WALTER AKIO GOYA

UMA SOLUÇÃO PARA O DESENVOLVIMENTO DE APLICAÇÕES DISTRIBUÍDAS VISANDO O GERENCIAMENTO AUTOMÁTICO DE RECURSOS NO CENÁRIO DE COMPUTAÇÃO EM NUVEM

Dissertação apresentada à Escola Politécnica da Universidade de São Paulo para obtenção do titulo de Mestre em Ciências

São Paulo

2014

WALTER AKIO GOYA

UMA SOLUÇÃO PARA O DESENVOLVIMENTO DE APLICAÇÕES DISTRIBUÍDAS VISANDO O GERENCIAMENTO AUTOMÁTICO DE RECURSOS NO CENÁRIO DE COMPUTAÇÃO EM NUVEM

Dissertação apresentada à Escola Politécnica da Universidade de São Paulo para obtenção do Título de Mestre em Ciências

Área de Concentração:
Engenharia da Computação

Orientador:
Prof. Dr. Marcos Antonio Simplicio Junior

São Paulo

2014

Catalogação-na-publicação

Goya, Walter Akio
Uma solução para o desenvolvimento de aplicações distribuídas visando o gerenciamento automático de recursos no cenário de computação em nuvem / W.A. Goya. -- São Paulo, 2014.
111 p.

Dissertação (Mestrado) - Escola Politécnica da Universidade de São Paulo. Departamento de Engenharia de Computação e Sistemas Digitais.

1.Computação em nuvem (Gerenciamento; Automação) I.Universidade de São Paulo. Escola Politécnica. Departamento de Engenharia de Computação e Sistemas Digitais II.t.

AGRADECIMENTOS

Ao professor Marcos Antonio Simplicio Junior pela paciência, disponibilidade e inspiração durante todo o processo de orientação. Muito obrigado por seu exemplo como professor, pesquisador e, principalmente, como pessoa.

À professora Tereza Cristina Melo de Brito Carvalho pela oportunidade de trabalhar nos projetos de pesquisa da Ericsson e da Petrobrás, e pelos sábios conselhos.

Aos meus pais, Kinuyo Nakazone e Walter Goya, pelos ensinamentos, pelo incentivo aos estudos, pelo carinho e dedicação ao longo de toda minha jornada (do início até o dia de hoje).

Aos professores e colegas do LARC (Laboratório de Arquitetura e Redes de Computadores), em especial aos amigos Charles Miers e Vlad Coroama (pela ajuda nos momentos difíceis, pelos conselhos e inspiração – noite do *pesto*), a Rony Sakuragui (pelo suporte nos momentos de dificuldade), aos amigos Pedro Evangelista, Marcos Schwarz, Nelson Mimura e Ewerton Andrade (pelas discussões científicas e não científicas, e pela companhia durante a caminhada do mestrado), a Karen Langona (pela liderança e inspiração como coordenadora e pessoa), Rosangela Pereira, Lucas Chigami, Selma, Fernando, Edivaldo, Lúcio, Fernando Almeida, Marcelo, Marco, Marco Aurélio, Bruno e professoras Regina e Cíntia.

Ao Centro de Inovação da Ericsson Telecomunicações S.A., Brasil, pelo suporte financeiro (através da FDTE) e experiência em pesquisa orientada ao mercado. Também minha gratidão à Maria Valéria Marquezini, Stefan Hellkvist, Joacim Halen, Jan-Erik Mangs, Victor Souza, Bob Melander e Per Karlsson.

À FDTE (Fundação para o Desenvolvimento Tecnológico da Engenharia) pelo suporte financeiro e apoio organizacional em todos os projetos que estive envolvido. Agradeço especialmente à Edilaine Lemos, que sempre buscou auxiliar de todas as maneiras possíveis a viabilização dos recursos envolvidos nos projetos.

E a todos que colaboraram direta ou indiretamente, na execução desta pesquisa.

RESUMO

Na segunda metade dos anos 2000, foram desenvolvidos projetos de pesquisa para o desenvolvimento de plataformas visando facilitar a criação de aplicativos para o ambiente de nuvem. A partir de estudos sobre as soluções de elasticidade para nuvens de computação desenvolvidas, observou-se a concentração de soluções de elasticidade com foco no gerenciamento de recursos de processamento e armazenamento para aplicações do tipo cliente-servidor. Porém, no caso das aplicações de distribuição de conteúdo, os recursos de rede que são limitados e também devem ser gerenciados de forma a evitar desperdícios. Devido a estas características, é interessante o desenvolvimento de uma plataforma aberta para a criação de aplicações distribuídas que auxiliem o gerenciamento de recursos e elasticidade no contexto de computação em nuvem. Esta dissertação apresenta o Trade Wind, uma solução que permite o desenvolvimento de aplicações e serviços distribuídos para o gerenciamento automático de recursos e elasticidade em nuvens de computação. A solução é composta por um modelo de desenvolvimento de soluções elásticas, um modelo de composição de aplicações a partir da implementação de funcionalidades e serviços, uma arquitetura e um *middleware*. Para a avaliação e validação da solução proposta foi implementado um protótipo de testes e uma aplicação de distribuição de fluxos de vídeo em tempo real, com redução automática de fluxos redundantes. Os resultados obtidos validaram o funcionamento da aplicação de prova de conceito adaptada para o funcionamento em conjunto com o Trade Wind, assim como sua funcionalidade adicional de fornecimento de fluxos de vídeo em multicanais. A aplicação de redução de fluxos redundantes provou reduzir pela metade o consumo de banda no cenário de teste configurado, tendo potencial de maior economia no caso de aumento do número de fluxos redundantes.

Palavras-chave: Computação em Nuvem, Gerenciamento Automático de Recursos, *Middleware*, Aplicações Distribuídas.

ABSTRACT

Research projects have started working on cloud computing platforms to help cloud applications to be developed in an easiest manner, from year 2000 on. Studies about cloud computing elasticity solutions showed many works were focusing in processing and storage resource management for client-server applications. However, only a small number of research works explore the potential of application contexts regarding network resource management (e.g., content distribution applications). Therefore it is interesting to develop an open platform for distributed applications development helping to manage resources and elasticity in clouds. This dissertation presents Trade Wind, a solution to help the development of distributed applications and services for cloud computing resource and elasticity management. The solution is composed by an elastic application development model, an application compostion model from features and services development, an architecture and a middleware. In order to evaluate and validate the suggested solution, it was developed a test prototype implementing an application for real time video streams distribution utilizing an automatic redundant streams reduction feature. The results collected from the test executions validate Trade Wind solution running the adapted proof of concept application. The tests also showed the multichannel feature added working in a adequate manner. The redundant streams reduction application has proven to reduce bandwidth consumption by the half in the configured test scenarios. And it also has potential to save more bandwidth resources in a scenario with higher number of redundant video streams.

Keywords: Cloud Computing, Automatic Resource Management, Middleware, Distributed Applications.

LISTA DE ABREVIATURAS E SIGLAS

ALM	*Application Layer Multicast*
	Multicast em Camada de Aplicação
API	*Application Programming Interface*
	Interface de Programação de Aplicação
APP	*Aplication*
	Aplicação
CDN	*Content Delivery Network*
	Rede de Entrega de Conteúdo
CEO	*Chief Executive Officer*
	Chefe Executivo de Ofício
CPU	*Central Processing Unit*
	Unidade de Central de Processamento
DHCP	*Dynamic Host Configuration Protocol*
	Protocolo de Configuração Dinâmica de *Host*
EP	Escola Politécnica
FTP	*File Transfer Protocol*
	Protocolo de Transferência de Arquivos
HTTP	*Hypertext Transfer Protocol*
	Protocolo de Transferência de Hipertexto
HTTPS	*Hypertext Transfer Protocol Secure*
	Protocolo de Transferência de Hipertexto Seguro
IaaS	*Infrastructure as a Service*
	Infraestrutura como Serviço

IEEE	*Institute of Electrical and Electronics Engineers*
	Instituto de Engenheiros Elétricos e Eletrônicos
iOS	*iPhone Operating System*
	Sistema Operacional do *iPhone*
LARC	Laboratório de Arquitetura e Redes de Computadores
LCP	*Longest Common Path*
	Caminho Comum Mais Longo
MPEG	*Moving Picture Experts Group*
	Grupo de Especialistas em Imagens com Movimento
MV	Máquina Virtual
NIST	*National Institute of Standards and Technology*
	Instituto Nacional de Padrões e Tecnologia
NTP	*Network Time Protocol*
	Protocolo de Tempo para Redes
PaaS	*Platform as a Service*
	Plataforma como Serviço
PCS	Departamento de Engenharia da Computação e Sistemas Digitais
PX	Pixel
RAM	*Random Access Memory*
	Memória de Acesso Aleatório
RR	*Redundant Request*
	Requisição Redundante
RTMP	*Real Time Messaging Protocol*
	Protocolo de Troca de Mensagens em Tempo Real

RTP	*Real-Time Protocol*
	Protocolo em Tempo Real
RTSP	*Real-Time Streaming Protocol*
	Protocolo de *Streaming* em Tempo Real
SA	Sociedade Anônima
SaaS	*Software as a Service*
	Software como Serviço
SLA	*Service Level Agreement*
	Contrato de Nível de Serviço
SO	Sistema Operacional
STB	*Set Top Box*
TCP	*Transmission Control Protocol*
	Protocolo de Controle de Transmissão
USP	Universidade de São Paulo
VLAN	*Virtual Local Area Network*
	Rede Local Virtual

LISTA DE FIGURAS

Figura 1 – Arquitetura de uma Nuvem de Computação Adaptado de: (ZHANG *et al*, 2010) 26
Figura 2: Interação entre as partes interessadas Adaptado de: (ARMBRUST *et al*, 2010) 29
Figura 3: Fornecimento de fluxos redundantes 44
Figura 4: Fornecimento de fluxo otimizado com a transmissão m*ulticast* 45
Figura 5: Fornecimento de fluxo otimizado com a técnica de *multicast* em camada de aplicação (nós externos à rede) 46
Figura 6: Interação entre os nós da arquitetura do *Wind* para distribuição de um fluxo de vídeo 48
Figura 7: (a) Caminho mais curto entre os nós 1 e 4; (b) caminho mais curto entre os nós 1 e 5; (c) maior caminho comum entre os nós 1, 4 e 5 50
Figura 8: Transformação do fluxo de vídeo em blocos numerados 51
Figura 9: Encaminhamento de blocos 52
Figura 10: Reconstituição do fluxo de vídeo 52
Figura 11: Modelo entidade-relacionamento das informações 62
Figura 12: Grafo representando as relações de vizinhança a Tabela 6 63
Figura 13: Grafo representando as máquinas virtuais 64
Figura 14: (a) Máquinas virtuais instanciadas pelo usuário ‘abacate’ (preto) (b) Máquinas virtuais instanciadas pelo usuário ‘banana’ (cinza) 66
Figura 15: Organização dos elementos da infraestrutura do Trade Wind 69
Figura 16: Interfaces de comunicação do módulo Core. Adaptador de: (MIERS *et al*, 2011) 70
Figura 17: Interfaces de comunicação do módulo Core Access. Adaptado de: (MIERS *et al*, 2011) 71
Figura 18: Visão geral da aplicação de fornecimento de fluxos de vídeo 73
Figura 19: Aplicação de fornecimento de fluxos e sua comunicação com o módulo de monitoramento 74
Figura 20: Interação entre os módulos do mecanismo de monitoramento e verificação 75
Figura 21: Nó ‘M1.host1’ fornecendo fluxos redundantes para os nós ‘M1.host3’e ‘M1.host4’ 77

Figura 22: ‘*Host2*’ identificado como ponto de criação de um novo nó *Relay* 77
Figura 23: Criação de um novo nó de repasse de fluxos de vídeo.....78
Figura 24: Fluxo reduzido após criação de instância 'M1.host2' 78
Figura 25: Topologia de aplicação do cenário de teste 1 82
Figura 26: Topologia de aplicação dos cenários de teste 2 e 3 (fluxo de vídeo I) 84
Figura 27: Topologia de aplicação do cenário de teste 3 para o fluxo de vídeo II 85
Figura 28: Topologia mínima 87
Figura 29: Topologia lógica.....88
Figura 30: Topologia física.....90
Figura 31: Composição do tempo de redução dos fluxos redundantes.....94
Figura 32: Análise do tráfego de dados no cenário 1 95
Figura 33: Análise do tráfego de dados no cenário 2 97
Figura 34: Análise do tráfego de dados no cenário 3 99

LISTA DE TABELAS

Tabela 1: Classificação de soluções de elasticidade. Adaptada de: (GALANTE; BONA, 2012) 36
Tabela 2: Modelo de desenvolvimento de serviços elásticos proposto 57
Tabela 3: Funcionalidades do serviço de redução de fluxos redundantes de acordo com o modelo 58
Tabela 4: Funcionalidades do serviço de Posicionamento Dinâmico de Servidores de Cache de acordo com o modelo 59
Tabela 5: Funcionalidades da aplicação de Posicionamento Dinâmico de Servidores de Cache de acordo com o modelo 59
Tabela 6: Tabela com informações de *host* e sua relação de vizinhança 62
Tabela 7: Tabela usuário com valores fictícios 63
Tabela 8: Tabela máquina virtual com seus campos e valores fictícios 63
Tabela 9: Tabela aplicação e seus valores fictícios 64
Tabela 10: Tabela de tipos de nó e a lista de nós a serem iniciados 65
Tabela 11: Tabela nó e seus valores fictícios 65
Tabela 12: Modelo de Desenvolvimento de Aplicações a partir da composição de Serviços e Funcionalidades 67
Tabela 13: Relação entre as ações de ajuste de demanda e o modelo de desenvolvimento de serviços elásticos 76
Tabela 14: Configurações do vídeo escolhido 92
Tabela 15: Resultados referentes à métrica de tempo de redução dos fluxos redundantes 94

SUMÁRIO

1 Introdução 14
1.1 Motivação 15
1.2 Objetivos 16
1.3 Justificativa 16
1.4 Método 18
1.5 Contribuições 19
1.6 Organização 19
2 Computação em Nuvem 21
2.1 Definições encontradas na literatura 21
2.2 Características 22
2.3 Os Três Modelos de Serviço 25
2.4 Os Quatro Modelos de Implantação 27
2.5 Partes Interessadas 28
2.6 Considerações sobre o Capítulo 30
3 Trabalhos Relacionados 32
3.1 Elasticidade: Definição e Conceitos 32
3.2 Critério para Classificação de Soluções de Elasticidade 34
3.3 Análise das Soluções 35
3.3.1 Soluções de Infraestrutura 36
3.3.2 Soluções de Plataforma 38
3.4 Aplicações Multimídia no contexto de Computação em Nuvem 40
3.4.1 *Content Delivery Network*s e Elasticidade na Nuvem 40
3.5 Wind: Uma Aplicação Distribuída para Distribuição de Fluxos de Vídeo em Tempo Real 42
3.5.1 O Problema dos Fluxos Redundantes 43
3.5.2 Visão Geral do Sistema 46
3.5.3 Arquitetura 47
3.5.4 Algoritmo de Caminho Comum Mais Longo 49
3.5.5 Detalhes de Implementação e Configurações 51
3.5.6 Análise do Wind e Oportunidades de Melhorias 52
4 A Solução Trade Wind 54
4.1 Uma Proposta de Modelo para Desenvolvimento de Aplicações Elásticas 55

4.1.1 Elasticidade Automática Reativa – Monitoramento e Controle 55
4.1.2 Modelo de Desenvolvimento de Serviços Elásticos.....56
4.1.3 Exemplos de Aplicações Utilizando o Modelo 58
4.2 *Middleware* e Funcionalidades.....60
4.2.1 Funcionalidade 1: Manipulação de Máquinas Virtuais.....60
4.2.2 Funcionalidade 2: Gerenciamento de Informações 61
4.3 Modelo Aplicação, Serviço, Funcionalidade 66
4.4 Arquitetura.....67
4.4.1 Organização da Infraestrutura 68
4.4.2 Módulos Essenciais do Trade Wind.....69
4.5 Aplicação – Fornecimento de Fluxos de Vídeo 72
4.5.1 Módulos de Distribuição de Fluxo.....72
4.5.2 Módulos do Modelo de Elasticidade 73
4.6 Elasticidade – Rotina de Ajuste de Demanda 75
4.6.1 Rotina – Redução de Fluxos Redundantes 76
5 Análise Experimental.....79
5.1 Métricas do Experimento.....79
5.2 Cenários de Teste 80
5.2.1 Rotinas comuns aos cenários de teste 80
5.2.2 Cenário 1: Redução do Fluxo Redundante.....81
5.2.3 Cenário 2: Alteração de Aplicação de *Download* do Cliente 82
5.2.4 Cenário 3 – Multicanais 84
5.3 Topologia do Ambiente de Testes.....85
5.3.1 Topologia Mínima.....86
5.3.2 Topologia Lógica (Virtualizada).....87
5.3.3 Topologia Física (Real) 89
5.4 Caracterização do Conteúdo Utilizado nos Testes 90
5.5 Método de Testes 92
5.6 Apresentação e Discussão dos Resultados 93
5.6.1 Resultados do Cenário 1 93
5.6.2 Resultados Cenário 2 96
5.6.3 Cenário 3 – Multicanais 98
5.6.4 Análise de Limitações da Solução 100
6 Considerações Finais.....102

6.1 Contribuições 103
6.2 Trabalhos Futuros 104

Referências 106

Apêndice A – Publicações e Premiações 110

1 INTRODUÇÃO

Computação em nuvem é um conceito que vem sendo amplamente divulgado e discutido ao longo dos últimos anos. Apesar de seu primeiro registro acadêmico conhecido datar de 1997 (CHELLAPPA, 1997), somente em 2007 o termo começou a constar nos registros históricos de busca em uma das mais populares ferramentas de busca da Internet, o Google[1]. No ambiente empresarial, esse é um dos temas de maior destaque e relevância entre os administradores de empresas, ocupando as primeiras posições nos *rankings* de pesquisa sobre as estratégias de negócio nos últimos quatro anos (MCDONALD, 2013). Na academia, uma evidência de sua relevância é o sítio dedicado ao tema[2] mantido pelo Instituto de Engenharia Elétrica e Eletrônica (IEEE).

Apesar de ser um tópico popular, ainda permanece uma confusão sobre o que realmente é a computação em nuvem e qual sua utilidade. Em linhas gerais, o termo refere-se a um novo modelo de computação no qual *software* e *hardware* são disponibilizados na forma de serviços. (ARMBRUST *et al*, 2010) faz uma clara distinção entre dois tipos de serviços. O primeiro, chamado de *utility computing*, é definido como um conjunto de serviços de *hardware* e *software* relacionados à operação e gestão dos recursos da nuvem (processamento, armazenamento e rede). O segundo tipo, são as aplicações providas como serviço por meio da Internet. O trabalho de (MELL; GRANCE, 2011) detalha ainda mais estes modelos de serviço, fazendo distinção entre três abordagens: infraestrutura como serviço (*Infrastructure as a Service* – IaaS), plataforma como serviço (*Platform as a Service* – PaaS) e *software* como serviço (*Software as a Service* – SaaS).

Dentre as características inovadoras da computação em nuvem, (ARMBRUST *et al*, 2010) cita a dinamicidade de aquisição e liberação de recursos computacionais de acordo com a necessidade dos usuários (sob demanda), de forma rápida e sem a necessidade de um planejamento prévio por parte do usuário dos serviços.

---

[1] http://www.google.com/trends/explore?q=cloud+computing#q=cloud%20computing&cmpt=q
[2] http://cloudcomputing.ieee.org/

1.1 Motivação

A partir da segunda metade dos anos 2000, iniciou-se o desenvolvimento de projetos de pesquisa para o desenvolvimento de plataformas abertas voltadas à infraestrutura da nuvem (modelo IaaS), tendo destaque projetos como OpenNebula (2005)[3], Eucalyptus (2007)[4] e OpenStack (2010)[5]. Na mesma época, esforços na direção de desenvolvimento de plataformas (PaaS) que facilitassem a criação de aplicativos para o ambiente de nuvem foram desenvolvidas por iniciativas privadas, como Microsoft Azure[6], Google AppEngine[7], Amazon Simple DB/S3[8], entre outras.

Embora as soluções de PaaS sejam eficientes e facilitem o trabalho dos desenvolvedores para o desenvolvimento de aplicativos adaptados ao ambiente de computação em nuvem, cada solução proprietária possui sua lista de serviços e APIs com diferentes formas de utilização, implicando em certa dependência por parte do usuário (GALANTE; BONA, 2012). Outro fator a ser considerado nestas plataformas é a dificuldade em manter a flexibilidade de configuração das aplicações e suas estruturas e, ao mesmo tempo, prover características inerentes à computação em nuvem como escalabilidade e tolerância a falhas (ARMBRUST *et al*, 2010).

Além disso, alguns domínios de aplicação, como os sistemas distribuídos, possuem características que dependem do controle e interação com os módulos de gestão de recursos de infraestrutura para que apresentem um melhor desempenho. No caso das aplicações de fornecimento de conteúdo, os recursos de rede são limitados e devem ser gerenciados de forma a evitar desperdícios. Este tipo de aplicação utiliza os recursos de processamento e armazenamento de dados para reduzir a utilização dos recursos de rede. Tais aplicações diferenciam-se daquelas comumente implementadas em nuvens de computação, onde os recursos de processamento e armazenamento de dados são mais utilizados e situam-se próximos (em um mesmo centro de processamento, mesmo *rack* de servidores).

---

[3] http://opennebula.org/about/project/
[4] https://www.eucalyptus.com/about/story
[5] http://docs.openstack.org/training-guides/content/module001-ch003-core-projects.html
[6] https://azure.microsoft.com
[7] https://developers.google.com/appengine/docs/python/gettingstartedpython27/introduction
[8] http://aws.amazon.com/pt/s3/

Devido a estas características, é interessante o desenvolvimento de uma plataforma aberta para a criação de aplicações e serviços distribuídos que auxiliem o gerenciamento de recursos no contexto de nuvem.

1.2 Objetivos

Este trabalho tem por objetivo a elaboração de uma solução que auxilie os provedores de nuvens de computação a melhorar o gerenciamento e operação de sua infraestrutura de suporte a aplicações distribuídas. Mais precisamente, o objetivo é especificar, projetar e avaliar uma arquitetura na qual o provimento de serviços considere a priorização de utilização dos recursos de acordo com suas necessidades. Para que este objetivo seja alcançado, são propostos os seguintes objetivos específicos:

- Levantar junto à literatura as soluções de infraestrutura e plataforma para computação em nuvem que realizam algum tipo de gerenciamento dinâmico de recursos alocados às aplicações sendo executadas;
- Estudar as características de aplicações para nuvem de computação que permitem a elasticidade de recursos;
- Especificar funcionalidades relevantes para uma plataforma de computação em nuvem que auxiliem na gestão do uso de recursos da nuvem de forma elástica;
- Projetar uma solução agrupando estas funcionalidades;
- Desenvolver um protótipo que permita a validação da solução proposta.

1.3 Justificativa

Apoiando-se na divisão de responsabilidades e interesses sobre os serviços da nuvem, o lucro dos provedores de serviço das camadas de infraestrutura e plataforma vem da venda de serviços provisionados em forma de recursos de *hardware* e *software*. Para alcançar tal lucro, é necessário que a plataforma e a infraestrutura sejam utilizadas em níveis mais próximos a sua capacidade máxima, embora ainda com garantias de confiabilidade. Assim sendo, a operação de

alocação e liberação de recursos sob demanda de uma forma rápida e eficiente, além de uma característica descrita nas definições de computação em nuvem, pode ser elencada como de considerável interesse por parte dos provedores de nuvem.

Neste contexto, é desejável uma solução que auxilie o gerenciamento e provisão dinâmicos dos recursos oferecidos, permitindo uma melhor alocação destes às diferentes demandas das aplicações envolvidas e, ao mesmo tempo, uma redução em termos de esforço de gestão e operação da nuvem. Isto pode ser feito a partir da identificação de necessidades específicas das aplicações desenvolvidas pelos provedores de serviço de *software*. Por exemplo, considerando o caso de uma aplicação que é executada em um servidor *web*, a qual processa requisições dos clientes. O recurso que se deseja gerenciar de forma automática é o processamento das requisições. À medida que o número de requisições ao servidor aumenta, o serviço tende a diminuir sua eficiência até que o servidor não seja capaz de tratar novas requisições. Para evitar a degradação do serviço, é necessária a alocação de recursos de processamento para manter o serviço ativo. Assim, é do interesse do provedor da nuvem utilizar um serviço de gerenciamento que garanta a alocação automática desses recursos.

Outro cenário interessante, desta vez considerando o gerenciamento de recursos de rede, seria uma aplicação de fornecimento de conteúdo, nas quais um servidor que armazena o conteúdo recebe requisições dos clientes. Neste caso, o recurso limitante é a largura de banda e não mais o processamento das requisições. Estratégias que diminuam a quantidade de dados redundantes que são trafegados, que identifiquem conteúdos de maior popularidade, além de facilitar o posicionamento de cópias mais próximas dos clientes, ajudariam na melhora da qualidade do serviço.

A partir destes exemplos, é possível estender a ideia para o gerenciamento de outros tipos de recursos da nuvem, sejam eles recursos de armazenamento/processamento de dados ou recursos de rede. É importante observar que parte dessas aplicações de gerenciamento possuem um conjunto de características comuns e, ao mesmo tempo, características específicas ao domínio de aplicação e ao recurso a ser gerenciado.

1.4 Método

Este trabalho empregou o método de pesquisa aplicada baseado em hipótese-dedução, utilizando referências científicas para definição do problema, especificação da hipótese de solução e sua avaliação. Além do levantamento bibliográfico das soluções existentes e da especificação de funcionalidades para a proposta de solução, foi implementado um protótipo para validação da hipótese de que “é possível otimizar dinamicamente o uso de recursos da nuvem por aplicações distribuídas usando uma solução de gerenciamento que leve em consideração as especificidades da aplicação alvo”.

A fim de levantar e avaliar as propostas relacionadas ao domínio do problema, foram estudadas soluções abertas de computação em nuvem e suas características. Tais estudos permitiram a identificação de conceitos relacionados às áreas investigadas e os principais desafios a serem trabalhados na especificação da solução. Em especial, foram identificadas funcionalidades comuns a diversas aplicações de gerenciamento de elasticidade, levando à definição de um módulo reutilizável para a elasticidade de recursos em diferentes contextos. A partir destes estudos foi escolhido como alvo específico aplicações para fornecimento de fluxos de vídeo em tempo real. O interesse nesta classe particular de aplicações justifica-se não apenas pelo elevado fluxo de dados envolvido nas mesmas, mas também pelo fato de existirem diversos estudos para otimização de seus recursos, como mecanismos de *multicast* implementados em camada de aplicação. Utilizando esta aplicação como estudo de caso, foi então adaptada uma solução para redução automática de fluxos redundantes com funcionalidades específicas deste contexto. Além da proposta de um modelo de desenvolvimento de aplicações elásticas, um modelo de composição de aplicações e de um *middleware*, foi implementado um protótipo baseado nos requisitos levantados na especificação do problema. Por fim, foram colhidos dados de desempenho com o objetivo de demonstração do correto funcionamento da solução proposta.

A pesquisa foi realizada dentro do Laboratório de Arquitetura e Redes de Computadores (LARC) do Departamento de Engenharia da Computação e Sistemas Digitais (PCS) da Escola Politécnica da Universidade de São Paulo (EP-USP).

1.5 Contribuições

Este trabalho traz uma revisão referenciada das principais soluções de infraestrutura e plataforma para garantia de elasticidade em nuvens de computação e discute o problema da redução de fluxos redundantes em aplicações de distribuição de vídeo. A partir do estudo do Wind, uma aplicação distribuída de fornecimento de fluxos de vídeo desenvolvida em meados de 2010 pela área de pesquisa e inovação da empresa sueca Ericsson Telecomunicações S.A.[9], pelos pesquisadores Stefan Hellkvist e Joacim Halén, foi levantado um conjunto de funcionalidades desejáveis em aplicações elásticas e organizado um modelo de desenvolvimento de tais aplicações.

Assim, a contribuição acadêmica do trabalho é formada pelo estudo e desenvolvimento de uma solução aberta que permite o desenvolvimento de aplicações de gerenciamento e operação de nuvens de computação visando à elasticidade de recursos de acordo com as necessidades dos serviços disponibilizados em sua plataforma e infraestrutura. A solução proposta é composta por um modelo de desenvolvimento de soluções elásticas, um modelo de composição de aplicações a partir da implementação de funcionalidades e serviços, bem como uma arquitetura de serviços e um *middleware* para uso dessa arquitetura. Sua validação é realizada por meio da coleta e análise de dados obtidos de um protótipo implementado como prova de conceito. A aplicação escolhida como estudo de caso realiza a distribuição de fluxos de vídeo em tempo real, para a qual é provido um mecanismo de redução automática de fluxos redundantes.

1.6 Organização

Este trabalho está dividido em seis capítulos principais, visando facilitar o fluxo de leitura. A partir da introdução, descrita no **Capítulo 1**, são apresentados no **Capítulo 2** conceitos importantes sobre computação em nuvem e definidos termos que são utilizados com frequência no desenvolvimento da dissertação. Em seguida, o **Capítulo 3** apresenta características de soluções abertas de elasticidade em

[9] http://www.ericsson.com/br/thecompany

computação em nuvem desenvolvidas para as camadas de infraestrutura e plataforma. Esse capítulo também introduz o problema do fornecimento de fluxos redundantes pela Internet e algumas soluções encontradas na literatura, como a técnica de *multicast*, e o *multicast* em camada de aplicação; ao final do capitulo, é apresentada uma descrição e um breve estudo da solução Wind. No **Capítulo 4**, é descrita a proposta de solução, denominada Trade Wind, sendo detalhados: o modelo de otimização de recursos, a arquitetura, o *middleware*, a API e a aplicação de fornecimento de fluxos de vídeo com redução automática de fluxos redundantes. O **Capítulo 5** apresenta e analisa os resultados obtidos através dos testes realizados com o protótipo de prova de conceito. Por fim, o **Capítulo 6** resume as contribuições e limitações da pesquisa, concluindo com os possíveis trabalhos futuros.

2 COMPUTAÇÃO EM NUVEM

O objetivo deste capítulo é apresentar e discutir o termo computação em nuvem e os termos relacionados que são utilizados ao longo deste trabalho. A partir da tradução e do estudo das definições mais referenciadas na literatura, o capítulo apresenta uma contextualização sobre a área, trazendo detalhes e exemplos sobre suas características, modelos de implementação e partes interessadas. Dado que tais conceitos são razoavelmente consagrados na literatura, leitores que têm boa familiaridade com a área de computação em nuvem podem preferir proceder para o próximo capítulo diretamente. Leitores interessados em uma discussão mais detalhada sobre o assunto, por outro lado, podem encontrar maiores informações em (ARMBRUST *et al*, 2010) e (MELL; GRANCE, 2011), que são as principais referências utilizadas na presente discussão.

2.1 Definições encontradas na literatura

Ao procurar pelo termo *Cloud Computing* no Google Scholar[10], uma das mais populares ferramentas de busca para o meio acadêmico, dois trabalhos apresentam destaque pelo seu número de citações: o primeiro é um trabalho de pesquisadores da Universidade da Califórnia, de (ARMBRUST *et al*, 2010); já o segundo é uma publicação de caráter especial do NIST, criada por (MELL; GRANCE, 2011). Apesar de essas duas referências compartilharem muitas características em comum, elas diferenciam-se pela contextualização e discussão de oportunidades e desafios da área, tópicos mais amplamente explorados no trabalho de (ARMBRUST *et al*, 2010). Segundo (MELL; GRANCE, 2011), a computação em nuvem pode ser definida como:

> "... um modelo que permite a **provisão / desprovisão de recursos computacionais** de forma **rápida**, e com o **mínimo esforço de gerenciamento** e de **interação** com o **provedor de serviço**. O **acesso** a estes recursos é feito através de uma **rede ubíqua**[11] **(onipresente), conveniente e sob demanda**." (tradução do autor)

[10] http://scholar.google.com.br/

[11] Pode ser acessada de qualquer lugar, por meio de dispositivos de diferentes capacidades de processamento (*thin/thick client*), por meio de interfaces padronizadas (SATYANARAYANAN, 2001)

Nesta definição, observa-se um foco maior na descrição do modelo, destacando suas características, e na maneira como os recursos são providos pelos proprietários/administradores das nuvens. É também interessante observar o importante papel das redes de comunicação na garantia de acesso a recursos computacionais, como processamento e armazenamento.

No trabalho de (ARMBRUST *et al*, 2010), a computação em nuvem é definida como:

> "... a soma de *software* como serviço (SaaS) e computação como *commodity*"
> (tradução do autor)

Nesta definição, (ARMBRUST *et al*, 2010) separa os serviços de computação em duas categorias: os serviços de *software*, oferecidos via Internet, são colocados na categoria chamada de Software como Serviço (*Software as a Service – SaaS*); e os serviços de *hardware* (e os *softwares* envolvidos), providos nos centros de processamento de dados, são chamados de computação como *commodity* (do inglês, *utility computing*), já que os recursos computacionais são consumidos sob demanda e pagos de acordo com o consumo. Assim, computação seria provida como o atual serviço de distribuição de água ou energia elétrica.

A partir destas duas breves definições, é possível descrever a computação em nuvem como um modelo de computação onde *hardware* e *software* são disponibilizados em forma de serviço, e onde a quantidade de recursos computacionais alocados se ajuste de forma rápida e com reduzido esforço de gerenciamento à real demanda dos usuários. Na próxima seção, são apresentadas e discutidas as características essenciais a uma nuvem de computação pela visão dos dois autores.

2.2 Características

Parte da confusão sobre o termo computação em nuvem se deve a sua ampla divulgação em diários eletrônicos e revistas não especializadas, que, na tentativa de explicar a computação em nuvem, acabavam por confundir os leitores sobre certas semelhanças com o que já existia em termos de computação. (ARMBRUST *et al*,

2010) cita uma fala de Larry Ellison, CEO da Oracle, no evento Oracle *OpenWorld* de 2008, que ilustrar tal confusão:

> "Uma coisa interessante sobre computação em nuvem é que nós redefinimos computação em nuvem para incluir tudo que nós já fazemos.... eu não entendo o que nós fazemos de forma diferente, a não ser trocar algumas palavras em nossos anúncios publicitários." (tradução do autor)

A fim de esclarecer estes erros de interpretação, algumas características essenciais à computação em nuvem foram descritas nas definições estudas. A seguir, são apresentadas as traduções dos dois trabalhos mais referenciados.

Em (MELL; GRANCE, 2011, p. 2), são propostas cinco características essenciais a uma nuvem de computação, descritas abaixo:

- Auto Serviço sob Demanda: um consumidor do serviço pode aumentar ou reduzir a utilização dos recursos da nuvem, como o tempo de uso de um servidor ou o espaço de armazenamento em uma rede, quando preciso. Isto é feito de forma automática, e sem exigir interação humana com cada provedor de serviço.

- Amplo Acesso à Rede: os serviços ficam disponíveis a partir da rede e podem ser acessados através de mecanismos padronizados, o que permite o uso de plataformas heterogêneas, tanto de pequeno como de grande porte (e.g., telefones móveis, *tablets,* computadores portáteis e estações de trabalho).

- Agrupamento de Recursos: os recursos computacionais dos provedores de serviço são agrupados para servir diversos consumidores. Utilizando um modelo de múltiplas instâncias, com diferentes recursos físicos e virtuais, dinamicamente alocados e liberados conforme a demanda do consumidor. Há certa independência de localidade, de modo que o consumidor não tem controle ou conhecimento exato sobre a localização dos recursos consumidos. Porém, ainda sim há a possibilidade de especificar a localização em um alto nível de abstração (e.g., país, estado, ou centro de processamento). Exemplos de recursos incluem armazenamento, processamento, memória e largura de banda de rede.

- Elasticidade Ágil: os serviços podem ser adquiridos ou liberados de forma ágil e elástica, em alguns casos, automaticamente. Para o consumidor, é comum

que os serviços disponíveis para consumo pareçam ser ilimitados, ou seja, podem ser consumidos em qualquer quantidade e a qualquer momento.

- Medição de Consumo do Serviço: sistemas de nuvem controlam e otimizam o uso de recursos com base em seu serviço de medição, alcançando um nível de abstração recomendado para a medição de um determinado tipo de serviço (e.g., armazenamento, processamento, banda de rede, e contas ativas de usuário). O uso de recursos pode ser monitorado, controlado e auditado, trazendo transparência tanto para provedor como para o consumidor do serviço utilizado.

Já (ARMBRUST *et al*, 2010), pontua três características essenciais da computação em nuvem:

- A ilusão de recursos computacionais infinitos, disponíveis sob demanda, e de forma rápida o bastante para atender picos de carga, eliminando a necessidade dos usuários planejarem o uso da infraestrutura a médio e longo prazo.

- A eliminação de um compromisso antecipado por parte dos usuários da nuvem, permitindo que empresas pequenas comecem pequenas e aumentem seus recursos computacionais somente quando houver aumento de suas demandas.

- A possibilidade de pagar pelo uso de recursos computacionais por períodos mais curtos (e.g., por horas de processamento ou dias de armazenamento), e sendo recompensado pela liberação de recursos ociosos.

Considerando o exposto, pode-se perceber que a descrição de (MELL; GRANCE, 2011, p. 2) é mais abrangente, trazendo características da rede de acesso aos recursos computacionais, do modelo de múltiplas instâncias e aspectos como localidade e quais recursos são consumidos (armazenamento, processamento, memória e largura de banda de rede). Já (ARMBRUST *et al*, 2010), foca principalmente nos pontos de escalabilidade e sua implicação na relação entre consumidor e provedor da nuvem, explicando as possibilidades e benefícios para empresas pequenas e médias.

Apesar dos pontos complementares acima expostos, observa-se que os dois autores dão ênfase à forma como a aquisição e liberação dos recursos deve ser realizada. Especificamente, ambos citam que estas ações devem ser realizadas sob

demanda, de forma rápida e automática. Portanto, pode-se afirmar que o gerenciamento da aquisição e liberação de recursos é um dos pontos importantes para os serviços prestados pelos provedores de nuvem, sendo este o foco da presente dissertação.

2.3 Os Três Modelos de Serviço

Os modelos de serviço definem as responsabilidades divididas entre as camadas de serviço. Apesar de existirem diversas nomenclaturas derivadas, o modelo básico aqui apresentado é descrito em três camadas: a camada de Infraestrutura como Serviço (*Infrastructure as a Service* – IaaS), a camada de Plataforma como Serviço (*Platform as a Service* – PaaS) e a camada de *Software* como Serviço (*Software as a Service* – SaaS). Na **Figura 1**, adaptada de (ZHANG *et al*, 2010), são apresentadas em três colunas principais: as camadas de serviço, os recursos gerenciados em cada camada (e suas subdivisões internas) e, por fim, alguns exemplos de implementações oferecidas no mercado.

A camada de infraestrutura é composta de recursos físicos (*hardware*) e virtuais (*softwares* responsáveis pela virtualização dos recursos físicos). A camada de plataforma possui um conjunto de ferramentas que permitem ao desenvolvedor a utilização dos recursos da infraestrutura para implementação dos serviços oferecidos pela camada de *software* (arcabouços de desenvolvimento, sistemas armazenamento de dados). A camada de *software* é composta de aplicações que prestam serviços de interesse aos usuários finais (aplicativos, serviços web, multimídia).

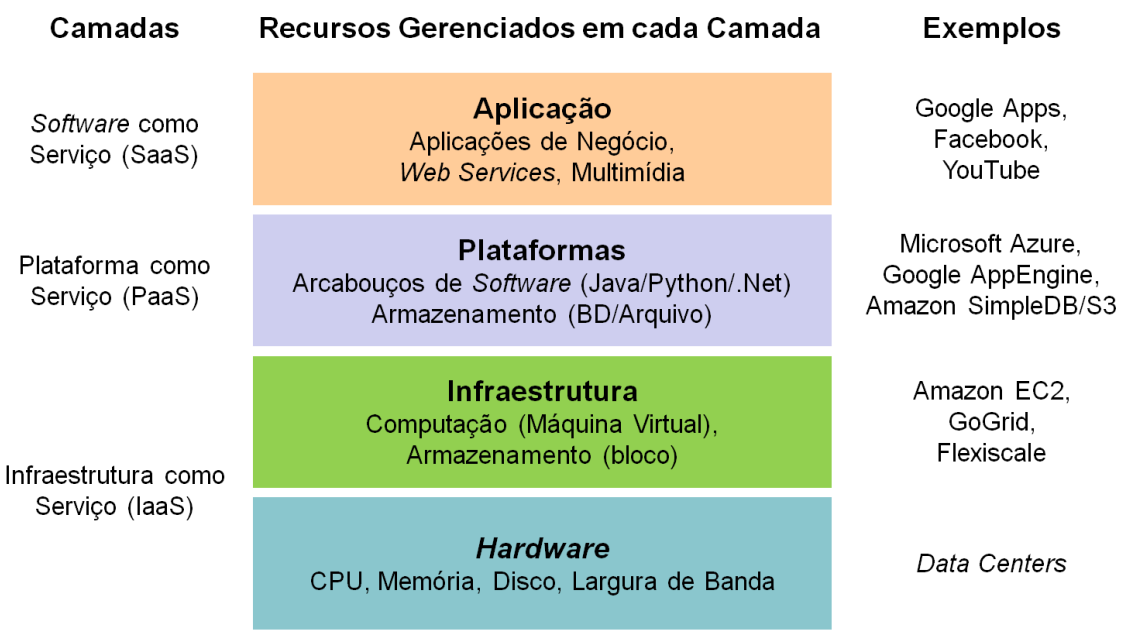

Figura 1 – Arquitetura de uma Nuvem de Computação

Adaptado de: (ZHANG *et al*, 2010)

Abaixo, são apresentadas as definições do modelo de serviços de (MELL; GRANCE, 2011, p. 2):

- Software como Serviço (SaaS): o serviço provido ao consumidor é o direito de utilização de aplicações que estão sendo executadas em uma infraestrutura de nuvem. As aplicações são acessíveis através de diversos dispositivos do cliente, comumente por meio da interface de um navegador Web. O consumidor não controla nem gerencia a infraestrutura de base da nuvem, como rede, servidores, sistemas operacionais, armazenamento, ou mesmo os serviços individuais da aplicação, exceto a configuração de poucos parâmetros específicos.

- Plataforma como Serviço (PaaS): o serviço provido ao consumidor é o desenvolvimento/instalação de aplicações, criadas ou adquiridas pelo consumidor, na infraestrutura de nuvem. Tal desenvolvimento se dá pela utilização de linguagens de programação e ferramentas oferecidas pelo provedor. O consumidor não controla nem gerencia a infraestrutura de base da nuvem, incluindo rede, servidores, sistemas operacionais, ou armazenamento, mas tem controle sobre as aplicações instaladas e sobre a configuração de detalhes do ambiente de hospedagem das aplicações.

- Infraestrutura como Serviço (IaaS): o serviço provido ao consumidor é a disponibilização de recursos de processamento, armazenamento, rede e outros recursos de computação fundamentais, sendo permitido ao consumidor instalar e executar qualquer tipo de software, incluindo sistemas operacionais e outras aplicações. O consumidor não controla nem gerencia a infraestrutura de base da nuvem, mas tem controle sobre o sistema operacional, armazenamento, e desenvolvimento de aplicações; ele também possui controle limitado sobre a seleção de componentes de rede (e.g., *firewalls*).

As três camadas apresentadas nesta seção podem ser implementadas independentemente, ou seja, é possível contratar os serviços de infraestrutura (IaaS) e gerenciar sua própria plataforma (PaaS) e *software* (SaaS). Um dos desafios para que esta independência entre as camadas seja alcançada na prática, entretanto, é a padronização de interfaces dos serviços, o que facilitaria os processos de delegação e federação das nuvens (VILLEGAS *et al*, 2012).

2.4 Os Quatro Modelos de Implantação

Além dos modelos de serviço apresentados na seção 2.3, existem quatro modelos de implantação para nuvens de computação. Tal divisão se faz importante devido às diferentes necessidades de segurança, privacidade e gerenciamento dos dados e aplicações sendo executadas em um ambiente computacional. Nos itens a seguir, traduzidos do trabalho de (MELL; GRANCE, 2011, p. 3), são definidos os quatro modelos de implementação de uma nuvem de computação:

- Nuvem Privada: a infraestrutura da nuvem é operada por apenas uma organização. Ela pode ser gerenciada pela própria organização ou por terceiros, e pode ser hospedada dentro ou fora de suas dependências.

- Nuvem Comunitária: a infraestrutura da nuvem é compartilhada por diversas organizações e atende a uma comunidade específica que tem interesses compartilhados (e.g., missão, requisitos de segurança, políticas e questões de compatibilidade). Pode ser gerenciada por organizações ou terceiros, e pode também pode ser hospedada dentro ou fora de suas dependências.

- Nuvem Pública: a infraestrutura de nuvem é disponibilizada para o público em geral ou para um grande grupo da indústria, e é de propriedade de uma organização que comercializa serviços de nuvem.

- Nuvem Híbrida: a infraestrutura de nuvem é uma composição de duas ou mais nuvens (privada, comunitária, ou pública) que permanecem entidades únicas, mas que firmam um parceria por meio de tecnologias padronizadas ou proprietárias. Essas tecnologias permitem a portabilidade de dados e aplicações (e.g., expansão das nuvens para balanceamento de carga entre as nuvens).

As diferentes implementações de nuvens de computação permitem diferentes níveis de segurança, privacidade e gerenciamento dos dados e aplicações (VILLEGAS *et al*, 2012). As nuvens públicas, por exemplo, se beneficiam do compartilhamento de recursos por um maior número de consumidores, permitindo que empresas de pequeno e médio porte reduzam os custos com infraestrutura e plataforma. Por outro lado, as nuvens privadas são interessantes para empresas com maior exigência quanto aos critérios de privacidade e segurança de seus dados, uma vez que o compartilhamento de máquinas físicas e virtuais aumenta os riscos de que os dados sejam acessados por outros processos ou usuários. Já as nuvens comunitárias são uma alternativa na qual o compartilhamento das máquinas físicas e virtuais é feita dentro de um grupo mais confiável que uma nuvem pública. Por fim, as nuvens híbridas, solução ideal para combinar diferentes requisitos de segurança e privacidade, esbarram no desafio de padronização de interfaces de serviços para que federação de nuvens seja possível (VILLEGAS *et al*, 2012).

2.5 Partes Interessadas

Em toda relação de prestação de serviço, existem ao menos duas partes interessadas, que desempenham os papéis de provedor e consumidor de um determinado serviço. Em serviços de nuvem isso não é diferente, por trás do serviço provido/consumido existem diferentes objetivos, expectativas e responsabilidades. Portanto, se faz necessário o acordo de alguns termos que reflitam os interesses e compromissos das partes interessadas em relação a esse serviço. Este capítulo descreve brevemente as principais relações entre as partes interessadas, de acordo

com os serviços prestados em cada camada do modelo de serviços detalhado na **seção 2.3**.

A **Figura 2**, baseada numa ilustração do artigo de (ARMBRUST, 2010), apresenta os papéis de usuário e provedor, e suas relações a partir de duas perspectivas distintas. A primeira perspectiva está centrada nos serviços de *utility computing* (*hardware* e *softwares* que disponibilizam os recursos computacionais, além das plataformas de desenvolvimento e implementação de aplicações finais). A segunda refere-se às aplicações finais, ou *software* como serviço (aplicações/*softwares* para diversos fins, como redes sociais, visualização e compartilhamento de arquivos multimídia, entre outros). Ainda sobre a **Figura 2**, é importante observar que o provedor de SaaS assume também o papel de consumidor de uma nuvem subjacente (no modelo PaaS ou IaaS). Em alguns casos, os papéis podem ainda ser representados por um mesmo ator (pessoa ou entidade que assume papéis das relações de serviço), ou seja, o provedor de SaaS pode operar e gerenciar sua própria plataforma e infraestrutura de nuvem, assumindo também o papel de provedor de nuvem de IaaS e PaaS, por exemplo.

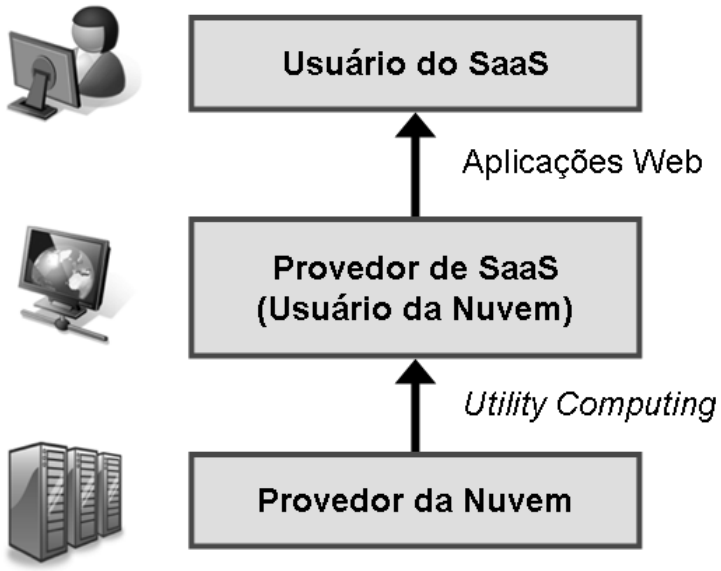

Figura 2: Interação entre as partes interessadas
Adaptado de: (ARMBRUST *et al*, 2010)

(BADGER *et al*, 2012, p. 2-2) ressalta que esses papéis podem ser desempenhados tanto por pessoas como por sistemas. Além disso, o próprio provedor de nuvem pode consumir os serviços de nuvem de outro provedor, caso sua infraestrutura não seja suficiente para atender à demanda de seus usuários. Estas demandas e expectativas dos clientes são um fator importante para estabelecer a prestação/consumo do serviço.

Dois instrumentos para a formalização dessas relações entre provedor e consumidor são definidos por (BADGER *et al*, 2012, p. 3-1): o contrato de serviço e os níveis de serviço contratados (da abreviatura em inglês, SLA – *Service Level Agreement*). O contrato de serviço é um documento legal que especifica as regras contratuais entre o consumidor e o provedor. Um SLA é um documento que descreve detalhes técnicos sobre o desempenho prometido pelo provedor, incluindo as medidas que devem ser tomadas em caso de falhas no serviço ou desempenho inferior ao acordado.

Dentre o conjunto de compromissos do provedor, que deve ser acordado junto aos usuários, são descritos por (BADGER *et al*, 2012, p. 3-2): a disponibilidade do serviço (e.g., porcentagem de disponibilidade ou desempenho mínimo esperado dado um período de tempo), compensações por parte do provedor caso o serviço não atenda os termos de disponibilidade, política de manutenção dos dados (e.g. o que deve ser feito em caso de quebras de contrato, ou término do serviço? Qual seria o período no qual o provedor de serviço deveria manter os dados do usuário em caso de término de um contrato de prestação de serviço?) ou, políticas sobre utilização das informações do usuário (e.g. provedores não devem ter permissão para vender ou divulgar os dados dos usuários, exceto em caso de requisições judiciais).

2.6 Considerações sobre o Capítulo

A partir das discussões apresentadas nesse capítulo, é possível descrever a computação em nuvem como um modelo de computação onde *hardware* e *software* são disponibilizados em forma de serviço. O que a difere de outros paradigmas de computação é a maneira como os recursos computacionais são alocados/liberados:

a quantidade de recursos é ajustada de forma rápida e com reduzido esforço de gerenciamento (pode ser manual ou de forma automática) para atender à demanda dos usuários.

Para garantir a implementação de tais características, foram desenvolvidos mecanismos e ferramentas que auxiliam os provedores de nuvem na operação e gestão de suas nuvens. Outra perspectiva interessante aos provedores de nuvem, com relação ao gerenciamento de recursos, é que a eficiência dos mecanismos e ferramentas utilizados interfere diretamente nos lucros gerados sobre a utilização de uma mesma infraestrutura. Em outras palavras, quanto mais eficiente a gestão dos recursos, maior o lucro possível.

No **Capítulo 3** são apresentadas as principais soluções abertas e proprietárias desenvolvidas para a operação e gestão automatizada de recursos em uma nuvem. Com base na análise de suas características, são discutidas as oportunidades de pesquisa, a partir do estudo de seus pontos positivos e limitações.

3 TRABALHOS RELACIONADOS

Dentre as definições e conceitos de computação em nuvem apresentados e analisados no **Capítulo 2**, destaca-se uma característica importante na definição deste novo paradigma de computação: a elasticidade. Basicamente, este termo refere-se à alocação/liberação de recursos da nuvem de forma rápida e com reduzido esforço de gerenciamento no atendimento da demanda dos usuários. Com o intuito de garantir a implementação do gerenciamento de recursos e da elasticidade em nuvens, diversas soluções abertas e proprietárias foram desenvolvidas. Neste capítulo são apresentadas, brevemente, as definições e classificações de elasticidade encontradas na literatura, assim como uma revisão comentada das soluções comerciais e acadêmicas de gerenciamento de recursos e elasticidade. É dado foco principalmente na revisão sobre as soluções de elasticidade voltadas a aplicações multimídia, em especial no estudo da solução Wind, para fornecimento de fluxos de vídeo com serviço de redução de fluxos redundantes.

3.1 Elasticidade: Definição e Conceitos

A elasticidade é uma característica chave no contexto da computação em nuvem (GALANTE; BONA, 2012). Porém, existe certa confusão quanto ao seu uso, sendo o termo muitas vezes utilizado como sinônimo de escalabilidade ou eficiência. Segundo (HERBST *et al*, 2013), a **escalabilidade** é um pré-requisito para a elasticidade e se trata da habilidade que um sistema tem em suportar o aumento de carga de trabalho a partir da utilização de recursos adicionais. A elasticidade diferencia-se da escalabilidade, pois leva em consideração aspectos como velocidade, frequência e granularidade das ações de escalabilidade. A **eficiência**, por outro lado, expressa uma relação entre a carga de trabalho e a quantidade de recursos consumidos para sua execução (e.g., um sistema A é dito mais eficiente que um sistema B, distinto, se A executa uma carga de trabalho alocando uma menor quantidade de recursos do que B). A eficiência nem sempre é influenciada somente pelos mecanismos de elasticidade de um sistema, mas também por diferentes implementações ou técnicas utilizadas para realizar uma mesma

operação. Após o levantamento e comparação realizados em diversos trabalhos sobre computação em nuvem e elasticidade, (HERBST *et al*, 2013) chegou à seguinte definição:

> "Elasticidade é o grau no qual um sistema é capaz de se adaptar às mudanças na carga de trabalho provendo e desprovendo recursos de uma forma autônoma, na qual, a cada momento, os recursos disponíveis se ajustam o mais próximo possível à atual demanda". (tradução do autor)

Ou seja, a elasticidade trata da capacidade de disponibilização de uma infraestrutura de recursos de nuvem adequada a uma demanda que varia no decorrer tempo. Esta disponibilização deve ser realizada de forma rápida e com reduzido esforço de gerenciamento. Entretanto, a capacidade dos recursos disponibilizados não deve exceder a demanda em demasia, para que não haja desperdício de recursos. Por exemplo, em casos de pico de demanda, a infraestrutura deve alocar recursos para aumentar sua capacidade e atender à nova demanda, porém, ao término do período de pico, os recursos devem ser liberados para que os recursos ociosos possam ser alocados para outras tarefas.

Os movimentos de adaptação da infraestrutura alocada para atender ao aumento ou diminuição de demanda dos usuários possuem diferentes nomenclaturas. Caso seja realizada uma alocação de recursos para atender a uma demanda acima da capacidade atual oferecida, o movimento é chamado de aumento de escalonamento (do inglês, *scale up*). Caso contrário, se há uma liberação de recursos diante de uma demanda muito abaixo da capacidade atual oferecida, visando sua economia, o movimento é chamado de diminuição do escalonamento (do inglês, *scale down*).

Geralmente, os recursos computacionais da nuvem são disponibilizados por meio de máquinas virtuais previamente configuradas com a quantidade de processadores, memória RAM e espaço em disco. De acordo com (ALI-ELDIN *et al*, 2012), existem duas possibilidades de elasticidade com base nas configurações dos recursos virtuais disponibilizados. A **elasticidade horizontal** é o aumento ou diminuição do número de máquinas virtuais (idênticas em quantidade de recursos) alocadas para a execução de um serviço de acordo com a carga de trabalho atual. A **elasticidade vertical** se trata do aumento ou diminuição dos recursos alocados por meio da mudança das configurações de uma máquina virtual em execução (e.g.,

aumento do número de processadores, da capacidade de memória ou armazenamento das máquinas virtuais alocadas).

3.2 Critério para Classificação de Soluções de Elasticidade

Antes de iniciar a apresentação e análise das soluções encontradas na literatura, é interessante definir alguns critérios de classificação para soluções de elasticidade no contexto de computação em nuvem. Tal abordagem facilita a comparação entre as soluções e permite observar as estratégias mais comuns. A seguir, são detalhados os critérios de classificação propostos por (GALANTE; BONA, 2012) em seu levantamento de soluções em elasticidade para computação em nuvem:

- Escopo: define onde as ações referentes à elasticidade são controladas. Especificamente, esse controle pode ser realizado no sistema de gerenciamento da **infraestrutura** (IaaS), na **plataforma** (PaaS) ou na **aplicação** (SaaS). O controlador dos mecanismos de elasticidade é responsável por monitorar dados da aplicação e tomar decisões sobre a necessidade de reajuste da quantidade de recursos alocados, convertendo requisitos do usuário (políticas) em ações para a elasticidade dos recursos.

- Política: para que as ações referentes à elasticidade sejam executadas é necessário que haja uma política que determine se uma ação deve ou não ser executada. A verificação dessa política, assim como a execução de ações, pode ser feita de forma **manual** ou **automática**. Na abordagem manual o usuário é responsável por monitorar suas aplicações e seu conjunto de recursos, executando as ações de elasticidade. No caso da abordagem automática, o controle e execução das ações de elasticidade são realizados por um sistema ou aplicação, a partir de configurações predefinidas pelo usuário. Dentro do grupo das abordagens automáticas, caso as ações sejam executadas através de **regras**, no formato ***condição → ação***, a abordagem é denominada **automática reativa**. Nesse caso, uma ação de ajuste dos recursos alocados só é executada quando as métricas monitoradas atingem uma determinada condição (valores limite). Se as ações de elasticidade forem executadas antecipadamente, numa tentativa de prever variações

da demanda através de heurísticas e técnicas matemáticas/analíticas, a abordagem é chamada de **automática preditiva**.

- Propósito: a elasticidade pode ser utilizada com diversos objetivos, como a manutenção do desempenho do sistema, o aumento de capacidade de uma infraestrutura de IaaS, a redução de custos, a economia de energia entre outros.

- Método: define o método utilizado para a implementação da elasticidade. A partir do estudo da literatura, podem ser observados três diferentes métodos: a **replicação**, o **redimensionamento** e a **migração**. A replicação, também chamada de elasticidade horizontal, consiste na adição/remoção de instâncias (máquinas virtuais, contêineres ou módulos de aplicação). O redimensionamento, ou elasticidade vertical, trata-se da alteração da alocação de recursos por meio da adição/remoção de processadores, memória ou espaço de armazenamento das instâncias. Por fim, a migração de máquinas virtuais é o processo de transferência de uma máquina virtual que está sendo executada em uma máquina física para outra.

3.3 Análise das Soluções

A partir do trabalho de (GALANTE; BONA, 2012), é possível ter uma visão geral das soluções proprietárias e acadêmicas de elasticidade para nuvens de computação. Foram analisados 27 soluções, classificadas de acordo com os critérios apresentados na **Seção 3.2**. Na **Tabela 1**, são apresentadas as 27 soluções analisadas por (GALANTE; BONA, 2012), distribuídas pelas classificações, onde os números em parênteses representam a quantidade de soluções que se enquadram em cada categoria. Observa-se que a maior parte das soluções avaliadas possui as seguintes características: controle de elasticidade no escopo de infraestrutura (IaaS), suas validações de política são realizadas de forma automática reativa (com base em regras), têm como propósito a melhoria do desempenho dos serviços e utilizam o método de replicação para alocação de novos recursos.

Elasticidade	**Escopo**	Infraestrutura (18)	
		Plataforma (3)	
		Aplicação (6)	
	Política	Manual (6)	
		Automática	Reativa (15)
			Preditiva (6)
	Propósito*	Desempenho (23)	
		Aumento de Capacidade de Infraestrutura (3)	
		Custo (2)	
		Energia (2)	
	Método*	Replicação (19)	
		Redimensionamento (8)	
		Migração (3)	

Tabela 1: Classificação de soluções de elasticidade. Adaptada de: (GALANTE; BONA, 2012)

***** Os trabalhos podem se classificados em mais de uma categoria

3.3.1 Soluções de Infraestrutura

A partir das soluções que se enquadram nas categorias com maior número de trabalhos relacionados, segundo o trabalho de (GALANTE; BONA, 2012), foram selecionados os trabalhos que visam automatização dos mecanismos de elasticidade, priorizando as soluções abertas e acadêmicas, para as quais é possível o estudo detalhado das soluções. A partir deste primeiro critério de seleção é possível traçar uma visão geral das soluções de infraestrutura (IaaS). Das 18 soluções que se encontram na categoria de infraestrutura, 6 são soluções de política automática preditiva e 2 manuais. Das 10 soluções restantes, 5 são proprietárias e um dos trabalhos acadêmicos não estava disponível de forma gratuita. Analisando os últimos 4 trabalhos, observou-se que um deles foi classificado como uma solução de infraestrutura, quando na realidade trata-se de uma solução de plataforma. A seguir são comentadas as 3 soluções restantes.

(CHAPMAN *et al*, 2010) propõe uma solução chamada RESERVOIR (acrônimo de *Resources and Services Virtualization without Barriers*). Esta solução propõe uma arquitetura onde o serviço/aplicação é definido como um conjunto de componentes de *software,* encapsulados na forma de uma ou mais imagens de máquinas virtuais (SO, *middleware*, aplicações, configurações e dados). RESERVOIR também provê uma sintaxe e um arcabouço para a definição de regras estáticas de elasticidade, garantindo seu gerenciamento dinâmico.

A solução de (FITÓ *et al*, 2010) apresenta um sistema de elasticidade que visa o cumprimento de SLAs e considera variáveis econômicas para a tomada de decisões. Diferente das métricas comumente monitoradas em sistemas de elasticidade de infraestrutura que utilizam medições de porcentagem de utilização do processador ou o número de leituras e escritas em disco, sua principal métrica é o tempo de resposta de requisições, sendo a medição repetida num intervalo de 10 segundos.

Dentre as soluções proprietárias de infraestrutura, a *Amazon Web Services*[12] disponibiliza um mecanismo de replicação de máquinas virtuais chamado *Auto Scaling*. A *Amazon Auto-Scaling* utiliza o método automático reativo, no qual regras de adição ou liberação de instâncias são definidas e associadas a grupos de escalabilidade chamados de *Auto Scaling Groups*. As regras se baseiam em valores de métricas coletadas na infraestrutura pelo serviço *CloudWatch* que monitora o uso de CPU, o tráfego da rede, a quantidade de leituras e escritas no disco rígido, entre outras, com base em regras estáticas (GALANTE; BONA, 2012). Assim como (CHAPMAN *et al*, 2010) e (FITÓ *et al*, 2010), diversas soluções proprietárias de infraestrutura como Amazon Web Sevices, OnApp[13], RightScale[14] e Scalr[15], utilizam regras estáticas para a automatização das ações de elasticidade.

Em comparação, (LIM *et al*, 2009) evidencia em seu trabalho o problema das regras estáticas.Sua solução propõe um mecanismo de controle de elasticidade a partir de regras que aceitam intervalos limítrofes ao invés de um único valor limítrofe para disparar ações de escalabilidade de recursos. Um exemplo é o caso de limites

[12] http://aws.amazon.com/pt/
[13] http://onapp.com/
[14] http://www.rightscale.com/
[15] http://www.scalr.com/

de processamento, para a qual uma métrica comumente utilizada é a porcentagem de utilização do CPU. Ao invés da definição de dois valores fixos como limites inferior ou superior que são sempre os mesmos independente da quantidade de instâncias alocadas, a solução permite que seja definido um intervalo de valores para cada limite. Por exemplo, 10 % de utilização de CPU em 100 instâncias é diferente de 10% em apenas 2 instâncias: no primeiro caso, parte das máquinas poderia ser liberada para uma melhor utilização de cada instância. A modificação proposta torna o mecanismo mais preciso em relação à quantidade de recursos alocada, se comparado a outros mecanismos que só permitem a especificação de valores fixos nas regras de escalonamento de recursos.

3.3.2 Soluções de Plataforma

Com base no estudo da **Subseção 3.3.1** é possível observar que a elasticidade em infraestrutura possui certa limitação. Isto ocorre porque os mecanismos de monitoramento de desempenho possuem acesso a apenas informações da infraestrutura, não havendo interação com a aplicação ou plataforma. Tal limitação implica no desenvolvimento de mecanismos genéricos de controle de elasticidade. Nessa subseção, são analisadas soluções de plataforma e aplicação.

(KRANAS *et al*, 2012) propõe um arcabouço genérico para o gerenciamento de elasticidade em qualquer ambiente ou aplicação para computação em nuvem, introduzindo o conceito de elasticidade como serviço. O trabalho descreve uma arquitetura e seus componentes, porém infelizmente não implementa qualquer protótipo para validar a proposta, além de não propor soluções de como descrever as regras para o aumento ou diminuição de escalonamento.

(MARSHALL *et al*, 2010) apresenta uma solução interessante para elasticidade de aplicações Web que utilizam o modelo cliente-servidor. Isto é feito por meio da adaptação de serviços tradicionais como agendamento do processamento de lotes de dados, armazenamento de dados e serviços Web. Seu mecanismo principal de elasticidade tem como conceito chave o monitoramento de uma fila de tarefas a serem executadas pelo cluster de recursos. Assim, toda vez

que a fila de tarefas começa a aumentar, um gerenciador de recursos adiciona novas instâncias de processamento ao cluster.

O trabalho de (CALHEIROS *et all*, 2012) descreve a solução Aneka, uma plataforma que prove um *middleware* para o desenvolvimento, implementação e gerenciamento de aplicações em nuvens públicas e privadas. Seu *middleware* é uma das primeiras abordagens a utilizar um arcabouço para contêineres orientado a serviço que permite a utilização de diferentes modelos de programação, como *Thread*, *Task* e *MapReduce*.

O Google App Engine[16] é uma solução de plataforma que permite a criação de aplicações do tipo cliente-servidor, nas linguagens Java, Python e PHP. A plataforma provê serviços de elasticidade de forma transparente para os desenvolvedores, mas não são fornecidos detalhes técnicos de sua arquitetura, implementação ou funcionamento interno.

O Microsoft Azure[17] é uma solução de plataforma para o desenvolvimento de aplicações Web do tipo cliente-servidor, e também disponibiliza serviços de escalabilidade de forma transparente para seus desenvolvedores. A solução permite o desenvolvimento de aplicações nas linguagens .NET, Java, PHP, Node.js, Python e Ruby. Novamente, nenhum tipo de detalhes sobre arquitetura, organização dos módulos ou implementação são disponibilizados.

Embora diversas soluções com diferentes abordagens tenham sido propostas, desafios continuam em aberto na implementação da elasticidade para computação em nuvem. (GALANTE; BONA, 2012) propôs cinco desafios a serem superados, entre eles: a disponibilidade de recursos por cliente (provedores de nuvem ainda limitam a quantidade máxima disponível por cliente), a interoperabilidade entre nuvens (falta de APIs padronizadas), a granularidade dos recursos (muitas vezes as configurações das máquinas virtuais não atendem às necessidades dos usuários da nuvem), o tempo de inicialização dos recursos e a disponibilidade de ferramentas e plataformas para o desenvolvimento de aplicações.

---

[16] https://developers.google.com/appengine/
[17] http://azure.microsoft.com/pt-br/

Destacando o último desafio, a necessidade de ferramentas e plataformas para o desenvolvimento de aplicações elásticas, argumenta-se que as ferramentas e arcabouços desenvolvidos são limitados a um tipo específico de aplicação: as baseadas em uma arquitetura cliente-servidor. Segundo (GALANTE; BONA, 2012), são necessárias plataformas e arcabouços que simplifiquem o desenvolvimento e implementação de aplicações elásticas e que levem em consideração características como paralelismo (e.g., **distribuição** e compartilhamento de memória), tipo de processamento (interativo, ou **tempo real**, e massivo, ou *batch*) e tipo de aplicação (alto desempenho, científico, **multimídia** e negócios). Os termos destacados nesse desafio são aqueles que são o foco da solução proposta nessa dissertação de mestrado.

3.4 Aplicações Multimídia no contexto de Computação em Nuvem

No final da **seção 3.3**, foram destacados alguns desafios em relação às soluções de elasticidade para o contexto de nuvens de computação. Um desses desafios é a expansão do domínio de aplicações de elasticidade para outros tipos de aplicação além do modelo cliente-servidor. Dentre os domínios sugeridos por (GALANTE; BONA, 2012) estão as aplicações distribuídas, aplicações em tempo real e aplicações multimídia. As aplicações multimídia, em especial, possuem uma característica distinta das soluções mais comuns, que se refere à eficiência na utilização dos recursos de rede ao invés de priorizar os recursos de processamento e armazenamento. Nesta seção é apresentada uma breve discussão sobre a relevância dos serviços multimídia e das Redes de Fornecimento de Conteúdo (do inglês, *Content Delivery Network* – CDN), e alguns dos benefícios em adaptar tal tipo de solução para o contexto de nuvens de computação.

3.4.1 *Content Delivery Network*s e Elasticidade na Nuvem

De acordo com (SANDVINE, 2013) a maior parte do conteúdo trafegado em 2013 pela Internet pode ser classificada como “Entretenimento em Tempo Real”. Nos Estados Unidos, mais de 68% do tráfego de *downstream* registrado se refere a esta categoria de tráfego (SANDVINE, 2013). Em outras palavras, 68% do tráfego de

downstream estado-unidense em horário de pico de 2013 foi gerado por aplicações e protocolos que permitem entretenimento sob demanda, como distribuição de fluxos de áudio e vídeo (e.g., protocolos RTSP, RTP, RTMP, Flash Video, MPEG; serviços como Netflix, Hulu, YouTube, Pandora).

Devido à crescente demanda mundial por este tipo de conteúdo e aos estritos requisitos exigidos, como sensibilidade ao tempo de entrega e largura de banda, (especialmente para aplicações de distribuição de fluxo de vídeo pela Internet), muitos provedores de conteúdo utilizam os serviços das chamadas Redes de Fornecimento de Conteúdo (do inglês, *Content Delivery Network* – CDN) (ZHUANG; GUO, 2012). O objetivo dos serviços prestados pelos provedores de CDN é a rápida entrega de fluxos de vídeo para os usuários finais, mantendo sua qualidade (redução de perda de pacotes e atendimento das requisições feitas pelos usuários finais) (ZHUANG; GUO, 2012). Este tipo de solução é o foco da presente seção.

(HOFMANN; BEAUMO, 2005, p. 21) propõem uma organização dos componentes funcionais de uma CDN, destacando quatro serviços. A **distribuição do conteúdo**, o **roteamento de requisições**, o **processamento dos conteúdos** e a **autorização, autenticação e contabilização**. A distribuição de conteúdo envolve *web caching* (armazenamento temporário em diferentes pontos da rede), assim como mecanismos e protocolos de transmissão de dados pela rede. O roteamento de requisições leva em consideração aspectos como a localização dos usuários e a disponibilidade dos sistemas e da rede. O processamento de conteúdo se trata de modificações no conteúdo original para atender a diferentes preferências do usuário e a adaptação a restrições de banda de rede ou capacidade dos dispositivos de visualização (e.g., transcodificação para diferentes formatos de vídeo, compressão, entre outros). Por fim, a autorização, autenticação e contabilização são responsáveis por serviços que permitam o monitoramento, o registro de atividades e a contabilização da utilização do conteúdo.

Dentre os serviços de CDN implementados em nuvem de computação, (ZHUANG; GUO, 2012), (KO *et al*, 2013), (MA *et al*, 2014) e (CHENG, 2014) (de CLOUD2014) tratam de serviços de processamento de conteúdo, desde a transcodificação para diferentes formatos, até a implementação de serviços como a inserção de legendas e linguagem de sinais. (LIANG, 2012) implementa a

transcodificação com processamento distribuído utilizando o arcabouço *Hadoop MapReduce* para adicionar certo nível de paralelismo na solução. (MIKITYUK *et al,* 2013) propõe a virtualização das *Set Top Boxes* (STBs), movendo parte do ambiente de execução de seus serviços para a intraestrutura da nuvem. (LIU *et al*, 2014) indica problemas de fluxos redundantes de vídeo para serviços de vídeo sob demanda devido a opções de gerenciamento de memória em aparelhos baseado nos sistemas operacionais iOS (proprietário da Apple); como solução, é proposta a virtualização do serviço de requisição de fluxos de vídeo para a nuvem.

Como pode ser observado nas soluções levantadas combinando serviços de vídeo e nuvens, o contexto de aplicações para CDNs é interessante do ponto de vista de elasticidade para nuvens de computação, pois o gerenciamento e a elasticidade dos recursos de rede passam a ter maior relevância em relação aos recursos de processamento ou armazenamento de dados (cenário diferente das soluções apresentadas na **Seção 3.3**). Neste contexto, o processamento e o armazenamento auxiliam a diminuir o tráfego de dados na rede. Outra característica interessante nos serviços de nuvem para CDN é a importância da localidade, ou seja, garantir a proximidade com os usuários finais é essencial para a eficiência de serviços de *caching* e, como indicado nos trabalhos atuais, no pré-processamento de fluxos de vídeo.

Tendo em vista a importância desse cenário, na **Seção 3.5** é apresentada uma solução distribuída de fornecimento de fluxos de vídeo com serviço de redução de fluxos redundantes que utiliza recursos de nuvem. Por seu caráter distribuído, é realizada uma análise detalhada de suas características a fim de identificar serviços de plataforma diferentes dos comumente implementados em soluções de elasticidade.

3.5 Wind: Uma Aplicação Distribuída para Distribuição de Fluxos de Vídeo em Tempo Real

O Wind é uma aplicação distribuída de fornecimento de fluxos de vídeo desenvolvida em meados de 2010, pela área de pesquisa e inovação da empresa sueca Ericsson Telecomunicações S.A., pelos pesquisadores Stefan Hellkvist e

Joacim Halén. Por se tratar de uma solução interna, não há registros públicos que a documentem disponíveis para consulta. O acesso aos relatórios de detalhamento da solução se deu através de um programa de colaboração entre o Laboratório de Arquitetura e Redes de Computadores (LARC) do Departamento de Engenharia da Computação e Sistemas Digitais (PCS) da Escola Politécnica (EP) da Universidade de São Paulo (USP) e área de pesquisa e inovação da Ericsson. A presente dissertação de mestrado é resultado dessa colaboração, onde o autor trabalhou em cooperação com outros pesquisadores na extensão do sistema Wind. Esta solução se diferencia das demais aplicações de distribuição de fluxos de vídeo em tempo real por sua arquitetura distribuída e pela implementação de um mecanismo automático de redução de fluxos redundantes, utilizando recursos de computação em nuvem.

A fim de contextualizar o problema dos fluxos redundantes é apresentada a seguir uma breve descrição do problema e o histórico das técnicas desenvolvidas para solucioná-lo. Os leitores que já tiverem familiaridade com o tema podem ignorar esta discussão e seguir para a **Seção 3.5.2**.

3.5.1 O Problema dos Fluxos Redundantes

Com o crescimento do número de aplicações do tipo um-para-muitos (e.g., fornecimento de vídeo sob demanda e fluxos de vídeo em tempo real), a estrutura da Internet, inicialmente projetada para os serviços do tipo um-para-um (e.g., aplicações de transferência de arquivos e envio de *e-mails*), impôs problemas de ineficiência de utilização dos recursos disponíveis (HOSSEINI *et al*, 2007). O problema é que, em aplicações de fornecimento de fluxos de vídeo em tempo real, um mesmo conteúdo é transmitido para todos os usuários requisitantes no mesmo instante, gerando um tráfego redundante que poderia ser reduzido em boa parte do caminho percorrido pelos fluxos de dados. A **Figura 3** ilustra o problema dos fluxos redundantes quando dois usuários requisitam o mesmo conteúdo de vídeo. O fluxo requisitado pelo cliente 1 é representado pela linha contínua e o fluxo requisitado pelo cliente 2 pela linha tracejada. Nessa figura, é possível observar fluxos redundantes no caminho entre o provedor do conteúdo e o roteador A e também no trecho da rede entre os roteadores A e C: em ambos os casos seria necessária a

transmissão de apenas um dos fluxos para atender ambos os clientes. Este problema é ainda pior em serviços com muitos usuários, pois quanto maior o número de clientes requisitantes, potencialmente maior a quantidade de fluxos redundantes sendo transmitida pela rede, reduzindo-se a eficiência do uso de recursos de rede..

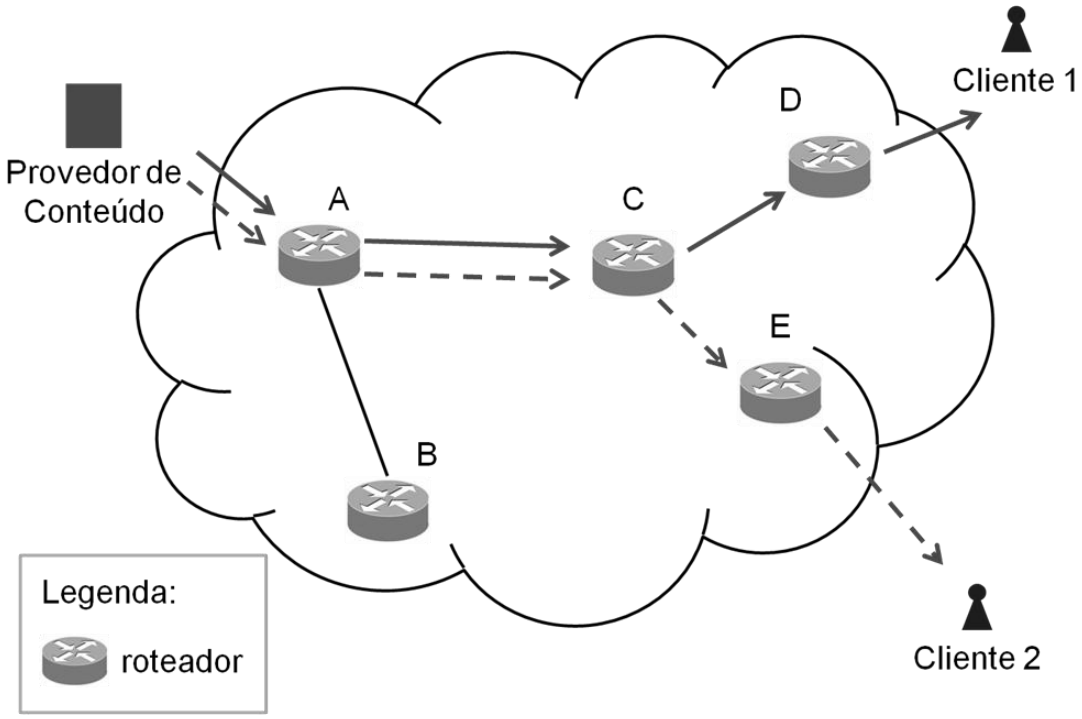

Figura 3: Fornecimento de fluxos redundantes

A fim de solucionar tal problema, foi proposto por (DEERING; CHERITON, 1990) a técnica de IP *Multicast* que descrevia as extensões necessárias em algoritmos de roteamento para que fosse possível a utilização da transmissão um-para-muitos para além das redes locais. Em resumo, a transmissão *multicast* permite que seja enviados os fluxos de dados para todos os clientes requisitantes, por meio do envio de apenas um fluxo por parte do provedor dos dados. Como ilustrado na **Figura 4**, o fluxo enviado pelo provedor de conteúdo pode ser transmitido uma única vez para o roteador A, que encaminha o fluxo ao roteador C, e este fica responsável pela replicação e encaminhamento aos roteadores D e E (representados pelas linhas contínua e tracejada, respectivamente).

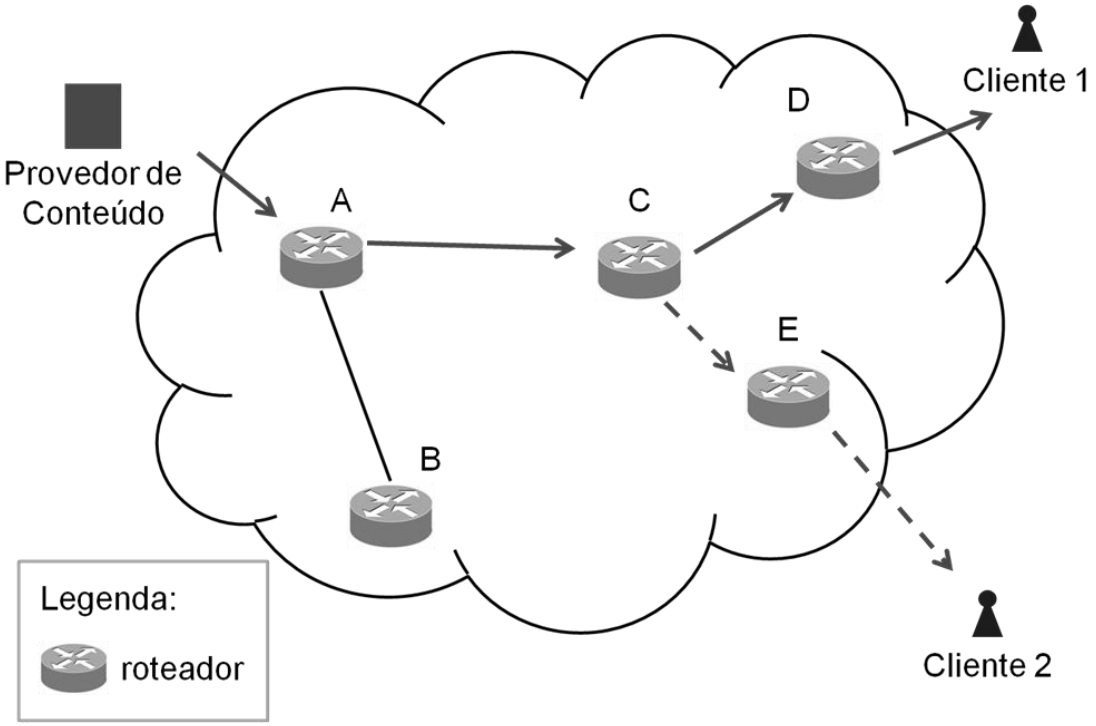

Figura 4: Fornecimento de fluxo otimizado com a transmissão m*ulticast*

Apesar da comprovada eficiência do IP *Multicast,* a solução não foi amplamente adotada devido a problemas técnicos, administrativos e de negócios (HOSSEINI *et al*, 2007). Com o intuito de contornar tais problemas, diversas soluções foram propostas, sendo uma categoria delas chamada de *Multicast* em Camada de Aplicação (do inglês, *Application Layer Multicast* – ALM). (HOSSEINI *et al*, 2007) fez um levantamento de soluções de ALM e a descreve como a implementação da técnica *multicast* em camada de aplicação, ao invés da implementação como um serviço da camada de rede.

A **Figura 5** ilustra um exemplo de técnica de *multicast* em camada de aplicação. Os nós da topologia fazem o papel dos roteadores na técnica de IP *Multicast*, mas através de serviços na camada de aplicação.

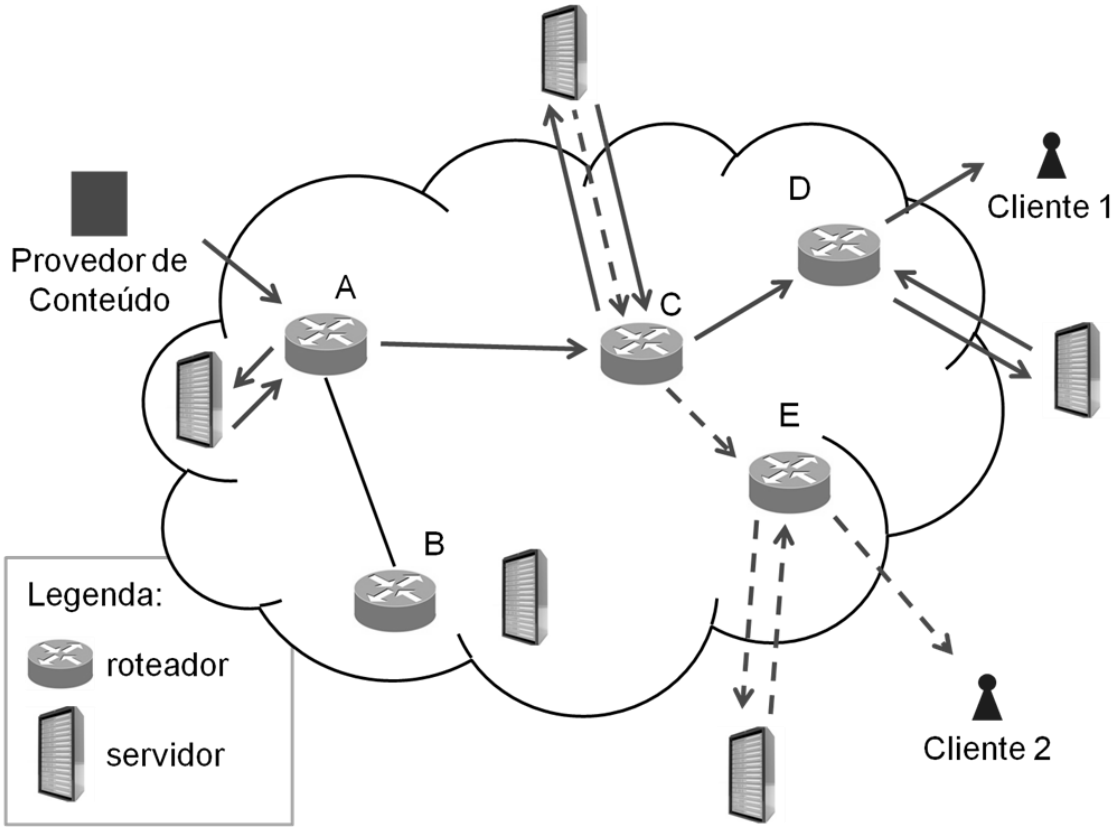

Figura 5: Fornecimento de fluxo otimizado com a técnica de *multicast* em camada de aplicação (nós externos à rede)

Este tipo de técnica encontra-se no cerne do Wind. Especificamente para realizar a redução de fluxos redundantes, o Wind implementa uma técnica de *multicast* em camada de aplicação combinada com a virtualização de recursos. Os nós de aplicação são instanciados automaticamente quando são detectados fluxos redundantes e nós subutilizados também são liberados de forma automática. Tal característica permite que os recursos da nuvem sejam alocados de acordo com a demanda dos clientes, conforme discutido mais detalhadamente a seguir.

3.5.2 Visão Geral do Sistema

A solução Wind funciona como uma CDN para distribuição de fluxos de vídeo em tempo real e, portanto, recebe um fluxo de vídeo de um provedor de conteúdo e deve distribuí-lo através de uma infraestrutura em diferentes localidades para os clientes requisitantes, garantindo a qualidade do fluxo de vídeo original. O Wind pode ser classificado como uma solução de elasticidade para nuvens de

computação implementada na camada de aplicação, cujo objetivo é aumentar a eficiência dos recursos de rede.

Diferentemente das soluções analisadas nas **Seções 3.3** e **3.4**, o Wind executa o controle de elasticidade e o monitoramento de métricas de forma distribuída, ou seja, todos os nós de aplicação monitoram eventos e qualquer um deles pode tomar a decisão pela alocação/liberação de recursos. A principal métrica monitorada pela aplicação é a quantidade de requisições para um mesmo fluxo de vídeo. Caso a quantidade exceda duas requisições (existência de fluxo redundante), o mecanismo verifica a possibilidade de otimização após calcular o maior caminho comum entre o nó que identificou a redundância e os nós requisitantes. Encontrado o ponto da topologia indicado como o último nó comum entre o nó otimizador e os nós requisitantes, é instanciado um novo nó e as requisições são redirecionadas para o nó recém-criado, que requisita o fluxo para o nó otimizador.

De acordo com a classificação definida na **Seção 3.2**, o Wind pode ser categorizado como uma solução de escopo em aplicação (SaaS). O sistema utiliza uma política automática reativa (baseada em regras) para a replicação de recursos com o propósito de aumentar a eficiência na distribuição de fluxos de vídeo em tempo real.

3.5.3 Arquitetura

A solução é organizada em uma arquitetura composta de cinco diferentes componentes de aplicação: o **Tail Node**, o **Intermendiary Node**, o **Head Node**, o **Statistics Server** e o **Visualizer**. A **Figura 6** ilustra a organização da arquitetura com base nos três nós descritos a seguir:

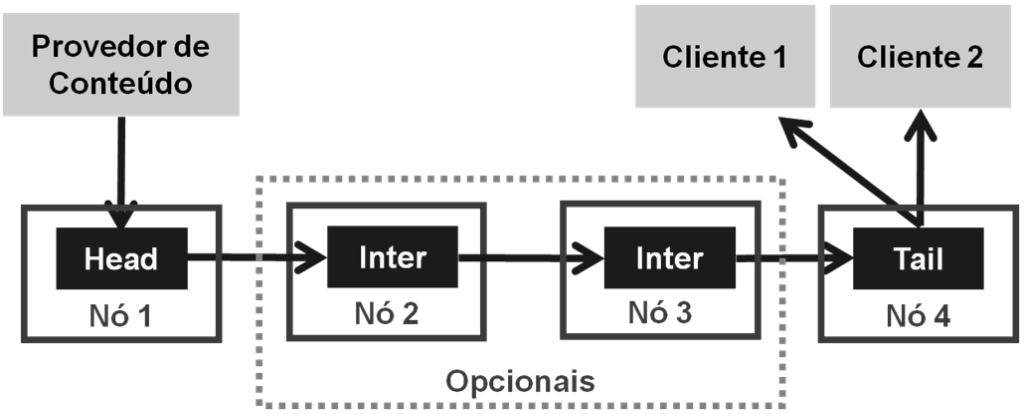

Figura 6: Interação entre os nós da arquitetura do *Wind* para distribuição de um fluxo de vídeo

- O **Tail Node** é o nó responsável por receber as requisições de fluxo dos clientes, encaminhando-as para um Tail Node ou Intermediary Node. Quando esse nó recebe um fluxo de vídeo, ele o encaminha para os clientes requisitantes;

- O **Intermediary Node** é um tipo de nó que recebe requisições de fluxo de nós do tipo Tail Node ou mesmo de Intermediary Nodes e as encaminha para os nós do tipo Head Node ou Intermediary Node. Ao receber um fluxo de vídeo, esse nó o encaminha para nós do tipo Intermediary Node ou Tail Node;

- O **Head Node** é um tipo de nó que recebe requisições de fluxo de nós do tipo Tail Node ou Intermediary Nodes e requisita um fluxo de vídeo do provedor de conteúdo. Ao receber um fluxo do conteúdo do provedor de conteúdo, esse nó encaminha os dados para nós do tipo Intermediary Node ou Tail Node.

Além dos nós responsáveis pelo tratamento das requisições existem dois tipos de nós que cumprem papéis importantes na arquitetura: o **Statiscs Server** e o **Visualizer**, detalhados a seguir.

- O **Statistics Server** é um servidor responsável por manter atualizadas as informações sobre o estado atual da aplicação, indicando quais nós da topologia possuem instâncias ativas;

- O **Visualizer** é uma interface gráfica que permite monitorar, em tempo real, o tráfego gerado pela aplicação na topologia de rede.

Além do encaminhamento de requisições e fluxos, os nós Intermediary Node e Head Node também são responsáveis por verificar a quantidade de requisições a um mesmo fluxo de vídeo; caso haja mais de uma requisição, o mecanismo verifica a possibilidade de redução dos fluxos redundantes e instancia um novo nó de aplicação do tipo Intermediary Node numa posição da topologia que otimiza a distribuição o fluxo utilizando o algoritmo de caminho comum mais longo.Como parâmetros desse algoritmo, são indicados a topologia configurada através de um arquivo de especificações e o estado atual da topologia de rede formada pelas das instâncias ativas.

3.5.4 Algoritmo de Caminho Comum Mais Longo

Para identificar um possível ponto de otimização da transmissão do fluxo na topologia, o nó que identifica uma condição de fluxo redundante executa o algoritmo chamado de caminho comum mais longo (do inglêss, *Longest Common Path* – LCP). Esse algoritmo foi desenvolvido para a utilização em um tipo específico de topologia em que, a partir de qualquer nó, é possível a construção um grafo conexo e acíclico (árvore) para sua representação. O cálculo do caminho mais longo em árvores foi resolvido por volta de 1960, pelo matemático Edsger W. Dijkstra (BULTERMAN *et al*, 2002), mas para implementação do algoritmo de caminho comum mais longo foi utilizada um técnica simples baseada no cálculo do caminho mais curto (do inglês, *Shortest Path*) por meio do algoritmo de Dijkstra, descrito por (DIJKSTRA, 1959). O algoritmo de caminho comum mais longo é descrito a seguir.

Dados um nó de origem O e (pelos menos) e um conjunto de nós de destino {D1, D2 ... D*n*}, sendo *n* a quantidade de nós de destino escolhidos, em uma árvore, o cálculo do caminho comum mais longo pode ser feito através do seguinte algoritmo:

1) Para cada nó de destino D*x*, calcular o menor caminho entre o nó de origem O e D*x*, gerando o caminho C*x*;

2) Comparar todos os caminhos C*x* encontrados em cada iteração do passo 1, a partir da origem O, à procura do último nó comum U. A sequência de nós entre a origem O e o último nó comum U é a solução do problema.

No caso do Wind, as reduções de fluxo redundante são executadas a partir de pares de nós. A **Figura 7** ilustra uma árvore na qual se deseja achar o caminho comum mais longo entre os nós 1, 4 e 5, sendo 1 o nó a origem e os nós 4 e 5 os nós de destino. Nas ilustrações, tem-se que: os nós de borda pontilhada são os nós de origem, os nós de borda tracejada são os nós de destino; e os nós de cor cinza são os nós que fazem parte do caminho especificado. A **Figura 7(a)** indica o caminho mais curto entre os nós 1 e 4, composto pelos nós 1, 3 e 4; a **Figura 7(b)**, indica o caminho mais curto entre os nós 1 e 5, composto pelos nós 1, 3 e 5; a **Figura 7(c)** indica o caminho comum mais longo em relação às figuras **Figura 7(a)** e **Figura 7(b)**, composto dos nós 1 e 3.

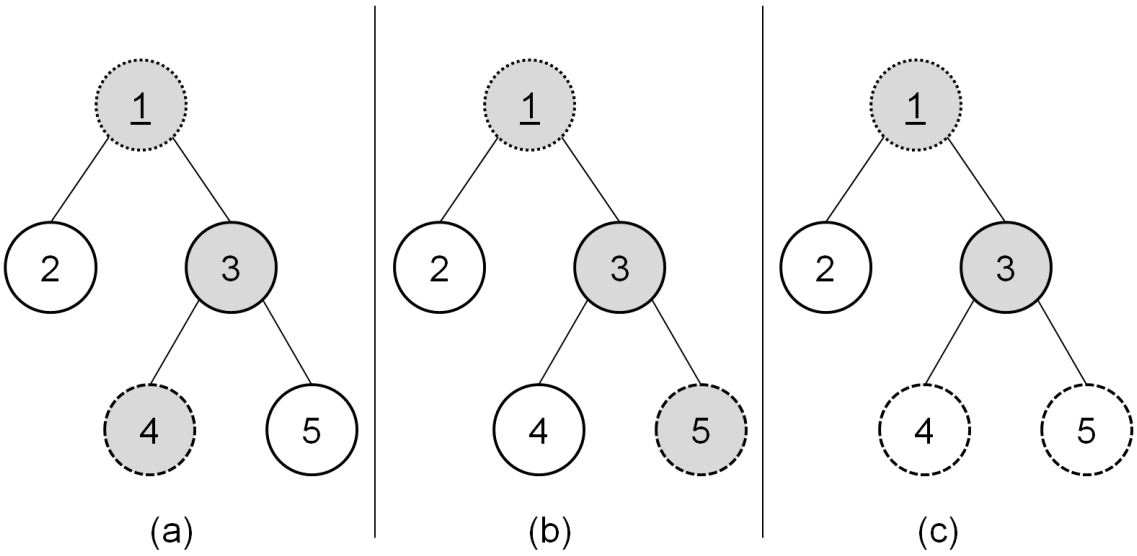

Figura 7: (a) Caminho mais curto entre os nós 1 e 4; (b) caminho mais curto entre os nós 1 e 5; (c) maior caminho comum entre os nós 1, 4 e 5

No exemplo ilustrado pela **Figura 7**, sendo 1 o nó que detectou o fluxo redundante e 4 e 5 os nós requisitantes, o nó 3 seria indicado como ponto de otimização entre os nós 4 e 5 por ser o último nó do caminho comum mais longo.

3.5.5 Detalhes de Implementação e Configurações

Na versão avaliada nesta pesquisa, o Wind suportava somente a distribuição de fluxos de vídeo utilizando o protocolo de transferência de hipertexto (do inglês, *Hypertext Transfer Protocol* – HTTP). A fim de habilitar o mecanismo de redução de fluxos redundantes em camada de aplicação, foi realizado um tratamento no fluxo original inserido na aplicação.

Este tratamento se trata da fragmentação do fluxo original em unidades de tamanho padronizado, chamados de blocos ou *chunks*. Na implementação do Wind o tamanho padrão de um bloco é de aproximadamente 100 kilobytes e cada bloco é identificado por um número sequencial (iniciado em 1) gerado na transformação do fluxo em blocos. A seguir são descritas as responsabilidades dos componentes descritos na **Subseção 3.5.3** no processo de transformação do fluxo de vídeo.

- **Head Node**: responsável por encapsular um fluxo de vídeo em blocos numericamente identificados. Como ilustrado na **Figura 8**, o fluxo de vídeo recebido pelo Head Node é fragmentado em blocos de tamanho padronizado e cada bloco é identificado. Tal processo possibilita ao mecanismo de distribuição detectar se um mesmo bloco de conteúdo está sendo requisitado por diferentes nós (situação de fluxo redundante).

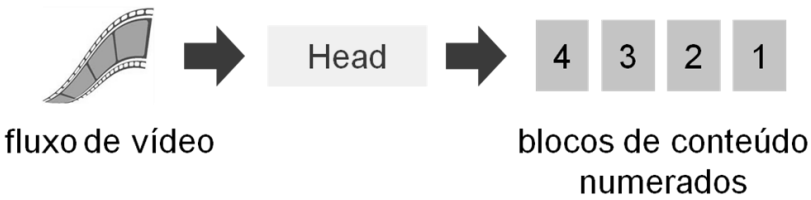

Figura 8: Transformação do fluxo de vídeo em blocos numerados

- **Intermediary Node**: sua principal função é o repasse dos blocos de conteúdo para outros nós. A **Figura 9** ilustra a operação de repasse dos blocos, sem qualquer processo de transformação.

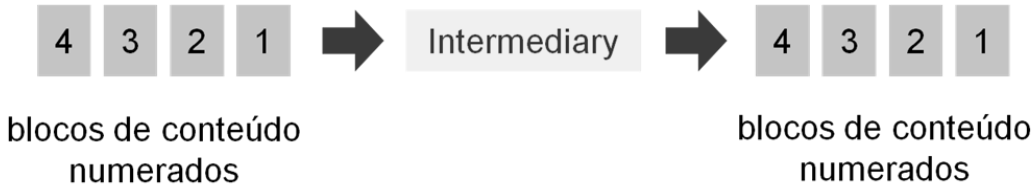

Figura 9: Encaminhamento de blocos

- **Tail Node**: realiza a operação inversa à do nó Head Node, reconstituindo os blocos numerados novamente em um fluxo de vídeo que pode ser lido pelas aplicações de destino. Na **Figura 10** é ilustrado o processo de transformação da sequência de blocos numerados em um fluxo de vídeo.

Figura 10: Reconstituição do fluxo de vídeo

3.5.6 Análise do Wind e Oportunidades de Melhorias

A solução Wind apresenta uma série de características inovadoras como a implementação do *multicast* em camada de aplicação utilizando recursos de nuvem para garantir a elasticidade do sistema. Dentre as soluções de elasticidade avaliadas neste capítulo, o Wind foi a única solução a apresentar um mecanismo de gerenciamento distribuído, por meio do qual os diversos nós da nuvem monitoram as métricas de elasticidade e também atuam em tarefas de controle, instanciando novos nós quando necessário.

Uma observação interessante na solução Wind é a simplicidade da regra de otimização monitorada pelo mecanismo. Por se tratar de uma solução específica desenvolvida em camada de aplicação (i.e., para um domínio bem definido), a métrica que leva em consideração o número de requisições redundantes simplifica o gerenciamento dos recursos de rede através de uma informação de alto nível. Caso o mesmo tipo de controle fosse realizado por uma solução de infraestrutura ou de

plataforma seria mais difícil identificar um fluxo redundante através da definição de regras.

Além dos pontos inovadores observados no Wind, também foram identificadas oportunidades de melhoria da solução. A seguir, são listadas e comentadas as principais oportunidades identificadas, as quais são exploradas na presente dissertação.

- O Wind é uma solução específica que implementa um modelo de gerenciamento de elasticidade que poderia ser expandido para outros domínios de aplicação. Para isso, é fundamental a identificação e separação das funcionalidades específicas de aplicação das funcionalidades do mecanismo de gerenciamento de recursos e elasticidade;

- Na solução Wind, o gerenciamento de informações da topologia e das instâncias ativas é feito de maneira centralizada; a configuração da topologia de recursos é feita de forma estática através de um arquivo de configuração. Seria possível realizar o gerenciamento dessas informações de forma distribuída e dinâmica, além de permitir a inclusão de novas informações como: topologia da nuvem, topologia de recursos alocados, configurações das aplicações, informações sobre usuários e suas instâncias de aplicação;

- Definidas as funcionalidades genéricas, seria possível desenvolver um *middleware* e disponibilizar uma API para acesso aos serviços de gerenciamento de recursos e elasticidade. Tal solução poderia ser expandida com a inserção de novos serviços e serviços comumente implementados para nuvens de computação (como as tradicionais aplicações cliente-servidor com elasticidade de processamento e armazenamento).

4 A SOLUÇÃO TRADE WIND

A partir do levantamento da literatura apresentado no **Capítulo 3**, foram apresentadas e analisadas soluções de gerenciamento de recursos visando a elasticidade de aplicações no ambiente de computação em nuvem. Também foram discutidos os desafios referentes à expansão dos domínios de aplicação cobertos pelas soluções de elasticidade, bem como a relevância do domínio de aplicações multimídia, em especial nos serviços de fornecimento de fluxos de vídeo. Assim, evidenciou-se o foco das aplicações multimídia e ferramentas de elasticidade nas operações de processamento de vídeo, mas a falta de soluções que permitissem o gerenciamento automático dos recursos de rede.

A partir do estudo e análise da aplicação Wind, foram observadas características da solução que poderiam ser generalizadas para sistemas de gerenciamento de elasticidade através de uma nova abordagem. Separadas as funcionalidades genéricas das específicas, análise esta aliada ao levantamento de serviços comuns às soluções de gerenciamento de elasticidade, foi vislumbrada a possibilidade de desenvolvimento de um *middleware* e a disponibilização de uma API para ao acesso aos serviços de gerenciamento de recursos e elasticidade. Tal solução poderia ser expandida com a inserção de novos serviços.

Este capítulo tem por objetivo apresentar uma análise de funcionalidades e serviços desejáveis para o desenvolvimento de aplicações elásticas em nuvens de computação, a partir dos estudos das soluções das **Seções 3.3** e **3.5**. Com base nesses estudos, é apresentado o Trade Wind, uma solução distribuída para o gerenciamento de elasticidade em nuvens de computação. Especificamente, são definidos uma arquitetura, um *middleware* de serviços de elasticidade e uma API para a utilização de seus serviços. Por fim, são apresentadas as adaptações necessárias na aplicação Wind para sua utilização junto à solução de elasticidade proposta.

4.1 Uma Proposta de Modelo para Desenvolvimento de Aplicações Elásticas

Na **Seção 3.3** foram apresentadas e avaliadas diversas soluções para implementação de elasticidade em nuvens de computação. Dentre as diversas características e classificações destacaram-se o número de soluções que utilizam estratégias de elasticidade em camada de infraestrutura (IaaS) utilizando políticas automáticas reativas. Um conceito comum a essas soluções é a adoção de regras para a automatização das ações de alocação//liberação de recursos da nuvem. Nesta seção são analisadas as diferenças de abordagem da solução Wind e de outras plataformas em relação aos mecanismos de elasticidade. Ao final, é proposto um modelo de desenvolvimento de serviços elásticos e são apresentados exemplos de implementação de elasticidade a partir do modelo proposto.

4.1.1 Elasticidade Automática Reativa – Monitoramento e Controle

As regras de política têm como principal objetivo definir um conjunto de ações a serem executadas caso uma determinada condição do sistema seja atingida. Para modelar essa condição devem ser definidas métricas que são monitoradas junto aos recursos da nuvem. As regras de política apresentam-se no formato geral: **regra**: ***condição → ação***

A condição é composta de valores limite para métricas e a ação determina o conjunto de ações de alocação/liberação de recursos e possíveis reconfigurações na nuvem. No exemplo a seguir é descrita uma regra, na qual a métrica monitorada é a porcentagem de utilização do processador.

- **condição**: caso a porcentagem de utilização do processador ultrapasse o valor de 80%;
- **ação**: alocar mais uma máquina virtual para a tarefa.

Uma prática comum para a implementação deste tipo de elasticidade baseada em regras (automática reativa) é o desenvolvimento de dois mecanismos: um de monitoramento das métricas e outro para execução das ações de elasticidade junto à intraestrutura da nuvem. O **monitor de métricas** deve receber uma descrição das

condições definidas pelo usuário da nuvem e disparar eventos ao **controlador de elasticidade** para alteração da alocação de recursos da infraestrutura. Com base nesses dois mecanismos são apresentadas duas principais diferenças do (Trade) Wind para as soluções analisadas na **Seção 3.3.**

A primeira se trata da camada onde as métricas são definidas e monitoradas. O modelo de monitoramento de regras e atuação, que é comumente implementado de forma genérica, foi implementado de forma específica. Nas soluções tradicionais, as regras são definidas na camada de infraestrutura (IaaS) ou plataforma (PaaS), enquanto o (Trade) Wind as define em camada de aplicação (SaaS). No caso dos fluxos redundantes, **a regra específica desta solução se baseia em uma informação de aplicação** (a quantidade de requisições a um mesmo conteúdo), que não poderia ser monitorada com a mesma facilidade de definição e precisão por um mecanismo genérico de plataforma (PaaS) ou infraestrutura (SaaS).

A segunda diferença se deve ao caráter distribuído da solução Wind, trazendo a necessidade do **gerenciamento de informações** que devem estar acessíveis a todos os nós. Assim o (Trade) Wind necessita de informações sobre a topologia de recursos disponíveis e recursos já alocados pela aplicação, além de algoritmos específicos que efetuam a verificação das condições para a execução das ações de elasticidade junto à infraestrutura. Qualquer nó que tenha como objetivo otimizar um fluxo redundante detectado localmente verifica estas informações junto a um servidor que centraliza estas informações.

4.1.2 Modelo de Desenvolvimento de Serviços Elásticos

A estratégia de definição e monitoramento de regras em camada de aplicação implementada pelo Wind é capaz de reduzir fluxos redundantes simplificando complexidades encontradas em estratégias implementadas em camada de infraestrutura (e.g., definir regras para a identificação de fluxos redundantes apenas monitorando enlaces de rede). Porém, o mecanismo de controle da elasticidade do Wind original também é implementado na aplicação, impossibilitando sua reutilização em outras aplicações devido ao forte acoplamento das funções de gerenciamento/otimização de recursos e da aplicação de redução de fluxos

redundantes. Assim, uma vantagem do Trade Wind é o fato de ser genérico o suficiente para utilização em diversas aplicações, dando suporte às funcionalidades comumente utilizadas para o desenvolvimento de aplicações elásticas.

Como pode ser observado no levantamento da **Seção 3.3**, um dos serviços mais triviais para as soluções de gerenciamento de elasticidade em nuvens é a manipulação de máquinas virtuais (MV). Além da criação e destruição das MVs por meio das ferramentas de virtualização, há também a questão da configuração de conectividade, da instanciação e configuração das aplicações.

Para o desenvolvimento de uma solução que atenda de forma eficaz às necessidades de elasticidade de uma ampla gama de aplicações, sem a necessidade de reimplementação das funções comumente utilizadas, o Trade Wind reorganiza a implementação da solução Wind original através de um modelo de desenvolvimento de serviços elásticos. Utilizando este modelo para é possível modularizar o desenvolvimento de aplicações elásticas, separando as funcionalidades específicas (a serem implementadas pelos desenvolvedores de aplicações) e as que deveriam ser fornecidas em forma de um *middleware* (funcionalidades comuns às aplicações elásticas). Na **Tabela 2**, o modelo é apresentado, indicando os mecanismos responsáveis pela elasticidade automática encontrados nas soluções revisadas na **Seção 3.3**, as funcionalidades a serem implementadas para cada aplicação e a reorganização das camadas onde devem ser implementados os mecanismos.

Mecanismo	Modelo de Desenvolvimento de Serviços Elásticos	Camada de Implementação
Monitor de métricas e regras	Definição e monitoramento de métricas e condições das regras	Aplicação (SaaS)
Monitor de métricas e regras	Verificação de possibilidade de otimização	Aplicação (SaaS)
Controlador de elasticidade	Gerenciamento de informações sobre recursos e aplicações	Plataforma (PaaS)
Controlador de elasticidade	Execução de ações de elasticidade (manipulação de máquinas virtuais)	Plataforma (PaaS)

Tabela 2: Modelo de desenvolvimento de serviços elásticos proposto

4.1.3 Exemplos de Aplicações Utilizando o Modelo

Na aplicação Wind detalhada anteriormente na **Seção 3.5**, a aplicação de redução de fluxos redundantes pode ser classificada como uma aplicação elástica. Na **Tabela 3** são apresentadas as funcionalidades do serviço de redução de fluxos redundantes adaptadas ao modelo de desenvolvimento proposto na **Subseção 4.1.2**. É interessante observar a clara separação entre funcionalidades específicas, descritas nas duas primeiras linhas da **Tabela 3**, e as funcionalidades genéricas, descritas nas duas últimas linhas da mesma tabela.

Modelo de Desenvolvimento de Serviços Elásticos	Serviço de Redução de Fluxos Redundantes
Definição e monitoramento de métricas e condições das regras	Monitor de requisições redundantes
Verificação de possibilidade de otimização	Algoritmo de cálculo do caminho comum mais longo
Gerenciamento de informações sobre recursos e aplicações	Topologia de recursos disponíveis e recursos alocados pela aplicação
Execução de ações de elasticidade (manipulação de máquinas virtuais)	Criação de máquina virtual,configuração de conectividade e da aplicação

Tabela 3: Funcionalidades do serviço de redução de fluxos redundantes de acordo com o modelo

Utilizando como exemplo o desenvolvimento de um serviço de posicionamento dinâmico de servidores de *cache*, são propostas as funcionalidades de monitoramento da quantidade de requisições a um conteúdo associada à localidade. Definida a condição de gatilho como quantidade mínima de requisições para uma mesma localidade, é calculada a localidade mais próxima das requisições e, se possível, instanciado um servidor de *cache* para o conteúdo no *host* disponível. Na **Tabela 4** são apresentadas as funcionalidades do serviço de Posicionamento Dinâmico de Servidores de *Cache* a partir do modelo.

Modelo de Desenvolvimento de Serviços Elásticos	Serviço de Posicionamento Dinâmico de Servidores de *Cache*
Definição e monitoramento de métricas e condições das regras	Monitor de Quantidade de Requisições por Localidade
Verificação de possibilidade de otimização	Algoritmo de cálculo de caminho mais Curto
Gerenciamento de informações sobre recursos e aplicações	Topologia de recursos disponíveis
Execução de ações de elasticidade (manipulação de máquinas virtuais)	Criação de máquina virtual,configuração de conectividade e da aplicação

Tabela 4: Funcionalidades do serviço de Posicionamento Dinâmico de Servidores de Cache de acordo com o modelo

Outro exemplo relacionado aos serviços de elasticidade, desta vez mais próximo dos modelos tradicionais de monitoramento de métricas em infraestrutura, é o armazenamento elástico. Definindo a porcentagem de utilização do espaço de armazenamento como condição de gatilho da regra, e tendo o cálculo do local mais próximo com recursos disponíveis como algoritmo de verificação, seria executada a ação de criação de uma nova instância desse serviço no host mais próximo. Na **Tabela 5** são apresentadas as funcionalidades do serviço de Armazenamento Elástico a partir do modelo.

Modelo de Desenvolvimento de Serviços Elásticos	Serviço de Armazenamento Elástico
Definição e monitoramento de métricas e condições das regras	Monitor de Espaço em Disco
Verificação de possibilidade de otimização	Algoritmo de cálculo de caminho mais Curto
Gerenciamento de informações sobre recursos e aplicações	Topologia de recursos disponíveis
Execução de ações de elasticidade (manipulação de máquinas virtuais)	Criação de máquina virtual,configuração de conectividade e da aplicação

Tabela 5: Funcionalidades da aplicação de Posicionamento Dinâmico de Servidores de Cache de acordo com o modelo

Como pode ser observado na **Tabela 3**, na **Tabela 4** e na **Tabela 5**, as funcionalidades de gerenciamento de informações sobre os recursos e aplicações, assim como a manipulação de máquinas virtuais, são funcionalidades úteis aos três serviços descritos como exemplo de utilização do modelo de desenvolvimento de serviços elásticos. A partir dessas primeiras funcionalidades identificadas, propõe-se o desenvolvimento de um *middleware* que as implemente e facilite a tarefa de programação de novos serviços de elasticidade. A seguir, na **seção 4.2**, são detalhadas as duas funcionalidades propostas e o *middleware* propriamente dito.

4.2 *Middleware* e Funcionalidades

Identificadas as funcionalidades comuns a aplicações elásticas, é necessário um detalhamento de suas características para posterior implementação. Nessa seção são descritas as funcionalidades desejáveis a um *middleware* que auxilie no desenvolvimento e execução de aplicações e serviços escaláveis. Dentre as primeiras funcionalidades descritas estão a manipulação de máquinas virtuais e o gerenciamento de informações.

4.2.1 Funcionalidade 1: Manipulação de Máquinas Virtuais

A funcionalidade de manipulação de máquinas virtuais envolve a criação e destruição de máquinas virtuais e suas configurações de conectividade, a atualização dos módulos do *middleware* e das aplicações, assim como suas inicializações e configurações iniciais. A criação de máquinas virtuais utiliza a clonagem (replicação) de imagens pré-configuradas. As configurações de conectividade são configuradas automaticamente a partir da distribuição de endereços de rede pelo *Dynamic Host Configuration Protocol* (DHCP). O *middleware* é iniciado automaticamente assim que o sistema operacional da máquina virtual é inicializado e se comunica com os outros nós do *middleware* distribuído, trocando informações de endereçamento (para registro no sistema de gerenciamento de informações) e recebendo as configurações da aplicação a ser executada. O *middleware* local também é responsável por configurar e inicializar os módulos da aplicação.

4.2.2 Funcionalidade 2: Gerenciamento de Informações

A funcionalidade de gerenciamento de informações está diretamente relacionada ao gerenciamento dos recursos disponíveis, das máquinas virtuais instanciadas, das instâncias de aplicação, das configurações de um nó e dos usuários. Esse gerenciamento é feito de forma distribuída, ou seja, as informações podem ser acessadas por qualquer nó do *middleware* por meio de uma aplicação de banco de dados distribuída[18]. Isto difere da solução Wind, que gerencia informações sobre as instâncias ativas de forma centralizada (Statiscs Server). Além do gerenciamento distribuído, foram inseridas no Trade Wind informações sobre os usuários e sobre a aplicação, permitindo a criação de fluxos de fornecimento de vídeo simultâneos. Já na solução Wind, a única informação gerenciada são as instâncias de máquina virtual ativas; logo, não é possível identificar as instâncias de aplicação para dois fluxos de vídeo simultâneos.

Na **Figura 11**, é apresentado o modelo entidade-relacionamento proposto para a modelagem do banco de dados. Na sequência, são descritas as tabelas e suas colunas, apresentando exemplos práticos que justificam a escolha da organização apresentada, bem como sua aplicação no caso de uma funcionalidade que verifica a possibilidade de otimização.

[18] A garantia de integridade e consistência entre as informações dos diferentes nós não é o foco deste trabalho, sendo transferida a responsabilidade para a aplicação de banco de dados.

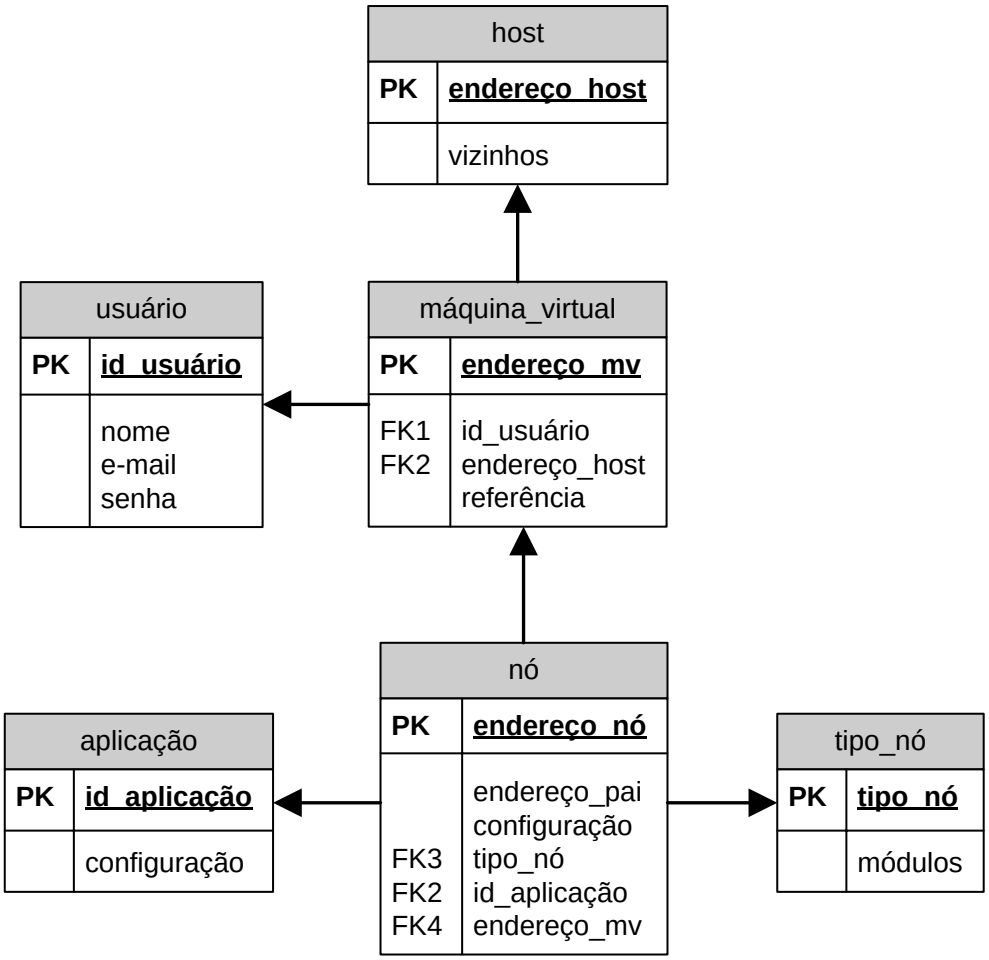

Figura 11: Modelo entidade-relacionamento das informações

- **Tabela *Host*:** mantém informações sobre a máquina física que irá hospedar as máquinas virtuais dos usuários da nuvem. Dentre as informações fornecidas estão o nome do *host* e sua relação de vizinhança com outros nós. Em um grafo, os nós são a representação dos hosts e as arestas a relação de vizinhança entre eles, ou seja, dois *hosts* que possuem um enlace de rede conectando-os diretamente.

endereço_host	vizinhos
‘core@host1’	[‘core@host2’]
‘core@host2’	[‘core@host1’, ‘core@host3’, ‘core@host4’]
‘core@host3’	[‘core@host2’]
‘core@host4’	[‘core@host2’]

Tabela 6: Tabela com informações de *host* e sua relação de vizinhança

A **Figura 12** ilustra a topologia representada na **Tabela 6**, na qual a relação de vizinhança indica um enlace bidirecional entre as máquinas vizinhas (i.e., há a possibilidade de *download* e *upload* de dados em cada enlace).

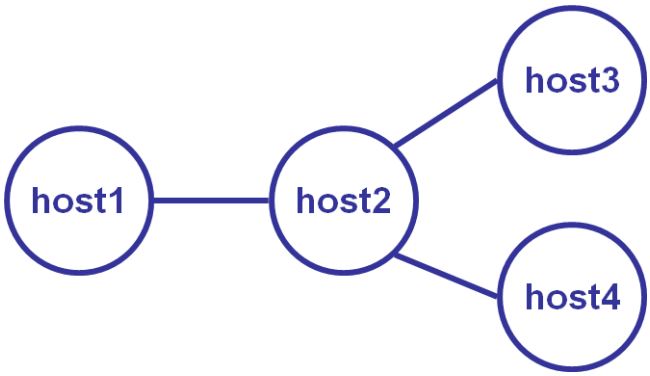

Figura 12: Grafo representando as relações de vizinhança a Tabela 6

- **Tabela Usuário:** armazena informações sobre o usuário da nuvem, permitindo que sejam associadas máquinas virtuais e aplicações. Na **Tabela 7** são apresentadas as informações de usuário, bem como um exemplo de valores possíveis no seu preenchimento.

id_usuário	nome	e-mail	senha
1	‘administrador_nuvem’	‘adm@nuvem.com.br’	‘admin’
2	‘abacate’	‘usuario@abacate.net’	‘1234’
3	‘banana’	‘usuario@banana.net’	‘5678’
4	‘caju’	‘usuario@caju.net’	‘90ab’

Tabela 7: Tabela usuário com valores fictícios

- **Tabela Máquina Virtual:** armazena informações sobre as instâncias de máquina virtual alocadas. Esta tabela possui indicação de usuário a qual pertence e em qual *host* está hospedada. O código de referência é utilizado pelo *hypervisor* dos *hosts*. A **Tabela 8** traz os campos desta tabela e valores fictícios de exemplo.

endereço_mv	id_usuário	endereço_host	Referência
‘m1.host1’	2	‘host1’	‘123456_1587968136’
‘m1.host3’	2	‘host3’	‘123456_1587968137’
‘m1.host4’	3	‘host4’	‘123456_1587968138’
‘m2.host1’	3	‘host1’	‘123456_1587968139’

Tabela 8: Tabela máquina virtual com seus campos e valores fictícios

A **Figura 13** ilustra as instâncias de máquina virtual descritas na **Tabela 8**. Os círculos em preto indicam as instâncias sendo executadas no *host* indicado em seus endereços.

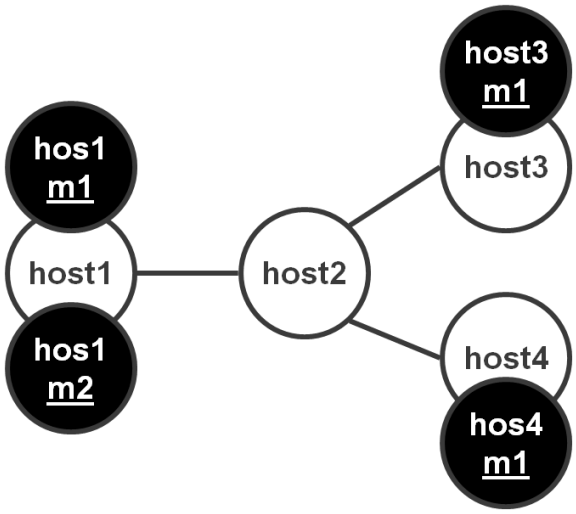

Figura 13: Grafo representando as máquinas virtuais

- **Tabela Aplicação:** armazena informações sobre a identificação de uma determinada instância de aplicação. No caso de uma aplicação de vídeo, pode armazenar o nome do vídeo distribuído; já no caso de uma aplicação de jogos online, é possível armazenar o nome do servidor, por exemplo. Na **Tabela 9** são representados os campos desta tabela, aplicação, preenchida de valores fictícios.

id_aplicação	configuração
1	[{canal, “futebol.ts”}]
2	[{canal, “desenho_animado.ts”}]
3	[{servidor, “assault_cube: server1”}]

Tabela 9: Tabela aplicação e seus valores fictícios

- **Tabela Tipo de Nó:** os tipos de nó simplificam a inicialização de módulos, pois um tipo de nó geralmente necessita de uma combinação de módulos trabalhando em conjunto para fornecer uma determinada funcionalidade. A **Tabela 10** traz exemplos de configurações dos tipos de nó, com os módulos a serem iniciados.

tipo	Módulos
ingestion	[ingestion, chunk_relay, lcp]
distribution	[distribution]
chunk_relay	[chunk_relay, lcp]
g_ingestion	[g_ingestion]
g_distribution	[g_distribution]

Tabela 10: Tabela de tipos de nó e a lista de nós a serem iniciados

- **Tabela Nó:** mantém informações sobre o relacionamento com outras instâncias de aplicação (endereço de nó pai), sobre as instâncias de aplicação, suas configurações, o tipo de nó, e o endereço da máquina virtual em que o nó está hospedado Na **Tabela 11** são apresentados os campos desta tabela e dados fictícios.

endereço_nó	endereço_pai
‘ingestion_1@m1.host1’	‘’
‘distribution_1@m1.host3’	‘ingestion_1@m1.host1’
‘ingestion_2@m1.host4	‘’
‘distribution_2@m2.host1’	‘ingestion_2@m1.host4’

configuração	tipo_nó	id_aplicação	endereço_mv
[...]	ingestion	1	‘m1.host1’
[...]	distribution	1	‘m1.host3’
[...]	ingestion	2	‘m1.host4’
[...]	distribution	2	‘m2.host1’

Tabela 11: Tabela nó e seus valores fictícios

Nas **Figura 14** são representadas as máquinas virtuais instanciadas nos *hosts* 1, 3 e 4, pertencentes a diferentes usuários. Através das informações da **Tabela 11**, é possível identificar: os usuários aos quais os nós pertencem (a partir da referência da **Tabela 7**) e a relação de parentesco entre máquinas, sendo o nó “M1.host1” pai do nó “M1.host3”, na **Figura 14(a)**, e o nó “M1.host” pai do nó “M2.host1”, na **Figura 14(b).**

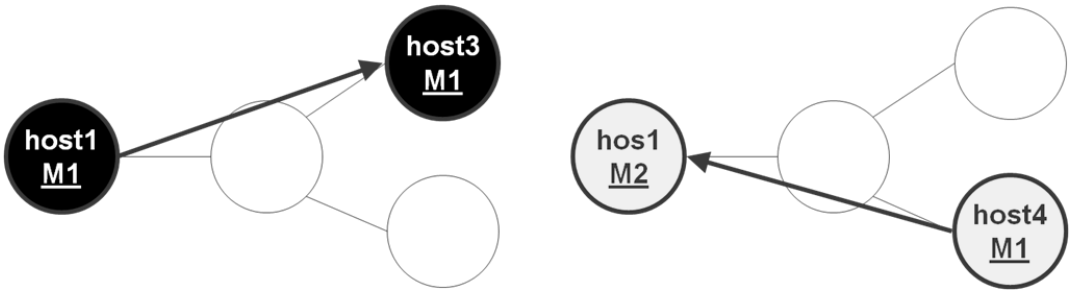

Figura 14: (a) Máquinas virtuais instanciadas pelo usuário 'abacate' (preto)
(b) Máquinas virtuais instanciadas pelo usuário 'banana' (cinza)

4.3 Modelo Aplicação, Serviço, Funcionalidade

Para facilitar a utilização dos serviços da plataforma de elasticidade e organizar o desenvolvimento de novas funcionalidades é proposto um modelo de desenvolvimento de aplicações baseado em composição por blocos de serviços. Nesse modelo, as **funcionalidades** são implementadas por meio da programação de módulos funcionais. Essas funcionalidades podem ser selecionadas para composição de **serviços** e os serviços combinados em forma de **aplicações**.

A **Tabela 12** apresenta como seria a composição de aplicações a partir dos blocos de serviços e funcionalidades. Observe que serviços tradicionais de monitoramento de métricas de infraestrutura também podem ser utilizados em aplicações de armazenamento de dados, por exemplo. Em cinza estão destacadas as funcionalidades do *middleware*, comuns às aplicações de exemplo.

Aplicação	Serviço	Funcionalidade
Fornecimento de Fluxos de Vídeo	Redução de Fluxos Redundantes	Manipulação de MV
		Gerenciamento de Informações
		Monitoramento de Requisições Redundantes
		Otimização de Caminho Comum Mais Longo
Armazenamento de Dados	Posicionamento Dinâmico de Servidores de *Cache*	Manipulação de MV
		Gerenciamento de Informações
		Monitoramento de Quantidade de Requisições por Localidade
		Otimização de Caminho Mais Curto
	Armazenamento Elástico	Manipulação de MV
		Gerenciamento de Informações
		Monitoramento de Espaço em Disco
		Otimização de Caminho Mais Curto

Tabela 12: Modelo de Desenvolvimento de Aplicações a partir da composição de Serviços e Funcionalidades

A organização do desenvolvimento de novas aplicações por meio da estrutura proposta facilita a programação e incentiva a reutilização de código, pois permite a composição de serviços e aplicações de forma simples e transparente. Ao manter o baixo acoplamento dos módulos funcionais, tal organização permite expandir a quantidade de funcionalidades disponíveis no *middleware*.

4.4 Arquitetura

O **Trade Wind** é uma solução para o desenvolvimento de aplicações elásticas e distribuídas para nuvens de computação. A solução é composta de uma API e um *middleware* distribuído que disponibiliza funcionalidades de gerenciamento de informações sobre a nuvem e de manipulação de máquinas virtuais e aplicações.

Esse *middleware* pode ser estendido através da programação de novas funcionalidades e da composição de novos serviços e aplicações de elasticidade com base no modelo de desenvolvimento de aplicações elásticas (regras de automatização de escalabilidade). Para detalhar esses conceitos, esta seção é dividia em três subseções que descrevem, respectivamente: a organização da infraestrutura de computação, os módulos essenciais do *middleware* da solução Trade Wind e os módulos da aplicação de fornecimento de fluxos de vídeo.

4.4.1 Organização da Infraestrutura

Para a disponibilização da plataforma e das aplicações de elasticidade é necessária a organização de sua estrutura e a definição das entidades que a compõem. Também é importante o suporte de uma infraestrutura por meio de seus recursos virtuais e físicos. A seguir são descritos os principais elementos da infraestrutura, visando facilitar a contextualização e compreensão de alguns aspectos específicos da arquitetura da solução.

- ***Host***: é um servidor físico no qual são alocados os recursos virtuais de acordo com as necessidades dos usuários. O host executa um software do tipo *hypervisor* que oferece uma plataforma virtual para os sistemas operacionais das máquinas virtuais, permitindo que eles compartilhem os mesmos recursos físicos de forma transparente. Em um *host* podem ser instanciadas diversas máquinas virtuais.

- **Máquina Virtual (Nó):** recurso virtual que pode ser alocado e liberado de acordo com a demanda. Para garantir a privacidade dos dados e da aplicação, não podem ser executadas aplicações de diferentes usuários em uma mesma máquina virtual. Assim, em cada host pertencente à topologia do Trade Wind, existe uma e apenas uma máquina virtual executando o núcleo *middleware* de gerenciamento de elasticidade da nuvem (funcionalidades de gerenciamento de informações e manipulação de máquinas virtuais). Podem ser executadas diversas aplicações em uma máquina virtual.

- **Módulo**: é uma unidade funcional programada para atuar dentro da arquitetura da solução. Os módulos são programados e, em conjunto, formam uma aplicação. Alguns módulos fazem parte do *middleware* e da API, sendo pouco modificados. Outros são partes componentes de uma aplicação e proveem serviços aos clientes finais, sendo modificados de acordo com as necessidades específicas de elasticidade.

- **Aplicação:** é composta de um conjunto de módulos que atuam de forma cooperativa, com o objetivo de prover serviços específicos de elasticidade.

A **Figura 15** ilustra a organização dos elementos da infraestrutura da solução Trade Wind. De baixo para cima são representados os elementos de acordo com a ordem de encapsulamento. Um módulo pertence a uma aplicação, que é executada em máquina virtual, que, por fim, é virtualizada em um *host*.

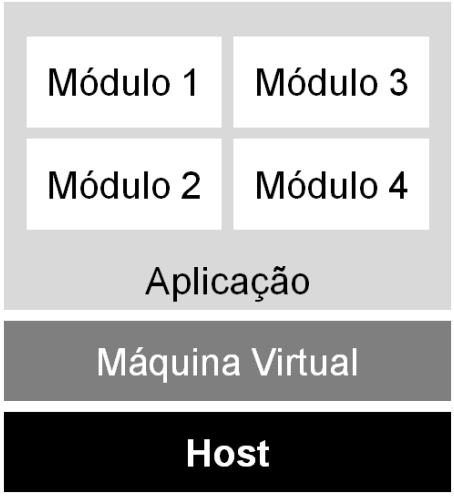

Figura 15: Organização dos elementos da infraestrutura do Trade Wind

4.4.2 Módulos Essenciais do Trade Wind

Na **Seção 4.2** foram definidas e descritas as funcionalidades desejadas em um *middleware* para o controle de soluções elásticas a partir do modelo de desenvolvimento de aplicações elásticas proposta na **Subseção 4.1.2**. A implementação desse *middleware* foi dividida em dois módulos principais: o módulo Core e o módulo Core Access. Nesta seção são descritos os dois módulos, detalhando-se sua organização e implementação.

- **Módulo Core (Middleware):** o módulo Core é o *middleware* que implementa as funcionalidades de manipulação de máquinas virtuais e gerenciamento das informações sobre a infraestrutura (máquinas físicas e virtuais), usuários da nuvem, aplicações em execução e suas configurações. Ele é organizado de forma distribuída, sendo executado em cada um dos *hosts* da nuvem de computação, gerenciando informações através de um banco de dados, também distribuído. Em cada *host* existe apenas um módulo Core executado em uma máquina virtual dedicada. Assim, para que um *host* faça parte do sistema Trade Wind, ele deve executar o *middleware*, mesmo que nenhum outro recurso virtual tenha sido alocado. Dentre as **responsabilidades do módulo Core** estão: o gerenciamento/disponibilização de informações sobre recursos físicos, recursos virtuais alocados, usuários, aplicações instanciadas: a manipulação (criação/destruição) de máquinas virtuais em seu próprio *host;* e a intermediação nos pedidos de criação de novas máquinas virtuais remotas (em outro *host*), requisitados por um módulo Core Access local (mesmo *host*). A **Figura 16** ilustra dois diferentes *hosts*, cada um deles executando seu módulo Core em uma máquina virtual dedicada. Repare nas setas que indicam a comunicação entre os módulos Core. Nesta figura também é possível observar comunicação entre o *middleware* e o banco de dados distribuído.

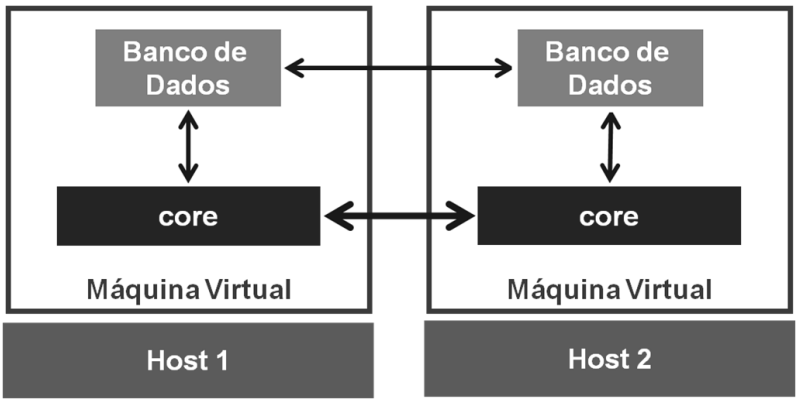

Figura 16: Interfaces de comunicação do módulo Core. Adaptador de: (MIERS *et al*, 2011)

- **Módulo Core Access (API):** o módulo Core Access atua como intermediário na comunicação entre uma aplicação e o módulo Core local (no mesmo *host*).

Qualquer máquina virtual instanciada deve inicializar o módulo, e apenas um. Em um mesmo *host*, podem existir diversas máquinas virtuais, cada uma executando um módulo Core Access dedicado. As funcionalidades de verificação da possibilidade de otimização são implementada nele, sendo necessária a comunicação com o *middleware* para a obtenção de informações sobre os recursos disponíveis e sobre os recursos instanciados pela aplicação. Outra função do Core Access é requisitar a alocação de recursos remotos (em outro *host*). O módulo ainda realiza a função de API para programação de aplicações de elasticidade, permitindo que os desenvolvedores criem novas funcionalidades de elasticidade de acordo com suas necessidades. Dentre as **responsabilidades do módulo Core Access** estão: o registro de informações sobre a máquina virtual junto ao módulo Core; a inicialização e configuração de módulos de aplicação; a requisição e repasse de informações para módulos que implementam funcionalidades de verificação da possibilidade de otimização; e a requisição de criação de instâncias locais ou remotas em de novas máquinas virtuais. Na **Figura 17**, é apresentada uma visão geral da solução, indicando as comunicações entre o módulo Core e o módulo Core Access, assim como do módulo Core Access e as aplicações de elasticidade.

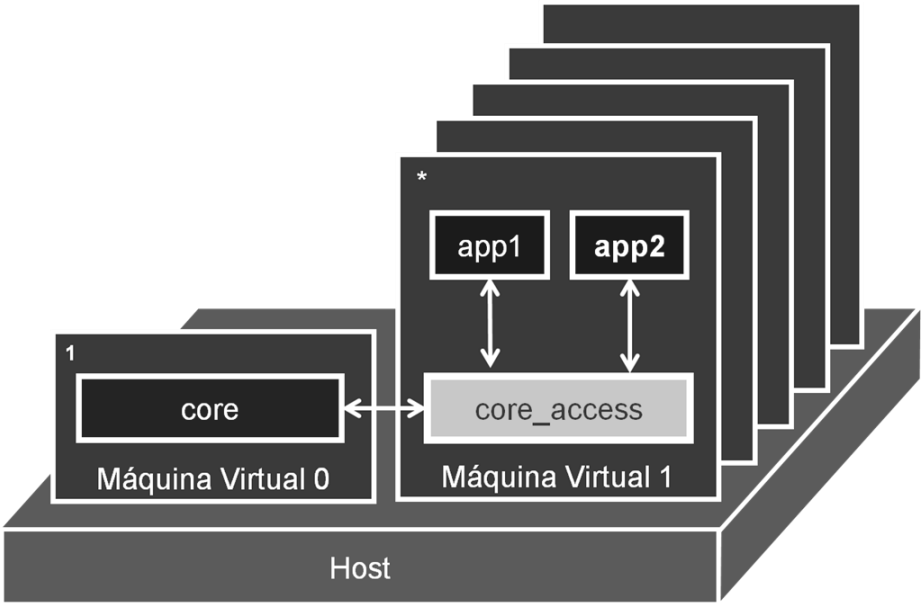

Figura 17: Interfaces de comunicação do módulo Core Access. Adaptado de: (MIERS *et al*, 2011)

4.5 Aplicação – Fornecimento de Fluxos de Vídeo

A aplicação escolhida como prova de conceito para o Trade Wind foi a mesma descrita no Wind original, ou seja, a aplicação de fornecimento de fluxos de vídeo com redução de fluxos redundantes, descrita na **Seção 3.5**. Para que pudesse interagir com o Trade Wind, a aplicação passou por adaptações. A principal adaptação realizada foi a separação entre as funcionalidades de fornecimento dos fluxos redundantes e as funcionalidades de elasticidade. Nessa seção são detalhados os módulos desses dois serviços. Na próxima seção são descritas em detalhes as ações relacionadas às otimizações omitidas na descrição dos módulos de elasticidade

4.5.1 Módulos de Distribuição de Fluxo

A aplicação de distribuição de vídeo utilizada pelo Wind original é composta de cinco módulos nos quais as funcionalidades de fornecimento de fluxos de vídeo e as funções de elasticidade encontram-se fortemente acopladas. Para permitir a implementação da aplicação junto à solução Trade Wind, foram adaptadas as funções dos módulos Tail Node, Intermediary Node e Head Node. Dentre as funcionalidades mantidas estão a transformação do fluxo de vídeo em blocos, o repasse dos blocos e a transformação dos blocos no fluxo original. Os eventos de recebimento de requisições por um conteúdo são encaminhados aos módulos responsáveis pela elasticidade da solução, que são descritos na **Seção 4.5.2**. A seguir são definidos os módulos Ingestion, Relay e Distribution, responsáveis pelo serviço de fornecimento de fluxos de vídeo.

- **Ingestion:** é o módulo requisita um fluxo de vídeo do provedor de conteúdo e, ao recebê-lo, realiza a transformação do fluxo de vídeo em blocos numerados. Esses blocos são encaminhados para o módulo Relay local (mesma máquina virtual);

- **Relay**: é o módulo responsável pelo repasse de requisições de conteúdo e dos blocos de conteúdo. Pode repassar requisições e blocos para os módulos: Ingestion, Relay e Distribution. O módulo Relay também é

responsável pelo encaminhamento de eventos de requisição de conteúdo para os módulos de controle de elasticidade;

- **Distribution**: é módulo responsável por receber as requisições de fluxo de video dos clientes, encaminhando-as para um módulo Relay local. Quando recebe os blocos de conteúdo, converte para o formato de fluxo de vídeo e encaminha para os clientes requisitantes.

A **Figura 18** ilustra o funcionamento de toda a aplicação de fornecimento de fluxos de vídeo e a interação entre os módulos de distribuição de fluxos de vídeo. As setas contínuas indicam a direção do fluxo de vídeo, enquanto as setas tracejadas indicam a direção dos blocos de conteúdo gerados a partir do fluxo de vídeo original.

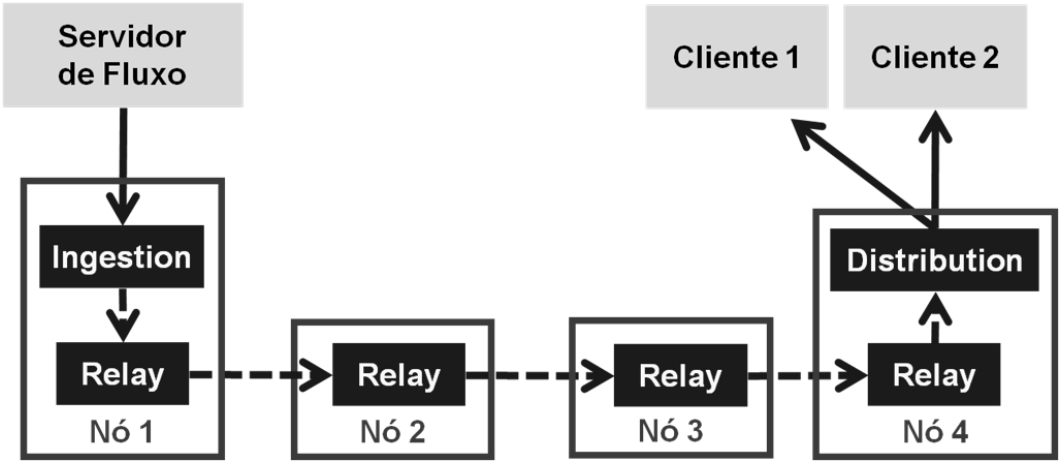

Figura 18: Visão geral da aplicação de fornecimento de fluxos de vídeo

4.5.2 Módulos do Modelo de Elasticidade

A aplicação descrita na S**ubseção 4.5.1** explica o funcionamento do serviço de distribuição de vídeo. Para que seja possível reduzir os fluxos redundantes de forma automática é necessário implementar os módulos responsáveis pela elasticidade do serviço de vídeo. De acordo com o modelo proposto na **Subseção 4.1.2** e sua descrição específica na **Tabela 2**, as funcionalidades de monitoramento de requisições e o algoritmo de cálculo do caminho comum mais longo devem ser implementadas. Como descrito na **Subseção 4.5.1**, quando um módulo Relay recebe requisições, ele encaminha os eventos para os módulos do modelo de elasticidade.

Nesta seção são descritos os módulos de monitoramento da quantidade de requisições e de cálculo do caminho mais longo comum, nomeados **Redundant Request Listener** e **Longest Common Path (LCP)**, respectivamente. A **Figura 19** apresenta uma visão geral da aplicação de distribuição de vídeo com o módulo de detecção de fluxos redundantes (**RR Listener**) e sua relação com o **módulo Relay**.

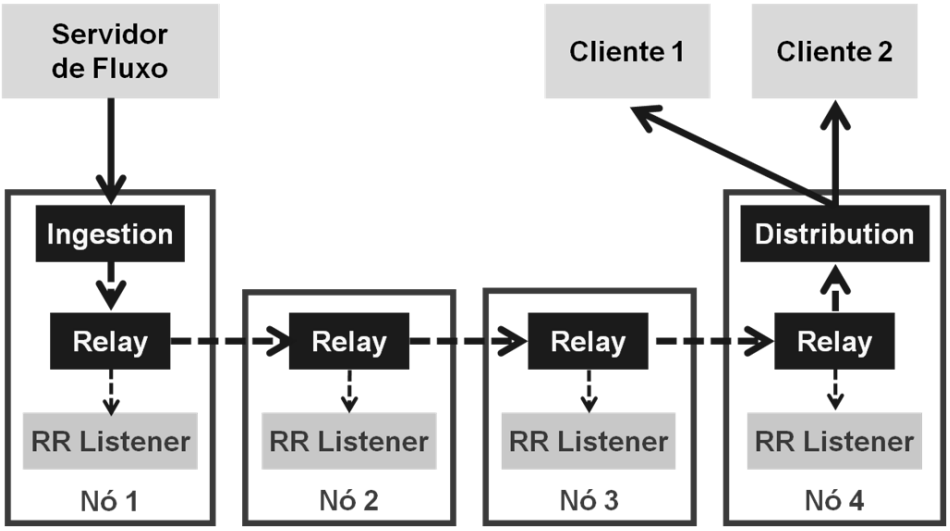

Figura 19: Aplicação de fornecimento de fluxos e sua comunicação com o módulo de monitoramento

- **Módulo Redundant Request Listener (RR Listener)**: o módulo RR Listener recebe um fluxo de mensagens enviadas pelo módulo Relay indicando eventos de requisição de blocos de conteúdo recebidos. Os eventos são então monitorados e, caso seja detectado um fluxo redundante (requisições para um mesmo conteúdo enviadas por diferentes nós Relay), o módulo Core Access é acionado para que verifique a possibilidade de otimização. A **Figura 20** ilustra a interação entre os módulos RR Listener e Core Access, em uma mesma máquina virtual.

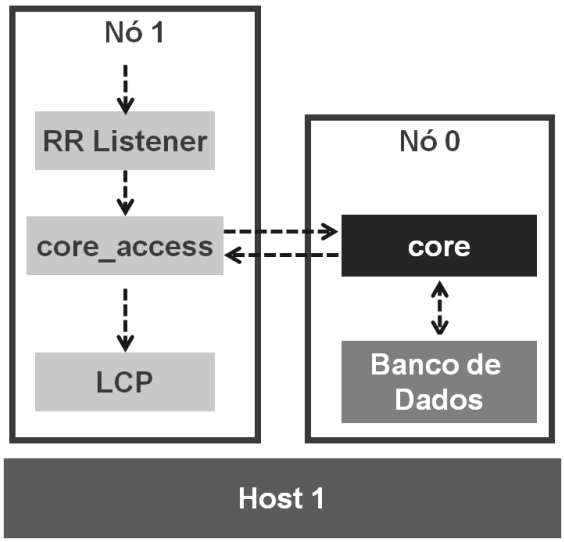

Figura 20: Interação entre os módulos do mecanismo de monitoramento e verificação

- **Módulo Longest Common Path (LCP):** o módulo LCP é responsável por calcular o caminho comum mais longo, através do algoritmo descrito na **subseção 3.5.4**. Como indicado na **figura 12**, ao receber as informações da topologia de *hosts* disponíveis e a topologia das instâncias da aplicação de fornecimento de fluxos redundantes (obtidas pelo módulo Core Access através da interação com o módulo Core), o módulo LCP indica o lugar da topologia de *hosts* onde deve ser instanciada uma nova máquina virtual com a aplicação Relay. O LCP também é responsável por solicitar que o Core Access crie uma máquina virtual no *host* indicado por este algoritmo.

4.6 Elasticidade – Rotina de Ajuste de Demanda

Na operação de uma rede de computadores, duas situações são desejadas pelos administradores: a economia de recursos e a melhoria de qualidade considerando métricas, como a diminuição no tempo de entrega de pacotes, ou a diminuição da taxa de perda de pacotes. Isto é interessante porque estas são situações que influenciam diretamente na qualidade de experiência do usuário em serviços com diferentes requisitos funcionais. No caso da aplicação de distribuição de fluxos de vídeo, a economia na utilização da banda disponível é bem clara: caso seja possível reduzir a quantidade de fluxos redundantes de N para apenas 1, economiza-se o recurso da largura de banda em N vezes.

Antes de iniciar a descrição da rotina de ajuste de demanda, é importante definir um conjunto de ações baseadas no modelo de desenvolvimento de serviços elásticos. Isto é feito na **Tabela 13**, que mostra como as funcionalidades são traduzidas em ações de ajuste de demanda. Note que o gerenciamento de informações encontra-se na categoria de "ação de alteração". juntamente com a execução das ações de elasticidade, pois, após as alterações na topologia das instâncias de aplicação, essas informações também devem ser atualizadas.

Modelo de Desenvolvimento de Serviços Elásticos	Ações de Ajuste de Demanda
Definição e monitoramento de métricas e condições das regras	Detecção
Verificação de possibilidade de otimização	Verificação
Gerenciamento de informações sobre recursos e aplicações	Alteração
Execução de ações de elasticidade (manipulação de máquinas virtuais)	Alteração

Tabela 13: Relação entre as ações de ajuste de demanda e o modelo de desenvolvimento de serviços elásticos

4.6.1 Rotina – Redução de Fluxos Redundantes

Seguindo a nomenclatura de ações de ajuste de demanda proposta na **Tabela 13**, existe uma sequência de três ações que devem ser executadas para que o ajuste de demanda se concretize. A seguir são descritas em detalhes cada uma das ações de ajuste de demanda.

1) Detecção: no caso da aplicação de redução de fluxos redundantes, o módulo RR Listener detecta eventos de requisição duplicados para um mesmo bloco de conteúdo (mesma numeração de identificação), partindo de diferentes módulos Relay requisitantes. Como ilustrado na **Figura 21**, o nó **M1.host1** detecta as requisições redundantes enviadas para os nós **M1.host3** e **M1.host4**. A **Figura 19**

ilustra a iteração entre os nós Relay e RR Listener para a detecção do fluxo redundante.

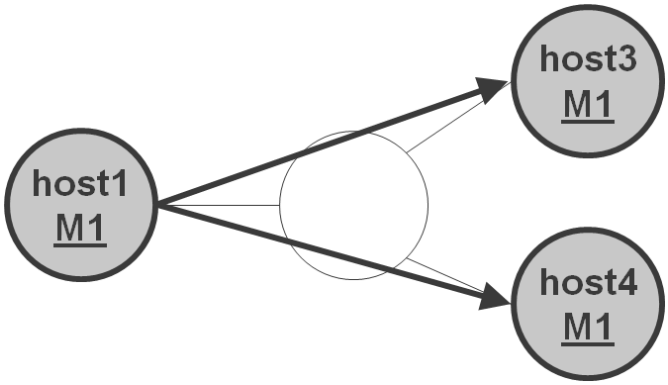

Figura 21: Nó 'M1.host1' fornecendo fluxos redundantes para os nós 'M1.host3'e 'M1.host4'

2) Verificação: após a detecção do fluxo redundante pelo RR Listener, o módulo Core Access é acionado para obter informações sobre a topologia de hosts disponíveis e sobre a topologia de instâncias da aplicação de fornecimento de fluxos de vídeo, do módulo Core local. Obtidas as informações, essas são repassadas ao módulo LCP, que por sua vez calcula o ponto da topologia de hosts onde deve ser instanciado um novo módulo Relay e verifica se há a possibilidade dessa ação. A **Figura 22** indica a situação na qual o *host2* é indicado como o host onde o nó deve ser instanciado. A iteração entre os nós RR Listener, Core Access, Core e LCP é mostrada na **Figura 20**.

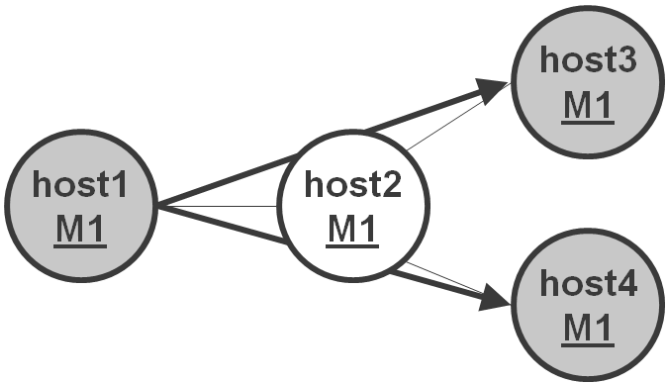

Figura 22: '*Host2*' identificado como ponto de criação de um novo nó *Relay*

3) Alteração: por fim, o módulo LCP solicita ao módulo Core Access que crie uma nova instância de nó Relay no *host* alvo (*host* identificado no passo anterior). O módulo Core Access repassa a solicitação ao módulo Core local, que repassa novamente a solicitação ao módulo Core Remoto no *host* alvo juntamente com as

informações de configuração do módulo Relay. A **Erro! Fonte de referência não encontrada.** ilustra a comunicação entre os nós LCP, Core Access e Core, até a criação de uma nova instância de máquina virtual e a configuração de um novo nó de aplicação. Na **Figura 24**, é possível observar a topologia da aplicação após a realização do ajuste de demanda.

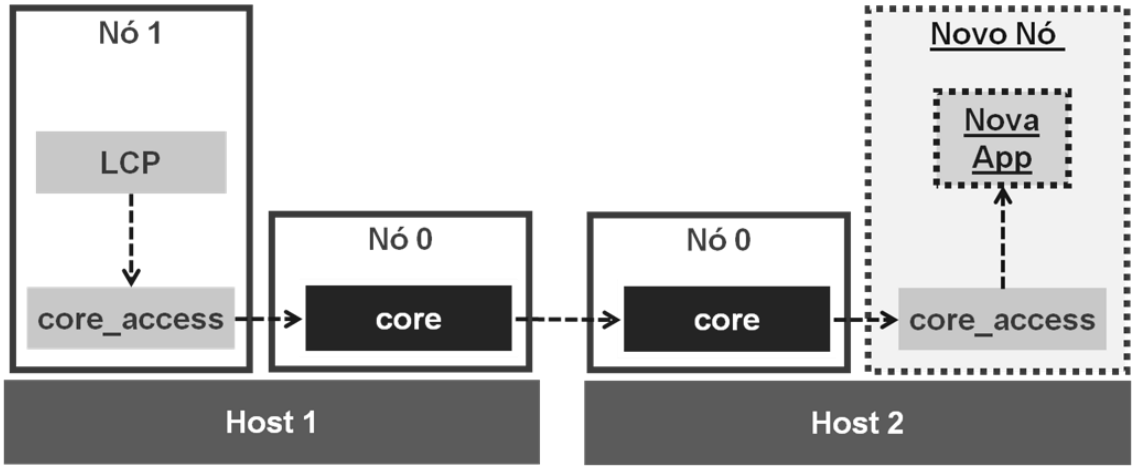

Figura 23: Criação de um novo nó de repasse de fluxos de vídeo

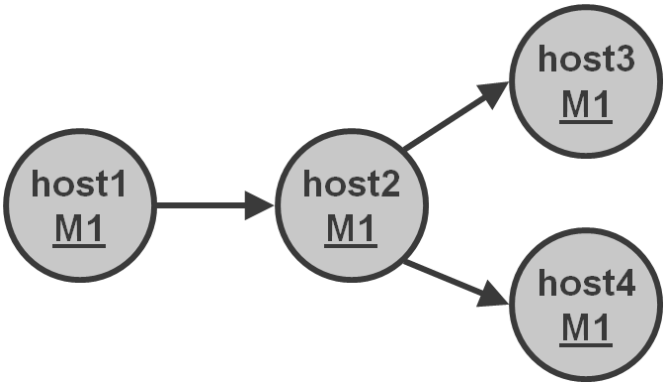

Figura 24: Fluxo reduzido após criação de instância 'M1.host2'

5 ANÁLISE EXPERIMENTAL

Após o estudo realizado nos **Capítulos 2** e **3**, que serviram como base para o entendimento das características de soluções elásticas para nuvens de computação, no **Capítulo 4** foi proposta e descrita em detalhes a solução Trade Wind. Com o intuito de avaliar o cumprimento dos objetivos estabelecidos na **Seção 1.2**, o presente capítulo apresenta o ambiente de testes, descrevendo o método de coleta de dados e analisando os resultados obtidos com a execução de experimentos de validação. Além disso, são apresentados a topologia escolhida e seus equipamentos e *softwares* auxiliares, os cenários de teste e o conteúdo de vídeo utilizado nos testes. Ao final do capítulo são discutidas as implicações dos resultados obtidos.

5.1 Métricas do Experimento

Parte da proposta da solução Trade Wind implicou no estudo da aplicação Wind, analisando suas características e decisões de projeto. A partir dessas observações, foram especificadas as funcionalidades genéricas que fariam parte de um *middleware* e poderiam ser acessadas por parte das aplicações. Dentre as funcionalidades genéricas escolhidas para o *middleware* estão a manipulação/configuração de recursos computacionais e o gerenciamento de informações sobre: os usuários; os recursos físicos e virtuais; as aplicações e suas configurações específicas; e as topologias real e virtual (realizada por meio da comunicação entre as aplicações). Com o intuito de validar o correto funcionamento da aplicação distribuída de fornecimento de fluxos de vídeo em conjunto com a solução Trade Wind, foram escolhidas duas métricas que refletissem este comportamento: **o tempo de inicialização e configuração de um novo nó** da aplicação e a **quantidade de dados trafegados nos enlaces** do núcleo da rede da nuvem.

- A métrica que quantifica o **tempo de inicialização e configuração de um novo nó** de aplicação pode ser dividida em três submétricas: **o tempo para a clonagem** de uma nova máquina virtual; **o tempo de inicialização** completa

dessa máquina virtual; **o tempo de configuração** até que o novo nó solicite blocos de conteúdo;

- A partir da medição da **quantidade os dados trafegados nos enlaces do núcleo** da rede da nuvem, foi possível observar a redução do tráfego redundante e, em consequência, a redução da quantidade dos dados totais trafegados no enlace.

Com base na métrica de tempo de inicialização e configuração de um novo nó de aplicação, foi realizada uma análise de viabilidade da solução Trade Wind. Especialmente para aplicações de fornecimento de fluxos de vídeo, o tempo de adaptação da infraestrutura é um fator crítico, uma vez que os vídeos distribuídos possuem um tempo de duração que varia entre algumas unidades até dezenas minutos. Assim, é necessário que o tempo gasto paa realizar a daptação seja bastante inferior à duração do vídeo para que a solução pode ser considerada viável.

Outra componente da análise de viabilidade da solução é a quantidade de dados trafegados nos enlaces a qual mede a eficiência na utilização dos recursos. A partir de sua medição foi possível avaliar o correto funcionamento da aplicação de redução dos fluxos de vídeo redundantes e a economia real de banda obtida.

5.2 Cenários de Teste

Para a validação da solução proposta foram escolhidos três cenários de teste que permitissem a observação do comportamento da aplicação com a introdução de fluxos redundantes, com a utilização de diferentes aplicações para aquisição do vídeo e com a utilização de dois canais simultâneos de transmissão de fluxos. Nesta seção são descritas e justificadas as escolhas para os cenários de teste propostos.

5.2.1 Rotinas comuns aos cenários de teste

A fim de facilitar a descrição das rotinas dos 3 cenários de teste, foram separadas duas rotinas comuns: a aquisição de fluxo de vídeo e a redução de fluxos

redundantes. Nesta subseção é detalhada a rotina de aquisição de um fluxo de vídeo, uma vez que a rotina de redução de fluxos redundantes foi discutida na **Subseção 4.6.1**.

Aquisição de fluxo de vídeo

1. Cliente requisita o vídeo para o nó Distribution;
2. Nó Distribution requisita o vídeo para nó Ingestion;
3. Nó Ingestion requisita o vídeo para servidor de conteúdo do provedor;
4. Servidor de conteúdo inicia o fornecimento de fluxo de vídeo para o nó Ingestion;
5. Nó Ingestion quebra o fluxo de vídeo em blocos numerados;
6. Nó Ingestion envia blocos numerados ao nó Distribution;
7. Nó Distribution converte os blocos numerados em um fluxo de vídeo, novamente;
8. Nó Distribution inicia o fornecimento do fluxo de vídeo ao Cliente.

5.2.2 Cenário 1: Redução do Fluxo Redundante

O primeiro cenário de testes tem como objetivo a validação da aplicação de redução do fluxo de vídeo redundante, utilizando apenas um vídeo para o fornecimento do fluxo (canal único). Tal cenário permite a medição da quantidade de dados trafegados por um enlace situado no núcleo da topologia, isolado da interferência de outros sinais. Na **Figura 25** é apresentada a topologia de aplicação que representa a o cenário de testes 1. Esse cenário permite o fornecimento de fluxos redundantes entre os enlaces: nó Ingestion e roteador A; roteadores A e B. A seguir é definida a sequência de iterações do cenário de testes 1, com a indicação (entre parênteses) do tempo no qual as tarefas são iniciadas.

1. (0 s) Topologia mínima com apenas um canal de fornecimento de fluxos de vídeo, com um nó Ingestion e dois nós Distribution instanciados;
2. (0 s) Cliente 1 realiza rotina de aquisição de fluxo de vídeo a partir do nó Distribution 1;
3. (10 s) Cliente 2 realiza rotina de aquisição de fluxo de vídeo a partir do nó Distribution 2 (início do fluxo redundante);
4. Trade Wind realiza rotina de redução do fluxo redundante;
5. Topologia otimizada (sem fluxo redundante).

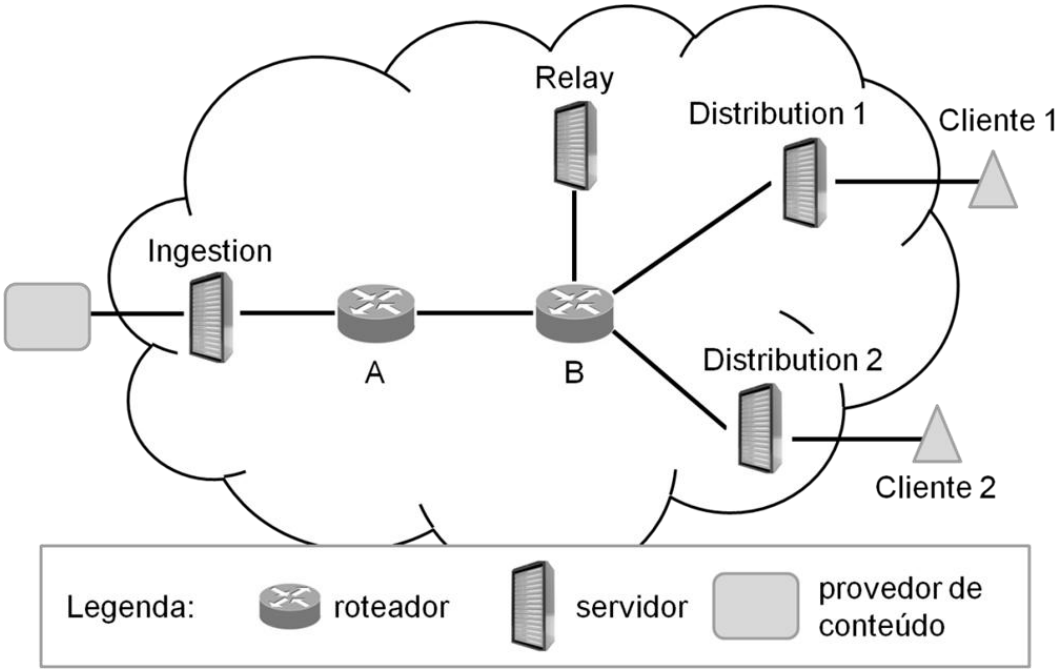

Figura 25: Topologia de aplicação do cenário de teste 1

5.2.3 Cenário 2: Alteração de Aplicação de *Download* do Cliente

O segundo cenário tem como objetivo verificar alguma possível diferença em relação à taxa de aquisição de blocos em cada uma das aplicações e suas implicações no funcionamento do mecanismo de redução de fluxos redundantes. Tal diferença pode ocorrer no caso de distribuição de fluxos de vídeo requisitados sob

demanda, ou seja, um fluxo obtido a partir de um arquivo de vídeo. O teste utiliza a topologia de aplicação ilustrada na **Figura 26**, com apenas um nó Ingestion e um nó Distribution instanciados. São realizadas duas execuções (não simultâneas) do mesmo teste, realizando uma aquisição de fluxo de vídeo completa, ou seja, o cliente adquire o conteúdo do início do fluxo de vídeo até o fim da transmissão de todo o conteúdo caracterizado na **Seção 5.4**. Na primeira execução é utilizada a aplicação *VLC Media Player*[19], a aplicação padrão para os clientes de outros cenários. Na segunda execução, o fluxo é solicitado por meio da aplicação *WGET*[20]. Foram medidos o tempo de download e o consumo de banda ao longo do período de aquisição do conteúdo de vídeo. A seguir é descrita a rotina de iterações para o cenário de testes 2, com a indicação (entre parênteses) do tempo no qual as tarefas são iniciadas.

1. (0 s) Topologia mínima com apenas um canal de fornecimento de fluxos de vídeo, com um nó Ingestion 1 e dois nós Distribution 1 instanciados;

2. (0 s) Cliente 1 realiza rotina de aquisição de fluxo de vídeo a partir do nó Distribution 1 (Execução 1: VLC *Media Player*; execução 2: *WGET*);

3. Fluxo de vídeo é enviado completamente.

[19] http://www.videolan.org/vlc/
[20] http://linux.about.com/od/commands/l/blcmdl1_wget.htm

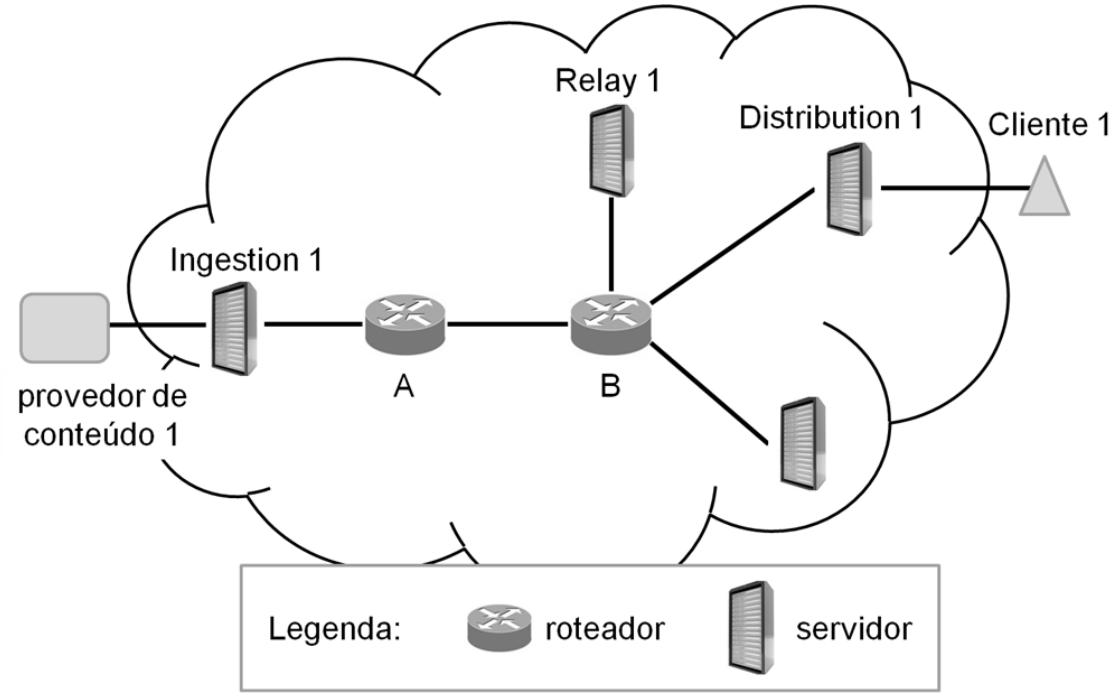

Figura 26: Topologia de aplicação dos cenários de teste 2 e 3 (fluxo de vídeo I)

5.2.4 Cenário 3 – Multicanais

Este cenário é semelhante ao cenário 2, mas tem como objetivo o teste da funcionalidade de múltiplos canais simultâneos. O teste consiste em transmitir, ao mesmo tempo, fluxos de vídeo por diferentes canais de distribuição utilizando a mesma topologia de *hosts*. O teste consiste na transmissão simultânea do **fluxo I**, que utiliza a topologia de aplicação ilustrada na **Figura 26**, e do **fluxo II,** que utiliza a topologia de aplicação ilustrada pela **Figura 27**. A seguir é descrita a rotina de iterações para o cenário de testes 3, com a indicação (entre parênteses) do tempo no qual as tarefas são iniciadas.

1. (0 s) Topologia mínima com dois canais de fornecimento de fluxos de vídeo, com um nó Ingestion e um nó Distribution instanciados para cada canal;

2. (0 s) Cliente 1 realiza rotina de aquisição de fluxo de vídeo I a partir do nó Distribution 1;

3. (20 s) Cliente 2 realiza rotina de aquisição de fluxo de vídeo II a partir do nó Distribution 2;

4. (60 s) Cliente 1 interrompe a aquisição do fluxo de vídeo I;

5. Fluxo de vídeo II enviado completamente.

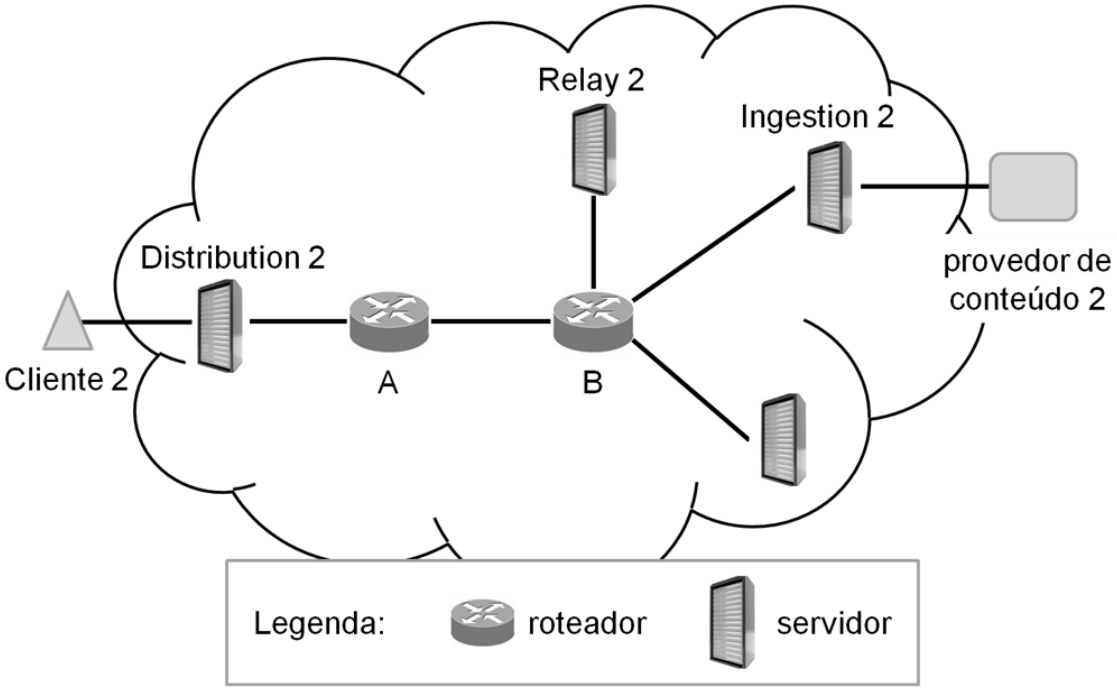

Figura 27: Topologia de aplicação do cenário de teste 3 para o fluxo de vídeo II

5.3 Topologia do Ambiente de Testes

Para efetuar a validação da solução foi configurado um ambiente de testes com uma topologia mínima que atendesse aos requisitos de todos os cenários de testes, na qual houvesse a possibilidade do fornecimento de fluxos redundantes a serem reduzidos. Outra adaptação realizada para a redução do número de equipamentos necessários para a execução dos testes foi a virtualização da topologia a partir dos equipamentos reais.

Com o intuito de esclarecer tais decisões do projeto do ambiente de testes e apresentar a topologia utilizada para a validação do protótipo são apresentadas nesta seção: a topologia mínima, a topologia lógica (a partir da virtualização de

servidores, roteadores e redes) e a topologia física (equipamentos reais) do ambiente de testes utilizado.

5.3.1 Topologia Mínima

Com base no cenário 1 (**Subseção 5.2.2**), para que exista uma situação onde existam fluxos de vídeo redundantes para um mesmo conteúdo sendo transmitidos pela infraestrutura de rede é necessário que estejam presentes, ao menos, um nó Ingestion e dois nós Distribution localizados em diferentes regiões e, portanto, em diferentes hosts. Além desses três hosts, a topologia mínima precisa de ao menos um host intermediário que permita a redução dos fluxos redundantes a partir da criação de um nó Relay.

Um detalhe que deve ser evidenciado na topologia mínima proposta é a necessidade de ao menos dois nós Distribution distintos, uma vez que a quantidade de clientes requisitantes não está necessariamente associada à existência de redundância na infraestrutura de rede (e.g., se 10 clientes requisitam um conteúdo de vídeo pelo mesmo nó Distribution, para o nó Ingestion, haverá apenas uma única requisição sendo feita pelo nó Distribution). Essa característica é intrínseca à arquitetura da aplicação.

Para realizar a operação inversa, de liberação de um recurso ocioso, a mesma topologia mínima pode ser reaproveitada. Nessa situação, após a otimização do fluxo redundante, estão instanciados o nó Ingestion, o nó Relay e dois nós Distribution (cada nó Distribution com um cliente requisitando o conteúdo). Caso um dos clientes interrompa a requisição do conteúdo, o nó Distribution associado também interrompe a requisição do conteúdo para o nó Relay. O nó Relay deixa de ter utilidade na rede, pois o nó Ingestion é capaz de distribuir o conteúdo para outro nó Distribution, sem que haja redundância de fluxos.

Para o cenário 3 (**Subseção 5.6.3**), no qual é necessária a transmissão de dois fluxos de vídeo diferentes e independentes, a mesma topologia mínima pode ser utilizada invertendo-se a direção dos fluxos. Por exemplo, o **fluxo** I instanciaria os nós Ingestion e Distribution como ilustrado na **Figura 26**; já o **fluxo II** instanciaria

os nós Ingestion 2 e Distribution 2 de forma invertida à apresentada na **Figura 27**, ou seja, o nó Ingestion 2 instanciado no lugar do nó Distribution 1, e o nó Distribution 2 sendo instanciado no lugar do nó Ingestion 1.

A **Figura 28** ilustra a topologia mínima, composta de 2 roteadores e 4 *hosts*. Os pontos de comunicação dessa topologia com usuários externos situam-se junto aos *hosts* 1, 3 e 4. Nestes pontos de aquisição podem ser conectados provedores de conteúdo ou clientes. Por exemplo, o provedor de conteúdo se conecta ao *host* 1, no qual o nó *Ingestion* é instanciado; os nós *Distribution* 1 e 2 são instanciados nos *hosts* 3 e 4, respectivamente; e o nó Relay é instanciado no *host* 2, próximo ao roteador B.

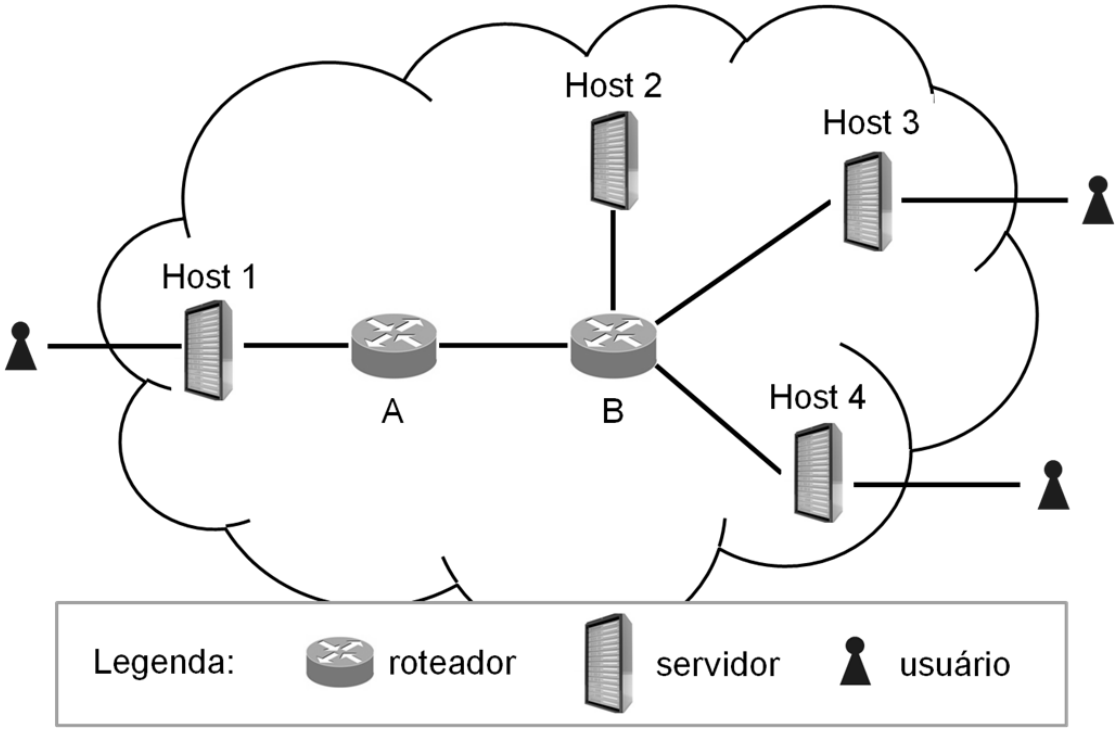

Figura 28: Topologia mínima

5.3.2 Topologia Lógica (Virtualizada)

A fim de configurar uma topologia similar à topologia mínima proposta na **Subseção 5.3.1**, foram utilizados 4 servidores disponíveis para a configuração de uma topologia lógica composta de equipamentos virtuais: três roteadores (A, B e C), três switches (VLAN) e cinco *hosts* (vHost1, vHost2a, vHost2a, vHost3 e vHost4). Os

hosts virtuais são na realidade máquinas virtuais executando o módulo Core. Em caso de criação de novas instâncias de aplicação, são criadas novas máquinas virtuais nos pontos determinados pelos *vHosts*. A **Figura 29** ilustra a topologia lógica configurada a partir da virtualização de equipamentos e redes locais.

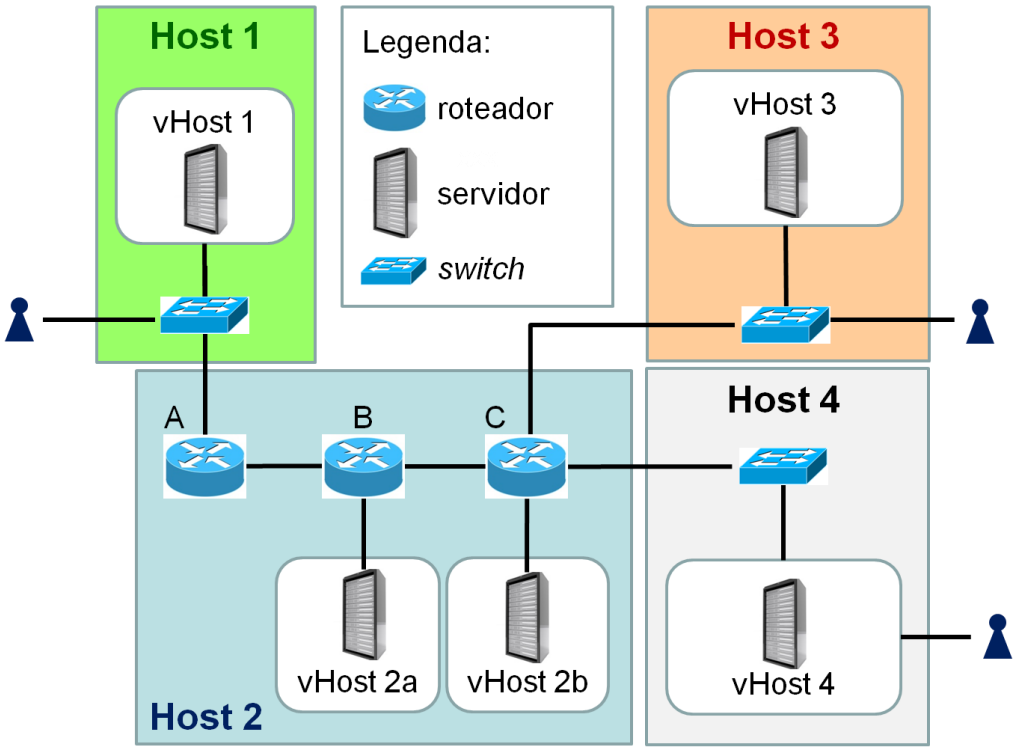

Figura 29: Topologia lógica

Como pode ser observado em detalhes na **Figura 29**, os *hosts* virtuais (*vHosts*) estão instanciados em diferentes *hosts* reais. No caso dos vHosts 2a e 3b, ambos são instanciados no mesmo *host* real. Junto a cada *host* virtual podem ser instanciados diferentes tipos de nós da aplicação de fornecimento de fluxos de vídeo (Ingestion, Relay ou Distribution). As cores em volta dos equipamentos virtuais representam em qual servidor físico estão sendo executadas as máquinas virtuais (a topologia física é apresentada na **Subseção 5.3.3**). Para a implementação do *multicast* em camada de aplicação, os *hosts* virtuais estão sempre próximos a um roteador, para que a comunicação entre o roteador e o nó da aplicação tenha um custo local (sem a necessidade de ser roteado para fora da nuvem). Foram criadas redes virtuais (VLANs) para isolar o tráfego das diferentes redes representadas na topologia lógica. A seguir, são descritas as características de *hardware* e *software* utilizados:

- **Especificação de hardware**: as máquinas virtuais instanciadas possuem um processador padrão com apenas um núcleo, 512 megabytes de memória RAM e 5 gigabytes de espaço em disco.

- **Especificação de software**: os seis servidores virtuais utilizam o sistema operacional Ubuntu Maverick 10.10 e os roteadores o software Vyatta 6.2-2011.02.09. Para o fornecimento de fluxos de vídeo foi utilizado o *VLC Media Player* em sua versão 0.98, tanto no serviço de distribuição de vídeo como no cliente (que consome os fluxos de vídeo). No cenário de testes detalhado na **Seção 5.2.3**, foi utilizado o pacote de *software* do sistema operacional Linux chamado *WGET*, que permite o download de dados utilizando os protocolos HTTP, HTTPS e FTP.

5.3.3 Topologia Física (Real)

Os cenários de testes propostos na **Seção 5.2** foram configurados a partir de uma topologia física composta de quatro servidores e um switch com funcionalidades de virtualização de redes locais (VLAN), o que permite o isolamento lógico de redes fisicamente compartilhadas. Na **Figura 29** é possível observar os equipamentos virtualizados instanciados por *host*, de acordo com as bordas coloridas. A cor verde representa o *host 1*, que virtualiza o vHost1 e um *switch* que conecta os clientes/provedores de conteúdo a uma mesma rede. A cor azul indica os equipamentos virtuais alocados no *host 2*, e engloba os três roteadores da topologia além dos v*Hosts* 2a e 2b. Os hosts 3 (cinza) e 4 (salmão), são semelhantes, possuindo um *vHost* e um *switch* cada.

A **Figura 30** ilustra a topologia física, de modo que os *hosts* são identificados pela numeração de 1 a 4. Os dois primeiros foram conectados através de um cabo *crossover*. Os *hosts* 2, 3 e 4 foram interligadas através de um *switch* e o tráfego de suas redes é isolado por meio de redes virtuais (VLANs).

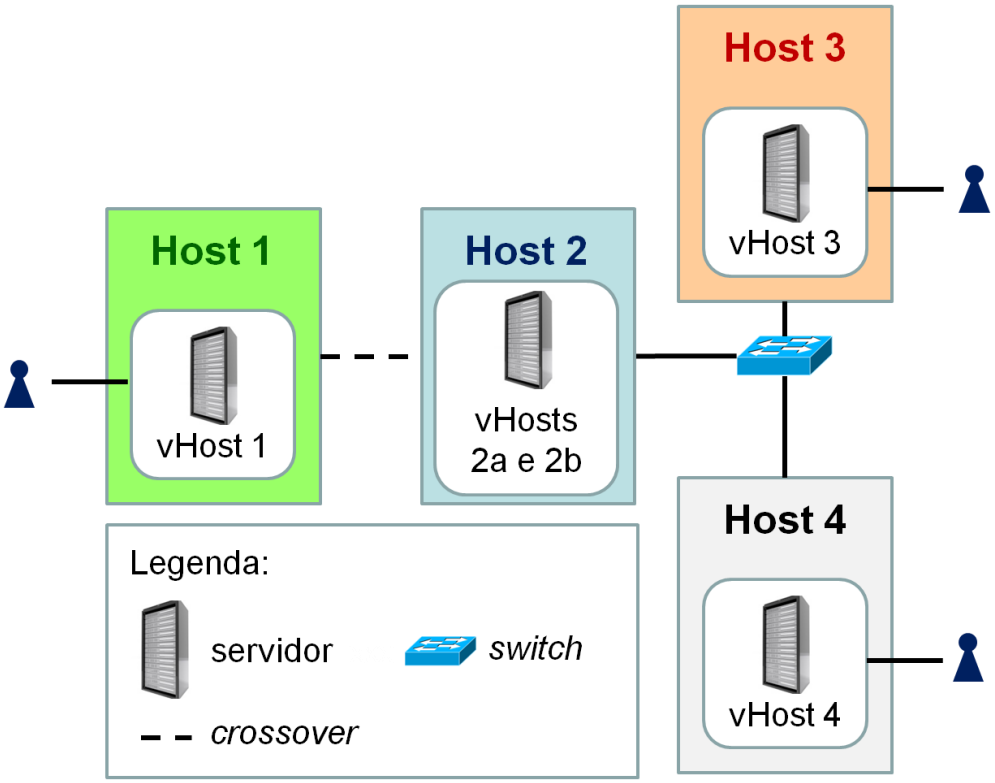

Figura 30: Topologia física

A seguir são descritas as características de *hardware* e *software* utilizados:

- **Especificação de hardware**: para a implementação do protótipo foram utilizados quatro servidores de mesma configuração: processadores Intel Xeon (Quadcore 2.13 GHz) com 12 gigabytes de memória RAM DDR3 e 800 gigabytes de espaço em disco, além de um switch Netgear GSM7328SO com funcionalidades de virtualização de redes.

- **Especificação de software**: todos os quatro servidores utilizam o sistema operacional Ubuntu versão 10.10 Maverick.

5.4 Caracterização do Conteúdo Utilizado nos Testes

O conteúdo de vídeo escolhido para o teste possui algumas características específicas necessárias para garantir a validação do funcionamento da solução. Seu principal atributo refere-se a sua duração, que deve estar de acordo com o tempo estimado para que a solução seja capaz de detectar e otimizar a distribuição de fluxos de vídeo. Este valor foi estimado com base em três tempos principais: de

detecção do fluxo redundante, de criação e configuração de uma máquina virtual e de redirecionamento dos fluxos redundantes para o nó recém-criado. A seguir, são detalhadas as estimativas que compõem o tempo total estimado (e posteriormente confirmado) para a realização da redução dos fluxos redundantes:

- **O tempo de detecção da redundância** foi estimado na ordem de alguns segundos, levando em consideração que o evento se dá pela comparação de requisições feitas a um mesmo nó;

- **O tempo de criação e configuração da máquina virtual** tende a ser o maior, pois trata-se da clonagem de uma máquina virtual utilizada como base (operação de leitura e escrita em disco rígido), sua inicialização e configuração. Assim, o tempo foi estimado na ordem de alguns minutos (2 a 3 minutos)

- **O tempo de redirecionamento dos fluxos** de vídeo foi estimado na ordem de dezenas de segundos.

Somando-se os tempos com uma margem de segurança, estimou-se um tempo de cerca de 4 minutos, esperando-se que o tempo total da otimização fosse menor que o estimado para a maioria dos casos. O vídeo escolhido possui duração total de aproximadamente 10 minutos, tempo suficiente para trafegar o fluxo de um único requisitante, o início do fluxo redundante, sua detecção e otimização e, ainda, a transmissão do fluxo otimizado para avaliação dos ganhos de banda.

Já a medição da transmissão de dados nos enlaces de rede é uma tarefa relativamente simples. Porém, é importante que a taxa de transmissão do vídeo seja alta o suficientepara que a mesma seja diferenciada das mensagens de controle da infraestrutura de rede ou do *middleware*. Como tais mensagens são enviadas em uma taxa inferior a dezenas de *kilobits* por segundo, um vídeo que é transmitido na taxa de *megabits* por segundo cumpre tal especificação.

Outra característica importante no vídeo escolhido é sua forma de licença de distribuição. O filme de curta duração chamado *Big Buck Bunny*, é um filme de código abeto, licenciado sob a *Creative Commons License Attribution 3.0*[21], que

[21] https://creativecommons.org/licenses/by/3.0/br/

permite o compartilhamento e adaptação do conteúdo de forma gratuita. Na **Tabela 14**, estão resumidas as características do conteúdo de vídeo escolhido (primeira coluna) e os valores para cada uma das características (segunda coluna).

Característica	Configuração
Titulo	Big Buck Bunny (código aberto)
Tamanho	350MB (367.863.808 bytes)
Duração (minutos:segundos)	09:58
Resolução	720 x 480 px
Data rate	4687kbps (0,586MBps)
Total bitrate	4921kbps (0,615MBps)
Frame rate	29 quadros por segundo
Formato	MPEG-2TS (*transport stream*)

Tabela 14: Configurações do vídeo escolhido

5.5 Método de Testes

Para a coleta dos dados analisados foram utilizadas duas fontes geradoras. A primeira fonte foram os arquivos de registro de eventos da aplicação, gerados automaticamente a partir da implementação de módulos adicionais presentes no *middleware* e na própria aplicação de distribuição de vídeo. A segunda fonte foram os dados coletados pela aplicação TCP Dump[22], que, ao ser executada, gera registros de todo o tráfego enviado e recebido pelas interfaces de uma máquina monitorada.

Como uma das métricas especificadas para a validação da aplicação é baseada no tempo e os nós da aplicação encontram-se distribuídos em diferentes hosts e máquinas virtuais, foi necessária a realização do processo de sincronização dos relógios desses computadores. O protocolo utilizado para essa sincronização foi o *Networking Time Protocol* (NTP). Realizada a sincronização dos relógios, foi possível analisar o comportamento da solução a partir do registro de eventos em ordem cronológica, mesmo em máquinas diferentes.

A compilação dos dados de tráfego de rede obtidos pelo TCP dump foi realizada de forma automática, através da criação de filtros na ferramenta Wireshark.

[22] http://www.tcpdump.org/

Tais filtros permitem que apenas os dados de interesse para análise sejam disponibilizados após o processamento do arquivo de registro original

Para garantir um nível mínimo de consistência dos dados coletados, cada experimento foi repetido por 10 vezes. Para realizar as tarefas repetitivas de execução do experimento, coleta e análise e compilação dos dados foi implementado um *script* na linguagem *Shell,* nativa em ambientes que utilizam o sistema operacional Linux.

5.6 Apresentação e Discussão dos Resultados

A seguir são apresentados e discutidos os resultados obtidos após a realização dos testes na **Seção 5.2.**

5.6.1 Resultados do Cenário 1

A seguir são apresentados os resultados obtidos sobre: o tempo necessário para a redução de fluxos redundantes, desde a detecção até a conclusão da otimização e a banda de rede utilizada durante a realização do experimento..

a. Tempo de Redução do Fluxo Redundante

A partir dos dados coletados pelos arquivos de registro de eventos da aplicação, foram observados os tempos necessários para: a detecção de uma requisição redundante (RR), a coleta de informações sobre as topologias de recurso junto ao módulo Core, o cálculo do ponto de otimização pelo Módulo LCP, a criação da máquina virtual (MV), a inicialização do Sistema Operacional de das aplicações (*boot*), e o tempo para o redirecionamento dos fluxos para o nó recém-instanciado. A **Tabela 1** apresenta cada um dos eventos e, seu tempo de duração, ou seja, o tempo necessário para a conclusão da tarefa e o tempo em que foi registrado o evento ao longo da execução do teste.

Evento	Duração (s)	Tempo (s)
Início do teste		0
RR detectada	10,004	20,004
Informações	0,155	20,159
LCP	0,012	20,171
Criação MV	74,494	94,665
Boot	45,004	139,741
Redirecionamento 1	0,523	140,264
Redirecionamento 2	0,044	140,308
TOTAL	**130, 236**	

Tabela 15: Resultados referentes à métrica de tempo de redução dos fluxos redundantes

Como pode ser observado na **Tabela 1**, as tarefas de obtenção das informações (~10 segundos), a criação da máquina virtual (~74,5 segundos) e a inicialização do Sistema Operacional e das aplicações (~45 segundos) são o tempos mais significativos na composição do tempo de redução dos fluxos redundante para o cenário 1. As medidas restantes ficaram todas abaixo da ordem de segundos. Na **Figura 31** é apresentada a composição do tempo de redução dos fluxos redundantes pelas tarefas mais significativas.

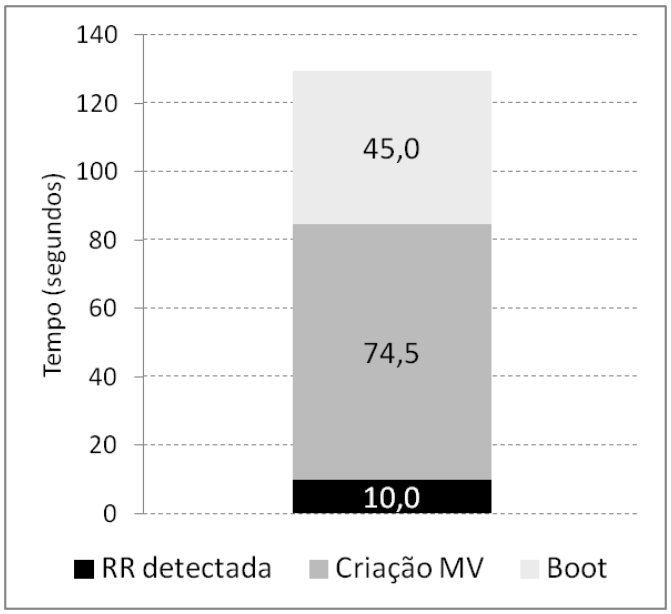

Figura 31: Composição do tempo de redução dos fluxos redundantes

No estudo de (MAO; HUMPHREY, 2012), foram levantados os tempos médios de inicialização de máquinas virtuais nos principais provedores de infraestrutura. Os resultados mostram que os valores variam, aproximadamente, entre 44 a 810 segundos. A partir da **Figura 31**, é possível observar que dos 130 segundos (2

minutos e 10 segundos) necessários para redução dos fluxos redundantes, cerca de 120 segundos são decorrentes da criação e inicialização da máquina virtual (dentro dos valores esperados) e apenas 10 segundos para a aplicação detectar a existência de requisições redundantes. Assim, o custo de tempo da solução Trade Wind representa apenas 8% do tempo total da redução dos fluxos redundantes.

b. Economia de Banda de Rede

Para a avaliação das métricas de consumo de banda, foram utilizados os aplicativos: TCP Dump, para coleta de dados de tráfego e Wireshark[23], para filtragem e análise dos arquivos gerados pelo TCP Dump. Os dados de consumo de banda foram coletados em dois enlaces da topologia apresentada na **Subseção 5.3.2:** entre os roteadores B e C (representando o tráfego total); e entre o roteador C e o vHost 3 (representando o Cliente 1). Na **Figura 32**, são apresentados os 200 primeiros segundos[24] do fornecimento dos fluxos de vídeo (duração de ~598 segundos).

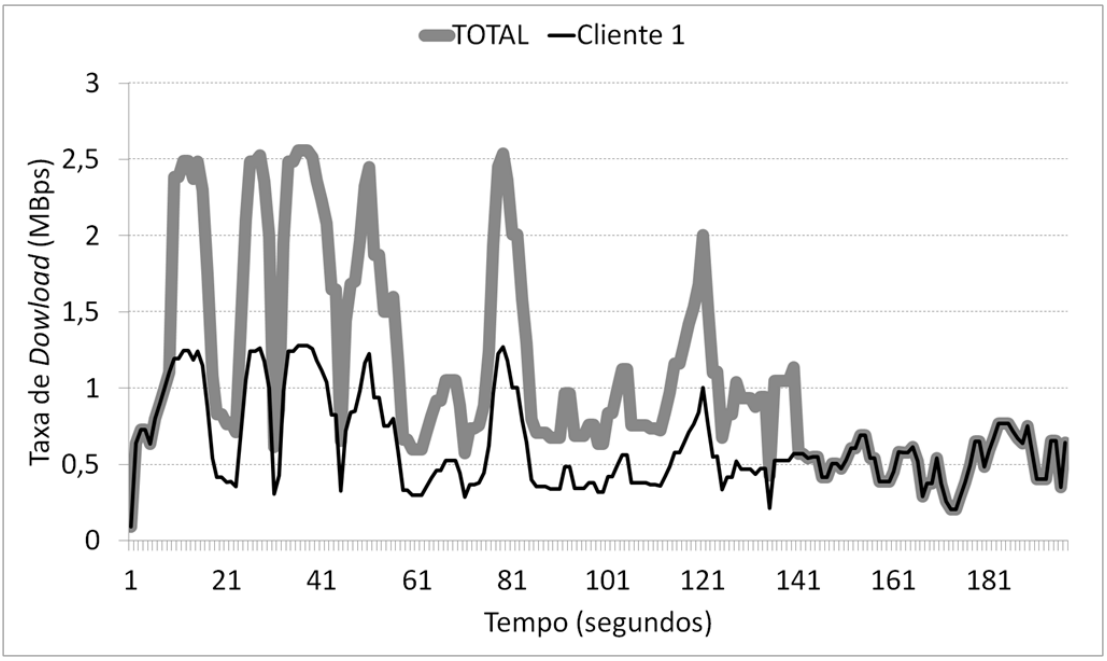

Figura 32: Análise do tráfego de dados no cenário 1

[23] https://www.wireshark.org/
[24] O tempo total de duração do conteúdo é de ~598 segundos, o que dificultaria a apresentação dos dados completos na **Figura 32**. A apresentação parcial dos dados não prejudica a observação da duplicação da taxa total na presença dos fluxos redundantes

A partir da **Figura 32,** é possível observar os instantes em que os eventos da descrição do cenário 1 (**Subseção 5.2.2**) são disparados.O teste inicia com a aquisição do primeiro fluxo de vídeo pelo Cliente 1 e a requisição realizada pelo Cliente 2 após 10 segundos. Note que o tráfego total dobra (total em 2,5 MBps e Cliente 1 em 1,25 MBps) e as linhas acompanham o tráfego do enlace do Cliente 1. Aproximadamente 130 segundos depois, os fluxos são reduzidos (próximo ao instante 141) e a linha da taxa total encontra-se com a linha da taxa do Cliente 1. Como pode ser observado nos resultados apresentados, a economia de recursos possui relação direta com o número requisições enviadas de diferentes nós Distribution. Quanto maior o número de nós Distribution requisitantes e a duração da aquisição dos fluxos de vídeo do vídeo, maior o potencial de economia de banda nos enlaces centrais da topologia pela redução dos fluxos redundantes transmitidos.

5.6.2 Resultados Cenário 2

O protocolo utilizado para o fornecimento dos fluxos de vídeo foi o HTTP, no qual não há um controle direto sobre taxa de transmissão dos dados. Este controle é realizado por mecanismos de controle de fluxo do protocolo TCP, como a variação do tamanho da janela de transmissão. O cenário 2 foi definido com objetivo de avaliar o comportamento da solução Trade Wind para a aquisição do fluxo de vídeo por meio da utilização de diferentes aplicações de aquisição de conteúdo.

A aplicação padrão utilizada nos testes, o VLC *Media Player*, realiza a aquisição do fluxo de acordo com a taxa de bits necessários para a visualização do vídeo (*bitrate*). Já a aplicação WGET, desenvolvida para *download* de dados a partir de um terminal de linha de comando, possui como objetivo a obtenção dos dados o mais rápido possível.

Os dados para avaliação do tempo de total e taxa de *download* do conteúdo foram medidos a partir do enlace entre os roteadores B e C da topologia apresentada na **Subseção 5.3.2**. Assim como nos testes realizados no cenário 1, foram utilizadas as aplicações TCP Dump e Wireshark para coleta e análise dos dados. A **Figura 33** ilustra a taxa de *download* do conteúdo de vídeo utilizando as

aplicações VLC player e WGET, entre os instantes 0 e 201[25]. Os resultados medidos separadamente foram agrupados no mesmo gráfico para fins de comparação entre as diferentes taxas de *download* e diferentes tempos de *download*.

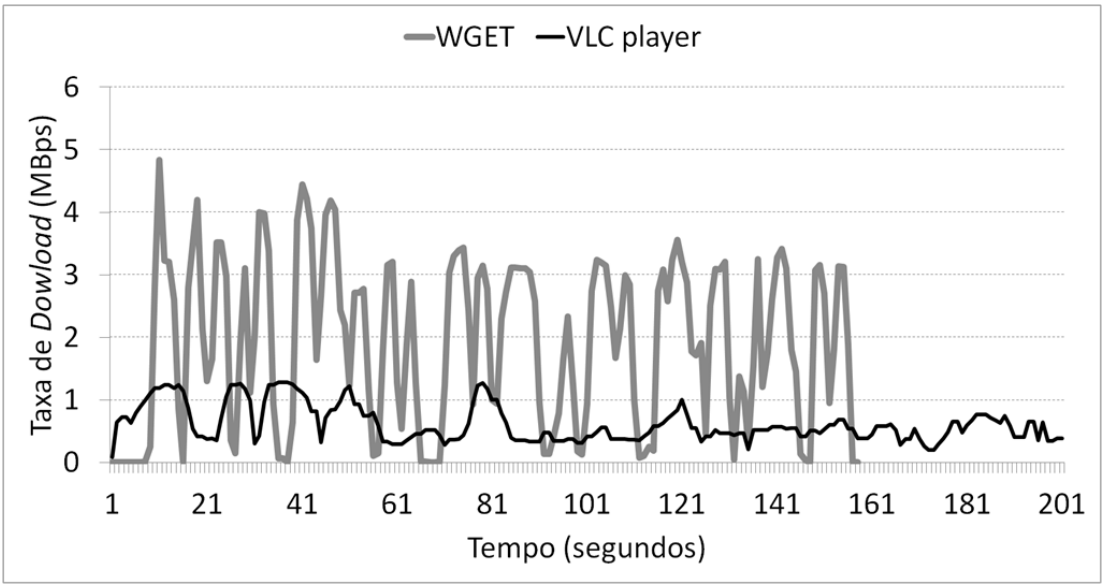

Figura 33: Análise do tráfego de dados no cenário 2

Como pode ser observado na figura **Figura 33**, a taxa de *download* praticada pela aplicação WGET é muito superior à taxa praticada pela aplicação VLC. Em média, a taxa medida na execução com WGET é de ~1,9 MBps, enquanto com VLC é de ~0,6 MBps, confirmando as características descritas no início desta subseção. O VLC possui uma taxa de *download* tecnicamente igual à taxa média de transcodificação do vídeo (0,615BMps) indicada na **Tabela 14**. Devido à diferença de taxas de *download*, os tempos totais de *download* também foram diferentes. Enquanto a aplicação WGET terminou o *download* do conteúdo em ~153 segundos, o VLC utilizou ~596 segundos (dois segundos a menos que tempo de duração do vídeo).

Com base nos resultados obsevados, é possível afirmar que há a necessidade de se implementar um mecanismo de armazenamento temporário de blocos, caso a aplicação de redução de fluxos redundantes do Trade Wind seja utilizada em conjunto com um serviço de fornecimento de fluxos de vídeo sob

[25] O tempo total de *download* do conteúdo pelo VLC é de ~596 segundos, o que dificultaria a apresentação completa dos dados na **Figura 33**. A apresentação parcial dos dados não prejudica a observação das diferenças de taxas de *download* entre as aplicações.

demanda[26]. Tal adaptação se faz necessária devido às diferentes configurações das taxas de *download* encontradas nas aplicações de visualização de fluxos de vídeo HTTP.

No caso do teste apresentado no cenário 3, o cliente utilizando o VLC *Media Player* como visualizador de conteúdo baixou os últimos blocos de conteúdo cerca de 440 segundos (7 minutos e 33 segundos) após a aplicação WGET. Em cálculos não precisos, considerando o tamanho do vídeo de 350MB e o tempo de duração 10 minutos, são transferidos aproximadamente 35MB de conteúdo por minuto. O mecanismo de armazenamento de blocos temporários teria de suportar um valor de 256,67MB de dados em cada nó Relay da aplicação de distribuição de fluxos de vídeo. Outra adaptação possível seria a limitação da taxa de *download* em um valor que auxiliasse na diminuição do tamanho da lista de blocos temporários armazenados (e.g., limitar a taxa de *download* num valor próximo ao *bitrate* do vídeo).

5.6.3 Cenário 3 – Multicanais

Proposto o modelo de desenvolvimento de aplicações elásticas e identificadas as funcionalidades que deveriam fazer parte do *middleware* da solução Trade Wind, foi implementado um mecanismo de gerenciamento de informações sobre os recursos disponíveis e alocados, sobre os usuários e suas instâncias de aplicação ativas, e informações de configuração sobre as aplicações desenvolvidas. Em especial as informações sobre os recursos alocados na forma de máquinas virtuais e instâncias de aplicação, permitiu a adição de uma nova funcionalidade à aplicação de fornecimento de fluxos de vídeo em tempo real com redução de fluxos redundantes desenvolvida na solução Wind: a criação de multicanais. Em outras palavras, com o gerenciamento dessas informações foi possível o fornecimento de fluxos de vídeo independentes e simultâneos, mesmo as instâncias de aplicação utilizando *hosts* em comum. Os testes dessa subseção têm por objetivo a validação da funcionalidade de multicanais.

---

[26] O fluxo de vídeo é gerado a partir de conteúdo previamente armazenado pelo provedor de conteúdo, diferente de um fluxo em tempo real, onde o fluxo de vídeo é gerado a partir da captação de imagens por meio de dispositivos como câmeras de vídeo.

Assim como nos cenários 2 e 3, foram utilizadas as aplicações TCP Dump e Wireshark na coleta e análise dos dados. A coleta de dados foi realizada por meio do monitoramento do enlace entre os roteadores B e C, da topologia apresentada na **Subseção 5.3.2**. A **Figura 34** ilustra a variação da taxa de *download* medida para o fluxo de vídeo 1 (linha em azul), para o fluxo de vídeo 2 (linha em vermelho) e o fluxo total (em cinza), entre os instantes 0 e 101[27].

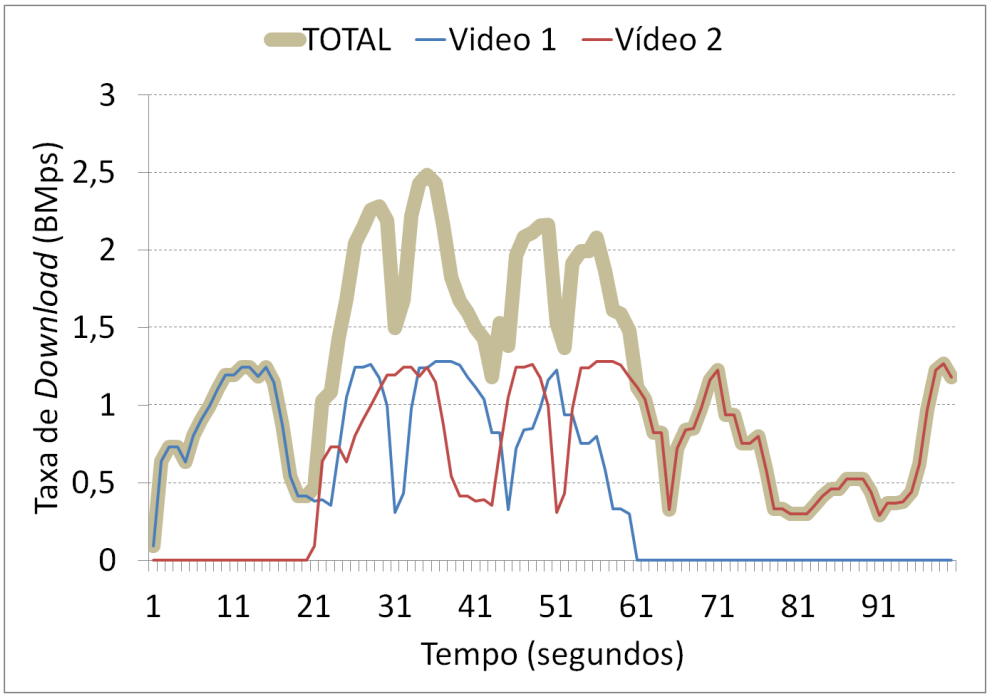

Figura 34: Análise do tráfego de dados no cenário 3

A partir da **Figura 34,** é possível observar os instantes em que os eventos descritos no cenário3 (**Subseção 5.2.4**) são disparados. O teste inicia com a aquisição do fluxo de vídeo I pelo Cliente 1. A requisição de aquisição do fluxo de vídeo II é realizada pelo Cliente 2 após 20 segundos. Note que o tráfego total aumenta no mesmo instante e a forma da curva não se assemelha à nenhuma das duas curvas. O tráfego volta a diminuir no instante 60, comprovando a transmissão dos fluxos simultâneos para diferentes canais.

[27] O tempo total de *download* do conteúdo pelo VLC é de ~596 segundos, o que dificultaria a apresentação completa dos dados na **Figura 34**. A apresentação parcial dos dados não prejudica a observação da sobreposição das taxas de *dowload* medidas durante a transmissão dos fluxos simultâneos.

5.6.4 Análise de Limitações da Solução

As modificações propostas na solução Trade Wind para permitir a separação de funcionalidades comuns a aplicações elásticas na organização de um *middleware* que as oferece em forma de serviço, introduziram elementos adicionais na aplicação de fornecimento de fluxos de vídeo em tempo real com a redução de fluxos redundantes, impactando no consumo de recursos da nuvem. Esta subseção se dedica à discussão dos impactos das decisões de projeto da solução Trade Wind e suas limitações.

a. Custos Computacionais Adicionais do Trade Wind

Dentre as limitações encontradas na solução é importante destacar a sobrecarga adicionada pela utilização do Trade Wind. Para sua execução, é necessária uma máquina virtual dedicada à execução do *middleware* em cada *host* da nuvem (módulo Core e banco de dados distribuído), além da execução de um módulo de controle e atuação em cada máquina virtual instanciada para aplicações (módulo Core Access).

A especificação da máquina virtual instanciada para a execução dedicada do *middleware* é a mesma das máquinas instanciadas para a execução das aplicações dos usuários. Cada máquina virtual possui um processador padrão com apenas um núcleo, 512 megabytes de memória RAM e 5 gigabytes de espaço em disco. Em trabalhos futuros, recomenda-se o estudo de um tamanho mínimo de máquina virtual dedicada à execução do *middleware*, com o intuito de minimizar a utilização de recursos da nuvem com a solução Trade Wind.

b. Limitação em Relação ao Tipo de Fluxo de Vídeo

Por meio dos resultados obtidos nos testes realizados no cenário 3 (detalhado na **Subseção 5.6.3),** observou-se que quando os clientes requisitantes utilizam diferentes aplicações para a aquisição de um mesmo fluxo de vídeo, a variação nos valores padrão para as taxas de *download* inviabiliza a transmissão de fluxos sob demanda, com a organização atual da aplicação de redução de fluxos redundantes. Para a transmissão de fluxos de vídeo sob demanda, é necessária a implementação de mecanismos de armazenamento temporário de blocos em todos os nós da

aplicação ou de mecanismos para a restrição da taxa de *download* dos dispositivos com taxas mais altas.

c. Sobrecarga de Mensagens de Controle

Uma dificuldade encontrada na avaliação da solução foi a medição do tamanho das mensagens de controle utilizadas pela solução Trade Wind e seu impacto no consumo de banda de rede. Parte da dificuldade foi devida à limitação da ferramenta de análise dos arquivos gerados pelo monitoramento do tráfego de rede durante o experimento, a qual não permite a identificação de mensagens específicas em camada de aplicação.

A fim de proporcionar uma estimativa do impacto da solução no consumo de banda, foi analisada a quantidade de mensagens de controle enviadas pela solução para a execução da rotina de redução dos fluxos redundantes. Cada mensagem possui cerca de 50 *kilobits* e são trocadas em torno de 100 mensagens por otimização realizada, totalizando ~0,6 *megabytes* trafegados. Considerando o tamanho total do vídeo transmitido nos testes de aproximadamente 350 megabytes, as mensagens de controle representam apenas 0,17% dos dados trafegados ao longo da transmissão do vídeo, um valor que pode ser considerado negligenciável.

6 CONSIDERAÇÕES FINAIS

Neste trabalho foi apresentado o Trade Wind, uma solução que permite o desenvolvimento de aplicações e serviços distribuídos para o gerenciamento automático de recursos e elasticidade em nuvens de computação. Por meio de estudo realizado sobre soluções de elasticidade para nuvens de computação, observou-se a concentração de soluções de elasticidade com foco no gerenciamento de recursos de processamento e armazenamento para aplicações do tipo cliente-servidor. Essas soluções implementam o gerenciamento automatizado de recursos através de mecanismos de infraestrutura, por meio de políticas automáticas reativas.

As políticas automáticas reativas utilizam a descrição de regras para definir métricas a serem monitoradas, e que atingido um valor limítrofe, disparam ações de ajuste dos recursos da nuvem à nova demanda. Para a garantia do funcionamento das regras, dois mecanismos são recorrentes na descrição das soluções de elasticidade: os mecanismos de definição e monitoramento de métricas e os mecanismos de controle de elasticidade. Comumente implementados na camada de infraestrutura da nuvem, essas soluções encontram certas dificuldades na definição e monitoramento de regras para o atendimento das demandas variáveis.

A partir do estudo do Wind, uma aplicação distribuída que fornece fluxos de vídeo em tempo real e realiza a redução automática de fluxos reundantes, foram identificadas diferenças de implementação em relação às soluções de elasticidade estudadas, principalmente, a implementação do mecanismo de monitoramento das métricas utilizadas como gatilhos em camada de aplicação. O mecanismo de definição e monitoramente de métricas do Wind permite a definição de regras mais específicas e precisas, para uma aplicação de redução de fluxos de vídeo redundantes.

Com base nessas observações, foi proposto um modelo de desenvolvimento de aplicações elásticas que define quatro funcionalidades (**Tabela 12**): definição e monitoramento de métricas e condições de regras, verificação de possibilidade de otimização, gerenciamento de informações sobre recursos e aplicações e a

execução das ações de elasticidade. A partir deste modelo, foram definidas as funcionalidades específicas, que deveriam ser implementadas pelos desenvolvedores e as funcionalidades comuns a aplicações elásticas, que poderiam ser reutilizadas na forma de um *middleware.*

Partindo da definição do *middleware* e seus serviços, foi proposta uma solução completa de gerenciamento de recursos e elasticidade chamada Trade Wind, que pode ser estendida de acordo com um modelo de composição de aplicações a partir da implementação de funcionalidades e serviços. Como prova de conceito, foi adaptada a aplicação de fornecimento de fluxos de vídeo com redução automática de fluxos redundantes para o funcionamento a partir do *middleware*. Para a avaliação e validação da solução proposta, foi especificado e implementado um ambiente de testes composto de cenários de execução, uma topologia, um conjunto de métricas a serem avaliadas, a caracterização do conteúdo de vídeo utilizado e o detalhamento do método de realização dos testes.

Os resultados obtidos validaram o funcionamento da aplicação de prova de conceito adaptada para o funcionamento em conjunto com o Trade Wind, assim como sua funcionalidade adicional de fornecimento de fluxos de vídeo em multicanais. Os resultados também indicaram que o Trade Wind e a aplicação de redução de fluxos de vídeo redundantes impactam em apenas 8% no tempo total de adaptação da infraestrutura, sendo o tempo de criação e inicialização da máquina virtual responsável pelo maior tempo consumido neste processo. A aplicação de redução de fluxos redundantes também provou reduzir pela metade o consumo de banda no cenário de teste, tendo potencial de maior economia no caso de adição de mais fluxos redundantes.

6.1 Contribuições

Entre as principais contribuições deste trabalho estão:

- A proposta de um modelo de desenvolvimento de aplicações elásticas, definindo as funcionalidades a serem implementadas de forma específica pelas aplicações de elasticidade;

- Um *middleware* que implementa duas funcionalidades de propósito geral propostas pelo modelo de desenvolvimento de aplicações elásticas. A primeira funcionalidade disponível é a manipulação de máquinas virtuais e aplicações, e a segunda, o gerenciamento de informações sobre a topologia de recursos disponíveis, a topologia de instâncias alocadas, os usuários, as aplicações e suas configurações;

- Implementação de uma aplicação baseada no Wind (**Seção 3.5**) para o fornecimento de fluxos de vídeo com redução automática de recursos, com adição da funcionalidade de fornecimento de fluxo de vídeo por multicanais simultâneos.

6.2 Trabalhos Futuros

Devido à limitação de tempo e à necessidade de prioridade/foco do estudo, alguns resultados interessantes relacionados à este trabalho de pesquisa tiveram de ser renunciados. Nesta seção, são apresentados e discutidos de forma superficial tais estudos relacionados.

- Medição de valores de impacto da solução no desempenho da aplicação de fornecimento de fluxos de vídeo em tempo real: sobrecarga de processamento, sobrecarga de memória e tempo de resposta do serviço de entrega de fluxo de vídeo em tempo real para os diversos cenários propostos;

- Refinamento dos requisitos de *hardware* e *software* das máquinas virtuais responsáveis pelos mecanismos de controle/gerenciamento de recursos do *host* (máquinas virtuais que executam o módulo Core): através da criação de máquinas virtuais com diferentes configurações, propondo uma quantidade mínima de recursos necessários para a execução da solução;

- Avaliação do impacto da diminuição dos recursos disponíveis no tempo de resposta do sistema (chegar à quantidade ideal de recursos a serem utilizadas)

- Execução de teste em ambiente mais próximo do real (sem virtualização de recursos como rotadores e switches);
- Extensão de serviços de nível plataforma para diversas aplicações, por exemplo jogos *online*, *load balancing*, entre outros;
- Criação de modelo de controle de acesso para mediar API e *Middleware*.

REFERÊNCIAS

ALI-ELDIN, A.; TORDSSON, J.; ELMROTH, E. An adaptive hybrid elasticity controller for cloud infrastructures. In: Proceedings of the 2012 IEEE Network Operations and Management Symposium, Maui, HI, USA: IEEEComputer Society, [S.l.], 2012. (NOMS). p. 204-212. ISSN 1542-1201. Disponível em: <http://dx.doi.org/10.1109/NOMS.2012.6211900>.

ARMBRUST, M.; FOX, A.; GRIFFITH, R.; JOSEPH, A. D.; KATZ, R.; KONWINSKI, A.; LEE, G.; PATTERSON, D.; RABKIN, A.; STOICA, I.; ZAHARIA, M. Above the Clouds: A View of Cloud Computing. Communication ACM, ACM, New York, NY, USA, v. 53, n. 4, Apr. 2010. Disponível em: <http://dl.acm.org/citation.cfm?doid=1721654.1721672>.

BADGER, L. ; GRANCE, T.; CORNER, T. P.; VOAS, J. Recommendations of the National of Standards and Technology. Special Publication 800-146. NIST, May 2012. Disponível em: <http://csrc.nist.gov/publications/nistpubs/800-146/sp800-146.pdf>. Acesso em: 13/06/2013.

BULTERMAN, R. W. et al. On computing a longest path in a tree. Information Processing Letters. Elsevier, [S.l], v. 81, n. 2, p. 93-96, Jan. 2002. Disponível em: <http://dx.doi.org/10.1145/78952.78953>.

CALHEIROS, R. N. et. al. The Aneka platform and QoS-driven resource provisioning for elastic applications on hybrid Clouds. Future Generation Computer Systems, Elsevier Science Publishers B. V., Amsterdam, The Netherlands, The Netherlands, v. 28, n. 6, June 2012. Disponível em: <http://dx.doi.org/10.1016/j.future.2011.07.005>.

CHAPMAN, C. et al. Elastic Service Management in Computational Clouds. In Proceedings of the 2010 1st IEEE/IFIP International International Workshop on Cloud Management, Osaka, Japan: IEEE Computer Society, [S.l], 2010. (CloudMan ‘2010).

CHELLAPPA, Ramnath K. Intermediaries in Cloud-Computing. Dallas: UFRGS, 26-29, 1997. Reunião Anual da INFORMS.

CHENG, B. MediaPaaS: a Cloud-based Media Processing Platform for Elastic Live Broadcasting. In: Proceedings of the 2014 7th IEEE International Conference on Cloud Computing, Anchorage, AL, USA: IEEE Computer Society, [S.l], 2014. (CLOUD). p. 713-720. Disponível em: <http://dx.doi.org/10.1109/CLOUD.2014.100>.

DEERING, S. E.; CHERITON, D. R. Multicast Routing in Datagram Internetworks and Extended LANs. ACM Transactions on Computer Systems. ACM, New York, NY, USA, v. 8, n. 2, p. 95-110, May 1990. Disponível em: <http://doi.acm.org/10.1145/78952.78953>.

DIJKSTRA, E. W. A note on two problems in connexion with graphs. Numerische Mathematik, Springer-Verlag, [S.l.], v. 1, n. 1, p. 269-271, Dec. 1959. ISSN 0945-3245. Disponível em: <http://link.springer.com/article/10.1007%2FBF01386390>.

FITÓ, J. O.; GOIRI, I.; GUITART, J. SLA-driven Elastic Cloud Hosting Provider. In: Proceedings of the 2010 18th Euromicro International Conference on Parallel, Distributed and Network-Based Processing, Pisa, Italy: IEEE Computer Society, [S.l.], 2010. (PDP). p. 111-118. ISSN 1066-6192. Disponível em: <http://dx.doi.org/10.1109/PDP.2010.16>.

GALANTE, G.; BONA, L. C. E. A Survey on Cloud Computing Elasticity. In: Proceedings of the 2012 IEEE Fifth International Conference on Utility and Cloud Computing, Chicago, IL, USA: IEEE Computer Society, [S.l.], 2012. (UCC) p. 263-270. ISBN 978-1-4673-4432-6. Disponível em: <http://dx.doi.org/10.1109/UCC.2012.30>.

HERBST, N. R.; KOUNEV, S.; REUSSNER, R. Elasticity in Cloud Computing: What It Is, and What It Is Not. In: Proceedings of the 10th International Conference on Autonomic Computing. San Jose, CA, USA;. USENIX, 2013. (ICAC '13), p. 23-27. ISBN 978-1-931971-02-7. Disponível em: <https://www.usenix.org/conference/icac13/technical-sessions/presentation/herbst>.

HOFMANN, M.; BEAUMO, L. R. The Diversity of Interests in Content Networking. In: _____. Content Networking: Architecture , Protocols, and Practice. San Francisco: Elsevier, 2005. p. 21.

HOSSEINI, M. et al. A survey of application-layer multicast protocols. Communications Surveys & Tutorials. IEEE, [S.l.], v. 9, n. 3, p. 58-74, Third Quarter 2007. ISSN 1553-877X. Disponível em: <http://dx.doi.org/10.1109/COMST.2007.4317616>.

KO, S.; PARK, S.; HAN, H. Design Analysis for Real-time Video Transcoding on Cloud Systems. In: Proceedings of the 28th Annual ACM Symposium on Applied Computing, Coimbra, Portugal> ACM, New York, NY, USA, 2013. (SAC '13). p. 1610-1615. ISBN 978-1-4503-1656-9. Disponível em: <http://doi.acm.org/10.1145/2480362.2480663>.

KRANAS, P. et al. ElaaS: An innovative Elasticity as a Service framework for dynamic management across the cloud stack layers. In: Proceedings of the 2012 Sixth International Conference on Complex, Intelligent and Software Intensive Systems, Palermo, Italy: IEEE Computer Society, [S.l.] 2012. (CISIS). p.1042-1049. ISBN 978-1-4673-1233-2. Disponível em: <http://dx.doi.org/10.1109/CISIS.2012.117>.

LIANG, L. The cloud video material transfer code system design in the global station network environment. In: Proceedings of the 2012 International Conference on Image Analysis and Signal Processing, Hangzhou, China: IEEE Compurter Society, [S.l.], 2012. (IASP). p. 1-3. ISBN 978-1-4673-2547-9 Disponível em: <http://dx.doi.org/10.1109/IASP.2012.6425033>.

LIM, H. C. Automated Control in Cloud Computing: Challenges and Opportunities. In: Proceedings of the 1st Workshop on Automated Control for Datacenters and Clouds, Barcelona, Spain: ACM, New York, NY USA, 2009. (ACDC '09). p. 13-18. ISBN 978-1-60558-585-7. Disponível em: <http://doi.acm.org/10.1145/1555271.1555275>.

LIU, Y. et al. Investigating Redundant Internet Video Streaming Traffic on iOS Devices: Causes and Solutions. IEEE Transactions on Multimedia, [S.l.], v. 16, n. 2, p.510-520, Feb. 2014. ISSN 1520-9210. Disponível em: <http://dx.doi.org/10.1109/TMM.2013.2293312>.

MA, H.; SEO, B.; ZIMMERMANN, R. Dynamic Scheduling on Video Transcoding for MPEG DASH in the Cloud Environment. In: Proceedings of the 5th ACM Multimedia Systems Conference, Singapore, Singapore: ACM, New York, NY, USA, 2014. (MMSys '14). p. 283-294. ISBN 978-1-4503-2705-3. Disponível em: <http://doi.acm.org/10.1145/2557642.2557656>.

MAO, M.; HUMPHREY, M. A Performance Study on the VM Startup Time in the Cloud. In: Proceedings of the 2012 IEEE 5th International Conference on Cloud Computing, HONOLULU, HI, USA: IEEE Computer Society, [S.l.], 2012. (CLOUD), p. 423-430. ISBN 978-1-4673-2892-0. Disponível em: <http://dx.doi.org/10.1109/CLOUD.2012.103>.

MARSHALL, P.; KEAHEY, K.; FREEMAN, T. Elastic Site: Using Clouds to Elastically Extend Site Resources. In: Proceedings of the 2010 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing, Melbourne, Australia: IEEE Computer Society, [S.l.], 2010. (CCGrid). p. 43-52. Disponível em: <http://dx.doi.org/10.1109/CCGRID.2010.80>.

MCDONALD, M. P. CIO Agenda 2013 - Implications for High-Tech Providers. Disponível em: <http://www.gartner.com/it/content/2299600/2299616/february_5_2013_http_cio_agenda_mmcdonald.pdf?userId=68342810>. Acesso em: 09/06/2013

MELL, P.; GRANCE, T. The NIST Definition of Cloud Computing. Special Publication 800-145. NIST, Sept. 2011. Disponível em: <http://csrc.nist.gov/publications/nistpubs/800-145/SP800-145.pdf>. Acesso em: 21/06/2013.

MIKITYUK, A.; SEIFERT, J. –P.; FRIEDRICH, O. The virtual Set-Top Box: On the shift of IPTV service execution, service & UI composition into the cloud. In: Proceedings of the 2013 17th International Conference on Intelligence in Next Generation Networks, Venice, Italy: IEEE Computer Society, [S.l.], 2013. (ICIN). p. 1-8. Disponível em: <http://dx.doi.org/10.1109/ICIN.2013.6670887>.

SANDVINE. Sandvine: Global Internet Phenomena Report 1H 2013. [S.l.], 2013. Disponível em: <https://www.sandvine.com/downloads/general/global-internet-phenomena/2013/sandvine-global-internet-phenomena-report-1h-2013.pdf>.

SATYANARAYANAN, M. Pervasive computing: vision and challenges. IEEE Personal Communications, IEEE, [S.l.], v. 8, n. 4, p. 10-17, Aug. 2001. ISSN 1070-9916. Disponível em: <http://dx.doi.org/10.1109/98.943998>.

VILLEGAS,D. et al. Cloud federation in a layered service model. Journal of Computer and System Sciences, Elsevier, [S.l], v. 78, n. 5, p. 1330-1344, Sept. 2012. ISSN 0022-0000. Disponível em: <http://dx.doi.org/10.1016/j.jcss.2011.12.017>.

ZHANG, Q.; CHENG, L.; BOUTABA, R. Cloud computing: state-of-the-art and research challenges. Journal of Internet Services and Applications, Springer-Verlag, [S.l.], v. 1, n. 1, p. 7-18, May 2010. ISSN 1869-0238. Disponível em: <http://link.springer.com/article/10.1007%2Fs13174-010-0007-6>.

ZHUANG, Z; GUO, C. Building cloud-ready video transcoding system for Content Delivery Networks (CDNs). In: Proceedings of the 2012 IEEE Global Communications Conference, Anaheim, CA, USA: IEEE Computer Society, [S.l.], 2012. (GLOBECOM). p. 2048-2053. ISBN 978-1-4673-0919-6. Disponível em: <http://dx.doi.org/10.1109/GLOCOM.2012.6503417>.

APÊNDICE A – PUBLICAÇÕES E PREMIAÇÕES

Artigos aceitos e publicados em anais de conferências internacionais durante o programa de mestrado:

- Goya, W. A. ; Andrade, M. R. ; Zucchi, A. C. ; Gonzalez, N. M. ; Pereira, R. F. ; Langona, K.; Carvalho, T. C. M. B. ; Mangs, J. ; Sefidcon, A. ***The Use of Distributed Processing and Cloud Computing in Agricultural Decision-Making Support Systems***. Em: CLOUD 2014 - IEEE 7th International Conference on Cloud Computing. 8 pags. Alaska/EUA, 2014.

- Pereira, R. F. ; Goya, W. A. ; Langona, K.; Gonzalez, N. M. ; Carvalho, T. C. M. B. ; Mangs, J. ; Sefidcon, A. ***Exploiting Hadoop Topology in Virtualized Environments***. Em: SERVICES 2014 – IEEE 10th World Congress on Services. 8 pags. Alaska/EUA, 2014.

- Leal, R. R. ; Simplicio Junior, M. A. ; Santos, M. A. S. ; Gomes, M. A. L. ; Goya, W. A. . ***Cheating detection in P2P online trading card games***. Em: XIII Simpósio Brasileiro em Segurança da Informação e de Sistemas Computacionais (SBSeg). v. 13. p. 170-183. Manaus/AM, 2013. Anais do XIII Simpósio Brasileiro em Segurança da Informação e de Sistemas Computacionais. Porto Alegre/RS: Sociedade Brasileira de Computação (SBC), 2013.

- Zucchi, A. C. ; Gonzalez, N. M.; Andrade, M. R. ; Pereira, R. F. ; Goya, W. A. ; Langona, K. ; Carvalho, T. C. M. B. ; Mangs, J. ***How Advanced Cloud Services Can Improve Gaming Performance***. Em: CloudCom 2013 - 5th IEEE International Conference on Cloud Computing Technology and Science. 8 pags. Bristol/Reino Unido, 2013.

- Gonzalez, N. M ; Goya, W. ; Langona, K. ; Pereira, R. F. ; Andrade, M. R. ; Carvalho, T. C. M. B. ; Zucchi, A. C. ; Mangs, J. ; Sefidicon, A. 2013. ***Cloud distributed processing using Trade Wind***. Em: IEEE 2nd International Congress on Big Data. 4 pags. Califórnia/EUA: 27 de junho – 2 Junho de 2013.

- Pereira, R. F. ; Andrade, M. R. ; Zucchi, A. C. ; Langona, K. ; Goya, W. ; Gonzalez, N. M. ; Carvalho, T. C. M. B. ; Mangs, J. ; Sefidicon, A. 2013. ***Distributed processing from large scale sensor network using Hadoop***. Em: IEEE 2nd International Congress on Big Data. 4 pags. Califórnia/EUA: 27 de junho – 2 Junho de 2013.

- Miers, Charles; Barros, Marcel; Simplício, Marcos; Gonzalez, Nelson; Evangelista, Pedro; Goya, Walter; Carvalho, Tereza et al. 2011. ***Using Trade Wind to sail in the clouds***. Proceedings of Parallel and Distributed Computing and Systems (PDCS), 8 pags. Dallas/EUA: 14-16 de Dezembro de 2011.

Artigo completo aceito e publicado em periódico internacional durante o programa de mestrado:

- Simplicio, M. A.; Santos, M. A. S.; Leal, R. R ; Gomes, M. A. L. ; Goya, W. A. ***SecureTCG: a lightweight cheating-detection protocol for P2P multiplayer online trading card games***. Security and Communication Networks, 2014.

Trabalhos de anteriores ao mestrado com participação do autor:

- Miers, C. C. ; Simplicio Junior, M. A. ; Goya, W. A. ; Carvalho, T. C. M. B. ; Souza, V. . ***An architecture for P2P locality in managed networks using hierarchical trackers***. Em: 6th International Conference on Network and Service Management Conference on Network and Service Management (CNSM), p. 206-213. Niagara Falls., 2010.

- Evangelista, P.; Amaral, M.; Miers, C.; Goya, W.; Simplicio, M.; Carvalho, T.; Souza, V.; , "***EbitSim: An Enhanced BitTorrent Simulation Using OMNeT++ 4***", Modeling, Analysis & Simulation of Computer and Telecommunication Systems (MASCOTS), 2011 IEEE 19th International Symposium, pags.437-440. Cingapura/Cingapura 25-27 de Julho de 2011

Prêmios recebidos pela apresentação dos estudos durante a escrita da dissertação:

- **(2013) Menção Honrosa por trabalho de Mestrado** no II Workshop de Pós-Graduação da Área de Concentração Engenharia de Computação (WPG-EC)., Comissão Coordenadora da Área de Concentração "Engenharia de Computação" - Poli/USP.

- **(2012) Menção Honrosa por trabalho de Mestrado** no I Workshop de Pós-Graduação da Área de Concentração Engenharia de Computação (WPG-EC)., Comissão Coordenadora da Área de Concentração "Engenharia de Computação" - Poli/USP.