Optimização de Algoritmos P2P Através de Sistemas de Coordenadas de Rede – Um Estudo Comparativo

Rui Silva Lopes, J. Legatheaux Martins and Sérgio Duarte

***Resumo*—Este artigo apresenta um estudo comparativo sobre como alguns algoritmos bem conhecidos de determinação de coordenadas de rede sintéticas (em particular, o GNP e o Vivaldi) podem ser usados para optimizar um algoritmo P2P do ponto de vista da latência. O algoritmo a optimizar maximiza a liberdade de escolha de nós parceiros, o que permite a um sistema de coordenadas maximizar a optimização providenciada. As versões do algoritmo baseadas em coordenadas são também comparadas com uma versão aproximadamente óptima, assim como com uma versão em que os parceiros são escolhidos aleatoriamente. Esta comparação põe em evidência os prós e os contras dos diferentes algoritmos e fornece pistas sobre como avaliar a sua utilização.**

This paper presents a comparative study on how well known synthetic network-coordinates algorithms (namely, GNP and Vivaldi) can be used to optimize a P2P algorithm. The algorithm being optimized allows a great deal of freedom in choosing each node's partners, thus enabling network-coordinates systems to provide their best possible optimization. The coordinates based variants of the algorithm are also compared with an almost optimal one, as well as with a version where partners are randomly chosen. The study highlights the pros and cons of the different algorithms and gives some hints of when their usage should be considered.

I. Introdução

Na generalidade das aplicações P2P não existem, a priori, restrições estritas sobre quais os parceiros com que cada nó colabora. Essa liberdade de escolha é geralmente usada para optimização, pois tomar em consideração a "distância" (e.g., latência, capacidade de transferência, ...) entre nós na rede, pode ser benéfico para aumentar globalmente a velocidade de resposta do sistema [17], [6], [16], [4].

O projecto LiveFeeds [1] está a desenvolver algoritmos P2P de difusão filtrada (*Content-Based Networking*) usando a difusão de *Feeds RSS* como caso de estudo. A intenção é explorar a distribuição de carga entre nós e a possibilidade de optimizar a difusão filtrada [8]. Para este efeito, uma das hipóteses em estudo é manter uma visão total em cada nó da filiação e dos filtros dos outros nós [11]. Esta aproximação resulta num sistema em que é possível encaminhar qualquer mensagem num único passo [5]. Para a comunicação multi-ponto (*multicasting* ou *broadcasting*) podem ser usadas árvores, dinamicamente estabelecidas caso a caso, usando no quadro do sistema LiveFeeds uma estratégia de escolha aleatória dos nós descendentes de um nó. No entanto, esta liberdade de escolha permite ensaiar estratégias de desenho das árvores em que o custo da comunicação entre nós seja tomada em consideração. Quando o número de nós envolvidos numa alternativa é pequeno, as medidas de distância na rede podem ser realizadas através de sondas (e.g., *echo packets*, medidas de tempos de transmissão, ...). Em alternativa, quando o número de escolhas possíveis é muito grande, foram introduzidos diversos algoritmos de cálculo de coordenadas de rede sintéticas que, através de uma função distância (e.g., *euclidiana*), permitem aumentar a escalabilidade do processo de escolha de parceiros [12], [3].

Neste documento apresenta-se um estudo em que se faz uma análise da viabilidade e do interesse em usar um sistema de coordenadas sintéticas, para optimizar o algoritmo de difusão do sistema LiveFeeds, que é baseado à partida em árvores aleatórias. O estudo distingue-se de outros estudos relatados na literatura com objectivos semelhantes (e.g., [2], [4]), por o algoritmo P2P a optimizar dar liberdade quase total de escolha dos parceiros, o que potencia ao máximo os ganhos possíveis. Por outro lado, procura-se colocar a potencial optimização em perspectiva, comparando-a com a de um algoritmo quase óptimo, baseado no conhecimento perfeito da latência.

A secção II deste documento apresenta o algoritmo de difusão total (não filtrada) proposto no contexto do projecto LiveFeeds. Na secção III são apresentadas brevemente duas aproximações que permitem factorizar a estimativa da latência entre nós numa rede. A secção IV descreve como os algoritmos seleccionados para cálculo de coordenadas foram introduzidos no algoritmo de difusão e que ferramentas e critérios foram usados para realizar o seu estudo comparativo. Os resultados obtidos e a sua discussão são o objecto da secção V. Finalmente, são discutidas as características e âmbito de validade do trabalho realizado e analisadas as conclusões obtidas.

II. O Algoritmo de Difusão Usado no Estudo

Para os efeitos deste estudo, os nós do sistema P2P têm identificadores gerados aleatoriamente (por hipótese sem colisões e com uma distribuição uniforme), existe conectividade global e total entre nós (*full mesh network*), o sistema é da filosofia *One Hop Routing* [5], isto é, cada nó tem uma tabela de encaminhamento que contém o identificador e o endereço IP de todos os outros nós, e a visibilidade da filiação é consistente [2].

[1]Projecto PTDC/EIA/76114/2006, Project *LiveFeeds – P2P Dissemination of Web Syndication Content*, financiado pela FCT/MCTES.

[2]A manutenção dessa consistência está para além da presente discussão.

Quando um nó pretende difundir uma mensagem, começa por dividir o espaço de todos os identificadores (o universo) em *fanout factor* (abreviado por f) blocos de igual tamanho, e envia uma cópia da mensagem para um nó, escolhido aleatoriamente, em cada um desses blocos. Cada mensagem contém a indicação do bloco (intervalo) pelo qual o receptor assume a responsabilidade de continuar a difusão. De forma recursiva, cada nó que recebe uma mensagem, guarda uma cópia e divide de novo o seu intervalo em f blocos (agora mais estreitos) e recomeça a difusão. Este processo continua até que todos os nós que recebem uma mensagem, recebem a responsabilidade de a difundir num bloco com um número de nós $\leq f$, caso em que realizam a entrega directa.

Dada a natureza do estudo que se pretende realizar, no que se segue vamos admitir que não se perdem mensagens, e que a filiação é estável durante uma difusão, isto é, não existe concorrência entre difusões e alterações da filiação. Com estas hipóteses, o algoritmo, pela sua natureza, não introduz duplicados [3]. Se todos os nós forem equidistantes na rede, o algoritmo apresentado é especialmente adequado para distribuir a carga [11].

Numa rede real, as latências não são iguais e portanto podem-se escolher os nós de forma a privilegiar uma árvore que assegure uma difusão mais rápida da mensagem. Para este efeito, o ideal seria ter acesso à matriz dos custos de comunicação entre os nós presentes no sistema. Um sistema de coordenadas com uma função distância correlacionada com latência fornece essa matriz.

III. Sistemas de Coordenadas Sintéticas para a Internet

A localização de um nó na Internet consiste em fazer corresponder um endereço IP a um tuplo de coordenadas, que o localizam num referencial, concreto (e.g., geográfico) ou sintético. Os sistemas de localização que nos interessam aqui fornecem localizações sintéticas que permitem estimar a latência entre nós. Existem duas principais categorias de sistemas de coordenadas, nomeadamente os sistemas que se baseiam numa infra-estrutura de localização, os chamados *landmarks*, e as abordagens que dispensam infra-estrutura. Segue-se uma breve descrição de dois sistemas que se destacaram por realizarem as primeiras, ou mais promissoras, propostas em cada uma destas duas categorias.

O sistema GNP (Global Network Positioning) [12] é baseado num conjunto de nós fixos, designados por *landmarks*, que calculam as suas próprias coordenadas num espaço Euclidiano, com base na latência que os separa entre si (figura 1). Os restantes nós, calculam as suas próprias coordenadas com base nas coordenadas dos *landmarks* e na latência que os separa destes últimos.

Para cálculo das coordenadas é necessário minimizar o erro entre as latências medidas e as distâncias no espaço de coordenadas, através de um método de minimização global em

[3]O algoritmo apresentado em [11] lida com a perca de mensagens e a concorrência.

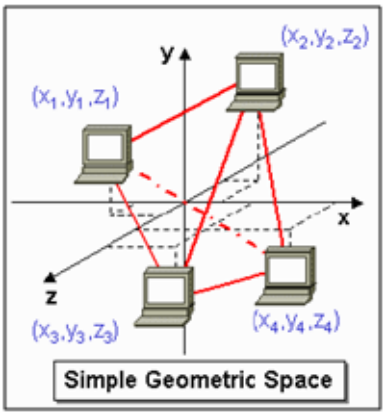


Figura 1: Esquema ilustrativo do espaço geométrico empregue pelo sistema GNP

múltiplas dimensões. O GNP usa o método Simplex Downhill. Foram propostos vários outros sistemas que usam também *landmarks* fixos ou escolhidos entre os pares [13], [15], [18], [2]. O GNP tem problemas de escalabilidade pois para cobrir adequadamente a Internet requer uma grande quantidade de *landmarks*. O sistema apresentado em [13] minimiza este problema usando uma hierarquia.

Ao contrário destes, o sistema Vivaldi [3] baseia-se num algoritmo iterativo, que sempre que dois nós comunicam entre si, procura que as coordenadas de ambos reflictam, cada vez com maior precisão, a latência que os separa. Quando um nó entra no sistema, escolhe coordenadas arbitrárias ou parte simplesmente da origem do referencial, e à medida que são trocadas mensagens com os outros nós, são medidas as latências entre os mesmos e trocadas as coordenadas. Estes dados simples permitem reajustar as coordenadas de modo a que no fim seja minimizado o erro global. O constante reposicionamento dos nós do sistema, pode ser comparado com uma estrutura de massas ligadas por molas (ver a figura 2) em que as massas (os nós) se posicionam no espaço até que todas as molas (as latências) fiquem em repouso. De modo a obter uma melhoria na precisão do método, os autores do introduziram a noção de altura nas coordenadas. A porção Euclidiana a duas dimensões modela o núcleo da Internet, com latências mais ou menos proporcionais à distância geográfica, enquanto a altura modela a latência que separa um nó do núcleo de rede.

Com o algoritmo Vivaldi, as coordenadas encontram-se naturalmente em constante actualização, o que faz com que o mesmo permita reagir a mudanças na rede. O tempo que decorre desde que um nó entra no sistema, até que este obtém coordenadas consideradas estáveis, é designado por tempo de aprendizagem ou de convergência.

Foi demonstrado em ambiente de simulação que este algoritmo atinge uma taxa de erro equivalente à do GNP, o que pode ser considerado um progresso visto que o sistema Vivaldi não depende de *landmarks*. No entanto, o sistema requer que sejam feitas medidas em quantidade razoável entre vários

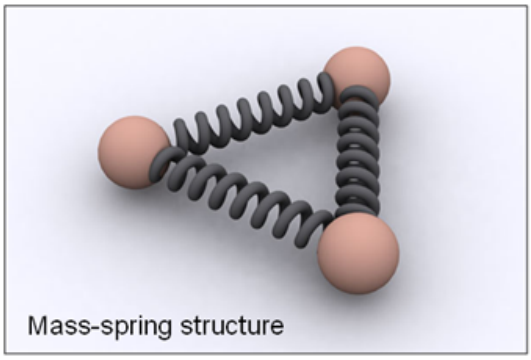


Figura 2: Analogia do sistema Vivaldi com uma estrutura de massas ligadas por molas

nós, incluindo nós próximos e alguns distantes escolhidos aleatoriamente [3]. Outro aspecto a não perder de vista tem a ver com o facto de que as coordenadas calculadas pelo Vivaldi após a entrada do nó no sistema têm um grande erro, ao contrário do GNP que as calcula de forma mais exacta logo no início do tempo de vida do nó.

Em ambos os sistemas apresentados a função distância é uma função da latência. No entanto, a característica variável e imprevisível da latência na rede global é o factor determinante para que modelar a rede com um modelo fixo de coordenadas, seja um problema difícil. O sistema Vivaldi, sendo adaptativo e dinâmico, tende a ser o que melhor poderia ajustar dinamicamente as coordenadas para reflectirem essa variação. No entanto, tal adaptação dinâmica tem-se revelado um grande desafio e continua a ser objecto de estudo [7], [19].

IV. Objecto do Estudo e Ferramentas Utilizadas

O objectivo deste estudo é comparar diversas versões do algoritmo de difusão descrito na secção II. Nomeadamente comparar a versão que escolhe aleatoriamente os descendentes de um nó, com versões que tomam em consideração a latência, entre o pai e os seus potenciais descendentes, calculada com base nos algoritmos GNP e Vivaldi. O critério de comparação usado é a média das latências com que uma mensagem difundida chega a cada um dos nós. Dado que os algoritmos que optimizam a latência não garantem uma distribuição de carga equitativa, o efeito sobre a distribuição de carga também foi estudado.

A. Ambiente de Experimentação

Este estudo foi realizado através de simulação que é um método de análise empírica relativamente eficaz (esforço versus resultados) mas também o que tem de ser utilizado com maiores precauções. Com efeito, os simuladores podem utilizar modelos que consideram muitos aspectos da rede e da pilha de protocolos dos nós (e.g., ns2 [14]), mas nesse caso não se conseguem estudar sistemas com mais do que poucas centenas de nós. O estudo de sistemas P2P com milhares de nós, como é o caso daquele em que estamos interessados, é geralmente baseado em modelos mais rudimentares.

O simulador usado foi desenvolvido especificamente no projecto LiveFeeds, é orientado aos eventos, e oferece um *framework* genérico que permite simular diversos tipos de nós, redes, algoritmos e sistemas. Mensagens podem ser enviadas de um nó para outro com uma latência calculada pelo simulador em função do modelo de rede utilizado[4].

O tipo de rede mais simples que foi usado é o que modela uma rede Euclidiana a duas dimensões. Neste caso, a rede está definida de forma que cada nó possui ligação directa a todos os outros nós da rede (*full mesh network*) e os nós encontram-se distribuídos aleatoriamente num plano. A latência entre os nós é proporcional à distância em linha recta no plano. Este tipo de modelo, relativamente elementar, apenas pode sustentar conclusões preliminares e de carácter qualitativo como veremos a seguir.

Em alternativa, foram também utilizadas duas topologias de rede geradas pelo gerador Orbis[10]. Neste caso são criados nós aplicação, que se ligam através de *local loops* a nós da rede constituída pelo grafo gerado pelo Orbis. A latência entre dois nós aplicação consiste na soma das latências dos respectivos *local loops*, adicionada às latências entre os nós da rede que fazem parte do melhor caminho entre a origem e o destino. Os grafos gerados pelas ferramentas Orbis modelam uma rede através de um modelo mais próximo da Internet real. No entanto, quão adequado é de facto este modelo, ultrapassa o âmbito deste trabalho dada a complexidade do problema envolvido[5].

Nas experiências realizadas não existe tráfego de *background* pois o único tráfego que se verifica é apenas o resultante das sucessivas difusões executadas com base nos algoritmos em estudo. Também não foi utilizada a modelação de filas de espera nos nós do núcleo da rede, mas o cálculo das latências extremo a extremo incluí também uma variação (*jitter*) das mesmas de forma aleatória, para de alguma forma modelar as variações que têm lugar na Internet.

B. Variantes do Algoritmo de Difusão Estudadas

O algoritmo de difusão aleatório descrito na secção II foi implementado no simulador. Para além desta estratégia de escolha dos nós descendentes na árvore de difusão, foram implementadas versões que escolhem os nós filhos usando um critério baseado na latência que separa os filhos do pai. Estas versões usam a latência calculada pelo algoritmo GNP, por duas variantes do algoritmo Vivaldi e ainda a latência conhecida pelo simulador (o oráculo perfeito).

A versão dos nós GNP foi realizada usando uma versão do algoritmo desenvolvida pela Universidade da Virginia. Foi também criado um nó para o algoritmo Vivaldi, utilizando uma simplificação da versão do algoritmo desenvolvida na Universidade de Harvard [7]. No início da simulação, os nós Vivaldi são colocados na origem do referencial.

[4]Mensagens de qualquer dimensão podem também ser transferidas por canais com semântica inspirada da do protocolo TCP mas esse método de comunicação não foi usado neste estudo.

[5]Este aspecto é abordado no documento []

C. Escala de Comparação dos Resultados

O algoritmo que optimiza a latência média com que uma mensagem chega a todos os nós para a qual é difundida é, neste contexto, a que minimiza sempre esta média. Determinar esta árvore é provavelmente um problema indecidível, mas uma possível heurística candidata a minimizar este tempo médio consiste em escolher em cada momento o nó mais próximo em cada sub-intervalo de nós. Este algoritmo, apesar de possivelmente fornecer uma boa aproximação do resultado óptimo, não o garante [9].

Da mesma forma, a heurística que consiste em escolher sempre o nó mais distante em cada intervalo, parece ser uma boa aproximação do pior algoritmo. Estes dois extremos permitem construir uma escala adequada para a comparação dos outros algoritmos. Diversas simulações, ao colocarem o algoritmo de escolha aleatória próximo do centro desta escala, confirmaram a sua viabilidade.

Assim, para além dos tipos de nós aleatório, GNP e Vivaldi, foi ainda criado um nó com acesso directo às latências da rede usada pelo simulador. Este nó pode funcionar em dois modos, com escolha do nó mais próximo, ou com escolha do nó mais distante ("Melhor" e "Pior" respectivamente).

V. Resultados e sua Discussão

Foram efectuados numerosos ensaios variando diversos parâmetros, nomeadamente o *fanout factor*, o número total de nós aplicação (LiveFeeds), o tipo e dimensão da rede *core* utilizada e os algoritmos. A maioria dos resultados obtidos está apresentada em [9] de forma detalhada. No que se segue, serão apenas apresentados os testes mais relevantes para as conclusões apresentadas. Os outros testes não modificaram essas conclusões nem introduziram outras.

A tabela I apresenta a média do tempo necessário para difundir 50000 mensagens através dos vários algoritmos e usando várias redes para simular a Internet. O nó emissor inicial é seleccionado aleatoriamente. O valor do *fanout factor* utilizado nestas experiências foi 3. A primeira coluna identifica o algoritmo. As restantes apresentam o tempo médio de chegada de cada mensagem a cada nó, em termos da escala introduzida em IV-C. O tempo total de simulação para obter os todos os resultados estudados foi de cerca de 200 horas usando um computador com um processador a 2.2 GHz e 2 GB de RAM.

No caso do algoritmo Vivaldi, utilizaram-se duas versões, com e sem noção de altura (ver a secção III). Como já foi referido, o algoritmo Vivaldi necessita de um tempo de aprendizagem. Na implementação realizada, a probabilidade de usar as coordenadas calculadas pelo algoritmo vai aumentando de 0 a 100%. Quando não são usadas as coordenadas, o nó descendente é escolhido aleatoriamente. O período de aprendizagem é realizado enviando primeiro 25000 mensagens extra. Na tabela Vivaldi (0 - 75000) indica que para cálculo da média foram usadas a totalidade das mensagens. Vivaldi (25000 - 75000) indica que para cálculo da média apenas foram usadas as mensagens enviadas depois do período de aprendizagem. A duração do período de aprendizagem foi ajustado através de diversos ensaios.

Algoritmo	Rede Euclidiana	Rede Orbis 50
Melhor	100	100
Pior	0	0
Aleatório	61,610	48,019
GNP	99,999	73,435
Vivaldi (0-75000)	89,319	72,380
Vivaldi (25000-75000)	95,233	78,217
Vivaldi+h (0-75000)	89,323	84,033
Vivaldi+h (25000-75000)	95,238	92,305

Tabela I: Tempo médio para recepção de 50000 mensagens numa rede com 10000 nós LiveFeeds em diversas configurações de rede. Resultados em % na escala Melhor (100%) a Pior (0%)

A rede Euclidiana é uma rede em que os 10000 nós aplicação estão distribuídos aleatoriamente num plano e com latência calculada pela geometria, o que dá uma distribuição das latências com alguma regularidade e explica o bom desempenho relativo do algoritmo aleatório ($\approx 62\%$), o desempenho quase perfeito do algoritmo GNP, e o razoável desempenho do algoritmo Vivaldi (no qual a noção de altura é irrelevante). O desempenho quase perfeito do algoritmo GNP é natural dadas as características geometricamente perfeitas da rede Euclidiana (triangulação perfeita e ausência de violações da desigualdade triangular).

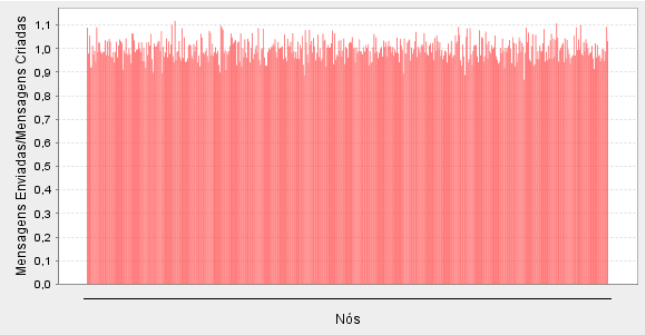
Nos ensaios com a rede Orbis foram utilizados 500 e 2000 nós internos (nós *core*) e 10000 nós aplicação a eles ligados. As redes resultantes são caracterizadas por uma distribuição menos uniforme das latências de comunicação entre nós, tal como na Internet real. No caso concreto, a latência entre nós do *core* é de 25 ms e as latências dos *local loops* distribuem-se por classes, entre os 10 e os 100 ms. Em resultado destes parâmetros, quanto menor o número de nós do *core*, maior é a variação das latências de comunicação entre nós da periferia (nós LiveFeeds) e mais significativo é o custo dos *local loops*.

Isso provavelmente explica porque razão o desempenho do algoritmo aleatório é pior na rede Orbis com 500 nós ($\approx 48\%$) do que na rede com 2000 nós ($\approx 58\%$) e sempre pior do que na rede Euclidiana (onde era $\approx 62\%$). Outra observação a reter é o desempenho do algoritmo GNP que piorou muito em relação ao obtido na rede Euclidiana.

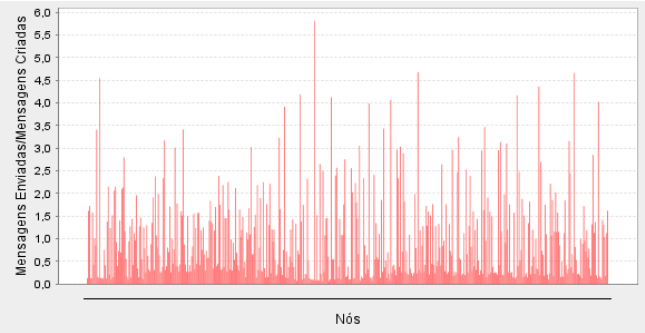
Os resultados apontam para uma superioridade do algoritmo Vivaldi, nomeadamente na sua variante Vivaldi+h, à medida que nos aproximamos de uma melhor aproximação da Internet. O óbice óbvio é o seu período de aprendizagem. Ao contrário, tudo indica que para melhorar a precisão do algoritmo GNP é necessário aumentar o número de *landmarks* disponíveis e a sua distribuição pela rede [13]. A menos do problema da aprendizagem, num sistema em que os nós comuniquem frequentemente uns com os outros, o algoritmo Vivaldi é mais

leve e mais preciso.

Em seguida são apresentados resultados relativos à forma como a escolha de nós pelos algoritmos afecta a carga nos mesmos. A figura 3 apresenta gráficos de barras com uma representação do número de mensagens enviadas por cada nó. De modo a facilitar a comparação dos gráficos, os valores apresentados são divididos pelo número total de mensagens difundidas. Assim, no eixo horizontal encontram-se os nós e no eixo vertical o número normalizado de mensagens enviadas por cada um.



(a)



(b)

Figura 3: Número de mensagens enviadas por nó: 500 nós Orbis, 500 nós LiveFeeds, *fanout factor* = 10. (a) algoritmo aleatório – máximo igual a 1,1 (b) Vivaldi+h – máximo igual a 6

Apenas são apresentados os gráficos relativos a ensaios com 500 nós LiveFeeds, com *fanout factor* = 10, na rede Orbis com 500 nós, para os algoritmos aleatório e Vivaldi+h. Estes podem ser considerados representativos visto que não existem diferenças significativas em relação a outros ensaios.

Numa distribuição de carga perfeita todos os nós recebem e enviam todas as mensagens difundidas no sistema e o resultado do quociente deve ser 1. É o que se passa de forma aproximada quando se usa o algoritmo aleatório. Ao contrário, usando o algoritmo Vivaldi+h, existe uma distribuição de carga muito desigual. Tal também acontece com os algoritmos GNP, Vivaldi (sem h) e Melhor. No caso do algoritmo Pior o desequilíbrio é ainda superior. Isto pode ser verificado através da tabela da figura II que apresenta um resumo dos resultados obtidos.

Estes resultados permitem apreciar noutra perspectiva o

Algoritmo	Tipo de Rede	N.º de nós	*Fanout Fac*
Aleatório	Euclidiana	500	3
Melhor	Euclidiana	500	3
Pior	Euclidiana	500	3
GNP	Euclidiana	500	3
Vivaldi	Euclidiana	500	3
Vivaldi+h	Euclidiana	500	3
Aleatório	Orbis (500 nós)	500	10
Melhor	Orbis (500 nós)	500	10
Pior	Orbis (500 nós)	500	10
GNP	Orbis (500 nós)	500	10
Vivaldi	Orbis (500 nós)	500	10
Vivaldi+h	Orbis (500 nós)	500	10

Tabela II: Desvio padrão da distribuição do número de mensagens enviadas por 500 nós em diversas situações e com diversos algoritmos

ganho do tempo de encaminhamento médio obtido com o conhecimento da latência que separa os nós. O algoritmo que é computacionalmente mais eficaz, o aleatório, está no meio da escala em termos de tempo de encaminhamento (um ganho de duas vezes entre o algoritmo aleatório e o melhor). Em contra partida, do ponto de vista da distribuição de carga, o desvio padrão da mesma cresce mais de 10 vezes entre o algoritmo aleatório e o algoritmo Vivaldi+h quando o *fanout factor* = 3, e mais de 20 vezes quando o *fanout factor* = 10.

Finalmente observou-se que todos os algoritmos têm melhor desempenho relativo quando o *fanout factor* é menor, aproximando-se do desempenho do algoritmo Melhor. Com efeito, um menor *fanout factor* aumenta a profundidade da árvore de difusão e a dimensão dos intervalos, criando assim maiores oportunidades de escolha de nós.

Em termos absolutos acontece precisamente o contrário, pois com um *fanout factor* superior é possível propagar as mensagens mais rapidamente na rede. Apesar das oportunidades de escolha serem menores, um grau crescente aproxima a difusão da solução óptima, a qual consiste em ter um grau igual ao número de receptores. Com efeito, neste caso limite, todas as mensagens são enviadas pelo caminho mais curto.

VI. Conclusões do Estudo e Trabalho Futuro

Este artigo apresenta um estudo comparativo sobre a utilização de sistemas de coordenadas de rede, calculadas com base nos algoritmos GNP e Vivaldi, para optimizar sistemas P2P do ponto de vista do tempo de encaminhamento.

O algoritmo GNP possui uma maior complexidade computacional que o algoritmo Vivaldi. O estudo demonstrou que essa complexidade não compensa pois o GNP não proporcionou uma maior precisão quando utilizado num cenário mais próximo da Internet real. O Vivaldi é mais leve, integra-se melhor com um sistema P2P e não requer a utilização de *landmarks*. Em contra-partida, o estudo também mostrou que a precisão do Vivaldi e o ganho que proporciona, estão

dependentes da duração relativa do período de aprendizagem e do rendimento obtido após o mesmo. Num sistema com um *churn* elevado, o impacto do Vivaldi será certamente menor.

Perante os dados obtidos, pode concluir-se que o algoritmo que maior optimização permite introduzir num sistema P2P em que a latência é relevante, os nós são estáveis e comunicam frequentemente entre si, é o algoritmo Vivaldi+h.

No entanto, é necessário não perder de vista o impacto dramático deste tipo de optimização na distribuição da carga. Esse aspecto contrasta com o comportamento mediano em termos de latência, mas insuperável em termos da simplicidade computacional e da distribuição de carga do algoritmo aleatório.

Para que o estudo permitisse retirar conclusões mais definitivas e em termos melhor quantificados, é importante introduzir modelos mais perfeitos do comportamento real da Internet [19], [7] pois a forma como esta é simulada tem um grande impacto nos resultados. Outro aspecto que merece estudo mais aprofundado é a forma como o algoritmo Vivaldi se comportaria quando confrontado com um modelo de *churn* dos nós. Para além disso, seria interessante analisar o impacto da utilização de coordenadas numa utilização mais racional dos canais de longa distância dos ISPs[1].

Apesar destas limitações, que são partilhados por muitos estudos realizados sobre a utilização de coordenadas em sistemas P2P, os resultados qualitativos apresentados são relevantes e dão indicações que permitem tomar opções em cenários específicos.

Numa outra linha de trabalho, será importante considerar a hipótese de acelerar o período de aprendizagem do algoritmo Vivaldi e melhorar a sua precisão e estabilidade, que tem sido objecto de desenvolvimentos recentes [7], [19]. Uma das hipóteses poderá consistir em tomar em consideração os mecanismos disponíveis actualmente para geo-localizar um nó. Sobretudo nas suas formas mais leves, que repousam na utilização de bases de dados aferidas por entidades que estudam a topologia da Internet (e.g., Caida) ou que da mesma têm informação privilegiada (e.g., Google).

Num sistema P2P é relativamente fácil obter e propagar um conjunto de dados sobre um nó (possíveis coordenadas geográficas, características de mobilidade ou não, capacidade dos *local loops* investidos no sistema, coordenadas sintéticas, sistema autónomo, etc.). Tais informações podem permitir a introdução de optimização em algoritmos P2P com base em múltiplos critérios.

REFERÊNCIAS

[1] R. Bindal, Pei Cao, W. Chan, J. Medved, G. Suwala, T. Bates, and A. Zhang. Improving traffic locality in bittorrent via biased neighbor selection. In *26th IEEE International Conference on Distributed Computing Systems, 2006. ICDCS 2006*, pages 66–77, July 2006.

[2] Manuel Costa, Miguel Castro, Antony Rowstron, and Peter Key. Pic: Practical internet coordinates for distance estimation. *icdcs*, 00:178–187, 2004.

[3] Frank Dabek, Russ Cox, Frans Kaashoek, and Robert Morris. Vivaldi: a decentralized network coordinate system. In *SIGCOMM '04: Proceedings of the 2004 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 15–26, New York, NY, USA, 2004. ACM.

[4] Frank Dabek, Jinyang Li, Emil Sit, James Robertson, M. Frans Kaashoek, and Robert Morris. Designing a dht for low latency and high throughput. In *NSDI'04: Proceedings of the 1st conference on Symposium on Networked Systems Design and Implementation*, pages 7–7, Berkeley, CA, USA, 2004. USENIX Association.

[5] Anjali Gupta, Barbara Liskov, and Rodrigo Rodrigues. One hop lookups for peer-to-peer overlays. In *Ninth Workshop on Hot Topics in Operating Systems (HotOS-IX)*, pages 7–12, Lihue, Hawaii, May 2003.

[6] John Jannotti, David K. Gifford, Kirk L. Johnson, M. Frans Kaashoek, and James W. O'Toole, Jr. Overcast: Reliable multicasting with an overlay network. In *The Fourth Symposium on Operating System Design and Implementation (OSDI)*, pages 197–212, 2000.

[7] Jonathan Ledlie, Peter Pietzuch, and Margo Seltzer. Stable and accurate network coordinates. In *ICDCS '06: Proceedings of the 26th IEEE International Conference on Distributed Computing Systems*, page 74, Washington, DC, USA, 2006. IEEE Computer Society.

[8] José Legatheaux Martins and Sérgio Duarte. Routing Algorithms for Content-based Publish/Subscribe Systems. *IEEE Communications Tutorials and Surveys – Accepted for Publication*, page 21, 2009.

[9] Rui Lopes. Sistemas de localização e de medida de distância na internet - contribuição para a avaliação da sua integração com algoritmos p2p. Tese de mestrado, Dep. Informática, FCT, Universidade Nova de Lisboa, Fevereiro 2009.

[10] Priya Mahadevan, Calvin Hubble, Dmitri Krioukov, Bradley Huffaker, and Amin Vahdat. Orbis: rescaling degree correlations to generate annotated internet topologies. *SIGCOMM Comput. Commun. Rev.*, 37(4):325–336, 2007.

[11] Simão Mata, José Legatheaux Martins, and Sérgio Duarte. Análise do custo e viabilidade de um sistema p2p com visibilidade completa. *Proposed for Publication*, page 12, 2009.

[12] T. S. Eugene Ng and Hui Zhang. Predicting internet network distance with coordinates-based approaches. In *INFOCOM*, 2002.

[13] T. S. Eugene Ng and Hui Zhang. A network positioning system for the internet. In *ATEC'04: Proceedings of the USENIX Annual Technical Conference 2004*, pages 11–11, Berkeley, CA, USA, 2004. USENIX Association.

[14] NS-2. The network simulator ns-2, 2008.

[15] M. Pias, J. Crowcroft, S. Wilbur, S. Bhatti, and T. Harris. Lighthouses for scalable distributed location. In *M. Pias, J. Crowcroft, S. Wilbur, S. Bhatti, and T. Harris. Lighthouses for scalable distributed location. In Second International Workshop on Peer-to-Peer Systems (IPTPS '03), Feb 2003.*, 2003.

[16] J.A. Pouwelse, P. Garbacki, D.H.J. Epema, and H.J. Sips. The Bittorrent P2P File-sharing System: Measurements and Analysis. *International Workshop on Peer-to-Peer Systems (IPTPS)*, 2005.

[17] Antony Rowstron and Peter Druschel. Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems. In *IFIP/ACM International Conference on Distributed Systems Platforms (Middleware)*, pages 329–350, November 2001.

[18] Liying Tang and Mark Crovella. Virtual landmarks for the internet. In *IMC '03: Proceedings of the 3rd ACM SIGCOMM conference on Internet measurement*, pages 143–152, New York, NY, USA, 2003. ACM.

[19] Guohui Wang and T.S. Eugene Ng. Distributed algorithms for stable and secure network coordinates. In *IMC '08: Proceedings of the 8th ACM SIGCOMM conference on Internet measurement*, pages 131–144, New York, NY, USA, 2008. ACM.