Universidade do Minho
Escola de Engenharia
Departamento de Informática

Estratégias de partição para a optimização da descarga distribuída da *Web*

por

José Luis Padrão Exposto

Dissertação apresentada à Universidade do Minho para
a obtenção do grau de Doutor em Informática

Orientador:
Prof. António Manuel Silva Pina
Co-Orientador:
Prof. Joaquim Melo Henriques Macedo

Guimarães
Novembro de 2008

Agradecimentos

Gostaria de agradecer aos meus orientadores, Prof. António Pina e Prof. Joaquim Macedo, pela sua orientação, disponibilidade e paciência no decurso da realização deste trabalho. A orientação deles foi além das questões científicas, tendo contribuído para o enriquecimento da minha formação académica e para o alargamento da minha visão sobre o mundo e a vida.

Um agradecimento especial ao Rufino e ao Albano que acompanharam mais de perto o meu trabalho e com quem pude trocar impressões e debater importantes questões técnicas. Ao Zé Adriano pelos votos de confiança e a moral incutida.

Agradeço também ao Instituto Politécnico de Bragança, particularmente à Escola Superior de Tecnologia e de Gestão, onde me foram dadas todas as condições para desenvolver a minha actividade académica.

À Marisa, por todo o apoio e compreensão.

Ao meu pai, à minha mãe e ao meu irmão, por todos os esforços que fizeram e que me permitiram chegar até aqui.

A todos os meus amigos.

Resumo

Face à imensidão de informação na *Web*, a descarga de páginas utilizada, por exemplo, em motores de pesquisa, sugere a criação de um sistema de agentes distribuídos (robôs) que descarreguem vários servidores em simultâneo.

Porém, quando procuramos optimizar os mecanismos de descargas somos confrontados, pela necessidade de obedecer às actuais políticas de delicadeza que obrigam à existência de um intervalo de tempo mínimo entre dois pedidos de descarga, ao mesmo servidor. Uma outra dificuldade resulta da forma como as páginas estão distribuídas, uma vez que a maior percentagem está alojada em apenas um pequeno número de servidores, provocando um desnivelamento significativo entre servidores com poucas páginas e servidores (mais densos) com elevado número de páginas.

O aumento do desempenho de cada agente é alcançável através da diminuição do intervalo de descarga, do aumento do número de conexões http persistentes e/ou repartindo as páginas existentes pelos servidores *Web* mais densos por vários robôs. No entanto, se considerarmos que o valor daquele intervalo deve permanecer constante, para o universo de todos os servidores, a tentativa de usar qualquer uma daquelas opções conduz necessariamente à violação das políticas de delicadeza. Todavia, sendo razoável esperar que a capacidade de processamento e comunicação de um servidor cresça na razão directa da respectiva densidade, podemos considerar que o intervalo de descarga pode ser reduzido na proporção inversa. Isto é, podemos fazer variar o tempo de delicadeza nos pedidos de descarga a um servidor no sentido de adaptar o intervalo à respectiva densidade.

Esta dissertação surge como uma alternativa aos processos tradicionais da descarga da *Web*, capaz de reduzir o tempo total de descarga de páginas e minimizar a sobrecarga de comunicação necessária à sincronização de uma colecção de agentes distribuídos. Subjacente está uma infra-estrutura de comunicações e uma arquitectura de descarga que comporta a existência de entidades dedicadas ao particionamento do espaço *Web* e a sua alocação a um conjunto de robôs distribuídos, organizados hierarquicamente através da definição de entidades lógicas.

Durante o processo de descarga das páginas é recolhida informação para gerar configurações de divisão da *Web*, através da aplicação de algoritmos de partição de grafos, baseados em modelos da Internet obtidos por aproximação calculada por caminhos mais curtos no grafo gerado.

Abstract

The hugeness of the *Web* suggests the creation of distributed agent systems (crawlers) to download several sites simultaneously, for instance, in search engine crawling operations.

However, the optimization of crawling download operations faces the need to comply with current politeness policies which require a minimum period of time between two consecutive requests to same server. Another difficulty arises from the distribution of the number of pages on the servers, where a large percentage is hosted in a small number of servers, causing a significant unevenness between servers with few pages and dense servers with a lot of pages.

On each crawler performance increase may be achieved through the reduction of the politeness time interval, increase of the number of `http` persistent connections and/or subdividing dense servers by several crawlers. Nevertheless, considering that the politeness interval should remain constant, for the universe of all the servers, any attempt to use any of the remaining options, leads, inevitably, to the violation of the politeness policies. However, considering the expectation that the density of a server grows proportional to its processing and communication capabilities, it is feasible to consider a reduction in the politeness interval inversely proportional. In other words, politeness interval may be adjusted during download requests to a server based on its density.

This dissertation appears as an alternative to the traditional download processes, able to reduce the total page download time and to minimize the communication overhead required to synchronize the collection of distributed crawlers. Underneath the crawling's operations lays a communication infrastructure and a download architecture embracing the existence of dedicated *Web* space partitioning entities and its allocation to the distributed crawler set, organized hierarchically through logical entities.

Throughout the download process additional information is gathered to generate the resulting division of the *Web* through the application of graph partitioning algorithms based on models of the Internet obtained by approximation by shortest paths on the generated graph.

Conteúdo

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

A descarga automatizada de conteúdos, normalmente associada aos motores de pesquisa, é concretizada por um sistema distribuído de agentes, designados de robôs[1], cuja principal vantagem é a dispersão geográfica, permitindo um aumento da largura de banda agregada do sistema e tirando partido da utilização de diferentes pontos de acesso à Internet.

Os recentes avanços tecnológicos nas áreas das comunicações e do armazenamento de dados, colocaram à nossa disposição poderosos instrumentos para a criação de sistemas em *cluster* com débitos na ordem dos Gigabits que, associados a infra-estruturas computacionais paralelas/distribuídas, permitem a criação de aplicações de elevado desempenho. Actualmente, as aplicações desenvolvidas em ambientes *cluster*, tais como, sistemas de ficheiros distribuídos, estruturas de dados elementares, entre as quais, conjuntos e tabelas de dispersão distribuídos, e camadas funcionais que operaram sobre os anteriores, oferecem já desempenhos agregados na ordem das dezenas de Gigabits. Na sequência da linha de processamento de uma página por um motor de pesquisa, desde que ela é descarregada até ser devolvida ao utilizador, um dos maiores constrangimentos é a descarga das páginas, por estar dependente de factores externos ao sistema, como é o caso, dos servidores *Web*, que podem ter limitações de comunicação, acessibilidade intermitente ou sobrecarga computacional. Além disso, dada a sobrecarga na rede e nos servidores *Web* que os robôs geram, devem ser criadas políticas de delicadeza que minimizem o impacto da operação dos robôs. Em contrapartida, os restantes componentes do motor pesquisa, operando em ambientes minimamente controlados de elevado desempenho, oferecem condições

[1] Em inglês, crawlers ou spiders

de elevada escalabilidade.

As necessidade de dispersão geográfica e de requisitos computacionais avultados, sugerem a criação de sistemas distribuídos multi-nível, onde se distinguem camadas agregadoras de elevado desempenho interligadas por redes LAN (Local Area Network) e SAN (System Area Network), e camadas de dispersão interligadas por redes WAN (Wide Area Network).

Neste contexto, é imperativo que, por um lado, a descarga de páginas seja realizada com a máxima eficiência, de modo a rentabilizar a largura de banda disponível, e por outro, se minimize a sobrecarga de comunicação na sincronização dos agentes, por se encontrarem interligados em redes de maior latência, devido à necessidade da dispersão geográfica.

Os mecanismos tradicionais de alocação de URLs aos robôs, tipicamente baseados em funções de dispersão, começam a perder significado, uma vez que não consideram medidas topológicas. Estas medidas, devidamente aproveitadas, poderiam ser utilizadas para definir um conjunto de critérios que permitisse a optimização dessa alocação, em termos de proximidade entre os robôs e os servidores *Web* e a minimização da sobrecarga na rede entre os robôs. Contudo, o balanceamento do trabalho pelos vários robôs não é uma tarefa trivial, tendo em conta que a distribuição do número de páginas pelos servidores segue uma da lei da potência. Nestas situações, o desempenho global do sistema pode ficar prejudicado quando um robô tem alocados vários servidores mais densos. Uma vez que os mecanismos baseados em funções de dispersão não contabilizam a quantidade de páginas existentes nos servidores, a sua alocação aos robôs pode implicar a colocação de servidores muitos densos no mesmo robô, deteriorando o desempenho desse robô em relação aos outros.

A recolha progressiva de páginas, acompanhada de uma prospecção da topologia física e geográfica da Internet, e lógica da *Web*, permite a sua modelação através de grafos. Com a aplicação de algoritmos de partição multi-objectivo aos grafos, torna-se possível a optimização dos critérios mencionados, permitindo a criação de sistemas de descarga distribuídos em larga escala, com o objectivo de optimizar as comunicações envolvidas.

1.1 Identificação do problema

A criação da Internet é considerada uma das grandes descobertas do século XX. As suas características intrínsecas de encaminhamento dinâmico e adaptativo, aliado à gestão descentralizada, permitem que milhares de máquinas se anexem à Internet com extrema facilidade. As últimas estimativas de Julho de 2007 reportam 489.774.269 de máquinas [ISC 07], com uma tendência crescente ao longo do tempo, tendo tido um crescimento de mais de 8000% desde 1995.

Apesar da grande multiplicidade de serviços e protocolos suportados pela Internet, o seu verdadeiro rosto é a *WWW* (World Wide Web ou doravante apenas *Web*). A *Web* foi iniciada por Tim Berners-Lee no CERN com uma proposta para um sistema de organização de informação, através da interligação de documentos por hiperligações e, posteriormente, formalizada numa junção do hipertexto e da Internet [Berners-Lee 90]. Desde então a *Web* teve um crescimento exponencial ao nível de conteúdo, servidores e utilizadores. O sistema, que se iniciou como um meio eficaz de disseminação e recuperação de informação de uma comunidade, maioritariamente, científica, rapidamente, passou estar acessível a todos, disponibilizando informação de uso geral.

No entanto, o crescimento rápido da *Web*, motivado pela facilidade com que é possível publicar um documento, não foi acompanhado das medidas adequadas de organização e classificação da informação e, por isso, a pesquisa de conteúdos neste vasto repositório tornou-se seriamente problemática.

Para aliviar este problema foram criados motores de pesquisa e directórios que, através de um processo exaustivo de recolha de páginas, procedem à sua indexação e devolvem uma referência de um subconjunto de páginas aos utilizadores, perante uma interrogação selectiva. Embora estes serviços tenham sido introduzidos na última década, o suporte científico e tecnológico em que se baseiam, remonta já dos anos 50 na área da Recuperação de Informação[2] [Salton 83]. O sucesso destes serviços ganhou tamanha popularidade perante os utilizadores da *Web* que, segundo a ComScore [ComScore 07], apenas no mês de Julho de 2007, 70% dos utilizadores em linha, visitaram páginas associadas aos serviços do Google [Google 07]. A grande competitividade pelo domínio do mercado destes serviços, alimentado por receitas de publicidade, criou algum sigilo nas técnicas utilizadas e nos números disponibilizados. Além disso, as estimativas até agora realizadas não são consensuais.

[2]Em inglês, Information Retrieval

Apesar da sua eficácia, os motores de pesquisa debatem-se com um problema de escala, ao nível computacional e de armazenamento, devido às dimensões da colecção em que operam. Segundo a Netcraft [Netcraft 07] existem, actualmente, perto de 130 milhões de servidores *Web* (também designados de *Web sites* ou *sites*), acedidos por cerca de mil milhões de utilizadores [Group 07].

Em termos de tamanho da *Web*, são apenas conhecidas as estimativas geradas a partir das recolhas realizadas pelos motores de pesquisa. Em 2005, Gulli e Signorini reportavam 11,5 mil milhões de documentos indexáveis, numa altura em que estavam indexados 9,5 mil milhões de documentos pelos motores de pesquisa com maior cobertura [Gulli 05].

Mas, mais impressionante do que o tamanho da *Web* é o seu ritmo de crescimento. Em 1999, Lawrence e Giles estimaram o tamanho da *Web* em 800 milhões de páginas [Lawrence 00], o que significa que, em apenas 6 anos, a *Web* ficou quase 14 vezes maior, o que, fazendo uma estimativa grosseira, significa que duplica todos os anos.

Considerando que a *Web* tem, por hipótese, 30 mil milhões de páginas, e que cada página tem em média 18,7 KB, o espaço ocupado seriam 523 TB, o que implicaria uma largura de banda de 1.7 Gbps, para descarregar a *Web* uma única vez num mês.

Uma outra questão é a natureza dinâmica da *Web* ao nível da criação de novos conteúdos, desaparecimento de outros e a alteração dos existentes. Neste contexto, existem indicadores, tais como, a frescura e a idade das páginas, que permitem estimar a frequência com que as páginas devem ser visitadas, implicando um escalonamento dinâmico [Cho 03].

A melhor forma de lidar com estas quantidades e com este dinamismo é, inevitavelmente, através da criação de um motor de pesquisa distribuído, que deverá permitir atingir larguras de banda agregadas daquela ordem de grandeza, para além das evidentes vantagens no processamento e armazenamento da informação.

Na distribuição de um motor de pesquisa subsistem, contudo, uma série de desafios, tais como, a descarga de conteúdos, o seu armazenamento, a indexação de termos de pesquisa e a sua recuperação face às interrogações dos utilizadores. Neste trabalho irá ser focada a questão da optimização da descarga dos conteúdos, não sendo consideradas questões relacionadas com a qualidade das páginas descarregadas, nem com os restantes componentes que fazem parte do motor de pesquisa. Devido à

necessidade em analisar as páginas descarregadas para extrair as hiperligações (ou URLs), assume-se que o processo de indexação é realizado localmente na entidade que efectua a descarga, ficando os índices disponíveis para serem processados para os restantes módulos do motor de pesquisa, que podem ser locais ou remotos.

Num sistema de descarga distribuído, a alocação dos URLs aos robôs responsáveis pela sua descarga é, usualmente, concretizada através da aplicação de uma função de dispersão ao nome da máquina do servidor *Web* que alberga esse URL. No entanto, é bem conhecido o desajuste no balanceamento do trabalho de descarga causado devido à distribuição da lei de potência (*power-law*) seguida pela quantidade de páginas pelos servidores. Além disso, a ausência de informação adicional neste método, implica uma decisão de alocação baseada unicamente no nome do servidor, sem qualquer envolvimento de métricas do ambiente topológico em que o sistema se insere.

A recolha incremental de URLs estabelece condições ideais para a criação de modelos instantâneos da topologia *Web*, da topologia de rede, quando acompanhado com a implantação de um sistema de sondas adicional, e do posicionamento geográfico dos servidores *Web* e dos robôs. Esta informação, sobreposta em grafos, constitui a base fundamental para a aplicação de algoritmos de partição multi-objectivo, dos quais se espera obter uma alocação de URLs optimizada para uma descarga mais eficiente.

Nos sistemas em que foram utilizadas outras técnicas de alocação, tais como partição de grafos, não foram observados resultados práticos acerca do desempenho obtido por estas abordagens. Além disso, muitos desses sistemas, pouca ou nenhuma atenção orientaram para questões importantes como a delicadeza.

Relativamente à modelação da topologia de rede, e apesar da extensa investigação que tem sido feita nesse sentido, é necessário criar um suporte de sondagem e de modelação que rentabilize a infra-estrutura existente, mantendo o grau de aproximação à realidade em níveis adequados, em situações de falta de informação. Os sistemas de descoberta da topologia da Internet, como por exemplo o *Skitter* [CAIDA 07], oferecem uma cobertura de destinos consideravelmente grande, na ordem dos 900 mil endereços, mas não é garantida a coincidência com os endereços utilizados no sistema de descarga.

A partição de grafos revelou-se uma técnica poderosa, mas também inadequada nos moldes em que tradicionalmente é utilizada. Além das diversas combinações parametrizáveis possíveis, a partição dos grafos gerada pelas topologias física e lógica, estão envolvidas restrições que não são suportadas pelos algoritmos de partição co-

nhecidos, na sua versão original. Por outro lado, a distribuição *power-law* do grau dos vértices destes grafos, torna inadequados os eficientes algoritmos de partição multi-nível, sendo mesmo necessário abdicar desta característica.

1.2 Contribuições

A investigação realizada teve como princípio orientador a optimização da descarga de páginas da *Web*, com o objectivo de rentabilizar a largura de banda disponível em cada agente participante num sistema distribuído, sem considerar a qualidade dos recursos descarregados. A largura de banda disponível é utilizada em dois nível: na descarga de páginas e na comunicação entre os agentes para sincronização dos URLs e disseminação de tabelas de encaminhamento. Deste modo, dado um conjunto inicial de URLs e um conjunto de pontos geográficos disponíveis para a colocação dos agentes de descarga, são calculadas as partições do espaço *Web*, que optimizam o tempo de descarga e minimizam o tempo despendido nas trocas de URLs entre os robôs.

O principal contributo desta tese reside na efectiva optimização do tempo total de descarga de um espaço *Web* limitado, atingindo uma redução de cerca de 91% comparado com as tradicionais técnicas baseadas em funções de dispersão. Esta redução considerável foi possível devido à utilização de informação adicional disponível durante a recolha, que permitiu uma maior aproximação topológica entre os robôs e os servidores e a criação de uma metodologia de descarga ajustada à densidade de páginas de cada servidor.

Além disso, mesmo sem considerar o ajuste à densidade de páginas, verificou-se ainda uma redução considerável da sobrecarga de comunicação na sincronização dos robôs, na ordem dos 76% quando comparada com os mesmo mecanismos baseados em função de dispersão.

Nesta linha de investigação, foram ainda analisadas e apresentadas quatro questões fundamentais que permitiram alcançar o contributo principal:

- A criação de uma arquitectura de descarga distribuída organizada em entidades físicas de elevado desempenho, suportadas por *clusters*, e entidades lógicas virtuais que dotam o sistema da escalabilidade necessária ao seu crescimento, no sentido de responder a cenários de descarga mais exigentes, dispostas de forma hierárquica.

- A elaboração de uma infraestrutura para a criação de modelos da Internet, em particular, a aproximação dos tempos de latência entre máquinas para as quais esse tempo é desconhecido, através de recolhas dos tempos médios de ida e volta (RTT).

- A adaptação dos modelos de partição de grafos tradicionais a problemas com elevado enviesamento do grau dos vértices e restrições de bloqueio na sua movimentação, durante a execução dos algoritmos de partição.

- A implementação e implantação de uma infraestrutura de suporte a estruturas de dados fundamentais, tais como, conjuntos, tabelas de dispersão e grafos, num ambiente *cluster*, para o aumento do desempenho de aplicações através da distribuição de memória RAM, utilizada nas diversas experiências conduzidas, em particular, para o aumento da eficiência de um motor de simulação de descarga criado para o efeito.

1.3 Organização da dissertação

A organização da dissertação segue a ordem das contribuições enunciadas, apresentando-se na Figura 1.1 um diagrama da relação entre os vários tópicos abordados.

No capítulo 2 é contextualizado o trabalho apresentado face às abordagens de outros autores, nomeadamente, no que concerne aos sistemas de descarga de conteúdos (robôs), às metodologias de modelação da Internet e às estratégias de partição da *Web*.

No capítulo 3 são discutidos os requisitos necessários para o funcionamento de um robô e proposto um modelo, e a respectiva arquitectura, de um sistema distribuído de descarga organizado hierarquicamente em entidade lógicas, suportado por um esquema de encaminhamento de URLs dinâmico, compatível com o resultado obtido pela partição da *Web* representada por grafos.

No capítulo 4 é apresentada a estratégia de modelação da Internet no contexto deste trabalho, no que se refere à recolha de informação topológica e geográfica, e às técnicas utilizadas para a aproximação de distâncias topológicas entre pares desconhecidos. O capítulo é concluído com a apresentação de dados estatísticos da colecção de páginas utilizadas, entidades recolhidas e características inerentes às topologias observadas.

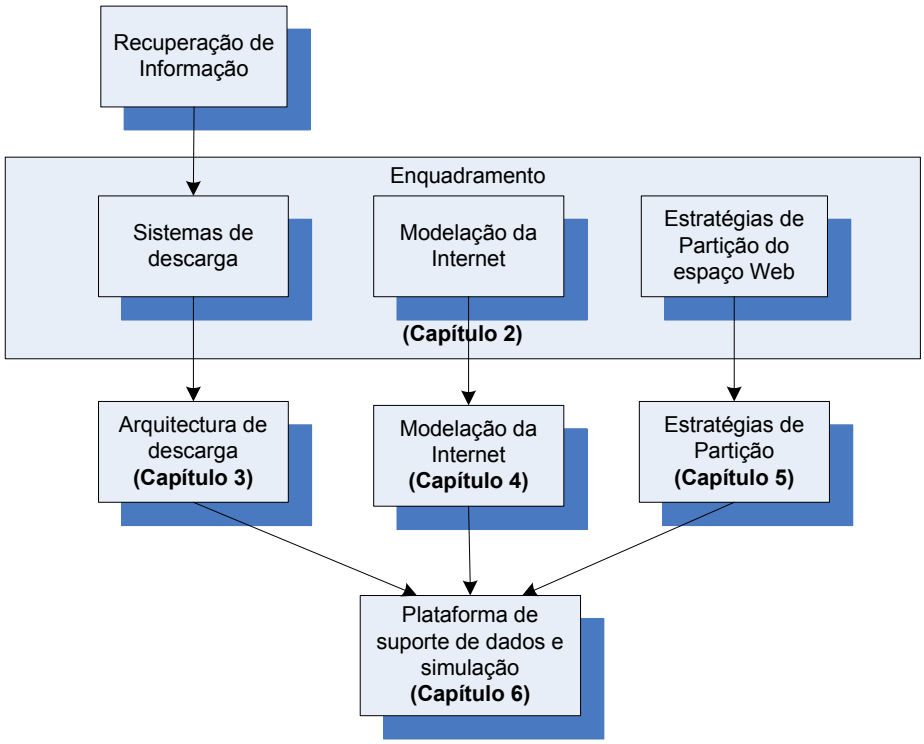


Figura 1.1: Esquema geral da organização da tese

No capítulo 5 são descritos os algoritmos de partição de grafos e apontadas as suas limitações na problemática discutida nesta dissertação. São apresentados os resultados fundamentais que suportam as contribuições principais desta tese.

O capítulo 6, de natureza mais prática, apresenta a infraestrutura utilizada no suporte à manipulação da grande quantidade de dados utilizados e à criação do motor de simulação utilizado para realizar as experiências de validação dos resultados apresentados.

Por último, no capítulo 7 são sistematizadas as principais contribuições deste trabalho em consonância com os resultados obtidos. Acrescentam-se, ainda, algumas orientações para futuros trabalhos.

Capítulo 2

Enquadramento

A descarga da *Web*, levada a cabo por agentes denominados robôs, é tão antiga como a própria *Web*. Apesar de utilizados nas mais variadas aplicações, tais como, verificação de ligações, validação da estrutura HTML, criação e detecção de replicas e geração de estatísticas, uma das suas principais aplicações é nos motores de pesquisa da *Web*.

Um motor de pesquisa responde às interrogações dos utilizadores, indicando a localização do recurso pesquisado, recorrendo a uma estrutura de dados de índices, designados por ficheiros invertidos, gerados a partir da descarga anterior e continuada das páginas *Web*. A Figura 2.1 apresenta uma visão simplificada de um motor de pesquisa. Os mecanismos de indexação e de interrogação, uma vez que se desviam dos objectivos deste trabalho, não serão aqui abordados, embora se assuma a existência do processo de extracção de hiperligações nas páginas descarregadas.

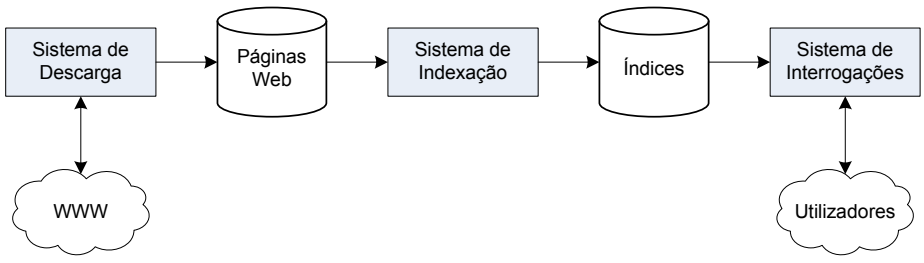


Figura 2.1: Diagrama simplificado de um motor de pesquisa

Em termos temporais, o funcionamento de um motor de pesquisa encerra duas fases distintas e concorrentes. Por um lado, é realizada a descarga das páginas e a

criação ou actualização dos índices, por outro, são respondidas as interrogações dos utilizadores.

2.1 Indexação de documentos

O desempenho global, no que se refere à rapidez e precisão na resposta às interrogações dos utilizadores do motor de pesquisa, depende, em larga escala, da eficiência na construção e utilização dos ficheiros invertidos (FI). Na última década diversos avanços científicos têm sido realizados no sentido de optimizar, de forma eficiente, a representação dos índices, reduzir a memória utilizada e o custo de recuperação [Zobel 06].

Dependendo do tipo de aplicações, o FI deve ser idealizado de forma a responder às necessidades da sua utilização, em particular no que diz respeito ao tamanho da colecção e à frequência de interrogações. No contexto dos motores de pesquisa, devido às características de tamanho avultado, dinamismo do conteúdo e potencial elevado número de interrogações, o FI deve, por um lado, permitir de forma eficiente a resposta às interrogações de centenas de utilizadores em simultâneo, e por outro, acompanhar o ritmo de actualização de conteúdos disponibilizados pelo sistema de descarga.

Um FI simples, recorrendo à utilização de memória secundária, consegue responder a uma interrogação em menos de um segundo [Zobel 06]. No entanto, através de diversas técnicas de aperfeiçoamento, estes valores podem ser drasticamente reduzidos. Em [Anh 02] são apresentados valores de 27 interrogações por segundo, utilizando mecanismos de normalização das frequências dos termos que, adicionalmente, facilitam o corte dinâmico de termos nas interrogações.

A utilização de memória intermédia[1], aplicada aos FI e às interrogações, é outra técnica que se tem revelado bastante eficiente. De acordo com o trabalho desenvolvido por Saraiva et al. é possível atingir cerca de 60 interrogações por segundo [Saraiva 01].

Recorde-se, contudo, que a eficiência na resposta às interrogações não é o único parâmetro a optimizar. A redução no espaço ocupado pelos FI é uma outra vertente alvo de investigação, devido, principalmente, à necessidade em lidar com colecções de grandes dimensões. Utilizando uma abordagem sem compressão, um FI pode

[1] Em inglês, caching

atingir mais de 30% do tamanho não compactado da colecção inicial, ao passo que, utilizando compressão estes valores podem ser reduzidos para proporções entre 10% e 15% [Scholer 02].

Do ponto de vista de um robô, é fundamental que a criação do FI acompanhe a taxa de descarga das páginas, ou que a ultrapasse de modo a evitar constrangimentos na descarga. Para colecções de elevada dimensão e dinamismo, como a *Web*, apontam-se, na literatura duas solução: a junção dos FIs dos documentos existentes com os FIs dos novos documentos [Cutting 90]; e actualização incremental [Tomasic 94]. No entanto, a preservação da disponibilidade do sistema de interrogações é colocada em causa, devido ao tempo necessário para a actualização dos índices, tendo sido sugerido um esquema híbrido de criação de índices [Búttcher 06]. Neste esquema são alcançados cerca de 70 GB/hora de documentos processados para a criação de índices, ao que corresponde um débito de cerca de 800 páginas/segundo, assumindo um tamanho médio de 25 KB por página, considerando uma única máquina com características convencionais.

No decorrer dos processos de investigação nestes domínios e com a evolução da tecnologia, nomeadamente ao nível do hardware, a tendência do débito de páginas processadas é de crescimento. Além disso, actualmente, a taxa de descarga dos robôs documentados não se aproxima destes valores, o que acentua a necessidade de optimização dos sistemas de descarga, como uma forma de aumentar o desempenho global dos motores de pesquisa.

Saliente-se, contudo, que esta optimização não significa apenas a maximização da utilização da largura de banda disponível pelos robôs, mas também a sua rentabilização, de acordo com políticas de delicadeza, tendo em conta que a ligação externa à Internet é, potencialmente, o recurso mais escasso e instável de todo o processo de um motor de pesquisa.

2.2 Sistemas de descarga

A descarga automatizada de páginas da *Web* é um processo que aparenta, à primeira vista, uma grande simplicidade. No entanto, existe um conjunto de factores que tornam o processo bastante elaborado. A Figura 2.2 apresenta um esquema genérico de um robô. O algoritmo consiste, basicamente, em preparar uma fila de URLs com uma semente inicial. Cada URL é retirado da fila por um `Escalonador` e descarre-

gado. O texto recolhido é analisado e são extraídas as hiperligações existentes, que são colocadas na fila novamente para descarga.

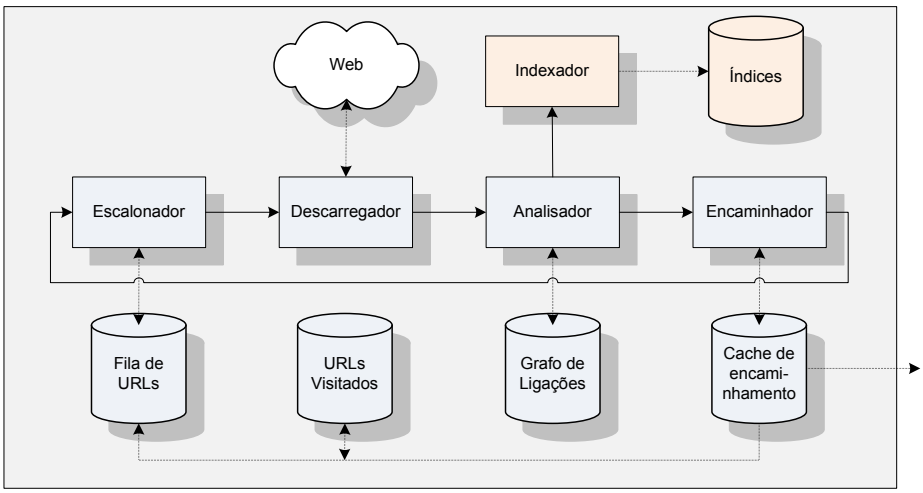


Figura 2.2: Esquema genérico de um robô

Mas esta simplicidade desvanece-se com a evidência da dimensão da *Web*, do seu rápido crescimento e mudança. Por este motivo, a criação de robôs deve ter em consideração as seguintes políticas em simultâneo [Castillo 04a]:

- Políticas de delicadeza;
- Políticas de selecção de URLs a descarregar;
- Políticas de revisita de URLs;
- Políticas de paralelização e distribuição.
- Políticas de armazenamento;

2.2.1 Políticas de delicadeza

Uma das questões que mais constrange o funcionamento dos robôs, mas com menor formalismo, é a delicadeza para com os servidores *Web* e a intrusão causada na utilização normal da rede.

Embora não existindo uma convenção acerca do intervalo de tempo entre cada página descarregada, existe uma premissa que alerta para o facto de um robô não dever

bombardear o mesmo servidor com pedidos muito frequentes. Caso contrário, existem sérios riscos de o robô ser bloqueado, surgirem vários e-mail ou telefonemas de reclamações [Brin 98].

Por outro lado, é também clara a necessidade em minimizar o impacto do tráfego gerado pelo robôs na utilização corrente dos serviços disponibilizados para os utilizadores. Uma solução que é tipicamente apontada para reduzir este impacto, consiste na dispersão da carga de comunicação na rede através de soluções distribuídas ou recorrendo a critérios de proximidade, utilizando a distância topológica de rede ou a distância geográfica, a fim de contribuir para uma melhor utilização dos recursos disponíveis.

Além disso, existem ainda questões pessoais relacionadas com a eventualidade de cada administrador dos servidores *Web* pretender, ou não, que o conteúdo das páginas seja recolhidas por robôs. Nesta situação encontra-se à disposição dos administradores um standard de exclusão de robôs [Koster] que, de uma forma generalizada, todos os robôs utilizam, e que define exclusões para secções dos sítios *Web*, páginas e robôs.

Embora centralizado, o robô descrito por Heydon e Najork utiliza várias centenas de filas de espera de URLs que são associadas pelo nome da máquina do servidor do URLs [Heydon 99]. Cada fio de execução é responsável pela descarga de apenas uma destas sub-filas, evitando, assim, a sobrecarga de um servidor. Mais tarde, os mesmos autores remodelam o trabalho referenciado, concluindo que uma política de delicadeza mais agressiva se tornaria necessária, adicionando um intervalo de tempo entre descargas de página de dez vezes o tempo que a página anterior demorou a ser descarregada [Najork 02].

A preservação da delicadeza realizada nos estudos desenvolvidos por Shkapenyuk e Suel utiliza um conjunto de filas de espera com quantidade igual ao número de servidores *Web* descobertos [Shkapenyuk 02]. Adicionalmente, é aguardado um período de 30 segundos entre cada nova descarga do mesmo servidor.

Embora não indicando o valor preciso do tempo de espera entre descargas sucessivas para o mesmo servidor, Boldi et al. referem apenas a utilização de valor "adequado". Além disso, a descarga é feita por múltiplos fios de execução cada um responsável por um servidor *Web*, evitando, assim, acessos concorrentes ao mesmo servidor.

No trabalho desenvolvido por Singh et al., uma vez que é utilizado um sistema P2P em que é aplicada uma função de dispersão ao nome da máquina, cada agente do

robô é sempre responsável pela descarga do mesmo servidor *Web*, preservando-se, deste modo, as políticas de delicadeza [Singh 04].

2.2.2 Políticas de selecção de URLs a descarregar

O processo de selecção dos URLs para descarga consiste em ordenar os URLs encontrados ou retirá-los com algum critério da fila de URLs. A necessidade deste processo reside no facto de um robô poder não conseguir descarregar todas as páginas da *Web* e, como contrapartida, descarregar as páginas consideradas mais "importantes".

Cho et al. apresentam várias métricas de importância de páginas, entre as quais: 1) focada por uma interrogação; 2) quantidade de hiperligações de entrada; 3) PageRank; 4) quantidade de hiperligações de saída; e 5) local ao URL [Cho 98].

As métricas 1) e 5) utilizam um mecanismo de comparação de similaridade (por exemplo, o modelo do espaço vectorial) para dimensionar a importância da página. Enquanto que a métrica 1) utiliza o conteúdo da página, a métrica 5) utiliza alguma informação local existente no URL, como exemplo, os domínios. Estas métricas são orientadas por uma interrogação e, por isso, dependentes de uma entrada. Estas métricas são particularmente utilizadas em robôs focados.

As métrica 2) e 4) utilizam uma contabilização da quantidade de hiperligações, que entram e que saem duma página, respectivamente. A métrica 3) descrita em originalmente por Brin e Page, calcula a importância de uma página com base na importância das páginas que ligam à primeira [Brin 98].

No que respeita ainda ao trabalho de Cho et al., a ordenação das páginas com a métrica PageRank conduziu à descoberta das páginas mais importantes em primeiro lugar. No entanto, sendo o peso computacional desta métrica bastante elevado, muitas vezes é preferida um métrica de ordenação baseada numa travessia das páginas em largura[2], obtendo-se, mesmo assim, as páginas importantes nas primeiras etapas de funcionamento do robô [Najork 01].

Um outro trabalho descrito por Castillo et al. aponta ainda uma métrica baseada na quantidade páginas a descarregar por servidor, optando por descarregar primeiro as páginas dos servidores que têm mais páginas para descarregar, o que permite uma optimização na descarga em configurações distribuídas, mantendo a descarga das páginas importantes no início [Castillo 04b].

[2]Em inglês, breadth-first search

2.2.3 Políticas de revisita

Se a descoberta de novas páginas é um factor importante para um robô, a preservação da actualidade das páginas já visitadas, deve ser seriamente considerada, devido aos tempos longos de operação dos robôs e à volatilidade da *Web*.

Cho e Garcia-Molina no trabalho conduzido neste temática definem duas métricas do estado de actualização de uma colecção: 1) a frescura, que mede a percentagem de páginas actuais; e 2) a idade, que mede o tempo que as páginas se encontram desactualizadas [Cho 00b].

A grande parte dos robôs já analisados não utilizam, aparentemente, qualquer mecanismo de revista. No entanto, considerando que um lote de páginas é descarregado uma vez, é sempre possível iniciar novas revisitas completas, permitindo a actualização das páginas. Uma outra alternativa é a descarga incremental, em que a descoberta de novas páginas e a actualização das existentes é alternada. Cho e Garcia-Molina conseguem provar que, com ambas as alternativas se obtém uma frescura média igual, utilizando uma velocidade média de visita igual [Cho 00a]. No entanto, para a segunda alternativa a descarga não necessita de picos de débito de descarga tão elevados.

A frequência de mudança das páginas *Web* é variável, o que sugere uma frequência de visita também variável, proporcional à mudança. Surpreendentemente, noutro trabalho de Cho and Garcia-Molina prova-se que uma visita uniforme, isto é, visitar todas as páginas com a mesma frequência, conduz a valores de frescura médios mais elevados em toda a colecção [Cho 03].

2.2.4 Políticas de paralelização e distribuição

A descarga concorrente de robôs pode ocorrer em dois níveis: 1) na paralelização local a uma máquina através de múltiplos fios de execução ou múltiplos processos [Najork 02], ou ainda pela gestão da sincronização assíncrona de vários *sockets* utilizados para a descarga [Burner 97, Brin 98]; e 2) na distribuição por múltiplos agentes [Burner 97, Brin 98, Teng 99, Shkapenyuk 02, Najork 02, Boldi 02, Singh 04, Loo 04, Papapetrou 04].

A paralelização em cada agente que conduza ao estabelecimento de diversas conexões simultâneas é um dos pontos fulcrais para a optimização da descarga que, suportada por várias filas de espera organizadas por nome da máquina servidora

ou por máquina física, garante a preservação das políticas de delicadeza, sem constranger profundamente o ritmo de descarga. Contudo, devido ao enviesamento da distribuição da quantidade de páginas pelos servidores, independentemente do grau de paralelismo, um agente fica ocupado com um pequeno número de servidores na parte final da descarga de um espaço *Web*. A minimização deste problema pode ser alcançado pela utilização de uma frequência de descarga maior para os servidores com mais páginas, diminuindo o intervalo de delicadeza [Baeza-Yates 02].

A distribuição do robô, por diversos agentes dispersos geograficamente, constitui um mecanismo fundamental para a dispersão do acesso à Internet, tirando partido de diversos fornecedores de serviço em simultâneo e, por consequência, um aumento do débito agregado e uma diminuição global do tráfego gerado, e também para a dispersão da carga computacional e de armazenamento.

Uma consequência desta abordagem é que a utilização do resultado da descarga, por exemplo, para um motor de pesquisa, deve também ser distribuída, envolvendo um processamento total ou parcial das páginas descarregadas, antes destas serem enviadas ao sistema central responsável pelo armazenamento dos índices, de modo a não criar um constrangimento computacional no ponto central de interrogações [Baeza-Yates 07].

Segundo Cho e Garcia-Molina, a coordenação de um robô distribuído[3] pode ser realizada segundo: 1) uma atribuição dinâmica, controlada por um coordenador central, cuja comunicação a partir dos agentes pode ser mais ou menos intensa consoante a granularidade da partição; 2) uma atribuição estática, que dispensa um coordenador central, uma vez que cada agente decide qual a parte da *Web* que é responsável [Cho 02].

A arquitectura descrita por Brin e Page é um exemplo de uma atribuição dinâmica em que um coordenador central delega os URLs a visitar para um conjunto de agentes, e estes devolvem as páginas descarregadas a um servidor de armazenamento [Brin 98]. Dada a pequena granularidade das partições, é possível que este sistema não escale convenientemente com o aumento do espaço *Web*.

No trabalho descrito por Teng et al. é realizada uma atribuição do espaço *Web* baseada em partição de grafos. Embora o sistema seja puramente distribuído, a coordenação é realizada através do sistema TSpaces e no momento do re-balanceamento um dos componentes é eleito como coordenador, recebendo a informação necessária

[3]Nesta referência bibliográfica designado de robô paralelo

para o efeito, dos restantes elementos. Neste trabalho são utilizados os tempos de acesso às páginas e as o número de hiperligações para gerar grafos em que são aplicados algoritmos de partição de grafos [Teng 99].

Um outro trabalho da autoria de Papapetrou e Samaras utiliza um esquema de partição baseado em dados recolhidos dos Registos Regionais da Internet (RIR) para criar aglomerados hierárquicos e assim realizar delegação de URLs em agentes migratórios, adicionando ainda critérios de localidade, através da sondagem parcial com ferramentas *ICMP-ping* e pedidos HTTP/HEAD [Papapetrou 04]. É reclamada um redução de uma ordem de magnitude no tempo total de descarga de 1000 servidores *Web*, utilizando os critérios de localidade em comparação com a não utilização desses critérios.

A atribuição estática é alcançada através de uma função de partição, tipicamente uma função de dispersão aplicada ao nome da máquina ou a todo o URL, sendo este último pouco recomendado devido à geração de uma sobrecarga de rede muito elevada na troca dos URLs encontrados [Loo 04]. Podem ainda ser utilizados mecanismos de partição baseados na hierarquia do espaço de nomes de domínios, mas com reconhecidos desajustes de balanceamento na distribuição do trabalho atribuído a cada agente.

Boldi et al. apresentam um sistema que utiliza um mecanismo de partição de dispersão consistente [Karger 97] aplicado ao nome do servidor *Web*, aliviando assim, problemas de re-distribuição dos URLs quando existem alterações na constituição do sistema distribuído, comuns em funções de dispersão normais baseado no módulo da quantidade de agentes [Boldi 02].

Utilizando o conceito de sistemas Peer-to-peer (P2P) e funções de dispersão consistente, construído com base no sistema Chord [Stoica 01], Singh et al. apresentam um robô distribuído com critérios de localidade. A escalabilidade do sistema é validada até 14 agentes, sendo também apresentados resultados das vantagens na utilização de critérios de localidade [Singh 04].

O trabalho realizado por Loo et al. apresenta uma outra solução de um robô distribuído, assente numa tabela de dispersão distribuída e, em que são analisadas várias funções de dispersão de atribuição estáticas, tais como, uma função de dispersão aplicada ao URL e ao nome da máquina, a descarga de um servidor por diversos robôs e, ainda, um mecanismo de re-direcção, em que um URL recebido por um agente pode ser encaminhado para outros em situações de sobrecarga [Loo 04].

2.2.5 Políticas de armazenamento

Internamente, um robô necessita lidar com uma grande variedade e quantidade de dados através de estruturas de dados eficientes. Duas dessas estruturas principais são a fila de URLs e os URLs vistos. A primeira, podendo ser uma simples fila de espera é, actualmente, aceite a necessidade de ser uma fila prioritária devido às políticas de selecção e revisita, como veremos em seguida. A segunda estrutura elimina redundância de URLs repetidos. Algumas soluções consideram ainda um mecanismo de detecção de conteúdo visto, evitando o processamento das páginas iguais que foram previamente descarregadas.

Um dos primeiros robôs publicados, o *RBSE Spider* que, implementava estas estruturas numa base de dados relacional, carecia, contudo, de qualquer funcionalidade paralelizada ou distribuída sendo, por isso, obsoleto para os requisitos actuais [Eichmann 94].

Num dos primeiros trabalhos que deu origem ao conhecido motor de pesquisa Google [Google 07], é reconhecida a necessidade em implantar uma *cache* local de DNS [Brin 98]. Na presença de um sistema distribuído e derivado das políticas de delicadeza, vulgarmente, o mesmo robô descarrega as páginas quase sempre do mesmo servidor. Além disso, tendo em conta a pequena volatilidade dos nomes das máquinas esta *cache* torna-se bastante eficiente.

Qualquer uma das estruturas referidas, devido às dimensões que podem atingir, necessitam ser armazenadas em memória secundária. A preocupação na optimização destas estruturas foi o tema do trabalho apresentado em [Heydon 99]. O grosso da fila de URLs é mantida em disco, utilizando centenas de sub-filas com um tampão de 600 URLs cada um para colocar ou retirar URLs do disco. Os URLs são atribuídos a estas filas por função de dispersão, garantindo que cada fio de execução apenas processa um servidor. Os URLs vistos são armazenados em disco utilizando uma *cache* em memória, criando uma combinação da assinatura[4] do nome da máquina e do URL completo, de modo a obter-se uma localidade adicional na verificação dos URLs, originada pela prevalência do número de hiper-ligações locais nas páginas. Adicionalmente, são ainda utilizadas estruturas de dados para a verificação de conteúdo visto utilizando assinaturas dos documentos e mantendo uma tabela de índices em memória para a optimização dos acessos ao disco. Para a resolução de nomes de máquinas é utilizada uma *cache* conjuntamente com uma implementação

[4]Em inglês, fingerprint

proprietária de resolução de nomes com múltiplos de fios.

Teng et al. apresentam um robô distribuído que utiliza o sistema TSpaces como suporte para o armazenamento e comunicação [Teng 99]. O sistema TSpaces é um middleware de rede que permite a comunicação e o armazenamento distribuídos. A organização dos dados em tuplos, oferece a versatilidade e a confiança dos sistemas de bases de dados tradicionais, aliado a um modelo de programação em linguagem Java, que fornece uma conveniente independência de plataformas [Wyckoff 98]. Embora não sendo apresentados detalhes sobre a implementação das estruturas, é evidenciada a necessidade na concretização de memória intermédias previamente à submissão de dados no sistema de armazenamento.

A implementação de um robô distribuído de alto desempenho descrito no trabalho desenvolvido por Shkapenyuk e Suel utiliza a Berkeley DB como suporte persiste de armazenamento das estruturas de dados de controle do robô, argumentando-se que, para recolhas volumosas é impossível conter, quer as filas de URLs, quer o conjunto de URLs vistos em memória principal [Shkapenyuk 02]. Contudo, é utilizado um mecanismo de *cache* para os URLs visto, através de uma árvore Red-Black implementada em memória, que é sincronizada com o conteúdo em disco com operações de mistura, sem perder a continuidade do funcionamento da estrutura.

Em 2002, Heydon e Najork apresentaram uma versão distribuída do seu robô Mercator [Najork 02]. Nesta evolução foram optimizados alguns aspectos das estruturas de armazenamento, nomeadamente, no que respeita às filas de URLs, que suportam agora um esquema de prioridades. As estruturas dos URLs vistos foram também melhoradas, suportando um mecanismo de mistura[5] para aumentar o desempenho de acesso.

O trabalho de Boldi et al. sobre um robô distribuído apenas refere a memória principal como suporte às estruturas de dados por ele utilizadas [Boldi 02]. O aumento de desempenho é alcançado através da criação proprietária de algumas colecções nativas da linguagem Java.

Singh et al. descrevem um robô distribuído baseado numa Tabela de Dispersão Distribuída (TDD)[6] assente no protocolo Peer-to-Peer (P2P), Chord [Singh 04]. As filas de URLs são implementadas em memória contidas numa tabela de dispersão organizadas por nome da máquina do servidor. As estruturas de dados utilizadas para armazenar os URLs já encontrados, baseiam-se em filtros bloom [Bloom 70]

[5]Em inglês, merge

[6]Em inglês, Distributed HashTable (DHT)

residentes em memória. Existe ainda um mecanismo de verificação de conteúdo já encontrado, que é armazenado na TDD utilizando as assinaturas dos documentos.

2.2.6 Resumo das políticas

A tabela 2.1 apresenta um resumo das diferentes políticas aplicadas em alguns dos sistemas de descarga analisados.

Em termos de armazenamento, tem havido grandes preocupações na optimização ao suporte dos dados manipulados devido à sua potencial dimensão. Para além de terem sido criadas estruturas de dados vocacionadas para o efeito, a distribuição por um conjunto de agentes, dilui o peso de armazenamento entre eles.

A tendência para a implementação de soluções de robôs distribuídos foi desde muito cedo colocada em prática, devido às claras vantagens no aumento do desempenho na descarga. À excepção de alguns trabalhos iniciais, a maioria implementa uma política de distribuição, embora, nem sempre é dito de forma clara a forma de o concretizar. Em alguns casos é utilizado um coordenador central, responsável pela decisão da atribuição de qual robô descarrega as páginas, noutros casos, é utilizada uma função de dispersão, que permite abdicar do constrangimento que o coordenador central pode acarretar, mas com informação limitada do espaço *Web* descarregado.

O escalonamento de quais URLs devem ser descarregados em primeiro lugar, também tem suscitado bastante a atenção dos investigadores. Técnicas de ordenação que utilizam o PageRank, podem oferecer uma maior qualidade nas páginas descarregadas, contudo o peso computacional do seu cálculo, pode surgir como uma desvantagem face a técnicas mais simples, como a travessia em largura ou a descarga dos servidores mais densos em páginas. Neste último caso, obtém-se ainda grandes vantagens no balanceamento da descarga de modo evitar a ociosidade dos agentes quando descarregam os servidores menos densos mais depressa.

As políticas de delicadeza são, sem margem para dúvida, a questão mais *delicada* de todo o processo, uma vez que não existe um valor pre-estabelecido adequado e a sua diminuição pode conduzir a desempenhos de descarga superiores. Em alguns, trabalhos foram utilizadas políticas de delicadeza constantes, na ordem entre os 15 e 30 segundos. Foram identificadas técnicas de ajuste dinâmico com base no tempo de descarga da página anterior e também inversamente proporcional à quantidade de páginas dos servidores.

Tabela 2.1: Comparação de características de robôs

		Políticas			
Referência	**Ano**	**Delicadeza**	**Selecção**	**Distribuição**	**Armazenamento**
[Eichmann 94]	1994	Protocolo de exclusão de robôs.	Travessia em largura.	Centralizado.	Base de dados relacional.
[Brin 98]	1998	Protocolo de exclusão de robôs.	PageRank.	Distribuído com coordenador centralizado.	Cache de DNS.
[Heydon 99]	1999	Protocolo de exclusão de robôs. Centenas filas às quais são atribuídos URLS por função de dispersão. Um fio de execução por servidor. Tempo de espera entre descargas do mesmo servidor: 10t, em que t é o tempo de descarga da página anterior.	Travessia em largura. PageRank.	Centralizado.	URLs vistos com assinaturas em disco com cache em memória. Tabela de índices das assinaturas dos documentos em memória RAM. Cache de DNS.

Continua na próxima página...

		Políticas			
Referência	**Ano**	**Delicadeza**	**Selecção**	**Distribuição**	**Armazenamento**
[Teng 99]	1999	-	-	Distribuído. Atribuição do espaço com partição de grafos.	Sistema de cache prévio ao à submissão ao suporte de armazenamento de dados.
[Shkapenyuk 02]	2002	Uma fila por nome de máquina. 30 segundos de espera entre cada descarga no mesmo servidor.	Travessia em largura.	Distribuído.	BerkeleyDB. NFS. URLs vistos em disco com cache em memória e operações de mistura. Diversas entidades de resolução de nomes com cache.
[Boldi 02]	2002	Um fio de execução por servidor.	-	Distribuído sem coordenador central.	URLs vistos em tabelas de dispersão em memória com assinaturas.

Continua na próxima página...

		Políticas			
Referência	**Ano**	**Delicadeza**	**Selecção**	**Distribuição**	**Armazenamento**
[Najork 02]	2002	Protocolo de exclusão de robôs. Centenas filas às quais são atribuídos URLS por função de dispersão. Um fio de execução por servidor. Tempo de espera entre descargas do mesmo servidor: 10t, em que t é o tempo de descarga da página anterior.	Fila de URLs com prioridade baseada no historial de mudança da página.	Distribuído.	URLs vistos com assinaturas em disco com cache em memória e operações de mistura. Cache de DNS com implementação assíncrona proprietária. Fila de URLs em disco, com o início e cauda em memória.
[Castillo 04b]	2004	Tempo de espera entre descargas no mesmo servidor: 15 segundos.	Descarga das páginas cujos servidores têm mais páginas em espera.	Distribuído.	-

2.3 Modelação da Internet

A Internet é uma imensa rede com gestão descentralizada, composta por redes mais pequenas interligadas, organizada em sistemas autónomos (AS).

A atribuição de endereços é realizada por delegação e segue uma hierarquia administrativa. A *The Internet Assigned Numbers Authority* (IANA) é a autoridade responsável pela delegação de grandes agregados de endereços aos cinco registos regionais (RIR, Regional Internet Registry):

- AfriNIC (*Africa Network Information Center*), para a África.
- APNIC (*Asia Pacific Network Information Center*), para as Regiões do pacífico da Ásia.
- ARIN (*American Registry for Internet Numbers*), para a América do Norte.
- LACNIC (*Regional Latin-American and Caribbean IP Address Registry*), para a América do Sul e Caraíbas.
- RIPE (*Rééseaux IP Europens*), para a Europa, Ásia Central e Médio Oriente.

Cada um dos RIR delega agregados do seu espaço de endereçamento aos registos locais (LIR, Local Internet Registry) que, por sua vez, distribuem sub-agregados de endereços aos utilizadores finais.

A identificação do conjunto de endereços delegados a cada registo utiliza um esquema de agregação designado por *Classless Inter-Domain Routing* (CIDR) [Group 93, Rekhter 93], que surgiu em substituição do esquema de cinco classes [Gerich 93], face à ineficiência no aproveitamento do espaço de endereçamento disponível. Foi devido à criação do CIDR que a Internet pôde crescer até as dimensões que tem actualmente, embora o sistema de endereçamento IPv4 esteja já a ser substituído gradualmente pelo IPv6.

Internamente, a Internet é composta por dispositivos de encaminhamento (*routers*) que asseguram a chegada da informação (pacotes de dados) ao seu destino, através de um conjunto de protocolos de encaminhamento, compostos por políticas administrativas e dinâmicas. A natureza dinâmica destes protocolos pode ter como consequência uma divergência nas rotas seguidas pelos pacotes para o mesmo destino e uma assimetria nas rotas opostas.

A modelação da Internet e a análise da sua topologia tem vindo a desempenhar um papel muito importante para a sua compreensão. A aplicação dos resultados obtidos é muito diversificada, mas a título de exemplo podem enumerar-se as seguintes aplicações: diagnóstico de problemas [Cheswick 00], gestão de redes e optimização [Siamwalla 98] e validação de novas tecnologias.

Algumas das aplicações recorrem à utilização de simuladores de redes, tais como o `ns2` [Bajaj 99], ou o NCTUns [Wang 03], que se baseiam em topologias criadas a partir de geradores de topologias, tais como, o `inet` [Winick 02], o `gt-itm` ou o `tiers` [Doar 96], que utilizam os resultados obtidos da modelação.

Embora as aplicações da modelação da Internet possam ser bastante diversificadas, de um modo geral são concretizadas através de técnicas, genericamente, designadas por tomografia de rede.

2.3.1 Tomografia de rede

A tomografia da rede visa, para além da eventual descoberta da topologia, a caracterização de parâmetros internos da rede, baseada em medições de tráfego num subconjunto de nós. Vulgarmente, o termo tomografia de rede está associado a uma classe de problemas estatísticos inversos. No entanto, é também utilizado para designar a extrapolação de medidas de rede, sem utilizar necessariamente procedimentos estatísticos [Rabbat 04].

O termo tomografia, utilizado pela primeira vez por Vardi no contexto da Internet [Vardi 96][7], numa analogia à tomografia do corpo humano, visava (1) a estimação do volume de tráfego com base em medições realizadas ao nível das ligações de rede [Cao 00, Tebaldi 98]. Segundo a classificação realizada em [Castro 04], a tomografia de rede pode também ser (2) a estimação de parâmetros ao nível das ligações de rede, tais como o atraso [Coates 01] e a perda de pacotes [Duffield 06], através de medições fim-a-fim. No mesmo contexto podem ainda ser consideradas as técnicas de (3) estimação de parâmetros de distância fim-a-fim, com base na medição fim-a-fim [Rabbat 04]. Complementarmente, através da tomografia de rede é possível a descoberta da topologia de rede.

[7] Embora seja feita a referência com o objectivo de contextualização, não foi possível encontrá-la, para efeitos de leitura

Monitorização passiva

A classe de problemas referidos em (1) é concretizada através da monitorização passiva ao nível dos encaminhadores, com o intuito de estimar a matriz origem-destino de tráfego de uma rede, que especifica o volume de tráfego entre uma origem e um destino. Embora esta técnica se tenha revelado útil, a introdução de heurísticas para lidar com o peso computacional das abordagens algorítmicas tem conduzido a algumas imprecisões nos resultados. Por outro lado, a necessidade de implantação de mecanismos de monitorização nos nós envolvidos, associado à complexidade computacional, não permite que esta técnica seja expandida em larga escala, sendo, por isso, preferida por operadores de rede na detecção de problemas inerentes aos nós que controlam [Lawrence 05].

Estimação ao nível das ligações de rede

A monitorização activa, através de sondas injectadas na rede, é a base da classes de problemas indicado em (2). O objectivo é inferir parâmetros nas ligações internas da rede, através da recolha de valores, dos mesmos parâmetros, em medições levadas a cabo a partir de sondas em nós extremos na rede. O funcionamento desta técnica baseia-se no envio de sondas, preferencialmente, com mecanismos de difusão selectiva[8] de forma a serem analisados as partes comuns dos percursos seguidos pelas sondas, e assim, inferir correlações de medidas localizadas a partir das medidas fim-a-fim [Caceres 99].

Por motivos de segurança, ou apenas por questões administrativas, nem sempre o protocolo de difusão selectiva se encontra activado, pelo que, recorre-se frequentemente ao envio de sondas unilaterais. Coates e Nowak apresentam uma alternativa à difusão selectiva, em que são envidados pares de sonda unilaterais quase em simultâneo, sendo a avaliação dos parâmetros de rede efectuado a partir da correlação do parâmetro medido de uma sonda em relação à outra [Coates 00]. Esta ideia baseia-se no facto de, por exemplo, para a estimação do atraso, ambas as sondas terão aproximadamente o mesmo atraso em saltos da rede atravessadas em comum. Deste modo, as diferenças nos atrasos totais das sondas, seriam causados por saltos não comuns.

Contudo, estas técnicas requerem a cooperação dos emissores e receptores de sondas, podendo assim, limitar o raio de acção das experiências conduzidas, por necessitarem

[8]Em inglês, multicast

de acesso para a instalação dos monitores. Em alternativa a estas medições de uma via (OTT, One way Trip Time), em [Tsang 04] é apresentada uma solução com bons resultados utilizando medições de ida e volta (RTT, Round Trip Time). Com este tipo de medições torna-se possível a realização de sondagens a uma escala muito mais alargada e sem a necessidade de intervenção dos pontos de destino, que em algumas situações pode ser impossível.

As técnicas utilizadas para a sondagem activa com medições RTT envolvem um procedimento semelhante ao `traceroute`[Jacobson 88], que permite conhecer o caminho desde uma origem até um destino, através da enumeração dos sucessivos RTT entre pontos intermédios e o destino final.

A comunicação na Internet é feita através do envio de mensagens, tipicamente fragmentadas em pacotes, nas quais está indicado o seu tempo de vida TTL (Time To Live). Cada encaminhador de rede decrementa o TTL, possibilitando um mecanismo de protecção que evita a sua circulação por períodos excessivos, evitando o congestionamento. O `traceroute` toma partido desta funcionalidade enviando pacotes ICMP ou UDP com TTL sucessivamente crescente iniciado em 1 até o destino ser alçado. Quando o TTL atinge o valor 0, os sucessivos encaminhadores devolvem uma mensagem à origem, permitindo desta forma identificar o caminho percorrido pelos pacotes até ao seu destino.

Apesar da sua prática utilização, algumas medidas de segurança tomadas pelos administradores de redes levam, muitas vezes, à desactivação do ICMP em alguns dos encaminhadores, impedindo a detecção de algumas rotas. Esta é a contrapartida entre a perda de alguma precisão das medições com RTT e a pesada infraestrutura necessária nas medições OTT.

Estimação ao nível dos caminhos fim-a-fim

Para além da estimação ao nível das ligações, a sondagem activa, em particular com medições RTT, pode também ser utilizada para estimar medidas fim-a-fim, tais como atrasos (latência ou distância), largura de banda e conectividade.

Neste contexto, diversas optimizações têm vindo a ser realizadas, nomeadamente, na determinação da quantidade ideal de pontos de prova [Horton 03], bem como a sua colocação [Kumar 04], assumindo que a topologia envolvida é conhecida à partida.

O conceito de distância de rede, em termos de latência (através do RTT) ou largura de banda, foi introduzido no trabalho desenvolvido por Francis et al., como um

mecanismo de optimizar a proximidade das aplicações cliente a espelhos de dados pela Internet [Francis 01]. Neste mesmo trabalho é apresentada uma arquitectura de medição de distância entre qualquer conjunto de pontos da Internet, designada IDMaps, baseada na colocação de pontos especiais, designados de *Tracers*, responsáveis pela medição de um conjunto pré-fixo de endereços (Grupos de máquinas associadas por um CIDR). A distância entre dois quaisquer pontos é calculada pelas somas das distâncias entre um ponto e o *Tracer* mais próximo, a distância entre este último e o *Tracer* mais próximo do ponto de destino, com a distância entre este último e o *Tracer* mais próximo dele.

Esta arquitectura tem como base a desigualdade triangular, em que se que verifica, $d(x, y) \leq d(x, z) + d(z, y)$. Se pelo menos uma das distâncias $d(x, z)$ ou $d(z, y)$ for suficientemente pequena, então o desvio causado por z é pequeno e a aproximação é precisa. Daqui resulta que a distância entre dois pontos da Internet poderia ser aproximada pela soma das distâncias entre pontos intermédios.

Contudo, este facto é peremptoriamente contradito por diversos autores [Wang 07, Savage 99], que comprovam a violação da desigualdade triangular nas rotas da Internet, mas apesar disso, os autores do IDMaps apoiam a exequibilidade deste cálculo, assumindo um erro na aproximação.

2.4 Estratégias de partição do espaço *Web*

A distribuição de um sistema de descarga por um conjunto de entidades conduz, naturalmente, à questão de que parte do espaço *Web* total, cada entidade é responsável. Embora se tenha já referido na secção 2.2.4 os mecanismos inerentes às políticas de distribuição, considerou-se relevante incluir esta secção por consistência da abordagem seguida na tese.

No trabalho realizado Brin e Page um coordenador central é responsável por armazenar os URLs e designar o robô destinado pela sua descarga [Brin 98].

Embora sem um coordenador centralizado explícito, o trabalho de Teng et al. utiliza uma infraestrutura de memória distribuída designada TSpaces para realizar a coordenação entre os robôs. Este trabalho utiliza a partição de grafos para concretizar a partição do espaço *Web*, realizado por um líder, escolhido entre os robôs. Apesar da extensa descrição deste trabalho, não são abordadas questões de delicadeza, nem apresentados resultados práticos da eficiência do processo [Teng 99].

Papapetrou e Samaras realizam um esquema de partição criando aglomerado hierárquicos com base em dados recolhidos dos Registos Regionais da Internet (RIR), adicionando ainda critérios de localidade [Papapetrou 04].

Boldi et al. apresentam um sistema que utiliza um mecanismo de partição de dispersão consistente aplicado ao nome do servidor *Web*, aliviando assim, problemas de re-distribuição dos URLs quando existem alterações na constituição do sistema distribuído, comuns em funções de dispersão normais baseado no módulo da quantidade de agentes [Boldi 02].

Utilizando o conceito de sistemas Peer-to-peer (P2P) e funções de dispersão consistente, construído com base no sistema Chord [Stoica 01], Singh et al. apresentam um robô distribuído com critérios de localidade. A escalabilidade do sistema é validada até 14 agentes, sendo também apresentados resultados das vantagens na utilização de critérios de localidade [Singh 04].

O trabalho realizado por Loo et al. apresenta uma outra solução de um robô distribuído, assente numa tabela de dispersão distribuída e, em que são analisadas várias funções de dispersão de atribuição estáticas, tais como, uma função de dispersão aplicada ao URL e ao nome da máquina, a descarga de um servidor por diversos robôs e, ainda, um mecanismo de re-direcção, em que um URL recebido por um agente pode ser encaminhado para outros em situações de sobrecarga [Loo 04].

Capítulo 3

Arquitectura de descarga

A abordagem a uma solução distribuída para a descarga optimizada de conteúdos pode ser baseada numa divisão do espaço *Web*, de tal forma que seja possível agrupar na mesma secção os recursos que se encontrem próximos do robô que os vai descarregar, segundo uma determinada função de proximidade, e retirar dessa secção os recursos que impliquem uma sobrecarga de comunicação nos intercâmbios de recursos entre entidades remotas, quando estes não são destinados para o robô que os encontrou.

Independentemente da forma em como é realizada a divisão do espaço *Web*, é necessário estabelecer uma infra-estrutura capaz de suportar e orquestrar o funcionamento de um conjunto de robôs de forma eficiente. Devido às grandes quantidades de informação que um sistema de descarga necessita manipular, as suas exigências enquadram-se, essencialmente, no armazenamento, mas também no processamento e na largura de banda necessária para o acesso aos recursos *Web*.

A atribuição dos URLs é concretizada, periodicamente, com recurso a um coordenador central no seu âmbito de actuação, cuja comunicação se limita à recepção de resumos da topologia física e lógica, e à difusão de regras de encaminhamento para os agentes do robô. O processo de atribuição baseia-se em informação recolhida pelos robôs que a enviam sintetizada em lotes para o coordenador.

Cada robô dispõe de um infra-estrura de comunicação e armazenamento baseada em *cluster*, que lhe garante, pelo menos uma ligação externa à Internet e diversos nós computacionais que ofereçam um ambiente adequado para a execução das tarefas de descarga da *Web*, análise das páginas e extracção das hiperligações, e encaminhamento dos URLs descobertos. Alguns membros desta arquitectura deverão, ainda,

suportar operações de encaminhamento intermédio (operando como representantes) e operações de coordenação de secções de robôs.

Este capítulo descreve uma proposta de arquitectura de um sistema de descarga de recursos e o seu funcionamento, tendo sido publicada uma versão preliminar no artigo [Exposto 03].

3.1 Requisitos da arquitectura

A vulgarização dos sistemas computacionais permitiu às instituições recursos computacionais que, embora podendo não ser de elevado desempenho, oferecem capacidades de processamento e armazenamento suficientes para manipular secções mais pequenas de um espaço *Web* mais abrangente. Deste modo, torna-se vantajosa a utilização de equipamento convencional e o aproveitamento de recursos, eventualmente já implantados nas instituições. Para que seja possível a rentabilização deste equipamento é lícito pensar num sistema de descarga que alcance as diversas dezenas de nós computacionais nos quais são implantados os robôs. O número potencialmente elevado de nós, a sua adição e a sua remoção devem ser abordados com especial cuidado, de modo a ser possível a realização destas operação em tempos que garantam a escalabilidade do sistema e a preservação da informação topológica e de encaminhamento em tamanhos compatíveis com a dimensão do problema.

Além disso, a utilização de um número considerável de instituições conduz à dispersão geográfica dos robôs pela Internet, de modo a que o sistema possa usufruir de vários pontos de acesso e, assim, aumentar a largura de banda global do sistema. Por outro lado, a emergência de tecnologias de comunicação em SAN (Storage Area Network) suportadas por Infiniband, 10Gigabit e, mais recentemente, Myrinet 10Gigabit permite a construção de *clusters* de elevado desempenho. No entanto, derivado da sua natureza localizada, os *clusters* carecem, só por si, da dispersão necessária para este tipo de sistemas, mas em contrapartida oferecem um óptimo recurso localizado de elevado desempenho em termos computacionais e de armazenamento.

Por outro lado, a participação de diversas entidades cooperantes pode ser significativamente estendida se for criada uma camada organizacional, com a qual se torna possível definição de critérios administrativos com a especificação do tipo de informação que cada entidade é responsável, quer ao nível do conteúdo, quer ao nível da contextualização geográfica.

De uma forma sistematizada, podemos enumerar os seguintes requisitos para um sistema de descarga distribuído:

- Escalável. No sentido de responder às necessidades actuais de dimensão da *Web* e poder adaptar-se facilmente, quer no que diz respeito à ampliação de capacidade para dar resposta ao crescimento previsível em numero de páginas, ou em espaço de *Web*, quer à redução de capacidade para se ajustar a constrangimentos económicos ou eventual diminuição do espaço *Web*.

- Delicado. Devem ser garantidas políticas de delicadeza para com os servidores *Web* de modo a não afectar o seu normal funcionamento. Durante um determinado período deve existir apenas um agente responsável pela descarga das páginas do mesmo servidor.

- Dinâmico. O sistema deve ser capaz de aumentar ou diminuir o número de entidades constituintes durante o tempo de funcionamento do sistema.

- Eficiente. O sistema deve ser suportado por estruturas de dados robustas distribuídas, para permitir a manipulação de grandes volumes de informação, sem perda de eficiência, quando comparada com a manipulação de estruturas de dados centralizadas equivalentes. Para além disso, a comunicação entre as entidades não pode constituir um gargalo no desempenho final do sistema.

- Disperso. A dispersão das entidades do sistema por zonas geográficas distintas, quando alimentada por relações de cooperação apropriadas, pode contribuir para reduzir, significativamente, os inconvenientes das limitações em termos da largura de banda total acumulada, disponível para o acesso à Internet.

- Organizado. O desenho de um sistema que lida com enormes volumes de informação, e para os quais se espera um elevado número de entidades participantes, coloca, necessariamente, problemas de escalabilidade para os quais se torna obrigatório encontrar abordagens compatíveis com aquelas dimensões. A estruturação é o conceito chave usado para lidar com estas questões, e está presente na definição da organização hierárquica e multi-nível, das entidades participantes, mas também na definição de critérios de localidade topológica das mesmas entidades e do estabelecimento de aglomerados de informação relacionada, por critérios adaptativos e administrativos.

- Cooperativo. A cooperação é uma propriedade das entidades do sistema que através da acção coordenada visam satisfazer os seguintes objectivos: 1) minimizar os custos gerais de computação e de comunicação, envolvidos no processo de recolha e tratamento da informação, e 2) aumentar a eficiência global do sistema. Em particular, pretende-se através da selecção criteriosa das entidades, com base na definição de regras administrativas e adaptativas, saber a cada momento quais as entidades responsáveis pelo estabelecimento das ligações à Internet, de forma a minimizar os custos de globais de comunicação e aumentar a largura de banda do sistema.

- Abrangente. Dentro dos critérios definidos para cada entidade assume-se que o sistema deve cobrir exaustivamente o espaço *Web* que foi definido.

- Actualizado (Fresco). Face ao dinamismo intrínseco da *Web*, o sistema deve permitir uma actualização concertada e escalonada das páginas previamente descarregadas, sem colocar em causa a descoberta de novas páginas.

- Baixo custo. Este sistema apresenta-se como uma alternativa às soluções centralizadas, ou comerciais, de elevado custo, baseadas em tecnologias e equipamentos proprietários, que tiram partido das tecnologias de conveniência para a construção de um *cluster* dando o suporte básico às entidades do sistema. Deste modo, o sistema deve ser suportado por equipamento convencional e de baixo custo.

3.2 Entidades e organização da arquitectura

O preenchimento dos requisitos referidos passa em primeiro lugar pela definição de um conjunto de entidades lógicas. Em seguida será analisado o funcionamento da arquitectura. Quanto à nomenclatura, iremos designar o sistema distribuído na sua totalidade por SIRe. O sistema SIRe é constituído por três tipos de entidades: (1) SIR, a mais elementar, e duas entidades compostas, (2) SIRe Local e (3) SIRe Alargado, que se organizam numa hierarquia multi-nível constituindo uma topologia. A Figura 3.1 apresenta as entidades do SIRe e a forma de se associarem entre si.

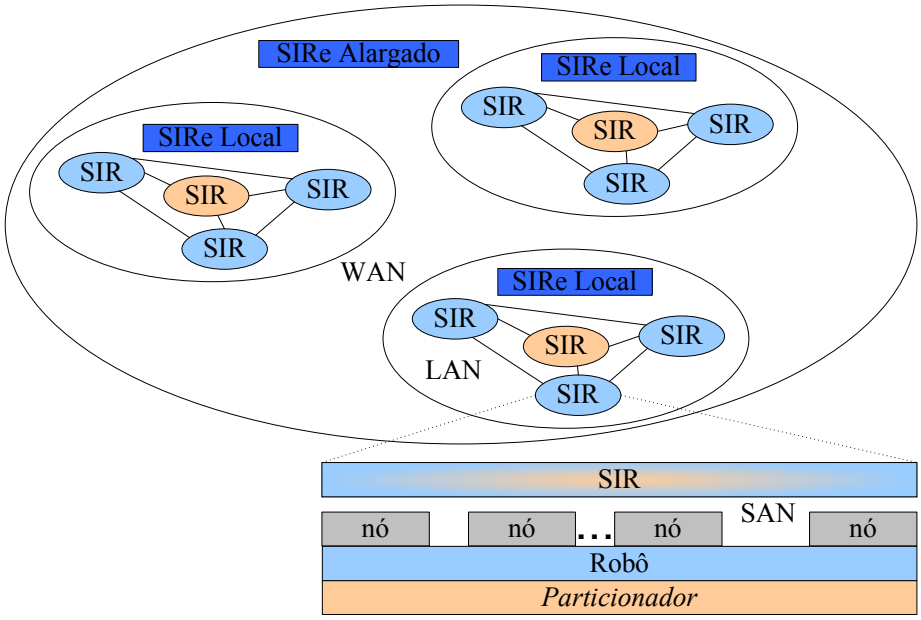


Figura 3.1: Entidades do SIRe

3.2.1 Entidade SIR

A entidade SIR é a entidade básica do sistema que permite a extensão das capacidades computacionais e de armazenamento de uma única máquina através da sua realização como um *cluster*. A utilização de tecnologias de comunicação de elevado desempenho, em cada um dos nós do *cluster*, garantem a eficiência de um conjunto de serviços indispensáveis ao funcionamento do SIR, como por exemplo, o suporte a estruturas de dados distribuídas (DHT) [Rufino 04, Rufino 05].

Do ponto de vista externo, o SIR funciona como uma Imagem de Sistema Única (*Single System Image*, SSI), identificável por um único endereço IP. Internamente, o SIR dispõe de um conjunto de serviços que permitem o aproveitamento distribuído dos recursos existentes, utilizando por exemplo, um gestor de recursos e escalonamento de tarefas, ou através da implementação proprietária de serviços de distribuição.

Em termos de serviços do sistema de descarga, cada SIR pode desempenhar o papel de robô ou *Particionador*. O primeiro serve de suporte às operações de descarga e o segundo oferece suporte à concretização da partição do espaço *Web*. Este assunto será mais detalhado na Secção 3.3.

Publicamente, cada SIR oferece um ou mais serviços que são disponibilizados para

os restantes SIRs, os quais recebem os pedidos de acesso ou colocação de informação e os distribuem pelo *cluster* através de uma camada de mais baixo nível de acesso a estruturas de dados distribuídas.

Cada SIR dispõe de uma tabela de encaminhamento com a qual é decidido o encaminhamento dos URLs quando estes são descobertos pelos robôs.

3.2.2 Entidade SIRe Local

Um SIRe Local é definido pela associação de um ou mais SIRs pertencentes à mesma rede institucional (LAN ou MAN) e que partilham a mesma linha de acesso externo à Internet. É clara a falta de dispersão no acesso à Internet no seio desta entidade, uma vez que o acesso é partilhado pelos diferentes constituintes. No entanto, possibilita a uma instituição a criação de um aglomerado de informação mais abrangente e estruturado, podendo o conteúdo por ele manipulado ser definido de acordo com as suas necessidades.

Cada SIRe Local pode ter filtros atribuídos administrativamente, que definem o âmbito de manipulação dos URLs recolhidos pelos seus robôs, como por exemplo, um domínio de rede, ou regras baseadas no conteúdo. Estes filtros sobrepõem-se à tabela de encaminhamento que é herdada por reunião dos seus SIRs descendentes.

3.2.3 Entidade SIRe Alargado

O SIRe Alargado é constituído pela associação de um ou mais SIRes Locais ou outros SIRes Alargados. O SIRe Alargado, para além de permitir a criação de um nível adicional de organização, vem concretizar a dispersão efectiva no acesso à Internet.

Os SIRes alargados herdam, também, as tabelas de encaminhamento dos seus descendentes. Ambos os SIRes (Local e Alargado) dispõem ainda de informação acerca dos seus representantes (Secção 3.4.1), que são as entidades que, fisicamente, respondem por eles.

3.3 Funcionamento da arquitectura

A base do funcionamento do sistema SIRe encontra-se materializada em cada um dos nós que compõem os SIRs, actuando, contudo, como se uma única entidade se

tratasse. A finalidade principal de cada SIR é a tarefa de descarga levado a cabo pelo robô no seu todo. Além disso, cada SIR pode ser especializado em tarefas de realização da partição do espaço *Web*, necessárias para a criação das partições, que resulta num conjunto de dados de encaminhamento a enviar para cada robô, de modo a que estes possam decidir o encaminhamento dos URLs que vão encontrando. Ambas as tarefas possuem uma estreita relação, comunicando entre si informação vital para o seu funcionamento.

Em termos computacionais, um SIR *Particionador* pode ter elevados requisitos de processamento e armazenamento. A sua materialização num SIR, permite-lhe usufruir de elevada capacidade e flexibilidade, dada a sua disposição em *cluster*. Cada SIRe (Local ou Alargado) dispõe de um ou mais SIRs *Particionador*, sendo este o responsável pela concretização do processo de partição para os SIRs, ou SIRes descendentes do SIRe imediatamente ascendente.

De acordo com a estrutura hierárquica criada pela topologia de SIRes, o *Particionador* de cada SIRe efectua as partições do seu contexto, ou seja, do número de entidades descendentes desse SIRe. Com este mecanismo, cria-se uma cadeia de mecanismos de partições multi-nível encadeados, reduzindo o peso computacional que existiria caso a estrutura de robôs fosse plana.

A Figura 3.2 visualiza a atribuição dos *Particionadores* pelos SIRes de uma topologia de oito SIRs, organizada hierarquicamente em três níveis. Como se pode observar, cada entidade lógica (SIRe) têm um SIR atribuído sendo responsável pelo processo de partição dos seus descendentes. No exemplo da figura, o SIR 8 realizaria o processo de partição em duas partições, uma para o SIRe E e outra para o SIRe F. O SIR 5, *Particionador* do SIRe F, calcularia as partições do espaço *Web* resultante da partição do nível anterior, dividindo esse espaço em duas partições, uma para o SIRe C e outra para o SIRe D.

Vejamos de seguida os esquemas funcionais de cada uma das possíveis especializações dos SIRs apresentados na Figura 3.3.

3.3.1 Robô

Os URLs são inseridos no *Verificador de URLs*, em que é verificada a sua existência para evitar duplicações, sendo descartados aqueles que já existam. Os que não existam são colocados na estrutura *URLs Pendentes*, donde são retirados segundo uma ordem de escalonamento pelo *Escalonador de URLs*. Este escalonamento deve incluir

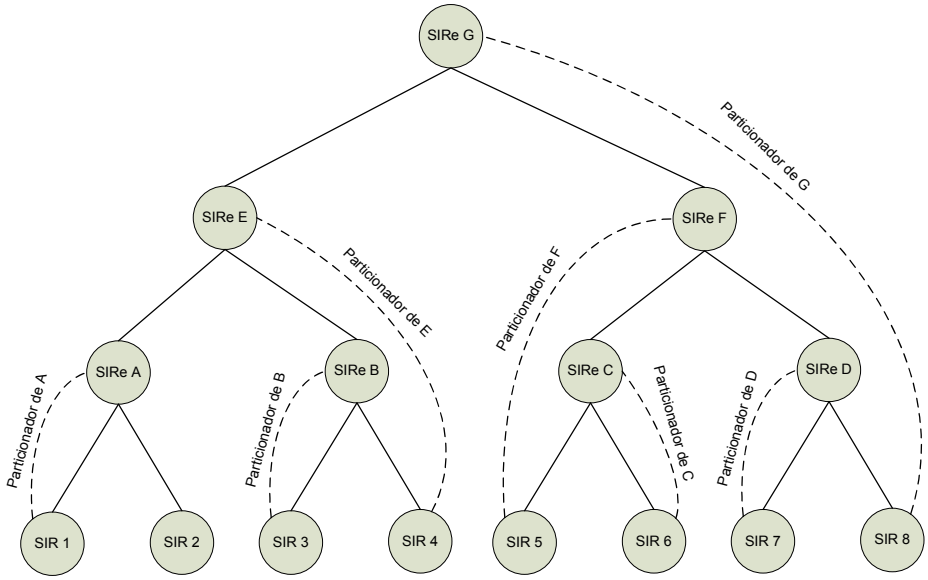


Figura 3.2: Atribuição dos *Particionadores* pelos SIRes

mecanismos para implementar políticas de delicadeza para com os servidores *Web*, bem como medidas de frequência de mudança de páginas de forma a implementar a revisita de páginas.

Depois de efectuada a análise ao URL, o *Determinador Geográfico* calcula a localização geográfica do servidor *Web* do URL utilizando bases de dados como o *NetGeo* [CAIDA 02] e o *GeoNames* [Agency 06]. Seguidamente, é calculada a rota desde o robô até ao servidor *Web* do URL utilizando o `traceroute` [Jacobson 88], donde é retirada informação acerca da topologia de rede até ao servidor desse URL.

Após estes dois passos, o *Descarregador* encarrega-se de descarregar o URL, sendo gerada uma assinatura[1] da página e colocada na estrutura *Conjunto de páginas*. Depois de analisadas e extraídas as hiperligações da página, o *Encaminhador de URLs* decide, com base na estrutura *Tabela de encaminhamento*, quais as entidades de destino desses URLs, sendo reiniciado o circuito para esses URLs nos SIRs a que se destinaram.

De realçar ainda, a existência de mais uma estrutura de dados, *Filtros*, definida administrativamente com regras de inclusão de conteúdos e a partir da qual é actualizada a *Tabela de encaminhamento*.

[1]Fingerprint

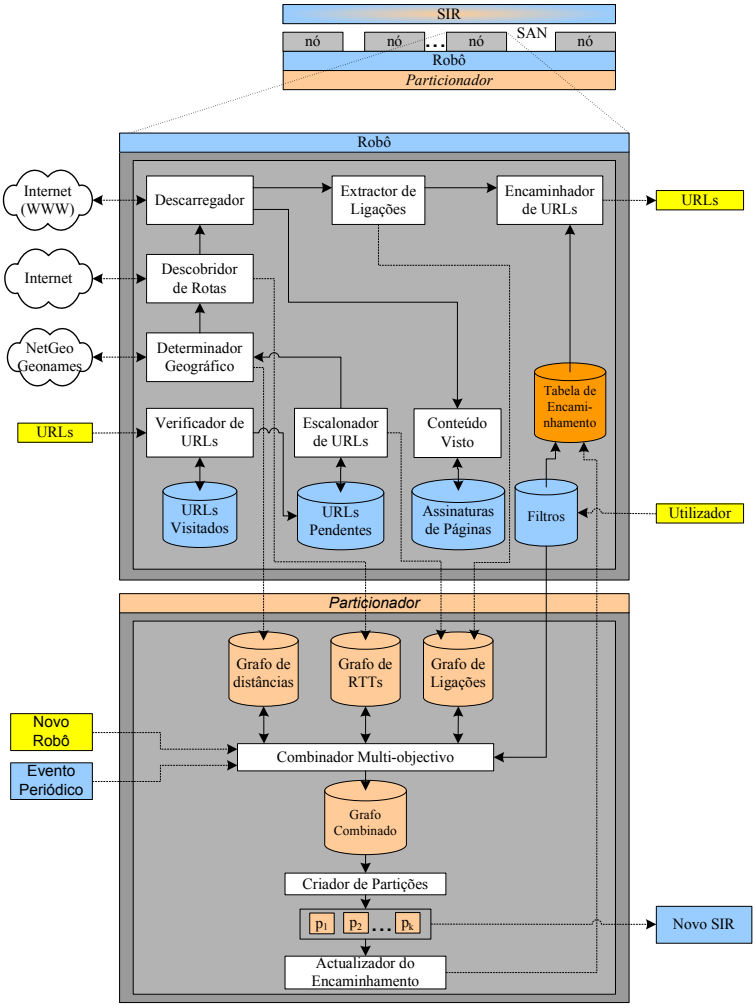


Figura 3.3: Esquema funcional do SIRe

3.3.2 *Particionador*

Anexado a cada SIRe existe um SIR especializado designado por *Particionador*. O objectivo desta entidade é reunir informação recolhida pelos robôs do SIRe em que se enquadrada e regenerar as tabelas de encaminhamento, após a geração de um novo conjunto de divisões (partições) do espaço *Web* da responsabilidade desse SIRe.

Pela observação da Figura 3.3, podemos constatar o estreito relacionamento entre o robô e o *Particionador*. Esta entidade mantém um conjunto de grafos de dados actualizados, em lote, pelo robô e que, depois de efectuada a sua combinação e partição, são actualizadas as tabelas de encaminhamento dos robôs do contexto dessa

entidade. Os detalhes que descrevem os mecanismos de partição serão abordados no Capítulo 5.

De modo a evitar uma sobrecarga excessiva dos *Particionadores* por parte dos robôs, a informação é enviada pelos robôs em lotes. Os vértices dos grafos do *Particionador* são tipicamente nomes de máquinas, podendo também ser endereços IPs de máquinas[2]. De momento estão previstos três tipos de grafos: 1) um grafo de distâncias geográficas entre actualizado pelo componente *Determinador Geográfico* do robô, 2) um grafo de RTTs[3], actualizado pelo *Descobridor de rotas* e 3) um grafo de ligações *Web*, actualizado pelo *Extractor de Ligações*. No entanto, a adição de outros grafos é facilmente suportada pelo mecanismo de partição multi-objectivo. O componente *Combinador Multi-objectivo* combina os vários objectivos representados por cada um dos grafos iniciais num novo grafo, o qual é, posteriormente, sujeito ao mecanismo de partição desencadeado pelo *Criador de partições*. A totalidade do processo de partição levado a cabo pelo *Particionador* é apenas desencadeado perante a saída ou entrada de novos robôs, ou quando é atingido um limiar temporal ou quantitativo de novos vértices nos grafos.

A informação acerca de quais os vértices que ficaram em cada partição é então difundida para os robôs do contexto desse *Particionador* e criadas as tabelas de encaminhamento.

3.4 Informação de Encaminhamento

Um dos pontos fundamentais para o funcionamento do sistema SIRe é o encaminhamento dos URLs que vão sendo descobertos. Esta funcionalidade assenta na estrutura *Informação de Encaminhamento*, a qual é composta por um identificador da entidade, por informação da topologia, que descreve a organização das entidades criadas no sistema, e pelas tabelas de encaminhamento, que indicam de que forma os URLs devem ser encaminhados.

Fisicamente, a tabela de encaminhamento encontra-se materializada em cada SIR, estando distribuída pelos nós que o compõem.

[2]No texto que se segue utilizar-se-á apenas IP para designar o endereço IP de uma máquina
[3]Round Trip Time

3.4.1 Informação topológica

O sistema SIRe é uma estrutura hierárquica de componentes organizados com base em associações de uma ou mais entidades. Dado o potencial elevado número de entidades no sistema, para manter a escalabilidade do sistema, torna-se necessário que, ao invés de cada entidade dispor de conhecimento total sobre todos os componentes de um dada configuração, ou topologia, se encontre um mecanismo que venha a permitir reduzir, substancialmente, os limites desse conhecimento.

Desta forma, cada SIR tem, apenas, conhecimento das entidades ascendentes na hierarquia de níveis a que pertence, e de todos os descendentes directos dos seus ascendentes. Esta informação é visualizada na entrada `Ascendentes`, sendo indicado o ascendente, seguido dos seus descendentes entre parêntesis.

Para além disso, é definido o conceito de representante de uma entidade composta (SIRe). Os representantes são SIRs e correspondem às entidades que representam fisicamente um SIRe, encontrando-se na descendência da sua hierarquia. Por sua vez, o SIR é representado por um dos nós que o compõem, utilizando-se a entrada `Endereços` para o efeito. Numa tabela de informação topológica, os endereços do SIR a que a tabela diz respeito são enumerados todos os nós que dele fazem parte. Em contrapartida, para os SIRs referenciados apenas é conhecido um número limitado do nó físico que os representam.

Tomando como exemplo o sistema da Figura 3.6(b), com 8 SIRes e 15 SIRs, teríamos, em termos de informação topológica, para o SIR 1 a organização no formato especificado na Tabela 3.1.

O campo de informação dos ascendentes de um SIR reflecte, não só a enumeração das entidades visíveis mas, também, o respectivo, encadeamento estrutural.

A argumentação da conveniência da criação deste tipo de informação topológica encontra-se facilmente, se tomarmos como exemplo uma árvore de l níveis, em que cada nível tem c filhos, em que os nós intermédios são SIRes e os nós folha são os SIRs, no total teríamos $n = c^l$ SIRs. Um conhecimento global desta estrutura implicaria cada SIR dispôr de c^l entradas. Com o conhecimento parcial seria apenas necessário o conhecimento de $l \times (c-1) + 1$ entidades, para poder conhecer toda a estrutura do SIRe, o que resulta numa ordem de complexidade $O(\log_c(n)$. O acesso a um determinado SIR remoto para efeitos de encaminhamento de URLs pode implicar uma série de saltos lógicos para atingir o seu destino, o que, no pior dos casos, corresponde a $\log_c(n)$ saltos, ou seja, l, o que é perfeitamente compatível

ID:	SIR 1	
Ascendentes:	SIRe A	(SIR 1, SIR 2, SIR 3)
	SIRe F	(SIRe A, SIRe C, SIRe H)
Representantes:	SIRe C:	SIR 7
	SIRe H:	SIR 4, SIR 10, SIR 13
Endereços:	SIR 1:	Nó a, Nó b, Nó c
	SIR 2:	Nó 2
	SIR 3:	Nó 3
	SIR 4:	Nó 4
	SIR 7:	Nó 7
	SIR 10:	Nó 10
	SIR 13:	Nó 13

Tabela 3.1: Informação topológica para o SIR 1

com um número de SIRs igual a c^l, assegurando, desta forma, a escalabilidade do sistema.

No que diz respeito à propagação de informação quando é adicionada um nova entidade, torna-se necessário difundir as tabelas de encaminhamento para todas as restantes entidades. Utilizando o mesmo exemplo da árvore, para se conseguir difundir toda a informação será necessário enviar $(c-1)\cdot l$ tabelas de encaminhamento, seguido de $(c-1)\cdot(l-1)$, até $(c-1)$, perfazendo l envios. Uma vez que estes envios decorrem em paralelo, a difusão realizar-se-ia em $l = \log_c(n)$ iterações.

Os SIRes, enquanto entidades lógicas, são, naturalmente, representados pelas entidades descendentes materializadas fisicamente pelos SIRs. A existência de um número de representantes de um SIRe superior a 1, permite a distribuição e o balanceamento de carga de encaminhamento, de processamento e, também, capacidade de tolerância a faltas. A utilização de todos os SIRs do SIRe como representantes poderia aumentar, significativamente, a quantidade de informação topológica que cada SIR teria que manter. Assim, considera-se que o ideal é utilizar um número de representantes de ordem logarítmica, obtido através da selecção dos SIRs descendentes da entidade lógica que se pretende representar. A selecção pode fazer-se com base em medidas de qualidade de serviço que garantam a escalabilidade e melhoria de desempenho no acesso às entidades remotas, como é o caso do tempo de resposta das comunicações e carga computacional dessa entidade.

Para visualizarmos melhor o funcionamento e a economia de recursos com o esquema de utilização de representantes, vejamos a Figura 3.4. Este exemplo, retrata um

sistema SIRe com 27 SIRs e 9 SIRes. Repare-se que, do ponto de vista do SIR 1, por exemplo, apenas os SIRs do SIRe a que pertence (SIRe A) são conhecidos (SIR 1, SIR 2 e SIR 3), mais os representantes dos SIRes B, C, K e L. Com este mecanismo reduz-se significativamente a quantidade da informação topológica, permitindo um encaminhamento sucessivo por saltos para alcançar uma determinada entidade, tal como veremos na secção seguinte.

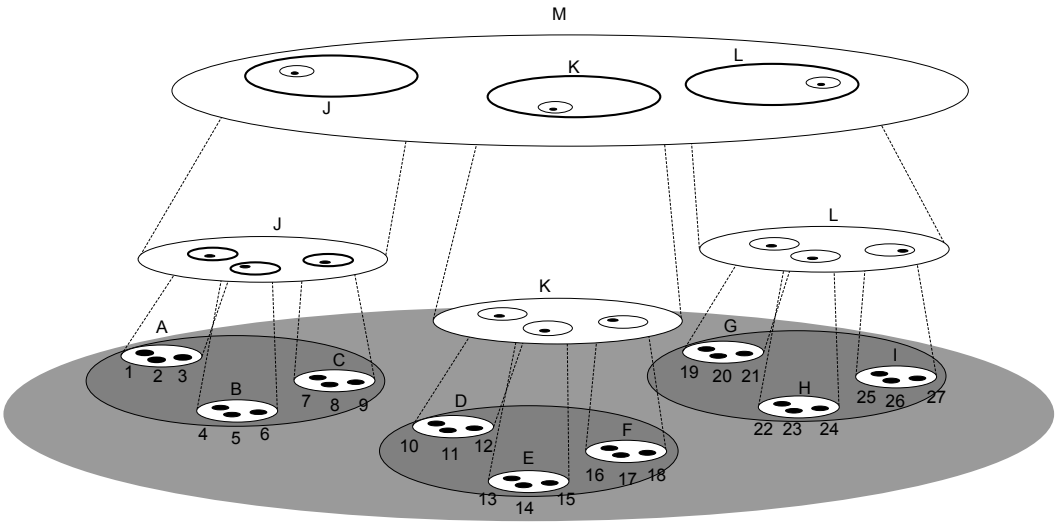


Figura 3.4: Exemplo de um SIRe

3.4.2 Tabelas de encaminhamento

Num sistema de descarga distribuído uma das tarefas principais é, com base num conjunto predefinido de partições do espaço *Web*, determinar para qual entidade um novo URL deve ser encaminhado. Uma vez que não é utilizado um mecanismo predeterminado e determinístico para encaminhar os URLs, deve ser criado um mecanismo de enumeração de URLs para cada SIR. No momento da criação de partições pelos *Particionadores*, as tabelas de encaminhamento, assim como a restante informação topológica é difundida para as entidades robô.

Evidentemente, que a enumeração individual de todos os URLs seria completamente descabida. No entanto, é possível criar um esquema de agregação de URLs utilizando a sua informação topológica, tal como, 1) o nome da máquina do servidor do URL, 2) o endereço IP do servidor do URL e 3) a um nível mais abrangente, o bloco

de subrede ou o bloco CIDR a que o IP do servidor do URL pertence. Qualquer uma destes *agregadores* diminui significativamente a quantidade de informação necessária para representar um conjunto substancial de URLs, valor que diminui em cada uma das propostas, mas aumentando a quantidade de excepções. Considera-se uma excepção um elemento que pertence a um bloco de subrede, mas que na realidade não está previsto para descarga por um robô, em resultado do processo de partição. A agregação por blocos de subrede tem já uma eficácia demonstrada no domínio das redes, de cuja área se podem aproveitar os conceitos de sobreposição e agregação de blocos de rede. Alternativamente à criação de excepções, existe a possibilidade de segmentar ainda mais os blocos, podendo aumentar a sua quantidade significativamente.

Para se compreender melhor o grau de simplificação que este mecanismo oferece, a Tabela 3.2 mostra as quantidades de *agregadores*, a média da quantidade de URLs em cada um deles e a percentagem do número de *agregadores* comparada com um conjunto total de 14.277.910 URLs.

Agregador	Quantidade	Média	Percentagem
Nome da máquina	45.767	311,97	0,32 %
IP	6.187	2.309,73	0,04 %
Subrede	1.896	7.536,82	0,01 %

Tabela 3.2: *Agregadores* de URLs

Como se pode observar, qualquer um dos processos de agregação conduz a uma redução substancial de entidades, sendo a agregação pela subrede a mais eficaz. Posteriormente, veremos ainda que a criação de partições é realizado tendo como unidade, não o URL, mas sim o nome da máquina do servidor e o IP, introduzindo de antemão uma redução de informação na criação das tabelas de encaminhamento.

Em termos práticos, anexado à informação topológica de cada SIR é adicionada uma tabela de encaminhamento contendo um descritor do *agregador* e o SIRe a que se destina o URL contido nesse *agregador*. Outro aspecto interessante nesta abordagem é que, devido à natureza hierárquica da topologia, as tabelas de encaminhamento dos SIRs e SIRes descendentes são condensadas numa única entrada para criar as tabelas dos seus ascendentes, podendo surgir fusões de *agregadores* e eliminação de excepções.

Também no encaminhamento, cada SIR não necessita conhecer toda a topologia para

poder encaminhar os URLs correctamente, uma vez que se processa por saltos num número máximo do logaritmo do número total de entidades, afunilando-se o espaço de procura nas tabelas de encaminhamento à medida que os saltos se processam.

Criação das tabelas de encaminhamento

Apesar de ainda não se ter discutido o processo de partição (Capítulo 5), vamos para já assumir que, independentemente da forma como é concretizado, dispomos do resultado deste processo. O mecanismo de partição é hierárquico, no sentido que são efectuados vários níveis de partições em separado, de acordo com a topologia criada para o sistema SIRe. Observando a Figura 3.4 seriam criadas três partições do espaço total de URLs, correspondentes ao primeiro nível de SIRes (SIRe J, K e L) e, posteriormente, para cada um deles, mais três partições, apenas com o espaço de URLs resultante da partição do nível anterior. A informação de encaminhamento para o SIRes do primeiro nível fica, desde logo, definida, sendo a dos SIRes descendentes definida nos processos de partição respectivos. O interesse desta abordagem reside, precisamente, no englobamento das tabelas de encaminhamento dos SIRes descendentes nas tabelas dos SIRes ascendentes, evitando a necessidade de conhecimento global por parte de cada entidade responsável por encaminhar URLs.

O resultado do processo de partição devolve um conjunto de nomes de máquinas ou IPs para cada partição, ou seja, um par `(Conjunto<Máquinas>, Partição` n`)`, para cada partição n. Para simplificar ainda mais as tabelas de encaminhamento, este conjunto é transformado num conjunto de subredes por partição e eventuais excepções, isto é, `(Conjunto<Subredes>, Conjunto<Excepções>, Partição` n`)`.

Estas subredes são geradas a partir das gamas de endereços atribuídas aos fornecedores de serviço Internet pelos RIR (Regional Internet Registries). Espera-se, com este mecanismo, reduzir o tamanho das tabelas de encaminhamento, apesar do eventual substancial número de excepções e, simultaneamente, alargar temporariamente o intervalo de coincidência de URLs dos quais não se conhece ainda o seu destino para descarga.

A existência de múltiplas subredes pode, inclusivamente, conduzir à eliminação de outros blocos de subredes que se sobreponham, isto é, blocos contidos dentro de outros, e também à agregação (união) de blocos contíguos, aumentando, ainda mais, a simplificação obtida para as tabelas de encaminhamento.

Paralelamente ao mecanismo automático de geração de tabelas de encaminhamento

poderão existir restrições administrativas em cada um dos SIRes, que poderão, por exemplo, limitar o espaço dos URLs alvo, através da indicação de sub-domínios, gamas de IPs, restrições de conteúdo ou âmbitos geográficos dos servidores e do conteúdo.

Funcionamento do encaminhamento

Uma vez difundidas as tabelas de encaminhamento e complementadas com a informação topológica, o sistema fica apto a encaminhar URLs. Os robôs são instalados em cada um dos SIRs descarregando os URLs que lhes são atribuídos e encaminhando os URLs das ligações contidas nas páginas recém-descarregadas. Quando um URL é descoberto é consultada a tabela de encaminhamento, sendo retido o SIRe (ou SIR) de destino que faz coincidência com a subrede em que o IP do URL se encontra. O URL é enviado para o representante desse SIRe que, por seu turno, consulta novamente a sua tabela de encaminhamento para decidir se deve re-encaminhar para outro SIRe ou se o URL chegou ao seu destino. Note-se, que faz sentido utilizar um mecanismo de lotes em que se armazena um conjunto de URLs destinados para o mesmo SIRe, sendo posteriormente enviados simultâneo. Quando o destino do URL é um SIR, este é colocado em fila para descarga pelo robô desse SIR.

Para exemplificar melhor o encaminhamento voltemos a atenção para o sistema da Figura 3.4. Para tornar o exercício mais consistente, imaginemos que dispomos de um conjunto de URLs em que cada um dos seus IPs pertence a uma das gamas de IPs da rede de classe C, 193.136.0.0/16. Um URL do IP 193.136.1.1, outro de 193.136.2.1, e assim sucessivamente até 193.136.255.1. Para cada um destes endereços são retiradas do serviço *whois* as subredes a que pertencem, resultando um total de 105 subredes, um número já bastante inferior ao número total de endereços inicialmente (256).

Embora pouco realista, mas com fins didácticos, vamos supor que se realizou o processo de partição e foram atribuídas as subredes aos SIRes J, K e L da forma indicada na Tabela 3.3, com a indicação do número total de subredes na linha final.

Aplicando o processo de agregação de subredes a cada um dos conjuntos obtém-se o resultado patente na Tabela 3.4, onde se pode verificar a considerável redução da quantidade de subredes em cada um dos SIRes, sem qualquer perda de semântica.

Vamos agora realizar o processo de partição do espaço do SIRe L para os SIRes G, H e I. A tabela 3.5 visualiza a hipotética atribuição das subredes a cada um destes

SIRe J	SIRe K	SIRe L
193.136.0.0/24	193.136.84.0/22	193.136.176.0/21
193.136.1.0/24	193.136.90.0/23	193.136.184.0/24
193.136.2.0/24	193.136.92.0/23	193.136.185.0/24
193.136.3.0/24	193.136.94.0/23	193.136.186.0/23
193.136.4.0/24	193.136.96.0//21	193.136.188.0/22
193.136.5.0/24	193.136.104.0/22	193.136.192.0/24
193.136.6.0/24	193.136.108.0/23	193.136.193.0/24
193.136.7.0/24	193.136.110.0/24	193.136.194.0/23
193.136.8.0/24	193.136.111.0/24	193.136.196.0/22
193.136.9.0/24	193.136.112.0/24	193.136.200.0/24
193.136.10.0/24	193.136.113.0/24	193.136.201.0/24
193.136.11.0/24	193.136.114.0/24	193.136.202.0/24
193.136.12.0/24	193.136.115.0/24	193.136.203.0/24
193.136.13.0/24	193.136.116.0/24	193.136.204.0/24
193.136.14.0/24	193.136.117.0/24	193.136.205.0/24
193.136.15.0/24	193.136.118.0/24	193.136.206.0/24
193.136.16.0/22	193.136.119.0/24	193.136.207.0/24
193.136.20.0/24	193.136.120.0/21	193.136.208.0/22
193.136.21.0/24	193.136.128.0/21	193.136.212.0/24
193.136.22.0/24	193.136.136.0/24	193.136.213.0/24
193.136.24.0/21	193.136.137.0/24	193.136.214.0/24
193.136.32.0/21	193.136.138.0/23	193.136.215.0/24
193.136.40.0/22	193.136.140.0/23	193.136.216.0/22
193.136.44.0/24	193.136.142.0/24	193.136.220.0/22
193.136.46.0/23	193.136.143.0/24	193.136.224.0/22
193.136.48.0/21	193.136.144.0/23	193.136.228.0/24
193.136.56.0/21	193.136.146.0/24	193.136.230.0/24
193.136.64.0/22	193.136.147.0/24	193.136.231.0/24
193.136.68.0/22	193.136.148.0/24	193.136.232.0/23
193.136.72.0/23	193.136.149.0/24	193.136.235.0/24
193.136.74.0/24	193.136.150.0/24	193.136.236.0/24
193.136.76.0/22	193.136.151.0/24	193.136.237.0/24
193.136.80.0/22	193.136.152.0/22	193.136.238.0/24
	193.136.156.0/22	193.136.239.0/24
	193.136.160.0/21	193.136.240.0/21
	193.136.168.0/21	193.136.250.0/24
33	36	36

Tabela 3.3: Atribuição de subredes aos SIRes J, K e L

SIRes já depois da agregação realizada.

Por último, com o espaço de subredes do SIRe G, vamos aplicar o processo de partição para os SIRs 19, 20 e 21, apresentado as subredes resultantes na Tabela 3.6 já depois de agregadas.

Naturalmente, estamos a viciar o processo de partição uma vez que se estão a utilizar redes muito próximas que, conduzem facilmente a uma agregação maior. No entanto, este exemplo pretende visualizar a potencial simplificação alcançada e, acima de tudo, demonstrar o processo de encaminhamento. Com a atribuição das subredes pode-se agora construir as tabelas de encaminhamento e a informação topológica do sistema. Por facilidade de demonstração, apenas iremos utilizar os SIRs 1, 20 e 26 que, como iremos ver, são os participantes no processo de encaminhamento.

SIRe J	SIRe K	SIRe L
193.136.0.0/20	193.136.84.0/22	193.136.176.0/20
193.136.16.0/22	193.136.90.0/23	193.136.192.0/19
193.136.20.0/23	193.136.92.0/22	193.136.224.0/22
193.136.22.0/24	193.136.96.0/19	193.136.228.0/24
193.136.24.0/21	193.136.128.0/19	193.136.230.0/23
193.136.32.0/21	193.136.160.0/20	193.136.232.0/23
193.136.40.0/22		193.136.235.0/24
193.136.44.0/24		193.136.236.0/22
193.136.46.0/23		193.136.240.0/21
193.136.48.0/20		193.136.250.0/24
193.136.64.0/21		
193.136.72.0/23		
193.136.74.0/24		
193.136.76.0/22		
193.136.80.0/22		
15	6	10

Tabela 3.4: Subredes dos SIRes J, K e L obtidas depois do processo de agregação

SIRe G	SIRe H	SIRe I
193.136.176.0/20	193.136.203.0/24	193.136.230.0/23
193.136.192.0/21	193.136.204.0/22	193.136.232.0/23
193.136.200.0/23	193.136.208.0/20	193.136.235.0/24
193.136.202.0/24	193.136.224.0/22	193.136.236.0/22
	193.136.228.0/24	193.136.240.0/21
		193.136.250.0/24
4	5	6

Tabela 3.5: Subredes dos SIRes G, H e I obtidas depois do processo de agregação

As tabelas 3.7, 3.8 e 3.9 apresentam a informação de encaminhamento completa existente nos SIRs 1, 20 e 26.

Para concluir o nosso exemplo, vamos imaginar que o robô do SIR 1 encontrou o URL `http://www.ipb.pt/`, cujo IP é 193.136.95.224. Depois de consultar a sua tabela de encaminhamento, o SIR 1 encontra uma coincidência na entrada 193.136.192.0/19, que lhe indica que deve encaminhar o URL para o SIRe L. Com tabela de representantes, o SIR descobre que o representante do SIRe L é o SIR 26, podendo contactá-lo no endereço Nó 26. O SIR 26 recebe o URL e volta a procurar a sua tabela de encaminhamento encontrando na entrada 193.136.192.0/21 o destino SIRe G, cujo representante é o SIR 20 no Nó 20. O SIR 20 recebe o URL e verifica

SIR 19	SIR 20	SIR 21
193.136.176.0/21	193.136.188.0/22	193.136.196.0/22
193.136.184.0/22	193.136.192.0/22	193.136.200.0/23
		193.136.202.0/24
2	2	3

Tabela 3.6: Subredes dos SIRs 19, 20 e 21 obtidas depois do processo de agregação

ID:	SIR 1	
Ascendentes:	SIRe A	(SIR 1, SIR 2, SIR 3)
	SIRe J	(SIRe A, SIRe B, SIRe C)
	SIRe M	(SIRe J, SIRe K, SIRe L)
Representantes:	SIRe B:	SIR 4
	SIRe C:	SIR 7
	SIRe K:	SIR 14
	SIRe L:	SIR 26
Endereços:	SIR 1:	Nó 1
	SIR 2:	Nó 2
	SIR 3:	Nó 3
	SIR 4:	Nó 4
	SIR 7:	Nó 7
	SIR 14:	Nó 14
	SIR 26:	Nó 26
Tabela de Encaminhamento:	... :	SIR 1
	... :	SIR 2
	... :	SIR 3
	... :	SIRe B
	... :	SIRe C
	... :	SIRe K
	193.136.176.0/20 :	SIRe L
	193.136.192.0/19	
	193.136.224.0/22	
	193.136.228.0/24	
	193.136.230.0/23	
	193.136.232.0/23	
	193.136.235.0/24	
	193.136.236.0/22	
	193.136.240.0/21	
	193.136.250.0/24	

Tabela 3.7: Informação de encaminhamento para o SIR 1

na sua tabela de encaminhamento que o URL lhe é destinado através da entrada 193.136.194.0/23.

3.5 Construção de topologias

A construção da topologia SIRe é um processo administrativo e iterativo de sucessivas adições de um conjunto de entidades a outras entidades já definidas. De modo a que uma máquina possa fazer parte do sistema é necessário executar um conjunto de serviços, nomeadamente, o serviço SIRe, por exemplo, composto por um serviço de invocação remota (RMI)e um serviço de estruturas de dados distribuídas (DHT).

O primeiro passo para a criação de topologias implica a definição obrigatória da entidade SIR. A correcta definição das entidade pressupõe sempre a definição de um identificador da entidade lógica e a existência de um SIRe como topo da topologia.

Em termos conceptuais considerem-se as seguintes entidades:

ID:	SIR 20	
Ascendentes:	SIRe G	(SIR 19, SIR 20, SIR 21)
	SIRe L	(SIRe G, SIRe H, SIRe I)
	SIRe M	(SIRe J, SIRe K, SIRe L)
Representantes:	SIRe H:	SIR 22
	SIRe I:	SIR 26
	SIRe J:	SIR 2
	SIRe K:	SIR 14
Endereços:	SIR 2:	Nó 2
	SIR 14:	Nó 14
	SIR 19:	Nó 19
	SIR 20:	Nó 20
	SIR 21:	Nó 21
	SIR 22:	Nó 22
	SIR 26:	Nó 26
Tabela de	... :	SIR 19
Encaminhamento:	193.136.188.0/22 :	SIR 20
	193.136.192.0/24	
	193.136.193.0/24	
	193.136.194.0/23	
	... :	SIR 21
	... :	SIRe H
	... :	SIRe I
	... :	SIRe J
	... :	SIRe K

Tabela 3.8: Informação de encaminhamento para o SIR 20

- `Cadeia`. Uma cadeia de caracteres usada para definir, por exemplo, o identificador de uma entidade.
- `Nó`. A representação de uma máquina com o seu endereço IP, tendo associado um serviço de invocação remota.
- `EntidadeSIR`. A representação de uma entidade SIR.
- `EntidadeSIRe`. A representação de uma entidade SIRe.
- `EntidadeLogica`. A representação de uma entidade SIR ou SIRe.

A criação de uma topologia pode ser concretizada com duas famílias de operações: uma para definir a entidade e outra para realizar a junção de entidades. Cada operação consiste numa invocação remota tendo como origem a máquina de onde se pretende efectuar a operação, e destino uma das máquinas pertencentes à entidade lógica destino. Deste modo, definiram-se as seguintes operações, considerando n o `Nó` de destino da operação:

- `n.defineSIR(Cadeia id)`. Define um SIR com identificador `id` no nó de destino `n` .

ID:	SIR 26	
Ascendentes:	SIRe I	(SIR 25, SIR 26, SIR 27)
	SIRe L	(SIRe G, SIRe H, SIRe I)
	SIRe M	(SIRe J, SIRe K, SIRe L)
Representantes:	SIRe G:	SIR 20
	SIRe H:	SIR 22
	SIRe J:	SIR 2
	SIRe K:	SIR 14
Endereços:	SIR 2:	Nó 2
	SIR 14:	Nó 14
	SIR 20:	Nó 20
	SIR 22:	Nó 22
	SIR 25:	Nó 25
	SIR 26:	Nó 26
	SIR 27:	Nó 27
Tabela de Encaminhamento:	... :	SIR 25
	... :	SIR 26
	... :	SIR 27
	193.136.176.0/20 :	SIRe G
	193.136.192.0/21	
	193.136.200.0/23	
	193.136.202.0/24	
	... :	SIRe H
	... :	SIRe J
	... :	SIRe K

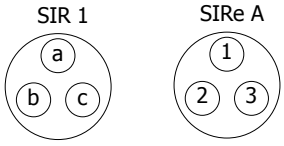
Tabela 3.9: Informação de encaminhamento para o SIR 26

- `n.defineSIRe(Cadeia id, EntidadeLogica e)`. Define um SIRe com identificador `id` na entidade `e`, que deve estar já definida na máquina destino.
- `n.adicionarNo(Node nl, Cadeia sirID)`. Adiciona a máquina `nl` (máquina local) ao SIR com identificador `sirID` definido na máquina de destino `n`.
- `n.adicionarSIR(EntidadeSIR sir, Cadeia sireID)`. Adiciona o SIR `sir` ao SIRe com identificador `sireID` na máquina de destino `n`.
- `n.adicionarSIRe(EntidadeSIRe sire, Cadeia sireID)`. Adiciona o SIRe `sire` ao SIRe com identificador `sireID` na máquina de destino `n`.
- `n.propagarSIR(EntidadeSIR sir)`. Esta operação permite efectuar a propagação da informação relativa a um novo SIR que foi adicionado a um SIRe. O parâmetro `sir` contém a informação relevante para a concretizar a propagação correctamente, nomeadamente, o identificador e o nó do SIR adicionado.
- `n.propagarSIRe(EntidadeSIRe sire)`. Esta operação permite efectuar a propagação da informação relativa a um novo SIRe que foi adicionado a outro SIRe, para o SIR ou SIRe que foi adicionado. O parâmetro `sire` contém a informação relevante para concretizar a propagação correctamente, nome-

adamente, o SIRe ao qual o anterior foi adicionado e os seus descendentes, juntamente com os representantes dos descendentes e os respectivos nós.

- `n.propagarSIReDentro(EntidadeSIRe sire)`. Esta operação permite efectuar a propagação da informação relativa a um novo SIRe que foi adicionado a outro SIRe, para os SIRes que já se encontram no sistema. O parâmetro `sire` contém a informação relevante para concretizar a propagação correctamente, nomeadamente, o identificador, o representante o nó deste, do SIRe que foi adicionado.

A título de exemplo, considere-se o SIR da Figura 3.5(a), sendo `a`, `b` e `c` nós. As operações para a criação do SIR 1, seriam : em a, `a.defineSIR("SIR 1")` e em b e c, `a.adicionarNo(nl, "SIR 1")`, em que `nl` seria o `Nó` da máquina a adicionar.



(a) (b)

Figura 3.5: Exemplo de construção de um SIRe simples.

A criação de um SIRe A seria realizada de forma análoga: em a, `a.defineSIRe("SIRe A")` e depois numa das máquina pertencentes a cada um dos outros SIRs, s_a`.joinSIRTo(sir, "SIRe A")`, em que `sir` seria o SIR a adicionar (Figura 3.5(b)).

A criação de uma estrutura mais elaborada poderia ser representada pela Figura 3.6. Neste caso, o SIRe A, definido nesse instante, irá ser adicionado a um SIRe F com a operação `g.adicionarSIRe(sire, "SIRe F")` realizada numa das máquina dos SIRe A, e em que `sire` é a representação do SIRe A e `g` é uma das máquinas do SIRe F.

Sempre que se processa uma operação de junção é necessário propagar a informação topológica para as entidades já existentes e para as que entram de novo.

Quando uma entidade `X` é adicionada, seja um SIR ou um SIRe, o nó ao qual ela se liga deve devolver a informação relativa ao sistema ao qual a entidade se vai adicionar. Neste caso, será enviada a informação dos SIRes irmãos (os descendentes do SIRe ao qual a nova entidade se adicionou) e todos os SIRes ascendentes e descendentes imediatos, juntamente com os seus representantes e endereços destes.

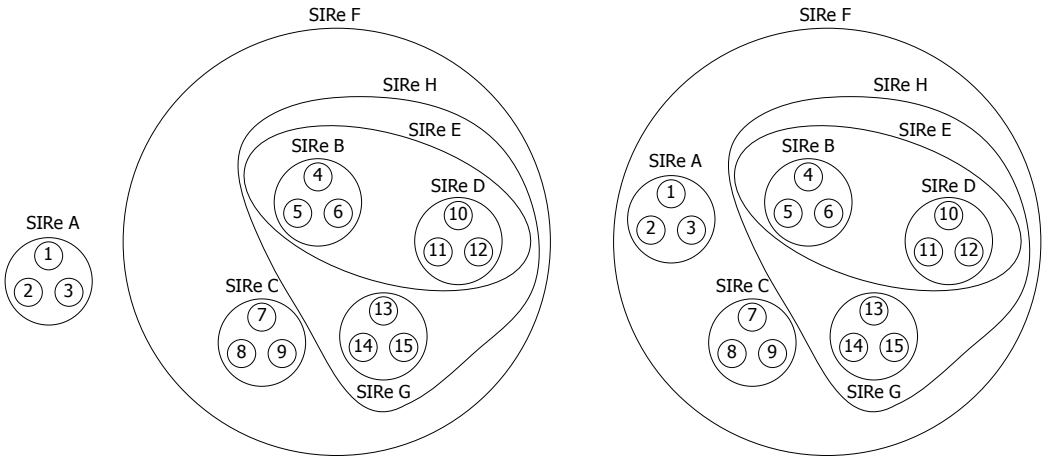


Figura 3.6: Exemplo de construção de um SIRe mais complexo.

O nó utilizado para iniciar a adição da entidade `X`, por sua vez, irá propagar a informação à sua parte da topologia, enviando a mesma informação aos irmãos e descendentes imediatos dos ascendentes, que por sua vez, farão o mesmo apenas para o ramo da topologia que lhes diz respeito.

A entidade que recebe o pedido de adição deve, igualmente, propagar a informação da entidade que se adicionou pela sua parte da topologia, da mesma forma que fez o nó que efectuou o pedido de adição. Neste caso, é propagada a entidade que se adicionou, juntamente com os seus representantes e nós respectivos.

A Figura 3.7 apresenta um exemplo de adição de um SIR a um SIRe, com a representação das operações de propagação envolvidas. Para simplicidade da figura, abdicou-se da representação de todos os nós, estando apenas visualizados um nó por SIR. Neste exemplo, existe um SIRe A com dois SIR, sendo a sua informação de encaminhamento a que se encontra nos rectângulos. Existe também definido um SIR 3, com intenções de se adicionar ao SIRe A. Para isso, no Nó 3 é invocada a operação `nó1.adicionarSIR(sir3, "SIRe A")`. O Nó 1 recebe o pedido, ao qual responde com a propagação resultante da adição deste novo SIR, da forma que se explicou anteriormente.

O exemplo da Figura 3.8 apresenta a adição de um SIRe D a um outro SIRe C. Com este exemplo pretende-se demonstrar a propagação em cadeia efectuada pelas entidades do sistema, que apesar de ser necessário um número de operações igual ao número de SIRs, a sua delegação distribuída permite concretizar a adição de entida-

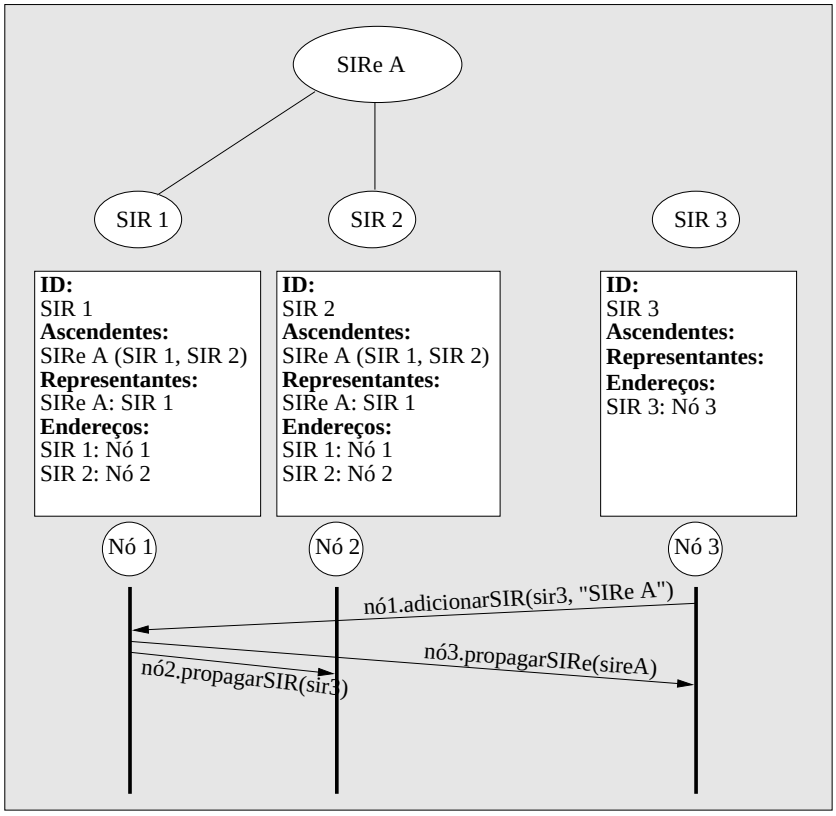


Figura 3.7: Exemplo de adição de um SIR

des, em topologias mais complexas e em tempos muito aceitáveis. Neste exemplo, o Nó 7 efectua a operação de adição `nó1.adicionarSIRe(sireD, "SIRe C")`. O Nó 1 responde, enviando uma operação de propagação para o nó da nova entidade (Nó 7) e para os nós das entidades imediatamente descendentes dos ascendentes a partir dele próprio, que neste caso seriam o Nó 4, Nó 3 e Nó 2. Os nós receptores verificam a necessidade de reencaminhar a propagação para os nós das entidades que fazem parte desse ramo da topologia, permitindo, desta forma, uma difusão completa de toda a informação topológica.

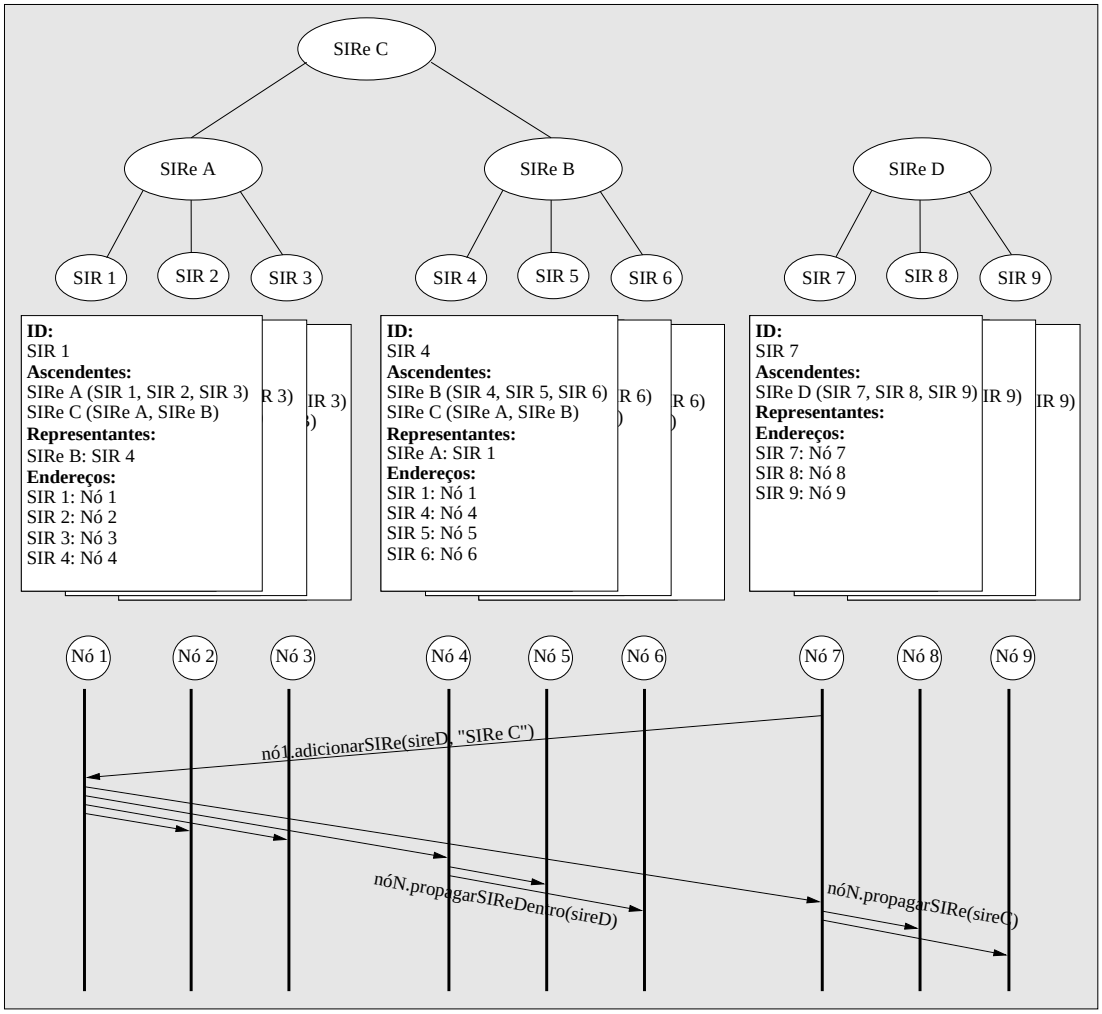


Figura 3.8: Exemplo de adição de um SIRe

Capítulo 4

Modelação da Internet

A arquitectura apresentada no capítulo anterior tem como base a utilização de mecanismos de divisão do espaço *Web* por um conjunto de entidades de descarga, que serão abordados no Capítulo 5. Estes mecanismos baseiam-se em informação adicional recolhida da infraestrutura de rede, da localização geográfica e da estrutura de hiperligações existentes entre as páginas *Web*.

Neste capítulo são apresentadas estatísticas dos dados recolhidos. É validada a cobertura da amostra utilizada comparada a população total de máquinas no domínio `pt` e a quantidade mínima de sondas a enviar a cada máquina para se obter um RTT significativamente estatístico. É ainda apresentado o modelo de aproximação do RTT para pares de máquinas para o quais não existe esse informação. E por fim, são analisadas as técnicas utilizadas para a determinação da localização geográfica.

A modelação da Internet tem como objectivo a recolha de informação topológica da infraestrutura de rede e medidas de distância fim-a-fim entre máquinas, de forma a estimar a latência real entre dois pontos e extrapolar os valores dessas distâncias entre pares de máquinas, cujo valor poderá ser desconhecido.

Por outro, esta recolha permitiu criar as condições necessárias para a elaboração de um sistema de simulação, no qual serão realizadas as experiências de descarga, utilizando topologias de rede e valores de tempos médios de ida e volta reais.

O processo de recolha da informação topológica de rede foi realizada recorrendo à ferramenta `traceroute` efectuando uma estimação ao nível dos caminhos fim-a-fim. A descrição detalhada do processo é efectuada na Secção 6.2.

A aquisição da informação geográfica foi obtida recorrendo a diversas fontes, tais

como, o DNS e bases de dados de outras aplicações, entre as quais, o GTrace [Periakaruppan 99], o MaxMind [MaxMind 07] e o NetGeo [CAIDA 02]. A descrição detalhada do processo é efectuada na Secção 6.2.

Um dos grandes desafios deste trabalho foi a sua validação com dados que se aproximassem o mais possível da realidade. A realização de experiências, num ambiente real, implicaria a implantação de uma grande quantidade de nós dispersos geograficamente, dada a natureza distribuída das técnicas utilizadas, o que resultaria numa infraestrutura de recursos materiais avultada e, para a qual não se dispunham desses recursos. Por outro lado, a experimentação, e a sua validação, seria dificultada devido às condições tipicamente instáveis da Internet, o que poderia implicar diferenças nos resultados para as mesmas experiências.

4.1 Estatísticas dos dados recolhidos

As experiências de recolha de informação topológica têm como base a utilização de um conjunto inicial de 16.152.324 de URLs sob o domínio de topo `pt`, recolhidos de duas colecções independentes da *Web* portuguesa, dos quais 13.248.199 do projecto NetCensus [Silva 02] e 3.775.611 do projecto WPT03 [Group 03]. Deste total de URLs foi necessário extrair os nomes e os endereços das máquinas servidoras, a fim de recolher a informação topológica física da rede, tais como, os encaminhadores intermédios, os RTTs, o número de saltos e a distância geográfica entre todos eles.

4.1.1 Entidades recolhidas

Em seguida apresentam-se alguns dados estatísticos das entidades recolhidas (Tabela 4.1). São apresentados valores para os nomes das máquinas e seus endereços físicos. As entidades nome de máquina e endereço de máquina foram, adicionalmente, subdivididas em entidades servidoras, por representarem um servidor *Web*, e entidades encaminhadoras, por resultarem da descoberta de encaminhadores de rede durante o processo de recolha de rotas.

A Figura 4.1 apresenta a distribuição das máquinas servidoras pelos países. É clara a predominância de máquinas em Portugal. Contudo apenas 45% se situam nesse país, considerando o âmbito dos servidores ser no domínio `pt`.

As Figuras 4.2, 4.3 e 4.4 apresentam a distribuição das rotas sobrepostas em mapas

Tabela 4.1: Quantidades das entidades identificadas no processo de recolha de dados.

Entidade	**Quantidade**		
	Encaminhadores	**Servidores**	**Total**
Nomes de máquinas	5.005	45.761	50.766
Endereços de máquinas (IPs)	4.654	6.176	10.830
Média de nomes por endereços de máquinas	1,08	7,41	4,69
Blocos de endereços	1.221	1.861	2.780
Agregados de endereços	666	669	1.175
Sistemas Autónomos	499	338	392
Cidades	312	342	488
Países	55	25	57

geográficos. Pode observar-se a elevada dispersão geográfica das rotas pelo globo terrestre.

4.2 Base estatística

A recolha de uma amostra de tamanho n de uma população de tamanho N é, tipicamente, realizada através de uma amostragem aleatória. Embora, no caso em estudo, a amostra não tenha sido escolhida aleatoriamente, por ser impossível dessa forma preservar a conectividade das hiperligações das páginas, considerou-se a selecção das máquinas semi-aleatória por serem originadas a partir de uma semente inicial aleatória. Em termos de recolha dos RTT, o processo é implicitamente aleatório condicionado pelo estado da rede no momento da recolha.

A seguir apresenta-se a base estatística que permita o cálculo da margem de erro obtida na estimação da média de população, a partir de uma amostra, e o cálculo da dimensão ideal da amostra, aproximando a média da população pela média da amostra, assumindo um determinado factor de confiança.

Os cálculos a seguir apresentados, utilizam o RTT como variável aleatória. Tendo sido observado o não ajustamento da amostra à distribuição normal, assume-se que esta também não é seguida pela população, podendo ser feita uma aproximação à distribuição normal das médias de todas as amostras possíveis de recolher, desde que $n \geq 30$ e, no caso de populações consideradas infinitas, se verifique, $20n \leq N$ [Murteira 02].

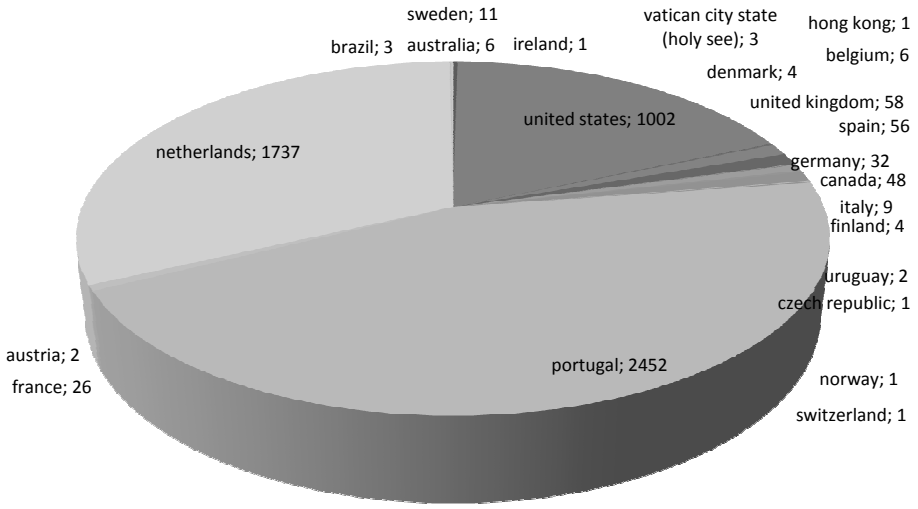


Figura 4.1: Distribuição das máquinas servidoras por país

Considere-se uma variável aleatória cuja população tem média, μ, e desvio padrão, σ, de tamanho N. Se a dimensão da amostra, n, for suficientemente grande (30 é um valor considerado suficientemente grande), então a distribuição das médias das amostras ($\overline{X}$) tendem a ser normais, com média μ e desvio padrão $\sqrt{\frac{\sigma^2}{n}\left(\frac{N-n}{N-1}\right)}$, independentemente da distribuição da população [Martins 05].

$$\overline{X} \sim N\left(\mu, \sqrt{\frac{\sigma^2}{n}\left(\frac{N-n}{N-1}\right)}\right) \tag{4.1}$$

Se $\overline{X}$ segue aproximadamente uma distribuição normal (Equação 4.1) então Z segue uma distribuição normal reduzida de média 0 e variância 1 (Equação 4.2).

$$Z = \frac{\overline{X} - \mu}{\sqrt{\frac{\sigma^2}{n}\left(\frac{N-n}{N-1}\right)}} \sim N(0,1) \tag{4.2}$$

Se considerarmos que a diferença entre a média da amostra, $\overline{X}$, e a média da população, μ, é a margem de erro, podemos calculá-la com um nível de confiança associado a Z (retirado da tabelas da distribuição normal) através da Equação 4.3.

$$\epsilon = \frac{Z\sigma}{\sqrt{n}}\sqrt{\frac{N-n}{N-1}} \tag{4.3}$$

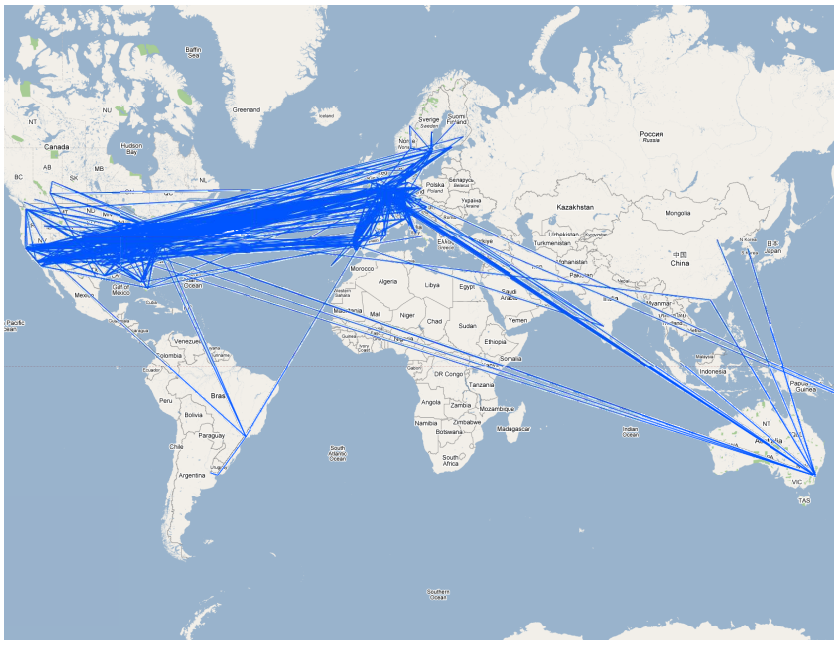


Figura 4.2: Distribuição das rotas pelo mundo

Para o cálculo da dimensão ideal da amostra, utilizando 4.3, e resolvendo em ordem a n obtém-se o tamanho da amostra (Equação 4.4)

$$n = \frac{NZ^2\sigma^2}{\epsilon^2(N-1) + Z^2\sigma^2} \tag{4.4}$$

No caso de a população ser extremamente grande (tipicamente $20n \leq N$), em vez da Equação 4.2, a média das amostras segue a distribuição da Equação 4.5.

$$Z = \frac{\overline{X} - \mu}{\frac{\sigma}{\sqrt{n}}} \sim N(0,1) \tag{4.5}$$

Agora o cálculo da margem de erro é feito pela Equação 4.6.

$$\epsilon = \frac{Z\sigma}{\sqrt{n}} \tag{4.6}$$

Nas mesmas condições para o tamanho da amostra deve utilizar-se a Equação 4.7.

$$n = \frac{Z^2\sigma^2}{\epsilon^2} \tag{4.7}$$

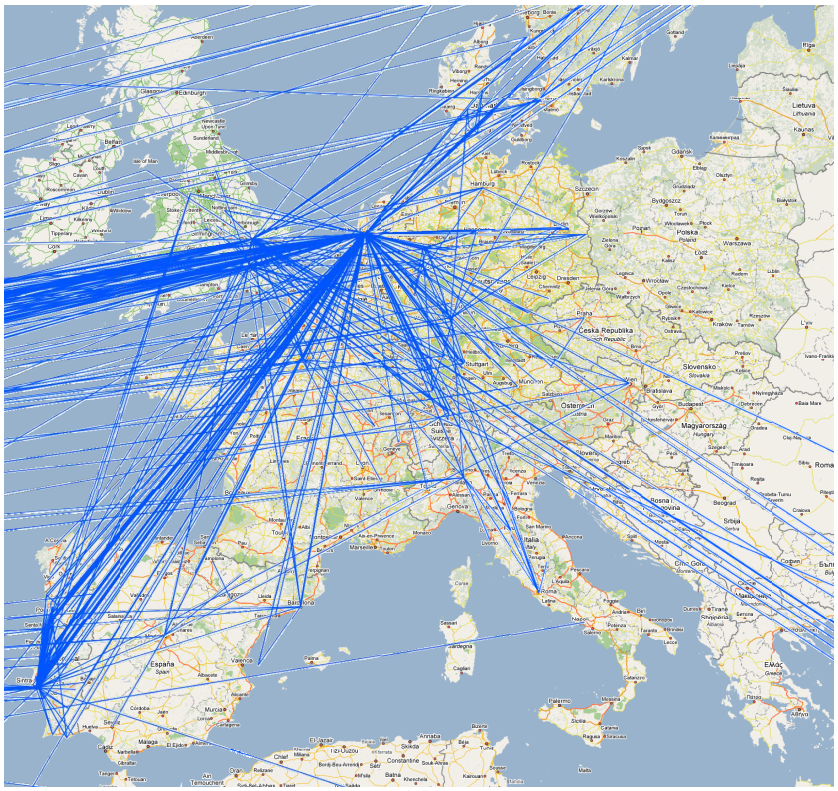


Figura 4.3: Distribuição das rotas pela Europa

Em qualquer das situações, seja uma população finita ou infinita, para o cálculo da margem de erro cometido, é necessário indicar:

1. O nível de confiança através de Z.
2. O desvio padrão da população, σ, se conhecido, ou aproxima-lo através do desvio padrão de uma amostra.
3. O tamanho da amostra.

Para o cálculo do tamanho ideal da amostra é necessário indicar:

1. O nível de confiança através de Z.
2. A margem de erro máxima admissível, através da diferença entre a média real da população (μ) e a média da amostra ($\overline{X}$), $\epsilon = \overline{X} - \mu$.

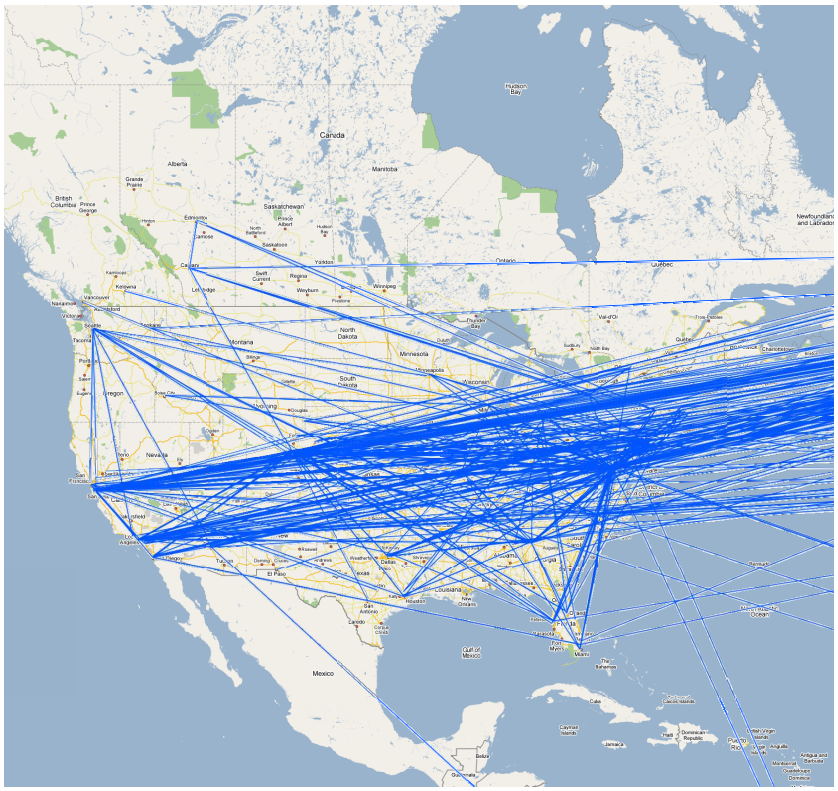


Figura 4.4: Distribuição das rotas pela América do Norte

3. O desvio padrão da população, σ, se conhecido, ou aproxima-lo através do desvio padrão de uma amostra piloto.

4.3 Cobertura da quantidade de máquinas utilizadas

No que respeita à recolha de informação topológica, a validação da quantidade de máquinas utilizadas na amostra ficou dependente da colecção de páginas inicialmente utilizada, donde resultaram 45.761 nomes de máquinas válidas.

Tentou verificar-se se este número de máquinas servidoras corresponde a uma cobertura aceitável das máquinas servidoras de *WWW* existentes na Internet. No entanto, das diversas empresas que realizam estudos sobre a evolução da Internet, tais como, a Netcraft [Netcraft 07], o Internet Systems Consortium [ISC 07], ou al-

gum dos Registos Regionais da Internet (RIR), não foi possível encontrar um valor exacto para a quantidade de máquinas no domínio `pt`. A informação encontrada disponível foi apenas a quantidade de máquinas servidores a nível mundial, na Netcraft, e a quantidade de máquinas servidores a nível mundial e do domínio `pt`, mas limitadas à região administrada pelo RIPE [NCC 07].

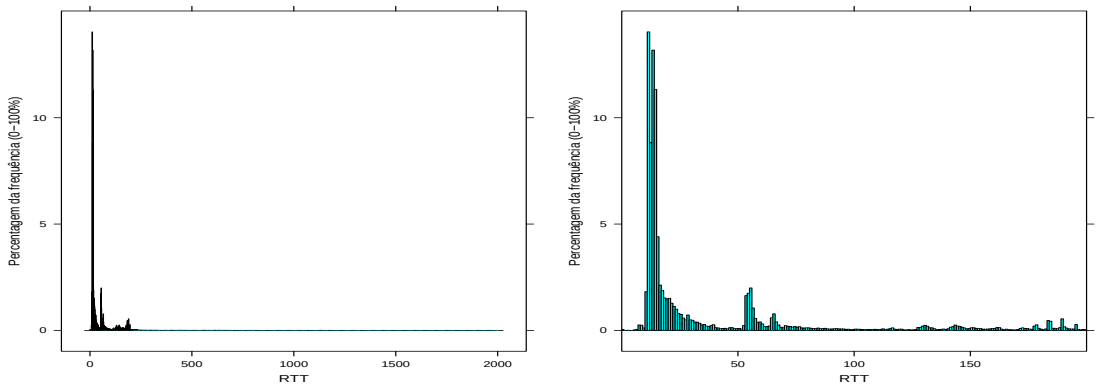
Deste modo, tentou induzir-se a quantidade total de máquinas do domínio `pt`, observando a percentagem do total de máquinas da região da RIPE em relação ao total de toda a Internet. Utilizaram-se valores correspondentes a Junho de 2003, altura mais próxima e posterior à recolha das colecções de URLs apresentadas. Nesse mês, a Netcraft reportou um total de 40.936.076 de máquinas servidoras, contra um total de 6.268.702 reportadas pelo RIPE na sua região. Para o domínio `pt`, a RIPE reportou 8.298 máquinas na sua região. Realizando a extrapolação da quantidade de máquinas do domínio `pt` da região administrada pela RIPE para toda a Internet, resulta um total de 54.188 máquinas servidoras. Comparando este número com a quantidade de máquinas utilizadas nas recolhas (45.761), obtém uma cobertura de aproximadamente 85%.

4.4 Quantidade de sondas a enviar para cada máquina

Uma questão que influencia fortemente a precisão do RTT entre duas máquinas é a quantidade de vezes que uma máquina deveria ser sondada. Na realidade, o tráfego na Internet é muito imprevisível devido a padrões de utilização variáveis por determinados períodos de tempo, variação nas políticas de encaminhamento, ou, simplesmente, a criação ou eliminação de máquinas ou encaminhadores.

As Figuras 4.5(a) e 4.5(b) apresentam a distribuição dos RTTs recolhidos, que como se pode observar, segue uma lei da potência e é bi-modal.

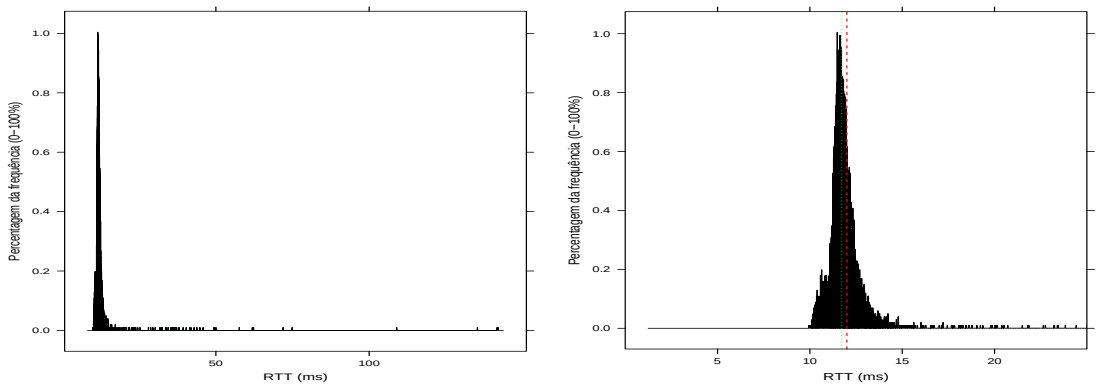
Durante o processo de recolha dos RTT, cada sonda, quando é enviada, atravessa uma sequência de encaminhadores até alcançar o seu destino. O RTT devolvido é o tempo médio de ida e volta desde a origem até ao encaminhador intermédio ou destino final. Sendo o RTT uma medida tão variável e instável, decidiu-se analisar o comportamento dos RTT para um destino fixo. Desta forma, enviaram-se 10.056 sondas a partir do endereço 193.136.195.95, localizado em Bragança, para o endereço 194.210.0.18, localizado em Lisboa. A Figura 4.6(a) apresenta a distribuição



(a) (b)

Figura 4.5: Distribuição dos RTT.

dos RTT observados. Pode observar-se a tendência notoriamente enviesada com uma aproximação a uma lei da potência. A Figura 4.6(b) apresenta em detalhe os tempos de maior densidade entre 0 e 25 ms, onde se pode verificar que a tendência central aproxima-se mais da mediana (traço picotado com 11,72 ms) do que da média (traço interrompido com 12,00 ms). Este comportamento reforçou a necessidade de utilização de uma medida de tendência central, imune ao enviesamento das distribuições dos RTT, como é o caso da mediana.



(a) (b)

Figura 4.6: Distribuição dos RTT para o endereço 194.210.0.18.

Observou-se, ainda, a variação dos RTT ao longo de um dia. Como se pode verificar pela Figura 4.7, os desvios acentuados de RTT em relação à mediana (traço verde horizontal) são ocasionais, sugerindo pequenas anomalias resultantes da eventual

congestão da rede ou alterações dinâmicas de rota.

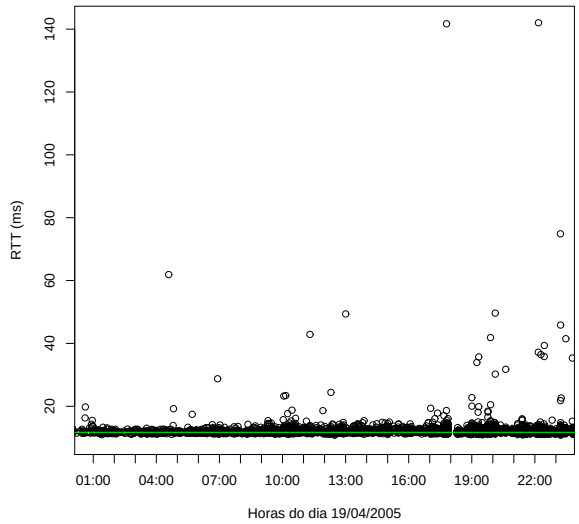


Figura 4.7: Variação dos RTT ao longo de um dia para o endereço 194.210.0.18.

Tendo em conta que o desvio padrão da amostra é de 3,21 ms, a margem de erro para a estimação da média é apenas de 0.08, com um nível de confiança de 99%.

Claramente, o erro desta grande amostra é bastante pequeno, tendo em conta o seu tamanho. Mas, numa situação real, não seria conveniente extrair tal quantidade de amostras, por ser necessário enviar sondas para uma grande quantidade de destinos. Idealmente, a quantidade de amostras para o mesmo endereço deveria ser o mais baixo possível, de modo a reduzir o tempo necessário para a recolha de uma quantidade vasta de RTTs. No entanto, para valores da amostra abaixo de 30, a estimação da média deixa de ser válida, por estarmos perante uma população, claramente, não normal.

Deste modo, tentou-se calcular a margem de erro da mediana dos destinos cuja quantidade de amostra se situa abaixo de 30, comparada com a mediana dos mesmos destinos com uma quantidade de amostras acima de 30. O exercício é realizado para cada destino, sendo depois calculada a média da margem de erro normalizada de todos os destinos, fixando uma determinada quantidade de amostras abaixo de 30.

Considere-se uma amostra de RTTs, T_d, para um determinado destino d, com tamanho maior ou igual do que 30 e com mediana $\eta(T_d)$. Considere-se ainda, para o mesmo destino d, as amostras com tamanho $t < 30$, P_{d_t}, cuja mediana é $\eta(P_{d_t})$. Para esse destino calcula-se a margem de erro normalizada entre as duas medianas,

$\epsilon_{d_t} = \frac{|\eta(T_d) - \eta(P_{d_t})|}{\eta(T_d)}$. De seguida é calcula a média das margens de erro de todos os destinos, fixando t, $m_t = \frac{1}{n}\sum_{i=1}^{n} \epsilon_{i_t}$, em que n é quantidade de destinos com amostras de tamanho maior do que 30.

O resultado é apresentada na Figura 4.8. Como se pode observar pelo gráfico a partir de amostras com tamanho 7, a margem de erro tende a estabilizar, sendo nesse ponto de 0,08.

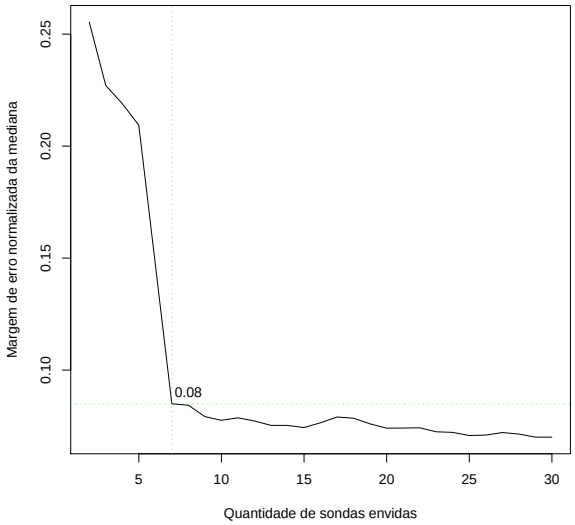


Figura 4.8: Média da margem de erro para a mediana.

4.5 Aproximação do RTT para pares desconhecidos

Na secção anterior validou-se a quantidade de sondas a enviar para cada um dos destinos. No entanto, a inexistência de um grafo completo entre todos os destinos levou ao cálculo de uma aproximação entre dois pontos, no grafo criado com as rotas recolhidas.

Como já foi referido anteriormente, são vários os trabalhos já publicados no sentido de concretizar essa aproximação, sendo, tipicamente, o objectivo principal a criação de modelos que permitam representar as rotas seguidas, efectivamente, pelos pacotes na Internet [Leguay 05] e utilizando como métrica de estudo o número de saltos entre dois pontos. No entanto, para o contexto deste trabalho, o objectivo não é modelar

os caminhos seguidos pelos pacotes, mas sim aproximar os RTT entre dois pontos com os RTT reais, num intervalo de tempo limitado.

O procedimento seguido tem como base a geração do grafo da Internet a partir dos RTTs recolhidos, em que os vértices são representações de máquinas físicas, aos quais lhes é atribuído o seu endereço como etiqueta, e os ramos são troços das rotas reais entre duas máquinas, aos quais lhes é atribuído um RTT como peso. Seguidamente, é calculado o caminho mais curto em termos de RTT entre todos os vértices considerados servidores *Web*, e para quais se pretende realizar esta aproximação.

Na mediação dos RTT para um destino, são recolhidos os RTT até aos encaminhadores pelos quais os pacotes atravessam a rede. No entanto, em algumas situações acontece que um encaminhador mais distante obtém um RTT mais baixo do que um mais próximo, resultando numa diferença negativa. Nestes casos, optou-se por truncar esses valores para zero, resultando somas nas diferenças dos RTTs de um ponto e o anterior, num valor maior do que o RTT efectivo até esse destino.

Para analisar o primeiro erro, calculou-se a diferença entre o RTT medido e a soma das diferenças dos RTT em cada salto com o anterior, designado por δ. A Figura 4.9 apresenta a função de distribuição acumulada dessa diferença, onde se pode constatar que, aproximadamente, 90% das diferenças estão próximo de zero, existindo no entanto um conjunto de 10% de diferenças com valores que chegam a atingir os 2500 ms.

A medição do RTT para um destino é feito através o envio sucessivo de pacotes para o destino com TTL crescente, iniciado em 1. Em cada envio, as rotas seguidas pelos pacotes podem diferir devido às políticas de encaminhamento. Outra justificação plausível é a ocorrência de congestionamento esporádico, donde poderá resultar um RTT diferente entre duas sondas enviadas.

4.6 Determinação da localização geográfica

Para determinar a localização geográfica dos endereços IP foram utilizadas diversas fontes de informação, baseadas nos algoritmos utilizados no GTrace [Periakaruppan 99] e no projecto NetGeo [CAIDA 02].

O GTrace utiliza um conjunto de heurísticas que determina a localização geográfica utilizando o registo LOC do DNS e fazendo um conjunto de equivalência a códigos

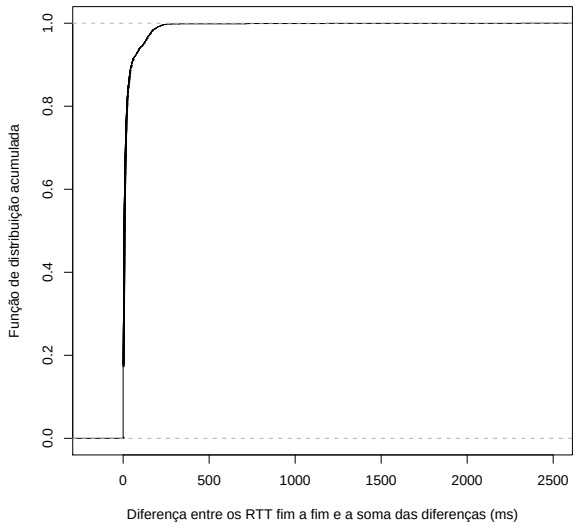


Figura 4.9: Função de distribuição acumulada para a diferença entre os RTT fim a fim real e a soma das diferenças em cada salto.

de aeroportos, nomes de cidades e instituições colocadas no nome das máquinas a pesquisar. Por fim, utiliza ainda o NetGeo.

Dado o contexto geográfico em que o GTrace foi desenvolvido, e tendo em conta a inserção geográfica do trabalho aqui desenvolvido, optou-se por efectuar actualizações nos dados do GTrace, adequados para Portugal, ao nível das cidades e instituições.

O NetGeo contém uma base de dados com localizações geográficas determinadas a partir da análise das moradas contidas nos registos do serviço `whois`, que, apesar de já não se encontrar em actualização, optou-se por utilizar. Permaneceu, contudo, a consciência da necessidade de implementar um analisador semelhante ao NetGeo, que, infelizmente, não foi realizado.

Para se dispor de uma nomenclatura geográfica uniforme recorreu-se ainda ao serviço da US National Geospatial Intelligence Agency [Agency 06] para retirar nomes e códigos dos 13 países dominantes encontrados nos endereços IP, e respectivas localidades e coordenadas geográficas, num total de 659.048 localizações

Assim sendo, para cada máquina encontrada foram pesquisas a seguintes heurísticas de localização geográfica:

1. DNS. Localização geográfica devolvida pelos registos do DNS (DNS LOC RR).

2. Aeroporto. Localização geográfica com base em códigos de aeroportos existen-

tes nos nomes das máquinas.

3. Cidade. Localização geográfica com base no nome da cidades existentes nos nomes das máquinas.

4. Instituição. Localização geográfica com base no nome de instituições existentes nos nomes das máquinas.

5. GTrace. Localização geográfica recolhida na base de dados utilizada pelo GTrace composta por IPs e nomes de máquinas já previamente localizados.

6. NetGeo. Localização geográfica consultada na base de dados do NetGeo.

A Tabela 4.2 apresenta a quantidade de IPs para os quais foi possível descobrir a sua localização:

Tabela 4.2: Quantidade de IPs localizados por diferentes heurísticas.

# IPs		11021		4627	41,98%	6394	58,02%
Heurística		**Todos IPs**	**%**	**IPs Encaminhadores**	**%**	**IPs Servidores**	**%**
DNS	1	160	1,45	154	3,33	6	0,09
Aeroporto	2	213	1,93	213	4,60	0	0,00
Cidade	3	205	1,86	205	4,43	0	0,00
Instituição	4	1165	10,57	58	1,25	1107	17,31
GTrace	5	91	0,83	91	1,97	0	0,00
NetGeo	6	9731	88,30	4125	89,15	5606	87,68

Como se pode observar, uma grande percentagem dos IPs é localizado com a heurística 6. Para os IPs servidores as heurísticas 2, 3 e 5 não obtiveram qualquer localização. A Tabela 4.3 apresenta a quantidade de IPs localizados simultaneamente por duas heurísticas (Intersecção), o número de localizações coincidentes, a percentagem de coincidentes em relação aos interceptados e a distância média entre as localizações interceptadas, separando os IPs encaminhadores e servidores.

A quantidade de IPs resultantes da intersecção não é muito significativa. No entanto, podemos considerar ligeiramente significativos os pares 1-6 (137 intersecções), 2-6 (162 intersecções), 3-6 (199 intersecções), 4-6 (1125 intersecções) e 5-6 (86 intersecções). A heurística 6, efectivamente, localiza uma grande percentagem de IPs, contudo a sua precisão é bastante duvidosa, devido à descontinuidade da sua

utilização. Repare-se que para as heurísticas 1 e 2 que, supostamente pela sua natureza, são heurísticas com boa precisão, a percentagem de localizações coincidentes em relação à 6 é muito baixa, o que releva uma precisão baixa da heurística 6 .

Tabela 4.3: Intersecção e coincidência de localizações entre diferentes pares de heurísticas.

		Todos os IPs					
A	**B**	**Intersecção**		**Localização Coincidente**		**Coincidentes / Intersectados**	**Distância Média**
		#	%	#	%	%	**Km**
1	2	0	0,00	0	0,00	0,00	0,00
1	3	0	0,00	0	0,00	0,00	0,00
1	4	1	0,63	1	0,63	100,00	0,00
1	5	6	3,75	1	0,63	16,67	224,36
1	6	137	85,63	7	4,38	5,11	1709,14
2	3	0	0,00	0	0,00	0,00	0,00
2	4	0	0,00	0	0,00	0,00	0,00
2	5	0	0,00	0	0,00	0,00	0,00
2	6	162	76,06	5	2,35	3,09	2059,79
3	4	0	0,00	0	0,00	0,00	0,00
3	5	0	0,00	0	0,00	0,00	0,00
3	6	199	97,07	41	20,00	20,60	622,21
4	5	2	0,17	2	0,17	100,00	0,00
4	6	1125	96,57	521	44,72	46,31	953,04
5	6	86	94,51	4	4,40	4,65	4069,95

Para as heurísticas 3 e 4, obteve-se uma percentagem de coincidentes considerável (20,6 % e 46,31 %, respectivamente). Como não existem localizações entre estas duas heurísticas e as heurísticas 1 e 2, não se pode concluir nada acerca da sua precisão. Assumindo, hipoteticamente, que a precisão das heurísticas 3 e 4 seria boa, significa que a precisão da heurística 6 é no máximo próxima de 46% para um universo de 1125 IPs.

Capítulo 5

Estratégias de Partição

Neste capítulo são descritas as estratégias de partição utilizadas para a descarga eficiente da *Web*.

As estratégias tradicionais de partição por função de dispersão são tipicamente consideradas adequadas ao seu objectivo, devido à autonomia existente em cada robô para a tomada de decisão de encaminhamento. Quando a função de dispersão é aplicada ao nome da máquina do URL, é garantido que apenas um agente descarrega páginas de um único sítio *Web*, não se garantindo essa exclusividade na máquina física. Além disso, devido à grande incidência de ligações para o mesmo sítio *Web*, é reduzida a sobrecarga de comunicação no reencaminhamento de URLs para outros agentes.

Apesar de robustos, devido à sua leveza e descentralização, estes esquemas não tomam em consideração, informação útil que poderia ser retirada da Internet e da *Web*, no sentido de criar alternativas optimizadas de descarga.

Revendo os objectivos principais de uma descarga distribuída podem enumerar-se os seguintes: 1) reduzir os tempos de descarga das páginas *Web*; 2) minimizar a sobrecarga de comunicação na sincronização dos agentes; 3) balancear o trabalho pelos agentes.

Paralelamente à descarga das páginas é possível recolher dados adicionais utilizados para a criação de modelos da topologia da Internet e da *Web*. Assume-se que o sistema é iniciado com um conjunto inicial de URLs e, para os quais, é gerada uma primeira modelação. Durante a recolha incremental de URLs são, também, recolhidos os dados necessários para actualizar o espaço *Web* já conhecido e acrescentar a informação de modelação para os novos URLs descobertos.

A actualização dos modelos para efeitos de encaminhamento é efectuada em estágios temporais através do envio de lotes de informação para as entidades responsáveis pela criação das tabelas de encaminhamento (*Particionadores*). Com este mecanismo podem surgir momentos de incapacidade de encaminhamento, quando são encontrados URLs para os quais não existem informação de encaminhamento. Nestas situações, estes URLs podem ser reservados até à ocorrência de um novo estágio de actualização, ou ser encaminhados por função de dispersão, de modo a distribuir a carga pelo sistema.

Para optimizar o tempo de descarga, e simultaneamente minimizar a sobrecarga de comunicação, são utilizadas medidas de proximidade entre os agentes e os servidores *Web*. A distância é calculada por uma medida de tendência central dos RTT entre estas entidades, utilizando as ferramentas de descoberta de topologias como o `traceroute`. Podem ainda ser utilizados dados geográficos que permitam a determinação de distâncias entre os robôs e servidores.

Durante a extracção dos URLs nas páginas descarregadas, é também contabilizada a quantidade de hiperligações entre as páginas. Estas quantidades são agregadas no número de hiperligações entre as máquinas que as albergam, sendo utilizadas como de medida de peso dos ramos dos grafos construídos.

Além disso, é ainda considerado o balanceamento do trabalho entre os agente, de forma a lhes ser atribuído um conjunto de URLs para descarga equilibrado. Este balanceamento tem em linha de conta não só a quantidade de URLs que cada robô deve descarregar, mas, também, a quantidade simultânea de conexões que cada um pode executar. Adicionalmente, é ainda considerada a distribuição da lei da potência da quantidade de páginas pelos servidores, de modo a não serem geradas partições em que um agente fica responsável por um número reduzido de servidores, constrangendo, desta forma, o desempenho desse agente.

A informação recolhida acerca dos RTTs, das localizações geográficas, das hiperligações e da quantidade de páginas por servidor é utilizada para a geração de modelos, representados por grafos. A cada grafo é aplicado um algoritmo de partição e calculado o seu nível de balanceamento (balanço) e o somatório dos pesos inter-partição (corte de ramo), utilizado para normalizar os pesos de um terceiro grafo criado a partir da combinação das medidas recolhidas.

5.1 Partição de grafos

A partição de grafos visa dividir os vértices de um grafo em partes aproximadamente iguais, de tal forma que, a soma dos pesos dos ramos que atravessam partes diferentes são minimizados. Dado um grafo $G = (V, E)$, em que, $|V| = n$, a partição de V em k subconjuntos, $V_1, V_2, \ldots, V_k$, é tal que $V_i \cap V_j = \emptyset, \forall_{i \neq j}$, $|V_i| = n/k$, $\bigcup_i V_i = V$, e $\min \sum_{j \in cut} w_j^{\mathrm{e}}$, em que *cut* é um subconjunto dos ramos de E cujos vértices incidentes pertencem a subconjuntos diferentes, sendo w_j^{e} o peso do ramo j. $\sum_{j \in cut} w_j^{\mathrm{e}}$ é definido como o corte dos ramos.

Os problemas de partição de grafos são NP-completos e, por esse motivo, foram desenvolvidos diversos algoritmos que encontram soluções aceitáveis em tempos razoáveis, recorrendo a determinadas heurísticas.

Uma das heurísticas mais comuns, tipicamente utilizada em diversos algoritmos, é a partição por bissecção. A concretização da divisão de um grafo em k partições é alcançado após $log(k)$ bissecções sucessivas [Karypis 98b].

Os mesmos autores subdividem os algoritmos de partição em três categorias:

1. Espectrais;
2. Geométricos;
3. Combinatórios.

Os métodos de partição espectrais [Pothen 90, Pothen 92] baseiam-se no cálculo de vectores próprios, designados de vector de Fiedler [Barnard 94]. Contudo, este cálculo é considerado computacionalmente pesado, mesmo utilizando técnicas de bissecção espectral (MSB), quando comparado com outras técnicas [Karypis 98b].

Os métodos de partição geométricos carecem da utilização de um esquema de mapeamento de coordenadas que, para a situação em questão, não se encontram disponíveis e, por isso, não serão alvo de análise.

Uma das soluções mais utilizadas é a partição multi-nível utilizando algoritmos combinatórios, que resolve o problema da partição, transformando o grafo inicial em grafos mais pequenos através de uma técnica de engrossamento (coarsening) sucessivo dos vértices. Posteriormente, é aplicado um algoritmo de partição ao grafo mais pequeno, cujas partições resultantes são projectadas para os níveis superiores até ao grafo inicial [Karypis 98c]. Esta técnica é designada de desengrossamento (uncoarsening).

As vantagens deste método são enormes, pois o algoritmo de partição propriamente dito é aplicado a grafos apenas com uma centena de vértices. O esquema partição utilizado nesta fase varia, podendo ser utilizados métodos espectrais ou métodos combinatórios.

Um dos métodos combinatórios tipicamente utilizados é uma adaptação do algoritmo Kernighan-Lin (KL) [Kernighan 70], descrito em [Fiduccia 82](FM), com alguns melhoramentos, ao nível das estruturas de dados utilizadas. Para este trabalho foi realizada uma implementação proprietária do algoritmo que reflectisse algumas restrições enumeradas mais à frente.

Basicamente, o algoritmo KL inicia-se com uma partição inicial sendo iterados todos os movimentos possíveis dos vértices de uma partição para as restantes, em que se atinge um corte de ramos mais baixo. Quando é descoberta uma sequência que reduz o corte dos ramos, os movimentos são efectuados, iniciando-se novamente o algoritmo. Quando não existir uma diminuição no corte dos ramos, o algoritmo termina.

Numa fase inicial do trabalho tentou-se utilizar a ferramenta Metis [Karypis 98a], adequada para a partição de grafos multi-nível. No entanto, a natureza do problema e dos grafos utilizados, impediu a sua aplicação.

Um dos requisitos para a partição do espaço *Web* é a fixação de um ou mais vértices a determinadas partições, em particular, os robôs. Além disso, é necessário garantir que não existam mais do que um robô em cada partição ou que uma partição não esteja alocada a pelo menos um robô.

As características próprias dos grafos utilizados, levou à criação de um conjunto de heurísticas mais adequadas, nomeadamente, na criação das partições iniciais e nas condições de movimentação dos vértices durante a execução do algoritmo KL.

Para resolver a partição multi-objectivo, resultante da utilização dos dois critérios de pesagem dos ramos, os RTTs e as hiperligações, utilizou-se um procedimento sugerido em [Schloegel 99].

Neste procedimento, para cada grafo representante de cada objectivo é aplicado o algoritmo de partição separadamente. Posteriormente, é criado um outro grafo em que os pesos dos ramos são calculados com base numa soma pesada dos pesos de cada objectivo, normalizados pelo corte dos ramos associado ao grafo correspondente.

Esta soma pesada permite a inclusão de um factor preferencial, de modo a manipular a importância com que cada objectivo é aplicado no algoritmo de partição. A

configuração final das partições é obtida pela partição deste último grafo.

Em termos matemáticos, considerando k objectivos e k grafos, um para cada objectivo, $G_1 = (V_1, E_1), G_2 = (V_2, E_2), \ldots, G_k = (V_k, E_k)$, depois da partição de cada um deles, obtém-se os cortes de ramos : $\mathrm{ec}_1, \mathrm{ec}_2, \ldots, \mathrm{ec}_k$.

O grafo combinado $G_c = (V_c, E_c)$ em que $V_c = \bigcup_{i=1}^{k} V_i$ e $E_c = \bigcup_{i=1}^{k} E_i$, onde cada E_c tem um peso de $w_c^e = \sum_{i=1}^{k} \frac{p_i w_i^e}{\mathrm{ec}_i}$, sendo p_i o factor preferencial de cada objectivo i.

As características intrínsecas da *Web*, bem como a organização topológica das suas páginas em nomes de máquinas e máquinas físicas, permite, a priori, a concretização de dois níveis de engrossamento realizados à medida. Este conhecimento específico dos grafos em causa, permite controlar a fase inicial de engrossamento e decidir o ponto de paragem do processo de desengrossamento.

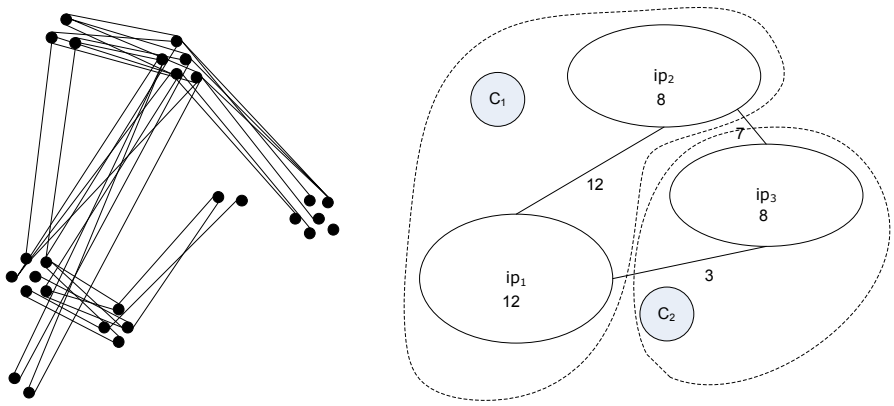
O estabelecimento do ponto de paragem do desengrossamento, que conduzirá ao estabelecimento de uma granularidade na criação e difusão de tabelas de encaminhamento, implica a aceitação de compromissos entre a existência de vértices mais densos e, em consequência disso, um eventual desajuste no balanço; ou vértices mais finos, que poderão causar sobrecargas de espaço de armazenamento avultados, ao nível da representação das tabelas de encaminhamento. Além disso, esta diminuição de grão poderá acarretar dificuldades no controle das políticas de delicadeza, por se tornar mais difícil o controle no acesso descentralizado aos servidores *Web* por parte dos diferentes agentes de descarga.

A utilização de nomes de máquinas como grão dos grafos servirá como base de comparação para os mecanismos baseados em funções de dispersão, uma vez que são os usualmente utilizados na literatura.

Vejamos um exemplo da aplicação dos algoritmos de partição combinados. A Figura 5.1(a) apresenta um grafo que representa 28 páginas *Web* com 38 hiperligações entre elas. Inicia-se o engrossamento deste grafo para uma grafo em que os vértices são IPs (Figura 5.1(b)). Considera-se um engrossamento directo de páginas para IPs, por simplicidade da demonstração.

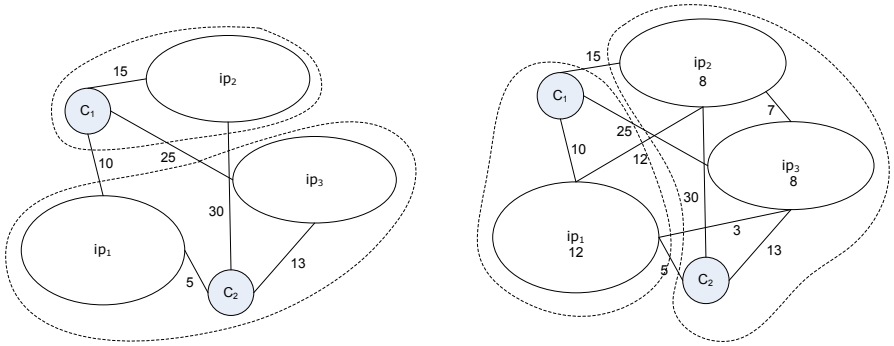
No grafo mais grosso, as páginas foram fundidas no mesmo IP, em que o peso destes vértices é calculado pela soma do número de páginas nesse IP. Para os ramos é calculada a soma dos ramos do grafo mais fino que atravessam os IPs.

Assumindo que se pretende calcular uma configuração de partições para dois robôs agentes, A Figura 5.2(a) representa o grafo de RTTs entre os robôs e os IPs servidores. As linhas interrompidas representam as partições resultantes.



(a) (b)

Figura 5.1: Exemplo de um grafo de IPs e páginas



(a) (b)

Figura 5.2: Exemplo de um grafo de RTTs nos IPs e grafo combinado

Depois de determinadas as partições dos dois grafos individualmente e calculados os cortes dos ramos, é criado o terceiro grafo de acordo com o que foi já referido, resultando um grafo como o da Figura 5.2(b).

5.2 Experiências realizadas com a partição do espaço *Web*

Todas as experiências conduzidas relacionadas com os mecanismos de partição têm como caso de controle a partição do espaço *Web* através de uma função de dispersão. As abordagens baseadas na partição de grafos serão comparadas com a anterior, a

fim de se obterem conclusões acerca das vantagens da utilização e variação dos parâmetros considerados.

Foram utilizadas três metodologias de avaliação: (1) uma através do cálculo algébrico de três métricas; (2) uma experimentação preliminar com a distância geográfica entre os robôs e os servidores; e outra (3) através da simulação do processo de descarga. Para cada uma dela serão descritos a amostra, bem como os parâmetros de avaliação utilizados.

Na primeira metodologia pretende-se avaliar a importância na utilização dos RTT e do número hiperligações na partição de grafos. A segunda metodologia tem como objectivo a experimentação de um mecanismo de partição que utiliza informação cuja recolha é mais económica, como é o caso da determinação da localização geográfica comparada com a determinação dos RTT. A terceira metodologia tem como objectivo a utilização de um mecanismo de experimentação mais próximo da realidade através da simulação da descarga. Além disso, é também o objectivo desta metodologia a comparação das abordagens utilizadas nas metodologias anteriores.

Na divisão do espaço *Web* assume-se a existência de unidades de partição indivisíveis, como é o caso dos nomes das máquinas dos servidores (ou o seus IPs). Esta indivisibilidade foi assumida, não só pelo diminuição de complexidade nos algoritmos de partição, mas também por se reduzir substancialmente o conjunto de regras de necessárias ao processo de encaminhamento de URLs no sistema de descarga.

Além disso, o estabelecimento de um equivalência, entre a unidade de distribuição e o servidor físico de páginas, permite a preservação e o controle das políticas de delicadeza no processo de descarga intensivo.

Uma vez que cada unidade tem uma densidade variável, dependente da quantidade de URLs nela existentes, os algoritmos de partição de grafos tomam em conta esse facto, contudo, preservando os tempos de delicadeza, o tempo global de descarga de um sistema fica sempre limitado, pela unidade mais densa. Como se pode verificar na Figura 5.3, a amostra utilizada dispõe efectivamente de uma distribuição *power-law* para o número de páginas existentes nos servidores, tal como foi também constatado por Henzinger [Henzinger 06], o que significa que existe um pequeno número de servidores com uma grande quantidade de páginas.

Nesta distribuição, a média de URLs por servidor é de 122,48, com um mínimo de 1 e um máximo de 78523.

Relativamente à partição por função de dispersão é necessário averiguar a qual das

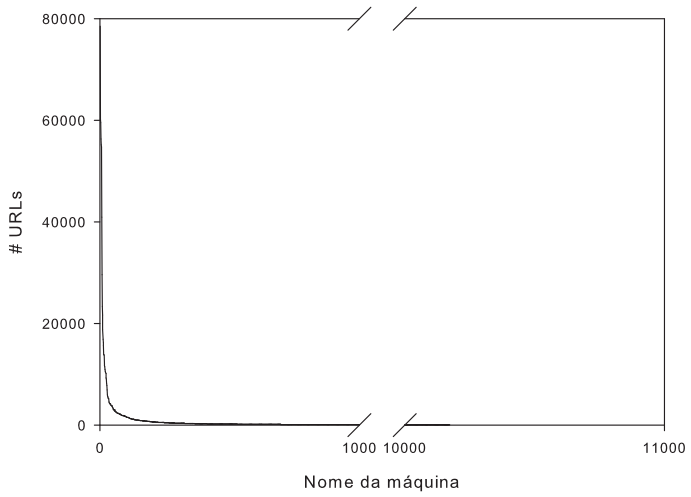


Figura 5.3: Distribuição dos URLs pelos servidores

partes do URLs a função deve ser aplicada. Intuitivamente, e tal como é confirmado em [Loo 04] e em [Cho 02], com a função de aplicada à totalidade do URL, obtém-se um maior balanceamento de carga entre os robôs participantes. No entanto, incorre numa maior sobrecarga de comunicação no encaminhamento dos URLs entre os robôs. Além disso, o controle das políticas de delicadeza torna-se mais complicado, por ser necessária uma sincronização entre os robôs acerca dos servidores contactados. A alternativa mais equilibra é a aplicação da função de dispersão ao nome da máquina servidora, em que se assume algum desajuste de balanceamento, devido à possibilidade de existência de uma quantidade grande de páginas nos servidores. A sobrecarga de comunicação é claramente inferior.

Além destas duas abordagens de aplicação da função de dispersão foi, ainda, utilizada a sua aplicação ao endereço IP da máquina servidora, que, pese embora o facto de aumentar o desajuste de balanceamento, poderá reduzir a sobrecarga de comunicação no encaminhamento de URLS.

Nos mecanismos de partição por grafos foi utilizado o nome das máquinas dos servidores e o endereço IP como vértices dos grafos, de forma a facilitar e equiparar as abordagens. Excluiu-se, no entanto, a possibilidade de utilização dos URLs como vértices dos grafos devido à elevada quantidade de elementos, o que induziria em tempos de processamento dos algoritmos de partição muito elevados, mesmo para a amostra utilizada, tornando-se impraticável numa aplicação real, em que o número de URLs é consideravelmente maior.

É esta observação que coloca em risco o desempenho das abordagens de partição, quando a unidade de partição é de grão grosso como os nomes de máquina, em que uma partição pode, na realidade, ter que descarregar milhares de páginas contidas num único servidor. Uma forma de minimizar este problema é reduzir o grão da unidade de partição para páginas. No entanto, a posterior sobrecarga na rede causada devido às trocas entre os robôs é inaceitável [Cho 02].

5.2.1 Descrição da amostra utilizada

Para a condução das experiências de partição, utilizando um sistema de simulação, foi feita uma selecção aleatória de um conjunto mais pequeno, com cerca de 100 máquinas, do conjunto de máquinas existente inicialmente, que assumiram o papel de robôs. Esta selecção permitiu a dispersão dos robôs pelo espaço *Web* inicialmente disponível. No entanto, existiram condicionantes para a sua escolha. As restrições da escolha das máquinas seguiu o mesmo esquema da metodologia algébrica, contudo, devido ao maior número de robôs o total de máquinas e URLs resultantes foi inferior.

Considerando as condicionantes referidas, resultaram 1.246.129 de URLs, 10.039 nomes de máquinas e 2.440 endereços de máquinas IPs (Tabela 5.2.1). Enumeram-se, ainda, a quantidade de hiperligações e a média de hiperligações por cada página.

Elemento	**Quantidade**
URLs	1.246.129
Hiperligações	5.657.734
Média de Hiperligações por página	21,63
Nomes de máquinas	10.039
Endereços de máquinas (IPs)	2.440

Tabela 5.1: Quantidade de elementos utilizados nas experiências.

5.2.2 Metodologia com expressões algébricas

A utilização de expressões algébricas para a validação da abordagem proposta, surgiu como uma alternativa imediata e de fácil concretização. A ideia subjacente a esta abordagem foi efectuar o cálculo de três métricas que permitissem obter uma noção aproximada na comparação do desempenho das abordagem propostas com a abordagem de referência por função de dispersão [Exposto 07, Exposto 08]. Contudo,

cientes do desfasamento desta abordagem de avaliação com a realidade, propõe-se ainda a criação de um simulador de descarga que permita uma maior aproximação à realidade.

Métricas de avaliação

Para a avaliação do desempenho das abordagens de partição de grafos utilizaram-se três métricas: o tempo de descarga, o tempo de trocas e o tempo de redistribuição. Aplicou-se a partição a dois tipos de grafos: um grafo cujos vértices são nomes de máquinas e um grafo cujos vértices são endereços de máquinas (IPs). Cada um destes tipos de grafos é o resultado da combinação de dois outros grafos com o mesmo tipo de vértices, mas cujos ramos são RTTs e hiperligações, respectivamente.

As métricas referidas têm por objectivo realizar medições de tempos de comunicação entre os robôs e entre estes e os servidores, utilizando os RTT recolhidos, a informação disponível da atribuição dos servidores aos robôs e a quantidade de páginas em cada servidor, definindo algumas constantes, como a largura de banda, número de ligação em simultâneo e o intervalo de delicadeza.

Tempo de descarga. O objectivo do cálculo do tempo de descarga é a estimação do tempo máximo que uma configuração de robô demora a descarregar o conjunto de URLs contidos nos servidores que lhe foram atribuídos.

Para um servidor i, o tempo necessário por um robô j para descarregar as suas páginas é calculado por:

$$dts_i = \frac{M_i}{L_j}(2RTT_{ij} + \frac{L_j \cdot ps_i}{BW_{ij}} + PT_i) \tag{5.1}$$

, em que L_j é o número de páginas descarregadas em paralelo pelas conexões persistente do protocolo `http`, para o mesmo servidor; M_i é o número de páginas do servidor i ; RTT_{ij} e BW_{ij} são o RTT e largura de banda disponível entre o robô j e o servidor i, respectivamente; ps_i é o tamanho médio das páginas; e PT_i é o intervalo de delicadeza entre conexões sucessivas.

Note-se que, em cada momento, um robô apenas realiza um conexão de cada vez para um servidor.

Considerando a descarga de S_j servidores para o robô j e N_j conexões simultâneas, o tempo total de descarga para o robô j pode ser aproximado por:

$$dt_j = \frac{1}{N_j} \sum_{l=1}^{S_j} dts_l \tag{5.2}$$

O tempo máximo de descarga corresponde ao robô mais lento, tendo em conta que todos eles descarregam em paralelo, sendo dado pela expressão: $\max_{i=1}^{p} dt_i$.

Tempo de trocas. A descoberta de hiperligações numa página descarregada por um robô contidas em servidores atribuídos a outros robôs implica o seu encaminhamento para esse robô. A esse tempo designamos de tempo de trocas, sendo estimado por:

$$et_j = \frac{1}{N_j} \sum_{l=1}^{P} 2RTT_{jl} + \frac{su \cdot nl_{jl}}{BW_{jl}}, l \neq j \tag{5.3}$$

em que RTT_{jl} é o RTT entre os robôs l e j, nl_{jl} o número total de hiperligações a encaminhar do robô j para o robô l, su é o tamanho médio de um URL, BW_{jl} é a largura de banda entre os robôs e N_j o número de hiperligações encaminhadas em simultâneo. O tempo total é estimado pelo valor máximo de todos os robôs.

Tempo de redistribuição. Adicionalmente, utilizando o método algébrico, calculou-se, ainda, o tempo necessário à redistribuição dos URLs atribuídos aos robôs quando um novo robô é adicionado à configuração existente.

O cálculo é semelhante ao tempo de trocas à excepção da variável nl_{jl} que em lugar de representar o número de hiperligações a encaminhar, representa a totalidade de hiperligações a transferir do robô j para o robô l. Da mesma forma, é calculado o valor máximo para todos os robôs.

Parâmetros de experimentação

Os valores de algumas variáveis presentes na fórmulas anteriores são extraídos da informação disponível nas recolhas efectuadas, entre as quais, M_i que representa o número de páginas do servidor, obtido a partir de das colecções de URLs disponíveis, e RTT_{ij} que representa o RTT entre duas máquinas, calculado a partir da aproximação realizada nos dados recolhidos pelo `traceroute`.

Uma vez que se optou por não armazenar o conteúdo das páginas, o seu tamanho foi

aproximado por um valor de $ps_i = 10$ KB. Em experiências posteriores será utilizada a média dos tamanhos das páginas das colecções disponíveis. Utilizou-se ainda uma valor $su = 40$ bytes para o tamanho dos URLs. $L_j = 10$, para o número de páginas descarregadas em paralelo pelas conexões persistente do protocolo `http`, para o mesmo servidor. $N_j = 10$ para o número de conexões em simultâneo para servidores diferentes. Um intervalo de tempo de espera (intervalo de delicadeza) entre dois pedidos para o mesmo servidor de $PT_i = 15$ s. E por fim, uma largura de banda em cada troço de $BW_{ij} = 16$ Kbps, que apesar de poder parecer muito conservador, induzindo em tempos de descarga mais prolongados, considera-se mais importante a igualdade de condições entre as abordagens. Além disso, valores mais elevados da largura de banda conduziam a uma variação baixa dos tempos de descarga, dificultando as diferenças entre as abordagens.

Resultados obtidos

Os resultados apresentados de seguida retratam a variação das métricas definidas com a variação do número de partições entre 2 e 30. Comparou-se o desempenho com a aplicação de funções de dispersão e do resultado da partição de grafos.

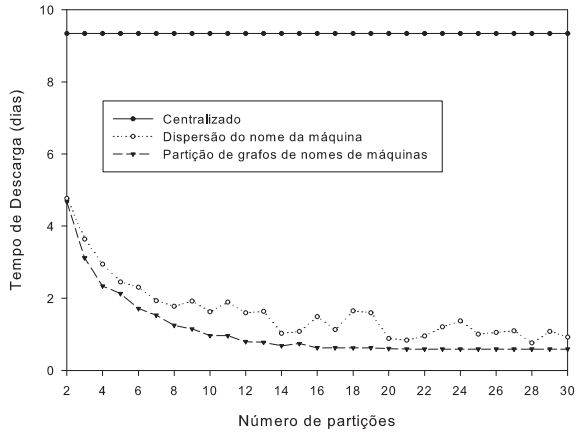


Figura 5.4: Tempo de descarga calculado algebricamente utilizando nomes de máquinas, com RTTs na partição de grafos

A figura 5.4 apresenta a evolução do tempo de descarga à medida que se aumenta o número de partições, comparando uma solução centralizada, outra que utiliza uma função de dispersão e outra utilizando a partição de grafos segundo a estratégia apresentada no capítulo anterior.

Pela observação da figura, podemos concluir que qualquer das abordagens distribuídas suplanta, em grande medida, a abordagem centralizada. Para a quantidade de partições utilizadas, podemos afirmar que a partição de grafos obtém tempos de descarga inferiores em relação à versão que utiliza função de dispersão.

Efectuando a média do tempo de descarga para todas as partições, as versões distribuídas permitem alcançar uma redução de aproximadamente 82% e 88% para a função de dispersão e partição de grafos, respectivamente.

A utilização da partição de grafos permite alcançar em média uma redução de cerca 33% em relação à utilização de funções de dispersão para a quantidade de partições analisadas.

A estabilização dos valores do tempo de descarga a partir das cerca de 20 partições no caso da partição de grafos, está relacionada com a quantidade de páginas a descarregar, podendo concluir-se que seriam suficientes 20 robôs para descarregar a colecção em questão, sem melhoramentos de desempenho com aumento de robôs.

Decidiu-se ainda avaliar o comportamento da partição de grafos utilizando IP de máquinas como unidade de partição. A Figura 5.5 compara o tempo de descarga das abordagens de partição de grafos utilizando o nome da máquina e o IP.

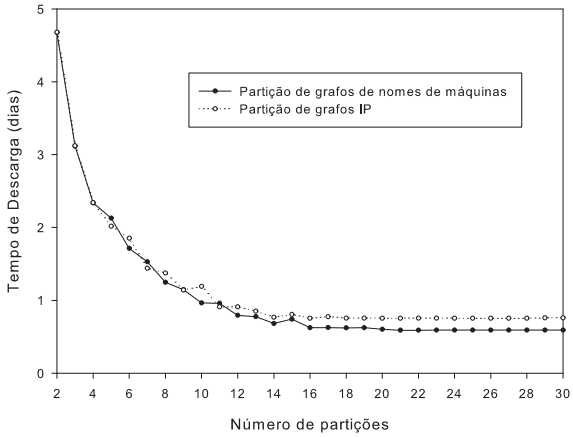


Figura 5.5: Tempo de descarga calculado algebricamente comparando os nomes de máquinas e IPs, com RTTs na partição de grafos

Como se pode constatar, não existe uma melhoria perceptível através da utilização de IP, verificando-se que a solução de nomes de máquinas tem uma redução no tempo de descarga de cerca de 8% comparado com a solução de IPs.

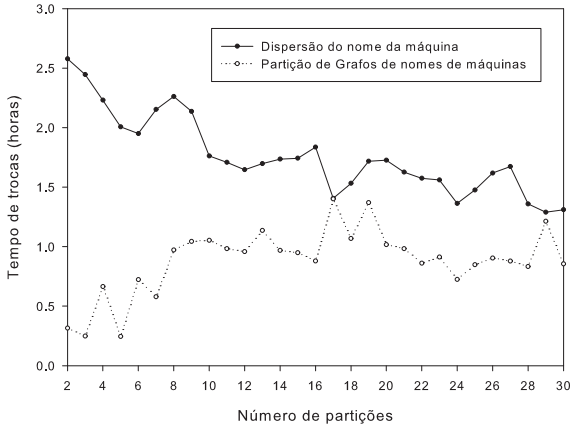


Figura 5.6: Tempo de trocas calculado algebricamente utilizando nomes de máquinas, com RTTs na partição de grafos

A figura 5.6 apresenta a evolução do tempo de trocas à medida que se aumenta o número de partições. Verifica-se uma clara vantagem da abordagem baseada na partição de grafos, com tempo de trocas inferior à abordagem baseada em função de dispersão, com uma redução na média do tempo de trocas de cerca de 50%.

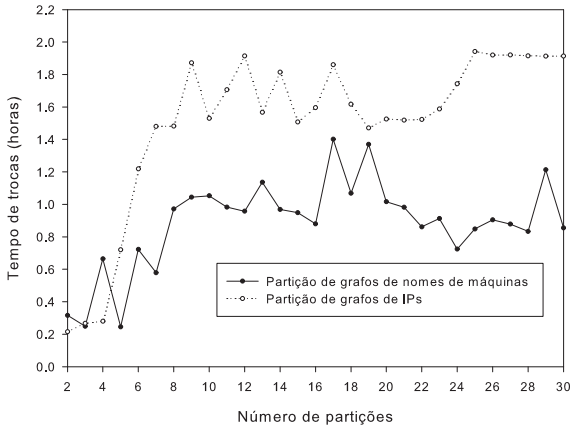


Figura 5.7: Tempo de trocas calculado algebricamente comparando os nomes de máquinas e IPs, com RTTs na partição de grafos

Efectuou-se ainda a comparação dos tempos de trocas obtidos pela partição de grafos utilizando IPs como unidade de partição (Figura 5.7), verificando-se um melhor desempenho dos nomes de máquinas como unidade de partição.

Foi também avaliada a capacidade de extensão de um sistema, quando com um

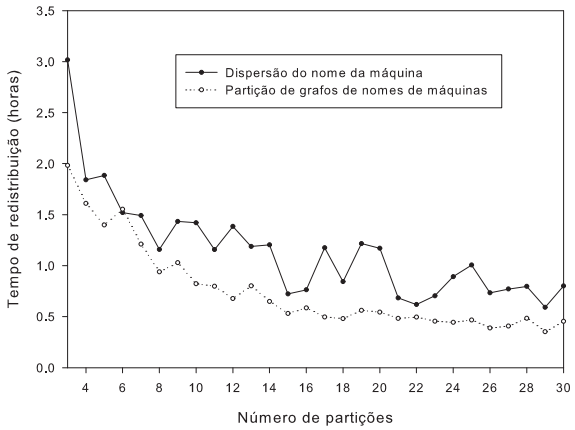


Figura 5.8: Tempo de redistribuição calculado algebricamente utilizando nomes de máquinas, com RTTs na partição de grafos

determinado número de partições, essa quantidade é aumentada. Os resultados obtidos apresentam-se na Figura 5.8, tendo-se observada uma tendência claramente decrescente para ambas as abordagens, indiciando um nível aceitável de escalabilidade no crescimento do sistema. Verifica-se também vantagem da partição de grafos sobre a função de dispersão, obtendo-se na média do tempo de redistribuição, uma redução considerável de cerca de 35%. A utilização dos IPs como unidade de partição também não revelou uma vantagem considerável.

5.2.3 Metodologia com a distância geográfica

A localização geográfica dos servidores e dos robôs, tal como apresentada na secção 6.2 e validada na secção 4.6, constitui uma medida de proximidade alternativa entre os intervenientes do processo de descarga, à distância de rede representada pelo RTT.

A utilização da distância geográfica como medida de peso dos grafos que representam os robôs e os servidores *Web*, tem como base a facilidade e a economia de recursos na recolha dessa informação, comparada com a recolha dos RTT, que exige uma infraestrutura mais pesada computacionalmente e mais demorada em termos temporais. Os resultados da partição da partição de grafos foram previamente publicados no artigo [Exposto 05].

Nessa primeira tentativa de analisar a importância da distância geográfica na criação

de partições eficientes no processo de descarga, criou-se um conjunto de experiências em que se utilizaram grafos aleatórios baseados redes evolucionárias sem escala[1] [Barabási 99], em substituição dos grafos reais das hiperligações entre as páginas. Para a geração dos grafos aleatórios utilizou-se plataforma Java Universal Network/-Graph Framework (JUNG) [jung 05].

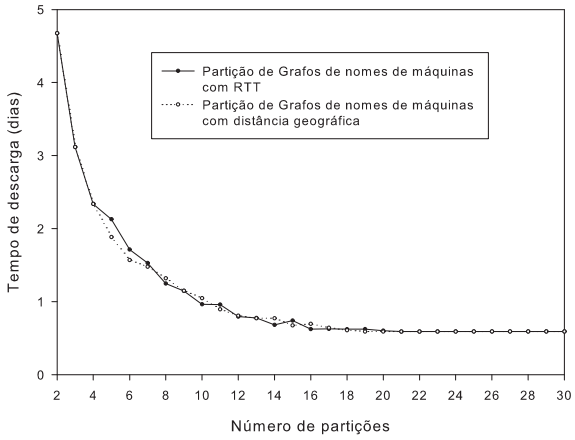


Figura 5.9: Tempo de descarga calculado algebricamente utilizando nomes de máquinas, com a distância geográfica na partição de grafos

Anteriormente, tinha já sido publicada uma descrição das relações entre da topologia de rede e a geografia em que os servidores de enquadravam [Exposto 04].

Os resultados aqui apresentados reflectem a utilização da mesma amostra da experimentação da secção anterior efectuada com os RTTs. Os dados relativos à localização geográfica de cada entidade permitiu o cálculo da distância geográfica entre elas, através da distância geodésica que, juntamente com a informação das hiperligações já disponível, permitiu criar os grafos necessários para a concretização das partições.

As variáveis das métricas de avaliação apresentadas, na metodologia anterior, foram instanciadas com valores semelhantes, sendo o tamanho médio das páginas ps_i = 10 KB, o tamanho médio dos URLs su = 40 bytes. Para o número de páginas descarregadas em paralelo pelas conexões persistente do protocolo `http`, para o mesmo servidor, utilizou-se $L_j = 10$, para o número de conexões em simultâneo para servidores diferentes $N_j = 10$. O intervalo de delicadeza entre dois pedidos para o mesmo servidor $PT_i = 15$ s. Uma largura de banda em cada troço de $BW_{ij} = 16$

[1]Em inglês, evolving scale free networks

Kbps.

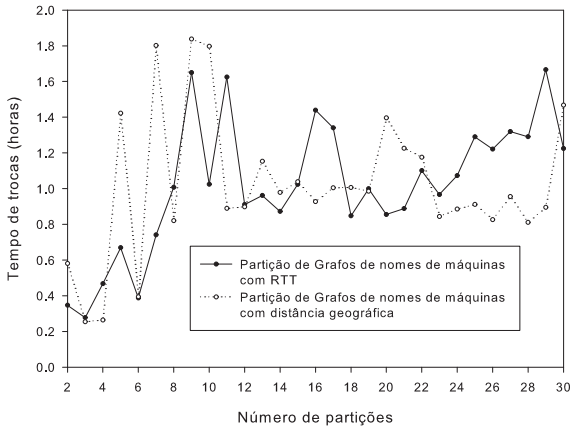


Figura 5.10: Tempo de Trocas calculado algebricamente utilizando nomes de máquinas, com a distância geográfica na partição de grafos

A Figura 5.9 apresenta o tempo de descarga obtido com esta metodologia, comparando os resultados obtidos com a partição de grafos, pesados com os RTTs e com a distância geográfica. Como se pode observar as diferenças nas abordagens são quase imperceptíveis.

A Figura 5.10 apresenta o tempo de trocas obtido com esta metodologia. A redução média obtida da partição de grafos com RTT comparada com a partição de grafos com distâncias geográficas é de cerca de 13%.

A Figura 5.11 apresenta o tempo de redistribuição obtido com esta metodologia. A redução média obtida da partição de grafos com com distâncias geográficas comparada com a partição de grafos com RTT é de cerca de 1,4%.

5.2.4 Metodologia com simulação

Para efectuar a avaliação dos mecanismos de partição utilizando uma metodologia de simulação foi implementado um simulador baseado em eventos discretos. O processo de simulação permitiu modelar com alguma precisão: (1) os tempos de circulação de dados na rede, utilizando os RTT extraídos nas recolhas anteriores; e (2) a gestão do tempo de espera dos URLs nos robôs, originado pela acumulação de URLs e pela imposição das políticas de delicadeza. Foram criados dois cenários de simulação: (1) um com políticas de delicadeza fixas e (2) outro com políticas de delicadeza

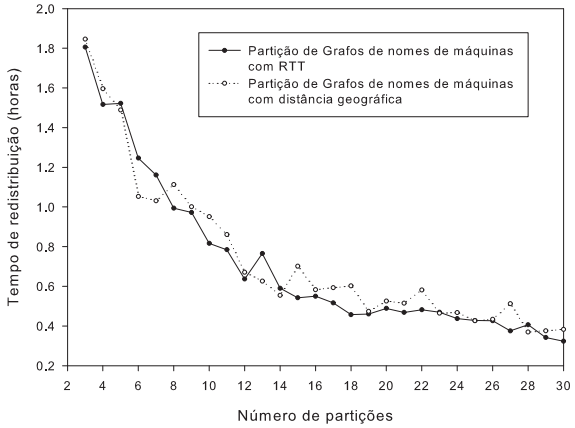


Figura 5.11: Tempo de redistribuição calculado algebricamente utilizando nomes de máquinas, com a distância geográfica na partição de grafos

ajustáveis, com tempos de delicadeza inversamente proporcionais ao logaritmo do número de URLs contidos nos servidores *Web*.

Métricas de avaliação

Nas experiências de simulação que se seguem, são analisados: (1) o tempo total descarga do sistema necessário para descarregar todos os URLs envolvidos, para avaliar a eficiência global de descarga do sistema; e (2) a percentagem de URLs encaminhados para robôs diferentes sobre o total de URLs encaminhados, para avaliar a sobrecarga imposta na rede.

Parâmetros de experimentação

O processo de simulação teve como objectivo a recriação do funcionamento de uma bateria de robôs distribuídos, num ambiente experimental. Estabeleceram-se valores constantes para a largura de banda ($lb = 250KBps$) e para o número máximo de conexões em simultâneo para diferentes servidores que cada robô pode realizar ($ncon = 50$). Em termos de funcionamento do robô é definido o tempo de delicadeza ($td = 15s$), que indica quantidade de tempo que um robô deve esperar para descarregar um novo URL do mesmo servidor. É ainda definido um valor para o tamanho das páginas, $tp = 25\,KB$, retirado da média dos tamanhos das páginas das colecções disponíveis. Neste ponto, não foi utilizado um valor real, por tornar

a simulação extremamente pesada, para além de que, muitos dos URLs conduziam a páginas que não estavam disponíveis no espaço inicial, mas que foram igualmente contabilizadas, desde que apontassem para um servidor conhecido.

Resultados obtidos

Iniciamos a apresentação dos resultados, na Figura 5.12, com a comparação do tempo total de descarga entre as abordagens da função de dispersão e a partição de grafos utilizando o RTT com medida dos ramos dos grafos, variando o número de partições.

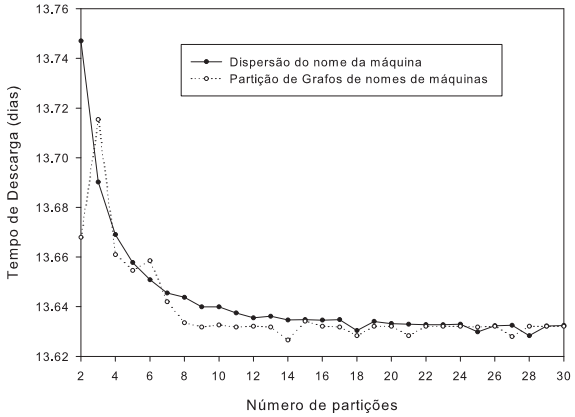


Figura 5.12: Tempo total de descarga obtido por simulação com o número de partições variável

A diferença das duas abordagens é muito ténue, existindo apenas uma redução média de 0,03% na partição de grafos em relação à função de dispersão. A utilização de IPs como unidade de partição, não teve diferenças perceptíveis, com uma redução muito próxima de zero.

A Figura 5.13 apresenta a variação da percentagem de trocas de URLs inter-partição em relação ao total de trocas entre as abordagens da função de dispersão e a partição de grafos utilizando o RTT com medida dos ramos dos grafos, variando o número de partições.

Como se pode observar, existe uma clara vantagem na utilização dos mecanismos de partição de grafos, tendo-se obtido uma redução na percentagem de trocas na ordem dos 76%.

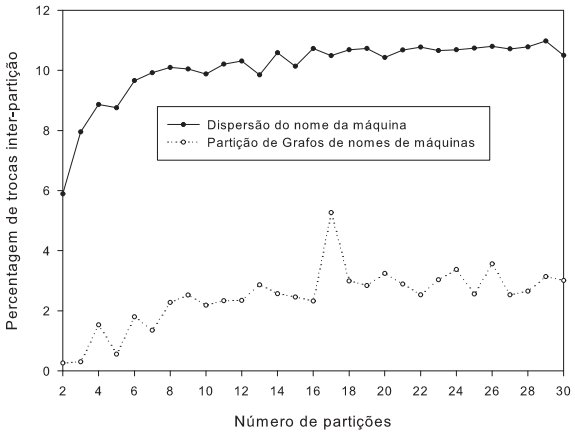


Figura 5.13: Percentagem de trocas inter-partição obtido por simulação com o número de partições variável

Esta grande redução na sobrecarga do sistema originado pelas trocas de URLs pode ser interpretada como um factor aliciante à utilização da partição de grafos, tendo, embora, a pesar o facto de não existir uma diferença significativa no tempo de descarga em relação aos mecanismos de função de dispersão.

Seguida apresentam-se os resultados obtidos pela utilização da distância geográfica em substituição do RTT na atribuição dos pesos dos ramos dos grafos para efeitos de partição. A Figura 5.14 apresenta a comparação do tempo de descarga obtido na comparação da partição de grafos com o RTT nos pesos dos ramos e a distância geográfica.

A utilização da distância geográfica neste contexto, não reflectiu um agravamento no tempo de descarga, verificando-se uma redução muito próxima de zero.

Em termos de percentagem de trocas, a utilização dos RTTs obtém apenas uma redução de cerca de 2,3% em relação à distância geográfica. A utilização da distância geográfica, apesar obter uma percentagem de trocas ligeiramente superior, relembre-se que a obtenção das distâncias geográficas é bem mais económica que a obtenção dos RTTs.

Seguindo ainda a convicção de que a criação de grafos de IPs poderiam induzir em vantagens na percentagem de trocas inter-partição, sem penalizar gravemente o tempo de descarga, apresenta-se na Figura 5.16 o tempo de descarga obtido com a partição de grafos utilizando nomes de máquinas e IPs como unidade de partição.

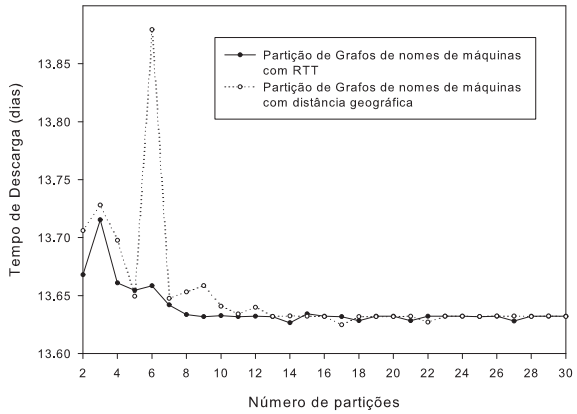


Figura 5.14: Tempo total de descarga obtido por simulação utilizando a distância geográfica com o número de partições variável

A utilização de IPs como unidade de partição conduz a uma degradação bastante acentuada do tempo de descarga, originado pela maior quantidade de páginas a descarregar em cada uma dessas unidades. No caso dos IPs, o tempo de delicadeza é o principal responsável por esse agravamento, obtendo-se uma redução de cerca de 22% na utilização de nomes de máquinas em relação à utilização de IPs.

No que respeita à percentagem de trocas verificada com a utilização de IPs, a Figura 5.17 apresenta os resultados obtidos.

Nesta situação a utilização de IPs também não trouxe vantagens em relação à utilização de nomes de máquinas, tal como já tivera sido observado na experimentação algébrica. Um ajuste nos pesos de ponderação na combinação dos grafos talvez pudesse inverter esta situação, contudo a alteração desses pesos fica remetida para trabalho futuro.

5.2.5 Metodologia com simulação com políticas de delicadeza ajustáveis

Os mecanismos de partição de grafos, ao contrário dos mecanismos que utilizam funções de dispersão, recolhem informação do sistema envolvente de descarga. Esta informação utilizada na criação de partições de URLs de forma a optimizar a descarga, não se manifestou eficaz na melhoria da descarga, devido à densidade de URLs nalguns dos servidores. Note-se, inclusivamente, que o aumento do paralelismo não

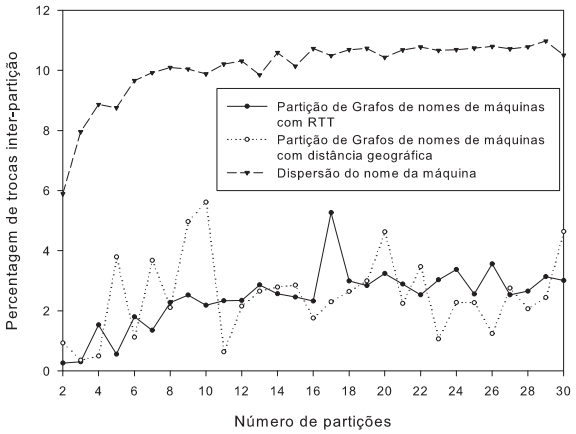


Figura 5.15: Percentagem de trocas inter-partição obtido por simulação utilizando a distância geográfica com o número de partições variável

acarreta grandes vantagens.

Uma vez que os mecanismos de partição dispõem de informação pormenorizada sobre os servidores que descarregam para a concretização dos algoritmos de partição, torna-se possível ajustar o tempo de delicadeza de forma dinâmica (td_{din}), com base na quantidade de páginas disponíveis nos servidores ($pags$), assumindo um tempo de delicadeza máximo (td_{max}) e uma quantidade de páginas máxima ($pags_{max}$), através da fórmula 5.4.

$$td_{din} = td_{max} - \frac{\ln pags}{\ln pags_{max}}(td_{max} - 1) \tag{5.4}$$

A fórmula anterior permite a atribuição de um tempo de delicadeza igual a um para o servidor com o maior número de páginas e um tempo de delicadeza igual a td_{max} para os servidores com apenas uma página. Para permitir que apenas os servidores com uma quantidade muito baixa de páginas lhes fosse atribuído um tempo de delicadeza máximo, aplicou-se a função logarítmica às quantidades de páginas, conduzindo naturalmente a uma descida acentuada do tempo de delicadeza médio.

Assumindo um tempo de delicadeza máximo de 15s, a Figura 5.18 retrata a evolução do tempo de descarga total, comparando a abordagem de função de dispersão com tempo de delicadeza constante de 15s e a abordagem de partição de grafos, utilizando este esquema de descarga com tempo de delicadeza ajustável.

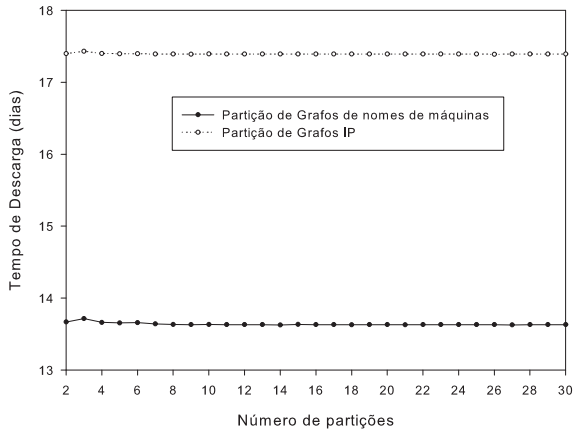


Figura 5.16: Tempo total de descarga obtido por simulação com nomes de máquinas e IPs

A redução da média do tempo de descarga atinge, para a partição de grafos com tempo de delicadeza ajustável, os 91%, o que constitui um aumento de desempenho na descarga bastante impressionante.

Embora a Figura 5.18 não deixe transparecer, a distribuição da descarga por 30 robôs permite a redução de quase 1 dia no tempo total de descarga como se pode visualizar na Figura 5.19(a), ao passo que, para a função de dispersão não é possível extrapolar a mesma conclusão para o número de partições analisadas (Figura 5.19(b)).

Naturalmente, que estes níveis de tempo de descarga apenas são possíveis devido a uma redução considerável do tempo de delicadeza médio do sistema, atingindo os 4,45s. Para tornar as comparações de abordagens mais justas, decidiu-se confrontar a descarga com uma política de delicadeza ajustável utilizando a partição de grafos com uma abordagem que utiliza a função de partição com uma política de delicadeza constante igual à média da abordagem anterior.

Apesar da utilização de tempos de delicadeza constantes da ordem de grandeza da obtida com a abordagem com tempo de delicadeza ajustável o desempenho da descarga fica aquém desta última, obtendo uma redução de cerca de 69%, tendo ainda como factor agravante a penalização dos servidores menos densos, que se assumiriam menos robustos pela sua menor dimensão em termos de páginas.

Comparando a aplicação do tempo de delicadeza ajustável entre as abordagens de partição de grafos e da função de dispersão (Figura 5.21), podemos observar que é possível obter uma redução média de cerca de 6%.

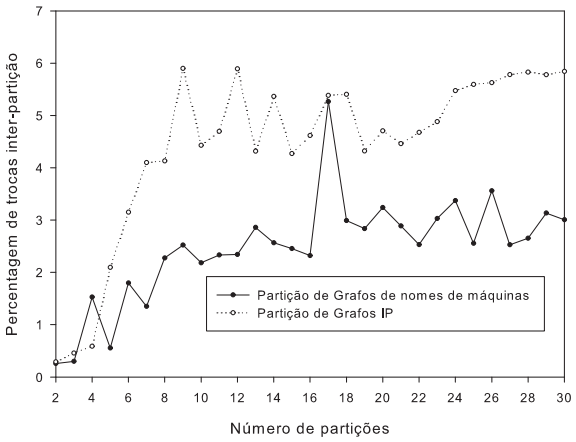


Figura 5.17: Percentagem de trocas inter-partição obtido por simulação com nomes de máquinas e IPs

Em termos de sobrecarga de trocas, a utilização dos tempos de delicadeza ajustáveis não afectam os resultados já obtidos.

Por fim, ainda enquadrado nas experiências com a política de delicadeza ajustáveis, observou-se o comportamento da distância geográfica comparada com o RTT. A Figura 5.22 apresenta esse resultado.

Claramente, com a utilização de RTTs obtém-se um tempo de descarga inferior comparado com a distância geográfica, com uma redução na ordem dos 57%, mas mesmo assim, a utilização da distância geográfica obtém um redução de cerca de 28% comparada com a utilização da função de dispersão com tempo de delicadeza constante nos 4.45 s.

Tendo em conta que a determinação da localização geográfica é menos onerosa do que a recolha dos RTT, a utilização da distância geográfica pode ser uma alternativa muito aliciante.

5.2.6 Comparação com outras abordagens

A revisão da literatura realizada não permitiu descobrir uma variedade substancial de abordagens efectuadas por outros autores, no que respeita à definição do tempo de delicadeza utilizado. Dos trabalhos analisados, alguns nem fazem referência a esse tempo, e frequentemente são utilizados tempos fixos, como por exemplo, Shkapenyuk e Suel, que utilizam um tempo fixo de 30 segundos [Shkapenyuk 02]. Heydon e

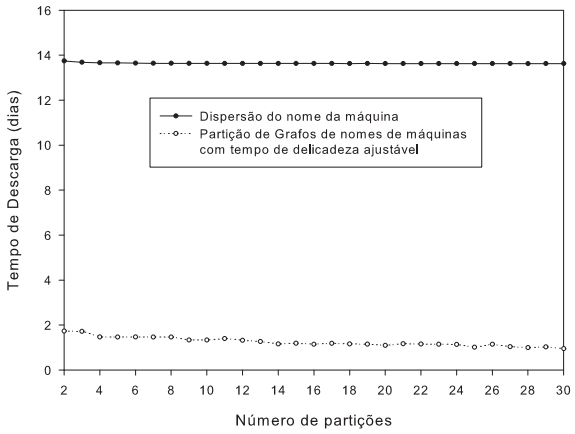


Figura 5.18: Tempo total de descarga obtido por simulação com o número de partições variável e tempo de delicadeza ajustável

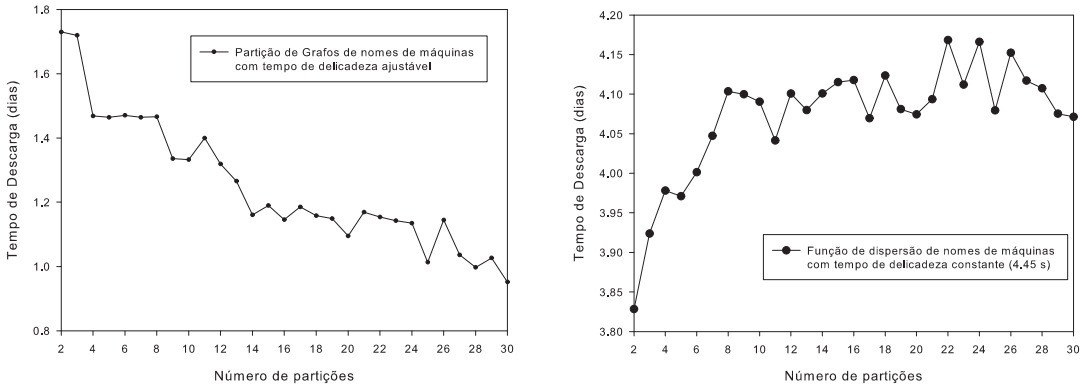
Najork referem uma técnica, mais dinâmica, definindo um intervalo de delicadeza de dez vezes o tempo que a página anterior demorou a ser descarregada [Najork 02].

Em relação ao tempo de delicadeza fixo, as experiências conduzidas já demonstraram a sua fraca eficiência. No que respeita à utilização de um tempo de delicadeza dinâmico, a comparação de esquemas como o referido por Heydon e Najork com os esquemas ajustáveis aqui apresentados, não é possível obter uma comparação directa, por causa da utilização de tempos delicadeza com ordens de grandeza diferentes.

Deste modo, realizou-se a simulação da descarga utilizando um tempo de delicadeza dinâmico em que o seu valor é igual a dez vezes o tempo de descarga da página anterior, confrontando a abordagem baseada em função de dispersão com a partição de grafos, apresentando-se os resultados obtidos na Figura 5.23.

Da observação do gráfico, podemos concluir que qualquer uma das abordagens que utiliza a definição de um tempo de delicadeza dinâmico obtém um tempo de descarga bem menor do que utilizando um tempo de delicadeza constante. Entre as duas abordagens, apesar de as diferenças não serem muito significativas, a partição de grafos obtém uma redução na média do tempo de descarga de cerca de 14%.

A percentagem de trocas de URLs mantém-se igual aos valores já analisados.



(a) (b)

Figura 5.19: Comparação da escalabilidade da partição de grafos e da função de dispersão

5.3 Discussão dos resultados

A distribuição da lei da potência seguida pela quantidade de páginas disponíveis nos servidores *Web*, não permite a optimização da descarga, mesmo utilizando mecanismos inteligentes de alocação do espaço *Web* a uma bateria de robôs distribuídos, quando se almeja a preservação das políticas de delicadeza.

As experiências algébricas iniciais, embora pouco fidedignas, revelaram-se um instrumento bastante útil na previsão de resultados e na preparação de uma plataforma de simulação mais elaborada e realista. Desde logo, se previu a pouca diferenciação nos tempos de descarga entre as abordagens baseadas em funções de dispersão e as abordagens com partição de grafos. Também foi possível prever, um óptimo comportamento por parte das abordagens com partição de grafos, no que respeita à redução nos tempos de trocas de URLs entre os robôs. Além disso, para ambas as métricas de avaliação, a utilização da distância geográfica como peso de medida nos ramos dos arcos obteve resultados similares comparados com a utilização dos RTT.

A plataforma de simulação criada, para além de permitir uma maior fidelidade com a realidade, ao nível do escalonamento da descarga e das trocas de URLs, tornou mais versátil a configuração de parâmetros adicionais.

Desde logo, se salientou os bons resultados obtidos na percentagens de trocas inter-partição, com reduções impressionantes na ordem dos 76% por parte das abordagens baseadas em partição de grafos comparadas com as de função de dispersão.

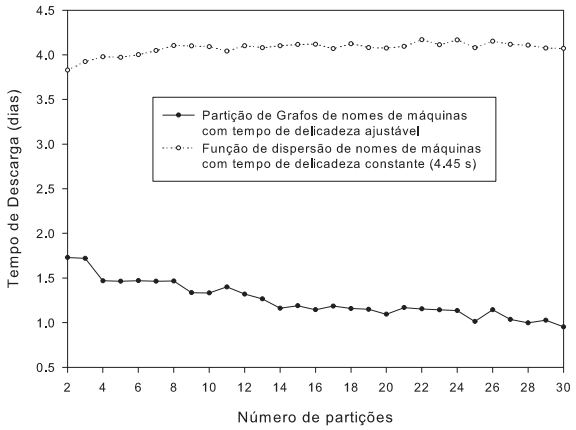


Figura 5.20: Tempo total de descarga obtido por simulação comparando o tempo de delicadeza ajustável e o constante

O tempo de descarga, contudo, ficaria sempre maximizado pelo robô responsável pela descarga do servidor mais pesado em termos de quantidade de URLs.

A inclusão de um tempo de delicadeza ajustável proporcional ao suposto desempenho do servidor, estimado pela quantidade de páginas disponibilizadas por ele, permitiu uma redução considerável no tempo de descarga na ordem dos 91% quando comparada com a função de dispersão com um tempo de delicadeza constante de 15 s. Mesmo utilizando um tempo de delicadeza na função de dispersão em média semelhante à abordagem de partição de grafos com tempo de delicadeza ajustável, a redução do tempo de descarga rondou os 69%, com indícios de boa escalabilidade.

A aplicação de tempos de delicadeza ajustáveis a funções de dispersão, que tivera já sido aplicado por outros autores, teve um desempenho ligeiramente inferior em relação à utilização da partição de grafos.

Quando se comparou a partição de grafos com a função de dispersão aplicando um tempo de delicadeza dinâmico proporcional ao tempo de descarga da página anterior, foi possível obter uma redução no tempo de descarga. Apesar da maior facilidade no cálculo do tempo de delicadeza dinâmico, o tempo de delicadeza ajustável com a densidade dos servidores traz maiores reduções no tempo de descarga.

A utilização da distância geográfica, apesar de ter obtido uma degradação em relação à utilização do RTT, permitiu atingir níveis de desempenho muito próximos, mas sempre melhores que as soluções baseadas em função de dispersão para as métricas avaliadas. Considerando a recolha das localizações geográficas mais económica do

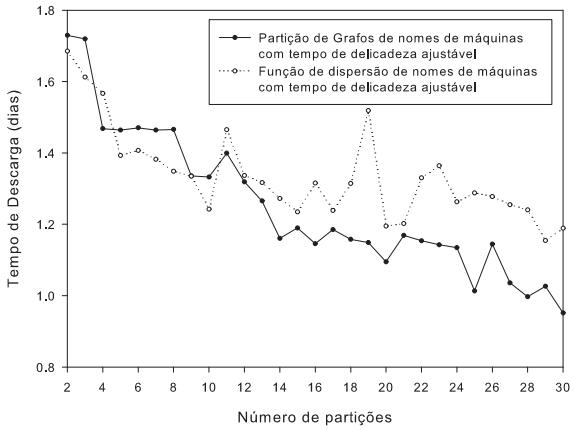


Figura 5.21: Tempo total de descarga obtido por simulação comparando o tempo de delicadeza ajustável entra a partição de grafos e a função de dispersão

que a recolha dos RTT, em termos de tempo e de infraestruturas necessárias, a distância geográfica constitui, sem dúvida, uma solução para a optimização da descarga com contrapartidas muito aceitáveis.

No que respeita à utilização de IPs como unidade de partição em substituição dos nomes das máquinas, existiu um pressuposto através de experiências iniciais com colecções de URLs mais pequenas, de que a percentagem de trocas poderia ser reduzida, assumindo uma penalização ao nível da descarga. Todas as experiências aqui apresentadas, mostram, precisamente, o contrário. Contudo, estamos convictos que através da afinação dos factores de ponderação na combinação dos grafos se poderá atingir essa redução.

Fica em aberto o estudo e a adequação desses factores de ponderação para os diferentes tipos de unidades de partição, relegando-o para trabalho futuro.

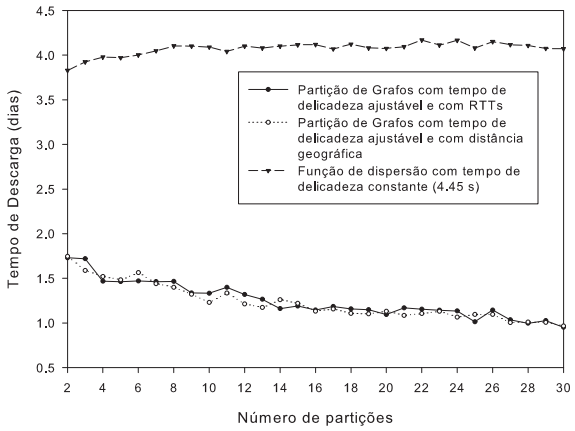


Figura 5.22: Tempo total de descarga obtido por simulação com tempo de delicadeza ajustável comparando o RTT e a distância geográfica

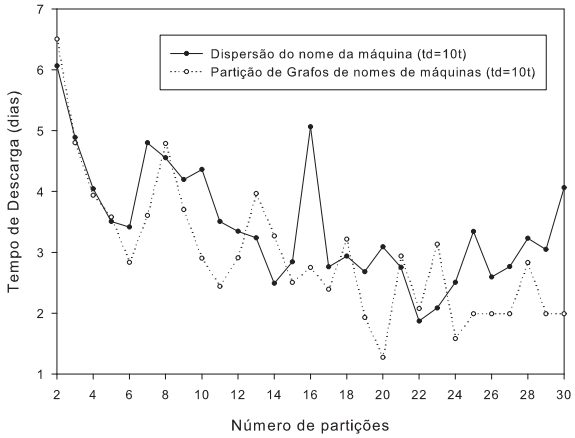


Figura 5.23: Tempo total de descarga obtido por simulação com tempo de delicadeza igual a 10 vezes o tempo de descarga da página anterior

Capítulo 6

Plataforma de suporte de dados e de simulação

Idealmente, a avaliação de um sistema de descarga de recursos *Web* deveria ser efectuada num ambiente real, por oferecer condições mais realistas ao nível da utilização de recursos de comunicação. No entanto, dada a complexidade da *Web*, pela volatilidade dos conteúdos publicados e pela instabilidade das rotas que interligam os servidores *Web*, a concretização de uma ambiente experimental nestas condições tornar-se-ia inviável. Para além disso, a experimentação realizada, envolve a validação de diversas técnicas e a variação de vários parâmetros, que para serem devidamente validados, torna-se conveniente a criação de condições semelhantes entre as diversas experiências.

Neste capítulo é descrita a plataforma de suporte de dados utilizada em diversas fases da concretização deste trabalho. É também apresentada a plataforma de simulação utilizada na realização da experimentação da descarga distribuída e validação dos vários parâmetros a ela associados.

As implementações foram realizadas principalmente em código Java, num total aproximado de 33000 linhas de código, organizadas em 230 ficheiros.

O processo de experimentação iniciou-se com a recolha inicial de URLs, reunidos de diversas fontes, nomeadamente, a colecção WPT03 e a NetCensus. Por conveniência no acesso e no processamento, oferecidos pela versatilidade das interrogações SQL, os dados recolhidos foram armazenados numa base de dados relacional, suportada pelo motor Oracle 9i, e posteriormente pela versão 10g [Oracle 07]. A opção da utilização de uma base de dados relacional revelou-se de extrema utilidade para

extracção de dados estatísticos, verificação de resultados e interrogações de agrupamento, operações sobre as quais se podia aceitar uma menor eficiência.

A simulação dos mecanismos de descarga estudados implicou o acesso aos dados de uma forma massiva e aleatória, pouco adequada à organização relacional de uma base de dados, tendo-se criado uma infraestrutura de armazenamento de dados distribuída por um *cluster* de máquinas.

Nas secções que se seguem são apresentados os seguintes sub-sistemas:

- O sistema de aquisição e armazenamento de URLs;
- O sistema de recolha de informação topológica de rede e geográfica
- O sistema de armazenamento distribuído;
- O sistema de simulação da descarga.

6.1 Sistema de aquisição e armazenamento de URLs

Desde o início da preparação deste trabalho ponderou-se a possibilidade da experimentação das estratégias de partição para optimização da descarga em larga escala. Considerando o tamanho da *Web*, e tendo em conta a vertente experimental do trabalho, optou-se por reduzir a ambição em utilizar como bancada de testes um conjunto muito elevado de URLs, sem qualquer organização lógica, e utilizar, em alternativa, um grupo de URLs bem definido. Optou-se, assim, pela utilização de URLs contido no domínio `pt`, que, como já se pôde observar não se encontram necessariamente contidos em Portugal.

De modo a simplificar a fase inicial do trabalho, optou-se ainda por utilizar colecções de URLs de domínio público disponibilizadas por outros grupos de trabalho, nomeadamente, a NetCensus [Silva 02] e a WPT03 [Group 03], donde se aproveitaram 13.248.199 e 3.775.611 de URLs, respectivamente, perfazendo 16.152.324 depois de reunidas as colecções numa base de dados relacional.

Contudo estas colecções apresentavam características bem distintas em termos da informação que ofereciam. Enquanto que a NetCensus seria voltada para a análise estatística dos recursos *Web* e para a sua evolução, a WPT03 teria objectivos voltados para a recuperação de conteúdos, estando, por isso, presente o conteúdo das páginas.

Após a reunião de ambas as colecções efectuou-se uma descarga da *Web* utilizando os URLs obtidos. Note-se que o objectivo desta descarga foi a de encontrar as hiperligações que as páginas dos URLs disponíveis teriam, bem como eliminar URLs inválidos ou obsoletos, não se procurando encontrar nem seguir novos URLs.

Para o efeito utilizou-se uma versão adaptada do robô Larbin [Ailleret 03] pela equipa de trabalho do projecto NetCensus. Além disso, foi também disponibilizado um carregador dos resultados obtidos pelo Larbin para uma base de dados relacional [Silva 03].

O processo consistiu em configurar o Larbin para descarregar páginas com um nível de profundidade zero, em que os URLs eram injectados num porto específico, depois de lidas da base de dados. Embora, aparentemente linear, o processo foi bastante moroso devido a frequentes erros de descarga não detectados pelo Larbin, tais como, URLs inválidas e a bloqueios da aplicação.

O desfasamento temporal entre as recolhas das colecções iniciais e a actualização das hiperligações efectuado com o Larbin de cerca de dois anos para a WPT03 e três anos para a NetCensus, conjugada com a decisão de não descarregar novos URLs, foi responsável por uma redução acentuada do tamanho inicial de URLs, contabilizando-se nesta fase 4.298.542 URLs e 10.365.741 hiperligações, que serviram de base para as experiências que se seguiram. No processo de armazenamento dos URLs, criou-se uma tabela adicional com informação acerca do nome da máquina em que as páginas estavam albergadas.

6.2 Sistema de recolha de informação topológica de rede e geográfica

A aquisição da informação topológica de rede foi concretizada através do envio de sondas utilizando o `traceroute`, tal como foi anteriormente explicado. A aquisição da informação geográfica foi obtida recorrendo a diversas fontes, tais como, o DNS e bases de dados de outras aplicações, entre as quais, o GTrace [Periakaruppan 99], o MaxMind [MaxMind 07] e o NetGeo [CAIDA 02].

A recolha de informação topológica de rede e geográfica foi efectuada em duas fases. Na primeira fase os nomes das máquinas contidos no domínio `pt` foram sondados com o `traceroute` a fim de obter os caminhos de rede até ao destino. Os dados obtidos com o `traceroute` foram guardados em ficheiros de texto, procedendo-se,

numa segunda fase, ao seu processamento e carregamento da base de dados, que detalharemos mais à frente.

Após o primeiro carregamento da base de dados com os nomes de máquinas resolvidos no DNS, foi possível reduzir a quantidade de sondas enviadas em cada ronda, passando a utilizar-se o endereço IP das máquinas, devido à existência de vários nomes de máquinas por endereço IP.

6.2.1 Fase de sondagem dos endereços das máquinas

A sondagem dos endereços das máquinas foi concretizada através da invocação do comando `traceroute` e posterior análise e interpretação do resultado. Por cada máquina foi criada uma lista ligada, em que cada elemento corresponde aos saltos sucessivos devolvidos pelo `traceroute`, contendo informação do endereço do salto correspondente, juntamente com a média dos tempos médios de ida e volta até esse ponto, quando existiam em quantidades superiores a um. No final da análise, a lista ligada era serializada para ficheiro.

Para colmatar a deficiência do número de fontes de sondagem, utilizaram-se diversos serviços de `traceroute` disponíveis através de aplicações *Web*. Nesta situação, o procedimento foi em todo semelhante ao processo anterior, com a diferença na invocação do `traceroute`, que passava a ser um pedido a um página *Web*, e subsequente análise.

No sítio `http://www.traceroute.org/` pode ser encontrado um índice de dezenas destes serviços organizados por país. Contudo, a sua utilização requer adequação individual dos analisadores de texto que extraem os resultados da sondagem, tornando-se impossível a automatização em larga escala.

Foram utilizados nove desses serviços dispersos por: Amesterdão, Toronto, Oslo, Singapura, Nova Iorque, Varsóvia, Dublin, Paris e Reykjavik. A escolha destas foi manual e aleatória, tentando obter um dispersão pelo globo terrestre.

Uma outra desvantagem dos serviços *Web* de `traceroute` comparados com a invocação tradicional, foi o intervalo de espera necessário entre cada pedido, de forma a evitar a exclusão no acesso, que inicialmente, chegou a ocorrer frequentemente.

6.2.2 Fase de processamento e carregamento da base de dados

O processamento e carregamento dos dados serializados envolveu a criação de um suporte relacional de forma a permitir, de forma simplificada, a consulta posterior dos dados obtidos. A Figura 6.1 apresenta o modelo relacional das tabelas utilizadas.

São armazenados os endereços IP dos servidores e dos encaminhadores descobertos, distinguidos pela relação para `IP_TYPE`. O mapeamento entre endereços IP e nomes de máquinas é estabelecido com a tabela `HOSTNAME_IP`. Armazena-se, ainda, a hierarquia de domínios dos nomes das máquinas com indicação da profundidade através das tabelas `HOSTNAME_DOMAIN` e `DOMAIN`.

A tabela `IP_CITY` permite associar uma localização geográfica a um endereço IP, sendo indicado o tipo de determinação geográfica utilizado. Existem ainda tabelas para o armazenamento dos países e continentes, correspondentes a cada cidade.

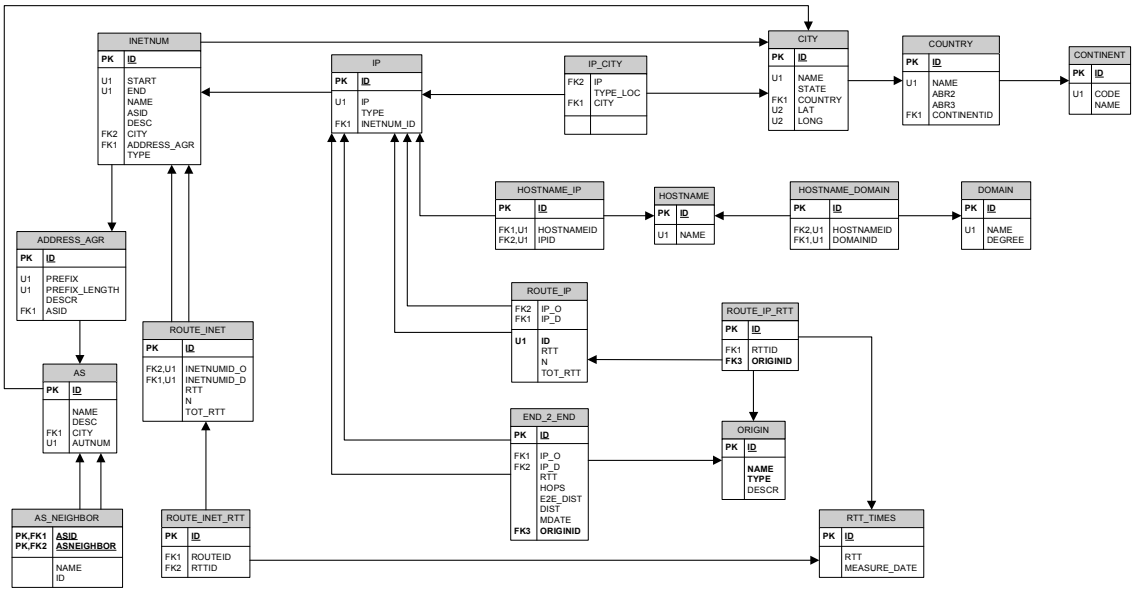


Figura 6.1: Modelo Relacional da base de dados utilizada para o armazenamento da topologia

Os resultados fornecidos pelo `traceroute` previamente serializados são carregados para a tabela `ROUTE_IP`, que armazena um salto entre dois endereços IPs, a tabela `ROUTE_IP_RTT`, que permite uma associação n-n entre cada salto (par de IPs) e os vários RTTs medidos nesse salto, armazenados na tabela `RTT_TIMES`. Adicional-

mente, é armazenado a origem da medições (local ou por diferentes serviços *Web*), e ainda um selo temporal da medição.

Complementarmente, armazena-se ainda informação do traçado de cada `traceroute` executado para cada destino, salvaguardando o RTT fim a fim, o número de saltos, a distância geográfica fim a fim e soma das distâncias geográficas dos vários troços, que constituem cada rota, utilizando a tabela `END_2_END`.

Adicionalmente, foi ainda armazenada informação da topologia Internet, como os blocos de endereços a que um endereços IP pertence (`INETNUM`), os endereços agregados anunciados pelos Sistemas autónomos no processo de encaminhamento de rede (`ADDRESS_AGR`) e os sistemas autónomos (`AS`).

Constituíram-se, ainda, grafos de rotas para blocos de endereços, extrapolados das rotas dos IP associados (`ROUTE_INET`), e grafos de vizinhança entre os Sistemas autónomos (`AS_NEIGHBOR`). Contudo, esta informação nunca viria a ser utilizada, reservando-se a utilização para trabalho futuro.

Criação e armazenamento do grafo de rotas

Tal como foi já referido na secção 4.5, os resultados obtidos pelo `traceroute` apenas fornecem uma medida temporal de atraso de rede, desde uma origem até um destino (ou encaminhador intermédio). Um dos objectivos desta abordagem era recriar um grafo de endereços IP, de tal modo que fosse possível calcular de forma aproximada o RTT entre quaisquer dois IPs. O procedimento seguido foi calcular a diferença de RTTs entre dois IP consecutivos durante a análise dos registos do `traceroute` e inserindo esse valor na tabela `RTT_TIMES` da rota associada na tabela `ROUTE_IP_RTT`. Desta forma, aproximou-se o RTT entre quaisquer dois IP calculando o caminho mais curto entre eles, utilizando as diferenças referidas. A apresentação dos resultados desta aproximação pode ser observada na secção 4.5.

Paralelamente, foram também colocados na tabela `END_2_END` os RTT reais da origem ao destino, possibilitando a verificação do erro cometido na aproximação efectuada.

Armazenamento da localização geográfica dos endereços IP

A localização geográfica foi determinada usando os mecanismos referidos na Secção 4.6. Para a definição dos grafos que utilizaram distâncias geográficas foram calculadas as distância geodésicas a partir das coordenadas geográficas obtidas.

6.3 Sistema de armazenamento distribuído

A motivação principal para a construção de uma infraestrutura computacional distribuída ,de suporte de dados, foi o elevado número de experiências a realizar, com a finalidade de validar toda a arquitectura de partição do espaço *Web*. Optou-se por uma distribuição ao nível dos dados em detrimento à distribuição fina de código, pela sua maior simplicidade e pela melhor adequação ao problema, que permitiria um aumento do espaço de dados disponível, através da distribuição de memória, e uma paralelização integral dos processos, que possibilitaria a sua execução simultânea, evitando alterações específicas para a sua paralelização interna.

Esta infraestrutura foi desenhada para utilização nos mecanismos de partição de grafos, tendo também sido aproveitada nos processos de simulação.

O acesso a dispositivos de E/S secundários persistentes são, actualmente, uma das maiores limitações no desempenho das aplicações com operações intensas de E/S. A massificação das redes de elevado desempenho veio permitir a utilização de memória remota de uma forma mais eficiente do que o acesso tradicional entre memória e discos [Markatos 96].

Para além das vantagens da extensão das capacidades de memoria central, torna-se possível no mesmo ambiente, a execução simultânea de processos, com a partilha dos dados. Em sistemas de armazenamento de dados relacionais, como é o caso dos motores de bases de dados, a selecção de dados e a agregação de resultados utilizando interrogações complexas, podem ser tarefas demoradas. Para evitar estas operações demoradas, a informação indispensável para o processo de partição, como é o caso da informação de base para descrição dos grafos (vértices e ramos), é resumida na infraestrutura de armazenamento de modo a aumentar o desempenho nos acessos posteriores.

Implementaram-se, de forma distribuída, três estruturas de dados convencionais: grafos, conjuntos e dicionários; e respectivas operações típicas de criação, inserção, acesso e remoção. Além disso, foram ainda criados iteradores distribuídos com acesso concorrente para todas as estruturas.

6.3.1 Infraestrutura de armazenamento

A infraestrutura de armazenamento foi montada no *cluster* Search da Universidade do Minho, actualmente composto por um total de 46 nós de computação, com

processadores dual Xeon a 3.2 GHz, com 2GB de RAM, com interfaces de rede Gigabit Ethernet e Myrinet 10G [Search 05].

A organização física da infraestrutura encontra-se representada na Figura 6.2, em que diversas máquinas são utilizadas como nós de suporte ao armazenamento (*cluster* Nodes).

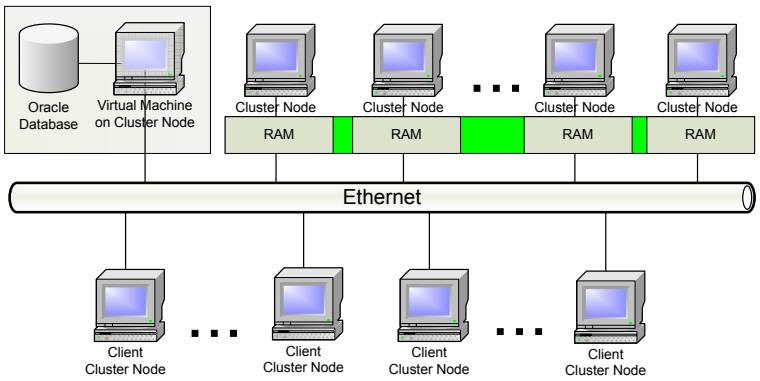


Figura 6.2: Esquema de organização dos nós do *cluster*

Os serviços instanciados nos nós desta configuração podem ser utilizados por diversos nós cliente, que executam determinadas tarefas, recorrendo à infraestrutura de armazenamento. Foi ainda instalada uma máquina virtual, contendo um servidor de gestão de base de base, a partir do qual são realizados os resumos que são, posteriormente, armazenados na infraestrutura de armazenamento.

6.3.2 Implementação do serviço

O sistema distribuído foi implementado com recurso à invocação remota de métodos (RMI) [Sun 04 04], da plataforma de desenvolvimento Java. De seguida apresentam-se os diagramas das classes utilizadas. Por questões de espaço e organização dividiu-se a totalidade do diagrama em várias partes, utilizando a class `Node` como elo comum de ligação.

O interface `Service` derivado de `java.rmi.Remote` especifica as operações disponibilizadas no serviço, sendo a sua implementação efectuada pela classe `ServiceImpl`. As operações disponibilizadas encontram-se enumeradas na Figura 6.3. A classe `Node` representa um nó físico computacional. A classe `SaveThread` é responsável pela persistência dos dados armazenados em cada nó em intervalos regulares.

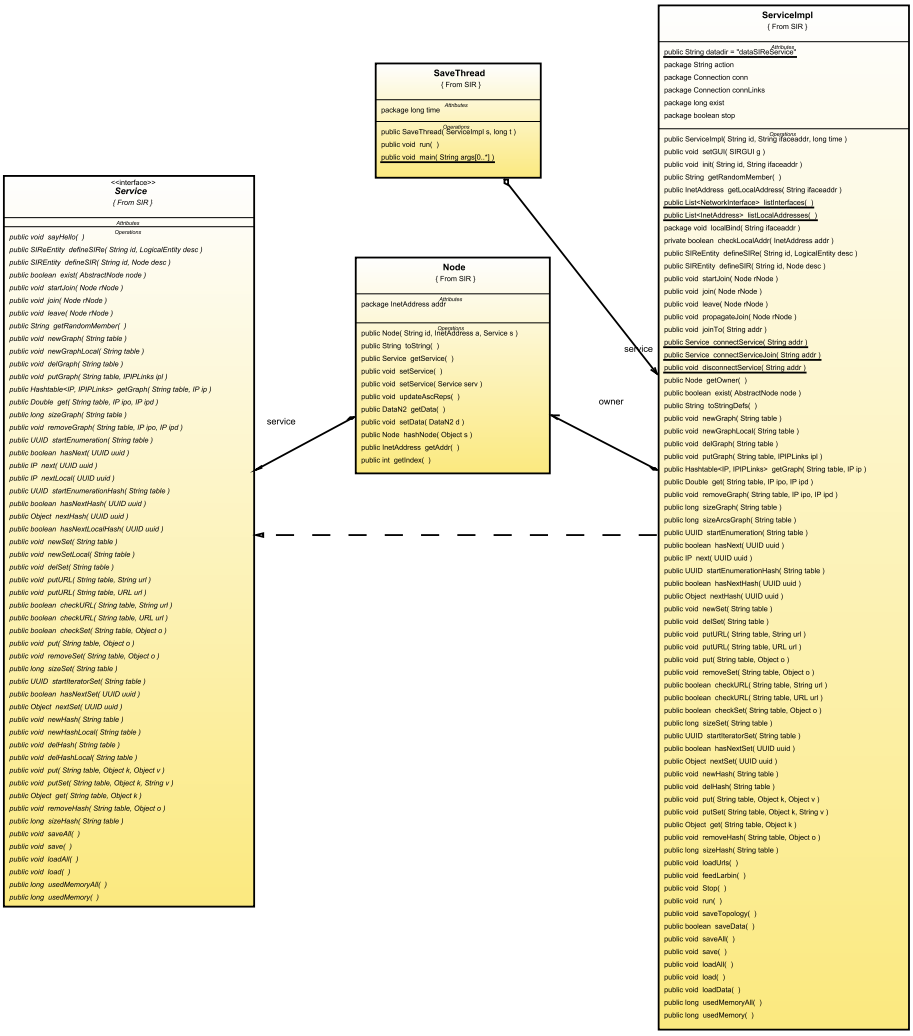


Figura 6.3: Diagrama de classes referente à implementação do serviço RMI

Das operações do serviço de invocação remota destacam-se as seguintes:

- Gestão de entidades lógicas.
- Gestão de grafos.
- Gestão de conjuntos.
- Gestão de dicionários.
- Gestão de iteradores de estruturas de dados.

Gestão de topologias lógicas

Para além do armazenamento distribuído de estruturas de dados, a infraestrutura implementada suporta a definição das entidades lógicas da topologia SIRe descritas na secção 3.2.

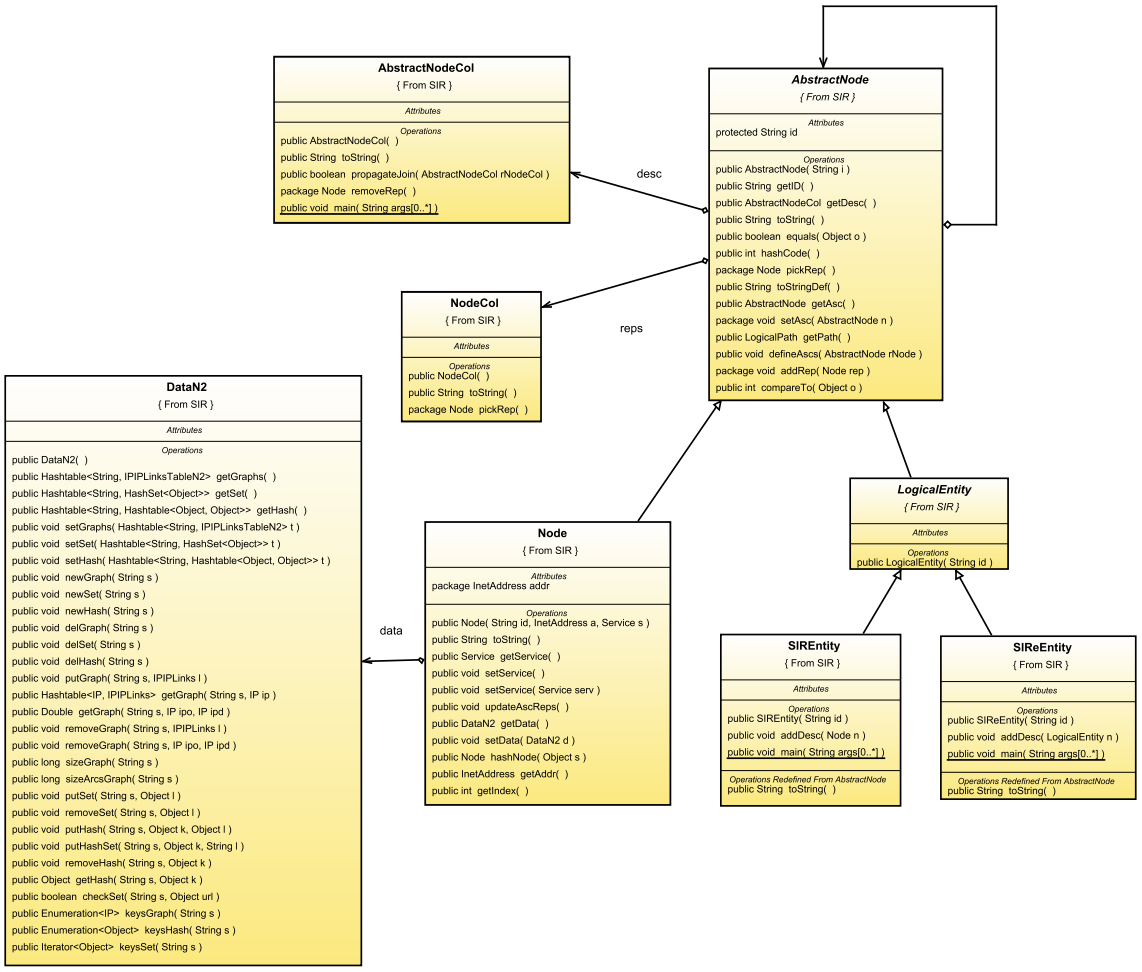


Figura 6.4: Diagrama de classes referente à modelação dos nós

Os dados necessários para a realização dos mecanismos de partição encontram-se armazenados numa base de dados Oracle. No entanto, a frequência de acesso a alguns destes dados (tais como, os RTTs e ligações entre os nós e o número de páginas em cada nó) por parte dos algoritmos de partição torna o processo muito lento. Por outro lado, existe ainda a necessidade de criar estruturas de dados temporárias que sendo armazenadas e acedidas em memória secundária tornar-se-iam igualmente

pouco eficientes.

De forma, a acelerar este processo tentou-se colocar a informação mais frequentemente utilizada em memória central. Como seria de esperar a memória central de uma única máquina manifestou-se claramente insuficiente, pelo que, se optou pela criação de um pequeno protótipo de estruturas de dados distribuídas em memória central por um conjunto de máquinas. Entre as estruturas de dados utilizados, implementaram-se, tabelas de dispersão, conjuntos e representações de grafos.

Apesar do aumento de desempenho obtido com estas estruturas, comparativamente ao Oracle, este continua a ser utilizado na preparação inicial das estruturas distribuídas e também em noutras consultas que envolvem um modelo relacional, tais como, a persistência de alguma da informação.

Para implementar o serviço descrito, é executado um processo servidor por cada nó pertencente ao sistema distribuído em questão. Cada serviço funciona como um demónio e encarrega-se de responder ao acesso e armazenamento de dados que são requeridos remotamente. Um outro processo (cliente) encarrega-se de efectuar os algoritmos de partição acedendo e criando dados armazenados nos nós servidor.

6.4 Sistema de simulação da descarga

As experiências relacionadas com a descarga, à excepção daquelas baseadas no cálculo algébrico, assentaram numa plataforma de simulação criada de raiz para este fim.

Em termos de nomenclatura da área de simulação foram definidos dois recursos básicos: o servidor, os robôs; e o cliente, os URLs. Os robôs recebem os URLs e colocam-os numa fila de espera. Para além, do esquema tradicional de eventos temporais, geridos por uma fila prioritária, responsável pela indicação dos momentos de descarga, duração das descargas, envio de URLs para outros robôs e duração destes envios, foi ainda implementado um sistema adicional para a gestão da política de delicadeza.

Em termos de eventos, Definiram-se os seguintes eventos, representados na Figura 6.6: chegada de URL ao robô (`Chegada`), início de descarga de página (`InicioDescarga`) e fim descarga da página (`FimDescarga`).

O evento `EventoChegada` modela a chegada de novos URLs a um robô, quer os URLs semente iniciais, quer os URLs encaminhados pelo próprio robô ou outros.

Quando um URL chega a um robô é adicionado na sua fila de espera de URLs, e é escalonado um evento de início de descarga, quando o URL é enviado pela semente inicial.

O evento `InicioDescarga` modela o início de uma descarga de uma página. Internamente, cada robô mantém uma fila prioritária de tempos e servidores, de modo a garantir as políticas de delicadeza. Caso exista um URLs disponível para descarga e que preserve as políticas de delicadeza, é agendado um evento de fim de descarga, caso contrário, é agendado um novo início de descarga.

O evento `FimDescarga` modela o final da descarga de uma página. Neste instante são encaminhados as hiper-ligações da página descarregada, sendo gerado um evento de chegada para cada uma. Sendo detectada ociosidade no robô é ainda gerado um evento de início de descarga.

Para realizar a gestão da política de delicadeza foi utilizada outra fila prioritária que organiza os momentos de descarga para cada servidor de modo a impedir o acesso ao servidor antes do tempo estipulado de delicadeza.

6.5 Avaliação do sistema de suporte de dados

O sistema de suporte de dados distribuído foi utilizado em diversas fases do trabalho desenvolvido, nomeadamente, na concretização dos algoritmos de partição e na sua avaliação através da utilização de um simulador de descarga.

Devido ao volume de dados utilizados foi utilizado um motor de base de dados para armazenar a informação. Além disso, para tornar mais célere a concretização das experiências de simulação, era foi possível executar diversas simulações em paralelo em que se fazia variar um conjunto de parâmetros.

Apesar da robustez e do bons desempenhos anunciados do motor de base de dados utilizado (Oracle), as interrogações que envolviam a junção de diversas tabelas com uma quantidade avultada de dados, eram bastante demoradas.

Por estes motivos, foi criada a plataforma de suporte de dados, que foi carregada com os dados necessários à execução dos algoritmos de partição e de simulação.

A justificação desta decisão foi claramente suportada pelos resultados de desempenhos comparativos efectuados. A Figura 6.7 apresenta esses resultados, em que se compara o tempo de execução de iteração de um conjunto de dados com 5.657.734

registos com 3 campos cada. Nesta experiência é comparado o desempenho da plataforma de suporte de dados desenvolvida (SIRe), a execução de uma interrogação no Oracle com apenas uma tabela e a execução de uma interrogação no Oracle que envolve a junção de 10 tabelas, fazendo variar a mesma iteração executada concorrentemente (Número de clientes).

Na prática, a interrogação que foi necessária de realizar é a que envolve as 10 tabelas, contudo apresentou-se a interrogação com uma tabela, a fim de confirmar a existência de vantagens na utilização da plataforma de suporte de dados, em condições mais modestas, comparativamente ao Oracle.

Como se pode verificar pela figura, a utilização de uma interrogação com diversas tabelas no Oracle, em condições de acesso concorrente, torna os tempos de acesso completamente inaceitáveis, chegando atingir 3 horas, quando 4 clientes executam a iteração em simultâneo.

A interrogação a uma tabela no Oracle obtém um bom desempenho para apenas 1 cliente, com um tempo de execução mais baixo que plataforma SIRe. Contudo à medida que se aumenta o número de clientes em simultâneo, o Oracle perde eficiência para a plataforma SIRe, obtendo este último uma redução de cerca de 68% em relação ao primeiro.

Figura 6.5: Diagrama de classes referente a funcionalidades auxiliares

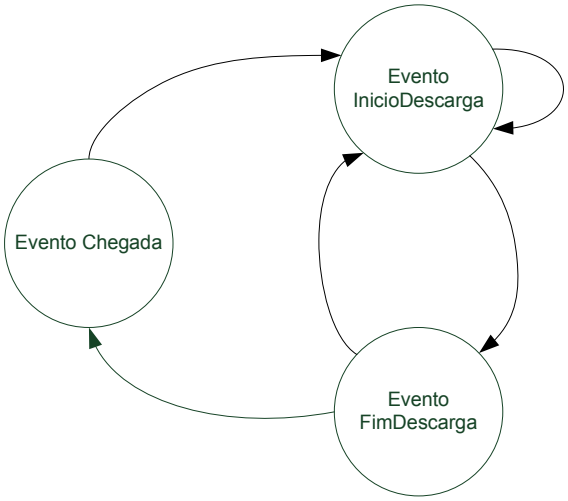


Figura 6.6: Diagrama de eventos do simulador de descarga

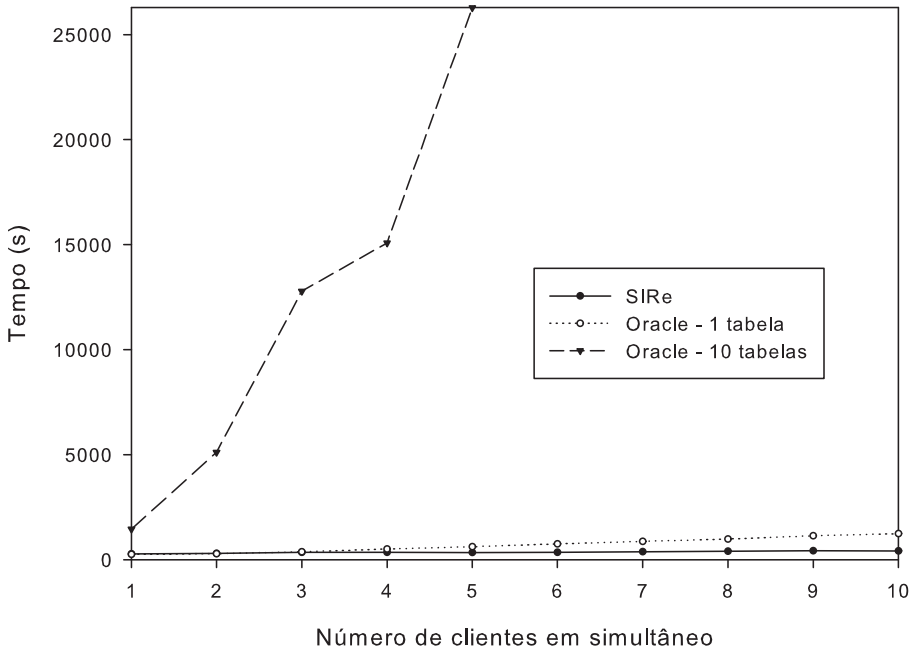


Figura 6.7: Comparação do desempenho do sistema de armazenamento distribuído (SIRe) com o Oracle

Capítulo 7

Discussão e trabalho futuro

O trabalho desenvolvido nesta dissertação utiliza a recolha adicional de informação retirada da Internet, como suporte à optimização da descarga da *Web* e à redução da sobrecarga de comunicação na cooperação de baterias de robôs distribuídos, utilizando dados topológicos, geográficos e de hiperligações entre as páginas. Foi apresentada uma arquitectura de descarga que permite suportar este paradigma descarga, através da organização hierárquica de entidades lógicas que suportam descarga focada e a gestão de tabelas de encaminhamento, resultantes da partição do espaço *Web*.

A divisão de trabalho, pelos diversos agentes, é realizada através da partição de grafos criados com a informação recolhida, combinando factores, tais como, a latência de comunicação entre os agentes e os servidores *Web*, a distância geográfica entre os agentes e os servidores *Web* e a infraestrutura de hiperligações entre as páginas. Para realizar esta recolha foi necessário implantar um sistema (1) de descoberta da topologia da Internet e utilizar heurísticas para determinar, de forma aproximada, a latência entre os pares em falta e (2) de determinação da localização geográfica de máquinas.

Os mecanismos de partição de grafos tradicionais, tendo-se revelado inadequados, foram adaptados às características intrínsecas do tipo de grafos aqui utilizados, nomeadamente, devido às características da distribuição da lei da potência da quantidade de páginas nos servidores *Web* e a necessidade de bloqueio de alguns vértices, referentes a entidades inamovíveis, como é caso dos robôs.

Inicialmente foi utilizada uma metodologia de avaliação, baseada no cálculo algébrico de um conjunto de métricas, que anteciparam a obtenção de resultados muito pro-

missores das técnicas de partição de grafos, comparadas com a utilização de funções de dispersão, nomeadamente, na redução da carga de comunicação entre os robôs. Contudo, optou-se por criar, adicionalmente, um motor de simulação de descarga que permitisse uma representação mais fiel das condições de funcionamento de um sistema descarga real.

Adicionalmente e, também incorporado no motor de simulação, criou-se uma infraestrutura de suporte para grandes volumes de dados e execução de tarefas distribuídas, que se revelou muito eficiente comparada com os tradicionais sistemas de gestão de bases de dados, em particular, nos acessos concorrentes.

7.1 Arquitectura de descarga

Tipicamente, os sistemas de descarga existentes na literatura utilizam funções de dispersão para a distribuição dos URLs pelos agentes envolvidos no processo de descarga distribuído, abdicando de informação adicional na decisão de encaminhamento. A arquitectura de descarga distribuída, aqui apresentada, coaduna-se com a inclusão de tabelas de encaminhamento no processo de distribuição dos URLs, resultantes de um processo de partição do espaço *Web* envolvido.

Estas tabelas de encaminhamento são baseadas em resumos de agregados de endereços de máquinas, como aqueles utilizados no encaminhamento TCP/IP através do mecanismo CIDR, que atinge um elevado grau de síntese.

A criação de uma organização hierárquica da infraestrutura de descarga permite uma síntese adicional nas tabelas de encaminhamento e possibilita a utilização de regras textuais para a descarga focalizada de conteúdos, aos níveis dos domínios de rede, temático e de contexto geográfico.

Cada nó lógico desta arquitectura hierárquica dispõe de uma entidade *particionadora*, encarregue pela atribuição da responsabilidade de descarga das páginas pelos seus descendentes associados. Por sua vez, os descendentes possuem uma entidade similar, responsável por essa atribuição, mas limitada ao espaço delegado pelo nível anterior. Cada um destes nós executa os algoritmos de partição de grafos para a criação das partições de forma localizada, utilizando as técnicas discutidas no Capítulo 5.

Uma vez que partição de grafos efectuada em cada nó da hierarquia concerne apenas ao espaço delegado pelo nível anterior, existe uma forte localidade nos algoritmos

de partição, impedindo uma optimização mais global.

Uma hipótese de trabalho futuro é a adaptação das técnicas de partição de grafos multi-nível e a sua sobreposição com o modelo hierárquico da arquitectura, de modo a permitir aos algoritmos de partição uma maior visibilidade e, por consequência, uma maior optimização.

7.2 Sistema integrado de modelação da Internet e descarga

A criação de mecanismos de optimização da descarga da *Web*, descritos neste trabalho, teve como motivação principal, a utilização de medidas de proximidade, calculadas a partir dos RTT, ou da localização geográfica entre os agentes de descarga e os servidores *Web*. A descoberta da topologia da Internet encaixa, convenientemente, numa arquitectura de descarga, motivada pela sua dispersão geográfica e pelo potencial elevado número de agentes, nos quais se pode colocar um ponto de sondagem dos RTTs. Estas características, aliadas ao envio de múltiplas sondas por cada ponto de origem, torna possível a estimação com grande precisão do RTT entre dois pontos bem conhecidos, calculando a mediana dos valores de RTT recolhidos, limitados a um intervalo de tempo passado.

A descoberta de novos URLs, por parte de cada agente, conduz ao reconhecimento de novas máquinas. Os algoritmos de partição de grafos que necessitam de valores do RTT de todos os agentes disponíveis para todos os servidores conhecidos, impõem a necessidade de, ocasionalmente, recorrer a aproximações do RTT entre pontos desconhecidos, utilizando a modelação da Internet já existente.

Para além da aplicação prática e imediata da descoberta da topologia da Internet, especificamente, para esta investigação, a infraestrutura instalada poderá alimentar outros trabalhos que utilizem este tipo de informação.

Neste trabalho apenas foram utilizadas técnicas de cálculo do caminho mais curto para a descoberta de RTT entre pontos desconhecidos. Esta técnica, embora simples, evita a instalação de infraestruturas de medição adicionais nos servidores *Web*, que certamente seriam impossíveis de concretizar, tendo em conta a sua quantidade e a sua privacidade. Mesmo assim, foram atingidos cerca 90% de acertos entre a aproximação e a medição real, utilizando uma origem para diversos destinos.

Um forma de aumentar esta taxa de acerto na aproximação, seria através da utilização de um maior número de origens. Seria também interessante verificar qual o número mínimo destas origens de modo a atingir-se a melhor aproximação.

7.3 Estratégias de partição

A distribuição de URLs através de funções de dispersão constitui, sem dúvida, um excelente mecanismo de divisão do espaço *Web*, considerando a ausência de informação adicional para a tomada dessa decisão. Devido a essa ausência, a inclusão de informação topológica de proximidade e da estrutura de hiperligações surgiu, inicialmente, como uma forte motivação para a criação de sistemas de descarga inteligentes mais eficientes.

Efectivamente, a utilização de mecanismos de partição de grafos, que optimizam o balanceamento entre os vértices e minimizam o corte dos ramos de um grafo, revelou-se uma abordagem promissora em termos de descarga da *Web*, tal como atestam os resultados obtidos com a utilização de expressões algébricas e por simulação. Outros trabalhos já orientaram a investigação neste sentido, contudo, não foram observados resultados práticos da sua eficiência, nem abordadas questões de delicadeza que, podem condicionar de forma acentuada a descarga das páginas.

A utilização da plataforma de simulação veio aproximar o modelo da realidade e confirmou a vantagem na utilização dos mecanismos de partição de grafos em relação às abordagens baseadas em função de dispersão, principalmente, no processo de trocas de URLs entre os robôs, onde se atingiram reduções na percentagem de trocas de URLs inter-partição na ordem dos 76%. Contudo, no tempo de descarga, as diferenças verificadas foram mínimas, quando se almejava a preservação das políticas de delicadeza.

Uma forma de acentuar esta diferença passaria pela diminuição do grão da unidade de distribuição, através da divisão das páginas do mesmo servidor por vários robôs, conduzindo, inevitavelmente, à violação descontrolada das políticas de delicadeza, uma vez que existiriam vários robôs a descarregar páginas do mesmo servidor, dificultando o processo de escalonamento de descarga.

Com o pressuposto de que os servidores mais densos deverão ter maior capacidade de resposta aos pedidos `http`, o mesmo objectivo pôde ser alcançado criando uma política de delicadeza ajustável, inversamente proporcional à dimensão do servidor.

Esta técnica oferece, simultaneamente, diversas vantagens. Por um lado, a gestão do acesso concorrente aos servidores fica simplificada, obtendo-se um controle sobre o tempo de delicadeza a atribuir a cada servidor, uma vez que apenas um robô acede ao mesmo servidor. Além disso, aproveita-se a vantagem do resultado do processo de partição de grafos que aproxima o servidor do robô mais adequado para se atingir a optimização resultante dos mecanismos de partição. Por fim, mantendo um servidor por robô, são evitadas trocas de URLs adicionais.

A técnica de ajuste da política de delicadeza trouxe uma diminuição considerável do tempo de descarga em relação às técnicas que utilizam funções de dispersão, atingindo uma redução de cerca de 91%, mantendo os mesmos valores de percentagem de trocas de URLs inter-partição, quando não é usado este ajuste na partição de grafos.

Sendo o ajuste do tempo de delicadeza inversamente proporcional à densidade do servidor e estabelecido num intervalo predefinido, a sua média para uma descarga completa sofre uma redução. Mesmo aplicando, à função de dispersão, um tempo de delicadeza constante igual à média do tempo de delicadeza ajustado das técnicas de partição de grafos, obtêm-se reduções bastante satisfatórias na ordem dos 69%. Nestas experiências, verificou-se, ainda, uma tendência de escalabilidade mais acentuada nas abordagens baseadas na partição de grafos do que nas de função de dispersão, tendo-se observado uma diminuição de quase um dia no tempo de descarga, quando o número de robôs é aumentado de 2 para 30. A função de dispersão revelou inicialmente, não só, um tempo descarga crescente com o aumento do número de robôs, com também, uma tendência posterior para a estabilização.

Quando se comparou a partição de grafos com a função de dispersão aplicando um tempo de delicadeza dinâmico proporcional ao tempo de descarga da página anterior, foi possível obter uma redução no tempo de descarga. Apesar da maior facilidade no cálculo do tempo de delicadeza dinâmico, o tempo de delicadeza ajustável com a densidade dos servidores traz maiores reduções no tempo de descarga.

A utilização de tempos de delicadeza ajustados ou dinâmicos não afectou os tempos de trocas, obtendo-se ainda vantagens na salvaguardada uma delicadeza adequada para os servidores com menores capacidades de resposta. Contrariamente, a função de dispersão, além de obter tempos de trocas maiores, não distingue a capacidade dos servidores, quando se aplica o mesmo tempo de delicadeza para todos os servidores.

A determinação da localização geográfica das máquinas, e o posterior cálculo de distâncias geográficas entre elas, é um processo bastante menos oneroso do que a

recolha dos RTTs. As experiências realizadas por simulação que utilizam a distância geográfica como medida dos ramos dos grafos obtiveram resultados muito próximos dos obtidos utilizando o RTT, e substancialmente melhores dos obtidos pela função de dispersão. Nesta conjectura, a utilização da distância geográfica pode ser suficiente para optimizar a descarga da *Web* e a sobrecarga de comunicação entre os robôs.

O processo de partição de grafos utilizado combina dois tipos de medidas (RTT ou distância geográfica com a quantidade de hiperligações) num único grafo. Uma vez que está em causa a optimização de duas métricas (tempo de descarga e tempo de trocas), cada uma delas é directamente afectada pelo peso atribuído a cada medida, no processo de combinação dos grafos.

No que respeita à utilização de IPs como unidade de partição em substituição dos nomes das máquinas, todas as experiências apresentadas, à excepção da função de partição, mostram, que é mais vantajoso utilizar os nomes das máquinas. Apesar de não ter sido possível efectuar a confirmação, estamos convictos que a partição de grafos requer uma parametrização diferente ao nível dos pesos atribuídos às medidas combinadas.

Como trabalho futuro, seria interessante analisar detalhadamente, de que forma a variação destes pesos altera os resultados obtidos com os IPs e como influencia, quer o tempo de descarga, quer o tempo de trocas em todas as experiências, e inferir os pesos ideais para se obter o melhor desempenho em ambas as métricas.

Por último, devido à concentração na problemática da optimização do tempo de descarga originada pela granularidade da unidade de partição e a forma de lidar com a política de delicadeza, pouca atenção foi prestada à afinação dos algoritmos de partição. Foi necessário abdicar das técnicas multi-nível para se obterem tempos de execução aceitáveis. Além disso, foi necessário retirar algumas heurísticas do algoritmo para manter a sua a eficácia. Para trabalho futuro, recomenda-se um aprofundamento nesta área, no que diz respeito a uma melhor compatibilização com o tipo de grafos aqui utilizados, em que os vértices robô conectam com todos os vértices servidores, formando um grafo semelhante a um grafo bipartes, que torna ineficiente o engrossamento do grafo no processo de partição.

Bibliografia

[Agency 06] US National Geospatial Intelligence Agency. *Geographic Names Database.* `http://earth-info.nima.mil/gns/html/index.html`, 2006.

[Ailleret 03] Sebastien Ailleret. *http://larbin.sourceforge.net/index-eng.html*, 2003.

[Anh 02] Vo Ngoc Anh e Alistair Moffat. *Impact transformation: effective and efficient web retrieval.* In SIGIR '02: Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval, pp. 3–10, New York, NY, USA, 2002. ACM.

[Baeza-Yates 02] Ricardo Baeza-Yates e Carlos Castillo. *Balancing Volume, Quality and Freshness in Web Crawling.* In Hybrid Intelligent Systems, 2002.

[Baeza-Yates 07] Ricardo Baeza-Yates, Carlos Castillo, Flavio Junqueira, Vassilis Plachouras e Fabrizio Silvestri. *Challenges in Distributed Information Retrieval.* In International Conference on Data Engineering (ICDE), Istanbul, Turkey, April 2007. IEEE CS Press.

[Bajaj 99] Sandeep Bajaj, Lee Breslau, Deborah Estrin, Kevin Fall, Sally Floyd, Padma Haldar, Mark Handley, Ahmed Helmy, John Heidemann, Polly Huang, Satish Kumar, Steven McCanne, Reza Rejaie, Puneet Sharma, Kannan Varadhan, Ya Xu, Haobo Yu e Daniel Zappala. *Improving Simulation for Network Research.* Technical report 99-702b, University of Southern California, March 1999. revised September 1999, to appear in IEEE Computer.

[Barabási 99] Albert-László Barabási e Réka Albert. *Emergence of Scaling in Random Networks*. SIAM Journal on Scientific Computing, vol. 286, no. 5439, pp. 509 – 512, 1999.

[Barnard 94] Stephen T. Barnard e Horst D. Simon. *A Fast Multilevel Implementation of Recursive Spectral Bisection for Partitioning Unstructured Problems*. Concurrency: Practice and Experience, vol. 6, pp. 101–107, 1994.

[Berners-Lee 90] T. Berners-Lee e R. Cailliau. *WorldWideWeb: Proposal for a HyperText Project*. Proposal, CERN, 1990.

[Bloom 70] Burton H. Bloom. *Space/time trade-offs in hash coding with allowable errors*. Commun. ACM, vol. 13, no. 7, pp. 422–426, 1970.

[Boldi 02] P. Boldi, B. Codenotti, M. Santini e S. Vigna. *Ubicrawler: A scalable fully distributed web crawler*. Software: Practice & Experience, vol. 34, no. 8, pp. 711–726, 2002.

[Brin 98] Sergey Brin e Lawrence Page. *The anatomy of a large-scale hypertextual Web search engine*. In WWW7: Proceedings of the seventh international conference on World Wide Web 7, pp. 107–117, Amsterdam, The Netherlands, The Netherlands, 1998. Elsevier Science Publishers B. V.

[Burner 97] Mike Burner. *Crawling towards Eternity. Building An Archive of The World Wide Web*. Web Techniques Magazine, vol. 2, no. 5, Maio, 1997.

[Búttcher 06] Stefan Búttcher e Charles L. A. Clarke. *A Hybrid Approach to Index Maintenance in Dynamic Text Retrieval Systems*. In ECIR, pp. 229–240, 2006.

[Caceres 99] R. Caceres, N. Duffield, J. Horowitz e D. Towsley. *Multicast-Based Inference of Network-Internal Loss Characteristics*. IEEE Transactions on Information Theory, vol. 45, no. 7, pp. 2462 – 2480, 1999.

[CAIDA 02] CAIDA. *NetGeo - The Internet Geographic Database*. `http://www.caida.org/tools/utilities/netgeo/`, 2002.

[CAIDA 07] CAIDA. *Macroscopic Topology Measurements*. `http://www.caida.org/analysis/topology/macroscopic/`, 2007.

[Cao 00] Jin Cao, Scott Vander Wiel, , Bin Yu e Zhengyuan Zhu. *A Scalable Method for Estimating Network Traffic Matrices from Link Counts*. Technical report, Bell Labs, 2000.

[Castillo 04a] Carlos Castillo. *Effective Web Crawling*. PhD thesis, School of Engineering, Santiago, Chile, December 2004.

[Castillo 04b] Carlos Castillo, Mauricio Marin, Andrea Rodriguez e Ricardo Baeza-Yates. *Scheduling Algorithms for Web Crawling*. In LA-WEBMEDIA '04: Proceedings of the WebMedia & LA-Web 2004 Joint Conference 10th Brazilian Symposium on Multimedia and the Web 2nd Latin American Web Congress, pp. 10–17, Washington, DC, USA, 2004. IEEE Computer Society.

[Castro 04] Rui Castro, Mark Coates, Gang Liang, Robert Nowak e Bin Yu. *Network Tomography: Recent Developments*. Technical report, Project Euclid (Hosted at Cornell University Library) [http://ProjectEuclid.org/Dienst] (United States), 2004.

[Cheswick 00] Bill Cheswick, Hal Burch e Steve Branigan. *Mapping and visualizing the internet*. In ATEC'00: Proceedings of the Annual Technical Conference on 2000 USENIX Annual Technical Conference, pp. 1–1, Berkeley, CA, USA, 2000. USENIX Association.

[Cho 98] Junghoo Cho, Hector Garcia-Molina e Lawrence Page. *Efficient crawling through URL ordering*. Comput. Netw. ISDN Syst., vol. 30, no. 1-7, pp. 161–172, 1998.

[Cho 00a] Junghoo Cho e Hector Garcia-Molina. *The Evolution of the Web and Implications for an Incremental Crawler*. In The VLDB Journal, pp. 200–209, 2000.

[Cho 00b] Junghoo Cho e Hector Garcia-Molina. *Synchronizing a database to improve freshness*. In SIGMOD '00: Proceedings of the 2000 ACM SIGMOD international conference on Management of data, pp. 117–128, New York, NY, USA, 2000. ACM Press.

[Cho 02] Junghoo Cho e Hector Garcia-Molina. *Parallel crawlers*. In WWW '02: Proceedings of the 11th international conference on World Wide Web, pp. 124–135, New York, NY, USA, 2002. ACM Press.

[Cho 03] Junghoo Cho e Hector Garcia-Molina. *Effective page refresh policies for Web crawlers*. ACM Trans. Database Syst., vol. 28, no. 4, pp. 390–426, 2003.

[Coates 00] M. Coates e R. Nowak. *Network loss inference using unicast end-to-end measurement*. In ITC Seminar on IP Traffic, Measurement and Modelling, 2000.

[Coates 01] M. Coates e R. Nowak. *Network tomography for internal delay estimation*. In IEEE International Conference on Acoustics, Speech and Signal Processing, 2001.

[ComScore 07] ComScore. *ComScore - Measuring the Digital World*. `http://www.comscore.com`, 2007.

[Cutting 90] D. Cutting e J. Pedersen. *Optimization for dynamic inverted index maintenance*. In SIGIR '90: Proceedings of the 13th annual international ACM SIGIR conference on Research and development in information retrieval, pp. 405–411, New York, NY, USA, 1990. ACM.

[Doar 96] M. Doar. *A Better Model for Generating Test Networks*. In Globecom '96, 1996.

[Duffield 06] Nick Duffield, Francesco Lo Presti, Vern Paxson e Don Towsley. *Network loss tomography using striped unicast probes*. IEEE/ACM Trans. Netw., vol. 14, no. 4, pp. 697–710, 2006.

[Eichmann 94] D. Eichmann. *The RBSE spider: balancing effective search against web load*. In First World Wide Web Conference, 1994.

[Exposto 03] José Exposto, António Pina, Joaquim Macedo, Albano Alves e José Rufino. *Um Modelo Cooperativo e Distribuído para a Recuperação de Informação na WWW*. In 6a Conferência sobre Redes de Computadores (CRC'2003), Bragança, Portugal, 2003.

[Exposto 04] José Exposto, António Pina, Joaquim Macedo, Albano Alves e José Rufino. *Relações entre a Topologia de rede e a Geografia dos servidores WWW em Portugal*. In VII Encontro de utilizadores de informação geográfica (eSIG 2005), Oeiras, Portugal, 2004.

[Exposto 05] José Exposto, Joaquim Macedo, António Pina, Albano Alves e José Rufino. *Geographical Partition for Distributed Web Crawling*. In 2nd International ACM Workshop on Geographic Information Retrieval (GIR 2005), pp. 55–60, Bremen, Germany, November 2005. ACM Press.

[Exposto 07] José Exposto, Joaquim Macedo, António Pina, Albano Alves e José Rufino. *Efficient Partitioning Strategies for Distributed Web Crawling*. In 21st International Conference on Information Networking (ICOIN 2007), Estoril, Portugal, 2007.

[Exposto 08] José Exposto, Joaquim Macedo, António Pina, Albano Alves e José Rufino. *Efficient Partitioning Strategies for Distributed Web Crawling*. In LNCS 5200, volume 5200, pp. 544–553. Springer-Verlag, Novembro 2008. Revised Selected Papers of the 21st International Conference on Information Networking.

[Fiduccia 82] C. M. Fiduccia e R. M. Mattheyses. *A linear-time heuristic for improving network partitions*. In DAC '82: Proceedings of the 19th conference on Design automation, pp. 175–181, Piscataway, NJ, USA, 1982. IEEE Press.

[Francis 01] Paul Francis, Sugih Jamin, Cheng Jin, Yixin Jin, Danny Raz, Yuval Shavitt e Lixia Zhang. *IDMaps: a global internet host distance estimation service*. IEEE/ACM Trans. Netw., vol. 9, no. 5, pp. 525–540, 2001.

[Gerich 93] E. Gerich. *Guidelines for Management of IP Address Space*. RFC 1466 (Informational), May 1993. Obsoleted by RFC 2050.

[Google 07] Google. *http://www.google.com*, 2007.

[Group 93] Internet Engineering Steering Group e R. Hinden. *Applicability Statement for the Implementation of Classless Inter-Domain Routing (CIDR)*. RFC 1517 (Historic), September 1993.

[Group 03] XLDB Group. *WPT03*. Linguateca, http://www.linguateca.pt, 2003.

[Group 07] Miniwatts Marketing Group. *Internet World Stats - Usage and Population Statistics*. `http://www.internetworldstats.com/`, 2007.

[Gulli 05] A. Gulli e A. Signorini. *The indexable web is more than 11.5 billion pages*. In WWW '05: Special interest tracks and posters of the 14th international conference on World Wide Web, pp. 902–903, New York, NY, USA, 2005. ACM Press.

[Henzinger 06] Monika Henzinger. *Finding near-duplicate web pages: a large-scale evaluation of algorithms*. In SIGIR '06: Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval, pp. 284–291, New York, NY, USA, 2006. ACM Press.

[Heydon 99] Allan Heydon e Marc Najork. *Mercator: A scalable, extensible Web crawler*. World Wide Web, vol. 2, no. 4, pp. 219–229, 1999.

[Horton 03] Joseph D. Horton e Alejandro López-Ortiz. *On the number of distributed measurement points for network tomography*. In IMC '03: Proceedings of the 3rd ACM SIGCOMM conference on Internet measurement, pp. 204–209, New York, NY, USA, 2003. ACM.

[ISC 07] Internet Systems Consortium Inc. `http://www.isc.org/index.pl`, 2007.

[Jacobson 88] V. Jacobson. *traceroute Software*. ftp://ftp.ee.lbl.gov/traceroute.tar.Z, 1988.

[jung 05] *JUNG the Java Universal Network/Graph Framework*. http://jung.sourceforge.net/, 2005.

[Karger 97] David Karger, Eric Lehman, Tom Leighton, Mathhew Levine, Daniel Lewin e 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, pp. 654–663, May 1997.

[Karypis 98a] G. Karypis e V. Kumar. *A Software Package for Partitioning Unstructured Graphs, Partitioning Meshes, and Computing Fill-Reducing Orderings of Sparse Matrices – Version 4.0*. Technical report, University of Minnesota, Department of Computer Science / Army HPC Research Center, 1998.

[Karypis 98b] George Karypis e Vipin Kumar. *A Fast and High Quality Multilevel Scheme for Partitioning Irregular Graphs*. SIAM Journal on Scientific Computing, vol. 20, no. 1, pp. 359 – 392, 1998.

[Karypis 98c] George Karypis e Vipin Kumar. *Multilevel k-way Partitioning Scheme for Irregular Graphs*. Journal of Parallel and Distributed Computing, vol. 48, no. 1, pp. 96–129, 1998.

[Kernighan 70] B. W. Kernighan e S. Lin. *An Efficient Heuristic Procedure for Partitioning Graphs*. Bell Sys. Tech. J., vol. 49, no. 2, pp. 291–308, 1970.

[Koster] Martijn Koster. *A Standard for Robot Exclusion*. `http://www.robotstxt.org/orig.html`.

[Kumar 04] Ritesh Kumar e Jasleen Kaur. *Efficient beacon placement for network tomography*. In IMC '04: Proceedings of the 4th ACM SIGCOMM conference on Internet measurement, pp. 181–186, New York, NY, USA, 2004. ACM.

[Lawrence 00] Steve Lawrence e C. Lee Giles. *Accessibility of information on the Web*. Intelligence, vol. 11, no. 1, pp. 32–39, 2000.

[Lawrence 05] E. Lawrence, G. Michailidis, V.N Nair e B. Xi. *Network Tomography: A Review and Recent Developments*. In Jianqing Fan e Hira L Koul, editores, Frontiers in Statistics. World Scientific Publishing, 2005.

[Leguay 05] Jeremie Leguay, Matthieu Latapy, Timur Friedman e Kavé Salamatian. *Describing and Simulating Internet Routes*. In NETWORKING 2005: 4th International IFIP-TC6 Networking Conference, pp. 659–670, 2005.

[Loo 04] B. Loo, S. Krishnamurthy e O. Cooper. *Distributed Web Crawling over DHTs*. Technical report, EECS Department, University of California, Berkeley, 2004.

[Markatos 96] Evangelos Markatos. *Using Remote Memory to avoid Disk Thrashing: A Simulation Study*. In MASCOTS '96: Proceedings of the 4th International Workshop on Modeling, Analysis, and Simulation of Computer and Telecommunications Systems, page 69, Washington, DC, USA, 1996. IEEE Computer Society.

[Martins 05] Maria Eugénia Graça Martins. *Introdução à Análise de Dados I*, 2005.

[MaxMind 07] MaxMind. *http://www.maxmind.com/app/ip-location*, 2007.

[Murteira 02] Bento Murteira, Carlos Silva Ribeiro, João Andrade e Silva e Carlos Pimenta. Introdução à estatística. Mc Graw Hill, 2002.

[Najork 01] Marc Najork e Janet L. Wiener. *Breadth-first crawling yields high-quality pages*. In WWW '01: Proceedings of the 10th international conference on World Wide Web, pp. 114–118, New York, NY, USA, 2001. ACM.

[Najork 02] Marc Najork e Allan Heydon. *High-performance web crawling*. Handbook of massive data sets, pp. 25–45, 2002.

[NCC 07] RIPE NCC. *RIPE Region Hostcount*. `http://www.ripe.net/`, 2007.

[Netcraft 07] Netcraft Ltd. `http://news.netcraft.com/`, 2007.

[Oracle 07] Oracle. *http://www.oracle.com*, 2007.

[Papapetrou 04] Odysseas Papapetrou e George Samaras. *IPMicra: Toward a Distributed and Adaptable Location Aware Web Crawler*. In ADBIS (Local Proceedings), 2004.

[Periakaruppan 99] Ram Periakaruppan e Evi Nemeth. *GTrace: A Graphical Traceroute Tool*. In 13th Conference on Systems Administration (LISA-99), pp. 69–78, 1999.

[Pothen 90] Alex Pothen, Horst D. Simon e Kan-Pu Liou. *Partitioning sparse matrices with eigenvectors of graphs*. SIAM J. Matrix Anal. Appl., vol. 11, no. 3, pp. 430–452, 1990.

[Pothen 92] A. Pothen, H. D. Simon, L. Wang e S. T. Barnard. *Towards a fast implementation of spectral nested dissection*. In Supercomputing '92: Proceedings of the 1992 ACM/IEEE conference on Supercomputing, pp. 42–51, Los Alamitos, CA, USA, 1992. IEEE Computer Society Press.

[Rabbat 04] Michael Rabbat, Robert Nowak e Mark Coates. *Multiple Source, Multiple Destination Network Tomography*. In IEEE Infocom, 2004.

[Rekhter 93] Y. Rekhter e C. Topolcic. *Exchanging Routing Information Across Provider Boundaries in the CIDR Environment*. RFC 1520 (Historic), September 1993.

[Rufino 04] José Rufino, António Pina, Albano Alves e José Exposto. *A cluster oriented model for dynamically balanced DHTs*. In International Parallel and Distributed Processing Symposium (IPDPS '04), Santa Fe, New Mexico, USA, April 2004. IEEE.

[Rufino 05] J. Rufino, A. Pina, A. Alves e J. Exposto. *Domus - An Architecture for Cluster-oriented Distributed Hash Tables*. In 6th International Conferenece on Parallel Processing and Applied Mathematics (PPAM'05), Poznan, Poland, September 2005. Springer-Verlag.

[Salton 83] G. Salton e M. J. Mcgill. Introduction to modern information retrieval. McGraw-Hill, Inc., New York, NY, USA, 1983.

[Saraiva 01] Patricia Correia Saraiva, Edleno Silva de Moura, Novio Ziviani, Wagner Meira, Rodrigo Fonseca e Berthier Riberio-Neto. *Rank-preserving two-level caching for scalable search engines*. In

SIGIR '01: Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval, pp. 51–58, New York, NY, USA, 2001. ACM.

[Savage 99] Stefan Savage, Andy Collins, Eric Hoffman, John Snell e Thomas Anderson. *The end-to-end effects of Internet path selection.* SIGCOMM Comput. Commun. Rev., vol. 29, no. 4, pp. 289–299, 1999.

[Schloegel 99] Kirk Schloegel, George Karypis e Vipin Kumar. *A New Algorithm for Multi-objective Graph Partitioning.* In European Conference on Parallel Processing, pp. 322–331, 1999.

[Scholer 02] Falk Scholer, Hugh E. Williams, John Yiannis e Justin Zobel. *Compression of inverted indexes For fast query evaluation.* In SIGIR '02: Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval, pp. 222–229, New York, NY, USA, 2002. ACM.

[Search 05] Universidade do Minho. *SeARCH: Serviços e Investigação em Computação Avançada com Clusters HTC/HPC, Refª: CONCREEQ/443/2001.* `http://www.di.uminho.pt/search/`, 2005.

[Shkapenyuk 02] Vladislav Shkapenyuk e Torsten Suel. *Design and Implementation of a High-Performance Distributed Web Crawler.* In ICDE '02: Proceedings of the 18th International Conference on Data Engineering, page 357, Washington, DC, USA, 2002. IEEE Computer Society.

[Siamwalla 98] R. Siamwalla, R. Sharma e S. Keshav. *Discovering Internet Topology.* Technical report, Technical Report, Cornell University, http://www.cs.cornell.edu/skeshav/papers/discovery.pdf, July 1998.

[Silva 02] Leopoldo Silva, Joaquim Macedo, António Costa, Orlando Belo e Alexandre Santos. *NetCensus: Medição da evolução dos conteúdos na Web.* In 5ª Conf sobre Redes de Computadores (CRC2002) FCCN, Faro, Portugal, Setembro 2002.

[Silva 03] Leopoldo Oliveira e Silva, Joaquim Macedo, António Costa, Orlando Belo e Alexandre Santos. *Um Sistema de Carregamento por Lotes de Conteúdos Web*. In Jornadas de Ingeniería del Software y Bases de Datos (JISBD), pp. 771–774, Alicante, Espanha, Novembro 2003.

[Singh 04] A. Singh, M. Srivatsa, L. Liu e T. Miller. *Apoidea: A Decentralized Peer-to-Peer Architecture for Crawling the World Wide Web*, 2004.

[Stoica 01] Ion Stoica, Robert Morris, David Karger, M. Frans Kaashoek e Hari Balakrishnan. *Chord: A scalable peer-to-peer lookup service for internet applications*. In SIGCOMM '01: Proceedings of the 2001 conference on Applications, technologies, architectures, and protocols for computer communications, pp. 149–160, New York, NY, USA, 2001. ACM.

[Sun 04 04] Sun Microsystems. *Java®RemoteMethodInvocation Specification*. `http://java.sun.com/j2se/1.5/pdf/rmi-spec-1.5.0.pdf`, 2004.

[Tebaldi 98] Claudia Tebaldi e Mike West. *Bayesian Inference on Network Traffic Using Link Count Data*. Journal of the American Statistical Association, vol. 93, no. 442, pp. 557–576, 1998.

[Teng 99] Shang-Hua Teng, Qi Lu, Matthias Eichstaedt, Daniel Ford e Tobin Lehman. *Collaborative Web Crawling: Information Gathering/Processing over Internet*. In Proceedings of the Thirty-second Annual Hawaii International Conference on System Sciences - Volume 5, 1999.

[Tomasic 94] Anthony Tomasic, Héctor García-Molina e Kurt Shoens. *Incremental updates of inverted lists for text document retrieval*. In SIGMOD '94: Proceedings of the 1994 ACM SIGMOD international conference on Management of data, pp. 289–300, New York, NY, USA, 1994. ACM.

[Tsang 04] Yolanda Tsang, Mehmet Yildiz, Paul Barford e Robert Nowak. *Network radar: tomography from round trip time measurements*.

In IMC '04: Proceedings of the 4th ACM SIGCOMM conference on Internet measurement, pp. 175–180, New York, NY, USA, 2004. ACM.

[Vardi 96] Y. Vardi. *Network tomography: estimating source-destination traffic intensities from link data*. Journal of the American Statistical Association, vol. 91, no. 433, pp. 365–377, 1996.

[Wang 03] S. Y. Wang, C. L. Chou, C. H. Huang, C. C. Hwang, Z. M. Yang, C. C. Chiou e C. C. Lin. *The design and implementation of the NCTUns 1.0 network simulator*. Comput. Networks, vol. 42, no. 2, pp. 175–197, 2003.

[Wang 07] Guohui Wang, Bo Zhang e T. S. Eugene Ng. *Towards network triangle inequality violation aware distributed systems*. In IMC '07: Proceedings of the 7th ACM SIGCOMM conference on Internet measurement, pp. 175–188, New York, NY, USA, 2007. ACM.

[Winick 02] Jared Winick e Sugih Jamin. *Inet-3.0: Internet Topology Generator*. http://topology.eecs.umich.edu/inet/inet-3.0.pdf, 2002.

[Wyckoff 98] P. Wyckoff, D. Ford e T. Lehman. *IBM TSpaces*. IBM Systems Journal, 1998.

[Zobel 06] Justin Zobel e Alistair Moffat. *Inverted files for text search engines*. ACM Comput. Surv., vol. 38, no. 2, page 6, 2006.