Prototipação de um *cache* DNS Pró-Ativo usando *Publish/Subscribe* em Redes P2P

Tiago Rigo Guasti[1], **Rodolfo da Silva Villaça**[1], **Hélcio Bezerra de Mello**[1]

[1] Departamento de Engenharias e Computação (DECOM)
Centro Universitário Norte do Espírito Santo (CEUNES)
Universidade Federal do Espírito Santo (UFES)

tiaomail@gmail.com, rodolfo.villaca@ufes.br, helciomello@ceunes.ufes.br

Resumo. *DNS (*Domain Name System*) é um serviço fundamental na Internet. Contudo, a necessidade de se adotarem valores baixos para o tempo de vida de seus registros compromete a eficiência de seu mecanismo de* cache*. Este trabalho propõe um mecanismo alternativo inovador, baseado na disseminação das alterações nos registros aos servidores que possuirem cópia deles em seu* cache*. A arquitetura utilizada emprega o mecanismo* Publish/Subscribe *em uma rede P2P (*peer-to-peer*) a fim de conseguir eficiência e robustez. A solução proposta é então avaliada a partir de um protótipo e seus resultados discutidos.*

Abstract. *DNS (*Domain Name System*) is a service of critical importance in the Internet. However, the efficiency of its caching mechanism is being impacted by the need of increasingly shorter times of live for its resource records. This paper proposes an innovative alternate approach, which disseminates changed resource records to the servers that have a copy of those records in their cache. The architecture presented here employs the Publish/Subscribe paradigm in a P2P (peer-to-peer) network in order to achieve the desired level of efficiency and robustness. A prototype of our solution has been implemented for evaluation purposes, and the resulting conclusions are discussed.*

1. Introdução

A Internet está onipresente na vida das pessoas hoje. Todos os dias, milhões de usuários enviam e recebem e-mail, fazem *downloads* e empregam a *web* para as mais variadas finalidades, como ler notícias, compras *on line*, etc. Cada uma dessas atividades, por sua vez, depende de um outro serviço, transparente para os usuários, que é o DNS (*Domain Name System*).

A principal finalidade do DNS é a de resolver nomes, a exemplo de www.sbc.org.br, em endereços IP, como 143.54.31.18. Esse serviço emprega um mecanismo de *cache* para melhorar seu desempenho, mas depende de valores relativamente altos para os tempos de vida dos registros (TTL — *time to live*) para atingir uma boa eficiência. Por outro lado, a necessidade de se oferecer conteúdo dinâmico e personalizado aos usuários de serviços como a *web* requer que o DNS trabalhe com valores reduzidos de TTL em seus registros, dessa forma sacrificando o desempenho do *cache*.

Como solução para esse problema, este trabalho propõe o uso de uma nova arquitetura de *cache*, que difere daquela utilizada pelo DNS por ser *pró-ativa*. Assim sendo,

em vez de utilizar valores de tempo de vida para os registros DNS, nossa abordagem dissemina eventuais alterações feitas em um registro para todos os servidores em que houver uma cópia dele armazenada em seu *cache*. Trata-se de uma aplicação típica do paradigma *Publish/Subscribe*, que oferece a vantagem de desacoplar a comunicação entre o servidor DNS de onde a alteração do registro se originou e os servidores que precisam ser notificados desse evento. A comunicação entre esses servidores é feita por meio de uma rede P2P, devido à sua eficiência e robustez [Pappas et al. 2006].

O restante deste artigo está organizado da seguinte maneira: a Seção 2 resume o funcionamento do DNS e do paradigma *Publish/Subscribe*, ao passo que a Seção 3 discute a arquitetura e a implementação do protótipo utilizado para a avaliação de nossa proposta. A metodologia e os resultados dessa avaliação são apresentados na Seção 4, seguida da análise de trabalhos relacionados na Seção 5. Finalmente, a Seção 6 conclui este artigo, apontando possíveis direções de trabalhos futuros.

2. Conceitos Básicos

Conforme mencionado na Seção 1, a resolução dos nomes das estações em seus endereços é feita pelo DNS, que armazena essas informações na forma de registros. Contudo, para que o serviço seja escalável, seu espaço de nomes foi dividido em domínios e subdomínios (por exemplo, *sbc.org.br*), de modo que cada servidor DNS seja responsável apenas pelos registros que se encontrem em seu domínio. Logo, se um servidor recebe uma consulta sobre um nome que esteja em outro domínio, torna-se necessário encaminhá-la ao servidor responsável por ele.

Registros oriundos de outros domínios são armazenados em *cache* pelos servidores DNS, de modo que eles possam resolver de imediato qualquer nome já consultado anteriormente, sem precisar recorrer a nenhum outro servidor. Apesar de essa prática acelerar bastante a resolução de nomes, é preciso impedir que registros fiquem armazenados em *cache* indefinidamente. Do contrário, quando um registro fosse alterado, as cópias em *cache* mantidas em outros servidores possuiriam a versão antiga desse registro por tempo indeterminado. Por esse motivo, um registro só pode ser mantido em *cache* por um período igual a seu tempo de vida (TTL), que, conforme já mencionado, não pode ser muito elevado. Por outro lado, valores baixos de TTL diminuem a eficiência do *cache*, pois nesse caso os registros frequentemente já terão sido retirados do *cache* (em virtude de seus TTLs terem expirado) antes de serem utilizados na resolução de algum nome.

Consequentemente, seria desejável que cada servidor armazenasse registros em *cache* indefinidamente, e que essas cópias fossem atualizadas sempre que os registros fossem modificados nos servidores de onde foram obtidos. Nesse contexto, definem-se claramente dois papeis: os servidores onde os registros são alterados e aqueles que possuem cópia desses registros em *cache*. Esses papeis correspondem, respectivamente, a *produtores* e *consumidores* no paradigma *publish/subscribe*, pois estes geram as notificações de que um registro foi alterado enquanto aqueles são os interessados em recebê-las.

Existe, ainda, um terceiro papel, os *intermediários* (*brokers*), responsáveis por receber as notificações dos produtores e encaminhá-las aos consumidores. Dessa forma, produtores não precisam manter a informação de quem são os consumidores, e vice-versa, resultando no desacoplamento da comunicação entre esses papeis. Os intermediários podem, ainda, ser organizados de forma centralizada (funcionando em um único compu-

tador) ou distribuída (em que vários servidores estão interconectados para se conseguir balanceamento de carga), sendo esta a mais adequada em ambientes de larga escala como a Internet [Cugola and Jacobsen 2002, Eugster et al. 2003].

3. Arquitetura e Implementação

Com o objetivo de testar a viabilidade de integração da arquitetura *Publish/Subscribe* com o *cache* de um servidor DNS foi implementado um protótipo usando a linguagem de programação Java, e as implementações livres Freepastry [Freepastry 2013] e Scribe [Scribe 2013].

A implementação realizada compõe-se de 3 partes principais: 1) o *Broker* distribuído, implementado sobre uma rede P2P Pastry [Rowstron and Druschel 2001]; 2) a interface de atualização de publicações; e 3) o DNSHandler, um *proxy* DNS capaz de inscrever-se em publicações de nomes de seu interesse e manter um *cache* atualizado destes nomes. A Figura 1 mostra os nós clientes (*hosts*) atualizando os seus nomes no *Broker* P2P e usuários fazendo consultas de nomes que são interceptadas pelo DNSHandler e encaminhadas ao *Broker* para resolução pró-ativa.

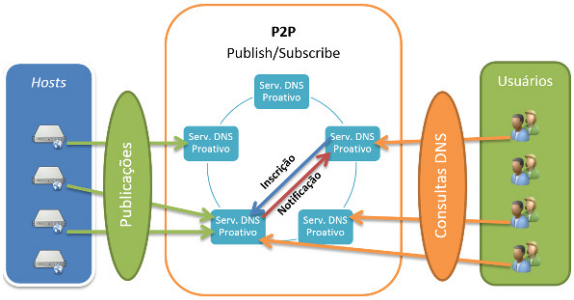

Figura 1. Proposta de implementação da solução Publish/Subscribe + Cache DNS.

3.1. DNSHandler

Pode-se dizer que a implementação iniciou-se com o DNSHandler, cujo protótipo é uma extensão do trabalho do grupo publicado em [Costa and Villaça 2010]. Essa primeira implementação encaminhava as consultas de resolução de nomes a uma DHT (*Distributed Hash Table*) sem nenhuma relação com o paradigma Publish/Subscribe. O DNSHandler consiste em um *proxy* DNS, instalado na porta 53/UDP do servidor DNS, capaz de interceptar todas as consultas que chegam até ele. Extrai-se o nome dessa consulta e verifica-se se o nome está em *cache* no DNSHandler. Se estiver, encaminha-se a resposta ao cliente diretamente. Caso contrário, faz-se o processo de resolução normal, usando a rede DNS na Internet e, ao receber a resposta da resolução do nome, faz-se o *cache* do mesmo localmente no DNSHandler e inscreve-se no Broker para receber atualizações desse nome.

Para manter o *cache* do DNSHandler atualizado, o mesmo possui uma função para inscrição (*subscribe*) no *Broker* em nomes de interesse. Na implementação realizada neste protótipo, por simplificação, o DNSHandler inscreve-se em todos os nomes que são adicionados em seu *cache*. Entretanto, com o objetivo de reduzir o tráfego e a carga na rede do *Broker*, pode-se fazer uma inscrição seletiva, somente para os nomes mais populares e dinâmicos.

3.2. Broker

Optou-se por usar um Broker distribuído baseado nas implementações do Freepastry e do Scribe. O Freepastry é uma implementação em Java da DHT Pastry, que fornece as primitivas `route(msg, key)` e `deliver(msg,key)`, dentre outras menos importantes para a compreensão desta implementação. A primeira entrega uma mensagem (`msg`) ao nó da rede Pastry responsável pelo armazenamento da chave (`key`). A segunda entrega a mensagem para a aplicação Scribe, que funciona sobre a rede Pastry. O Scribe é uma implementação de um serviço *Publish/Subscribe* cujas principais primitivas são `create(topicId)`, `subscribe(topicId)`, `publish(topicId, event)`. A primeira é usada na criação de um tópico, a segunda para inscrição no mesmo e a terceira para publicação de eventos relacionados ao tópico. Na rede Pastry o `topicId` corresponde à chave (`key`) de roteamento na DHT.

Neste trabalho a chave de roteamento da DHT (`key`) e o tópico (`topicId`) no qual os consumidores se inscreverão correspondem ao *hash* dos nomes DNS. Os produtores publicarão as atualizações sob a forma de tópicos que também correspondem ao *hash* dos nomes DNS. As publicações contêm o nome e os endereços IP associados ao mesmo.

3.3. Interface de atualização

Por fim, a solução disponibiliza aos servidores de aplicações com nomes dinâmicos uma interface para publicações das atualizações desses nomes. Essa interface é necessária pois os servidores de aplicação que possuem nomes dinâmicos não fazem parte da rede do *Broker* e, portanto, não possuem acesso à primitiva `publish()`. Essa interface de atualização dos tópicos (nomes) é implementada no DNSHandler, que faz parte da rede do Broker e é acessível pelos servidores de aplicação.

4. Avaliação

Para testar a implementação foram elaborados quatro cenários de teste:

- Cenário direto: as consultas foram realizadas sem utilizar a implementação, com o objetivo de obtermos valores de referência de tempo de resposta para as consultas;
- Cenário sem *cache* DNSHandler: as consultas foram realizadas usando o DNSHandler, mas tiveram que ser resolvidas pela rede de DNS da Internet, uma vez que elas não estavam em *cache*. Nas consultas deste cenário o *cache* era sempre esvaziado ou usava-se nomes inéditos nas consultas;
- Cenário com *cache* DNSHandler: as consultas foram resolvidas pelo DNSHandler com dados contidos em seu *cache*. Neste cenário os nomes eram repetidos do cenário anterior;
- Cenário com *cache Publish/Subscribe*: neste cenário, os nomes em *cache* do cenário anterior foram atualizados usando a Interface de Atualização. As consultas foram resolvidas pelo cache do DNSHandler, atualizado pela a publicação das atualizações.

A ferramenta DIG (*Domain Information Groper*) foi utilizada para a medição do tempo de resolução e verficação do endereço retornado nas consultas. Foram utilizados os nomes mais visitados no Brasil, medidos por um estudo realizado pelo site *www.alexa.com* em Janeiro/2013. A seguir, na Tabela 1 são apresentados alguns resultados para os 10 primeiros nomes dessa lista. É importante reforçar que esses resultados

só têm valor qualitativo, para mostrar que os nomes foram realmente resolvidos via *cache* após a atualização. Os valores não possuem significado estatístico válido, uma vez que foram obtidos em laboratório e estão sujeitos às condições da rede local e acesso à Internet da universidade durante o período de avaliação.

Tabela 1. Tempos médios de consulta nos 5 cenários, em milissegundos (ms).

Nome	**Direto**	**DNS**	***Cache***	***Publish/Subscribe***
www.facebook.com	26	29	3	2
www.google.com.br	26	30	3	3
www.google.com	26	31	4	3
www.youtube.com	26	32	3	4
www.uol.com.br	29	40	2	2
www.live.com	26	29	3	4
www.globo.com	26	29	3	2
www.blogspot.com.br	29	33	2	3
www.yahoo.com.br	26	31	3	3
www.mercadolivre.com.br	26	30	2	4
Média	26,6	31,3	2,8	3
Desvio	1,3	3,4	0,6	0,8

As pequenas diferenças notadas entre os cenários Direto e sem *cache* DNSHandler se dá pelo fato de haver um processamento extra, nesse segundo cenário, para consulta ao *cache* do DNSHandler antes de recorrer ao serviço DNS tradicional, além de sub-rotinas necessárias para registrar o interesse pelo nome (*Subscribe*) no Broker.

Ao comparar os cenários com *cache* DNSHandler e *cache Publish/Subscribe*, podemos notar resultados bastante próximos. Isso pode parecer estranho no primeiro momento, mas o ganho da solução acontece no instante em que um registro em *cache* expira ou torna-se inválido; nesse momento a solução proposta continuará respondendo com *cache Publish/Subscribe* devido à pró-atividade e atualização do DNSHandler, enquanto que na solução tradicional o DNS ou resolveria o nome desatualizado, ou demoraria o tempo do cenário sem *cache* pois o registro encontrar-se-ia expirado.

5. Trabalhos Relacionados

Os resultados apresentados em [Jung et al. 2001] mostram que o *cache* é peça fundamental no desempenho do DNS, e que a redução do TTL diminui os *cache hits* dos servidores, afetando o desempenho do serviço. CoDNS [Park et al. 2004] apresenta uma proposta de *cache* cooperativo de servidores DNS, implementado sob uma rede P2P. A ideia é compartilhar as entradas de *cache* de todos os servidores que fazem parte da rede. Ao contrário da nossa proposta, não há nenhuma implementação de *cache* pró-ativo. Proposta semelhante ao CoDNS é apresentada em [Poolsappasit and Ray 2007].

A ideia de um *cache* DNS pró-ativo é também apresentada em [Cohen and Kaplan]. Entretanto, a abortagem usada nesse trabalho é fazer uma pré-resolução de nomes mais populares de tal forma a mantê-los atualizados, antecipando-se às possíveis falhas de *cache*. Essa abordagem é diferente da apresentada neste trabalho, que usa o paradigma de comunicação *Publish/Subscribe* para manutenção das entradas do *cache* DNS.

6. Conclusões e Trabalhos Futuros

O mecanismo de *cache* possui forte influência no desempenho do DNS. A arquitetura próativa baseada no paradigma *publish/subscribe* proposta neste trabalho mostrou-se uma alternativa funcional comparada à abordagem tradicional, que era suscetível tanto ao armazenamento de registros desatualizados quanto à demora em obtê-los dos servidores externos. Além disso, a solução proposta é transparente ao usuário.

Contudo, tendo em vista tratar-se de uma investigação preliminar sobre o assunto, os experimentos foram conduzidos em escala reduzida e a implementação do protótipo sofreu algumas simplificações. Existem, portanto, várias possibilidades de trabalhos futuros, tais como restringir a aplicação da arquitetura proposta aos registros mais acessados nos servidores, a fim de se limitar a quantidade de registros atualizados pela rede e diminuir a sobrecarga no *broker*, e avaliação de desempenho de rede da solução proposta.

Referências

Cohen, E. and Kaplan, H. Proactive caching of DNS records: addressing a performance bottleneck. In *Applications and the Internet, 2001. Proceedings. 2001 Symposium on*, pages 85–94.

Costa, M. V. and Villaça, R. S. (2010). Serviço de Resolução de Nomes usando DHTs para uso na Internet de Nova Geração. In *Anais da 8ª Escola Regional de Redes de Computadores (ERRC)*, ERRC '10, Alegrete/RS.

Cugola, G. and Jacobsen, H.-A. (2002). Using publish/subscribe middleware for mobile systems. *SIGMOBILE Mob. Comput. Commun. Rev.*, 6(4):25–33.

Eugster, P. T., Felber, P. A., Guerraoui, R., and Kermarrec, A.-M. (2003). The many faces of publish/subscribe. *ACM Comput. Surv.*, 35(2):114–131.

Freepastry (2013). Freepastry. `http://www.freepastry.org/FreePastry/`. acessado: 5/3/2013.

Jung, J., Sit, E., Balakrishnan, H., and Morris, R. (2001). DNS Performance and the Effectiveness of Caching. In *1st ACM SIGCOMM Internet Measurement Workshop*, San Francisco, USA.

Pappas, V., Massey, D., Terzis, A., and Zhang, L. (2006). A Comparative Study of the DNS Design with DHT-Based Alternatives. In *INFOCOM 2006*, Barcelona, Spain.

Park, K., Pai, V. S., Peterson, L., and Wang, Z. (2004). CoDNS: improving DNS performance and reliability via cooperative lookups. In *Proc. of the 6th Symp. on Operating Systems Design & Implementation*, OSDI'04, pages 14–14, Berkeley, CA, USA.

Poolsappasit, N. and Ray, I. (2007). Enhancing Internet Domain Name System Availability by Building Rings of Cooperation Among Cache Resolvers. In *8th Annual IEEE SMC Information Assurance Workshop (IAW 2007)*, New York, USA.

Rowstron, A. I. T. and Druschel, P. (2001). Pastry: Scalable, Decentralized Object Location, and Routing for Large-Scale Peer-to-Peer Systems. In *Proc. of the IFIP/ACM Int. Conf. on Distributed Systems Platforms*, Middleware '01, pages 329–350, London, UK, UK.

Scribe (2013). Scribe, a scalable group communication system. `http://www.freepastry.org/SCRIBE/default.htm`. acessado: 5/3/2013.