Desenvolvimento de técnicas de anycast na camada
de aplicação para a provisão de qualidade de
serviço em computação na nuvem

Lucas Junqueira Adami

SERVIÇO DE PÓS-GRADUAÇÃO DO ICMC-USP

Data de Depósito:

Assinatura: _____

Lucas Junqueira Adami

Desenvolvimento de técnicas de anycast na camada de aplicação para a provisão de qualidade de serviço em computação na nuvem

Dissertação apresentada ao Instituto de Ciências Matemáticas e de Computação – ICMC-USP, como parte dos requisitos para obtenção do título de Mestre em Ciências – Ciências de Computação e Matemática Computacional. *VERSÃO REVISADA*

Área de Concentração: Ciências de Computação e Matemática Computacional

Orientador: Prof. Dr. Julio Cezar Estrella

USP – São Carlos
Outubro de 2015

Ficha catalográfica elaborada pela Biblioteca Prof. Achille Bassi
e Seção Técnica de Informática, ICMC/USP,
com os dados fornecidos pelo(a) autor(a)

A198d Adami, Lucas Junqueira
Desenvolvimento de técnicas de anycast na camada
de aplicação para a provisão de qualidade de serviço
em computação na nuvem / Lucas Junqueira Adami;
orientador Julio Cezar Estrella. - São Carlos -
SP, 2015.
86 p.

Dissertação (Mestrado - Programa de Pós-Graduação
em Ciências de Computação e Matemática Computacional)
- Instituto de Ciências Matemáticas e de Computação,
Universidade de São Paulo, 2015.

1. *anycast*. 2. seleção de serviços. 3. sistemas
distribuídos. I. Estrella, Julio Cezar, orient. II.
Título.

Lucas Junqueira Adami

**Development of application layer anycast techniques for
quality of service provision in cloud computing**

Master dissertation submitted to the Instituto de
Ciências Matemáticas e de Computação – ICMC-
USP, in partial fulfillment of the requirements for the
degree of the Master Program in Computer Science
and Computational Mathematics. *FINAL VERSION*

Concentration Area: Computer Science and
Computational Mathematics

Advisor: Prof. Dr. Julio Cezar Estrella

USP – São Carlos
October 2015

AGRADECIMENTOS

Agradeço à todo o pessoal que me ajudou de forma direta ou indireta no desenvolvimento deste projeto de graduação. Agradeço ao Laboratório de Sistemas Distribuídos e Programação Concorrente (LaSDPC) e à Universidade de Leicester por fornecer toda a infraestrutura necessária para a execução do trabalho. Agradeço em especial ao meu orientador, Júlio Cezar Estrella, por ter dado essa oportunidade de me envolver com o projeto de mestrado. Agradeço também meu orientador no exterior, Stephan Reiff-Marganiec, por ter me dado a oportunidade de desenvolver este trabalho por 4 meses na Inglaterra.

Agradeço também à FAPESP (Fundação de Amparo à Pesquisa do Estado de São Paulo) pelo suporte desta pesquisa (Processos 2014/10771-7 e 2013/05635-4), cujo apoio financeiro e programas oferecidos permitiram conduzir este trabalho de mestrado durante dois anos.

RESUMO

ADAMI, L. J.. **Desenvolvimento de técnicas de anycast na camada de aplicação para a provisão de qualidade de serviço em computação na nuvem**. 2015. 86 f. Dissertação (Mestrado em Ciências – Ciências de Computação e Matemática Computacional) – Instituto de Ciências Matemáticas e de Computação (ICMC/USP), São Carlos – SP.

Nos últimos anos, houve um aumento da complexidade e variedade de serviços disponíveis na Internet, fato que tem levado à busca por técnicas eficientes de roteamento de requisições de um cliente ao melhor servidor disponível, sendo uma delas conhecida como *application layer anycast* (ALA). O objetivo deste mestrado é elaborar meios eficientes de prover *anycast* na camada de aplicação com qualidade de serviço no contexto de computação em nuvem. Para atingir esse objetivo, um novo sistema foi proposto (GALA, *Global Application Layer Anycast*). Ele herda características de um outro sistema existente e emprega a geolocalização como diferencial, a fim de melhorar o desempenho geral do algoritmo. Experimentos foram realizados por meio de simulação e os resultados mostraram que esse novo sistema, comparado ao algoritmo herdado, mantém a eficiência das requisições realizadas pelos clientes e diminui consideravelmente o tempo de latência dessas operações. Ainda, o sistema proposto foi desenvolvido em um ambiente real a fim de fortalecer os resultados das simulações. Com os resultados obtidos, o sistema modelado foi validado e sua eficácia confirmada.

Palavras-chave: *anycast*, seleção de serviços, sistemas distribuídos.

ABSTRACT

ADAMI, L. J.. **Desenvolvimento de técnicas de anycast na camada de aplicação para a provisão de qualidade de serviço em computação na nuvem**. 2015. 86 f. Dissertação (Mestrado em Ciências – Ciências de Computação e Matemática Computacional) – Instituto de Ciências Matemáticas e de Computação (ICMC/USP), São Carlos – SP.

In the past years, the complexity and variety of available services expanded in the Internet, fact that is drawing attention of many researchers that wish to find out efficient techniques of routing client requests to the closest server, being one of them known as application layer anycast (ALA). Thus, the objective of this research is to elaborate ways to offer application layer anycast that are scalable and select the closest servers with the shortest latency possible, in the context of cloud computing. To achieve this goal, a new system was proposed (GALA, Global Application Layer Anycast). It inherits features from an existing system and applies geolocation to improve its overall performance. Simulation results indicated that the new system, compared to its antecessor, has the same efficiency but decreases considerably the requests latency. Yet, the proposed system was deployed in a real environment to strengthen the simulations results. With the obtained data, the modeled system was validated and its efficiency confirmed.

Key-words: anycast, service selection, distributed systems.

LISTA DE ILUSTRAÇÕES

Figura 1 – Exemplo de <i>anycast</i> , adaptado de Piper (2013).	18
Figura 2 – Esquema <i>anycast</i> na camada de aplicação, adaptado de Zegura <i>et al.</i> (2000).	19
Figura 3 – Perda de pacotes em um sistema com e sem <i>anycast</i> , adaptado de Chellouche <i>et al.</i> (2011).	20
Figura 4 – Arquitetura do OASIS, adaptado de Freedman, Lakshminarayanan e Mazières (2006).	26
Figura 5 – Arquitetura do Proxima, adaptado de Wang <i>et al.</i> (2009).	27
Figura 6 – Representação do sistema DOAT, adaptado de Mykoniati <i>et al.</i> (2009).	28
Figura 7 – Arquitetura do sistema AICN, adaptado de Yuan <i>et al.</i> (2012).	29
Figura 8 – Vizinhança de um nó no GAA, adaptado de Ma, Zhou e Zhang (2009).	30
Figura 9 – Visão geral do sistema GALA, adaptado de Adami <i>et al.</i> (2015b).	34
Figura 10 – Correlação entre RTT e distância geográfica, adaptado de Freedman, Lakshminarayanan e Mazières (2006).	36
Figura 11 – Mapeamento no resolvedor GALA, adaptado de Adami <i>et al.</i> (2015).	38
Figura 12 – Exemplo de mapeamento em 36 regiões geográficas (JONES, 1998).	40
Figura 13 – Exemplo de mapeamento de servidores GALA no DNS do sistema.	41
Figura 14 – Vizinhança do sistema GALA.	42
Figura 15 – Fluxograma do algoritmo de busca.	45
Figura 16 – Diagrama de classes simplificado do simulador GALASim.	49
Figura 17 – Diagrama do banco de dados do GALASim.	50
Figura 18 – Entrada com servidores GeoNames como clientes e nós PlanetLab como servidores (ADAMI <i>et al.</i> , 2014).	51
Figura 19 – Entrada com servidores PlanetLab como clientes e nós GeoNames como servidores (ADAMI <i>et al.</i> , 2015b).	52
Figura 20 – Eficiência dos níveis de mapeamento, adaptada de Adami <i>et al.</i> (2015b).	53
Figura 21 – Latência dos níveis de mapeamento, adaptada de Adami <i>et al.</i> (2015b).	53
Figura 22 – Saltos dos níveis de mapeamento, adaptada de Adami <i>et al.</i> (2015b).	54
Figura 23 – Eficiência: Gráficos de barras, adaptada de Adami <i>et al.</i> (2014).	55
Figura 24 – Eficiência: Gráficos de efeitos, adaptada de Adami <i>et al.</i> (2014).	56
Figura 25 – Eficiência: Gráficos de efeitos principais.	56
Figura 26 – Eficiência: Gráficos de interações.	57
Figura 27 – Latência: Gráficos de barras, adaptada de Adami <i>et al.</i> (2014).	57
Figura 28 – Latência: Gráficos de efeitos, adaptada de Adami <i>et al.</i> (2014).	58

Figura 29 – Latência: Gráficos de efeitos principais.	58
Figura 30 – Latência: Gráficos de interações.	59
Figura 31 – Saltos: Gráficos de barras, adaptada de Adami <i>et al.</i> (2014).	60
Figura 32 – Saltos: Gráficos de efeitos, adaptada de Adami <i>et al.</i> (2014).	60
Figura 33 – Saltos: Gráficos de efeitos.	61
Figura 34 – Saltos: Gráficos de interações.	61
Figura 35 – Gráfico de intervalos do RTT, para a primeira configuração de entrada, adaptada de Adami <i>et al.</i> (2015b).	63
Figura 36 – Gráfico de intervalos do RTT, para a segunda configuração de entrada, adaptada de Adami <i>et al.</i> (2015b).	63
Figura 37 – Histograma da latência, para a primeira configuração de entrada, adaptada de Adami <i>et al.</i> (2015).	64
Figura 38 – Histograma da latência, para a segunda configuração de entrada, adaptada de Adami <i>et al.</i> (2015).	64
Figura 39 – Histograma acumulado da latência, adaptada de Adami <i>et al.</i> (2015).	65
Figura 40 – Visão geral do sistema GALA real, adaptada de Adami <i>et al.</i> (2015b).	70
Figura 41 – Resultados para os tempos de busca e serviço, adaptada de Adami <i>et al.</i> (2015b).	74

LISTA DE TABELAS

Tabela 1 – Formato do registro de faixas de endereço IP.	38
Tabela 2 – Formato do registro de coordenadas dos países.	39
Tabela 3 – Parâmetros das Simulações.	51
Tabela 4 – Fatores do Primeiro Experimento.	52
Tabela 5 – Configurações do experimento.	55
Tabela 6 – Latências mínimas e máximas dos experimentos.	59
Tabela 7 – Fatores do terceiro experimento.	62
Tabela 8 – Tempos de seleção máxima.	66
Tabela 9 – Mapeamento dos servidores	72
Tabela 10 – Configurações de experimento.	73
Tabela 11 – Parâmetros do experimento.	74
Tabela 12 – Organização das atividades realizadas.	81

SUMÁRIO

1	INTRODUÇÃO	17
1.1	Contextualização	17
1.2	Motivação	19
1.3	Objetivos	20
1.4	Organização do Trabalho	21
2	TRABALHOS RELACIONADOS	23
2.1	Considerações Iniciais	23
2.2	Qualidade de Serviço	23
2.3	Propostas Semelhantes	24
2.4	Sistemas de <i>Anycast</i> na Camada de Aplicação	25
2.5	Considerações Finais	31
3	DESENVOLVIMENTO DO TRABALHO	33
3.1	Considerações Iniciais	33
3.2	O Sistema GALA	33
3.3	Início da Busca <i>Anycast</i>	35
3.4	Geolocalização de Endereços IP	35
3.5	Mapeamento da Pergunta <i>Anycast</i>	37
3.6	Resolução da Pergunta <i>Anycast</i>	39
3.7	Vizinhança dos Servidores GALA	40
3.8	Busca da Réplica mais Próxima	43
3.9	Considerações Finais	44
4	AVALIAÇÃO EXPERIMENTAL POR SIMULAÇÕES	47
4.1	Considerações Iniciais	47
4.2	O Simulador GALASim	48
4.3	Configurações dos Experimentos Simulados	50
4.4	O Primeiro Experimento	52
4.5	O Segundo Experimento	54
4.6	O Terceiro Experimento	62
4.7	Tempo Máximo de Seleção	65
4.8	Considerações Finais	67

5	PROTÓTIPO DO SISTEMA	69
5.1	Considerações Iniciais	69
5.2	Visão Geral do Sistema	69
5.3	Balanceamento de Carga	70
5.4	Detalhes de Implementação	71
5.5	Detalhes do Ambiente	72
5.6	Avaliação de Desempenho	73
5.7	Considerações Finais	75
6	PUBLICAÇÕES	77
6.1	Considerações Iniciais	77
6.2	Trabalhos Publicados	77
6.3	Trabalhos Submetidos	77
6.4	Considerações Finais	78
7	CONCLUSÕES	79
7.1	Considerações Iniciais	79
7.2	Conclusões	79
7.3	Trabalhos Futuros	80
7.4	Etapas Cumpridas	80
7.5	Considerações Finais	82
	REFERÊNCIAS	83

INTRODUÇÃO

1.1 Contextualização

A Internet é responsável por fornecer conectividade aos seus usuários. Segundo [Chellouche et al. \(2010\)](#), a complexidade e variedade de serviços disponíveis aumentaram muito, sendo uma das preocupações dos pesquisadores selecionar o melhor servidor disponível com base em informações de carga, tempo de resposta, disponibilidade e outras métricas.

Para conseguir atender a demanda de seus usuários, a replicação de servidores é utilizada. Assim, vários *hosts* que fornecem o mesmo serviço são disponibilizados ao cliente. A replicação permite que a carga seja distribuída entre os servidores, aumentando a capacidade total de atendimento do sistema. Uma técnica utilizada para o roteamento de requisições para a melhor réplica é denominada *anycast*.

Anycast é uma técnica usada para entregar datagramas de um cliente a um dentre vários servidores, sendo definida por [Partridge e Milliken \(1993\)](#). Segundo essa definição, um endereço IP (*Internet protocol*) *anycast* mapeia um grupo de servidores que fornecem o mesmo serviço. Na Figura 1, há um exemplo ilustrando uma situação onde três servidores pertencem a um mesmo grupo, cujo IP é *10.10.10.10*. Ao se comunicar com esse endereço, o cliente envia a pergunta ao IP *10.10.10.10*. Assim, a rede fica encarregada de rotear o pacote para o *host* mais próximo (de acordo com uma métrica do protocolo de roteamento).

A definição original de *anycast* descreve essa técnica sendo utilizada na camada da rede, e é chamada de *IP layer anycast* na literatura. [Bhattacharjee et al. \(1997\)](#), [Fei et al. \(1998\)](#) e [Zegura et al. \(2000\)](#) introduzem um novo conceito, denominado *application layer anycast* (*anycast* na camada de aplicação), em razão de vários problemas no *anycast* tradicional.

No *IP layer anycast*, uma parte dos endereços IP deve ser alocada a endereços *anycast*. Além disso, o uso dessa técnica é dependente de roteadores que a suporte. Ainda, o usuário não tem controle do roteamento, uma vez que ele é realizado no nível da rede. Além disso, quando é

Figura 1 – Exemplo de *anycast*, adaptado de Piper (2013).

necessário o envio de vários datagramas, há a possibilidade destes serem entregues a servidores distintos, quebrando o fluxo de informação. Adicionalmente, as métricas utilizadas por essa técnica estão limitadas pela quantidade de saltos dos roteadores, enquanto no *application layer anycast*, as informações de carga dos servidores podem ser utilizadas.

Na Figura 2, o modelo *anycast* na camada de aplicação é apresentado. O cliente realiza uma pergunta *anycast*, formada por um nome e especificações de filtro, ao resolvidor de nomes *anycast* local. O nome na pergunta representa o domínio *anycast* (ADN, *anycast domain name*). Um ADN é identificado por meio do uso do sufixo “.any”, sendo que as especificações de filtro definem como a requisição deve ser tratada. Essas especificações incluem por exemplo a quantidade de servidores retornada e o algoritmo de seleção aplicado. A hierarquia dos resolvidores *anycasts* é semelhante à utilizada nos servidores de nomes (DNS, *domain name system*), onde nessa hierarquia, há resolvidores locais e autoritativos. Caso um nome não seja encontrado em um resolvidor local, a pergunta é repassada a um resolvidor autoritativo e a resposta fica armazenada em um *cache* no primeiro para consultas futuras. Assim, a resposta retornada ao cliente é interpretada e utilizada para a invocação do serviço desejado.

O *anycast* na camada de aplicação pode ser utilizado em sistemas que utilizam métricas de distância na rede, tais como o RTT (*round trip time*) para selecionar o melhor servidor disponível. Desta forma, o servidor mais próximo do cliente pode ser escolhido para realizar uma determinada tarefa. Segundo Mykoniati *et al.* (2009), há vários cenários onde isso é vantajoso, tais como em redes de distribuição de conteúdo, sistemas de distribuição de dados *peer-to-peer* (P2P), arquiteturas de replicação de servidores *Web* e jogos cooperativos.

Também, esse tipo de *anycast* pode ser utilizado em um sistema de distribuição de

Figura 2 – Esquema *anycast* na camada de aplicação, adaptado de [Zegura et al. \(2000\)](#).

multimídias a fim de balancear a carga nos servidores que disponibilizam esse tipo de conteúdo. Isso é feito no trabalho de [Chellouche et al. \(2011\)](#), que considera tanto a carga dos servidores quanto a distância desses componentes. Na Figura 3, os resultados da utilização do *anycast* nesse trabalho estão expostos. O eixo *x* indica o experimento realizado ao longo do tempo e o eixo *y* mostra a média de pacotes perdidos durante a transferência dos dados. É possível observar o grande impacto da utilização dessa técnica, em que a perda de pacotes é próxima de 0%.

1.2 Motivação

Há três principais desafios que envolvem o *anycast* na camada de aplicação segundo [Ma, Zhou e Zhang \(2009\)](#). Primeiramente, ele deve ser escalável. Em segundo lugar, ele deve minimizar a latência das perguntas *anycast*. Em terceiro, o resultado da pergunta deve ser o melhor nó disponível, levando em conta critérios pré estabelecidos, tais como carga e latência.

Para lidar com o primeiro desafio, o sistema *anycast* não pode manter informações de todos os servidores de forma centralizada. As sondagens do desempenho de cada servidor não podem ser feitas de modo global, pois isso resultará em muito tráfego na rede, aumentando os

Figura 3 – Perda de pacotes em um sistema com e sem *anycast*, adaptado de [Chellouche et al. \(2011\)](#).

tempos de resposta. O segundo desafio deve ser vencido por um algoritmo eficiente capaz de localizar *hosts* mais próximos do cliente de forma rápida. No terceiro desafio, o resultado da pergunta dependerá da forma de seleção dos servidores.

1.3 Objetivos

O objetivo deste trabalho de mestrado é desenvolver um sistema que forneça *anycast* na camada de aplicação de forma descentralizada e eficiente, que possua uma pequena latência de perguntas *anycast* se comparado com outros trabalhos da área e que forneça respostas que representem o melhor servidor disponível.

O contexto da computação em nuvem neste trabalho remete ao fato de que o sistema desenvolvido deve ser genérico o suficiente para fornecer quaisquer serviços, como por exemplo armazenamento de arquivos, páginas *web* e editores de texto em tempo real.

1.4 Organização do Trabalho

No Capítulo 2, o conceito de qualidade de serviço é apresentado e os trabalhos relacionados são descritos. No Capítulo 3, o sistema proposto é modelado. No Capítulo 4, experimentos são realizados por meio de simulação, a fim de validar o sistema apresentado. No Capítulo 5, detalhes sobre o sistema real desenvolvido são descritos, além dos resultados obtidos dos experimentos com esse sistema. As publicações realizadas por meio deste trabalho são listadas no Capítulo 6. No Capítulo 7, as conclusões finais são apresentadas e as limitações e trabalhos futuros são discutidos.

TRABALHOS RELACIONADOS

2.1 Considerações Iniciais

Neste Capítulo, uma breve descrição de qualidade de serviço é discutida com uma contextualização no trabalho realizado. Adicionalmente, uma revisão bibliográfica sobre sistemas que fornecem *anycast* na camada de aplicação é realizada.

2.2 Qualidade de Serviço

O termo *Quality of Service* (QoS) pode ser interpretado de diferentes formas dependendo do contexto em que está situado. Neste trabalho, a QoS está relacionada ao atendimento dos servidores que compõem o sistema. As métricas que podem ser utilizadas para descrever a QoS são: largura de banda, distância na rede, utilização de CPU, poder de processamento, dentre outras ([ESTRELLA; SANTANA; SANTANA, 2011](#)). As métricas consideradas no projeto em questão estão limitadas à distância dos computadores na rede e à carga dessas máquinas.

A distância entre dois nós na rede pode ser determinada por meio de troca de pacotes UDP (*user datagram protocol*). O remetente envia o pacote ao destinatário que responde a mensagem logo que a recebe. O remetente registra o tempo de envio e de chegada dos dados, calculando o atraso causado pela rede. A carga de um *host* pode ser enviada juntamente com o pacote UDP citado anteriormente. Assim, o remetente conhece a distância na rede e a carga da máquina destinatária. Essa métrica pode ser determinada utilizando o comando *uptime*. Por meio dele, é possível determinar a média da carga do sistema, que é calculada pelo número de processos que estão no estado executando ou não-interrompíveis. Um processo está nesses estados se estiver utilizando a CPU, esperando para utilizá-la ou esperando por um acesso de E/S (entrada/saída) ([GREENFIELD; JOHNSON, 1993b](#)).

Considerando QoS, surge o termo SLA (*service level agreement*), que baseia-se no

conceito de serviços fixos e acordos entre clientes e provedores de serviços com relação à qualidade de atendimento. Tais acordos são utilizados para definir o tipo, escopo, a qualidade de serviços e para verificar as especificações conhecidas (ESTRELLA; SANTANA; SANTANA, 2011). Neste projeto, discute-se a implantação de um SLA: o tempo máximo de seleção de um servidor, definida pelo cliente. Porém, nos experimentos não há diferenciação de clientes e servidores. Portanto, não há acordos entre clientes e servidores. Isso significa que o sistema avaliado sempre fará o melhor esforço para fornecer o *anycast* na camada de aplicação aos usuários.

2.3 Propostas Semelhantes

Apesar do trabalho de Wendell *et al.* (2010) não estar relacionado ao *anycast*, o objetivo é muito próximo de um sistema de *anycast*. Os autores propõem o DONAR (*Decentralized Server Selection for Cloud Services*). DONAR é um sistema distribuído capaz de diminuir a complexidade da seleção de servidores e fornecer serviços por meio de uma interface com polípticas de mapeamento. Nesse trabalho, os autores consideram capacidade de requisições, largura de banda e custo como métricas de seleção. Nós de mapeamento são introduzidos para que um mapeamento distribuído possa ser realizado, eliminando assim um coordenador central. DONAR tem algumas desvantagens, como a complexidade da comunicação do sistema. As informações dos nós, como custo, são armazenadas nos servidores de mapeamento. Esses servidores trocam dados para atualizar as informações do sistema, resultando em uma complexidade da ordem do número de nós do sistema multiplicado pelo número de nós de mapeamento. Portanto, em um ambiente dinâmico, isso é um problema, uma vez que em cada atualização as tabelas de dados precisam ser realimentadas. Adicionalmente, os nós de mapeamento precisam conhecer o tamanho da população de clientes, sendo uma decisão relativa, porque é dependente do ambiente e é uma estimativa. Ainda, os autores não definem qual é uma boa escolha para o tamanho da população de clientes.

Jin (2009) propõem um algoritmo de roteamento *anycast* utilizando algoritmos genéticos. O trabalho destaca o *anycast* na camada de rede. Ao adaptar o ambiente de roteamento para o algoritmo genético, os autores podem aplicá-lo e escolher o melhor servidor disponível. Eles consideram apenas atraso na rede como métrica de seleção. A saída do algoritmo genético é o melhor caminho para rotear o datagrama alvo. Os autores simulam o sistema em um ambiente pequeno. Pelo fato das tabelas de roteamento precisarem ser transferidas entre os nós, essa abordagem sinaliza problemas de escalabilidade.

No trabalho de Qin, Taoshen e Zhihui (2009), os autores propõem um algoritmo de roteamento *anycast* baseado em um algoritmo genético e otimização de grupos de partículas. Otimização de grupos de partículas (PSO) é uma técnica de otimização estocástica baseada em populações, inspirada pelo comportamento social de um pássaro ou um peixe. No algoritmo

PSO, o sistema é inicializado com uma população de soluções aleatórias e a busca pela solução ótima é feita por meio da atualização das gerações. O algoritmo proposto é um algoritmo de *anycast* na camada de rede. São consideradas as características dos caminhos como atraso, custo, *jitter* e perda de pacotes. Para encontrar o caminho da fonte para o destino, o algoritmo PSO é executado juntamente com um algoritmo genético. O principal problema com essa estratégia é a necessidade de manter as informações completas do grafo dos roteadores e todas as métricas citadas. Uma vez que esses valores mudam ao longo do tempo, o sistema precisa atualizar o grafo constantemente. Portanto, quando o número de nós cresce, o esforço para manter os dados concisos aumenta consideravelmente.

Walkowiak e Klinkowski (2013) focam seu trabalho em uma comunicação *anycast* em redes óticas elásticas (EONs). EONs pertencem ao processo evolucionário das redes de comunicação óticas. Nesse tipo de rede, os recursos de espectro são utilizados de maneira mais eficiente, sendo suportados larguras de banda elásticas e sob demanda. Nesse trabalho, os autores tentam solucionar o problema de alocação de espectro e roteamento (RSA), que é um problema NP-difícil, utilizando o roteamento *anycast*. Uma estratégia utilizando programação linear inteira é utilizada para resolvê-lo e essa abordagem apresenta melhores resultados que as heurísticas de referência, como *first fit* e *longest path first*.

2.4 Sistemas de Anycast na Camada de Aplicação

Wang Xiaonan diz que o estudo sobre o *anycast* ainda está em seu estágio inicial e há muitos problemas em atender serviços desse tipo, sendo um deles a escalabilidade (XIAONAN, 2010). O autor propõe um esquema escalável para a aplicação do *anycast* na camada de aplicação e, no modelo proposto, bancos de dados compartilham parâmetros relevantes sobre os membros *anycast*. Assim, um servidor de nomes de domínio necessita periodicamente consultar cada membro do sistema por suas informações (como carga e uso de banda) e atualizar os dados armazenados localmente. O problema com essa estratégia é que esse servidor precisa consultar todos os nós do sistema frequentemente, já que as informações dos nós podem variar em um curto período de tempo. Portanto, o sistema não pode escalar muito, mesmo com múltiplos servidores de nomes de domínio. Adicionalmente, o autor afirma que na opinião do cliente, quanto mais curto o RTT, melhor a qualidade do serviço. Essa afirmação não é verdadeira se um servidor próximo está sobrecarregado, sendo melhor escolher um nó ocioso mais distante.

Latif L; Mykoniati (2008) propõem um algoritmo *n-casting*. Em sua arquitetura, o *anycast* é estendido para muitos servidores sendo selecionados para atingir um objetivo. Isso é vantajoso por exemplo em *streaming* de vídeos, onde o cliente pode receber *streams* em paralelo do mesmo vídeo. Os nós do sistema são agrupados por suas distâncias na rede. Para contornar o problema do crescimento exponencial dos dados do sistema, os autores utilizam *hash sketches*. Esses *hashs* diminuem a acurácia a medida que a cardinalidade do número de conjuntos de nós

aumenta. Essa perda de acurácia pode resultar em seleções de servidores ruins (ZHANG *et al.*, 2006). Ainda, o artigo não apresentou nenhum resultado experimental.

Freedman, Lakshminarayanan e Mazières (2006) propõem o OASIS (*Overlay-based Anycast Service Infrastructure*), um sistema de *anycast* na camada de aplicação cujos principais objetivos são fornecer uma resposta *anycast* rápida e correta, minimizar a sondagem por métricas nos servidores do sistema, escalar a quantidade de serviços, prover alta disponibilidade dessas aplicações e integrar-se a qualquer ambiente sem realizar modificações nos clientes.

O sistema OASIS é composto por clientes, réplicas de servidores e nós núcleo, conforme ilustrado na Figura 4. Os clientes são aqueles que utilizam o sistema. As réplicas contêm os serviços a serem executados. Já os nós núcleo ajudam os clientes a selecionar as réplicas apropriadas no processo de execução de um serviço.

Figura 4 – Arquitetura do OASIS, adaptado de Freedman, Lakshminarayanan e Mazières (2006).

Todas as réplicas conhecem sua localização geográfica, isto é, latitude e longitude. No plano de fundo, o OASIS estima a localização geográfica de cada bloco da Internet. Isso é feito utilizando o prefixo dos IPs, uma vez que eles preservam localidade. Esse sistema utiliza despejos (*dumps*) BGPs (*border gateway protocol*) disponíveis ao público para agregar os IPs às localidades.

As informações de serviço, de *bucketing* (mapeamento de endereços IP para prefixos IP), de proximidade (mapeamento de prefixos IP para coordenadas) e de vivacidade (réplicas do sistema) ficam armazenadas em um banco de dados descentralizado.

O processo de resolução de um nome é feita como segue. Primeiramente, o cliente busca nos servidores DNS raízes por um servidor de nomes do OASIS. Esse nó encontra um nó núcleo próximo do cliente e retorna essa informação ao cliente. O cliente então consulta esse nó núcleo por réplicas próximas. O nó núcleo retorna a melhor réplica ao cliente que finalmente invoca a execução do serviço diretamente nesse servidor.

Wang *et al.* (2009) apresentam Proxima, uma infraestrutura da camada de aplicação baseada em NCs (*network coordinates*) que é capaz de fornecer o serviço de *anycast*. Em um sistema NC, cada nó possui um conjunto de coordenadas e a distância na rede entre quaisquer dois nós é calculada utilizando esses valores em uma função de distância (CHEN *et al.*, 2009).

Figura 5 – Arquitetura do Proxima, adaptado de Wang *et al.* (2009).

A arquitetura Proxima possui dois principais componentes e pode ser observada na Figura 5. O primeiro deles, o sistema NC engloba os participantes do sistema. Todos os clientes e servidores computam seus próprios NCs. O sistema NC utilizado é o Pharos. O segundo, a rede de sobreposição de DHTs (*distributed hash tables*), armazena as coordenadas dos *hosts*. As coordenadas são enviadas pelos próprios *hosts* periodicamente.

A resolução de um endereço *anycast* é descrita a seguir. Primeiramente, o cliente realiza uma requisição *anycast*. Ele consulta o DHT buscando por coordenadas virtuais de servidores que possam atender sua requisição. Em seguida, o cliente calcula a distância entre ele e os servidores candidatos, selecionando o servidor mais próximo.

Mykoniati *et al.* (2009) apresentam o DOAT (*Distributed Overlay Anycast Table*), um sistema de *anycast* na camada de aplicação que utiliza curvas de preenchimento espacial para localizar nós próximos pertencentes ao sistema. Um nó possui uma coordenada virtual (NC) X que é mapeada para a coordenada unidimensional DOAT utilizando a curva citada. Assim, utilizando esse valor, os nós do sistema conseguem identificar vizinhos próximos em uma requisição *anycast*.

Na Figura 6, o sistema DOAT está representado. Ele possui três principais componentes, os membros de grupo, nós DOAT e clientes. Os membros de grupo se registram a um grupo do sistema a fim de receber mensagens *anycast*. Os nós DOAT são representantes de grupo e atuam como roteadores das requisições. Eles possuem tabelas de repasse para que possam direcionar as requisições. Os clientes são aqueles que realizam requisições ao sistema.

Figura 6 – Representação do sistema DOAT, adaptado de [Mykoniati et al. \(2009\)](#).

Em uma requisição, quando o nó DOAT recebe uma pergunta, ele verifica em seu registro local se há um membro de seu grupo apropriado para a resposta. Se isso não for possível, ele repassa a requisição para outro nó DOAT utilizando filtros *bloom*, uma estrutura de dados utilizada para testar se um membro faz parte de um grupo. Esse processo se repete até que um membro do grupo local seja encontrado.

OASIS, Proxima e DOAT utilizam previsões para calcular as distâncias entre os nós participantes do sistema. O OASIS analisa os prefixos dos endereços IPs para estimar a localidade dos componentes. A infraestrutura Proxima utiliza o sistema Pharos que oferece coordenadas virtuais como função de distância. No sistema DOAT, a coordenada obtida X é uma estimativa da localização de um *host* na rede. Mesmo utilizando previsão para melhorar o desempenho do sistema *anycast*, a performance final pode ser muito pior que o caso ótimo onde as distâncias reais são conhecidas ([ZHANG et al., 2006](#)). Adicionalmente, a topologia da rede tem grande influência na acurácia dessa previsão ([XING; CHEN, 2008](#)).

Há trabalhos que utilizam aferições reais (por meio do RTT) para calcular a distância entre dois nós na rede, como o AICN, o GAA e o Meridian ([YUAN et al., 2012](#)) ([MA; ZHOU; ZHANG, 2009](#)) ([WONG; SLIVKINS; SIRER, 2005](#)).

[Yuan et al. \(2012\)](#) descrevem o sistema *anycast* AICN (*Anycast for Information Centric Network*) que integra tanto o *anycast* na camada de rede quanto o *anycast* na camada de

aplicação. A arquitetura descrita pelos autores engloba clientes, servidores de mapeamento autoritativo, roteadores de borda (do inglês, *border routers*, BRs) e réplicas. Os clientes realizam requisições de um serviço. As réplicas proveem os serviços. Os BRs são responsáveis por rotear a requisição a uma réplica. Os servidores de mapeamento autoritativo atuam como coordenadores intermediários entre as réplicas e os BRs.

Figura 7 – Arquitetura do sistema AICN, adaptado de [Yuan et al. \(2012\)](#).

Na Figura 7, a arquitetura do AICN está exposta. A requisição de um serviço por um cliente é descrita a seguir. Em primeiro lugar, o cliente realiza a requisição de um serviço. Essa requisição é enviada ao BR mais próximo utilizando *anycast* na camada de rede. Em seguida, o BR requisita ao servidor de mapeamento para que encontre a melhor réplica disponível de acordo com o nome do serviço solicitado, por meio de um algoritmo de otimização de colônia de formigas. Ao receber a resposta, o BR redireciona a requisição por meio do núcleo da rede ao BR mais próximo da réplica. Esse BR então repassa a requisição à réplica em questão. Por fim, a réplica executa o serviço desejado e retorna a resposta diretamente ao cliente.

O problema do sistema proposto por [Yuan et al. \(2012\)](#) é a dependência dos BRs na rede. Além disso, não é possível determinar a eficácia do AICN, uma vez que o sistema não foi comparado a outros sistemas de *anycast* e os testes realizados envolveram poucos nós em funcionamento.

[Ma, Zhou e Zhang \(2009\)](#) descrevem o GAA (*Global Application-layer Anycast*), um *framework* escalável que oferece o *anycast* na camada de aplicação. Ele é derivado de um outro sistema, o Meridian, proposto por [Wong, Slivkins e Sirer \(2005\)](#). Os trabalhos são muito semelhantes. A diferença é que o primeiro utiliza sondagens centralizadas no cliente enquanto o

segundo utiliza sondagens centralizadas nos nós do sistema.

Figura 8 – Vizinhança de um nó no GAA, adaptado de [Ma, Zhou e Zhang \(2009\)](#).

De acordo com os resultados obtidos a partir de simulações, é possível afirmar que a latência média das consultas *anycast* é menor no GAA, se comparado com o Meridian. Considerando a eficiência, que é a razão entre a distância entre o nó selecionado e o nó mais próximo realmente, ambos os algoritmos apresentam esse valor próximo a 75%.

No sistema GAA, cada nó possui uma vizinhança que é dividida em anéis. Cada anel possui no máximo $\log n$ vizinhos, onde n é o número total de nós do sistema. Quanto mais próximo um vizinho é do nó, mais interior é o anel ao qual ele pertence. A proximidade é definida pelo RTT entre dois *hosts*. A vizinhança de um nó arbitrário está ilustrada na Figura 8.

Em uma consulta *anycast*, um nó aleatório inicial é escolhido na rede de sobreposição formada pelos nós do sistema. O cliente então calcula sua distância para esse nó. O nó, de acordo com um parâmetro definido, seleciona vizinhos e envia essa lista ao cliente. Em seguida, o cliente calcula a distância para todos esses nós da lista. Se o nó mais próximo é o inicial, o algoritmo termina e esse nó é eleito como o mais próximo. Caso contrário, esse processo se repete sendo escolhido o nó mais próximo da lista como o nó inicial.

O problema do *framework* GAA é a escolha aleatória do nó inicial. A desvantagem dessa ação é o fato de que um nó muito distante do cliente pode ser escolhido inicialmente. Se isso ocorrer, o algoritmo leva mais passos para convergir, resultando em tempos totais de requisição elevados.

2.5 Considerações Finais

Conforme descrito neste Capítulo, as principais métricas de qualidade de serviço que serão utilizadas neste trabalho são a distância na rede entre dois nós e a carga dos servidores. Muitos trabalhos utilizam predição de distância para encontrar o melhor nó disponível. Porém, outras pesquisas mostraram que o erro introduzido por essa técnica pode prejudicar o desempenho de seleção. Também foi possível identificar problemas nos outros sistemas de *anycast* na camada de aplicação propostos. Ainda, os trabalhos apresentados realizam pouca ou nenhuma experimentação. Além disso, não são feitas comparações entre eles. Isso mostra que sistemas desse tipo ainda possuem pontos em aberto que podem ser explorados em novas pesquisas.

O projeto proposto utiliza algoritmos utilizados por sistemas de *anycast* na camada de aplicação existentes. Ele diferencia-se dos outros trabalhos combinando distâncias reais entre nós na rede e predições por meio de geolocalização. Ainda, são feitas comparações com um sistema existente por meio de experimentos simulados e reais.

DESENVOLVIMENTO DO TRABALHO

3.1 Considerações Iniciais

Neste Capítulo, o sistema proposto é modelado. Primeiramente, uma visão geral é apresentada. Em seguida, as principais etapas que envolvem o funcionamento do sistema são detalhadas.

3.2 O Sistema GALA

A visão geral do sistema GALA está exposta na Figura 9. De acordo com o ilustrado na Figura, há seis passos principais em uma busca *anycast*. São eles: início da busca *anycast* (1), mapeamento da pergunta *anycast* (2), resolução da pergunta *anycast* (3), busca da réplica mais próxima (4), invocação do serviço na réplica escolhida (5) e consulta de dados para a execução do serviço (6).

O cliente é responsável por enviar requisições *anycast* para o sistema GALA e receber as respostas de forma transparente. Exemplos de aplicações clientes são: navegadores *web*, programas que acessam a rede, etc. O sistema operacional (SO) do cliente é responsável por procurar um resolvidor de nomes de domínio na sua lista local de resolvedores. No sistema operacional Linux, essa lista é tipicamente definida em `/etc/resolv.conf`. O SO também é responsável por enviar perguntas ao resolvidor. No sistema GALA, a lista de resolvedores contém um endereço IP de um publicador GALA. Isso é suficiente para enviar perguntas *anycast* para o sistema de *anycast* na camada de aplicação.

Publicadores são nós que contêm os dados de localização dos servidores do sistema. Neste projeto, um servidor DNS é utilizado, porém GALA permite ao analista do sistema utilizar outras tecnologias, como servidores HTTP. O sistema pode conter um único servidor DNS ou vários. Por simplicidade, um único servidor é utilizado no cenário apresentado.

Figura 9 – Visão geral do sistema GALA, adaptado de [Adami et al. \(2015b\)](#).

A aplicação resolvidora GALA recebe perguntas de nome de domínio de clientes e as responde. Se o nome a ser resolvido é um tradicional (i.e., não é um nome *anycast*), a pergunta é resolvida utilizando as rotinas tradicionais do sistema. Se o nome a ser resolvido é do tipo *anycast*, ele é processado como discutido na Subseção 3.6.

Servidores oferecem serviços aos clientes. Eles tipicamente possuem três módulos: o recebedor de requisições, os dados de serviço e o *daemon* GALA. O recebedor de serviços é a aplicação do serviço propriamente dita. Ela dependerá do serviço sendo oferecido. Por exemplo, se é um conteúdo *web*, o recebedor de requisições é um servidor HTML. Os dados do serviço são as informações armazenadas no nó necessárias para a execução do serviço. Considerando o exemplo do servidor HTML, esses dados são documentos das páginas *web*. Finalmente, o *daemon* GALA é responsável por integrar o nó ao sistema em geral.

3.3 Início da Busca *Anycast*

O primeiro passo da busca *anycast* consiste em realizar a tradução de um nome *anycast* para um nome de hospedeiro tradicional. O nome *anycast* segue a estrutura “*.any”. O cliente pode iniciar essa resolução de nomes de várias formas. Por exemplo, ela pode ser realizada por meio de um navegador ou utilizando a chamada de sistema *gethostbyname*.

O sistema do cliente deve ser configurado de forma a consultar um DNS que suporte o formato *anycast*. Ao invés de utilizar um DNS com suporte total às requisições, um resolvidor próprio é proposto, denominado resolvidor GALA. As principais funções desse resolvidor são:

- Receber mensagens de resolução de nomes (pacotes DNS), de acordo com a especificação definida por [P. Mockapetris \(1987\)](#);
- Diferenciar nomes *anycast* de nomes padrão;
- Repassar resolução de nomes que não sejam do tipo *anycast*;
- Responder às perguntas cujo nome é da forma “*.any”.

Para receber mensagens, o resolvidor GALA escuta na porta 53 utilizando o protocolo UDP. A identificação de nomes *anycast* é feita por meio de um *parsing* do pacote de pergunta. O resolvidor não cobre todas as possibilidades de perguntas definidas no RFC. Caso ele não suporte alguma pergunta o resolvidor delega a responsabilidade da requisição a um DNS comum.

3.4 Geolocalização de Endereços IP

A pergunta *anycast* deve ser mapeada para um grupo de servidores que responderão de forma rápida e eficiente à requisição do cliente. Conforme mostrado na Figura 10, a correlação entre o RTT e a distância geográfica entre dois *hosts* na rede é alta. Por isso, a distância geográfica é utilizada para encontrar o grupo citado.

A geolocalização é a ação de, a partir de um endereço IP, mapear uma posição geográfica onde aquele endereço está localizado. Ela é importante para pesquisas acadêmicas, aplicações comerciais e segurança nacional. A obtenção dos valores de latitude e longitude pode ser feita de várias formas segundo [Padmanabhan e Subramanian \(2001\)](#). São elas: incorporar a informação de localização nos registros do DNS, utilizar o banco de dados *whois* para determinar a localização da organização ao qual o endereço IP pertence, utilizar a ferramenta *traceroute* para mapear os nomes dos roteadores no caminho de uma rota e realizar uma tabulação exaustiva dos endereços IP e suas localizações. Adicionalmente, vários bancos de dados e ferramentas que realizam esse mapeamento estão disponíveis na Internet.

Figura 10 – Correlação entre RTT e distância geográfica, adaptado de [Freedman, Lakshminarayanan e Mazières \(2006\)](#).

A tarefa de avaliação do desempenho de serviços de geolocalização é complexa. [Shavitt e Zilberman \(2011\)](#) utilizam PoPs (*points of presence*) na avaliação desses bancos de dados. PoPs são grupos de roteadores os quais pertencem a um único AS (*autonomous system*) e são fisicamente localizados no mesmo sítio ou campus. Por esse motivo, há maior probabilidade de que haja informações públicas de latitude e longitude desses locais que sejam bem conhecidas. Assim, endereços IP são mapeados para esses PoPs. Segundo os autores, as chances de dois endereços IP pertencerem ao mesmo PoP e não estarem situados na mesma localização geográfica é muito pequena. Ao total, foram identificados aproximadamente 3800 PoPs no trabalho. [Huffaker, Fomenkov e Claffy \(2011\)](#) comparam as respostas das bases de dados usando uma metodologia que compara cada resposta com a resposta da maioria das bases para um dado endereço IP.

A melhor forma de se avaliar a acurácia dos bancos de geolocalização é comparar a informação disponível neles com a localização real de cada endereço IP (chamado de base verdadeira, *ground truth database*). Porém, esse banco ideal não existe. No trabalho, [Shavitt e Zilberman \(2011\)](#) utilizam um banco de dados disponibilizado pela CAIDA (*The Cooperative Association for Internet Data Analysis*) com 25 mil endereços IP que possuem a localização geográfica bem conhecida. [Huffaker, Fomenkov e Claffy \(2011\)](#) utilizam três bases distintas, contendo respectivamente, 1 mil, 6 milhões e 23 mil endereços IP.

Os bancos de geolocalização estudados no primeiro trabalho foram o GeoBytes, HostIP, IP2Location, IPLigence, MaxMind, NetAcuity e Spotter. Utilizando a base verdadeira, os autores descobriram que no nível de cidade (considerando uma distância de 100km) a acurácia das bases não ultrapassou os 30%, com exceção do banco NetAcuity, que respondeu corretamente a 79% dos endereços. A acurácia no nível de país na média ficou próximo a 80% para todas as tabelas com exceção novamente do NetAcuity, que acertou 96% de suas respostas. Esse resultado pode ser explicado porque a CAIDA possui uma parceria com o provedor dessa base

de dados. Ainda, outro teste de acurácia foi realizado. Esse teste tomou como base a média das respostas de todas as bases para determinar o resultado correto da localização de um endereço. As observações foram realizadas considerando distâncias de até 40km, caracterizando uma cidade, e 500km, caracterizando uma região. Os melhores bancos de geolocalização foram o IPLigence e o MaxMind, que acertaram de 62% a 73% dos endereços IP para os 40km e mais de 80% dos endereços IP para os 500km.

No segundo trabalho, as bases de dados utilizadas foram a Software77, HostIP, IPLigence, Cyscape, MaxMind e NetAcuity. Os autores classificaram as tabelas em relação à granularidade de blocos IP em três classes: pequena, média e grande. As bases MaxMind e NetAcuity foram classificadas como granularidade alta e também obtiveram a maior acurácia de respostas utilizando a metodologia de centróides vencedores, com valores 93% e 78% para um raio de 40km, respectivamente. O banco NetAcuity obteve o melhor resultado para as duas bases verdadeiras menores enquanto que o banco MaxMind obteve o melhor resultado para a base verdadeira maior.

Com base nos trabalhos estudados, para este projeto, a base de dados MaxMind foi utilizada. Ela disponibiliza uma versão grátis de seu serviço de geolocalização. Essa base de dados é menos precisa que a completa. Porém, por motivos financeiros, ela foi escolhida neste trabalho. Adicionalmente, os endereços são mapeados no nível de país, uma vez que os resultados apresentados mostraram uma acurácia melhor para esse nível de geolocalização. Ainda, conforme discutido na Seção 5.5, a acurácia dessa base de dados não foi suficiente para a execução do sistema em um ambiente real.

3.5 Mapeamento da Pergunta Anycast

O mapeamento da pergunta *anycast* é realizado no próprio resolvidor *anycast*. A partir de um endereço IP, coordenadas geográficas são extraídas e mapeadas para um servidor que possui as réplicas disponíveis pertencentes àquela região. Na Figura 11, os passos desse mapeamento estão expostos.

O cliente envia um pacote DNS com um nome *anycast* que deseja resolver, por exemplo *gala.com.any*, ao resolvidor (1). O resolvidor, por sua vez, extrai o endereço IP do cliente a fim de mapear suas coordenadas geográficas. As coordenadas geográficas são obtidas por meio de uma consulta em tabelas estáticas e locais, fornecidas pelo serviço MaxMind (2 e 3). Os registros dessas bases seguem os formatos descritos nas Tabelas 1 e 2.

Na Tabela 1, o número IP representa o endereço IP na forma binária, sendo um inteiro de 4 bytes sem sinal. O código do país segue o formato definido pelo ISO 3166 (2013). Os campos utilizados são o número IP inicial, o número IP final e o código do país. Esses registros são carregados de um arquivo no formato *.csv* e colocados em um vetor ordenado. A ordenação do vetor leva em consideração a faixa inicial do endereço IP e assume-se que não há sobreposição

Figura 11 – Mapeamento no resolutor GALA, adaptado de [Adami et al. \(2015\)](#).

Tabela 1 – Formato do registro de faixas de endereço IP.

Campo	Descrição
Endereço IP inicial	Endereço inicial da faixa de endereços para o país do registro
Endereço IP final	Endereço final da faixa de endereços para o país do registro
Número IP inicial	Número IP inicial da faixa citada
Número IP final	Número IP final da faixa citada
Código do país	Código do país do registro, 2 caracteres
Nome do país	Nome do país do registro

de faixas. Portanto, a consulta por um registro a partir de um endereço IP é feita em $O(\log n)$.

Na Tabela 2, os códigos dos países seguem o formato especificado pelo [ISO 3166 \(2013\)](#). Os valores de latitude e longitude foram obtidos do *World Factbook* da [CIA \(2013\)](#). Esses registros são carregados de um arquivo no formato *.csv* e armazenados em uma árvore rubro-negra. A busca é feita por meio da chave do registro: o código do país. Assim, a consulta nessa tabela é feita em $O(\log n)$.

A primeira base é utilizada para encontrar o registro correspondente ao endereço IP

Tabela 2 – Formato do registro de coordenadas dos países.

Campo	Descrição
Código do país	Código do país do registro, 2 caracteres
Latitude	Latitude média do país
Longitude	Longitude média do país

do resolvedor (2). Ao obter o código do país do registro da primeira busca, a segunda base é consultada e o registro do país encontrado é recuperado juntamente com as coordenadas geográficas (3).

Ainda, é necessário mapear as coordenadas obtidas para uma região geográfica. Adicionalmente, o número de áreas geográficas a serem utilizadas deve ser flexível. Portanto, o cálculo da área geográfica, definida por x e y , é feito segundo as Equações 3.1 e 3.2, respectivamente. A variável “regionNodesCount” configurada define o número de regiões do sistema. Os valores de latitude e longitude são primeiramente convertidos para a projeção Azimuthal (Weisstein, Eric W., 2015).

$$y = \frac{\text{latitude} + 90}{\frac{180}{\text{regionNodesCount}}}, \quad (3.1)$$

$$x = \frac{\text{longitude} + 180}{\frac{360}{\text{regionNodesCount}}}. \quad (3.2)$$

Na Figura 12, um exemplo de mapeamento considerando 36 regiões geográficas está exposto. A origem da grade [0,0] se situa no canto superior esquerdo e os valores de x e y aumentam em direção ao canto inferior direito, cujo valor é [5,5]. O Brasil está localizado na área [2,3].

Ao obter os valores x e y e utilizando o registro do país encontrado (seus respectivos valores de latitude e longitude), o resolvedor constrói um novo nome de hospedeiro no formato $xYyY.nome$ (no mapeamento acima, o Brasil está mapeado para $x2y3.gala.com$) e inicia uma consulta ao DNS (4) para sua tradução. A resposta dessa operação possui um ou mais endereços IP, os quais correspondem aos endereços dos *hosts* do grupo pertencente à área geográfica do resolvedor. Ao obter essa lista, o resolvedor GALA retorna ao cliente os registros obtidos nesse passo final (5).

3.6 Resolução da Pergunta Anycast

A resolução da pergunta *anycast* é feita utilizando a hierarquia DNS. No sistema, há um único servidor DNS que contém todos os registros da rede de servidores. Para fins de disponibilidade, esse servidor pode ser replicado utilizando *anycast* na camada de rede, assim como é feito nos servidores raízes (MYKONIATI *et al.*, 2009). Adicionalmente, esse servidor

Figura 12 – Exemplo de mapeamento em 36 regiões geográficas (JONES, 1998).

pode utilizar DLZ (*dynamically loadable zones*), permitindo o armazenamento dos registros em um banco de dados (Stichting NLnet, 2002).

Os servidores do sistema registram-se no DNS por meio de *dynamic updates* (VIXIE *et al.*, 1997) (Donald E. Eastlake 3rd, 1997). Uma mensagem de registro, dentre outros dados, deve conter as informações de endereço IP e o nome de hospedeiro a ser atualizado. O primeiro dado é o próprio endereço IP do servidor e a segunda informação é obtida por meio de um resolvidor *anycast* que realiza o mapeamento utilizando a geolocalização citada na Seção anterior.

Na Figura 13, um exemplo de mapeamento de servidores GALA em um DNS real é apresentado. É possível observar as configurações para um sistema identificado por *gala.com*, onde há quatro regiões geográficas, mapeadas por *x2y5gala.com*, *x4y4.gala.com*, *x5y4.gala.com* e *x5y5.gala.com*. Ainda há um registro global *g.gala.com* que é utilizado quando uma requisição é feita a uma área geográfica onde não há servidores.

3.7 Vizinhança dos Servidores GALA

Os servidores do sistema são aqueles que fornecem o serviço aos clientes. Eles se comunicam para formar uma vizinhança concêntrica, conforme mostrado na Figura 14. A vizinhança de um servidor é definida por R anéis. Os vizinhos mais próximos se encontram nos anéis mais internos enquanto que os vizinhos mais distantes se situam em anéis mais externos. A proximidade de um servidor é dada pela aferição de seu RTT.

O cálculo do anel ao qual um servidor vizinho pertence é feito utilizando dois parâmetros, a e s , conforme descrito a seguir. Obtido o RTT entre os dois componentes, a fórmula $R_i = as^i$ é

```

$ORIGIN .
$TTL 259200      ; 3 days
gala.com                IN SOA  ns.gala.com. hostmaster.gala.com. (
                                5821      ; serial
                                28800     ; refresh (8 hours)
                                7200      ; retry (2 hours)
                                2419200   ; expire (4 weeks)
                                86400     ; minimum (1 day)
                                )
                                NS       ns.gala.com.

$ORIGIN gala.com.
$TTL 30 ; 30 seconds
g                                A       104.131.15.150
                                A       107.170.224.226
                                A       107.170.240.117
                                A       128.199.158.76
                                A       162.243.134.211
                                A       178.62.113.124
                                A       178.62.113.132
                                A       178.62.214.156
                                A       178.62.214.166
                                A       178.62.227.207

$TTL 259200      ; 3 days
localhost          A       127.0.0.1
ns                  A       104.131.189.189
$TTL 30 ; 30 seconds
x2y5                A       104.131.79.70
                                A       128.199.158.76
                                A       128.199.223.34
x4y4                A       104.131.15.150
                                A       104.131.98.167
                                A       107.170.224.226
                                A       128.199.236.251
                                A       162.243.134.211
                                A       178.62.113.132
                                A       178.62.113.164
                                A       178.62.214.156
                                A       178.62.214.166
                                A       178.62.227.207
x5y4                A       107.170.240.117
x5y5                A       178.62.113.124

```

Figura 13 – Exemplo de mapeamento de servidores GALA no DNS do sistema.

utilizada para determinar qual o anel i correspondente. Por exemplo, todo servidor cujo RTT for menor que as^0 fará parte do anel 0, menor que as^1 do anel 1 e assim sucessivamente até o anel R_{R-2} . O último anel, R_{R-1} , abriga todos os servidores cujos RTTs são maiores que as^{R-1} .

A vizinhança de um servidor não contém todos os nós do sistema para que a escalabilidade do sistema não seja afetada. Assim, por meio de um parâmetro k , o número máximo de

Figura 14 – Vizinhança do sistema GALA.

hosts em um anel é definido. Consequentemente, o número máximo de vizinhos de um servidor é kR .

Um servidor, ao entrar no sistema, escolhe outro *host* para inicializar sua vizinhança. Essa escolha é feita da mesma forma que um cliente busca um nó inicial. Conhecendo esse componente, o servidor envia um pacote de requisição de vizinhança a ele. Ao receber a resposta, o servidor constrói sua vizinhança com todos os vizinhos do nó escolhido incluindo-o como vizinho.

Periodicamente, cada servidor informa aos seus vizinhos uma parte da sua vizinhança de forma independente. Esse processo é denominado de *gossip*. Primeiramente, o servidor escolhe um nó de cada anel de sua vizinhança de forma aleatória, formando um conjunto de servidores G . Após essa escolha, ele escolhe novamente um nó de cada anel, arbitrariamente, e envia um pacote

contendo o conjunto G aos nós escolhidos. Cada *host* destinatário então adiciona os servidores contidos no pacote recebido à sua vizinhança, incluindo o remetente. Caso um servidor em G já esteja na vizinhança de nó destinatário, seu anel é atualizado. É possível que o próprio nó que recebeu o pacote esteja no conjunto de servidores. Nesse caso, ele ignora esse elemento, não adicionando a si próprio em sua vizinhança. Se o nó destinatário recebe um novo vizinho, ele envia um outro pacote a esse elemento com todos os nós no anel desse servidor. Esse novo nó ao receber esse pacote, adiciona os nós a sua vizinhança ou atualiza o anel do componente para casos em que ele já é um vizinho.

Um servidor pode receber vários pacotes *gossip* ao longo do tempo. Assim, ele vai adquirindo novos vizinhos e o número de nós em um anel pode ultrapassar o valor estipulado pelo parâmetro k . O máximo de nós extras, ou secundários, em um anel é definido por um parâmetro l . Então, de forma periódica e independente do sistema, o servidor realiza o processo de poda de sua vizinhança, descrito a seguir. Para cada anel, o servidor verifica se há mais que $k + l$ nós vizinhos. Caso isso seja verdade, vizinhos secundários são escolhidos de forma aleatória e descartados até que restem l nós secundários. Após esse passo, o servidor envia um pacote de consulta para cada nó no anel, contendo o conjunto de nós desse anel. Ao receber um pacote de consulta, um servidor do sistema calcula o RTT para todos os nós contidos no conjunto do pacote e retorna essa informação ao remetente. Ao obter esse conjunto de informações, o servidor constrói um vetor de distâncias de cada nó. Utilizando esses dados ele determina por meio de uma função de distância os m vizinhos ($1 \leq m \leq l$) com menor distância e determina que eles sejam os novos nós vizinhos extras.

Um anel com nós distribuídos geograficamente oferece muito mais utilidade que um anel com nós que estão agrupados (WONG; SLIVKINS; SIRER, 2005). Isso é explicado pelo fato de que nós geograficamente diversos permitem um nó a repassar uma requisição a uma região mais ampla. Portanto, quanto menor a distância entre os nós do anel, pior a diversidade geográfica desse anel. Para o cálculo de distância dos nós em um anel, funções de distância podem ser utilizadas. Neste trabalho, a distância de Chebchev e Euclideana, descritas pelas Equações 3.3 e 3.4, respectivamente, foram avaliadas.

$$D_{ij} = \max_{l=1}^k RTT_{il}, \quad (3.3)$$

$$D_{ij} = \sqrt{\sum_{l=1}^k RTT_{il}^2}. \quad (3.4)$$

3.8 Busca da Réplica mais Próxima

O algoritmo descrito a seguir pode ser representado pelo fluxograma presente na Figura 15. O cliente envia um pacote ao nó inicial do sistema, obtido conforme descrito anterior-

mente. A distância entre o nó inicial e o cliente é calculada e vale D_{cs} . Então, o nó inicial responde ao cliente com todos seus vizinhos cuja distância, D_{sx} , está entre $(1 - \beta)D_{cs}$ e $(1 + \beta)D_{cs}$. O parâmetro β é um parâmetro do sistema. Ao receber a resposta, o cliente calcula a distância para todos os vizinhos. Caso exista um nó cuja distância seja menor que βD_{cs} , esse componente é definido como o nó inicial e o processo se repete. Se houver mais que um servidor próximo do cliente, o servidor com a menor distância é considerado. O processo se repete até que não haja um nó que satisfaça a regra descrita acima. Quando isso ocorre, o sistema encontrou o nó mais próximo do cliente, que é o nó inicial da iteração atual (MA; ZHOU; ZHANG, 2009).

3.9 Considerações Finais

Conforme descrito neste Capítulo, o sistema GALA possui os seguintes componentes: o cliente, o resolvidor *anycast*, a hierarquia de DNS e os servidores do sistema. Adicionalmente, esse sistema envolve quatro passos principais: início da busca *anycast*, mapeamento da pergunta *anycast*, resolução da pergunta *anycast* e busca pela réplica mais próxima. A busca pela réplica mais próxima depende da vizinhança construída pelo sistema, inerente de trabalhos realizados por Ma, Zhou e Zhang (2009) e Wong, Slivkins e Sirer (2005).

Figura 15 – Fluxograma do algoritmo de busca.

AVALIAÇÃO EXPERIMENTAL POR SIMULAÇÕES

4.1 Considerações Iniciais

Experimentos foram realizados com o objetivo de avaliar o comportamento do sistema GALA inicialmente por meio de simulações. O principal objetivo dos experimentos simulados é responder à hipótese de que com a seleção de nós iniciais mais próximos dos clientes, utilizando a geolocalização, o desempenho geral do sistema deve melhorar. Para a configuração dos experimentos, considerou-se utilizar o fatorial completo com fatores variando entre dois níveis cada. A análise dos resultados foi feita utilizando o modelo de regressão linear e os gráficos foram gerados pelo software Minitab ([Minitab Inc, 2013](#)).

Nos gráficos de efeitos (Figuras 24, 28 e 32), o eixo x representa o efeito padronizado de cada combinação de fatores obtido pela regressão linear, enquanto o eixo y é o resultado da ordenação das combinações dos efeitos e os valores são dados pela equação da ordem da mediana:

$$\frac{i - 0.3}{n + 0.4}, \quad (4.1)$$

onde n equivale ao número de combinações e i a ordem da combinação. A reta no gráfico é uma reta do tipo $x = y$. Caso um ponto caia à sua esquerda, ele possui um efeito negativo na variável de resposta, diminuindo esse valor na troca dos níveis dos fatores envolvidos de baixo para alto. Caso um ponto caia à sua direita, ele possui um efeito positivo na variável de resposta, aumentando esse valor na troca dos níveis dos fatores envolvidos de baixo para alto. Para indicar se um efeito é significativo ou não, o Minitab utiliza o teste do p -valor.

Nos gráficos de interações (Figuras 26, 30 e 34), cada quadro indica a interação entre dois fatores, utilizando a média de cada nível de um fator e mantendo o nível do segundo fator

constante. Nesses gráficos, é possível julgar a presença de interações, que ocorrem quando as linhas dos gráficos estão perpendiculares. Quanto mais perpendiculares forem os segmentos maior a interação. Quanto mais paralelos, menor a interação.

Nos gráficos de barras (Figuras 20, 21, 22, 23, 27 e 31), a distribuição de uma variável de resposta é sumarizada, mostrando sua tendência central e sua variabilidade. O mesmo é válido para os gráficos de intervalos (Figuras 35 e 36). Porém, as médias das variáveis de resposta não são evidenciadas no segundo.

Nos gráficos de efeitos principais (Figuras 25, 29 e 33), as médias da variável de resposta para cada nível de cada fator são plotadas, conectando-os por uma linha.

Nos histogramas (Figuras 37 e 38), a porcentagem de ocorrências para os possíveis valores das variáveis de resposta são ilustradas. No histograma cumulativo (39), as frequências são somadas e o gráfico é aproximado para uma normal. O Minitab estima os parâmetros da normal a partir dos dados (média e desvio padrão). O eixo x representa os possíveis valores da variável de resposta, enquanto que o eixo y indica a porcentagem de ocorrências para tais valores sobre todas as observações obtidas.

4.2 O Simulador GALASim

O simulador GALASim foi desenvolvido utilizando a linguagem Java. Ele é baseado em turnos. Um turno é composto dos seguintes eventos, nessa ordem de execução: iteração em todos os nós servidores, realizando os processos de *gossip* e poda, e criação de requisições. A iteração dos nós servidores é feita de forma aleatória, não seguindo a mesma ordem dos nós. A criação de requisições é realizada somente após um número determinado de turnos. Isso é feito para que o sistema possa se estabilizar antes da realização dos testes. Em cada turno, cada cliente realiza uma requisição *anycast* e os resultados obtidos para cada requisição são registrados em um banco de dados MySQL. A simulação é executada até que um número determinado de turnos tenha sido completado.

Na Figura 16, o diagrama de classes simplificado do simulador está exposto. A classe “NetworkLoader” é responsável por carregar os nós servidores e clientes que compõem o sistema. As classes “Node” e “Client” são as entidades que participam do sistema. Ambas herdam características da classe “Computer”. A classe “Simulation” é a classe principal do simulador e é responsável por gerenciar a simulação. Ela possui características do sistema GAA. A classe “GalaSimulation” herda a classe simuladora e sobrescreve os métodos necessários para a implantação do sistema GALA. A classe “Database” é responsável por registrar *logs* de requisições de clientes na simulação.

Os registros de *log* são armazenados em um banco de dados relacional conforme ilustrado na Figura 17. Há três tabelas: “Request”, “Response” e “Step”. Os registros dessas tabelas são

Figura 16 – Diagrama de classes simplificado do simulador GALASim.

indexadas pelos campos “id” e “runId” que correspondem, respectivamente, ao identificador do experimento e o número de sua replicação.

A tabela “Request” armazena informações gerais de uma requisição *anycast*. O campo “simulationClass” indica qual classe é utilizada como responsável pela simulação (“Simulation” ou “GalaSimulation”). O campo “simulationStep” indica qual o turno atual da simulação. Os campos “clientLatitude” e “clientLongitude” indicam qual a localização geográfica do cliente que gerou o registro. O atributo “worstRingNodeMethod” registra qual o método de distância utilizado. Os campos “nodesFile” e “clientsFile” armazenam os nomes dos arquivos utilizados no carregamento dos nós servidores e clientes, respectivamente, do sistema. O campo “nodes-RegionsCount” indica em quantas regiões geográficas os servidores do sistema são divididos. “mappingLevels” indica os níveis de subdivisões criadas para as regiões geográficas e “mapping-Factors” descreve a granularidade em cada nível. Os dois últimos campos são utilizados na visão do cliente a fim de que o algoritmo de busca encontre servidores em regiões vizinhas caso não haja servidores em uma determinada região. Por exemplo, considere um mapeamento com 3 níveis e fator 2, sendo que o número de regiões geográficas definido é 16. O número de regiões geográficas será 16, 64 e 256 para o primeiro, segundo e terceiro níveis, respectivamente.

A tabela “Response” mantém resultados sumarizados de uma requisição. O campo “hopsCount” armazena o número de servidores consultados no processo de busca do *host* mais próximo. O valor de “efficiency” é obtido pela razão entre a distância do nó mais próximo real do cliente e a distância do nó mais próximo encontrado pelo sistema. Finalmente, o atributo “queryLatency” representa a latência total da requisição do cliente.

A tabela “Step” armazena informações de cada servidor utilizado em uma busca *anycast*.

Figura 17 – Diagrama do banco de dados do GALASim.

O campo “hop” representa qual o servidor consultado para uma busca. “serverLatitude” e “serverLongitude” indicam qual a localização geográfica do servidor. O campo “rtt” registra qual a distância entre o servidor e o cliente que gerou a requisição.

4.3 Configurações dos Experimentos Simulados

Para todos os experimentos simulados por meio do GALASim, os parâmetros presentes na Tabela 3 (fatores fixos) foram utilizados. Na Tabela, são descritos todas as variáveis de configuração do sistema. Os valores foram impostos considerando experimentos anteriores dos trabalhos de Wong, Slivkins e Sireer (2005) e Ma, Zhou e Zhang (2009). O fator RTT é o valor utilizado no cálculo do RTT entre dois nós e considera uma linha geodésica entre dois componentes (HUFFAKER; FOMENKOV; CLAFFY, 2011). Os clientes iniciam as requisições a partir do quarto turno.

Os dados de entrada utilizados foram o conjunto de nós do PlanetLab (Princeton Uni-

Tabela 3 – Parâmetros das Simulações.

Parâmetro	Valor
Turnos de Requisição	20
Fator RTT	$5.128205 * 10^{-9}$
Nós por Anel (k)	8
Nós Secundários por Anel (l)	2
Anéis (R)	11
a	1
b	0.5
s	2
Turno Inicial de Requisições	4
Regiões Geográficas	64
Replicações de Experimento	10

versity, 2012) e GeoNames (Marc Wick, 2013). Os dados são lidos de um arquivo ASCII pelo simulador. Cada linha do arquivo possui a latitude e a longitude de um servidor. Os grupos de nós foram obtidos dos dados fornecidos publicamente no site do PlanetLab e GeoNames. Na primeira configuração de entrada, os nós do PlanetLab são servidores, enquanto que os componentes do GeoNames são clientes (PlanetLab/GeoNames), conforme ilustrado na Figura 18. Na segunda configuração de entrada, um subconjunto das máquinas do GeoNames são servidores e os nós do PlanetLab são clientes (GeoNames/PlanetLab), mostrado na Figura 19. Ainda, nas Figuras, as 64 regiões geográficas estão representadas.

Figura 18 – Entrada com servidores GeoNames como clientes e nós PlanetLab como servidores (ADAMI *et al.*, 2014).

Como variáveis de resposta, são consideradas as médias dos números de saltos, das eficiências e das latências das requisições dos clientes. O número de saltos representa o número

Figura 19 – Entrada com servidores PlanetLab como clientes e nós GeoNames como servidores (ADAMI *et al.*, 2015b).

de servidores que participaram na requisição. A eficiência é obtida pela razão entre a distância na rede do servidor mais próximo de fato e o servidor encontrado pelo sistema. A latência é o tempo total da requisição.

4.4 O Primeiro Experimento

O objetivo do primeiro experimento é analisar como a granularidade das regiões geográficas afetam o desempenho do sistema. Na Tabela 4, os fatores de dois níveis utilizados no experimento são expostos. Os níveis de mapeamento indicam o número de subdivisões criadas. Por exemplo, 3 níveis de mapeamento para 64 regiões significam 1024 regiões, ao invés de 64 para um nível de mapeamento de valor 1. Adicionalmente, o algoritmo GALA foi utilizado na busca pelos melhores servidores para atender os clientes.

Tabela 4 – Fatores do Primeiro Experimento.

Fator	Níveis
Níveis de Mapeamento	1, 3
Dados de Entrada	PlanetLab e GeoNames

Na Figura 20, a eficiência das requisições, para cada combinação de fatores, é apresentada. A eficiência é estatisticamente a mesma para diferentes níveis de mapeamento. Esse comportamento é o esperado, já que o algoritmo de busca converge para o melhor servidor independentemente da escolha inicial do servidor.

Figura 20 – Eficiência dos níveis de mapeamento, adaptada de [Adami et al. \(2015b\)](#).

Na Figura 21, as latências das requisições do experimento para cada combinação de fatores estão expostas. Novamente, as latências são estatisticamente as mesmas para diferentes níveis de mapeamento. A granularidade não é suficiente para diminuir o tempo das requisições.

Figura 21 – Latência dos níveis de mapeamento, adaptada de [Adami et al. \(2015b\)](#).

Na Figura 22, o número de saltos do algoritmo de busca é ilustrado. Neste caso, a média do número de saltos para 3 níveis de mapeamento é menor que a média para 1 nível

de mapeamento. Pela granularidade ter aumentado, o servidor inicial de busca está ainda mais próximo do cliente, diminuindo assim o número de passos necessários para a convergência do algoritmo.

Figura 22 – Saltos dos níveis de mapeamento, adaptada de [Adami et al. \(2015b\)](#).

Adicionalmente, para todas as variáveis de resposta, a entrada não afetou o comportamento do sistema. O resultado das requisições são os mesmos para ambas as configurações de entrada.

Portanto, a granularidade apenas diminui o número de saltos das requisições. Apesar disso, esse ganho não é suficiente para afirmar que é melhor ter uma maior granularidade de regiões geográficas para o sistema GALA. Uma vez que o número de regiões aumenta, a carga no servidor de mapeamento também aumenta. Para o experimento, haverá 16 vezes mais nomes a serem mapeados. A diminuição no número de saltos resultará em menos comunicação entre os servidores do sistema na busca. Como um *trade-off*, a carga nos servidores de nomes aumentará, por causa do aumento do número de possíveis nomes de domínio.

4.5 O Segundo Experimento

Para o segundo experimento, foram consideradas as trocas entre os algoritmos de busca GALA e GAA. O objetivo do experimento é mostrar o ganho do algoritmo GALA, se comparado com seu antecessor, GAA. Os fatores e níveis do experimento se encontram na Tabela 5.

No gráfico da Figura 23, é possível observar que a eficiência para todas as configurações de experimento se manteve acima de 95%. Analisando os efeitos dessa variável de resposta,

Tabela 5 – Configurações do experimento.

Fator	Níveis	Total
Algoritmo	GAA, GALA	2
Método de Distância	Chebchev, Euclideana	2
Dados de Entrada	PlanetLab e GeoNames	2

na Figura 24, é possível concluir que a entrada é o fator mais influente. O algoritmo também é significativo, sendo que o GAA resulta em eficiências maiores se comparado com o GALA. O método de distância apresenta uma pequena influência positiva, onde o método Euclideano resulta em eficiências melhores se comparado ao método de Chebchev.

Figura 23 – Eficiência: Gráficos de barras, adaptada de [Adami et al. \(2014\)](#).

Analisando a Figura 25, nota-se que o sistema utilizando o algoritmo GAA possui melhores eficiências que o algoritmo GALA. O mesmo pode ser afirmado para o método de distância Euclideana se comparado com o método de Chebchev e para as configurações de entrada GeoNames/PlanetLab em relação à entrada PlanetLab/GeoNames.

Na Figura 26, é possível observar a interação entre os fatores método de distância e entrada. Para todas as outras combinações de fatores, não há interação significativa. Essa significância pode ser observada também no gráfico da Figura 24.

Observando a Figura 27, é evidente que o algoritmo GALA resulta em latências bem menores que o algoritmo GAA. De acordo com o observado na Figura 28, para essa variável de resposta, esse é o único fator que tem muita influência.

Em termos de latência, conforme ilustrado na Figura 29, o único fator cuja variação dos

Figura 24 – Eficiência: Gráficos de efeitos, adaptada de [Adami et al. \(2014\)](#).

Figura 25 – Eficiência: Gráficos de efeitos principais.

níveis apresenta uma diferença aparente nessa variável de resposta é o algoritmo utilizado. Para os outros fatores, a latência obtida permanece praticamente constante.

Na Figura 30, há uma sobreposição das retas na combinação dos fatores algoritmo e entrada. Para todas as outras combinações, não há interação, uma vez que as retas estão paralelas.

Figura 26 – Eficiência: Gráficos de interações.

Figura 27 – Latência: Gráficos de barras, adaptada de [Adami et al. \(2014\)](#).

A sobreposição citada representa uma pequena interação entre os fatores. Esse fato pode ser observado na Figura 28.

Figura 28 – Latência: Gráficos de efeitos, adaptada de [Adami et al. \(2014\)](#).

Figura 29 – Latência: Gráficos de efeitos principais.

Para essa variável de resposta é importante observar que o desvio padrão obtido foi grande. Isso indica que os tempos de requisições estão dispersos das médias observadas. Na Tabela 6, os tempos mínimos e máximos de todas as requisições para cada combinação de fatores estão expostos. De acordo com os dados apresentados, os valores máximos para o algoritmo

Figura 30 – Latência: Gráficos de interações.

Tabela 6 – Latências mínimas e máximas dos experimentos.

Algoritmo	Entrada	Mét. de Distância	Latência Mín. e Máx. (ms)
GAA	GeoNames/PlanetLab	Chebyshev	3 e 494
GALA	GeoNames/PlanetLab	Chebyshev	3 e 254
GAA	GeoNames/PlanetLab	Euclidiana	3 e 494
GALA	GeoNames/PlanetLab	Euclidiana	3 e 227
GAA	PlanetLab/GeoNames	Chebyshev	3 e 559
GALA	PlanetLab/GeoNames	Chebyshev	3 e 397
GAA	PlanetLab/GeoNames	Euclidiana	3 e 553
GALA	PlanetLab/GeoNames	Euclidiana	3 e 383

GALA sempre são inferiores ao GAA. Para o algoritmo GAA, essa variação pode ser explicada pela aleatoriedade da escolha inicial do nó. Há a possibilidade da escolha ser muito próxima do cliente ou muito distante, fazendo com que as latências variem. Para o algoritmo GALA, isso é explicado pelo fato de que há clientes em regiões geográficas onde não há servidores disponíveis. Assim, servidores de regiões vizinhas são utilizados. Portanto, esses clientes afastados geram requisições mais demoradas, aumentando a variabilidade dessa variável de resposta.

A média do número de saltos das requisições se manteve menor para o algoritmo GALA se comparado com o GAA, fato ilustrado na Figura 31. Como exposto na Figura 32, a entrada teve uma influência significativa nessa variável de resposta, além do algoritmo. Isso ocorre porque o servidor inicial das requisições está mais próximo do cliente. Assim, o número de passos necessários para o algoritmo convergir é menor.

Nota-se, na Figura 33, uma diferença no número de saltos considerando os fatores

Figura 33 – Saltos: Gráficos de efeitos.

Na Figura 34, é possível observar uma sobreposição de retas para a combinação dos fatores algoritmo e entrada. Esse comportamento indica que há uma influência da combinação desses fatores no número de saltos, fato observado na Figura 32. Para todas as outras combinações, as retas estão “paralelas”, o que indica a não interação entre os fatores.

Figura 34 – Saltos: Gráficos de interações.

Analisando os resultados apresentados, pode-se concluir que o algoritmo GALA obteve um desempenho geral melhor que o algoritmo GAA. A geolocalização diminuiu o número de passos para encontrar o melhor servidor e, conseqüentemente, diminuiu o tempo das requisições. Ainda, os resultados mostraram eficiências menores quando aplicado o algoritmo GALA, em comparação ao GAA. Porém, deve-se levar em consideração que a entrada é o principal fator que influencia nessa variável de resposta e os valores obtidos não foram tão menores assim. Portanto, é possível afirmar que o algoritmo GALA é melhor que o GAA, considerando o tempo de seleção.

4.6 O Terceiro Experimento

O objetivo do terceiro experimento é mostrar a convergência do algoritmo GALA comparando-o com o algoritmo GAA. A convergência é importante porque quanto mais rápido o algoritmo de busca convergir, mais rápido ele consegue resolver requisições dentro de um tempo limite, respeitando possíveis restrições de tempo. Os fatores utilizados para esse experimento estão expostos na Tabela 7.

Tabela 7 – Fatores do terceiro experimento.

Fator	Níveis
Algoritmo	GAA, GALA
Dados de Entrada	PlanetLab, GeoNames

Para mostrar a convergência, são consideradas as médias do RTT por salto do algoritmo, nas requisições. Adicionalmente, histogramas são utilizados para ilustrar a distribuição dos tempos totais das requisições. Ainda, uma normal é plotada utilizando um histograma cumulativo demonstrando a porcentagem de requisições que já terminaram ao longo do tempo.

As médias de RTT dos experimentos para a primeira e segunda configurações de entrada estão ilustradas nas Figuras 35 e 36, respectivamente.

O termo médias RTT significa a distância do servidor selecionado atual para o cliente no algoritmo de busca. Cada passo é representado no eixo x , enquanto que as médias das requisições estão plotadas no eixo y , com um intervalo de confiança.

Como mostrado nas Figuras, GALA começa seu processo de busca em um nó mais próximo se comparado com o algoritmo GAA. Ambos os algoritmos convergem aproximadamente com o mesmo número de saltos. Porém, se considerado o tempo total de seleção, GALA demora menos milissegundos para convergir, conforme descrito a seguir. Adicionalmente, a mudança nos dados de entrada não afeta o comportamento geral dos algoritmos.

Nas Figuras 37 e 38, o histograma das latências totais das requisições está ilustrado. Na primeira Figura, há o resultado para a primeira configuração de entrada, enquanto que na segunda há o resultado para a segunda configuração de entrada.

Figura 35 – Gráfico de intervalos do RTT, para a primeira configuração de entrada, adaptada de [Adami et al. \(2015b\)](#).

Figura 36 – Gráfico de intervalos do RTT, para a segunda configuração de entrada, adaptada de [Adami et al. \(2015b\)](#).

Na simulação, a execução do serviço propriamente dito não ocorre. Portanto, a latência total compreende o tempo entre o pedido de requisição de um cliente até a resposta do sistema com o melhor nó disponível. O eixo x representa as latências observadas. O eixo y mostra o numero total de requisições que demoraram o tempo observado para selecionar o melhor servidor, em porcentagem.

Figura 37 – Histograma da latência, para a primeira configuração de entrada, adaptada de [Adami et al. \(2015\)](#).

Figura 38 – Histograma da latência, para a segunda configuração de entrada, adaptada de [Adami et al. \(2015\)](#).

É possível observar a aleatoriedade dos tempos de requisição para o algoritmo GAA. Isso ocorre devido à seleção do servidor inicial, a qual é feita de forma aleatória. Por outro lado, as observações do algoritmo GALA tendem a se acumular próximas à origem. Esse comportamento é o esperado, já que GALA tenta utilizar servidores próximos para iniciar o processo de busca. Também, como afirmado anteriormente, a mudança na configuração de entrada não afetou muito as observações de forma geral.

Finalmente, na Figura 39, as frequências acumuladas das latências das requisições estão expostas. As curvas do gráficos são normais ajustadas geradas pelo Minitab. Cada painel mostra as frequências acumuladas para um configuração de entrada. O eixo x representa as latências observadas. O eixo y mostra a quantidade total de requisições que levaram a quantidade observada de tempo ou menos para selecionar o melhor servidor, em porcentagem.

Figura 39 – Histograma acumulado da latência, adaptada de [Adami et al. \(2015\)](#).

Nessa Figura, está evidenciado claramente que GALA alcança sua convergência em muito menos tempo que o GAA. Por exemplo, considerando o painel esquerdo, 100% das requisições já convergiram em 150 milissegundos, enquanto que para o GAA, essa porcentagem é de aproximadamente 30%. Esse fato se repete para o painel da direita. Ainda, a entrada mais uma vez não influenciou o comportamento dos algoritmos de forma significativa.

4.7 Tempo Máximo de Seleção

Acordos de Nível de Serviço (SLAs) são contratos entre o servidor e o cliente envolvendo alguns parâmetros de qualidade de serviço. Por exemplo, o cliente pode requerer limitar o custo de uma requisição de um serviço ou seu tempo de execução máximo.

Nos sistemas de *anycast* na camada de aplicação estudados, os autores limitaram o uso de métricas para selecionar servidores em distância na rede e carga dos servidores. No sistema proposto, um SLA bem conhecido é utilizado, o tempo máximo de seleção de um serviço. Essa métrica é definida pelo cliente e diz a quantidade de tempo que o sistema tem para selecionar um servidor para atender a requisição do cliente.

Essa nova métrica se encaixa bem no sistema proposto pelo fato de que o algoritmo de busca é baseado em iterações. Também, a transparência do sistema alcançada não previne sua utilização. Clientes enviam perguntas *anycast* utilizando nomes de domínio no formato *domain.name.any*. Se um cliente deseja definir um tempo de seleção máximo, a pergunta é feita da seguinte forma: *domain.name.any%omst=1000*. MST é traduzido como tempo de seleção máxima e 1000 significa que o sistema tem um segundo para selecionar o melhor servidor para atender ao cliente.

O resolvidor do sistema processa a *string* de pergunta e identifica o SLA anexado ao nome de domínio. Então, no algoritmo de busca, o resolvidor se propõe a selecionar um servidor antes que o tempo expire. Para isso, um temporizador é inicializado de forma a expirar no MST dado. Quando o tempo expirou, o servidor atual (na iteração atual) é retornado.

O comportamento do sistema GALA utilizando o SLA proposto foi analisado, comparando-o com o GAA. Na Tabela 8, os possíveis tempos de seleção máxima e a porcentagem de requisições atendidas de forma a cumprir o SLA com o melhor servidor disponível estão expostos. Na Tabela, ambas as configurações de entrada são consideradas.

Tabela 8 – Tempos de seleção máxima.

MST (ms)	% de requisições GALA	% de requisições GAA
25	36.1119	4.3399
50	63.0173	11.3483
75	79.9391	17.0272
100	88.5698	20.9199
125	95.5922	24.5248
150	98.6353	27.9213
175	99.7261	32.5492
200	99.8923	40.1592
225	100	48.0782
250	100	57.7364
275	100	70.4354
300	100	81.7650
325	100	91.7416
350	100	97.3455
375	100	99.3375
400	100	99.7917
425	100	99.9415
450	100	99.9883

É importante notar que o melhor servidor conhecido não significa necessariamente o melhor servidor. Nos experimentos anteriores, foi mostrado que ambos GALA e GAA possuem aproximadamente 95% de acurácia. Adicionalmente, se o algoritmo é incapaz de convergir antes que o tempo de seleção se expire, não significa que a seleção falhou. O sistema irá retornar o servidor selecionado atual, que é o melhor para a iteração corrente.

Analisando a Tabela, fica evidente que o GALA é capaz de atender ao SLA do cliente melhor que o GAA. Com um valor de MST 125, ele consegue cumprir com a qualidade de serviço em 95% das requisições, enquanto que o GAA, esse valor é abaixo de 25%. GAA atinge seu estado ótimo com um valor de MST de aproximadamente 350 milissegundos, sendo capaz de atender 99% das requisições de forma adequada. O algoritmo GALA oferece a mesma qualidade com um valor de MST de 175 milissegundos, exatamente metade do tempo para o GAA.

4.8 Considerações Finais

Neste Capítulo, experimentos foram realizados seguindo critérios de avaliação de desempenho. Dentre esses critérios, destacam-se a replicação de experimentos, a variação de fatores e o cálculo de intervalos de confiança. Os resultados apresentados nesta Seção foram coletados por meio dos *logs* gerados por uma simulação, descrita em detalhes, e gráficos foram criados por meio do *software* Minitab. Os resultados mostraram que o algoritmo GALA obteve melhores resultados que o algoritmo GAA.

PROTÓTIPO DO SISTEMA

5.1 Considerações Iniciais

Para fortalecer a validação do sistema modelado, que foi testado por meio de simulações, GALA foi lançado em um ambiente real. O objetivo do sistema real é avaliar o comportamento geral da arquitetura e discutir aspectos não observados na simulação do sistema de *anycast* na camada de aplicação. O principal objetivo da experimentação é responder à hipótese de que com a seleção de nós iniciais mais próximos dos clientes, utilizando a geolocalização, o desempenho geral do sistema deve melhorar. Para a configuração dos experimentos, considerou-se utilizar o fatorial completo com fatores variando entre dois níveis cada. A análise dos resultados foi feita utilizando o modelo de regressão linear e os gráficos foram gerados pelo software Minitab ([Minitab Inc, 2013](#)).

O gráfico de barra apresentado (Figura 41) tem como função sumarizar a distribuição de uma variável de resposta, mostrando sua tendência central e sua variabilidade.

5.2 Visão Geral do Sistema

Na Figura 40, detalhes do sistema GALA lançado em um ambiente real são expostos. Todos os módulos estão presentes. São eles: o cliente, o resolvidor, o publicador (nesse caso, um servidor DNS) e os servidores. Como discutido anteriormente, para lidar com uma requisição *anycast*, 6 passos são necessários: (1) procura de um resolvidor, (2) envio da pergunta *anycast* ao resolvidor, (3) identificação do nome *anycast*, mapeamento da localização do cliente, identificação e listagem dos servidores da região geográfica, (4) início do algoritmo de busca com um servidor inicial, (5) escolha do servidor na convergência do algoritmo e (6) recebimento da requisição pelo servidor selecionado.

Figura 40 – Visão geral do sistema GALA real, adaptada de [Adami et al. \(2015b\)](#).

5.3 Balanceamento de Carga

As cargas nos servidores não foram consideradas nos experimentos simulados. Em um ambiente real, o balanceamento de carga é um aspecto que não pode ser ignorado, uma vez que ele está diretamente relacionado ao desempenho dos servidores.

Para incluir consciência de carga no sistema de *anycast* na camada de aplicação, o

algoritmo de busca foi modificado. O resolvedor, ao consultar o RTT de um servidor, também lê sua carga. O valor de carga de um nó é obtido por meio do comando Linux *uptime*. Esse comando mostra a média de carga do sistema no decorrer do último minuto. A média de carga do sistema é a o número médio de processos que estão no estado “executando” ou “não-interrompível”. Um processo no estado “executando” está utilizando a CPU ou aguardando para utilizá-la. Um processo no estado “não-interrompível” está esperando por um acesso de entrada/saída (GREENFIELD; JOHNSON, 1993a). Tipicamente, para uma máquina com uma CPU, esse valor varia entre 0 e 1. Nos experimentos, todos os nós possuem 1 núcleo.

No processo de busca, o algoritmo mantém uma lista dos servidores selecionados em cada iteração. Ao terminar a busca, essa lista contém os servidores e seus respectivos RTTs e valores de carga, ordenados por RTT. Então, o algoritmo, de forma iterativa, procura pelo servidor mais próximo tal qual a carga não ultrapasse um limiar estabelecido como uma variável do ambiente. Se nenhum servidor satisfaz essa restrição, um elemento da lista é selecionado de forma aleatória.

Portanto, nenhum servidor sobrecarregado será selecionado para atender à requisição de um cliente, aumentando a capacidade do servidor de atender clientes e oferecer uma melhor qualidade de serviço. Isso é verdade a não ser que todos os servidores estejam sobrecarregados. Esse cenário ocorre quando o sistema como um todo não é capaz de atender à demanda dos clientes. Nessa situação, novos servidores devem ser adicionados ao sistema.

5.4 Detalhes de Implementação

Preza-se pela transparência do sistema aos clientes. Portanto, foram feitas o mínimo de modificações no lado do cliente. O cliente executa uma distribuição Linux Ubuntu e invoca os serviços por meio da linha de comando. A única mudança necessária é a adição do endereço IP do resolvedor GALA no arquivo “/etc/resolv/conf”, o que fará com que as perguntas de nome de domínio sejam enviadas ao resolvedor *anycast*. Para buscar nomes *anycast*, basta adicionar “.any” no final de um nome de domínio.

O resolvedor consiste de um *daemon*, implementado utilizando C++, que recebe pacotes de consulta DNS de acordo com o RFC 1035 (P. Mockapetris, 1987), identifica nomes de domínio, repassa resoluções que não são nomes *anycast* e responde perguntas com formato *anycast*. É importante ressaltar que o processo de busca é centralizado no resolvedor, para manter a transparência no cliente. Por esse motivo, cada cliente possui um resolvedor correspondente em sua região. Assim, o processo de busca terá o resultado muito próximo de uma busca realizada no próprio cliente.

O publicador é um nó executando o *software* BIND. BIND é uma implementação dos protocolos de sistema de domínio BIND (Internet Systems Consortium, 2014).

O servidor oferece páginas HTML como serviço. Ele executa o *software* Nginx (Nginx, 2014). Para os experimentos, uma única página HTML foi utilizada. O servidor também executa um *daemon* GALA, implementado em C++, o qual é responsável por registrar o servidor no DNS, atualizar e podar a vizinhança e responder à consultas dos resolvedores. Toda comunicação entre servidores e resolvedores é feita por meio de UDP.

5.5 Detalhes do Ambiente

Para implantar o sistema de *anycast*, o serviço de nuvem DigitalOcean¹ foi utilizado. Esse serviço oferece servidores privados virtuais (VPS) ao seus clientes. Todos os VPS utilizados possuem 512MB de RAM, 1 núcleo, 20GB SSD de disco rígido e executam o sistema operacional Ubuntu 14.04 LTS.

Foram lançados 21 nós em 5 regiões geográficas distintas. São elas Nova Iorque, São Francisco, Londres, Amsterdã e Singapura. Cada região possui dois servidores, um cliente e um resolvidor. Adicionalmente, um nó DNS foi implantado em Nova Iorque.

Tabela 9 – Mapeamento dos servidores

Endereço IP	Mapeamento Geográfico	Mapeamento Estático	X,Y
178.62.***.***	Rússia	Grã Bretanha	3,5
178.62.***.***	Rússia	Grã Bretanha	3,5
178.62.***.***	Rússia	Grã Bretanha	3,5
178.62.***.***	Rússia	Grã Bretanha	3,5
104.131.***.***	X	Estados Unidos	2,5
104.131.***.***	X	Estados Unidos	2,5
104.131.***.***	X	Estados Unidos	2,5
104.131.***.***	X	Estados Unidos	2,5
107.170.***.***	X	Estados Unidos	2,5
107.170.***.***	X	Estados Unidos	2,5
162.243.***.***	X	Estados Unidos	2,5
104.131.***.***	X	Estados Unidos	2,5
178.62.***.***	Rússia	Holanda	4,5
178.62.***.***	Rússia	Holanda	4,5
178.62.***.***	Rússia	Holanda	4,5
178.62.***.***	Rússia	Holanda	4,5
128.199.***.***	Grã Bretanha	Singapura	6,4
128.199.***.***	Grã Bretanha	Singapura	6,4
128.199.***.***	Grã Bretanha	Singapura	6,4
128.199.***.***	Grã Bretanha	Singapura	6,4

Na Tabela 9, os resultados de mapeamento estão expostos, considerando os nós implantados. Foram utilizadas 64 regiões geográficas e o banco de dados de geolocalização MaxMind de 2013. Pelo fato do banco de dados não fornecer resultados corretos para os endereços IP

¹ <http://www.digitalocean.com>

utilizados, eles foram mapeados de forma estática. A primeira coluna indica o endereço IP de um servidor do sistema. A segunda coluna corresponde ao mapeamento do banco de dados MaxMind. “X” significa que não havia um intervalo de endereços IP que o endereço em questão se encaixava. A terceira coluna mostra o mapeamento correto, utilizado pelo sistema. A quarta coluna indica qual o nome de domínio que o endereço correspondente foi mapeado pelo sistema.

5.6 Avaliação de Desempenho

Para validar o sistema implantado, experimentos foram feitos utilizando o ambiente descrito a seguir. Na Tabela 10, os fatores e níveis considerados estão expostos. O fator requisições por cliente representa o número de requisições por cliente a cada segundo. O tempo total do experimento é de um minuto. O fator de consciência de carga é utilizado para analisar o desempenho do sistema considerando a métrica de carga ou ignorando-a.

Adicionalmente, duas variáveis de resposta foram analisadas: os tempos de busca e serviço. O tempo de busca é o tempo entre o cliente realizar a pergunta *anycast* e o resolvidor GALA encontrar um servidor adequado para atender à requisição do cliente. O tempo de serviço envolve a invocação do serviço no servidor. Considerando os passos mostrados na Figura 40, o tempo de busca compreende os quatro primeiros passos, enquanto que o tempo de serviço compreende os dois últimos.

Para coletar essas variáveis de resposta, nos clientes, há um *script* que chama a ferramenta *dig* (CONSORTIUM, 2000). Depois, com o resultado obtido, o comando *wget* (NIKSIC, 1996) é invocado. Para ambos comandos, o programa de medidas de tempo do sistema, *time* (MACKENZIE, 19XX), é utilizado.

Tabela 10 – Configurações de experimento.

Fator	Níveis
Algoritmo	GAA, GALA
Consciência de Carga	Sim, Não
Requisições por Cliente (por segundo)	30, 50

Na Tabela 11, os parâmetros de ambiente para os experimentos estão expostos. Em cada região geográfica, há um cliente, dois servidores e um resolvidor GALA. Também, há um único servidor DNS em Nova Iorque. Os outros parâmetros são variáveis do sistema e são as mesmas utilizadas nas simulações. Adicionalmente, as execuções do experimento foram replicadas 6 vezes, uma vez que esse número de replicações foi suficiente para obter resultados satisfatórios.

Na Figura 41, os resultados para o experimento real estão ilustrados. O eixo *x* representa as combinações de fatores do experimento e o eixo *y* representa as médias dos tempos de serviço e busca em milissegundos.

Tabela 11 – Parâmetros do experimento.

Parâmetro	Valor
Clientes	5
Servidores	10
Resolvedores	5
Servidores DNS	1
Nós por anel (k)	8
Nós secundários (l)	2
Anéis (R)	11
a	10
b	0.5
s	2
Regiões geográficas	64
Replicações	6

Figura 41 – Resultados para os tempos de busca e serviço, adaptada de [Adami et al. \(2015b\)](#).

Para os tempos de busca, é possível notar que a busca do algoritmo GALA é mais rápida que o algoritmo GAA. Esse comportamento condiz com o observado nas simulações. Como afirmado anteriormente, GALA converge mais rápido para um servidor adequado, uma vez que ele começa sua busca em um servidor próximo do cliente. Adicionalmente, os tempos de busca aumentaram com o aumento de carga, como esperado. Ainda, a consciência de carga não afetou a busca de forma significativa.

Para os tempos de serviço, o sistema utilizando o algoritmo GALA pôde executar as

tarefas em menos tempo, se comparado com o GAA. Isso ocorre porque o sistema demora mais para selecionar servidores no segundo algoritmo, fazendo com que as requisições acumulem e os servidores fiquem mais carregados executando mais trabalho concorrentemente. Também, é importante notar que o aumento da carga também aumentou os tempos de serviço, um comportamento esperado. Considerando a consciência de carga, sua presença teve uma influência negativa para a carga menor. Por outro lado, para a carga maior, ela teve um influencia positiva. O resultado negativo é explicado pelo fato de que a sobrecarga causada pela leitura de carga do sistema é maior que o ganho com a consciência da carga para esse caso. Em um trabalho futuro, é possível tornar o sistema adaptável e escolher utilizar a carga de forma automática em casos onde isso seja vantajoso.

5.7 Considerações Finais

Os resultados apresentados nesta Seção foram coletados por meio da execução de experimentos em um sistema real, descrito em detalhes, e gráficos foram criados por meio do *software* Minitab. Os resultados mostraram que o algoritmo GALA obteve melhores resultados que o algoritmo GAA, assim como observado anteriormente nas simulações. Esses resultados fortalecem a afirmação de que o sistema GALA supera o sistema GAA como um sistema de *anycast* na camada de aplicação.

PUBLICAÇÕES

6.1 Considerações Iniciais

Com os resultados obtidos deste trabalho de mestrado, artigos foram produzidos, submetidos e publicados. Eles podem ser divididos em duas categorias: artigos que descrevem os resultados das simulações e artigos que descrevem os resultados das execuções do ambiente real.

6.2 Trabalhos Publicados

Com os resultados obtidos por meio de simulação, foram publicados dois artigos. Ambos são destacados a seguir:

Adami, J. L.; Estrella, C. J.; de Oliveira, M. E.; Reiff-Marganiec S.; **Providing Quality of Service on Services Selection using Anycast Techniques**. In: 11th IEEE International Conference on Services Computing, 2014, DOI: 10.1109/SCC.2014.46, Qualis B1.

Adami, J. L.; Estrella, C. J.; Libardi, R.; Reiff-Marganiec S.; **Quality of Service on Services Selection using Anycast Techniques: a Convergence Analysis**. In: 14th IFIP/IEEE Symposium on Integrated Network and Service Management, 2015, DOI: 10.1109/INM.2015.7140381, Qualis A2.

6.3 Trabalhos Submetidos

Com os resultados obtidos com a implantação do sistema real, foi submetido um novo artigo para a Revista PLOS One.

Adami, J. L.; Estrella, C. J.; Libardi, R.; de Oliveira, M. E.; Reiff-Marganiec S.; **GALA: An Application Anycast System for Distributed Environments**. In: PLOS One Journal, 2015, submetido, Qualis A1.

Essa revista é uma mídia internacional que é meio de publicação para tópicos multidisciplinares, incluindo a área de redes de computadores.

6.4 Considerações Finais

Ao todo, foram submetidos três artigos. Dois deles descrevem as simulações realizadas enquanto que o terceiro descreve os resultados obtidos do ambiente real. Os dois primeiros foram publicados enquanto que o terceiro artigo está em fase de análise.

CONCLUSÕES

7.1 Considerações Iniciais

As etapas do projeto de pesquisa foram desenvolvidas e o cronograma previsto foi respeitado, seguindo conforme o planejado inicialmente. As próximas seções descrevem os resultados finais obtidos e as contribuições deste trabalho.

7.2 Conclusões

O experimentos realizados no Capítulo 4 tiveram como objetivo mostrar os ganhos do algoritmo GALA, se comparado com seu antecessor, GAA. Nas simulações, foram analisados os tempos de busca, número de saltos e eficiência da seleção. Ainda, foi analisado como a granularidade das regiões geográficas afetam o desempenho do sistema. Também, foi mostrado a convergência do algoritmo GALA comparando-o com o algoritmo GAA.

Uma vez que as simulações indicavam um ganho no algoritmo *anycast*, o sistema real foi implantado e os experimentos do Capítulo 5 foram realizados para estudar o comportamento do sistema de *anycast* na camada de aplicação, fortalecendo a validação do sistema modelado e simulado. Adicionalmente, foram discutidos aspectos não observados na simulação do sistema de *anycast* na camada de aplicação, como a influência da carga nos nós.

Analisando os resultados apresentados no Capítulo 4 e 5, pode-se concluir que o algoritmo GALA obteve um desempenho geral melhor que o algoritmo GAA. A geolocalização diminuiu o número de passos para encontrar o melhor servidor e, conseqüentemente, diminuiu o tempo das requisições. Além disso, outro fato que contribuiu para a diminuição da latência das requisições é o algoritmo GALA sempre iniciar sua busca em uma região próxima do cliente, ondes os RTTs das requisições são pequenos. No algoritmo GAA, isso pode não ocorrer. Há a possibilidade do nó inicial estar muito distante do cliente. Nesse caso, a consulta em um servidor distante resulta

em tempos elevados. Ainda, os resultados mostraram eficiências menores quando aplicado o algoritmo GALA, em comparação ao GAA. Porém, deve-se levar em consideração que a entrada é o principal fator que influencia nessa variável de resposta e os valores obtidos não foram tão menores assim. Portanto, é possível afirmar que o algoritmo proposto avaliado traz um ganho significativo ao sistema.

Concluindo, a principal contribuição científica deste trabalho é a proposta, modelagem, simulação e implantação de um sistema *anycast* na camada de aplicação. Ainda, incluem a análise de sua acurácia, velocidade de seleção de servidores e convergência. Finalmente, soma-se a validação dos resultados simulados, mostrando o ganho em relação ao sistema existente.

7.3 Trabalhos Futuros

Vale a pena ressaltar que, com o desenvolvimento deste trabalho, novas oportunidades de pesquisa foram criadas e estão descritas na listagem abaixo. Adicionalmente, um trabalho derivado deste projeto já está sendo desenvolvido: uma ferramenta *web* que auxilia na elaboração de experimentos simulados. Sua interface permite visualizar os resultados em forma de gráficos e tabelas. Assim, a configuração e execução de experimentos se torna mais fácil, rápida e flexível. Esse trabalho está sendo desenvolvido pelo aluno de iniciação científica Henrique Pizzol Grando, cujo processo FAPESP é 2014/11080-8.

- A geolocalização utilizada não foi acurada o suficiente a fim de fornecer bons resultados à seleção de servidores. Novas formas de geolocalização e mapeamento dos nós podem ser exploradas, aumentando a confiabilidade do sistema.
- O sistema real implantado limitou-se a uma quantidade pequena de máquinas e regiões geográficas. Muitos cenários ainda podem ser explorados em ambientes maiores.
- Considera-se a adição de novas métricas ao sistema, como classes de cliente. Este item inclui a adição automática de métricas conforme características específicas do ambiente.
- Todos os experimentos realizados foram feitos de forma manual. Uma automação desse processo é possível e interessante de ser explorada, assim como foi feito para os experimentos simulados em um trabalho de iniciação científica.

7.4 Etapas Cumpridas

A seguir, as principais atividades previstas para o programa de mestrado são descritas. Na Tabela 12, a organização desses itens durante o tempo está ilustrada e os itens devidamente destacados.

Tabela 12 – Organização das atividades realizadas.

Fases	Período (por bimestres)													
	2013					2014						2015		
	2	3	4	5	6	1	2	3	4	5	6	1	2	3
1	✓	✓	✓	✓	✓									
2	✓													
3	✓	✓	✓	✓	✓	✓	✓	✓				✓	✓	
4		✓	✓	✓	✓	✓								
5		✓	✓	✓	✓	✓	✓	✓				✓	✓	
6						✓	✓	✓	✓	✓	✓	✓	✓	✓
7									✓	✓	✓			
8						✓	✓	✓				✓	✓	✓

1. **Cumprimento de créditos referente às disciplinas do mestrado:** Para realizar a defesa da dissertação foi necessário cumprir um total de 48 créditos durante os dois semestres de 2013;
2. **Preparação para o exame de proficiência em inglês:** Essa etapa compreende o período de preparação para obter o certificado de proficiência em língua inglesa;
3. **Análise crítica dos trabalhos relacionados:** As atividades iniciais foram focadas na análise crítica dos trabalhos relacionados à arquiteturas que utilizam *anycast* na camada de aplicação, planejamento de experimentos e avaliação de desempenho. A continuidade da análise bibliográfica foi de suma importância durante todo o período de trabalho, de maneira que se agreguem novas metodologias ao projeto de pesquisa;
4. **Preparação para o exame de qualificação:** Essa fase inclui a preparação para o exame de qualificação que deve ser realizado no início do primeiro semestre de 2014;
5. **Desenvolvimento da proposta:** Essa fase envolveu o desenvolvimento de todos os componentes do sistema. Decisões de projeto mais específicas foram tomadas nessa etapa. Essas decisões envolvem desde questões de linguagem até a estrutura da comunicação entre os servidores na rede;
6. **Elaboração de artigos científicos:** Preparação de artigos que promovam a divulgação dos resultados alcançados e a apresentação das contribuições científicas;
7. **BEPE:** Permanência do candidato em um estágio de pesquisa no exterior (BEPE) com período de 4 meses e meio;
8. **Redação da dissertação:** Fase da escrita da dissertação de mestrado, que deve ser entregue para defesa até o final de agosto de 2015.

7.5 Considerações Finais

Este Capítulo apresentou o cronograma e as etapas desenvolvidas do trabalho. Todo o projeto seguiu o cronograma calculado e este foi executado com êxito. Adicionalmente, as conclusões, principais contribuições e trabalhos futuros foram apresentados.

REFERÊNCIAS

ADAMI, L. J.; ESTRELLA, J. C.; LIBARDI, R. M. de O.; REIFF-MARGANIEC, S. Quality of Service on Services Selection using Anycast Techniques: a Convergence Analysis. In: **IEEE IM, 2015. 14th IFIP/IEEE Symposium on Integrated Network and Service Management**. [S.l.: s.n.], 2015. Citado 5 vezes nas páginas 11, 12, 38, 64 e 65.

ADAMI, L. J.; ESTRELLA, J. C.; OLIVEIRA, E. M. de; REIFF-MARGANIEC, S. Providing Quality of Service on Services Selection using Anycast Techniques. In: **IEEE SCC, 2014. 11th IEEE International Conference on Services Computing**. [S.l.: s.n.], 2014. Citado 8 vezes nas páginas 11, 12, 51, 55, 56, 57, 58 e 60.

ADAMI, L. J.; ESTRELLA, J. C.; OLIVEIRA, E. M. de; REIFF-MARGANIEC, S.; LIBARDI, R. M. de O.; OLIVEIRA, E. M. de. GALA: An Application Anycast System for Distributed Environments. In: **PLOS One**. [S.l.: s.n.], 2015b. Citado 9 vezes nas páginas 11, 12, 34, 52, 53, 54, 63, 70 e 74.

BHATTACHARJEE, S.; AMMAR, M.; ZEGURA, E.; SHAH, V.; FEI, Z. Application-layer anycasting. In: **INFOCOM '97. Sixteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Driving the Information Revolution., Proceedings IEEE**. [S.l.: s.n.], 1997. v. 3, p. 1388–1396. Citado na página 17.

CHELLOUCHE, S.; NEGRU, D.; BORCOCI, E.; LEBARS, E. Anycast-based context-aware server selection strategy for VoD services. In: **GLOBECOM Workshops (GC Wkshps), 2010 IEEE**. [S.l.: s.n.], 2010. p. 1513–1517. Citado na página 17.

_____. Context-aware distributed multimedia provisioning based on anycast model towards Future Media Internet. In: **Consumer Communications and Networking Conference (CCNC), 2011 IEEE**. [S.l.: s.n.], 2011. p. 880–885. Citado 3 vezes nas páginas 11, 19 e 20.

CHEN, Y.; XIONG, Y.; SHI, X.; ZHU, J.; DENG, B.; LI, X. Pharos: accurate and decentralised network coordinate system. **Communications, IET**, v. 3, n. 4, p. 539–548, 2009. Citado na página 27.

CIA. **CIA World Factbook**. 2013. <https://www.cia.gov/library/publications/the-world-factbook/>. Acessado em 08/08/2013. Citado na página 38.

CONSORTIUM, I. S. **Linux User's Manual - dig**. 2000. Citado na página 73.

Donald E. Eastlake 3rd. **Secure Domain Name System Dynamic Update**. 1997. RFC 2137. Citado na página 40.

ESTRELLA, J. C.; SANTANA, R. H. C.; SANTANA, M. J. **WSARCH: An Architecture for Web Services Provisioning with QoS Support - Performance Challenges**. [S.l.]: Saarbrücken : VDM Verlag Dr. Müller GmbH & Co, 2011. <http://www.amazon.com/WSARCH-Architecture-Provisioning-Performance-Challenges/dp/3639378245>. Citado 2 vezes nas páginas 23 e 24.

- FEI, Z.; BHATTACHARJEE, S.; ZEGURA, E.; AMMAR, M. A novel server selection technique for improving the response time of a replicated service. In: **INFOCOM '98. Seventeenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE**. [S.l.: s.n.], 1998. v. 2, p. 783–791. Citado na página 17.
- FREEDMAN, M. J.; LAKSHMINARAYANAN, K.; MAZIERES, D. OASIS: anycast for any service. In: **Proceedings of the 3rd conference on Networked Systems Design & Implementation - Volume 3**. Berkeley, CA, USA: USENIX Association, 2006. (NSDI'06), p. 10–10. Citado 3 vezes nas páginas 11, 26 e 36.
- GREENFIELD, L.; JOHNSON, M. K. **Linux uptime command**. 1993. Linux manual. Citado na página 71.
- _____. **Linux User's Manual - uptime**. 1993. Citado na página 23.
- HUFFAKER, B.; FOMENKOV, M.; CLAFFY, K. Geocompare: a comparison of public and commercial geolocation databases. May 2011. Citado 2 vezes nas páginas 36 e 50.
- Internet Systems Consortium. **BIND**. 2014. Accessed in 11/12/2014. Citado na página 71.
- ISO 3166. **ISO 3166 Codes (Countries)**. 2013. "http://www.iso.org/iso/country_codes/iso_3166_code_lists/country_names_and_code_elements.htm". Acessado em 08/08/2013. Citado 2 vezes nas páginas 37 e 38.
- JIN, C. Z. M. An anycast routing algorithm based on genetic algorithm. In: **WSEAS TRANSACTIONS on COMPUTERS**. [S.l.: s.n.], 2009. Citado na página 24.
- JONES, B. **The World**. 1998. "http://cse.ssl.berkeley.edu/segwayed/lessons/search_ice_snow/ski.2b1BigMap.html". Acessado em 02/10/2013. Citado 2 vezes nas páginas 11 e 40.
- LATIF L; MYKONIATI, E. L. R. C. R. G. D. R. M. Multi recipient anycast routing. In: **University College London**. [S.l.: s.n.], 2008. Citado na página 25.
- MA, Z.-Y.; ZHOU, J.; ZHANG, L. A Scalable Framework for Global Application Anycast. In: **Information and Computing Science, 2009. ICIC '09. Second International Conference on**. [S.l.: s.n.], 2009. v. 1, p. 250–253. Citado 7 vezes nas páginas 11, 19, 28, 29, 30, 44 e 50.
- MACKENZIE, D. **Linux User's Manual - time**. 19XX. Citado na página 73.
- Marc Wick. **GeoNames**. 2013. [Http://download.geonames.org/export/dump/](http://download.geonames.org/export/dump/). Acessado em 13/11/2013. Citado na página 51.
- Minitab Inc. **Software for Statistics, Process Improvement, Six Sigma, Quality - Minitab**. 2013. "<http://www.minitab.com>". Acessado em 06/12/2013. Citado 2 vezes nas páginas 47 e 69.
- MYKONIATI, E.; LATIF, L.; LANDA, R.; YANG, B.; CLEGG, R.; GRIFFIN, D.; RIO, M. Distributed overlay anycast tables using space filling curves. In: **INFOCOM Workshops 2009, IEEE**. [S.l.: s.n.], 2009. p. 1–6. Citado 5 vezes nas páginas 11, 18, 27, 28 e 39.
- Nginx. **Nginx Website**. 2014. Accessed in 11/12/2014. Citado na página 72.
- NIKSIC, H. **Linux User's Manual - wget**. 1996. Citado na página 73.
- P. Mockapetris. **Domain Names - Implementation and Specification**. 1987. RFC 1035. Citado 2 vezes nas páginas 35 e 71.

- PADMANABHAN, V. N.; SUBRAMANIAN, L. An investigation of geographic mapping techniques for internet hosts. **SIGCOMM Comput. Commun. Rev.**, ACM, New York, NY, USA, v. 31, n. 4, p. 173–185, aug 2001. ISSN 0146-4833. Citado na página 35.
- PARTRIDGE, T. M. C.; MILLIKEN, W. **Host Anycasting Service**. 1993. RFC 1546. Citado na página 17.
- PIPER, P. H. **Anycast DNS - Part 1, Overview**. 2013. [Http://netlinxinc.com/netlinx-blog/45-dns/118-introduction-to-anycast-dns.html](http://netlinxinc.com/netlinx-blog/45-dns/118-introduction-to-anycast-dns.html). Acessado em 12/02/2013. Citado 2 vezes nas páginas 11 e 18.
- Princeton University. **PlanetLab**. 2012. [Https://www.planet-lab.org/](https://www.planet-lab.org/). Acessado em 09/03/2013. Citado na página 51.
- QIN, X.; TAOSHEN, L.; ZHIHUI, G. A qos anycast routing algorithm based on genetic algorithm and particle swarm optimization. In: **Genetic and Evolutionary Computing, 2009. WGECC '09. 3rd International Conference on**. [S.l.: s.n.], 2009. p. 125–128. Citado na página 24.
- SHAVITT, Y.; ZILBERMAN, N. A geolocation databases study. **Selected Areas in Communications, IEEE Journal on**, v. 29, n. 10, p. 2044–2056, 2011. Citado na página 36.
- Stichting NLnet. **BIND DLZ**. 2002. Acessado em 07/10/2013. Citado na página 40.
- VIXIE, P.; THOMSON, S.; REKHTER, Y.; BOUND, J. **RFC 2136**. 1997. Acessado em 07/10/2013. Citado na página 40.
- WALKOWIAK, K.; KLINKOWSKI, M. Joint anycast and unicast routing for elastic optical networks: Modeling and optimization. In: **Communications (ICC), 2013 IEEE International Conference on**. [S.l.: s.n.], 2013. p. 3909–3914. Citado na página 25.
- WANG, G.; CHEN, Y.; SHI, L.; LUA, E. K.; DENG, B.; LI, X. Proxima: Towards Lightweight and Flexible Anycast Service. In: **INFOCOM Workshops 2009, IEEE**. [S.l.: s.n.], 2009. p. 1–2. Citado 2 vezes nas páginas 11 e 27.
- Weisstein, Eric W. **Wolfram MathWorld - Azimuthal Equidistant Projection**. 2015. [Http://mathworld.wolfram.com/AzimuthalEquidistantProjection.html](http://mathworld.wolfram.com/AzimuthalEquidistantProjection.html). Acessado em 23/05/2015. Citado na página 39.
- WENDELL, P.; JIANG, J. W.; FREEDMAN, M. J.; REXFORD, J. Donar: decentralized server selection for cloud services. **SIGCOMM Comput. Commun. Rev.**, ACM, New York, NY, USA, v. 41, n. 4, p. –, ago. 2010. ISSN 0146-4833. Citado na página 24.
- WONG, B.; SLIVKINS, A.; SIRER, E. G. **Meridian: a lightweight network location service without virtual coordinates**. New York, NY, USA: ACM, 2005. 85–96 p. (SIGCOMM '05). Citado 5 vezes nas páginas 28, 29, 43, 44 e 50.
- XIAONAN, W. A scheme on solving anycast scalability on application layer in ipv6. In: **Challenges in Environmental Science and Computer Engineering (CESCE), 2010 International Conference on**. [S.l.: s.n.], 2010. v. 1, p. 118–120. Citado na página 25.
- XING, C.; CHEN, M. Impact of network topology on distance prediction accuracy. In: **Networking, Sensing and Control, 2008. ICNSC 2008. IEEE International Conference on**. [S.l.: s.n.], 2008. p. 1425–1429. Citado na página 28.

YUAN, S.; LIN, T.; ZHANG, G.; LI, Y.; TANG, H.; CI, S. A future anycast routing scheme for information-centric network. In: **Communications (APCC), 2012 18th Asia-Pacific Conference on**. [S.l.: s.n.], 2012. p. 173–178. Citado 3 vezes nas páginas 11, 28 e 29.

ZEGURA, E.; AMMAR, M.; FEI, Z.; BHATTACHARJEE, S. Application-layer anycasting: a server selection architecture and use in a replicated Web service. **Networking, IEEE/ACM Transactions on**, v. 8, n. 4, p. 455–466, 2000. Citado 3 vezes nas páginas 11, 17 e 19.

ZHANG, R.; TANG, C.; HU, Y.; FAHMY, S.; LIN, X. Impact of the inaccuracy of distance prediction algorithms on internet applications - an analytical and comparative study. In: **INFOCOM 2006. 25th IEEE International Conference on Computer Communications. Proceedings**. [S.l.: s.n.], 2006. p. 1–12. Citado 2 vezes nas páginas 26 e 28.