além disso, algumas características não tem como objetivo classificar os trabalhos mas sim ajudar na comparação entre os mesmos, por exemplo, durante o processo de análise, todos os trabalhos foram classificados em relação à forma de avaliação da técnica que podiam ser de três formas: simulação, implementação e modelagem matemática. Não faz sentido utilizar essa informação como uma dimensão da taxonomia, pois qualquer trabalho pode ser submetido a qualquer um dos três modos de avaliação. A utilização de tal classificação entretanto auxilia no processo de análise, principalmente em relação ao entendimento de detalhes do algoritmo.

Esse processo de classificação é um processo iterativo, pois à medida que o conhecimento vai se estruturando e amadurecendo é possível modificar as dimensões da taxonomia acrescentando ou retirando dimensões. Assim, os diversos artigos foram analisados diversas vezes, e a cada vez que o processo de classificação se repetia os ajustes foram sendo incorporados à taxonomia final.

4.2 Descrição da taxonomia obtida

Com base na análise dos trabalhos apresentados, foi criada uma taxonomia constituída de oito dimensões identificadas, que são ilustradas na Figura 4.1

4.2.1 Ambiente de aplicação

A definição do ambiente de aplicação permite identificar a quantidade de controle sobre determinada técnica de balanceamento de carga. Embora a tecnologia DHT tenha sido desenvolvida para resolver o problema de redes par-a-par de grande escala, nada impede a utilização dessa tecnologia num ambiente controlado centralizado. Nesse caso, um ambiente centralizado pode ainda se valer das vantagens de adaptação à entrada e saída de nós em uma rede presentes em uma rede DHT. Por outro lado, é importante também definir quais técnicas de balanceamento funcionam bem em ambientes muito distribuídos e de grande escala.

Dessa forma, as técnicas foram divididas de acordo com o ambiente de aplicação em *WAN* (Wide Area Network), normalmente ambientes de redes P2P estruturadas, e *LAN* (Local Area Network), normalmente ambientes de clusters. Alguns trabalhos, entretanto, são bem teóricos e nem levam em consideração o ambiente de aplicação, baseando-se unicamente no espaço de endereçamento sem mencionar questões de roteamento de mensagens. Esses trabalhos tiveram

seus ambientes de classificação definidos como WANs, uma vez que a tecnologia DHT se aplica às redes P2P estruturadas naturalmente.

4.2.2 Dinamicidade

Classificar as técnicas de balanceamento em relação a sua capacidade de lidar com a dinamicidade da rede é fundamental. Dessa forma, as técnicas de balanceamento podem ser classificadas, de acordo com a taxonomia criada, como *estáticos* ou *dinâmicos*. Com essa informação é possível saber se a técnica foi desenvolvida pensando em um ambiente onde as máquinas podem sair e entrar na rede com freqüência, uma vez que essa é uma das características mais importantes de uma rede par-a-par estruturada, nesse caso os trabalhos foram classificados dentro da subcategoria *Passiva* já que se não houver entradas e saídas de nós o sistema não continua o balanceamento. Não consideramos aqui, entretanto, que um algoritmo é dinâmico apenas se ele lida com saídas e entradas de nós na rede. Outra forma de dinamicidade que é considerada é se o algoritmo lida com possíveis mudanças das cargas dos nós, ou seja, se o a técnica de balanceamento é executada de periodicamente, nesse caso, os algoritmos foram classificados dentro da subcategoria *Ativa* pois a técnica de balanceamento funciona periodicamente independente da entrada e saída de nós da rede.

Embora possa parecer estranho que uma técnica desenvolvida para um ambiente naturalmente dinâmico, o caso das redes P2P estruturadas, não considere questões de dinamicidade, várias técnicas analisadas se preocupavam apenas com a situação estática da rede. Em alguns casos isso acontece porque a técnica busca balancear a carga da rede através de uma distribuição justa do espaço de endereçamento, seja para redes homogêneas ou não. Em outros casos o que acontece é simplesmente a análise da rede como se a distribuição de carga não se alterasse.

4.2.3 Tipo de nó

Separar os diversos algoritmos de balanceamento de carga entre os que utilizam ou não servidores virtuais é importante, pois a escolha de servidores virtuais, como visto na seção 2.4.0.5, implica em uma maior flexibilidade para a transferência de carga entre nós da rede. Além disso, o simples uso de servidores virtuais já realiza um melhor balanceamento da carga se a distribuição da carga for uniforme. O uso desses servidores, entretanto, aumenta bastante o custo de manutenção da rede devido às mensagens de atualização das tabelas de roteamento. Assim, algoritmos que utilizam servidores virtuais têm que trabalhar com o compromisso de aumentar a flexibilidade de transferência de carga com o aumento da comunicação para atualização de tabelas.

4.2.4 Distribuição da carga

A carga aplicada a uma rede DHT pode ser constituída de diversas formas como, por exemplo, o número de requisições que uma determinada máquina tem que responder, o tempo de execução de tarefas, o número de itens que uma máquina deve armazenar, etc. A distribuição dessa carga entre as diversas máquinas de uma rede pode ser realizada uniformemente, ou seja, cada máquina recebe a sua fração justa do total de carga atribuída ao sistema, ou pode ser não uniforme, ou seja, algumas máquinas da rede são responsáveis por mais carga que outras máquinas. Uma distribuição de carga pode se comportar de acordo com algum padrão de distribuição conhecido, por exemplo, nos trabalhos analisados foram encontradas distribuições do tipo *zipf*, *pareto* e *gaussiana*.

Na taxonomia criada, os algoritmos foram classificados como *uniformes* e *não uniformes* em relação à distribuição de carga. Essa classificação é essencial, pois diversos algoritmos estudados assumem que a carga aplicada é uniforme. Esses algoritmos buscam em geral balancear o espaço de endereçamento da rede pois assim já estariam balanceando a carga. Entretanto, numa rede par-a-par essa situação de distribuição uniforme pode não refletir a realidade. Por exemplo, numa rede de compartilhamento de arquivos existem arquivos que são muito mais populares que outros, como músicas muito famosas ou vídeos que se tornam rapidamente populares na Internet, para esses casos as máquinas que possuem esses arquivos terão que lidar com um número de maior de requisições.

4.2.5 Heterogeneidade

Da mesma forma que a distribuição da carga influencia no balanceamento de uma rede par-a-par, a heterogeneidade (ou distribuição de capacidades) dos nós também tem um papel fundamental na definição de qual par está ou não sobrecarregado. Por heterogeneidade entende-se a forma pela qual os recursos (CPU, disco, banda de rede, etc) estão distribuídos entre as máquinas. Esta, pode ser uniforme (ou homogênea), no caso em que todas as máquinas da rede possuem a mesma quantidade de recursos e podem também ser não uniforme, ou heterogênea, no caso em que algumas máquinas possuem maior quantidade de um determinado recurso que outras máquinas da rede. As distribuições de capacidades encontradas durante a análise dos algoritmos foram as distribuições *zipf*, *pareto* e distribuição de capacidade real de uma rede Gnutella.

Essa classificação é importante para o entendimento das técnicas de balanceamento, pois, em ambientes de distribuição de carga não uniforme, uma forma bastante utilizada de amenizar o problema de sobrecarga de alguns nós é exatamente fazendo com que máquinas que possuam maior capacidade sejam responsáveis por recursos mais populares. Mesmo em ambientes com distribuição de carga uniforme, a utilização da informação de heterogeneidade dos nós para uma melhor distribuição desses no espaço de endereçamento permite uma distribuição

mais justa da carga do sistema, pois é razoável que máquinas que possuam maior capacidade processem mais carga que máquinas que possuem menor capacidade. Assim, foi criada na taxonomia uma dimensão para classificar os algoritmos de acordo com a sua heterogeneidade para lidar com uma distribuição de capacidades *homogênea* e *heterogênea*.

4.2.6 Padrão de comunicação

O padrão de comunicação dos algoritmos de balanceamento em uma rede DHT é dado principalmente pelo uso da primitiva de busca de objetos nessa rede. Ao obter informações sobre a carga ou sobre o tamanho do segmento do espaço de endereçamento associado a um nó é possível obter informações suficientes para execução desses algoritmos. A maioria das técnicas analisadas contacta nós aleatoriamente para obter informações para tomada de decisão sobre quanto de carga e para quem ela será transferida. Para realizar essa tarefa são normalmente contactados então um ou mais nós que são escolhidos aleatoriamente dentro do espaço de chaves e uma vez escolhido um conjunto de nós esses são contactados utilizando a primitiva de busca da rede. Outros algoritmos entretanto utilizam nós da sua vizinhança para obter informações e/ou transferir carga. Essa vizinhança pode ser física, ou seja, é utilizado algum mecanismo para obter os vizinhos que estão mais próximos fisicamente do nó que deseja transferir carga ou informação de carga. A vizinhança pode, contudo, ser definida logicamente, nesse caso são considerados vizinhos de um par os nós sucessores e antecessores desse par em relação ao espaço de endereçamento, ou ainda, os nós que fazem parte da tabela de roteamento do par. Por fim, embora uma DHT seja mais apropriada para um ambiente de grande escala e bastante distribuído, em algumas técnicas os nós precisam se comunicar com todos os outros nós da rede definindo, assim, um padrão global de comunicação. Para representar essas quatro formas de comunicação foram definidos as seguintes classificações:

- *comunicação global*: quando todos os nós precisam trocar informações entre si para execução do algoritmo de balanceamento.
- *conjunto de nós escolhidos aleatoriamente*: quando os identificadores dos nós que participam do algoritmo são escolhidos aleatoriamente. Em alguns casos apenas um nó é escolhido aleatoriamente para realizar o balanceamento, tais casos também foram incluídos nesta classificação.
- *vizinhança física*: Nesta categoria estão os algoritmos nos quais os nós tenham comunicação direta física com seus vizinhos.
- *vizinhança lógica*: Nesta categoria estão os algoritmos nos quais os nós tenham comunicação direta lógica com seus vizinhos, ou seja, nos quais um par da rede possua comunicação com seus vizinhos e contados em relação ao espaço de endereçamento.

4.2.7 Objetivo do balanceamento

Em redes do tipo DHT que funcionam de acordo com uma estrutura de mapeamento de chaves num espaço de endereçamento, a porção desse espaço associada a uma máquina define a quantidade de carga pela qual essa máquina será responsável se a distribuição de chaves/objetos na rede for uniforme. Assim, manipular a estrutura da rede para manter os segmentos do espaço de identificadores parecido para cada máquina faz com que a carga fique balanceada no sistema. Por outro lado, se a distribuição de chaves é não uniforme, a manipulação da estrutura pode ser utilizada para balancear o sistema destinando segmentos maiores para máquinas com pouca carga e segmentos menores para máquinas com muita carga.

Uma outra forma de balancear a carga de uma DHT é utilizando as técnicas de balanceamento já conhecidas para sistemas distribuídos, nas quais os recursos importantes são monitorados em cada máquina e se o nível de carga de uma máquina excede um determinado valor ou se essa máquina se torna mais sobrecarregada que um outro por um determinado fator, alguma ação é tomada para equilibrar a carga do sistema, por exemplo, a transferência de itens de dados para outra máquina menos carregada. Essa ação, nesse caso, não modifica a estrutura de endereçamento do sistema. Assim, para diferenciar os algoritmos de balanceamento que modificam a estrutura de endereçamento daqueles que não a modificam foi criada a dimensão de objetivo do balanceamento, que reflete a estratégia utilizada para equilibrar a carga do sistema e os algoritmos podem ser classificados nessa dimensão em *espaço de endereçamento* quando a estrutura da rede é modificada ou em *recursos de um nó* quando apenas informações de carga de cada nó são utilizadas e não há modificação da estrutura de balanceamento.

4.2.8 Tomada de decisão

Antes de decidir como transferir carga de uma máquina para outra numa rede DHT é preciso reunir informações que permitam tomar a decisão de transferência da melhor forma possível de acordo com os propósitos da estratégia de balanceamento. Sempre ao decidir como reunir as informações da rede é preciso decidir entre o custo de comunicação pra obter tais informações e a qualidade final da decisão tomada. Por exemplo, se um algoritmo consegue um bom balanceamento utilizando apenas informações locais de cada máquina, esse algoritmo pode ser considerado então muito bom sobre esse aspecto, pois o custo de comunicação para tomada de decisão foi o mínimo possível. Por outro lado, se um algoritmo utiliza informações globais para tomada de decisão sem necessidade, um grande nível de comunicação vai ser exigido da rede. Nas redes DHTs os algoritmos de balanceamento podem utilizar três formas de reunir informações para tomada de decisão, essas três formas indicam em que tipo de informação a carga transferida é baseada. As três formas são as seguintes:

- *Global*: para os casos em que os nós têm acesso as informações de carga de todo o sistema.
- *Local*: para os casos em que os nós têm acesso apenas a informações deles próprios.
- *Aleatório*: para os casos em que os nós têm acesso a alguns outros nós, escolhidos de alguma forma e a partir desses nós são tomadas as decisões. As decisões podem refletir um balanceamento local ou global (por meio de estimativas).
- *Vizinhança*: para os casos em os nós tem acesso apenas à região de sua vizinhança.

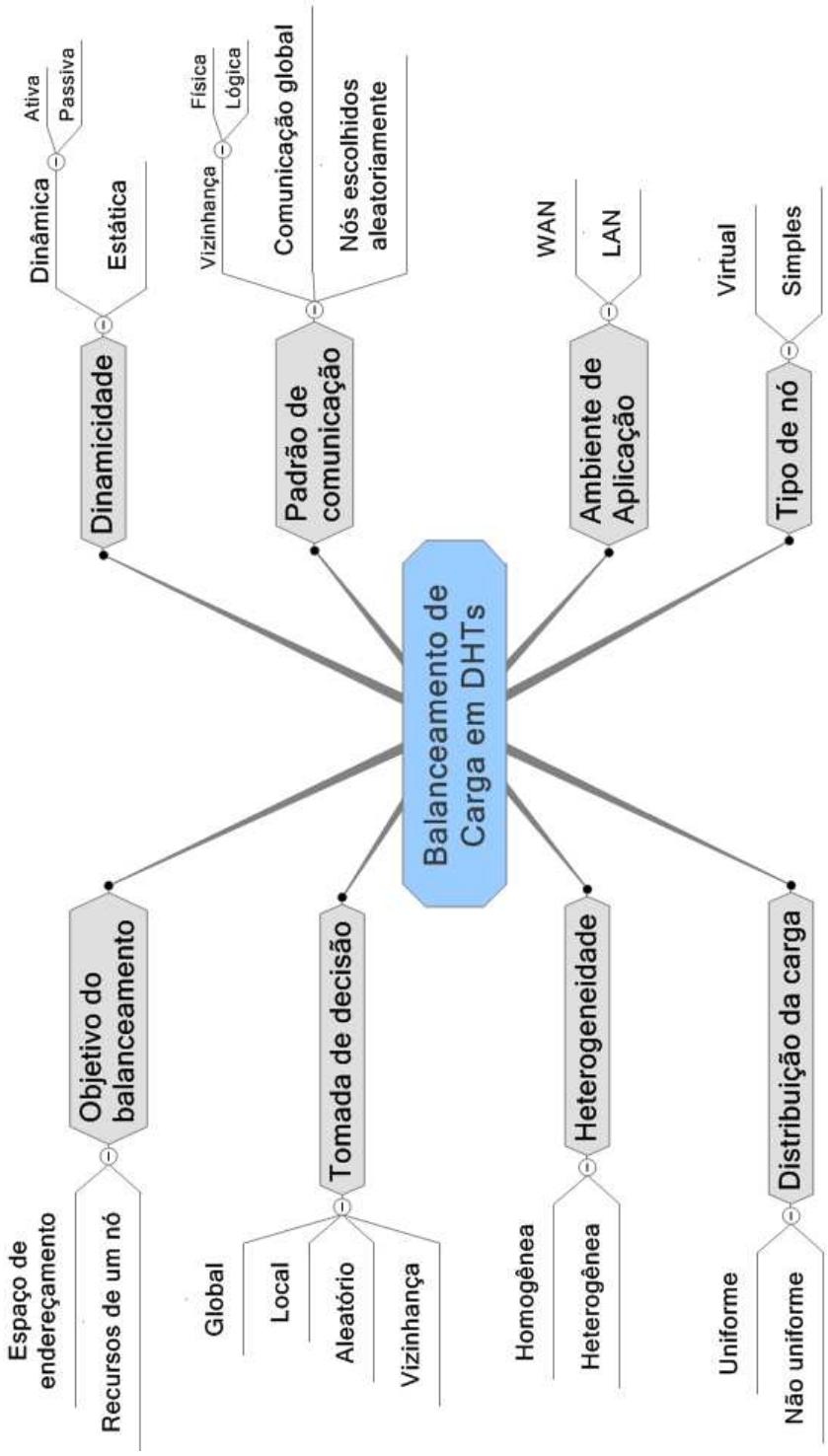


Figura 4.1: Taxonomia criada para a área de balanceamento de carga em redes DHT.

4.3 Aplicação da taxonomia aos trabalhos de balanceamento de carga em DHTs

A partir da definição da taxonomia para balanceamento de carga em redes DHTs é possível classificar os trabalhos nessa área utilizando as dimensões da taxonomia desenvolvida, essa classificação será apresentada nas próximas seções:

4.3.1 Toward a dynamically balanced cluster oriented DHT

Esse algoritmo foi desenvolvido para um ambiente de *clusters* de computadores caracterizando o ambiente de aplicação como ***LAN***. A estratégia de balanceamento consiste em controlar a estrutura de endereçamento, ou seja, o espaço de identificadores é dividido em pequenas partições e essas partições são associadas à vários servidores virtuais, isso permite a classificação do trabalho em relação ao objetivo do balanceamento dentro de ***Espaço de Endereçamento*** e em relação ao tipo de nós utilizados dentro de ***Virtuais***. O padrão de comunicação é determinado pela transferência de informações entre todos os pares da rede, o que determina a classificação do trabalho de acordo com o padrão de comunicação dentro de comunicação ***Global***, essa troca de informação ocorre até que o registro de distribuição de partições esteja completo, com informações de todos os servidores virtuais da rede. Essa informação é então utilizada para realizar a troca de carga entre os diversos servidores virtuais. Essa forma de reunir informação determina o modo de tomada de decisão do algoritmo dentro da classificação ***Global***.

Esse processo de reunião e transferência de partições ocorre sempre que os nós entram ou saem da rede, determinando a dinamicidade da rede dentro da classificação ***Dinâmica Passiva***. Os autores comentam a possibilidade de utilizar informações de capacidade dos nós para criar servidores virtuais proporcionalmente as suas capacidades, entretanto, essa informação não foi utilizada na descrição da técnica ou na forma de avaliação por simulação. Dessa forma, o trabalho foi classificado de acordo com a heterogeneidade dentro da categoria ***Homogênea***. Como os autores não comentam a forma da distribuição de carga com a qual o algoritmo é capaz de lidar consideramos a distribuição que melhor se adapta a algoritmos que atuam na estrutura do endereçamento. A distribuição de carga, então, foi classificada como ***Uniforme***.

4.3.2 Heterogeneity and load balance in distributed hash tables

Desenvolvido como uma nova DHT que suporta balanceamento de carga através de uma melhor distribuição dos nós pelo espaço de endereçamento, essa técnica abrange as DHTs de forma geral e, dessa forma, define o ambiente de aplicação como ***WAN***. Nessa técnica um conjunto de servidores virtuais são criados proporcionalmente à capacidade de cada nó, isso determina duas dimensões da taxonomia criada, a primeira é a definição do tipo de nós como ***Virtuais*** e a heterogeneidade dos nós como ***Heterogênea***, nesse caso, as distribuições utilizadas na avaliação foram uma distribuição sintética que segue uma distribuição *power low* e heterogeneidade real baseada na rede *Gnutella*. A estrutura de endereçamento é definida de acordo com a capacidade de nós, nesse caso é utilizado uma técnica de agrupamento, que faz os servidores virtuais ficarem dentro de um segmento controlado, permitindo a eliminação do excessivo número de conexões presentes em técnicas que utilizam servidores virtuais. A modificação da estrutura de endereçamento define a técnica em relação ao objetivo do balanceamento como ***Espaço de Endereçamento***. De acordo com os próprios autores a DHT proposta é capaz de lidar com uma distribuição de carga ***Uniforme*** definindo, assim, essa dimensão da taxonomia.

Em relação ao padrão de comunicação, a técnica foi classificada como ***Conjunto de Nós Aleatórios*** pois essa é forma de obter os pontos com os quais serão criados os servidores virtuais durante o processo de inserção de um nó na rede. O algoritmo necessita de informação global (número de nós da rede) para realizar a tomada de decisão que, nesse caso, é decidir o local onde será colocado o novo nó. Os autores argumentam, entretanto, que há informação é obtida por estimativas com conjuntos de máquinas. Por isso, o algoritmo é classificado em relação à tomada de decisão na categoria ***Aleatório***. Uma vez que esses valores estimados se modifiquem os parâmetros que definem a estrutura de endereçamento também são alterados o que constitui um comportamento dinâmico, assim, em relação à dinamicidade o algoritmo é considerado ***Dinâmico Ativo***.

4.3.3 Dynamic load balancing in distributed hash tables

Essa técnica foi definida para ambientes de larga escala, caracterizando um ambiente de aplicação ***WAN***. Ela tenta suavizar os segmentos do espaço de endereçamento destinados aos nós da rede. Uma rede suave, nesse caso, significa que a diferença entre o maior e o menor seguimento é pequena. Essa característica define a classificação segundo o objetivo do balanceamento como sendo uma estratégia para balancear o ***Espaço de endereçamento*** da rede. Considera, como vários outros algoritmos a distribuição de carga como ***Uniforme***. Nesse algoritmo não foram considerados questões de heterogeneidade dos nós da rede. Assim, foi classificado de acordo com a heterogeneidade dos nós como ***Homogêneo***. Os autores

pregam a não utilização de servidores virtuais e indicam que as transferências de carga podem ser realizadas pela simulação de entrada e saída dos nós da rede. Em relação ao tipo de nó utilizado a técnica foi classificada então como ***Simples***.

No algoritmo cada nó classifica seu seguimento como *pequeno*, *médio* ou *grande*. Se um nó possui um seguimento pequeno ele então contata um conjunto de servidores utilizando a primitiva de busca e procura um nó com seguimento grande para dividi-lo ao meio. Esse padrão de comunicação define a classificação do algoritmo na categoria ***Conjunto de Nós Aleatórios***. Para tomada de decisão, no entanto, não há necessidade de contatar outros nós pois toda a informação necessária é obtida localmente (para isso cada nó estima o número de nós na rede) constituindo uma tomada de decisão ***Local***. Como o algoritmo funciona em ciclos mantendo continuamente a atualização do tamanho dos segmentos do espaço de endereçamento, essa técnica foi considerada em relação a dimensão de dinamicidade como uma técnica ***Dinâmica Ativa***.

4.3.4 Effective load balancing of peer-to-peer systems

Nesse trabalho a estrutura de endereçamento não é utilizada para balancear a carga, na verdade, os autores não comentam questões de endereçamento que é uma questão fundamental em redes baseadas em DHTs, essa técnica poderia, por exemplo, da forma como foi apresentada, ser aplicada a redes par-a-par não estruturadas. A técnica é proposta para atuar em um ambiente distribuído de larga escala, definindo o ambiente de aplicação como ***WAN***. A idéia desse trabalho é balancear a distribuição de itens de dados entre os nós da rede, definindo a dimensão de objetivo do balanceamento como ***Recursos de um Nó***. Não consideram a idéia de servidores virtuais, assim, a técnica é classificada como ***Simples*** em relação ao tipo dos nós. Para simular itens de dados que são mais requisitados que outros é utilizada uma distribuição de itens que reflete a popularidade dos itens. Isso define a dimensão de distribuição de carga como sendo ***Não Uniforme***.

Embora os detalhes da comunicação não tenham sido mencionados, de forma geral, a comunicação para transferência de informações é trocada entre nós vizinhos que são separados em agrupamentos, além disso, cada agrupamento possui um líder e as informações de carga são trocadas também entre líderes vizinhos. Esses agrupamentos são formados de forma que máquinas que estejam numa mesma rede local são consideradas participantes de um mesmo cluster. Esse padrão de comunicação se encaixa na taxonomia criada na categoria ***Vizinhança Física***. Já a tomada de decisão sobre transferência de itens é definida pelas informações coletadas dos nós vizinhos. Assim, o trabalho foi classificado em relação a tomada de decisão dentro da categoria ***Vizinhança***. Não são mencionadas questões sobre a heterogeneidade da rede, assim, o algoritmo será considerado ***Homogêneo*** em relação à heterogeneidade. Os lideres dos agrupamentos monitoram as informações de carga e sempre que é detectada um diferença maior que um determinado fator parametrizado pelo algoritmo acon-

tecem então as transferências de itens. Esse comportamento de monitoração das informações define a dinamicidade da técnica, que nesse caso, é considerada ***Dinâmica Ativo***.

4.3.5 A scheme for load balancing in heterogeneous distributed hash tables

O trabalho apresenta uma técnica para redes par-a-par estruturadas de larga escala, portanto, o ambiente de aplicação é definido como ***WAN***. O algoritmo utiliza as entradas e saídas de nós da rede para aplicar a técnica de balanceamento, portanto, o algoritmo é considerado ***Dinâmico Passivo*** em relação a dimensão de dinamicidade. Foram apresentados dois algoritmos um para ambientes onde as capacidades dos nós são iguais, ou seja, apresentam uma distribuição homogênea e outro, que é uma modificação do primeiro, para ambientes nos quais as capacidades das máquinas são diferentes, vamos apenas considerar aqui este último, uma vez que ele também é aplicável a ambientes homogêneos. A heterogeneidade, então, é definida como ***Heterogênea***.

A divisão do espaço de endereçamento é feita entre servidores virtuais. Definindo assim, a dimensão de tipo de nó como ***Virtuais***. Quando um novo nó vai entrar na rede são enviadas buscas para um conjunto aleatório de servidores virtuais. O maior dos segmentos retornados é então dividido ao meio e metade é associada ao novo nó. Essa característica define a dimensão de objetivo do balanceamento como ***Espaço de Endereçamento***. Da mesma forma que define também o padrão de comunicação como ***Conjunto Aleatório de Nós***. A única informação levada em consideração durante o processo de tomada de decisão é o tamanho do segmento que é obtida pela consulta aleatória aos nós da rede, o que constitui uma tomada de decisão por consulta ***Aleatória***. Como os autores não comentam o tipo de distribuição de carga, a técnica foi considerada em relação a essa distribuição dentro da categoria ***Uniforme***, já que ela trabalha com ao idéia de modificação da estrutura dos nós.

4.3.6 Distributed, secure load balancing with skew, heterogeneity, and churn

Essa técnica de balanceamento de carga foi definida para trabalhar com redes de larga escala, caracterizando, portanto, um ambiente de aplicação do tipo ***WAN***. Nessa técnica os nós da rede armazenam informações sobre o seu nível de carga em relação a algum recurso. Se a carga de um nó atingir um determinado limiar o nó é considerado sobrecarregado. Essas informações são utilizadas para definir onde um novo nó deverá entrar. Por não se preocupar com a distribuição dos nós dentro do espaço de endereçamento essa técnica é considerada, em relação ao objetivo do balanceamento, dentro da categoria ***Recursos de um Nó***. São utilizados servidores virtuais para distribuição da carga, definindo assim o tipo de nó dentro

da categoria ***Virtuais***. Ao entrar na rede é contatado um conjunto de nós e com base em uma função gulosa que minimiza a carga é escolhido o local onde o novo nó será inserido. O padrão de comunicação é então definido por meio da utilização da primitiva de busca, caindo dentro da categoria ***Conjunto Aleatório de Nós***.

As informações coletadas nesses nós contactados são utilizadas para a tomada de decisão feita pela função de otimização. Esse comportamento define a tomada de decisão dentro da como um processo ***Aleatório*** de consulta. As distribuições de carga e a heterogeneidade são classificadas respectivamente para esse algoritmo como ***Não Uniforme*** por meio de uma distribuição *zipf* que reflete popularidade e ***Heterogênea*** por meio de uma distribuição *pareto*. Por ser desenvolvida para um ambiente com um grande número de entrada e saída de nós, essa técnica é classificada como ***Dinâmica Passiva*** em relação à dinamicidade.

4.3.7 Multitifaceted simultaneous load balancing in DHT-based P2P systems

Desenvolvido para ambientes de redes par-a-par estruturadas o algoritmo foi classificado como ***WAN*** em relação ao ambiente de aplicação. Essa técnica não utiliza servidores virtuais e portanto se encaixa, em relação ao tipo de nó, na categoria ***Simples***. É apresentado um mecanismo de construção de uma estrutura distribuída que se adapta à distribuição de carga, portanto, a dimensão de distribuição de carga foi definida dentro da categoria ***Não Uniforme***. Entretanto, em relação a heterogeneidade nada é comentado e portanto o algoritmo foi considerado ***Homogêneo***. A estrutura criada é dinamicamente atualizada com a entrada e saída de nós da rede, portanto, o algoritmo é ***Dinâmico Passivo***. A estrutura é criada de forma a particionar o espaço de identificadores de acordo com a distribuição de carga e, por isso, o objetivo do balanceamento é o ***Espaço de Endereçamento***.

A atualização é feita com base em pesquisa ***Aleatória*** de informações de carga dos nós que é utilizada para tomada de decisão. Alguns nós são escolhidos aleatoriamente para realização da amostragem, portanto padrão de comunicação se dá por meio de um ***Conjunto Aleatório de Nós***.

4.3.8 Efficient, proximity-aware load balancing for DHT-based P2P systems

O algoritmo é desenvolvido para grandes sistemas distribuídos e é classificada dentro da categoria ***WAN*** em relação ao ambiente de aplicação. São utilizados servidores virtuais para transferência de carga dentro do sistema o que determina a categoria ***Virtuais*** para a dimensão tipo de nó. Para obter informações sobre a carga do sistema é construída uma estrutura

de dados distribuída em forma de árvore. Nessa árvore, os nós folhas são associados aos nós da rede. As informações de carga são então agrupadas e enviadas para os nós superiores da árvore de forma a acumular informações e possibilitar a transferência de carga entre servidores virtuais. Para formação da árvore são utilizadas técnicas de obtenção de proximidade entre os nós. Assim a comunicação ocorre entre vizinhos próximos fisicamente o que define o padrão de comunicação como ***Vizinhança Física***. Além disso a reunião de informações caracteriza uma tomada de decisão apoiada por consulta ***Aleatória*** às informações de carga. As informações são recolhidas em relação aos recursos dos nós, não há intenção de modificar a estrutura de endereçamento diretamente, o que caracteriza a dimensão objetivo do balanceamento como ***Recursos de um Nó***.

As distribuições de carga utilizadas na avaliação do trabalho são as distribuições *gaussiana* e *pareto* que definem a categoria ***Não Uniforme*** para a dimensão de distribuição de carga. Os servidores virtuais são criados proporcionalmente à capacidade dos nós. Foram considerados em relação à heterogeneidade os tipos real *Gnutella* e sintética *zipf*. Essas distribuições caracterizam o algoritmo dentro da categoria ***Heterogêneo*** em relação a heterogeneidade da rede. O algoritmo considera que durante a formação da estrutura de balanceamento (a árvore) o sistema é estável, além disso, não comenta como a árvore será reconstruída com as saídas e entradas de nós na rede. Por essa razão o algoritmo é considerado ***Estático*** em relação a dimensão de dinamicidade.

4.3.9 Simple load balancing for distributed hash table

Nesse algoritmo não são utilizados servidores virtuais o que determina a categoria ***Simples*** para a dimensão de tipo de nó. O trabalho utiliza uma modificação da estrutura de endereçamento para realiza o balanceamento de carga, entretanto essa modificação é um pouco da diferente pois são utilizadas diversas funções *hash* para mapear os objetos na rede. O algoritmo foi então classificado em relação ao objetivo do balanceamento como ***Espaço de Endereçamento***. O algoritmo considera uma distribuição de carga de itens de dados uniforme, sendo então classificada dentro da categoria ***Uniforme*** em relação a distribuição de carga. Como foi desenvolvido para ambientes altamente distribuídos e de larga escala o algoritmo é classificado dentro da categoria ***WAN*** em relação ao ambiente de aplicação.

O padrão de comunicação é determinado pelo quantidade de funções hash utilizadas sendo feita uma busca para cada função, o que constitui um padrão de comunicação dentro da categoria ***Conjunto Aleatório de Nós***. Após essas buscas são recolhidas informações sobre os nós pesquisados e o que possuir menos itens no momento, ou seja, o que possuir o menor segmento. Esse tipo de tomada de decisão é classificada dentro da categoria ***Aleatório***. Os nós da rede são considerados todos iguais em relação às suas capacidades, determinando assim um algoritmo ***Homogêneo*** para a dimensão de heterogeneidade. O algoritmo não considera questões de entrada e saída de nós da rede nem define formas de redistribuição de carga entre

os nós, por isso foi classificado como ***Estático*** em relação à dinamicidade.

4.3.10 A thermal-dissipation-based approach for balancing data load in DHTs

Nesse trabalho é destinado a redes que funcionam em um ambiente de ***WAN*** sendo essa então a sua classificação como ambiente de aplicação. São utilizados servidores virtuais, o que define a dimensão tipo de nó como ***Virtuais***, que são utilizados para gerenciar os segmentos do espaço de endereçamento. São utilizados mais de um nó para responsabilidade de um determinado segmento e dentro desse segmento os itens de dados são replicados entre os nós. Assim, o padrão de comunicação é determinado por transferência de itens entre os nós vizinhos no espaço de endereçamento determinando o algoritmo dentro da categoria ***Vizinhança Lógica***. A distribuição de carga entre os nós é considerada ***Uniforme*** pelos autores. A heterogeneidade entretanto é não uniforme, de forma que os nós da rede podem ter servidores virtuais proporcionalmente a sua capacidade, portanto o algoritmo é considerado ***Heterogêneo*** em relação à dimensão de heterogeneidade.

A tomada de decisão é realizada com base nas informações conseguidas com os vizinhos por meio de ***Vizinhança***. Por modificar a estrutura para que a carga se acomode equilibradamente entre os nós o algoritmo foi classificado dentro da categoria de ***Espaço de Endereçamento*** em relação à dimensão objetivo do balanceamento. O algoritmo foi considerado ***Dinâmico Passivo*** por realizar adaptações na estrutura de endereçamento sempre que um nó entre ou sai da rede.

4.3.11 Load balancing in hypercubic DHTs with heterogeneous processors

Esse algoritmo tem por objetivo construir uma distribuição dos nós de forma justa dentro do espaço de endereçamento. O objetivo do balanceamento é portanto o ***Espaço de endereçamento***. Essa manipulação do espaço de identificadores é realizado por um algoritmo *off line* antes das máquinas entrarem na rede, portanto, o algoritmo não precisa saber quantos nós participarão da rede antes da entrada deles, o que caracteriza uma tomada de decisão baseada em informação ***Global*** e uma dinamicidade dentro da categoria ***Estática***. Não há comunicação entre os nós pois não balanceamento de carga com participação dos nós. O padrão de comunicação será considerado então ***global*** pois o algoritmo só se aplica a um ambiente centralizado, uma vez que é um algoritmo *off line*. Da mesma forma foi considerado o ambiente de aplicação como ***LAN***.

O algoritmo usa uma variação do algoritmo de Huffman para gerar a distribuição de

identificadores, de forma que não são utilizados servidores virtuais e, portanto, os tipos dos nós estão na categoria ***Simples***. As capacidades dos nós são consideradas para criação da distribuição de identificadores o que caracteriza a dimensão de heterogeneidade como ***Heterogênea***. A distribuição de carga entretanto não é comentada e será considerada aqui como ***Uniforme***.

4.3.12 Simple efficient load balancing algorithms for peer-to-peer systems

Essas técnicas foram desenvolvidas para ambientes distribuídos e se encaixam na categoria ***WAN*** para a dimensão ambiente de aplicação. Não utilizam servidores virtuais, estando portando dentro da categoria ***Simples*** na dimensão de tipo de nós. São descritos dois algoritmos, um para distribuição equilibrada dos identificadores e outro para transferência de itens entre os nós da rede. Entretanto, os dois algoritmos se baseiam na modificação da estrutura da rede para atingir o balanceamento e portanto estão classificados dentro da categoria ***Espaço de Endereçamento*** em relação ao objetivo do balanceamento. Os Algoritmos não comentam questões de distribuição de carga e de heterogeneidade e por isso essas dimensões serão classificadas respectivamente como ***Uniforme*** e ***Homogênea***.

No primeiro algoritmo cada nó possui um conjunto de potenciais nós, que são identificadores da rede que podem ser utilizados. São utilizadas informações do próprio nó para decidir qual identificador ficará ativo, o que define a dimensão tomada de decisão dentro da categoria ***Local***. A mudança de identificadores altera o sistema apenas entre os vizinhos de um nó caracterizando um padrão de comunicação por ***Vizinhança Lógica***. Essa mudança podo ocorrer sempre que entrada ou saída da rede definindo assim o algoritmo como ***Dinâmico Passivo*** em relação à dinamicidade.

O segundo algoritmo utiliza um esquema de seleção aleatória de nós para realizar o balanceamento portanto o padrão de comunicação está na categoria ***Conjunto Aleatório de Nós*** e essas informações coletadas são utilizadas então para tomada de decisão, o que indica a categoria ***Aleatório*** para essa dimensão da taxonomia. Essa consulta a outros nós é realizada periodicamente para manter o sistema balanceado, portanto o algoritmo, em relação à dinamicidade está dentro da categoria ***Dinâmico Ativo***.

4.3.13 Load balancing in structured P2P systems

Esse trabalho foi desenvolvido para redes P2P estruturadas de grande escala, portanto é classificado dentro categoria ***WAN*** em relação ao ambiente de aplicação. O algoritmo executa as operações de balanceamento periodicamente, entretanto não lida com saídas e entradas de nós na rede, foi classificado, dessa forma, dentro da categoria ***Dinâmico Ativo*** em relação a

dimensão de dinamicidade. Utiliza servidores virtuais para realizar a transferência de carga, definindo o tipo de nó utilizado como ***Virtuais***.

São utilizadas informações de capacidade dos nós definindo a dimensão de heterogeneidade com a categoria ***Heterogêneo***. Da mesma forma a distribuição de carga considerada pelo algoritmo está dentro da categoria ***Não Uniforme***. Não se preocupando com o endereçamento, mas apenas com a transferência de servidores virtuais o algoritmo é considerado, em relação a dimensão de objetivo do balanceamento dentro da categoria ***Recursos de Nó***. Devido à forma como são realizadas a coleta de informações e a transferência entre os nós a dimensão de tomada de decisão foi classificada dentro da categoria ***Aleatório*** e a dimensão padrão de comunicação dentro da categoria ***Conjunto Aleatório de Nós***.

4.3.14 Load balancing in dynamic structured P2P systems

O algoritmo foi desenvolvido para um ambiente de aplicação ***WAN***, ou seja, redes de grande escala. É um algoritmo que possui funcionamento periódico e, além disso, aceita entradas e saídas de nós na rede, portanto possui sua dimensão de dinamicidade classificada com a categoria ***Dinâmica Ativa***. São utilizados servidores virtuais para transferir a carga do sistema de uma máquina para outra, assim, os nós são do tipo ***Virtuais***. Para criação dos servidores virtuais são utilizadas informações sobre a capacidade dos nós o que determina que o algoritmo é capaz de lidar com uma heterogeneidade ***Heterogênea***. O algoritmo considera uma distribuição ***Não Uniforme*** de carga, definindo assim essa dimensão. Existe apenas a transferência de servidores para balancear a carga e nenhuma modificação da estrutura da rede é requerida, portanto a técnica apresenta um objetivo de balanceamento dentro da categoria ***Recursos de um Nó***.

O padrão de comunicação é definido pela seleção aleatória do diretório onde será armazenada as informações de carga, caracterizando assim um algoritmo de comunicação com um ***Conjunto Aleatório de Nós***. As informações armazenadas nos diretórios são então utilizadas para decidir quais servidores virtuais serão transferidos. Essa tomada de decisão é então conseguida devido à consulta ***Aleatória*** realizada nos nós do sistema.

4.3.15 Hash-based proximity clustering for load balancing in heterogeneous DHT networks

O trabalho apresenta dois algoritmos para balanceamento de carga em DHTs de aplicação geral, assim o ambiente de aplicação de ambos os algoritmos é ***WAN***. Os algoritmos utilizam como idéia principal a criação de uma rede auxiliar formada supernós (nós com maior capacidade) que são responsáveis por coletar informações e distribuir a carga entre os nós. O objetivo principal do balanceamento são os ***Recursos de um Nó*** e não se preocupam com

a modificação da estrutura de endereçamento. Consideram distribuição de carga ***Não Uniforme*** entre os nós do sistema. A forma como a carga é transferida não é detalhada, mas não são utilizados servidores virtuais, assim, os tipos de nós são classificados como ***Simples***. Os supernós coletam informações regionais para a tomada de decisão o que define essa dimensão da taxonomia dentro da categoria ***Vizinhança***.

Os dois algoritmos atualizam periodicamente as informações de carga, caracterizando para os dois algoritmos como ***Dinâmicos Ativos*** em relação a dinamicidade das técnicas. A heterogeneidade da rede segue uma distribuição *pareto* e por isso nessa dimensão os algoritmos são definidos como ***Heterogêneos***. Os algoritmos funcionam de forma diferente em relação ao padrão de comunicação, um deles utiliza informações de proximidade física e é por isso classificado dentro da categoria ***Vizinhança Física*** e o outro utiliza apenas proximidade dentro do espaço de endereçamento e por isso é classificado dentro da categoria ***Vizinhança Lógica***.

4.4 Relacionamento com taxonomias de balanceamento de carga

Casavant e Kuhl [4] apresentou um trabalho que descreve uma taxonomia para a área de distribuição de recursos em sistemas distribuídos de propósito geral. Essa taxonomia, assim como a aqui apresentada, tem por objetivo criar um conjunto de termos aceitáveis dentro da área estudada, de forma a permitir a comparação de trabalhos antigos e futuros. Além disso, por meio dos relacionamentos entre os trabalhos, descritos pela classificação taxonômica, outro objetivo do trabalho era, também em semelhança com este trabalho, identificar possíveis áreas para pesquisas futuras. A taxonomia apresentada por Casavant e Kuhl [4] é constituída de duas formas de classificação, uma classificação hierárquica e uma classificação plana, que serão vistas a seguir:

Classificação hierárquica foi composta com o objetivo de facilitar a comparação entre os diversos trabalhos da área, bem como trabalhos passados e trabalhos futuros. que pode ser visualizada na Figura 4.2.

Classificação plana foi composta por termos que classificam os trabalhos de forma independente dos outros termos de classificação. Uma vez que esse estilo de classificação se assemelha mais ao proposto neste trabalho, vamos detalhar melhor a classificação plana:

- *Adaptativo versus não adaptativo:* Classifica o trabalho de gerência de distribuição de recursos em relação sua capacidade de se adaptar dinamicamente a novas situações de acordo com o comportamento do sistema. Esta característica pode

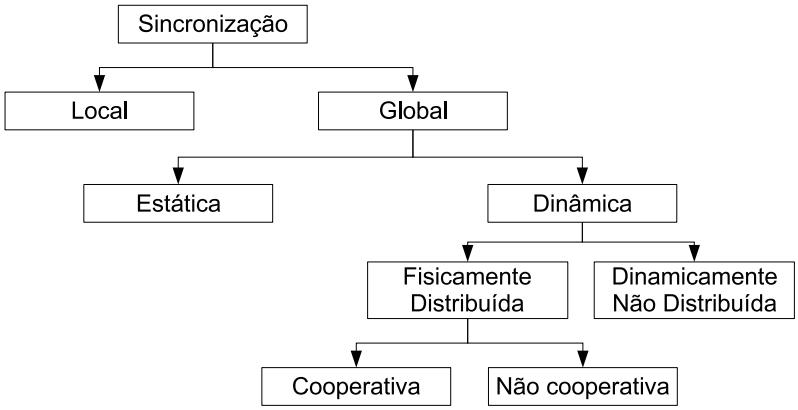


Figura 4.2: Taxonomia criada para a área de gerenciamento de recursos de sistemas distribuídos de propósito geral.

ser mapeada na característica de *dinamicidade* proposta neste trabalho, pois as duas possuem os mesmos objetivos de classificação.

- *Balanceamento de carga:* Como a taxonomia proposta abrange a gerência de recursos de forma geral, o esquema de balanceamento de carga é visto como mais uma forma de classificação. Nesse caso uma série de políticas de balanceamento de carga são então consideradas. Políticas como informações para *tomada de decisão* são descritas na taxonomia descrita aqui. Outras características como a política de transferência de tarefas não foram consideradas na taxonomia para DHT's pois não foi observada a necessidade de separar os algoritmos nesse nível de detalhe.
- *Ligação:* Consiste em classificar as técnicas de acordo com a forma de assinalamento de tarefas aos nós da rede. No caso de redes baseadas em tabelas *hash* distribuídas a forma de assinalamento de tarefas/objetos já é definida pela estrutura da própria rede. Dessa forma não faz sentido uma classificação da técnica segundo esse critério.
- *Probabilístico:* Classifica sistemas que possuem assinalamento de tarefas baseado em informações obtidas por seleção randômica dos nós e dos processos a serem assinalados. Possui relação direta com a categoria *Tomada de decisão* apresentada aqui, pois as duas categorias procura classificar a forma pela qual uma decisão de reassinalamento ou transferências de chaves é tomada.
- *Assinalamento estático versus dinâmico* Ainda em relação ao assinalamento essa classificação procura separar os sistemas que, uma vez que tenham tomada uma decisão de para onde e quanto mandar uma tarefa, não mais modifique essa decisão, esse estado reflete um sistema estático. Se há a possibilidade de interromper uma tarefa de acordo com a análise de comportamento do sistema esse sistema é dinâmico. Na taxonomia aqui proposta não preocupação com esse nível de detalhe, uma vez que o objetivo é refletir características importantes das redes DHTs.

Embora exista uma relação entre as dimensões das duas taxonomias, a descrita em Casavant e Kuhl [4] e a aqui proposta, neste trabalho procuramos atingir uma classificação que represente mais o universo das redes DHTs. Como toda rede DHT também é um sistema distribuído, pode-se utilizar a taxonomia de Casavant para obter uma classificação dos trabalhos de balanceamento de carga em DHTs, entretanto, uma taxonomia específica para área traz benefícios à classificação. Nada impede também que as duas taxonomias sejam utilizadas em conjunto e com isso se tenha tanto uma visão geral como uma visão mais focada da área de balanceamento de carga em DHTs.

Capítulo 5

Conclusão

A partir do estudo de diversos trabalhos da área de balanceamento de carga em DHTs foi possível chegar uma uma taxonomia para essa área. Essa taxonomia leva em consideração diversos aspectos dos algoritmos. Com ela é possível classificar os trabalhos da área em relação à capacidade de lidar com mudanças de carga na rede, seja por entrada e saída de nós ou por mudança da distribuição de carga. É possível comparar os trabalhos em relação ao nível de conhecimento necessário para tomada de decisão sobre transferências de carga.

Outra característica importante é a possibilidade de comparar os algoritmos quanto ao padrão de comunicação utilizado. O tipo de distribuição de carga que é suportada pelos algoritmos também pode ser obtida da taxonomia, bem como a heterogeneidade da rede, informando se determinado algoritmo leva ou não essa informação em consideração. A estratégia de balanceamento utilizada pelos algoritmos também pode ser classificada dentro da taxonomia como sendo uma estratégia de balanceamento do espaço de endereçamento ou dos recursos utilizados por um nó. O ambiente de aplicação dos algoritmos pode, dentro da taxonomia, ser classificado como WAN ou LAN abrangendo sistemas para larga escala ou não. Por fim, o tipo de nó utilizado na DHT para balancear a carga é classificado como simples ou virtual, o que indica a flexibilidade e o custo de manutenção desses nós.

A taxonomia desenvolvida foi aplicada aos diversos trabalhos analisados. A partir da utilização dessa taxonomia é possível comparar trabalhos atuais, passados e futuros comparando as diversas características que diferenciam esses trabalhos. Isso permite um maior entendimento dos algoritmos dessa área e uma melhor organização do conhecimento. Consideramos portanto que a taxonomia cumpriu seu papel como uma boa forma de classificar os trabalhos da área de balanceamento de carga permitindo uma melhor organização do conhecimento dessa área.

5.1 Distribuição de tarefas no ambiente distribuído Formigueiro

O trabalho aqui desenvolvido tem como principal motivação reunir conhecimento suficiente para desenvolver e implantar um sistema de distribuição de tarefas entre as máquinas do ambiente distribuído Formigueiro. Esse ambiente, como descrito na seção 1.1, serve de base para a execução de aplicações distribuídas de mineração de dados. O Formigueiro tem características especiais por ser um ambiente de agrupamento em que cada máquina recebe tarefas de outras máquinas segundo um esquema de nomeação de destino por rótulos, chamado *labeled-stream*. As aplicações utilizam, então, essa estrutura para distribuir corretamente as tarefas pelo sistema.

Acreditamos que a utilização de uma camada que implemente uma DHT para definir o mapeamento de nós entre o Formigueiro e as aplicações pode melhorar o desempenho do sistema por meio da redução de comunicação face a uma reconfiguração. Acreditamos também que é possível desenvolver um algoritmo para distribuição de tarefas que leve em consideração a carga do sistema. A seguir indicaremos um conjunto de características, definido com base nas dimensões da taxonomia criada, que poderiam ser consideradas para o desenvolvimento de tal algoritmo:

Ambiente por ser um ambiente centralizado o Formigueiro possui as mesmas características de um ambiente ***LAN***. Assim, acreditamos que um algoritmo de distribuição de tarefas possa tirar proveito de técnicas utilizadas em algoritmos de balanceamento de carga desenvolvidos para esses ambientes. Nesse caso um bom exemplo de algoritmo é o de Rufino *et alii* [27].

Objetivo do Balanceamento Um dos maiores problemas das aplicações que são executadas no Formigueiro se relaciona ao tempo de execução de determinadas tarefas, ou seja, o balanceamento de uso CPU. Por isso, acreditamos que os algoritmos desenvolvidos com o objetivo de balancear os ***Recursos de um nó*** possuem características que podem auxiliar no processo de desenvolvimento do algoritmo. Exemplos de trabalhos classificados nessa categoria são descritos em Aberer *et alii* [1] e Rao *et alii* [23] entre outros.

Tomada de Decisão Em relação a essa dimensão, numa análise inicial, é possível escolher diferentes categorias dependendo do nível de escalabilidade pretendida para o algoritmo de distribuição de tarefas. Por ser um ambiente de agrupamento é possível ter informações globais sobre os sistema. Entretanto, a coleta de informações para tomada de decisão influencia no padrão de comunicação do algoritmo e nesse caso teríamos uma comunicação global para obter tais informações. Entretanto é possível aplicar outros tipos de tomada de decisão mesmo em um ambiente centralizado. É possível, por exem-

plo, utilizar informações coletadas localmente ou na vizinhança. Entretanto, o impacto dessa decisão teria que ser avaliada em função do restante do sistema.

Comunicação Pelos mesmos motivos citados acima só é possível indicar a melhor forma de comunicação com um estudo mais detalhado dos requisitos do Formigueiro. Embora o algoritmo centralizado estudado neste trabalho [27] utilize um padrão de comunicação global, acreditamos que é suficiente um padrão de comunicação menos custoso como os baseados em ***Vizinhança***, ***Conjunto de Nós*** ou mesmo a utilização de um padrão de comunicação ***Local*** de carga. Os algoritmos de balanceamento aqui estudados utilizam, na sua maioria, uma das três técnicas citadas anteriormente apresentando resultados interessantes e apenas dois trabalhos apresentam um padrão de comunicação ***Global***.

Dinamicidade Como discutido anteriormente, o principal recurso a ser balanceado no ambiente Formigueiro é a utilização de CPU. Uma distribuição de tarefas que considerasse apenas as entradas e saídas de máquinas da rede (***Dinâmico passivo***) por si só já melhoraria o funcionamento do sistema. Entretanto, devido à dificuldade de se medir o tempo de execução de uma tarefa seria interessante um algoritmo ***Dinâmico ativo***, ou seja, que monitorasse periodicamente a carga do sistema, de forma a transferir tarefas que estivessem exigindo muito processamento para máquinas mais capacitadas. Nesse caso existe, como visto na seção 2.1.1, o problema de ter que transferir o estado das tarefas junto com elas para a máquina de destino. Um monitoramento ativo também permite que novas tarefas sejam desviadas do seu destino normal se a máquina de destino estiver sobrecarregada. Exemplos de algoritmos que possuem dinamicidade ativa são encontrados, entre outros, em Godfrey *et alii* [11, 10] e em *et alii* [2].

Distribuição de carga Devido à natureza das aplicações distribuídas de mineração de dados, algumas tarefas podem ser mais demoradas que outras. Isso significa que uma tarefa pode exigir mais processamento da máquina que outras. Assim, um algoritmo de distribuição de tarefas deve considerar essa uma carga ***não uniforme*** e distribuir as tarefas de forma a balancear a carga pelo sistema. Quase metade dos algoritmos analisados consideram essa característica em seus algoritmos, entre eles podemos destacar os trabalhos de Zhu e Hu [36] e de Shen e Xu [29].

Heterogeneidade O Formigueiro foi construído para trabalhar com um ambiente distribuído heterogêneo, ou seja, um ambiente no qual a capacidade das máquinas podem ser diferentes. Dessa forma, é interessante que esse tipo de informação de heterogeneidade seja considerada durante o desenvolvimento de um algoritmo para o Formigueiro. Dentre os algoritmos analisados mais da metade consideram esse tipo de informações em seus algoritmos e normalmente o que é feito é distribuir as tarefas para as máquinas de forma que elas recebam carga proporcional à sua capacidade. Como exemplos de algoritmos que utilizam informações de heterogeneidade podemos citar os trabalhos de Godfrey *et alii* [11] e o trabalho de Liu e Adler *et alii* [17] entre outros.

Tipo de Nó A escolha do tipo de nó influencia significativamente na flexibilidade da implementação da transferência de carga. Definindo-se que a carga será associada a um servidor virtual estamos definindo uma nova camada no sistema de distribuição de tarefas. Assim, é possível desenvolver com maior facilidade um sistema de distribuição de tarefas e balanceamento de carga modulares. Tal característica permitiria mais facilmente que diferentes abordagens de balanceamento de carga fossem utilizadas dependendo da aplicação que estivesse sendo executada. Vários dos trabalhos analisados aqui utilizam a abstração de servidores virtuais, entre eles podemos citar os trabalhos de Giakkoupis e Hadzilacos [9] e de Rieche *et alii* [25].

A criação de uma camada DHT para distribuir as tarefas mantendo a carga do sistema balanceada é um grande desafio e aqui apenas indicamos um possível conjunto de características interessantes. O desenvolvimento de um algoritmo de distribuição de tarefas para o ambiente Formigueiro requer uma investigação mais profunda desse ambiente e uma boa definição dos requisitos desse sistema. Acreditamos que a taxonomia criada pode ajudar durante o esse processo, pois permite uma organização adequada do conhecimento construído recentemente nessa área.

5.2 Trabalhos Futuros

A partir de um melhor entendimento da área de balanceamento de carga e de uma maior facilidade em comparar as características dos diversos trabalhos dessa área é possível identificar novas soluções de balanceamento em subáreas que se mostrem mais carentes de novos algoritmos por não terem sido ainda bem exploradas. Além disso, permite que determinadas características desejáveis a um algoritmo possam ser facilmente obtidas na taxonomia e assim permite uma construção mais modular do algoritmo, restando depois a preocupação com os detalhes de implementação em relação às categorias da taxonomia escolhidas para montar o algoritmo.

Um possível trabalho, em especial, é a definição e a implementação de uma técnica de balanceamento de carga que utilize características de tabelas *hash* distribuídas para melhorar o desempenho do sistema *labeled-stream* do ambiente distribuído Formigueiro do laboratório e-Speed no DCC.

Referências Bibliográficas

[1] Karl Aberer, Anwitaman Datta, and Manfred Hauswirth. Multifaceted simultaneous load balancing in dht-based P2P systems: A new game with old balls and bins, 2004.

[2] Marcin Bienkowski, Miroslaw Korzeniowski, and Friedhelm Meyer auf der Heid. Dynamic load balancing in distributed hash tables. In *4th International Workshop, IPTPS 2005, Ithaca, NY, USA*, Fevereiro 2005.

[3] J. Byers, J. Considine, and M. Mitzenmacher. Simple load balancing for distributed hash tables, 2002.

[4] Thomas Casevant and Jon Kuhl. A taxonomy of scheduling in general-purpose distributed computing systems. In *IEEE Transactions of Software Engineering*, volume 14-2, pages 141–154, Fevereiro 1988.

[5] Andre Ribeiro da Silva, Helio Marcos Paz de Almeida, Tiago Macambira, Dorgival Olavo Guedes, Wagner Meira Jr., and Renato Antonio C. Ferreira. Hash consistente como uma ferramenta para distribuição de tarefas em sistemas distribuídos reconfiguráveis. In *WSCAD - VI Workshop em sistemas computacionais de alto desempenho*, Rio de Janeiro - RJ, Outubro 2005.

[6] Carlos E. de M. Bicudo. Taxonomia. *Biota Neotropica - ISSN-1676-0603*, 4(1):Editorial, 2004.

[7] Renato Ferreira, Wagner Meira Jr., Dorgival Olavo Guedes, and Lúcia Drummond. Anthill: A scalable run-time environment for data mining applications. In *17th International Symposium on Computer Architecture and High Performance Computing*, Rio de Janeiro - RJ - Brasil, Outubro 2005.

[8] M. Flynn. Some computer organizations and their effectiveness. In *IEEE - Transactions on Computers*, volume C-21, pages 948–960, Setembro 1972.

[9] George Giakkoupis and Vassos Hadzilacos. A scheme for load balancing in heterogenous distributed hash tables. In *PODC '05: Proceedings of the twenty-fourth annual ACM SIGACT-SIGOPS symposium on Principles of distributed computing*, pages 302–311, New York, NY, USA, 2005. ACM Press.

[10] Brighten Godfrey, Karthik Lakshminarayanan, Sonesh Surana, Richard Karp, and Ion Stoica. Load balancing in dynamic structured P2P systems. In *Proc. IEEE INFOCOM*, Hong Kong, 2004.

[11] P. Brighten Godfrey and Ion Stoica. Heterogeneity and load balance in distributed hash tables. In *INFOCOM'05*, Fevereiro 2005.

[12] D. Gupta and P. Bepari. Load sharing in distributed systems. In *Proceedings of the National Workshop on Distributed Computing*, Janeiro 1999.

[13] K. Hildrum, J. Kubiatowicz, S. Rao, and B. Zhao. Distributed object location in a dynamic network, 2002.

[14] D. Karger and M. Ruhl. Simple efficient load balancing algorithms for peer-to-peer systems. In *ACM Symposium on Parallelism in Algorithms and Architectures*, Junho 2004.

[15] David Karger, Eric Lehman, Tom Leighton, Mathhew Levine, Daniel Lewin, and Rina Panigrahy. Consistent hashing and random trees: Distributed caching protocols for relieving hot spots on the world wide web. In *ACM Symposium on Theory of Computing*, pages 654–663, Maio 1997.

[16] Jonathan Ledlie and Margo Seltzer. Distributed, secure load balancing with skew, heterogeneity, and churn. In *IEEE INFOCOM'05*, Março 2005.

[17] Junning Liu and Micah Adler. Load balancing in hypercubic distributed hash tables with heterogeneous processors. In *ESA04*, volume 3221 of *LNCS*, pages 496–507. Springer, 2004.

[18] P. Maymounkov and D. Mazieres. Kademlia: A peer-to-peer information system based on the xor metric. 2002.

[19] Dejan S. Milojicic, Vana Kalogeraki, Rajan Lukose, Kiran Nagaraja, Jim Pruyne, Bruno Richard, Sami Rollins, and Zhichen Xu. Peer-to-peer computing. In *Technical Report HPL-2002-57*. HP Lab, 2002.

[20] Anirban Mondal, Kazuo Goda, and Masaru Kitsuregawa. Effective load-balancing of peer-to-peer systems. In *Proceedings of Data Engineering Workshop - DEWS'03*, Março 2003.

[21] Christos H. Papadimitriou and Mihalis Yannakakis. Towards an architecture-independent analysis of parallel algorithms. *SIAM J. Comput.*, 19(2):322–328, 1990.

[22] C. Greg Plaxton, Rajmohan Rajaraman, and Andrea W. Richa. Accessing nearby copies of replicated objects in a distributed environment. In *ACM Symposium on Parallel Algorithms and Architectures*, pages 311–320, 1997.

[23] Ananth Rao, Karthik Lakshminarayanan, Sonesh Surana, Richard Karp, and Ion Stoica. Load balancing in structured P2P systems. In *2nd International Workshop on Peer-to-Peer Systems (IPTPS '03)*, Fevereiro 2003.

[24] Sylvia Ratnasamy, Paul Francis, Mark Handley, Richard Karp, and Scott Shenker. A scalable content addressable network. Technical Report TR-00-010, Saarland University — Department of Computer Science, Berkeley, CA, 2000.

[25] Simon Rieche, Leo Petrak, and Klaus Wehrle. A thermal-dissipation-based approach for balancing data load in distributed hash tables. In *29th Annual IEEE International Conference on Local Computer Networks*, pages 15–23. IEEE, Novembro 2004.

[26] Antony Rowstron and Peter Druschel. Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems. *Lecture Notes in Computer Science*, 2218:329–??, 2001.

[27] José Rufino, Albano Alves, José Exposto, and António Pina. Toward a dynamically balanced cluster oriented dht. In *Parallel and Distributed Computing and Networks - PDCN'04 - Innsbruck, Austria*, Fevereiro 2005.

[28] Detlef Schoder and Kai Fischbach. Peer-to-peer prospects. *Commun. ACM*, 46(2):27–29, 2003.

[29] Haiying Shen and Cheng-Zhong Xu. Hash-based proximity clustering for load balancing in heterogeneous dht networks. In *International Parallel and Distributed Processing Symposium*. IEEE, Abril 2006.

[30] Behrooz A. Shirazi, Krishna M. Kavi, and Ali R. Hurson, editors. *Scheduling and Load Balancing in Parallel and Distributed Systems*. IEEE Computer Society Press, Los Alamitos, CA, USA, 1995.

[31] Nirajan G. Shivaratri, Phillip Krueger, and Mukesh Singhal. Load distributing for locally distributed systems. In *IEEE Computer*, volume 25-12, pages 33–45, Dezembro 1992.

[32] Ion Stoica, Robert Morris, David Karger, Frans Kaashoek, and Hari Balakrishnan. Chord: A scalable Peer-To-Peer lookup service for internet applications. In *Proceedings of the 2001 ACM SIGCOMM Conference*, pages 149–160, 2001.

[33] Yung-Terng Wang and Robert J. T. Morris. Load sharing in distributed systems. In *IEEE Transactions on Computers*, volume 34-3, pages 204–216, Março 1984.

[34] Wikimedia. Taxonomia. http://en.wikipedia.org/wiki/Taxonomy, Junho 2006.

[35] Beverly Yang and Hector Garcia-Molina. Designing a super-peer network. Technical Report 2002-13, Stanford University, 2002.

[36] Yingwu Zhu and Yiming Hu. Efficient, proximity-aware load balancing for dht-based P2P systems. In *EEE Trans. Parallel Distrib. Syst.*, volume 16-4, pages 349–361, Abril 2005.

[37] Nivio Ziviani. *Projeto de Algoritmos com Implementação em Pascal e C*, volume 1. Pioneira Thomson, 2ª edition, 2004.

Apêndice A

Classificação dos trabalhos analisados

	Ambiente	Tipo de Nó	Objetivo	Comunicação
rufino05	LAN	Virtuais	Endereçamento	Global
godfrey05	WAN	Virtuais	Endereçamento	Conjunto de Nós
bienkowski05	WAN	Simples	Endereçamento	Conjunto de Nós
mondal03	WAN	Simples	Recursos	Vizinhança Lógica
giakkoupis05	WAN	Virtuais	Endereçamento	Conjunto de Nós
ledlie05	WAN	Virtuais	Recursos	Conjunto de Nós
aberer04	WAN	Simples	Endereçamento	Conjunto de Nós
zhu05	WAN	Virtuais	Recursos	Vizinhança Física
byers03	WAN	Simples	Endereçamento	Conjunto de Nós
rieche04	WAN	Virtuais	Endereçamento	Vizinhança Lógica
liu04	LAN	Simples	Endereçamento	Global
karger04 - 1	WAN	Simples	Endereçamento	Vizinhança Lógica
karger04 - 2	WAN	Simples	Endereçamento	Conjunto de Nós
rao03	WAN	Virtuais	Recursos	Conjunto de Nós
godfrey04	WAN	Virtuais	Recursos	Conjunto de Nós
shen06 - 1	WAN	Simples	Recursos	Vizinhança Física
shen06 - 2	WAN	Simples	Recursos	Vizinhança Lógica

Tabela A.1: Classificação dos trabalhos analisados

	Dinamicidade	**Tomada Decisão**	**Distr. Carga**	**Heterogeneidade**
rufino05	Passivo	Global	Uniforme	Homogênea
godfrey05	Ativo	Aleatório	Uniforme	Heterogênea
bienkowski05	Ativo	Local	Uniforme	Homogênea
mondal03	Ativo	Vizinhança	Não Uniforme	Homogênea
giakkoupis05	Passivo	Aleatório	Uniforme	Heterogênea
ledlie05	Passivo	Aleatório	Não Uniforme	Heterogênea
aberer04	Passivo	Aleatório	Não Uniforme	Homogênea
zhu05	Estático	Aleatório	Não Uniforme	Heterogênea
byers03	Estático	Aleatório	Uniforme	Homogênea
rieche04	Passivo	Vizinhança	Uniforme	Heterogênea
liu04	Estático	Global	Uniforme	Heterogênea
karger04 - 1	Passivo	Local	Uniforme	Homogênea
karger04 - 2	Ativo	Aleatório	Uniforme	Homogênea
rao03	Ativo	Aleatório	Não Uniforme	Heterogênea
godfrey04	Ativo	Aleatório	Não Uniforme	Heterogênea
shen06 - 1	Ativo	Aleatório	Não Uniforme	Heterogênea
shen06 - 2	Ativo	Vizinhança	Não Uniforme	Heterogênea

Tabela A.2: Classificação dos trabalhos analisados - Continuação