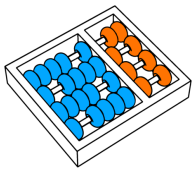
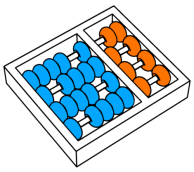


2013

Milton Aparecido Soares Junior

"Uma arquitetura de gerência autonômica de redes virtuais baseada em sistemas multiagentes"

CAMPINAS



Universidade Estadual de Campinas
Instituto de Computação

Milton Aparecido Soares Junior

"Uma arquitetura de gerência autonômica de redes virtuais baseada em sistemas multiagentes"

Orientador(a): **Prof. Dr. Edmundo Roberto Mauro Madeira**

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Ciência da Computação do Instituto de Computação da Universidade Estadual de Campinas para obtenção do título de Mestre em Ciência da Computação.

Este exemplar corresponde à versão final da Dissertação defendida por Milton Aparecido Soares Junior, sob orientação de Prof. Dr. Edmundo Roberto Mauro Madeira.

Assinatura do Orientador(a)

CAMPINAS

FICHA CATALOGRÁFICA ELABORADA POR
MARIA FABIANA BEZERRA MULLER - CRB8/6162
BIBLIOTECA DO INSTITUTO DE MATEMÁTICA, ESTATÍSTICA E
COMPUTAÇÃO CIENTÍFICA - UNICAMP

Soares Junior, Milton Aparecido, 1984-
So11a Uma arquitetura de gerência autonômica de redes virtuais baseada em sistemas multiagentes / Milton Aparecido Soares Junior. – Campinas, SP : [s.n.], 2013.

Orientador: Edmundo Roberto Mauro Madeira.
Dissertação (mestrado) – Universidade Estadual de Campinas, Instituto de Computação.

1. Redes de computadores - Gerência. 2. Virtualização de redes. 3. Sistemas multiagentes. I. Madeira, Edmundo Roberto Mauro,1958-. II. Universidade Estadual de Campinas. Instituto de Computação. III. Título.

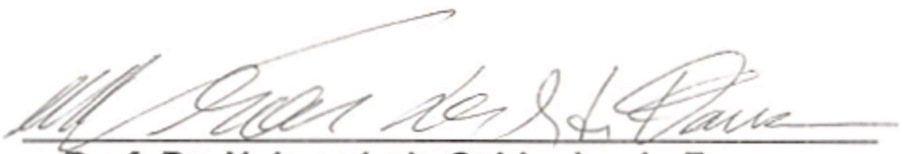
Informações para Biblioteca Digital

Título em inglês: An architecture for autonomic management of virtual networks based on multi-agent systems
Palavras-chave em inglês:
Computer networks - Management
Network virtualization
Multi-agent systems
Área de concentração: Ciência da Computação
Titulação: Mestre em Ciência da Computação
Banca examinadora:
Edmundo Roberto Mauro Madeira [Orientador]
Nelson Luis Saldanha da Fonseca
Otto Carlos Muniz Bandeira Duarte
Data de defesa: 30-01-2013
Programa de Pós-Graduação: Ciência da Computação

TERMO DE APROVAÇÃO

Dissertação Defendida e Aprovada em 30 de Janeiro de 2013, pela Banca examinadora composta pelos Professores Doutores:

Prof. Dr. Otto Carlos Muniz Bandeira Duarte
COPPE / UFRJ



Prof. Dr. Nelson Luis Saldanha da Fonseca
IC / UNICAMP

Prof. Dr. Edmundo Roberto Mauro Madeira
IC / UNICAMP

Instituto de Computação
Universidade Estadual de Campinas

Uma arquitetura de gerência autonômica de redes virtuais baseada em sistemas multiagentes

Milton Aparecido Soares Junior[1]

30 de Janeiro de 2013

Banca Examinadora:

- Prof. Dr. Edmundo Roberto Mauro Madeira (Orientador)
- Prof. Dr. Nelson Luis Saldanha da Fonseca
  IC - UNICAMP
- Prof. Dr. Otto Carlos Muniz Bandeira Duarte
  COPPE - UFRJ
- Profa. Dra. Islene Calciolari Garcia
  IC - UNICAMP (Suplente)
- Prof. Dr. Fábio Luciano Verdi
  DComp - UFSCar (Suplente)

[1]Suporte financeiro de: bolsa da CAPES de 03/09 a 02/11 e bolsa da FINEP de 03/11 a 03/12.

Abstract

Despite its success, the current architecture of the Internet is a source of many problems for current applications and future demands. The virtualization of network infrastructure is proposed as an alternative to solve these problems without the need to change the core of the Internet, as it enables the network architecture pluralism. We have developed an architecture for autonomic management of virtual networks based on multi-agent systems. Based on this architecture, we implemented a prototype that performs the function of self-healing virtual networks. New algorithms and mechanisms have been developed to improve the efficiency of the prototype. A case study on the management of virtual networks that takes into consideration the requirements of the applications that are running on a cloud is also presented. For the execution of the experiments was created a experimentation platform based on virtual machines and on OpenFlow. The prototype and the platform integrate current tools creating a single solution for management of virtual networks. The results contributed to bring network virtualization and autonomic management closer to reality.

Resumo

Apesar do seu sucesso, a arquitetura atual da Internet é uma fonte de vários problemas para as aplicações atuais e as demandas futuras. A virtualização da infraestrutura da rede é proposta como alternativa para solucionar esses problemas sem a necessidade de alterar o núcleo da Internet, pois ela habilita o pluralismo de arquiteturas de rede. Neste trabalho, foi desenvolvida uma arquitetura de gerência autonômica de redes virtuais baseada em sistemas multiagentes. Um protótipo que realiza a função de autocura de redes virtuais foi implementado a partir dessa arquitetura. Novos algoritmos e mecanismos foram desenvolvidos para melhorar a eficiência do protótipo. Foi realizado, também, um estudo de caso sobre a gerência de redes virtuais que leva em consideração os requisitos das aplicações que estão sendo executadas em uma nuvem. Uma plataforma de experimentação baseada em máquinas virtuais e no OpenFlow foi criada para a execução dos experimentos. Tanto o protótipo quanto a plataforma de experimentação integram ferramentas atuais criando uma única solução para a gerência de redes virtuais. Os resultados apresentados contribuem para aproximar a virtualização de redes e a gerência autonômica da realidade.

Agradecimentos

Agradeço aos integrantes do Instituto de Computação da Unicamp, que conta com excelentes docentes, funcionários e alunos, deixando um agradecimento especial ao pessoal da secretaria de pós-graduação, com quem tive um contato mais estreito e pude apreciar a eficiência no trabalho e o apoio aos alunos.

Também gostaria de agradecer aos órgãos de fomento CAPES e FINEP pelas bolsas de estudo concedidas durante o mestrado e também à CNPq e à FAPESP, que juntamente com a CAPES contribuem com o financiamento do Instituto de Computação da Unicamp.

Agradeço ao Prof. Dr. Nelson Luis Saldanha da Fonseca, que tive a honra de ser aluno e trabalhar junto, e também a todos os colegas do Laboratório de Redes de Computadores, que sempre mantiveram um ambiente de trabalho produtivo, colaborativo e descontraído.

Eu também agradeço aos integrantes do projeto Horizon e à FINEP, que financiou esse projeto. Agradecimentos especiais aos Profs. Drs. Otto Carlos Muniz Bandeira Duarte da UFRJ, coordenador do projeto no Brasil, e Guy Pujolle da UPMC, coordenador do projeto na França.

E já não bastasse tê-lo agradecido três vezes, como respeitável professor e pesquisador do Instituto de Computação da Unicamp, como coordenador do Laboratório de Redes de Computadores e como professor membro do projeto Horizon, eu agradeço ao Prof. Dr. Edmundo Roberto Mauro Madeira que me orientou nesta dissertação, nos projetos, nos artigos, nas apresentações, nas disciplinas, na dúvidas interiores, enfim, em todas as muitas vezes que eu recorri.

Por fim, agradeço à minha família e aos meus amigos que sempre me suportaram e me motivaram ao longo desse período de estudo e trabalho por qual passei com muito prazer.

Sumário

Abstract **vii**

Resumo **viii**

Agradecimentos **ix**

1 Introdução **1**
1.1 Motivação 2
1.2 Objetivos e contribuições 4
1.3 Organização 6

2 Fundamentos básicos **8**
2.1 Internet do futuro 8
2.2 Redes virtuais 12
2.2.1 Modelos para a virtualização da Internet 13
2.2.2 Mapeamento de redes virtuais 14
2.2.3 Tecnologias de virtualização de redes 16
2.3 Gerência autonômica 20

3 Trabalhos relacionados **25**
3.1 Arquiteturas de sistemas e gerentes autonômicos 25
3.2 Modelos de informação 26
3.3 Algoritmos e protocolos 28

4 Gerência autonômica de redes virtuais **29**
4.1 Visão geral do sistema 30
4.2 Arquitetura multiagente 32
4.2.1 Comportamentos 33
4.2.2 Base de conhecimento 34
4.2.3 Orquestrador dinâmico 36

4.2.4 Interface de monitoramento e controle 36

5 Autocura de redes virtuais 38
5.1 Tecnologias utilizadas 39
5.2 Modelo de informação 41
5.3 Laço autonômico 41
5.4 Algoritmos de seleção de nós físicos suplentes 45
5.5 Mecanismo de recuperação de nós virtuais 47
5.6 Interface gráfica 49

6 Avaliação do sistema 51
6.1 Plataforma de experimentação 51
6.2 Adaptações em redes virtuais 53
6.3 Autocura de redes virtuais 56
6.3.1 Experimentos em ambiente real 56
6.3.2 Experimentos em ambiente simulado 60

7 Estudo de caso: gerência de redes virtuais em nuvens 64
7.1 Arquitetura de gerência de integrada 66
7.2 Plataforma de experimentação 67
7.3 Resultados experimentais 68
7.3.1 Impacto das redes virtuais na execução de um *workflow* 69
7.3.2 Redes virtuais concorrentes 70
7.3.3 Adaptação das redes virtuais 73
7.4 Análise do estudo de caso 77

8 Conclusão 78

Lista de Tabelas

2.1 Plataformas de virtualização de máquina. 17

4.1 Primitivas da interface do agente com o elemento gerenciado. 37

7.1 Configurações dos servidores da nuvem. 68
7.2 Cenários utilizados para avaliar o desempenho de redes virtuais concorrentes. 71

Lista de Figuras

1.1 Arquitetura geral do projeto Horizon. 5

2.1 Representação dos modelos de virtualização para a Internet do futuro. . . . 13
2.2 Laço de controle autonômico (adaptado do trabalho de Dobson et al. [2006]). 22

4.1 Arquitetura do sistema de gerência autonômica de redes virtuais. 30
4.2 Arquitetura dos agentes do sistema. 33
4.3 Modelo de informação do sistema. 35

5.1 Tecnologias empregadas para a implementação das redes virtuais e do protótipo do sistema. 40
5.2 Detalhes do modelo de informação utilizado para autocura de redes virtuais na linguagem XML. 42
5.3 Arquivo de políticas que controla a frequência do laço de controle autonômico. 44
5.4 Interface gráfica do sistema de gerência autonômica de redes virtuais. . . . 50

6.1 Exemplo da configuração de redes virtuais na plataforma de experimentação. 52
6.2 Montagem do experimento de adaptações em redes virtuais. 53
6.3 Resultados da avaliação do protótipo no estudo de caso sobre adaptação de enlaces virtuais. 55
6.4 Resultados dos experimentos sobre impacto da migração de nós da rede virtual. 57
6.5 Cenário utilizado na avaliação da função de autocura de redes virtuais em ambiente real. 58
6.6 Desempenho da função de autocura de redes virtuais em ambiente real. . . 59
6.7 Cenário utilizado na avaliação da função de autocura em ambiente simulado. 62
6.8 Avaliação da eficiência da alocação de recursos da função de autocura de redes virtuais em ambiente simulado. 63

7.1 Gerência em dois níveis de *workflow* e rede virtual para computação em nuvem. 65

7.2 Arquitetura do sistema de gerência de *workflow*. 67
7.3 Montagem da plataforma de experimentação para o estudo de caso. 68
7.4 *Workflow* que aplica o filtro de mediana em uma imagem. 69
7.5 Avaliação de desempenho da aplicação utilizando redes virtuais em comparação com uma rede física. 70
7.6 Avaliação de desempenho das redes virtuais concorrendo pelos enlaces físicos de 1Gbps e 100Mbps. 72
7.7 *Workflow* que aplica o filtro de mediana em paralelo. 73
7.8 Avaliação de desempenho da aplicação em função de adaptações das redes virtuais durante a execução do *workflow*. 74
7.9 *Workflow* que aplica uma sequência de filtros em paralelo. 75
7.10 Avaliação de desempenho das transferências de dados dos *workflows* concorrentes em função das adaptações nas redes virtuais. 76

Capítulo 1

Introdução

Segundo dados recentes do Internet World Stats[1], a Internet conta com 2,4 bilhões de usuários, o que equivale a 34,3% da população mundial. Esses usuários estão divididos em 26.702 domínios administrativos, de acordo com o mapeamento topológico realizado pela CAIDA[2] (*Cooperative Association for Internet Data Analysis*). Desde a criação da Internet, no início dos anos 70, houve um grande aumento no número de redes interconectadas e no volume de tráfego. Esse crescimento foi possível graças aos seus princípios arquiteturais de simplicidade do núcleo, controle distribuído e comutação de pacotes. Os serviços disponíveis para a Internet também cresceram e se diversificaram nesses 40 anos. Vieram se somar aos serviços tradicionais - como o correio eletrônico e a Web - as aplicações multimídia, as aplicações de *e-science*, os sistemas de tempo-real, os sistemas distribuídos, entre outros.

Há um desenvolvimento constante na Internet com o surgimento de novas demandas e tecnologias. O seu núcleo simples e estável permite integrar diversas tecnologias de enlaces em uma única rede de propósito geral capaz de atender a vários tipos de aplicações. Se por um lado o que está abaixo e acima da camada de rede pode evoluir rapidamente, os protocolos de rede não são facilmente modificados [Roberts, 2009]. Isso porque eles são padrões em todos os dispositivos interligados na Internet para formar uma camada de cobertura que homogeniza as comunicações. Essa camada realiza o serviço de transporte não confiável de pacotes fim-a-fim, chamado de melhor esforço. Garantias de entrega, controle de congestionamento, confidencialidade dos dados e outros requisitos das comunicações devem ser realizados pelas outras camadas.

Entretanto, algumas demandas não podem ser totalmente atendidas se não tiverem o suporte do núcleo. Proteção contra ataques DDoS (*Distributed Denial of Service*), garantias de QoS (*Quality of Service*), engenharia de tráfego e mobilidade são alguns exemplos.

[1] http://www.internetworldstats.com
[2] http://www.caida.org

Além disso, o sucesso da Internet a levou a atingir seus limites de escalabilidade. Para atender às novas demandas e resolver os problemas atuais é preciso alterar a funcionalidade do seu núcleo. Existem duas abordagens para dar continuidade ao desenvolvimento da Internet [Rexford e Dovrolis, 2010]. A evolucionária defende que sua arquitetura deve ser mantida e que as alterações devem ser realizadas de forma a manter a compatibilidade com a versão atual. Incrementos arquiteturais e *middleboxes* são utilizados para evoluir a arquitetura da Internet e são muitas vezes chamados de "remendos" na literatura. A abordagem disruptiva, também conhecida como *clean-slate*, defende a criação de uma arquitetura nova para a Internet, levando em consideração a arquitetura atual e tudo que foi aprendido com ela, mas sem a necessidade de manter a compatibilidade. Essa abordagem permite a criação de soluções mais ideais, porém, difíceis de serem implantadas.

Uma forma de conciliar as duas abordagens e obter o melhor de ambas é habilitar o pluralismo de arquiteturas na Internet. Com isso, novas arquiteturas disruptivas poderiam coexistir com os protocolos atuais utilizando a mesma infraestrutura física. A separação lógica de uma rede pode ser feita através de técnicas de virtualização [Anderson et al., 2005; Turner e Taylor, 2005]. Cada rede virtual opera de maneira isolada e tem seu próprio plano de controle. Pensando nessas redes virtuais em escala global pode-se imaginá-las como Internets Paralelas, que agregam o tráfego de uma mesma classe de serviço.

Também é possível pensar em um cenário em que uma rede virtual pertence a uma única organização. Esse cliente, chamado de provedor de serviço, pode estar interessado em interligar seus servidores e seus clientes através de uma rede virtual. Nesse caso, os provedores de infraestrutura fornecem os servidores virtuais de processamento, memória e disco, e também os recursos de rede, como roteadores e enlaces virtuais [Truong Huu et al., 2011]. A rede virtual utilizada por um provedor de serviços dever ser isolada e oferecer garantias de QoS.

A virtualização gera um aumento da complexidade das redes de computadores, que já são sistemas bastante complexos. Por outro lado, ela contribui para a sua gerência, criando uma melhor separação das atividades gerenciais da infraestrutura e dos serviços oferecidos pelas redes virtuais. Além disso, cada rede virtual pode ter uma gerência própria, tanto em relação à alocação dos seus recursos, quanto dos seus serviços. Isso permite uma especialização melhor dos algoritmos e protocolos utilizados para o controle das redes virtuais. Elas também contribuem com o aumento da flexibilidade, permitindo o desenvolvimento de uma gerência mais dinâmica.

1.1 Motivação

Atualmente, cada domínio administrativo na Internet é responsável pela gerência da sua infraestrutura e dos seus serviços. A gerência dentro do domínio (intradomínio) é bas-

tante centralizada e dependente de intervenção humana. Ela é baseada principalmente no modelo FCAPS (*Fault, Configuration, Accounting, Performance, Security*) e nos protocolos SNMP (*Simple Network Management Protocol*) e CMIP (*Common Management Information Protocol*).

Não apenas no mundo da Internet, mas de TI como um todo, acredita-se hoje que a evolução dos sistemas computacionais produz complexidade como inevitável insumo e ela tem crescido além da capacidade humana de gerenciá-la. Os custos com gerência dos sistemas são elevados e não garantem o funcionamento de sistemas que quando caem geram prejuízos de milhões de dólares por hora [Ganek e Corbi, 2003]. Enquanto a Internet era praticamente uma rede experimental, eventuais indisponibilidades eram aceitáveis. Com sua integração às redes de telecomunicação e sua utilização em sistemas críticos seus requisitos de confiabilidade tornaram-se mais firmes.

A computação autonômica [IBM, 2001] foi proposta para lidar com a complexidade através de sistemas mais fáceis de usar e gerenciar. Ela busca eliminar ou reduzir a intervenção humana na gerência de baixo-nível. Este conceito é bioinspirado pelo sistema nervoso autônomo que realiza a regulação do corpo em função das mudanças ambientais sem um controle consciente. Tarefas mais simples de configuração, otimização, cura e proteção podem ser realizadas pela gerência autonômica do próprio sistema, permitindo o desenvolvimento de uma gerência de mais alto-nível.

A gerência autonômica pode ser aplicada no problema da gerência dos recursos físicos do provedor de infraestrutura para atender às redes virtuais. Esse novo tipo de gerência visa instanciar e liberar redes virtuais rapidamente, otimizar a alocação de recursos para maximizar o lucro, negociar SLAs (*Service Level Agreements*) com clientes e realizar o controle de admissão, reparar as redes virtuais em cenários de falhas e ataques maliciosos, controlar a provisão de recursos para manter níveis de serviço especificados no SLA e prover isolamento e segurança para as redes virtuais.

Tanto as redes virtuais quanto a gerência autonômica são temas novos, que começaram a ganhar a atenção da comunidade científica há pouco tempo. Elas têm como motivação principal a superação do impasse da Internet através de uma nova arquitetura pluralista e mais fácil de gerenciar. Ainda não existem ferramentas maduras para a criação de redes virtuais ou de sistemas de gerência autonômica. Não existem também definições amplamente aceitas do que sejam as redes virtuais ou redes autonômicas. A ausência de ferramentas e definições dificulta o desenvolvimento de soluções tecnológicas para essas áreas.

Existem ainda muitos desafios para a utilização de redes virtuais com gerência autonômica na prática, mas também muitas recompensas, uma vez que essas tecnologias estão ligadas com o futuro da Internet e com a evolução dos sistemas. Novas formas de gerência precisam ser criadas para controlar os recursos das redes virtuais, que po-

dem ser alterados dinamicamente. Um recurso virtual, como um enlace de dados, por exemplo, pode ser criado, destruído, expandido, contraído, migrado, algo que não acontece atualmente nas redes de computadores, cujos recursos são estáticos. Para gerenciar esses recursos de maneira autonômica novas arquiteturas, modelos de informações, algoritmos e protocolos devem ser desenvolvidos. Também é preciso estudar o impacto dessas adaptações tanto nas comunicações quanto nas aplicações que utilizam as redes virtuais.

1.2 Objetivos e contribuições

O objetivo deste trabalho é a aplicação dos conceitos da computação autonômica para a gerência de redes virtuais. As contribuições são o desenvolvimento de uma arquitetura de gerência autonômica de redes virtuais baseada em sistemas multiagentes, a implementação de um protótipo baseado nessa arquitetura com o foco na autocura de redes virtuais e o estudo do impacto dessa gerência nas aplicações.

Na arquitetura multiagente desenvolvida neste trabalho, os agentes fazem o papel dos gerentes autonômicos e eles realizam a gerência e o controle das redes virtuais de maneira distribuída. O objetivo desse sistema multiagente é realizar as funções de gerência e controle de um provedor de infraestrutura sem intervenção humana. Para isso, os agentes utilizam uma base de conhecimento para troca e interpretação das informações do sistema. Essas informações são utilizadas pelos laços de controle autonômicos que executam as funções de gerência e controle.

Um protótipo que realiza a autocura de redes virtuais, que é uma das funções da gerência autonômica, foi implementado a partir da arquitetura proposta. O laço de controle autonômico foi especializado para realizar essa função. Foram desenvolvidos algoritmos proativos para orientar o processo de remapeamento de recursos virtuais com o objetivo de manter a eficiência da alocação de recursos e também um mecanismo de recuperação rápida de nós virtuais baseado no salvamento prévio do sua memória em disco.

Para a avaliação do protótipo, foi desenvolvida uma plataforma de experimentação em pequena escala, empregando as tecnologias atuais de virtualização de máquinas e de redes definidas por software. Com essa plataforma é possível criar redes virtuais de controle distribuído com seus planos de dados isolados. Essas redes virtuais são agnósticas em relação à rede de substrato e às demais redes virtuais e, portanto, elas podem utilizar os protocolos de rede atuais sem modificações.

O desenvolvimento do protótipo sobre a plataforma de experimentação mostra como ferramentas de finalidades diversas podem ser integradas para a criação das redes virtuais e da sua gerência autonômica. Na implementação foi utilizada a plataforma de virtualização KVM para a criação das máquinas virtuais; o comutador virtual Open vSwitch, que

controla os fluxos entre os nós virtuais para criar os enlaces virtuais; o *framework* Ginkgo para o desenvolvimento do protótipo; e a biblioteca Libvirt que provê uma interface de controle com diversos monitores de máquinas virtuais e torna o sistema independente da tecnologia de virtualização.

Além de possibilitar o pluralismo de arquiteturas e facilitar a aplicação de QoS, a virtualização contribui com a flexibilidade das redes de computadores. Os recursos das redes virtuais são dissociados da infraestrutura física, e com isso, é possível realizar uma gerência mais dinâmica desses recursos. O estudo de caso realizado sobre o impacto das redes virtuais em aplicações distribuídas têm como objetivo mostrar como adaptações no mapeamento das redes virtuais sobre a rede de subtrato durante a execução das aplicações podem melhorar o seu desempenho. Isso contribui para o desenvolvimento futuro de uma gerência de redes virtuais cientes de serviços. A plataforma de experimentação citada anteriormente foi utilizada também no estudo de caso.

Este trabalho foi realizado no contexto do projeto Horizon[3], uma iniciativa franco-brasileira para o desenvolvimento de uma nova arquitetura para a Internet do futuro baseada em virtualização e inteligência. A Figura 1.1 apresenta a arquitetura inicial do projeto Horizon. O plano de virtualização habilita o pluralismo de arquiteturas no núcleo da Internet. O plano de pilotagem é responsável por coletar todas as informações de elementos de rede heterogêneos e alimentar os algoritmos de controle e gerência. Através dessas informações também podem ser criados algoritmos de autogerência baseados em objetivos de alto-nível no plano de conhecimento.

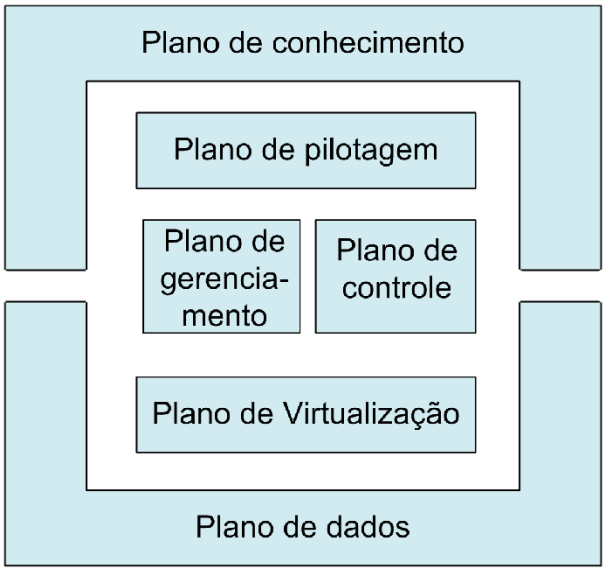


Figura 1.1: Arquitetura geral do projeto Horizon.

[3]http://www.gta.ufrj.br/horizon

1.3 Organização

Este trabalho está organizado da seguinte maneira:

- O Capítulo 2 apresenta os tópicos que fundamentam este trabalho. Ele inicia com uma discussão sobre a importância do desenvolvimento da arquitetura atual da Internet e as principais abordagens e iniciativas para isso. Em seguida são levantados os aspectos teóricos e tecnológicos para o desenvolvimento de redes virtuais que atendam os requisitos da Internet do futuro. Por fim a gerência autonômica é apresentada, detalhando o laço de controle autonômico, as funções de autogerência e as propostas da sua aplicação na Internet e nas redes de computadores em geral.

- No Capítulo 3 são elencados os trabalhos relacionados. Eles foram divididos em três seções. A primeira seção elenca os trabalhos sobre arquiteturas para a gerência autonômica de redes virtuais e dos seus gerentes autonômicos. Na segunda seção são apresentados os trabalhos sobre modelos de informação para especificação das redes virtuais. A terceira seção é dedicada aos trabalhos sobre algoritmos e protocolos desenvolvidos para uma gerência autonômica de redes virtuais.

- No Capítulo 4 é apresentada a arquitetura do sistema de gerência autonômica de redes virtuais. Os tipos de agentes do sistema são descritos para dar uma visão geral do funcionamento do sistema, porém, o restante do capítulo foca nos agentes distribuídos na rede de substrato do provedor de infraestrutura. A estrutura interna desse tipo de agente contém módulos que realizam o armazenamento distribuído das informações, os laços de controle autonômicos, a orquestração das funções de controle e gerência, e a interface com os elementos gerenciados.

- O Capítulo 5 complementa o anterior com o detalhamento da função de autocura de redes virtuais a partir da arquitetura apresentada. Nesse capítulo são apresentadas as tecnologias utilizadas para implementação do protótipo do sistema, a especialização do laço de controle autonômico para a função de autocura, os algoritmos desenvolvidos para seleção de nós físicos suplentes que visam manter a eficiência da alocação de recursos, o mecanismo de recuperação de nós virtuais criado a partir dos recursos oferecidos pelas plataformas de virtualização, e a interface gráfica do sistema para visualização das topologias e do tráfego das redes virtuais.

- No Capítulo 6 são apresentados os resultados dos experimentos realizados para avaliar a arquitetura do sistema de gerência autonômica e a função de autocura de redes virtuais. Neste capítulo são detalhados a montagem e o funcionamento da plataforma de experimentação. Os primeiros experimentos mostram como adaptações no

mapeamento das redes virtuais realizadas pelos agentes podem beneficiar as redes de computadores. Também foram realizados experimentos para avaliar o desempenho da solução desenvolvida para autocura de redes virtuais em ambiente real. Para a avaliação da eficiência do remapeamento dos recursos virtuais na alocação de recursos, o protótipo foi implementado sobre um ambiente de rede simulado.

- O Capítulo 7 contém um estudo sobre redes virtuais aplicadas em nuvens computacionais. Na nuvem foram executadas aplicações distribuídas descritas através de *workflows*. É apresentado um modelo de gerência integrada de aplicação e de rede virtual e uma arquitetura que detalha os módulos dessa integração. A avaliação realizada através de experimentos sobre a plataforma de experimentação mostrou como a flexibilidade das redes virtuais pode contribuir para a execução de aplicações distribuídas e concorrentes com diferentes níveis de prioridade.

- O Capítulo 8 conclui este trabalho, apresentado as principais contribuições e os resultados, além de elencar os trabalhos futuros.

Capítulo 2

Fundamentos básicos

Neste capítulo serão apresentados e discutidos os temas que fundamentam este trabalho. A Seção 2.1 apresenta os principais problemas da arquitetura atual da Internet e as iniciativas para a criação de novas arquiteturas. Também é discutido como o pluralismo de arquiteturas pode contribuir para conciliar as abordagens evolutiva e disruptiva no desenvolvimento da Internet. O pluralismo de arquiteturas é possível através das redes virtuais, tema da Seção 2.2. Nela são abordados os requisitos e os modelos das redes virtuais para a Internet do futuro, o problema do mapeamento, e as tecnologias atuais para virtualização de redes. Por fim, a Seção 2.3 apresenta a gerência autonômica e sua aplicação no cenário das redes de computadores.

2.1 Internet do futuro

Desde a interligação dos dois primeiros nós da ARPANET, em 1969, houve um grande crescimento de redes e serviços na Internet, fazendo com que ela se difundisse por todo o planeta. A Internet penetrou na sociedade, sendo hoje parte integrante da economia e da cultura mundial. Muitos fatores foram importantes para esse sucesso, incluindo interesses políticos e econômicos, mas os fatores tecnológicos foram fundamentais. Sua arquitetura provou ser apropriada aos seus propósitos depois de mais de 40 anos de concorrência com outras tecnologias e de validação da academia e da indústria. Novos tipos de enlaces e de aplicações puderam ser rapidamente implantadas, graças a essa arquitetura baseada em um núcleo simples e estável. Apesar de todo esse sucesso, e também em decorrência dele, existem diversos aspectos que não puderam ser previstos no início da Internet e que precisam ser tratados agora.

A arquitetura da Internet é baseada em uma rede de comutação de pacotes agnóstica em relação à tecnologia de enlace e à aplicação, capaz de encaminhar pacotes fim-a-fim sem garantias de entrega. Os princípios arquiteturais do núcleo simples, do controle distribuído

e da comutação de pacotes, geralmente são tratados como sinônimos de escalabilidade, robustez e eficiência. Porém, há limites para que isso seja verdade e atualmente na prática a Internet é menos escalável, robusta e eficiente do que as redes de comutação de circuitos, como as redes de telefonia e de transporte no *backbone* da Internet [Molinero-Fernández et al., 2003]. Em relação à escalabilidade, os roteadores IP (*Internet Protocol*) demandam muita memória para suas tabelas de rotas e em altas taxas de transmissão, consomem muita energia, pois seus circuitos são complexos. Quanto à robustez, não há uma separação clara entre o plano de dados e o de controle na Internet, o que gera muita latência na recuperação de falhas, congestionamentos e ataques. Em relação à eficiência, os provedores geralmente utilizam o superaprovisionamento de recursos para lidar com a elasticidade do tráfego e obter atrasos menores, o que torna a utilização dos recursos ineficiente.

Além disso, o surgimento de novas tecnologias de enlace e aplicações criaram demandas que não são compatíveis com a arquitetura atual. Os enlaces ópticos permitem altas taxas de transmissão e comunicação a grandes distâncias. Essa tecnologia tem sido utilizada no *backbone* da Internet para interligar redes, porém, sua expansão para as bordas da rede, incluindo as redes de acesso, é advogada atualmente. Entretanto, a comutação de pacotes não é uma boa alternativa para as redes ópticas, pois é preciso realizar a conversão do meio óptico para o eletrônico para a leitura do cabeçalho dos pacotes. Por outro lado, a comutação de circuitos é facilmente empregada em redes WDM (*Wavelength-Division Multiplexing*). Outra tecnologia que não é nova, mas tem evoluído bastante é a das redes sem fio. Com a miniaturização do hardware, os dispositivos móveis têm se tornado computacionalmente mais potentes e atualmente podem ser utilizados da mesma forma que computadores pessoais. Esses aparelhos têm limitações de energia que não foram levadas em consideração no desenvolvimento da arquitetura da Internet. A mobilidade também é um problema, pois os endereços IP sobrecarregam as funções de identificador e localizador de dispositivos.

Em relação às aplicações, a arquitetura da Internet não foi desenvolvida para sistemas críticos e de tempo-real. Ela oferece um único serviço não confiável, chamado de melhor esforço, que não dá garantias de atraso e nem mesmo da entrega dos pacotes. Novas aplicações como VoIP, videoconferência e IPTV, requerem garantias de banda, atraso e *jitter*. O princípio de um núcleo simples da Internet deixa a complexidade a cargo dos dispositivos nas extremidades. Isso significa que vários aspectos da comunicação serão resolvidos pelas camadas de transporte e aplicação. Porém, algumas funções são difíceis ou até impossíveis de resolver se não houver suporte do núcleo. QoS é uma delas e embora atualmente existam mecanismos para garantias estatísticas de serviço intradomínio, eles nem sempre são suficientes. Os sistemas distribuídos são outro tipo de aplicação que não têm total suporte da Internet, embora eles sejam comuns hoje em

dia. É complicado fazer o balanceamento de carga entre diversos servidores espalhados pelo mundo, com a sobrecarga semântica dos endereços IP. Esse problema foi apenas amenizado com o surgimento do DNS que criou uma certa separação de identificador e localizador na Internet.

Além dessas demandas atuais, algumas outras são previstas para o futuro. Hoje em dia, a Internet tem fins comerciais, muito mais do que os militares ou acadêmicos do seu passado. No futuro isso deverá ser exacerbado com o aumento da sua dependência para o comércio [Jain, 2006]. Além disso, a Internet foi criada para interligar computadores, mas acredita-se que ela deve ser utilizada para conectar pessoas, serviços e todos os tipos de objetos do mundo real. Surge assim uma grande demanda por segurança para obter confiabilidade e privacidade. Atualmente o mau comportamento de um roteador pode causar a interrupção dos serviços de uma rede. As comunicações também não são seguras pela ausência de autenticação dos atores e por falta de privacidade e integridade dos dados. O novo conceito de computação em nuvem também tem mudado a maneira de enxergar a Internet. Hoje há uma clara separação entre a rede e os serviços que rodam sobre ela. No futuro, para atingir uma gerência mais eficiente e melhores níveis de serviço, as redes devem ser conscientes dos serviços e os serviços conscientes das redes. Elas também devem ser autoconscientes e capazes de se autogerenciar [Galis et al., 2009].

As pesquisas que envolvem propostas de incrementos arquiteturais ou novas arquiteturas para essas questões recebem o rótulo "Internet do futuro". Desde a criação da Internet, vários problemas surgiram e foram sendo contornados e talvez os principais tenham sido os de escalabilidade. Para suportar a nova realidade de sucesso, o roteamento na Internet passou a ser hierárquico com a criação de domínios administrativos e do BGP (*Border Gateway Protocol*), o protocolo padrão para o roteamento interdomínio. O endereçamento deixou de ser global com a criação dos endereços privados e do NAT (*Network Address Translation*) e eles puderam ser atribuídos dinamicamente através do DHCP (*Dynamic Host Configuration Protocol*). O protocolo TCP (*Transmission Control Protocol*) foi alterado para fazer um controle de congestionamento e se autoajustar em casos de sobrecarga da rede.

Outras propostas de melhorias para vários problemas que se tornaram padrões do IETF (*Internet Engineering Task Force*) como Mobile IP, IPSec, IntServ, IP Multicast e IPv6 não foram completamente implantadas apesar de serem atrativas. Os principais motivos são que essas propostas não são interoperáveis com a arquitetura atual, não trazem ganhos imediatos aos primeiros adotantes, e geram custos na troca e reinstalação de equipamentos. O desenvolvimento da Internet através de uma abordagem evolutiva propõe a realização de alterações e incrementos na arquitetura, mantendo a compatibilidade com a atual. Esse tipo de modificação arquitetural é algumas vezes chamada de "remendo" na literatura. É óbvio que essa abordagem é importante, pois atualmente ela é a única

maneira de fazer com que a Internet evolua. Entretanto, as restrições impostas impedem a criação de propostas completas para os problemas e aumentam a complexidade da arquitetura.

Por isso, tem crescido o movimento *clean-slate*, que defende o desenvolvimento da Internet através de uma nova arquitetura disruptiva em relação à atual. Essa abordagem é interessante pois permite a elaboração de arquiteturas mais ideais para os desafios da Internet. A disciplina de redes de computadores ainda é nova e não tem uma sólida base teórica, mas ela evoluiu bastante nas últimas décadas [Rexford e Dovrolis, 2010]. O objetivo da abordagem *clean-slate* é utilizar esses novos conhecimentos, incluindo os de natureza empírica, para a criação da Internet do futuro. A grande dificuldade é validar essas propostas em um cenário real na mesma escala de tamanho e de tráfego da Internet.

A virtualização de redes é uma tecnologia interessante que pode ser utilizada para conciliar ambas as abordagens, pois ela permite que as novas arquiteturas *clean-slate* coexistam com a atual sobre a mesma infraestrutura física. Os provedores de Internet poderiam criar "fatias" para oferecer serviços diferenciados e manter o serviço estável de melhor esforço do TCP/IP. Isso poderia trazer ganhos imediatos aos primeiros adotantes, mesmo que a virtualização não fosse globalmente implantada.

Existem importantes iniciativas governamentais para o desenvolvimento de projetos *clean-slate* para a Internet do futuro e, de maneira mais geral, para NGNs (*Next-Generation Networks*) [Roberts, 2009]. Nos Estados Unidos, o NSF (*National Science Foundation*) lançou os programas FIND (*Future Internet Design*) e NetSE (*Network Science and Engeeniring*) para estimular propostas de novas arquiteturas. A Comissão Europeia alavancou pesquisas em Internet do futuro através da iniciativa FIRE (*Future Internet Research and Experimentation*). No Japão, o NICT (*National Institute of Information and Communications Technology*) lançou o projeto AKARI para o desenvolvimento de arquiteturas de NGN.

A Internet ainda é utilizada para fins acadêmicos, e diversas pesquisas em redes de computadores a utilizam como plataforma de experimentação em escala global. O *testbed* mais conhecido é o PlanetLab [Chun et al., 2003], porém ele é mais utilizado em projetos de sistemas distribuídos. Infraestruturas paralelas à Internet têm surgido com incentivos governamentais para auxiliar as pesquisas em Internet do futuro. Dentre esses novos *testbeds* há o GENI (*Global Environment for Network Innovation*) nos Estados Unidos; o Federica (*Federated E-infrastructure Dedicated to European Researchers In Computing network Architectures*) na União Europeia; e o JGN2plus no Japão e na Coreia. Espera-se que no futuro essas plataformas de experimentação sejam federadas em uma grande infraestrutura global e que usuários e provedores de serviços a utilizem para criar tráfego real de produção. Todos os *testbeds* citados utilizam os conceitos de virtualização para rodar experimentos isolados em paralelo.

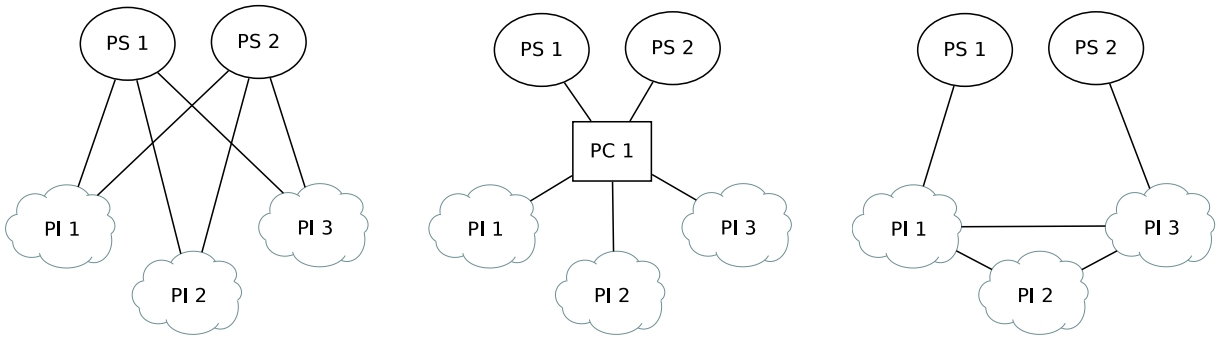
No Brasil também existem iniciativas para desenvolvimento de *testbeds* baseados em virtualização. O objetivo principal do projeto FIBRE [1] [Sallent et al., 2012] é o planejamento, a implementação e a validação de uma infraestrutura compartilhada para pesquisas em Internet do futuro, apoiando a experimentação conjunta de pesquisadores europeus e brasileiros. O projeto FITS [2] [Mattos et al., 2012] descreve uma experiência bem sucedida com uma rede de teste interuniversitária. Ele fornece uma oportunidade para aumentar a cooperação entre as universidades com o objetivo de gerenciar e configurar redes virtuais. O modelo de virtualização deste *testbed* é baseado nas ferramentas VNEXT [Pisa et al., 2011], para gerência roteadores virtuais, e OMNI [Mattos et al., 2011], para gerência de fluxos OpenFlow[3].

2.2 Redes virtuais

O objetivo da virtualização é a separação lógica de um recurso computacional em partes isoladas. Ela surgiu na década de 60 para portar sistemas legados em *mainframes* e atualmente é bastante utilizada para a consolidação de servidores, isto é, a criação de máquinas virtuais para uma utilização mais eficiente da infraestrutura física [Carissimi, 2008]. Apesar de prover uma maior flexibilidade na utilização do recurso computacional, a virtualização causa uma degradação de desempenho. Porém, diferente da emulação que realiza uma conversão de arquitetura, na virtualização, o recurso virtual é semelhante ao físico e roda diretamente sobre ele, sendo minimamente vigiado por uma camada de gerência que provê o isolamento.

Técnicas de virtualização também são utilizadas em redes de computadores [Chowdhury e Boutaba, 2009]. As VLANs (*Virtual Local Area Networks*) separam logicamente uma rede local, isolando os hospedeiros através das portas do comutador em que estão ligados, ou dos endereços (camada 2 ou 3) das suas interfaces de rede. No entanto, todas as VLANs utilizam os mesmos protocolos de comunicação. Outro tipo de virtualização em redes é o tunelamento de uma comunicação ponto-a-ponto. Nessa técnica, um pacote é encapsulado dentro de outro para transitar por uma rede insegura ou de outra arquitetura. Como exemplo de tunelamento há as VPNs (*Virtual Private Networks*), que utilizam criptografia para enviar dados de uma rede privada por uma rede pública. Por fim, as redes de sobreposição criam uma rede virtual sobre a camada de aplicação como, por exemplo, as redes P2P (*Peer-to-Peer*). Porém, elas utilizam apenas os nós presentes na borda da rede de substrato e são dependentes dos serviços oferecidos por ela. As redes virtuais que são abordadas neste trabalho constituem um caso mais geral dessas, pois elas

[1] http://www.fibre.org.br
[2] http://www.gta.ufrj.br/virtualtestbed
[3] http://www.openflow.org



(a) Cabo (b) Cabernet (c) Agave

Figura 2.1: Representação dos modelos de virtualização para a Internet do futuro.

podem ter diferentes arquiteturas e topologias, e o núcleo da rede de substrato também é virtualizado.

Conceitualmente as redes virtuais são constituídas de roteadores virtuais e enlaces virtuais. Elas rodam sobre uma infraestrutura chamada rede de substrato que deve ser capaz de isolar as redes virtuais e encaminhar os fluxos de dados dos enlaces virtuais. Os principais objetivos das redes virtuais são o pluralismo de arquiteturas e a provisão de QoS. O pluralismo de arquiteturas é obtido através da coexistência de planos de controle distintos sobre a mesma infraestrutura física. Eles podem executar de maneira distribuída em máquinas virtuais criadas nos nós da rede de substrato ou em controladores centralizados. A provisão de QoS é decorrente do isolamento das redes virtuais. As quantidades de recursos alocados aos roteadores e enlaces virtuais são mais facilmente gerenciáveis.

2.2.1 Modelos para a virtualização da Internet

No modelo atual da Internet, os ISPs (*Internet Service Providers*) são responsáveis pela gerência tanto da infraestrutura física quanto do serviço de transporte de dados. Eles também não possuem serviços diferenciados, além da taxa de transmissão no enlace de acesso do cliente. Como o controle e a gerência são decentralizados na Internet, não são oferecidas garantias de QoS fim-a-fim. É proposta a divisão de papéis dos ISPs para a especialização da gerência e dos serviços prestados através das redes virtuais. Na Figura 2.1 estão representadas as propostas de novos modelos para a Internet do futuro baseados em virtualização. Os provedores de infraestrutura (PI) são representados pelas nuvens; os provedores de serviço (PS), pelas elipses; e os provedores de conectividade (PC), pelos retângulos. As linhas representam contratos entre as partes e não suas conexões físicas.

A proposta do projeto CABO (*Concurrent Architectures are Better than One*) [Fe-

amster et al., 2007], da Figura 2.1(a), visa a reestruturação da Internet com separação de provedores de infraestrutura e de serviço. O provedor de infraestrutura é responsável pela gerência dos recursos físicos intradomínio e oferece a seus clientes "fatias" desses recursos através da virtualização. Seus clientes são os provedores de serviço que contratam provedores de infraestrutura ao longo da comunicação fim-a-fim para a implementação do seu serviço.

Seria complicado para uma única organização interessada em criar um novo serviço na Internet contratar diversos provedores de infraestrutura para criar uma rede virtual em escala global. A proposta do projeto Cabernet (*Connectivity Architecture for Better Network services*) [Zhu et al., 2008], apresentada na Figura 2.1(b), é um modelo de três camadas com a inclusão do provedor de conectividade. Ele é responsável pela contratação dos provedores de infraestrutura, a criação e a gerência de uma WAN (*Wide-Area Network*) virtual. Essa rede carrega o tráfego agregado de diversos provedores de serviço.

Uma proposta diferente das anteriores, de abordagem mais evolutiva, é a do projeto AGAVE (*A liGhtweight Approach for Viable End-to-end IP-based QoS services*) [Boucadair et al., 2009] que introduz os conceitos de NPs (*Network Planes*) e PIs (*Parallel Internets*). Essa proposta é apresentada na Figura 2.1(c). Os provedores de infraestrutura criam NPs internamente para atender determinada classe de tráfego, tendo o controle sobre eles. Os serviços com requisitos de QoS semelhantes são atendidos por este NP de uma maneira agregada. Através de interconexões horizontais com outros provedores que possuem NPs semelhantes, formam-se as PIs. Assim, o provedor de serviço precisa contratar apenas um provedor para ter um serviço diferenciado fim-a-fim.

2.2.2 Mapeamento de redes virtuais

Uma das principais questões envolvendo as redes virtuais é o seu mapeamento sobre a rede de substrato. Ela é importante pois a otimização da alocação dos recursos pode reduzir os custos para os clientes das redes virtuais e aumentar as receitas para o provedores de infraestrutura. Não há na literatura um modelo matemático consensual para esse tipo de problema, mas há alguns pontos fundamentais em relação aos algoritmos de mapeamento aplicados no mundo real. O primeiro é que há limitações de recursos físicos, tanto dos nós quanto dos enlaces. Além disso, as requisições devem ser *online*, i.e., os pedidos de alocação e desalocação de redes virtuais chegam dinamicamente no tempo, ao contrário do modelo *offline*, em que todas as requisições são conhecidas previamente. O problema do mapeamento é NP-difícil, mesmo em alguns casos simplificados, como o *offline*. Uma estratégia comum baseada em heurística é separar o problema em dois: primeiro realizar o mapeamento dos nós e em seguida dos enlaces. Apenas a segunda parte do problema, o mapeamento de enlaces virtuais com restrições de largura de banda dadas as posições dos

nós virtuais, também é NP-difícil. Os algoritmos de mapeamento propostos na literatura são geralmente centralizados e possuem uma visão global da rede [Zhu e Ammar, 2006; Lu e Turner, 2006; Yu et al., 2008; Chowdhury et al., 2009; Davy et al., 2011; Fajjari et al., 2011a; Alkmim et al., 2011]. Um algoritmo distribuído para o mapeamento de redes virtuais é apresentado por Houidi et al. [2008].

A modelagem matemática do problema é baseada em grafos ponderados. A rede de substrato é um grafo $G^S = (N^S, E^S)$, onde N^S é o conjunto de nós e E^S, o conjunto de enlaces da rede de substrato. Cada nó $n_i^S \in N^S$ possui atributos como custo, localização, capacidade de processamento e memória, assim como enlace $e_i^S \in E^S$ possui atributos como custo, largura de banda e atraso. As redes virtuais também são grafos $G^V = (N^V, E^V)$, onde N^V é o conjunto de nós e E^V, o conjunto de enlaces da rede virtual. Os nós $n_i^V \in N^V$ e enlaces $e_i^V \in E^V$ também possuem atributos referentes aos requisitos das redes virtuais. O objetivo do algoritmo de mapeamento é encontrar funções $f^N : N^V \rightarrow N^S$, e $f^E : E^V \rightarrow P^S$, onde P^S é o conjunto de caminhos acíclicos da rede de substrato, que otimize uma função de custo (minimização) ou de receita (maximização).

Essas funções podem ser modeladas com matrizes que relacionam os recursos virtuais aos recursos físicos. A matriz M^N relaciona cada nó virtual a um único nó físico e um nó físico pode estar relacionado a vários nós virtuais. Portanto, o mapeamento de nós virtuais sobre os nós físicos é do tipo um-para-muitos. A matriz M^E relaciona cada enlace virtual a um caminho da rede de substrato, que pode conter um ou mais enlaces físicos, e um enlace físico pode estar relacionados a vários enlaces virtuais que passam por ele. Portanto o mapeamento de enlaces virtuais sobre os enlaces físicos é do tipo muitos-para-muitos.

O mapeamento de redes virtuais na Internet do futuro pode ser realizado pelo provedor de infraestrutura. O cliente da rede virtual - um provedor de serviço ou de conectividade - envia o grafo da rede virtual com seus atributos, ou uma especificação das suas necessidades em um nível mais alto que passa por uma transformação até chegar a um grafo da rede virtual, e o provedor de infraestrutura realiza o mapeamento de maneira agnóstica para o cliente. Esse caso é o mais factível em ambientes comerciais. Porém, é possível imaginar também um mapeamento realizado pelo cliente. O provedor de infraestrutura disponibiliza a descrição dos recursos físicos e os clientes das redes virtuais se encarregam de mapear as redes virtuais. Em *testbeds* e redes experimentais esse modelo pode ser utilizado.

Os algoritmos de mapeamento de redes virtuais não estão dentro do escopo deste trabalho, mas eles fazem parte de um módulo interno da arquitetura de gerência autonômica de redes virtuais. Na arquitetura proposta, o mapeamento é realizado pelo provedor de infraestrutura por um agente que atua em nome do cliente da rede virtual. Cada rede virtual é gerenciada em alto nível por um desses agentes, portanto o algoritmo de mape-

amento utilizado por eles deve resolver o problema de mapear uma única rede virtual na rede de substrato, de acordo com as condições atuais desta. Nesse cenário poderia haver uma coexistência de diferentes algoritmos com estratégias heurísticas voltadas para tipos específicos de redes, com topologias e requisitos de QoS diversos.

2.2.3 Tecnologias de virtualização de redes

As redes virtuais estão no contexto das SDNs (*Software Defined Networks*), que facilitam a programabilidade das redes de computadores alavancando inovações. Nesse tipo de rede, o plano de controle é programável e ele pode ser distribuído ou centralizado. No caso distribuído, os planos de controle das redes virtuais podem executar em máquinas virtuais instanciadas nos nós da rede de substrato. Existem diversas plataformas voltadas para a criação de máquinas virtuais e alguns trabalhos que serão elencados mais a frente para a especialização delas em roteadores virtuais. Nas redes virtuais de controle centralizado, a principal tecnologia é o OpenFlow, que permite a separação lógica da rede através da classificação de fluxos. Essas tecnologias serão abordadas a seguir.

Máquinas virtuais

O modelo atual de um sistema computacional é baseado no hardware, que realiza as funções de processamento, armazenamento, E/S (Entrada e Saída), etc; no sistema operacional, que gerencia e abstrai o hardware; e nos processos de usuário. Os sistemas operacionais multiprogramados já realizam um tipo de virtualização, pois cada processo do usuário utiliza um hardware abstrato. A virtualização de máquina adiciona um nível de abstração ao modelo, permitindo a criação de máquinas virtuais isoladas, que possuem seus próprios processos de usuário. Elas podem compartilhar o mesmo sistema operacional da máquina física, ou ter o seu próprio, que será "vigiado" por um MMV (Monitor de Máquina Virtual) ou *hypervisor*.

As plataformas de virtualização que compartilham o sistema operacional têm um desempenho melhor, porém, são menos flexíveis. Elas utilizam contêineres para separar logicamente seus recursos e processos [Vaughan-Nichols, 2006]. Nas plataformas que utilizam um MMV, o sistema operacional da máquina virtual não pode executar em um modo totalmente privilegiado. Há duas formas de tratar isso e garantir o isolamento: o MMV pode interceptar as instruções das máquinas virtuais em tempo de execução, ou os seus sistemas operacionais podem ser alterados em tempo de compilação para fazer chamadas ao MMV em algumas instruções privilegiadas. A primeira abordagem é chamada de virtualização total e a segunda de paravirtualização [Barham et al., 2003], sendo que a segunda possui um desempenho melhor em relação à primeira. Algumas arquiteturas de microprocessadores atuais possuem mecanismos de suporte à virtualização total, fazendo

Tabela 2.1: Plataformas de virtualização de máquina.

Plataforma	Contêiner	Paravirtualização	Virtualização Total
LXC	×		
OpenVZ	×		
UML		×	
Xen		×	×
VMware			×
VirtualBox			×
KVM			×

com que seu desempenho se aproxime da paravirtualização.

Na Tabela 2.1 são listadas e classificadas algumas ferramentas atuais para virtualização de máquinas. O LXC[4] (*Linux Conteiners*) é uma ferramenta de virtualização baseada em contêineres. Ele está disponível no kernel do Linux a partir da versão 2.6.29. O kernel do hospedeiro é compartilhado pelas máquinas virtuais. O OpenVZ[5] também é uma solução de virtualização baseada em contêineres, entretanto, ela utiliza um kernel do Linux modificado. O LXC e o OpenVZ realizam o isolamento de recursos e processos com uma pequena sobrecarga, por isso, eles permitem a instanciação de milhares de máquinas virtuais em um único servidor. O UML[6] (*User-mode Linux*) roda máquinas virtuais no espaço do usuário. Ele utiliza sistemas operacionais modificados nas máquinas virtuais e por isso é classificado como paravirtualização. O Xen[7] é uma ferramenta que inicialmente era baseada em paravirtualização, mas atualmente ela também oferece a virtualização total para algumas arquiteturas. Em 2007 ela foi adquirida pela Citrix System Inc., que desenvolveu sua versão comercial. O VMware[8] é uma plataforma de virtualização total desenvolvida pela VMWare Inc., uma subsidiária da EMC Corp., e é a líder de mercado na atualidade. O VMware oferece muitas ferramentas para a gerência do sistema de virtualização. O VirtualBox[9] é a plataforma de virtualização da Oracle Corp. para arquiteturas x86. Ela também oferece uma interface gráfica para facilitar a gerência das máquinas virtuais. O KVM[10] (*Kernel-based Virtual Machine*) é um software de virtualização total para arquiteturas x86 com tecnologia Intel VT ou AMD-V, contido nas versões atuais do *kernel* do GNU/Linux.

A virtualização de máquina pode ser utilizada para criação de redes virtuais. Diversos

[4]http://lxc.sourceforge.net
[5]http://wiki.openvz.org
[6]http://www.usermodelinux.org
[7]http://www.xen.org
[8]http://www.vmware.com
[9]http://www.virtualbox.org
[10]http://www.linux-kvm.org

roteadores virtuais podem ser hospedados em um nó físico programável. Cada roteador virtual pode ser de uma arquitetura de rede diferente, ou seja, ter seus próprios protocolos de encaminhamento e controle. Interligando roteadores virtuais e servidores da mesma arquitetura através de enlaces virtuais criam-se as redes virtuais. Os nós físicos devem ser capazes de multiplexar e demultiplexar pacotes de e para seus roteadores virtuais. Isso pode ser realizado com o uso de interfaces virtuais e de uma comutação interna de pacotes no nó físico. Um pacote originado de uma interface virtual deve ser encaminhado para uma interface de saída de acordo com alguma regra de comutação, e um pacote que chegue na interface de entrada do nó físico tem que ser encaminhado para a interface virtual através de informações contidas no cabeçalho do pacote.

Um roteador normalmente realiza funções tanto do plano de dados quanto do plano de controle de uma rede, embora elas sejam conceitualmente separadas. O plano de dados do roteador é responsável por comutar pacotes entre as interfaces de rede através de regras de encaminhamento. O plano de controle é responsável por criar essas regras através dos algoritmos de roteamento e também por traduzir endereços, filtrar pacotes, etc. Em um roteador virtual, o plano de dados pode estar junto com o plano de controle na máquina virtual e, dessa forma, ser mais isolado e flexível, pois cada roteador virtual tem sua tabela de encaminhamento com estruturas de dados próprias, ou ele pode utilizar o plano de dados da máquina física para evitar o chaveamento de contexto no encaminhamento de cada pacote e melhorar o desempenho.

Nenhuma das plataformas da Tabela 2.1 são específicas para criação de roteadores virtuais. Embora elas provejam um bom isolamento no processamento e memória, há problemas de E/S, em especial com relação às interfaces de rede no caso das redes virtuais. Os trabalhos de Bourguiba et al. [2010], Fernandes e Duarte [2011], e Couto et al. [2011] apresentam extensões das plataformas de virtualização para prover isolamento e QoS para roteadores virtuais. A migração de roteadores virtuais também requer uma atenção especial, pois o tempo de inatividade gerada no processo pode levar a uma grande perda de pacotes. O trabalho [Wang et al., 2008] apresenta uma proposta para redução do impacto da migração de roteadores virtuais através da separação dos planos de controle e de dados.

OpenFlow

O OpenFlow é uma projeto que teve como motivação inicial a execução de experimentos em paralelo com a rede de produção de um campus universitário [McKeown et al., 2008]. Ele utiliza a tabela de fluxos, que é uma característica comum dos comutadores atuais, originalmente utilizada para implementar serviços de filtragem de pacotes e tradução de endereços. No OpenFlow, as tabelas de fluxos dos comutadores da rede são manipuladas por um controlador centralizado através de um protocolo aberto. Para que um comutador

funcione em um ambiente OpenFlow ele deve possuir a tabela de fluxos e uma interface de comunicação com o controlador baseada nesse protocolo. Essa interface deve implementar também o protocolo SSL (*Secure Socket Layer*) por motivos de segurança.

Um fluxo é definido por um conjunto de campos do cabeçalho dos pacotes. Esses campos podem ser arbitrários, no caso de um comutador OpenFlow dedicado, ou baseados nos cabeçalhos dos protocolos da Internet (Ethernet, VLAN, IP, UDP, TCP), no caso de um comutador não dedicado. Na tabela de fluxos são armazenadas também as ações sobre os pacotes dos fluxos e dados para a geração de estatísticas. Essas ações podem ser o encaminhamento do pacote para uma ou mais portas de saída, para o controlador, para a pilha de protocolos padrão do comutador, ou o seu descarte. Pacotes de fluxos desconhecidos são encaminhados ao controlador para que este defina as ações para esses novos fluxos nos comutadores. A granularidade de um fluxo pode ser desde um tipo de protocolo definido até uma comunicação ponto a ponto entre duas interfaces.

No OpenFlow, os planos de dados e de controle têm uma separação bem definida. O plano de dados é executado diretamente pelos comutadores através da tabela de fluxos, enquanto que o plano de controle é centralizado no controlador de fluxos. Em geral o controlador recebe apenas o primeiro pacote de um fluxo e, dessa forma, ele não se torna um gargalo. Caso o controlador tenha que agir sobre todos os pacotes de um fluxo pode haver problemas de escalabilidade. Como o controlador tem um papel fundamental na rede, a comunicação entre ele e os comutadores deve ser feita de maneira segura.

O controlador pode utilizar um software que é compreendido como um sistema operacional de uma rede OpenFlow. Além de implementar o protocolo de comunicação ele fornece uma interface amigável para a criação de aplicações de rede. O NOX [Gude et al., 2008] é o sistema operacional de rede OpenFlow mais conhecido, e é baseado em C++ e Python. Outros exemplos são o BEACON[11], controlador multiplataforma na linguagem Java, e o DIFANE [Yu et al., 2010], que permite a criação de um plano de controle distribuído, com mais de um controlador

Para criar diversos planos de controle na rede, é possível utilizar diversos controladores OpenFlow com a inclusão de mecanismos para isolar as comunicações entre comutadores e controladores. Um exemplo simples seria utilizar VLANs para criar redes locais virtuais com um controlador OpenFlow. Outra maneira é fazer essa segmentação lógica através do FlowVisor [Sherwood et al., 2009], um *gateway* para as comunicações entre comutadores e controladores que utiliza o conceito de fluxo para encaminhar os pacotes de uma rede virtual a seu controlador e vice-versa. Dessa forma, os controladores das diversas redes virtuais podem ser implementados em diferentes máquinas físicas ou virtuais.

Uma das grandes vantagens da utilização do OpenFlow para a criação de redes virtuais é a facilidade do processo de migração, através de alterações nas tabelas de fluxos dos

[11] http://www.beaconcontroller.net

comutadores [Fernandes et al., 2011]. Porém, o OpenFlow é uma solução menos flexível em relação às redes virtuais baseadas em máquinas virtuais. Se forem utilizados comutadores OpenFlow não dedicados, os formatos de pacotes serão restritos aos da suíte de protocolos da Internet. Além disso, as ações executadas por um comutador são limitadas e caso haja necessidade de outro tipo de processamento os pacotes precisam ser enviados ao controlador, o que pode ocasionar em problemas de escalabilidade.

No presente trabalho são utilizadas redes virtuais de controle distribuído, baseadas em máquinas virtuais. Os planos de dados das redes virtuais também são implementados nos roteadores virtuais, criando uma maior flexibilidade de configuração. A rede de substrato é responsável por encaminhar os pacotes dos enlaces virtuais, que são implementados através de fluxos OpenFlow em comutadores virtuais Open vSwitch [Pfaff et al., 2009]. Esse modelo de plano de dados em camadas é baseado no trabalho [Zhu et al., 2008]. Ele contribui para uma separação melhor da gerência entre as redes virtuais e a rede de substrato. Há um custo de desempenho envolvido no processo de encaminhamento de pacotes com máquinas virtuais, porém, roteadores virtuais são necessários apenas quando houver alguma ação na rede virtual que não puder ser executada diretamente pelos comutadores virtuais contidos nos nós da rede de substrato.

2.3 Gerência autonômica

Lidar com a complexidade é um dos temas principais da ciência da computação. Graças a técnicas para controlá-la foi possível a criação de sistemas complexos: de grande porte, heterogêneos, e de aplicação crítica. A gerência desses sistemas é uma atividade importante que visa manter seu funcionamento com um desempenho próximo ao ótimo. Ela é realizada através de um laço de monitoramento, planejamento e execução de ações corretivas. Embora as atividades de monitoramento e execução possam ser mais facilmente automatizadas, as tarefas de planejamento são mais difíceis pois requerem um certo grau de cognição. A complexidade dos sistemas vem crescendo e a atividade de gerência tem se tornado cada vez mais custosa. Por outro lado, as atividades econômicas atualmente são dependentes dos sistemas computacionais, e indisponibilidades destes geram grandes prejuízos financeiros [Ganek e Corbi, 2003].

A gerência através de intervenção humana é cara, ineficiente e defectiva. É cara pois requer profissionais qualificados com alto grau de especialização. Ineficiente pois uma equipe precisa de um tempo considerável para analisar uma situação e decidir suas ações. E defectiva pois humanos tendem a errar e inserir falhas no sistema. Por outro lado, a mente humana é criativa e capaz de aprender e esses são os fatores pelos quais a gerência é ainda bastante dependente de intervenção humana. A atividade de gerência de sistemas complexos requer uma grande quantidade de conhecimento especializado e a capacidade

de lidar com situações imprevistas.

Para que os sistemas sejam capazes de realizar tarefas de autogerência, novas técnicas devem ser empregadas para automatizar o planejamento estratégico e habilitar o aprendizado de máquina. Sistemas capazes de se autogerenciar, isto é, autoconscientes a ponto de se autoconfigurarem buscando otimizar suas funções, se autocurarem em eventos de falhas, se autoprotegerem contra ataques, e capazes de interagirem com outros sistemas através de interconexões simples e "sem costuras" para se auto-organizarem, são mais complexos do que os sistemas atuais. O desafio portanto é lidar com a complexidade para permitir o desenvolvimento desses sistemas.

A computação autonômica [IBM, 2001] é bioinspirada no SNA (Sistema Nervoso Autônomo), responsável pela regulação do corpo de acordo com mudanças ambientais sem a necessidade de um controle consciente. Seu objetivo é reduzir a intervenção humana na gerência permitindo que tarefas mais simples da gerência sejam realizadas pelo próprio sistema, liberando os administradores para tarefas mais complexas do negócio. A computação autonômica é baseada em um laço de controle fechado, com as tarefas de monitoramento, análise, planejamento e execução. Esse laço é alimentado e alimenta um repositório chamado de base de conhecimento. Na arquitetura de Kephart e Chess [2003], o laço de controle é executado por gerentes autonômicos distribuídos. Cada gerente é responsável por uma parte do sistema que é o seu elemento gerenciado. O laço de controle autonômico e algumas técnicas que podem ser aplicadas em cada etapa são apresentados na Figura 2.2.

Para que os sistemas sejam capazes de se autogerenciar, a computação autonômica defende que eles devem possuir algumas propriedades, também chamadas de funções auto*. Essas funções são semelhantes às do modelo FCAPS, que trata da gerência de falhas, configuração, contabilidade, desempenho e segurança. As funções da gerência autonômica são autoconfiguração, auto-otimização, autocura e autoproteção e são descritas abaixo:

Autoconfiguração: Dado o que deve ser feito através de políticas de alto nível, esta função sabe como realizar.

Auto-otimização: O sistema continuamente busca oportunidades para melhorar seu desempenho e sua eficiência.

Autocura: Detecta falhas e realiza o diagnóstico automaticamente para realizar as devidas ações corretivas.

Autoproteção: O sistema autoajusta seus níveis de segurança e busca prevenir-se a ataques maliciosos e falhas.

As redes autonômicas são baseadas nos princípios da computação autonômica voltados para a gerência de redes. Não há dúvidas que as redes de computadores sejam sistemas

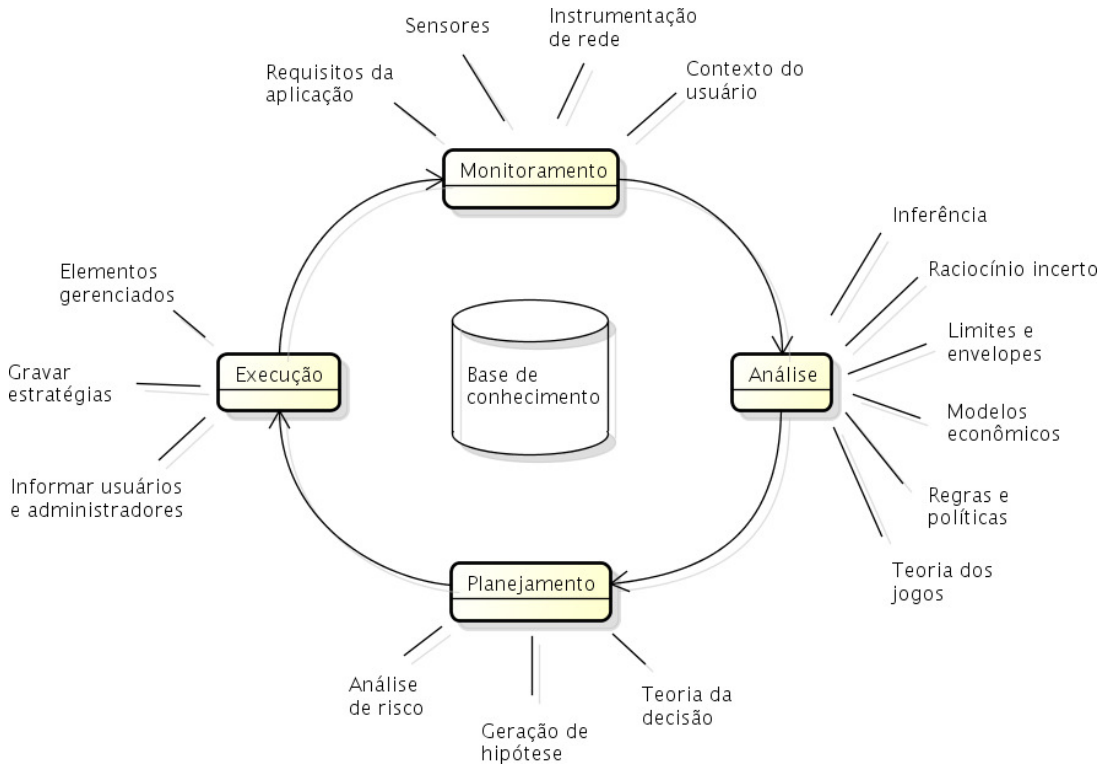


Figura 2.2: Laço de controle autonômico (adaptado do trabalho de Dobson et al. [2006]).

altamente complexos devido à quantidade de dispositivos interconectados e à diversidade de tecnologias e aplicações. Atualmente, os administradores têm que se preocupar não apenas com o que deve ser feito, mas também em como fazer, pois a gerência é realizada em baixo nível. Na Internet, foi proposta a inclusão de um novo plano no modelo de referência, chamado plano de conhecimento, para permitir o desenvolvimento de uma gerência de alto nível. Esse plano conta com ferramentas de IA e dos sistemas cognitivos para realizar tarefas de autogerência [Clark et al., 2003].

Pujolle [2008] também define um novo plano para o modelo de referência, o plano de pilotagem. Esse plano é composto por dois subplanos: o plano de conhecimento e o plano de orquestração. O plano de conhecimento é composto por uma base de dados distribuída na rede e pelos mecanismos de disseminação das informações. Os gerentes autonômicos têm acesso a essa base e são responsáveis por alimentá-la com as suas informações locais geradas no monitoramento de seus elementos gerenciados. O plano de orquestração utiliza as informações da base de conhecimento para coordenar os algoritmos de gerência e controle da rede. Ele faz isso para compatibilizar os diferentes objetivos da gerência e federar redes. Como esses dois planos são dificilmente dissociados eles podem ser tratados como um único, o plano de pilotagem. A inclusão desses novos planos no modelo de referência

da Internet contribui para o desenvolvimento da sua gerência autonômica.

Existem outras propostas para a autogerência de redes de computadores, como as redes cognitivas [Mahmoud, 2007]. Esse conceito é uma generalização dos rádios cognitivos, que têm por objetivo o compartilhamento eficiente do espectro de radiofrequência. As redes cognitivas utilizam o conceito de *cross-layer* para coletar informações de diversas camadas e fazer o controle integrado visando uma otimização global da rede. Outra área importante para a autogerência de redes de computadores é a de redes de sensores sem fio [Braga et al., 2006]. Uma de suas aplicações é o monitoramento de ambientes inóspitos de difícil acesso, como vulcões ou campos de batalha. Portanto, desde sua instalação, que pode ser realizada lançando os nós sensores na área através de aviões ou helicópteros, eles devem realizar por si próprios todas as funções de controle e gerência da rede. Os algoritmos e protocolos desenvolvidos para as redes de sensores podem servir de inspiração para a autogerência para outros tipos de redes de computadores.

Uma arquitetura de redes autonômicas é a FOCALE [Strassner et al., 2006]. Nela, o laço de controle autonômico é definido em tempo de execução, baseado no contexto e nas políticas de alto nível. A arquitetura também define o uso de uma camada MBTL (*Model-Based Translation Layer*) para possibilitar a utilização de equipamentos legados no sistema de gerência. Ela utiliza técnicas de aprendizado e cognição para comparar, através de em modelos e ontologias, se o comportamento atual é o mais adequado ou se ele deveria ser substituído. A arquitetura FOCALE foi aplicada como estudo de caso nos projetos *Beyond 3G Networks* e *Motorola's Seamless Mobility* [Strassner et al., 2007].

Ginkgo

Nossa arquitetura foi desenvolvida em cima do Ginkgo [Ginkgo Networks, 2008], uma plataforma de desenvolvimento de agentes para redes autonômicas. Esse *framework* permite a criação de agentes leves e portáveis, o que facilita a sua instalação em ambientes de rede heterogêneos: roteadores, comutadores, servidores, que utilizam redes cabeadas ou sem fio. Ele possui os blocos de construção básicos para implementação dos módulos dos agentes: os comportamentos, a base de conhecimento e o orquestrador dinâmico.

As ações dos agentes são executadas pelos seus comportamentos. O Ginkgo define uma interface padrão para seus comportamentos, que dispara ações dos agentes na sua instanciação (*onInstall*), no início da sua execução (*onStart*), no seu encerramento (*onStop*), e também a cada passo (*onStep*). Os comportamentos são executados ciclicamente durante o seu tempo de vida e suas frequências podem ser definidas dinamicamente. Os comportamentos podem estar em três estados: *NOT-STARTED*, *STOPPED* e *RUNNING*. A transição entre esses estados é realizada pelos agentes através de seus próprios comportamentos ou pelo seu orquestrador dinâmico.

Os agentes Ginkgo armazenam suas informações estruturadas na sua base de conheci-

mento. A informações são organizadas em classes, que agrupam indivíduos semelhantes. Os indivíduos são blocos de informação que contêm um conjunto de propriedades e seus valores. As classes e as propriedades de seus indivíduos são predefinidas na especificação do agente. Para isso é possível utilizar um arquivo XML, que é carregado e interpretado pelo agente para definição do modelo de informação da sua base de conhecimento.

O ciclo de vida do agente é controlado pelo orquestrador dinâmico. Esse ciclo de vida deve ser definido no seu arquivo de políticas, que utiliza uma linguagem própria do Ginkgo. Através das políticas é possível controlar a execução dos comportamentos, com as primitivas *start* e *stop*; passar parâmetros para eles, com a primitiva *setcontrol*; definir suas frequências de execução, com a primitiva *changerate*; e definir suas ordens de prioridade, com a primitiva *changeprio*. Através das políticas também é possível manipular a base de conhecimento, resgatando valores de propriedades de indivíduos através da notação *<indivíduo>.<propriedade>* e inserindo valores em propriedades de indivíduos, com a primitiva *set*.

Capítulo 3

Trabalhos relacionados

Esse capítulo é dedicado a elencar os trabalhos da área de gerência autonômica de redes virtuais, explicitar suas contribuições, relacioná-los, e comparar com o presente trabalho. Os trabalhos relacionados estão separados em três seções. A Seção 3.1 apresenta os trabalhos de arquiteturas de sistemas de gerência autonômica de redes virtuais e arquiteturas de gerentes autonômicos. Na Seção 3.2 são mostrados os trabalhos sobre novos modelos de informação para redes virtualizadas que contribuem com a gerência autonômica. A Seção 3.3 fecha o capítulo com os algoritmos e protocolos para a autogerência da alocação de recursos físicos do provedor de infraestrutura às redes virtuais.

3.1 Arquiteturas de sistemas e gerentes autonômicos

As redes autonômicas são baseadas nos gerentes autonômicos, que são distribuídos em seus nós e agem coletivamente para dotar a rede da capacidade de se autogerenciar. Eles se localizam nas proximidades de seus elementos gerenciados, por exemplo, internamente em um roteador, o que permite que a coleta de dados e as ações de controle sejam rápidas e eficientes. Esses gerentes trocam informações de suas bases de conhecimento para uma gerência no nível de rede. Os gerentes autonômicos podem ser utilizados no contexto das redes virtuais para: instanciar e liberar as redes virtuais, negociar com agentes externos, otimizar a alocação dos recursos físicos, reparar recursos virtuais na presença de falhas, etc. Nesta seção, são elencados os trabalhos relacionados que buscam definir uma arquitetura geral do sistema de gerência autonômica de redes virtuais e a arquitetura interna dos gerentes autonômicos.

O trabalho de Fajjari et al. [2011b] apresenta a AAVP (*Autonomic Architecture for Virtual network Piloting*), uma arquitetura multiagente com o objetivo de gerenciar redes virtuais. Os gerentes possuem comportamentos executados em *threads* para ações de monitoramento, decisão e execução. No monitoramento, todas as informações de contexto

são coletadas e armazenadas em uma base de conhecimento, que é utilizada por todos os comportamentos. Informações da base de conhecimento podem ser compartilhadas na vizinhança que não se resume apenas aos nós adjacentes, mas pode incluir outros nós de interesse. Na nossa arquitetura, cada função de gerência e controle é organizada em laços de controle autonômico, diferente da AAVP. Além disso, foi desenvolvida e implementada a função de autocura, criando novos mecanismos para recuperação de nós virtuais e algoritmos de seleção de nós físicos suplentes.

Uma arquitetura para gerência de recursos em provedores de infraestrutura que utilizam máquinas virtuais é apresentada no trabalho de Rodríguez-Haro et al. [2009]. O LRM (*Local Resource Manager*) é responsável por monitorar e controlar os recursos físicos atribuídos às máquinas virtuais e prover uma interface com clientes/agentes externos que realizam a gerência de alto nível. A plataforma de virtualização foi estendida para habilitar um controle de recursos fino e autoajustável. A validação foi realizada com a implementação de um mecanismo para ajuste dinâmico de recursos de CPU baseado nos requisitos de QoS da aplicação. Diferente do LRM, o foco do nosso trabalho é a gerência de redes virtuais, porém, a arquitetura do gerente LRM e o mecanismo de ajuste dinâmico de recursos virtuais são semelhantes.

A arquitetura ASA (*Autonomic Service Architecture*) [Cheng et al., 2006] é guiada pelo conceito de que "tudo é um serviço", de serviços complexos multimídia ao encaminhamento de pacotes. A virtualização para implementação de serviços escaláveis e gerência eficiente de recursos é integrada na arquitetura. Uma camada de recursos virtuais fornece uma interface uniforme e consistente para todos os recursos físicos, que simplifica a gerência de recursos, a composição de serviços e o compartilhamento dinâmico de recursos. É apresentado também um mecanismo para reserva de banda passante de enlaces com empréstimo de recursos ociosos. A nossa arquitetura apresenta uma hierarquia de gerência semelhante à da ASA, entretanto a virtualização de redes é utilizada para habilitar o pluralismo de arquiteturas e não apenas para uniformizar os recursos gerenciados.

3.2 Modelos de informação

O plano de conhecimento foi idealizado para uma gerência de alto nível das redes. Para esse fim, novos modelos de informação como os baseados em ontologias são interessantes para a descrição do conhecimento. Isso é importante para a gerência autonômica, pois ela requer um nível de cognição maior em relação aos algoritmos tradicionais. As redes autonômicas devem ter as propriedades de autoconhecimento e de ciência do contexto para serem capazes de negociar com agentes externos, interpretar informações, tomar decisões em situações não previstas, etc. Na gerência autonômica de redes virtuais, esses modelos de informações são utilizados nos SLAs entre clientes e provedores, na especificação da

rede de substrato e das redes virtuais, e nas políticas de alto nível, que guiam a gerência. Abaixo são listados os trabalhos dedicados aos modelos de informação para a autogerência de redes virtuais.

No trabalho de Fajjari et al. [2010], é apresentado um modelo de informação para provisão de redes virtuais. Esse modelo é utilizado para a definição do SLA, que contém a descrição das partes envolvidas - provedores de infraestrutura, provedores de redes virtuais e usuários de redes virtuais -; a especificação da rede virtual e o seu SLS (*Service Level Specification*); e as políticas do contrato que envolvem garantias e penalidades. As informações contidas no SLA são utilizadas para a gerência das redes virtuais. O modelo é representado por um diagrama de classes e é implementado através da linguagem XML. O modelo não foi validado, mas alguns exemplos de uso são apresentados no trabalho. O modelo de informação desenvolvido no nosso trabalho também é organizado em classes, mas ele é mais voltado para a especificação da rede de substrato, das redes virtuais, e seus mapeamentos.

Outro modelo de informação é o apresentado no trabalho de Houidi et al. [2009]. O objetivo desse modelo é permitir a descrição dos recursos físicos e virtuais para uso do provedor de infraestrutura. Essas informações são utilizadas para formar um agrupamento hierárquico de nós com características semelhantes. Para realizar o aprovisionamento de recursos, um algoritmo de mapeamento poderia levar em consideração apenas o agrupamento cujas características são semelhantes à descrição dos nós de uma rede virtual. Foi realizada uma avaliação de desempenho dos algoritmos para descoberta de recursos candidatos que satisfazem a descrição da requisição. No presente trabalho, o modelo de informação utiliza uma separação em dois níveis: real e abstrato, que estão ligados ao escopo das informações. As informações do nível real são utilizadas pelo agente local, enquanto as informações do nível abstrato são difundidas para os demais agentes do sistema.

O trabalho de Davy et al. [2008] propõe a primeira iteração de um sistema de gerência de redes virtuais baseada em políticas, onde estas são utilizadas para definir quando e onde roteadores virtuais precisam ser instanciados e configurados para oferecer serviços de rede de alto nível, apresentando características autonômicas. A implementação é baseada na plataforma Xen e em regras JBoss. Os objetivos do negócio são representados em políticas e elas são usadas para guiar o comportamento da rede virtual. Essas políticas são traduzidas em políticas de nível de sistema e processadas pelo mecanismo de regras. O caso de uso apresentado realiza reconfiguração dinâmica de rede para apoiar uma solução flexível para VPNs. Na nossa arquitetura também foram utilizadas políticas para controlar em alto nível as funções de gerência e controle, porém, elas foram baseadas no Ginkgo.

3.3 Algoritmos e protocolos

A gerência autonômica de redes virtuais é um tema novo que requer o desenvolvimento de novos algoritmos e protocolos. Eles têm o objetivo de aumentar o nível de automatização das tarefas através das funções de autoconfiguração, auto-otimização, autocura e autoproteção. No desenvolvimento desses algoritmos e protocolos deve-se levar em consideração também questões de desempenho e escalabilidade. Nesta seção são apresentados os trabalhos que propõem novos algoritmos e protocolos para a gerência autonômica de redes virtuais em um provedor de infraestrutura.

No trabalho de Houidi et al. [2010], sistemas multiagentes são utilizados para manter os níveis definidos nos SLAs em eventos de falhas de recursos e degradação severa de desempenho. Os agentes formam grupos baseados na similaridade dos nós físicos que são gerenciados por eles. Esses grupos são utilizados na escolha dos nós físicos onde serão recuperados os nós virtuais que tiveram problemas. Todas as ações dos agentes são executadas após o diagnóstico da falha. No nosso trabalho, o planejamento das ações corretivas é antecipado através de cálculos e difusões de informações realizados periodicamente. Foi desenvolvido um mecanismo de recuperação de redes virtuais através de cópias da memória virtual para reduzir a latência da inicialização do sistema operacional e o tempo de convergência do algoritmo de roteamento. Também foram desenvolvidos algoritmos de seleção de nós físicos suplentes que otimizam a alocação de recursos físicos.

Uma arquitetura de gerência distribuída é apresentada por Marquezan et al. [2010], com o objetivo de auto-organizar as redes virtuais para manter uma boa utilização dos recursos físicos. O algoritmo para auto-organização utilizado é baseado no laço de controle autonômico. Ele monitora os enlaces e busca minimizar a carga de tráfego na rede através da migração de nós virtuais. A avaliação foi realizada através de simulação com um módulo desenvolvido para o Omnet++. Na nossa arquitetura, o laço de controle autonômico implementado nos agentes do sistema de gerência autonômica realiza a autocura de redes virtuais. Além disso, uma plataforma de experimentação foi desenvolvida para a avaliação das soluções propostas em ambiente real.

Na arquitetura DaVinci (*Dynamically Adaptive Virtual Networks for a Customized Internet*) [He et al., 2008], as redes virtuais são responsáveis pela gerência personalizada de cada classe de tráfego e sua separação flexível em múltiplos caminhos. Para cada enlace físico, a largura de banda compartilhada entre as redes virtuais é periodicamente recalculada através de informações locais, como os níveis de congestionamento e os objetivos dessas redes. Essa arquitetura maximiza o desempenho das classes de tráfego para cada rede virtual e na rede física como um todo. Nosso trabalho também se baseia em um controle dinâmico dos recursos das redes virtuais. Soluções como as apresentadas no trabalho de He et al. [2008] podem ser implementadas nos agentes da arquitetura proposta.

Capítulo 4

Gerência autonômica de redes virtuais

A gerência autonômica proposta neste trabalho tem o objetivo de gerenciar a alocação de recursos físicos para as redes virtuais sem intervenção humana. Os clientes do provedor de infraestrutura, que podem ser um provedor de serviço, interessado em interligar os seus servidores, ou um provedor de conectividade, que pretende criar uma rede virtual distribuída entre múltiplos domínios, negociam contratos de serviços de rede e enviam requisições com as definições básicas de suas redes virtuais. A gerência autonômica do provedor de infraestrutura deve automatizar a negociação e processar as requisições para obter definições detalhadas das redes virtuais a serem criadas. Essa gerência utiliza um algoritmo de mapeamento de redes virtuais para realizar sua incorporação inicial na rede de substrato ou negar o pedido por falta de recursos, fazendo assim um controle de admissão. Após a incorporação inicial, a gerência autonômica realiza adaptações automáticas com o objetivo de autoconfigurar, auto-otimizar, autocurar e autoproteger as redes virtuais. As ações realizadas pela gerência autonômica no período em que a rede virtual está em operação, é o foco deste trabalho.

A arquitetura utiliza a modelagem multiagente com o intuito de tornar o sistema distribuído e com um baixo acoplamento entre seus componentes. Cada nó da rede de substrato possui um agente da arquitetura que executa os algoritmos de controle e gerência e realiza a autoconfiguração dos recursos virtuais. A base de conhecimento é a parte central da arquitetura. Através dela os agentes armazenam suas informações e se comunicam. Todas as funções dos agentes são realizadas através de laços de controle autonômicos que são controlados pelo orquestrador dinâmico. Também foi definida uma interface com as primitivas essenciais para monitoramento e controle dos elementos gerenciados. Através da padronização dessa interface o sistema de gerência autonômica pode se tornar independente das tecnologias de virtualização. Essa arquitetura serve como base

para o próximo capítulo que apresentará a implementação de um protótipo que realiza a função de autocura de redes virtuais.

4.1 Visão geral do sistema

A arquitetura proposta para o sistema de gerência autonômica de redes virtuais é apresentada na Figura 4.1. Nessa arquitetura, os agentes são organizados hierarquicamente em 2 níveis. No nível inferior, os agentes realizam a gerência dos elementos de rede. No nível superior, os agentes são responsáveis pela gerência das redes virtuais de maneira centralizada. Há também um agente que serve de ponto único de contato para novos clientes e novos agentes do provedor de infraestrutura. Os tipos de agentes e suas funções são:

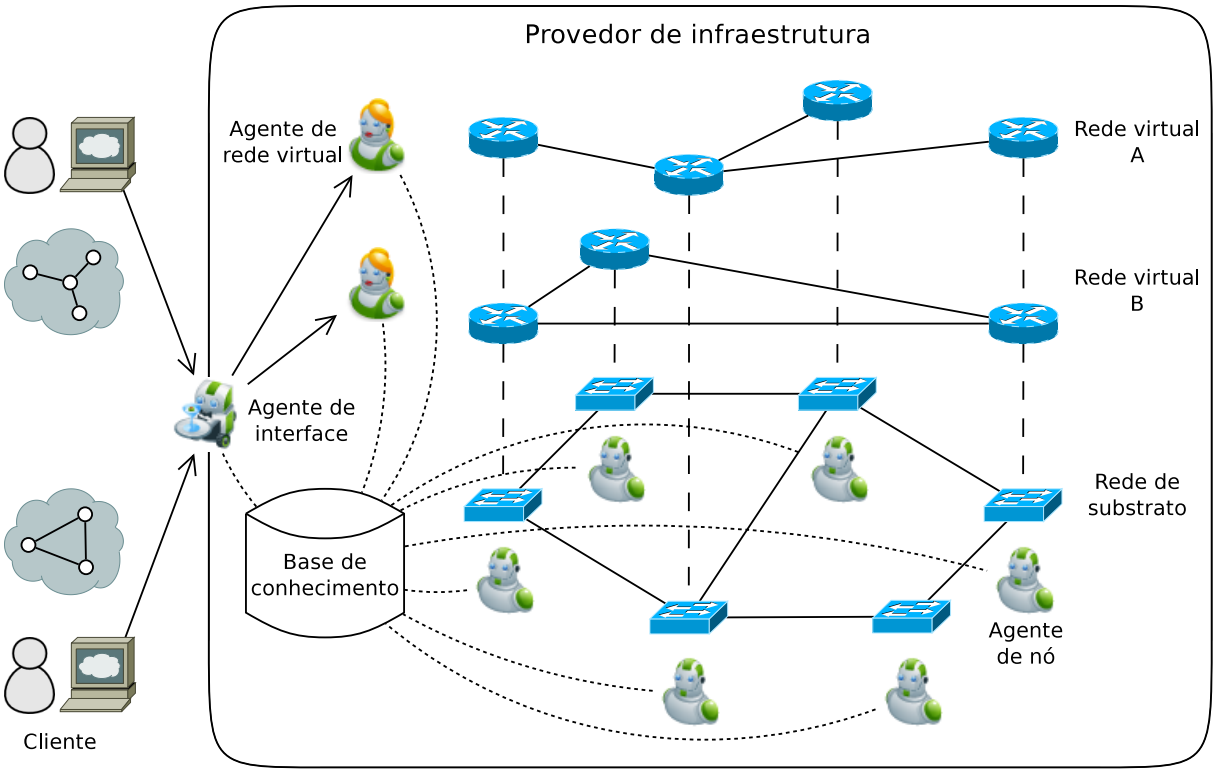


Figura 4.1: Arquitetura do sistema de gerência autonômica de redes virtuais.

Agente de interface: Há apenas um agente de interface no sistema de gerência autonômica. Ele é o ponto único de contato para os clientes do provedor de infraestrutura através de uma interface Web, e para os agentes do sistema através da base de conhecimento. Para cada requisição de cliente, o agente de interface cria um agente

de rede virtual. Novos agentes do sistema registram sua localização e recuperam a localização dos demais agentes através do agente de interface.

Agente de rede virtual: Esse tipo de agente realiza a comunicação com os clientes do provedor de infraestrutura para receber requisições de redes virtuais, negociar SLA e realizar o faturamento. O agente de rede virtual atua em nome do cliente para manter os níveis de serviço acordados e minimizar os custos. Ele possui uma visão global da rede para executar o algoritmo de mapeamento de redes virtuais e também para lidar com situações em que os agentes de nó não foram capazes de resolver.

Agente de nó: O papel desses agentes é realizar a configuração dos recursos virtuais através das plataformas de virtualização. Eles também realizam adaptações locais para corrigir falhas e otimizar a alocação de recursos físicos para as redes virtuais. Os agentes de nó atuam em nome do provedor de infraestrutura, para maximizar receitas e minimizar custos, realizando ações de balanceamento de carga e de redução de consumo de energia, por exemplo.

Um exemplo de criação de uma rede virtual será utilizado para explicar o funcionamento do sistema. Um provedor de serviço ou conectividade acessa o provedor de infraestrutura através do agente de interface, que cria um novo agente de rede virtual e fornece sua localização. O agente de rede virtual negocia o SLA com o cliente e processa sua requisição para gerar a especificação de uma rede virtual. Esse agente executa o algoritmo de mapeamento de redes virtuais com as informações das condições da rede de substrato recebidas periodicamente dos agentes de nó. A definição da nova rede virtual mapeada sobre a rede de substrato é disseminada para os demais agentes através da base de conhecimento. Os agentes de nó verificam se eles devem realizar alguma tarefa de configuração, como a criação de um nó virtual ou a inserção de alguma regra na tabela de fluxos para a criação de um enlace virtual. Quando todas as ações forem realizadas, o agente de rede virtual notifica ao cliente o término da requisição.

A arquitetura do sistema de gerência autonômica de redes virtuais foi apresentada para dar uma visão geral do sistema, incluindo as interações com os clientes do provedor de infraestrutura. Entretanto, o foco deste trabalho é a alocação de recursos e as adaptações nas redes virtuais realizadas pelos agentes de nó. A negociação de SLA e o tratamento das requisições de redes virtuais realizados pelos agentes de rede virtual não serão abordados. O restante deste capítulo irá tratar das funções e das estruturas internas dos agentes de nó que doravante serão chamados apenas de agentes.

4.2 Arquitetura multiagente

A modelagem orientada a agentes é um paradigma interessante para a implementação de redes autonômicas. Os agentes realizam o papel dos gerentes autonômicos, que atuam sobre os elementos gerenciados. Agentes são entidades autônomas capazes de observar o ambiente e agir sobre ele, podendo ter algum nível de cognição. Em um sistema multiagente, eles atuam colaborativamente para atingir os objetivos da aplicação e, para isso, eles precisam trocar informações. Os agentes se comunicam através de uma linguagem padrão, o que os tornam independentes de implementação.

Sistemas multiagentes têm particularidades de sistemas distribuídos, como escalabilidade e robustez. Na arquitetura proposta cada nó da rede de substrato, que pode ser um roteador, um comutador, um servidor, etc., contém um agente do sistema de gerência autonômica. Esse agente é responsável pelo monitoramento e controle dos recursos físicos e virtuais desse nó. Essa divisão de trabalho contribui com a escalabilidade do sistema. Além disso, as falhas tornam-se mais isoladas e não comprometem todo o sistema, tornando-o mais robusto.

Cada agente é responsável pela gerência de recursos do seu nó físico e pela alocação desses recursos para os nós virtuais. Esses recursos são: processador, memória, discos, interfaces de rede, etc. Além dos recursos dos nós, os agentes gerenciam também os enlaces da rede. Na arquitetura proposta, um enlace é definido como uma comunicação direcionada entre dois nós. Exemplificando: dois nós (A e B) conectados por um cabo de comunicação possuem 2 enlaces no sistema de gerência: um de A para B, que é de responsabilidade do agente em A; e outro de B para A, que é de responsabilidade do agente em B. No caso de um domínio de difusão, cada elemento da permutação de dois nós do domínio será considerado um enlace.

Os agentes possuem acesso privilegiado no seu nó para recuperar dados dos recursos físicos e virtuais, que são armazenados na sua base de conhecimento local. Muitas atividades de controle e gerência requerem informações além das locais como, por exemplo, a descoberta da topologia da rede. Essas informações são proativamente disseminadas entre os agentes e não solicitadas sob demanda pelos algoritmos. Assim, os algoritmos de controle e gerência possuem todas as informações necessárias disponíveis localmente no momento da sua execução. Na arquitetura proposta, os agentes disseminam uma informação apenas se ela sofreu alguma atualização.

Os módulos da arquitetura dos agentes do sistema são: os comportamentos, a base de conhecimento, o orquestrador dinâmico, o repositório de políticas, e a interface de monitoramento e controle dos elementos gerenciados, como mostra a Figura 4.2.

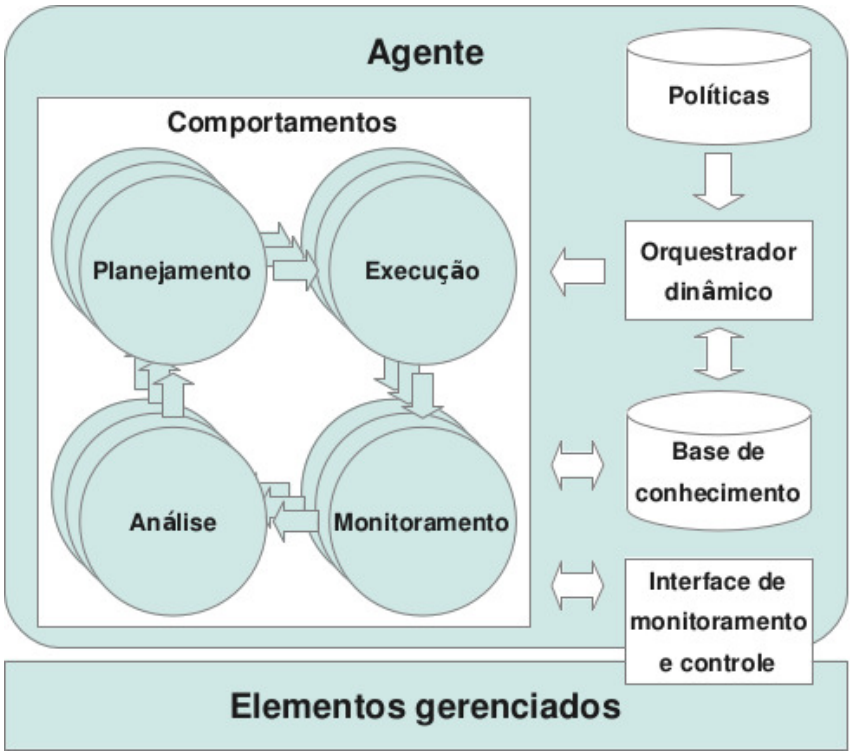


Figura 4.2: Arquitetura dos agentes do sistema.

4.2.1 Comportamentos

Os "comportamentos" são os módulos dos agentes que efetivamente realizam as ações de monitoramento, cognição e atuação dos agentes sobre seus elementos gerenciados. Cada comportamento é executado ciclicamente, e a sua frequência pode ser ajustada dinamicamente em tempo de execução de acordo com mudanças no ambiente. Por exemplo, o período do comportamento que realiza a coleta de dados pode ser reduzido para aprimorar o monitoramento quando a vazão de algum enlace atingir um certo limiar crítico.

Na arquitetura proposta, os comportamentos são organizados em laços de controle autonômicos, e realizam os passos de: monitoramento, análise, planejamento e execução. Eles utilizam a base de conhecimento local do agente para passarem informações entre as etapas do laço. Cada função de gerência, como autoconfiguração, auto-otimização, autocura e autoproteção, contém um laço de controle independente, embora alguns comportamentos possam fazer parte de mais de um laço. Um comportamento que realiza o monitoramento, por exemplo, pode coletar informações que servem para diversas funções de gerência.

Em geral, os laços de controle autonômicos operam como é descrito a seguir. O monitoramento periodicamente coleta informações dos recursos e alimenta a base de co-

nhecimento. Essas informações são utilizadas pela análise que realiza inferências e sinaliza possíveis problemas ou oportunidades. O planejamento, independente da análise, realiza proativamente algoritmos que direcionam as ações realizadas pelo agente. A execução observa a situação descrita na análise, executa as adaptações baseadas nas informações do planejamento, e atualiza a base de conhecimento para o próximo laço.

4.2.2 Base de conhecimento

A base de conhecimento é um repositório comum onde os comportamentos dos agentes armazenam e compartilham suas informações locais. É através dela que os comportamentos se comunicam para formar um laço de controle autonômico. Por ser um repositório compartilhado entre os comportamentos, a base de conhecimento utiliza um mecanismo de trava para evitar condições de corrida.

A base de conhecimento também é o repositório das informações que são trocadas entre os agentes. Portanto, as bases de conhecimento locais dos agentes formam uma base de dados distribuída no sistema. A difusão das informações da base de conhecimento é realizada automaticamente por um comportamento especial que verifica se as informações foram atualizadas e as envia para os demais agentes. As informações que chegam na interface de comunicação do agente são prontamente armazenadas na sua base de conhecimento.

Entretanto, o sistema não mantém uma consistência rigorosa da sua base de conhecimento. Isso poderia causar atrasos e uma sobrecarga muito grande com o aumento da escala da rede. Quando necessário, os comportamentos devem verificar a consistência das informações contidas na base de conhecimento. Além disso, apenas parte das informações de um agente são difundidas para os demais agentes do sistema. As informações que são necessárias apenas para o agente que as gerou são mantidas localmente na sua base de conhecimento e não são difundidas.

Na arquitetura proposta, cada recurso da rede, seja físico ou virtual, é gerenciado por apenas um agente. Da mesma forma, as informações referentes a esses recursos são de responsabilidade desse agente. Apenas ele é autorizado a atualizar essas informações e a difundi-las. Quando ocorre a falha em algum agente, ocasionada pela queda do seu nó físico, por exemplo, os recursos virtuais incorporados neste são migrados para outros nós físicos. Os agentes desses nós passam a ser responsáveis pelos recursos virtuais e recriam suas informações sobrescrevendo-as na base de conhecimento.

Para que os comportamentos possam interpretar as informações, eles utilizam um modelo de informação. Através desse modelo o sistema pode contextualizar as informações, relacioná-las a outras, e se tornar consciente do ambiente, da aplicação e de si próprio.

A Figura 4.3 apresenta o modelo de informação utilizado na arquitetura proposta.

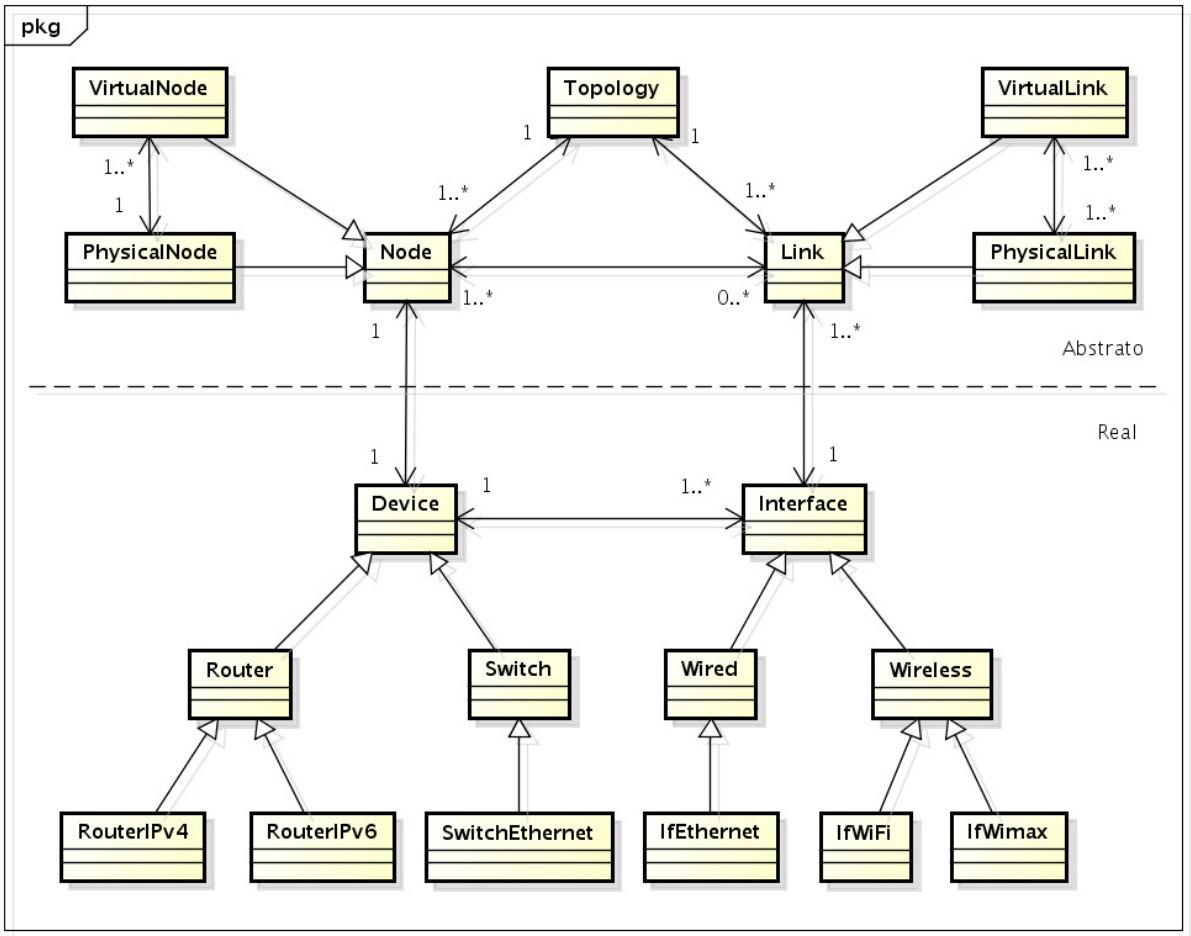


Figura 4.3: Modelo de informação do sistema.

Ele é separado em dois níveis: real e abstrato. No nível real, os recursos computacionais são descritos em detalhes. Em geral, essas informações são utilizadas pelos agentes para controle local e não são difundidas. Tanto os recursos físicos quanto os virtuais possuem representações no nível real. Por exemplo: as interfaces de rede da máquina física e das máquinas virtuais possuem indivíduos da classe *Interface*. O nível abstrato define as topologias das redes, também físicas e virtuais, e os mapeamentos dos recursos virtuais sobre os físicos. As informações desse nível são difundidas pelos agentes para que tenham uma visão global da rede. Um enlace (classe *Link*) é tratado no sistema como um recurso abstrato e é definido como uma comunicação direcionada ponto a ponto. Um domínio de difusão não é considerado um enlace, e sim cada permutação de dois elementos entre os nós do domínio. Em uma rede sem fio, por exemplo, para cada vizinho ao alcance de um determinado nó é criado um indivíduo da classe *Link*. O mapeamento das redes virtuais

também é representado no nível abstrato. Como é possível ver na Figura 4.3, os nós possuem um mapeamento um-para-muitos, enquanto os enlaces possuem mapeamentos muitos-para-muitos, pois os enlaces virtuais são mapeados em caminhos da rede física.

4.2.3 Orquestrador dinâmico

A ideia por trás do orquestrador dinâmico é um controle de alto nível sobre os planos de gerência e controle para habilitar uma adaptação dinâmica dos protocolos em função do contexto e resolver conflitos da federação de redes. Olhando de outra forma, os planos de gerência e controle das redes de computadores são responsáveis por tornar o plano de dados dinâmico, entretanto eles próprios são estáticos. O orquestrador dinâmico cria um dinamismo nesses planos, controlando as funções em execução e ajustando parâmetros dos protocolos. Essa orquestração é importante para um controle mais fino dos algoritmos, o que contribui com o desenvolvimento da gerência autonômica.

Organizar todas as funções de gerência de maneira padronizada em laços autonômicos é interessante por permitir a ação de um controlador de alto nível sobre elas, capaz de iniciar ou interromper o laço, ajustar sua frequência e alterar os parâmetros dos comportamentos dinamicamente. Na arquitetura proposta, cada agente contém um módulo chamado de orquestrador dinâmico que realiza essas funções. Ele interpreta um arquivo de políticas que contém a configuração inicial do agente e sua mudança de estado de acordo com eventos. O orquestrador dinâmico também é capaz de acessar os dados da base de conhecimento.

As políticas consistem em um conjunto de regras condicionais que são avaliadas ciclicamente por um comportamento especial do agente. Quando uma condição retorna o valor verdadeiro, esse comportamento executa a sequência de comandos definidos dentro da regra. Esses comandos mudam o estado do agente, isto é, eles alteram os laços de controle que executam as funções da gerência autonômica. O estado inicial dos agentes também está contido no arquivo de políticas, onde são definidos quais são os comportamentos que serão executados no início. Com isso, é possível diferenciar os tipos de agentes utilizando o mesmo código compilado, alterando apenas o arquivo de políticas.

4.2.4 Interface de monitoramento e controle

Para realizar tarefas de monitoramento e controle sobre os elementos gerenciados, os agentes utilizam uma interface padrão para se tornarem independentes da tecnologia de virtualização. A Tabela 4.1 mostra quais são as primitivas utilizadas pelo sistema de gerência autonômica e onde elas são aplicadas. Os recursos computacionais (segunda coluna) referem-se tanto aos recursos físicos quanto aos virtuais, de todo o sistema computacional até suas partes: processador, memória, armazenamento, interfaces de rede. As

regras de fluxo (terceira coluna) descrevem parâmetros do fluxo, como os endereços de origem e destino, e as ações sobre os pacotes do fluxo, como o encaminhamento para uma ou mais interfaces de saída. Os nós virtuais (quarta coluna) são implementados através de máquinas virtuais, que também são recursos computacionais, mas eles possuem primitivas especiais.

Tabela 4.1: Primitivas da interface do agente com o elemento gerenciado.

Primitiva	Recursos computacionais	Regras de fluxo	Nós virtuais
Habilitar	×		
Desabilitar	×		
Descrição	×		
Capacidade	×		
Utilização	×		
Estatísticas	×	×	
Criar		×	×
Liberar		×	×
Migrar			×
Salvar			×
Restaurar			×

As primitivas *Habilitar* e *Desabilitar* são utilizadas para tarefas de controle como suspender recursos inativos para economia de energia ou interrompê-los para manutenção. *Descrição* e *Capacidade* são primitivas utilizadas para coletar informações sobre os recursos computacionais como identificação, versão do sistema operacional, número de núcleos do processador, quantidade de memória, etc. A primitiva *Utilização* retorna o consumo atual do recurso em valores absolutos ou relativos. Alguns recursos computacionais e fluxos podem também manter informações de histórico sobre sua utilização, como totais e médias. Essas informações são recuperadas através da primitiva *Estatística*. *Criar* e *Liberar* são primitivas que controlam as instâncias dos nós virtuais e das regras de fluxo. A migração dos nós virtuais é realizada através da primitiva *Migrar*. É interessante notar que a migração de enlaces virtuais é realizada com a criação de regras de fluxo no novo caminho da rede de substrato e a liberação das regras no antigo caminho. Por fim, as primitivas *Salvar* e *Restaurar* são utilizadas para criar cópias do estado dos nós virtuais, que podem ser restauradas em caso de falhas, por exemplo.

Capítulo 5

Autocura de redes virtuais

A autocura de redes virtuais é uma das funções do sistema de gerência autonômica e ela contribui com os requisitos de confiabilidade e disponibilidade. Graças à flexibilidade das redes virtuais, de seus recursos serem mapeados dinamicamente sobre a rede de substrato, eventuais problemas ocasionados por falhas na infraestrutura física podem ser rapidamente solucionados com a migração dos recursos virtuais. Essas migrações de recursos virtuais na rede de substrato também podem ser utilizadas para a realização de uma manutenção preventiva ou para desativar nós físicos quando a utilização dos recursos está baixa. Para isso basta marcar os recursos como se eles estivessem falhos, que a função de autocura do sistema de gerência autonômica de redes virtuais se encarrega de migrar os recursos virtuais.

O sistema de gerência autonômica realiza o diagnóstico e o reparo automático de recursos virtuais em caso de falhas. As falhas podem ocorrer nos próprios nós e enlaces virtuais, ou nos nós e enlaces da rede de substrato que dão suporte aos recursos virtuais. No primeiro caso, os recursos virtuais podem ser apenas recriados sobre os mesmos recursos físicos, e no segundo, os recursos virtuais devem ser migrados. A falha em um nó da rede de substrato causa a interrupção dos nós virtuais incorporados nele e também dos enlaces virtuais que passam por ele. Nesse caso, tanto nós quanto enlaces virtuais precisam ser migrados.

Um protótipo do sistema de gerência autonômica de redes virtuais, baseado na arquitetura proposta, foi implementado para a realização de experimentos apresentados no Capítulo 6. Na Seção 5.1 serão abordadas as tecnologias empregadas para o desenvolvimento da plataforma de experimentação e do protótipo. A Seção 5.2 apresenta os detalhes de implementação do modelo de informação. Na Seção 5.3 é descrito o laço de controle autonômico que realiza a função de autocura de redes virtuais. Nessa seção também são apresentados os algoritmos de seleção de nós físicos suplentes, que são utilizados no planejamento proativo das adaptações para corrigir falhas. A Seção 5.5 descreve o mecanismo

proposto para recuperação de nós virtuais, baseado na cópia da memória virtual. E a Seção 5.6 apresenta a interface gráfica desenvolvida para visualização do mapeamento das redes virtuais sobre a rede de substrato e do tráfego das redes virtuais.

5.1 Tecnologias utilizadas

Esta seção irá apresentar as principais tecnologias utilizadas neste trabalho para a implementação da plataforma de virtualização e do protótipo do sistema. Elas são: KVM [Kivity et al., 2007], para criação de nós virtuais; Open vSwitch, [Pfaff et al., 2009], para implementação dos enlaces virtuais; Libvirt [Coulson et al., 2010], para o monitoramento e controle dos nós virtuais; e Ginkgo, [Ginkgo Networks, 2008] para o desenvolvimento do protótipo do sistema de gerência autonômica de redes virtuais.

A Figura 5.1 apresenta onde essas tecnologias são utilizadas nos nós físicos e como elas são organizadas. Os agentes desenvolvidos com o Ginkgo monitoram e controlam as máquinas virtuais através da Libvirt, que fornece uma interface com as plataformas de virtualização. Para gerenciar os enlaces virtuais, implementados com o encaminhamento por fluxos, os agentes Ginkgo utilizam a interface de controle fornecida pelo Open vSwitch. O KVM realiza a alocação de recursos físicos e monitora os recursos virtuais para garantir seu isolamento. O Open vSwitch controla a tabela de fluxos, criando regras para encaminhamento dos pacotes para as interfaces de rede físicas ou virtuais. Com exceção do Ginkgo, que já foi abordado no Capítulo 2, essas tecnologias serão descritas com mais detalhes a seguir.

O KVM é uma plataforma de virtualização total, isto é, suas máquinas virtuais utilizam um sistema operacional sem modificações. Ele roda em arquiteturas x86 com novas tecnologias de virtualização, como Intel VT e AMD-V. O KVM consiste no monitor de máquinas virtuais, implementado em um módulo do kernel do Linux, e nas ferramentas de gerência no espaço do usuário. Ele foi desenvolvido a partir do emulador de máquinas virtuais QEMU Bellard [2005]. Algumas características interessantes do KVM são a migração em tempo real, o mecanismo de expansão de memória e o suporte a SMP (*Symmetric Multi-Processing*) dos processadores físicos e virtuais.

O Open vSwitch é um comutador virtual desenvolvido para ambientes virtualizados. Ele lida com alguns problemas relacionados ao isolamento e à mobilidade de máquinas virtuais. O Open vSwitch exporta uma interface para controle fino do estado da configuração e do comportamento do encaminhamento. Ele é implementado como um módulo que substitui o mecanismo de comutação interna padrão do kernel do Linux. O Open vSwitch utiliza a tabela de fluxos da mesma maneira que um comutador OpenFlow. Na verdade, os fluxos do Open vSwitch podem ser controlados através do protocolo do OpenFlow, portanto, ele é um comutador OpenFlow. Na arquitetura proposta neste trabalho,

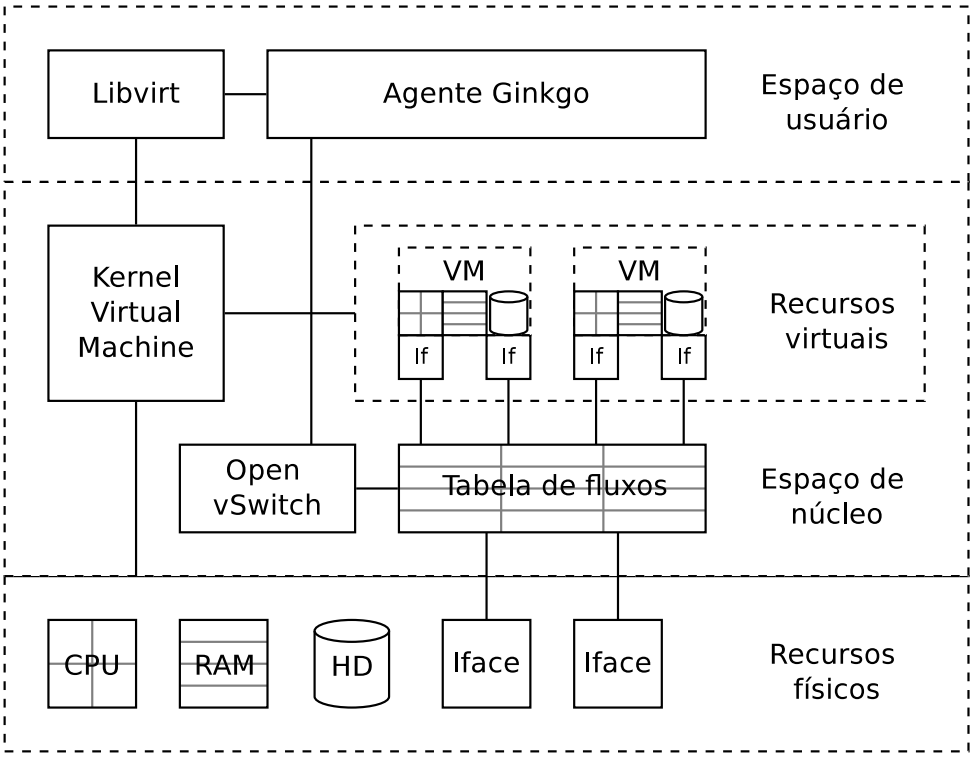


Figura 5.1: Tecnologias empregadas para a implementação das redes virtuais e do protótipo do sistema.

os fluxos são controlados localmente pelos agentes e não por um controlador centralizado, portanto o protocolo OpenFlow não foi utilizado. A interface de controle do Open vSwitch permite criar e liberar fluxos, assim como recuperar estatísticas de utilização através de um único comando, o ovs-ofctl.

A Libvirt é uma biblioteca para monitoramento e controle de máquinas virtuais. Ela oferece uma interface de gerência para várias plataformas de virtualização, incluindo o KVM. A utilização da Libvirt no sistema de gerência autonômica de redes virtuais é importante para torná-lo independente da tecnologia de virtualização. A Libvirt é implementada em C e inclui suporte direto para Python, porém, há *bindings* para outras linguagens, incluindo Java, que foi utilizada no desenvolvimento dos agentes. Através da Libvirt os agentes podem criar, liberar, salvar e restaurar as máquinas virtuais, além de recuperar dados de descrição, capacidade e utilização de recursos virtuais e físicos.

5.2 Modelo de informação

Uma versão simplificada do modelo de informação apresentado na Figura 4.3 foi implementada no protótipo. O código XML do modelo de informação utilizado nos agentes Ginkgo é apresentado na Figura 5.2. Ele é importante para o entendimento dos algoritmos apresentados ao longo deste capítulo. Como é possível observar, as informações na base de conhecimento são organizadas em classes. As instâncias das classes são chamadas de indivíduos. Cada indivíduo possui um nome único na base de conhecimento e as informações são armazenadas nas suas propriedades.

A classe *Topology* contém as listas de nós e enlaces de uma rede, seja ela física ou virtual. A classe *Node* é a generalização de um nó, contendo as propriedades comuns às suas classes especializadas: *PhysicalNode* e *VirtualNode*. Da mesma forma, a classe *Link* é a generalização de um enlace contendo as propriedades comuns às suas classes especializadas: *PhysicalLink* e *VirtualLink*. Os indivíduos são criados a partir das classes especializadas.

As propriedades *cpu* da classe *Node* e *bw* da classe *Link* contêm a quantidade total desses recursos, tanto para nós físicos quanto para virtuais. As classes *Node* e *Link* possuem a propriedade *fail*, que é onde o sistema informa a falha de um recurso físico ou virtual na base de conhecimento. A classe *VirtualNode* possui o atributo *spareNode* que contém o nó da rede de substrato onde o nó virtual deve ser recuperado em caso de falha. O restante das propriedades são referentes aos conectores das classes, como as propriedades *nodes* e *links* da classe *Topology*, e mapeamentos entre recursos físicos e virtuais. Como cada indivíduo de uma classe possui um nome único na base de conhecimento, seu nome é utilizado como chave primária nas buscas por indivíduos.

5.3 Laço autonômico

A função de autocura é realizada através de um laço de controle autonômico. Esse laço realiza as funções de monitoramento, análise, planejamento e execução das adaptações nas redes virtuais. Na arquitetura proposta, cada uma dessas etapas é executada por um comportamento do agente e apesar de formarem um laço de controle fechado, eles agem independentemente, de maneira assíncrona. A base de conhecimento local do agente é compartilhada pelos comportamentos e é através dela que as informações são passadas pelo laço.

A seguir serão descritos os comportamentos dos agentes do protótipo que formam o laço de controle autonômico que realiza a função de autocura de redes virtuais:

Monitoramento: Esse comportamento utiliza a interface de monitoramento e controle do elemento gerenciado, em especial as primitivas de *Capacidade*, *Utilização* e *Es-*

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<GkbModel name="no name" version="1.0">
  <Classes>
    <Class name="THING" isa="NONE"/>
    <Class name="Control" isa="THING">
      <property name="hostName" type="string"/>
      <property name="speed" type="double"/>
      <property name="speedup" type="boolean"/>
    </Class>
    <Class name="Topology" isa="THING">
      <property name="nodes" type="string" multiple="yes"/>
      <property name="links" type="string" multiple="yes"/>
    </Class>
    <Class name="Node" isa="THING">
      <property name="cpu" type="double"/>
      <property name="fail" type="boolean"/>
      <property name="topology" type="string"/>
      <property name="links" type="string" multiple="yes"/>
    </Class>
    <Class name="Link" isa="THING">
      <property name="bw" type="double"/>
      <property name="fail" type="boolean"/>
      <property name="topology" type="string"/>
      <property name="interface" type="string"/>
      <property name="source" type="string"/>
      <property name="target" type="string"/>
    </Class>
    <Class name="PhysicalNode" isa="Node">
      <property name="virtualNodes" type="string" multiple="yes"/>
      <property name="cost" type="double"/>
    </Class>
    <Class name="VirtualNode" isa="Node">
      <property name="physicalNode" type="string"/>
      <property name="spareNode" type="string"/>
    </Class>
    <Class name="PhysicalLink" isa="Link">
      <property name="virtualLinks" type="string" multiple="yes"/>
    </Class>
    <Class name="VirtualLink" isa="Link">
      <property name="physicalLinks" type="string" multiple="yes"/>
    </Class>
  </Classes>
</GkbModel>
```

Figura 5.2: Detalhes do modelo de informação utilizado para autocura de redes virtuais na linguagem XML.

tatísticas, processa os dados e insere as informações da utilização atual dos recursos na base de conhecimento. Nesse processo, ele também pode observar algum problema, como a desconexão de um enlace físico, o que também será notificado na base de conhecimento. Algumas dessas informações serão disseminadas para os demais agentes do sistema.

Análise: Utiliza informações locais, colhidas pelo monitoramento, e remotas, enviadas por outros agentes, para realizar o diagnóstico de falhas. No caso de uma falha em um nó físico, o seu agente irá falhar também e suas informações não serão mais disseminadas. O comportamento *Análise* percebe a falha em um nó físico remoto quando as informações de responsabilidade do agente daquele nó estão desatualizadas.

Planejamento: Esse comportamento define para cada nó virtual de responsabilidade do agente um nó da rede de substrato para recuperar esse nó virtual em caso de falhas. Para isso, ele utiliza um algoritmo que faz a escolha do nó físico suplente de acordo com sua distância relativa ou sua disponibilidade. Esses algoritmos serão descritos mais a frente neste capítulo.

Execução: Caso o comportamento *Análise* tenha apontado falhas, o comportamento *Execução* realiza as migrações de nós virtuais para os nós físicos suplentes escolhidos no comportamento *Planejamento*. Ele também migra os enlaces virtuais afetados para um caminho da rede de substrato definido por um algoritmo de menor caminho não ponderado. Os mapeamentos dos recursos virtuais são atualizados na base de conhecimento após o término da execução.

Os comportamentos dos agentes são executados ciclicamente, assim como a difusão das informações em suas bases de conhecimento. Quanto mais rápidos forem esses ciclos, mais recursos computacionais um agente irá consumir e maior será a sobrecarga na rede. Por outro lado, se o período entre os ciclos forem muito longos, o tempo de resposta dos agentes pode ser demasiadamente longo. Para obter um compromisso entre o tempo de resposta e a sobrecarga das mensagens de controle, foi criado um mecanismo de ajuste dinâmico da frequência do laço de controle autonômico. Esse mecanismo foi desenvolvido através do orquestrador dinâmico e o arquivo de políticas.

A Figura 5.3 mostra o código do arquivo de políticas utilizado pelos agentes. Foram utilizadas duas constantes: *FAST* e *SLOW*, que representam valores de tempo, e a constante *THRESHOLD*, que representa um valor percentual. A política contém um conjunto de regras que são avaliadas ciclicamente a cada 0,5s pelo orquestrador dinâmico. Esse valor é definido no Ginkgo por padrão.

As regras contêm um nome e uma expressão condicional. A regra *RINIT* (linha 4) utiliza a condição *impulse*, que retorna o valor *verdadeiro* apenas uma vez no início da

```
(policy
  (subgoal main
    (rules
      (rule RINIT if (impulse init)
        (
          (set control.speed SLOW)
          (changerate Monitoramento SLOW)
          (changeprio Monitoramento 3)
          (start Monitoramento)
          (setcontrol Analise.threshold THRESHOLD)
          (changerate Analise SLOW)
          (changeprio Analise 2)
          (start Analise)
          (changerate Planejamento SLOW)
          (changeprio Planejamento 1)
          (start Planejamento)
          (changerate Execucao SLOW)
          (changeprio Execucao 4)
          (start Execucao)
        )
      )
      (rule SPEEDUP if (and control.speedup (= control.speed SLOW))
        (
          (set control.speed FAST)
          (changerate Monitoramento FAST)
          (changerate Analise FAST)
          (changerate Planejamento FAST)
          (changerate Execucao FAST)
        )
      )
      (rule SLOWDOWN if (and (not control.speedup) (= control.speed FAST))
        (
          (set control.speed SLOW)
          (changerate Monitoramento SLOW)
          (changerate Analise SLOW)
          (changerate Planejamento SLOW)
          (changerate Execucao SLOW)
        )
      )
    )
  )
)
```

Figura 5.3: Arquivo de políticas que controla a frequência do laço de controle autonômico.

execução do agente. Com essa regra, os comportamentos são configurados e inicializados com a primitiva *start*. A primeira ação da regra *RINIT* é atribuir o valor *SLOW* à propriedade *speed* do indivíduo *control* (linha 6). Essa propriedade representa o atual período do ciclo dos comportamento em segundos. Esse período é definido através da primitiva *changerate*. A sequência de chamadas dos comportamentos pode ser controlada através de prioridades, definidas com a primitiva *changeprio*, e a primitiva *setcontrol* é utilizada para atribuir valores aos parâmetros dos comportamentos. O valor *THRESHOLD* é atribuído ao parâmetro *threshold* do comportamento *Análise* (linha 10). Esse parâmetro representa o percentual de utilização de um dos enlaces físicos do agente para que ele passe de um estado a outro, sendo que se a utilização ultrapassar *THRESHOLD*, o agente deve realizar a transição descrita na regra *SPEEDUP*, e se ela retroceder, o agente deve realizar a transição descrita pela regra *SLOWDOWN*. A regra *SPEEDUP* (linha 22) altera o período dos comportamentos para *FAST* quando o comportamento *Análise* atribui *verdadeiro* à propriedade *speedup* e o valor atual da propriedade *speed* do indivídio *control* é *SLOW*. E a regra *SLOWDOWN* (linha 31) retorna o período para *SLOW* quando o comportamento *Análise* atribui *falso* à propriedade *speedup* e o valor atual da propriedade *speed* do indivíduo *control* é *FAST*.

5.4 Algoritmos de seleção de nós físicos suplentes

O comportamento *Planejamento* do laço de controle autonômico define para cada nó virtual gerenciado pelo agente um nó físico suplente para recuperar o nó virtual. Dessa forma, no caso de uma falha única, o comportamento *Execução* pode utilizar os nós físicos suplentes para recuperar rapidamente a rede virtual. A função de autocura tem como objetivo realizar as ações de reparo nas redes virtuais buscando manter a eficiência da alocação de recursos da rede de substrato. Foram desenvolvidos dois algoritmos para a seleção de nós físicos suplentes com esse objetivo. A função de autocura utiliza um desses algoritmos para o planejamento das ações de reparo da rede virtual em caso de falhas. Em cada ciclo do laço autonômico, o algoritmo é executado proativamente, isto é, antes do diagnóstico da falha.

Os algoritmos são apresentados a seguir em pseudocódigo. Os conjuntos de nós são descritos através da mesma notação matemática utilizada no Capítulo 2. Todas as informações utilizadas para a execução dos algoritmos estão na base de conhecimento dos agentes. O acesso aos valores da base de conhecimento é mostrado através da notação *<indivíduo>.<propriedade>*. A função *GetCandidates* retorna uma lista de nós físicos que não estão falhos e que não possuem incorporados nós virtuais da mesma topologia do nó passado como parâmetro. E a função *GetMinCost* recebe uma lista de nós e retorna aquele que possui o menor valor na propriedade *cost*.

O primeiro algoritmo visa balancear a alocação dos recursos do nó físico. Na primeira etapa do algoritmo (linhas 1 a 8), os custos iniciais dos nós físicos são calculados com base na alocação de recursos do processador. A quantidade de núcleos foi utilizada como parâmetro da capacidade de processamento do nó físico. Os nós virtuais também possuem essa propriedade e seus recursos são subtraídos para ter a quantidade de núcleos residuais do nó físico (linha 4). O custo do nó físico é definido como o inverso dos seus núcleos residuais (linha 8), portanto, aquele que tiver a maior disponibilidade terá o menor custo. Nos algoritmos de seleção não é realizado um controle de admissão e, portanto, a quantidade nominal dos recursos virtuais alocados em um nó ou enlace da rede de substrato pode ser maior do que a quantidade de recursos físicos destes. Nesse caso, o processador não terá núcleos residuais e o custo do nó será o maior valor possível (linha 6). Na segunda etapa do algoritmo (linhas 9 a 14), um nó físico será definido como suplente para cada nó virtual gerenciado pelo agente. Para isso ele seleciona os nós candidatos com a função *GetCandidates* (linha 10) e utiliza o critério de menor custo para selecionar um nó entre os candidatos, através da função *GetMinCost* (linha 11). Ao definir um nó físico suplente para um nó virtual, o algoritmo aumenta o custo do nó físico selecionado para a próxima iteração (linhas 13 e 14). Isso é feito porque quando o nó físico falha todos os nós virtuais incorporados nele serão migrados ao mesmo tempo.

Algoritmo 1: MostAvailableNode

```
1  foreach sNode ∈ N^S do
2      residue ← sNode.cpu
3      foreach vNode ∈ N^V, M^N[sNode, vNode] = 1 do
4          residue ← residue − vNode.cpu
5      if residue <= 0 then
6          sNode.cost ← +∞
7      else
8          sNode.cost ← 1/residue
9  foreach vNode ∈ N^V, M^N[host, vNode] = 1 do
10     candidates ← GetCandidates(vNode)
11     spareNode ← GetMinCost(candidates)
12     vNode.spareNode ← spareNode
13     cost ← 1/(spareNode.cost^−1 − vNode.cpu)
14     spareNode.cost ← cost
```

O segundo algoritmo visa minimizar a utilização dos enlaces. Esse algoritmo também utiliza as funções *GetCandidates* e *GetMinCost*, e duas novas. A função *GetNeighbors* retorna os vizinhos de um nó virtual, isto é, os nós que estão a um salto de distância

na topologia virtual. Para realizar sua tarefa, essa função utiliza as informações dos enlaces virtuais na base de conhecimento. A função *ShortestPathLength* retorna o custo do menor caminho entre dois nós físicos. Esse custo é definido como a distância em números de saltos.

Para cada nó virtual incorporado no nó físico do agente, o algoritmo realiza a seleção dos candidatos através da função *GetCandidates* (linha 2) e zera os custos deles (linha 4). Então o algoritmo constrói uma lista com os vizinhos do nó virtual com a função *GetNeighbors* (linha 5). É importante observar que um salto na rede virtual pode corresponder a mais de um salto na rede de substrato. Então para cada um desses vizinhos o algoritmo recupera seu nó físico (linha 7) e calcula o menor caminho entre esse nó e cada um dos candidatos com a função *ShortestPathLength* (linha 9), somando esse valor no custo do candidato (linha 10). Ao final desse último procedimento a solução de recuperar o nó virtual no candidato de menor custo será a que consumirá a menor quantidade de enlaces físicos e, portanto, esse candidato será selecionado como nó suplente, através da função *GetMinCost* (linha 11).

Algoritmo 2: NearestNode

```
1  foreach vNode ∈ N^V, M^N[host, vNode] = 1 do
2      candidates ← GetCandidates(vNode)
3      foreach candidate ∈ candidates do
4          candidate.cost ← 0
5      neighbors ← GetNeighbors(vNode)
6      foreach neighbor ∈ neighbors do
7          pNode ← neighbor.physicalNode
8          foreach candidate ∈ candidates do
9              length ← ShortestPathLength(candidate, pNode)
10             candidate.cost ← candidate.cost + length
11     vNode.spareNode ← GetMinCost(candidates)
```

5.5 Mecanismo de recuperação de nós virtuais

A migração de um recurso virtual se refere à troca dos recursos físicos que servem de substrato àquele recurso. Essa é a ação tomada pela função de autocura do sistema de gerência autonômica de redes virtuais quando ocorre uma falha na rede de substrato. A migração de um recurso virtual pode ou não causar um período de inatividade desse recurso, que tem como consequência em um cenário de redes de computadores a perda

de pacotes. No caso de falhas em um recurso físico, os recursos virtuais incorporados nele ficarão inativos, portanto, é mais importante um método de recuperação rápida que restabeleça os recursos virtuais do que um método que evite perdas por inatividade durante a migração.

Nas plataformas de virtualização baseadas em máquinas virtuais, a migração é realizada enviando o conteúdo da memória e dos registradores da máquina virtual de uma máquina física de origem para outra de destino. Depois de completar a cópia, a máquina virtual é liberada na origem e executada no destino. Na migração simples, a máquina virtual é interrompida antes do início da cópia da memória e fica inativa até que seja colocada em execução no destino. Na migração em tempo real, a máquina continua em execução durante o processo de cópia da memória, que é realizado iterativamente, enviando em cada iteração as páginas de memória que foram modificadas, até que haja apenas um pequeno número arbitrário de páginas modificadas. A máquina virtual é então interrompida na origem, as últimas páginas da memória e os registradores são enviados, e a máquina é executada no destino. A migração em tempo-real resulta em um tempo menor de inatividade da máquina virtual, porém, ela consome mais recursos ao transmitir pela rede a mesma página de memória mais de uma vez.

Também considera-se como migração o processo que interrompe a máquina virtual na origem e a reinicializa no destino. A diferença desse processo com as migrações citadas acima é que o estado da máquina virtual é perdido, pois a sua memória não é copiada. Ela precisa carregar novamente o sistema operacional na memória virtual para voltar a entrar em operação. Com isso, ela perde dados voláteis como as rotas definidas por um protocolo de roteamento dinâmico. As plataformas de virtualização normalmente oferecem um serviço de salvamento do estado da memória virtual em disco, que pode mais tarde ser utilizado para restaurar a máquina virtual. Esse serviço também pode ser utilizado para migrar uma máquina virtual, mas, ao invés de transmitir diretamente o conteúdo da máquina virtual da origem para o destino, ele é copiado em um repositório acessível por ambas as máquinas.

É possível perceber que os métodos de migração simples e migração em tempo real oferecidos pelas plataformas de virtualização são mais eficientes do que os processos de interrompimento e reinicialização, e de salvamento e restauração de máquinas virtuais. Entretanto, em um cenário de falhas, a máquina física de origem pode estar inoperante e não será possível migrar a máquina diretamente da origem ao destino. Nesse caso a máquina precisa ser recuperada, por reinicialização, carregando o sistema operacional do arquivo de imagem, ou por restauração, através de um arquivo que contenha o estado da memória virtual.

O método de recuperação de uma máquina virtual por restauração é interessante por dois motivos. O primeiro é que ele não tem o impacto do tempo de carregamento

do sistema operacional. O segundo é em função da máquina virtual estar configurada para ser um roteador. Em um cenário de roteamento dinâmico, quando uma máquina entra ou sai da rede, leva um tempo para que o algoritmo de roteamento reconfigure as tabelas de rotas, o que é chamado de tempo de convergência. Se a máquina virtual que falhou for reinicializada ela perderá todas as suas informações de roteamento e o tempo de convergência do algoritmo de roteamento será maior.

Neste trabalho, foi desenvolvido um mecanismo de recuperação de nós virtuais através de uma cópia prévia do estado da memória. Essas cópias podem ser armazenadas no mesmo repositório onde estão as imagens das máquinas virtuais, ou elas podem ser enviadas para os seus nós suplentes. Essa segunda abordagem é interessante pois torna o armazenamento decentralizado, o que também pode reduzir a sobrecarga na rede de substrato. Além disso, o arquivo com o estado da memória virtual já estará armazenado localmente no nó suplente e será carregado mais rápido no caso de haver a necessidade de recuperar o nó virtual.

O serviço oferecido pelas plataformas de virtualização para o salvamento da memória normalmente causa o interrompimento da máquina virtual antes do início da cópia. Isso é um problema para o mecanismo de recuperação de nós virtuais. Uma solução encontrada foi a adaptação do método de migração em tempo real para essa tarefa. A cópia da memória virtual é feita enquanto a máquina virtual está em operação, mas diferente da migração o controle não é transferido.

5.6 Interface gráfica

Uma interface gráfica foi criada para a visualização das redes virtuais e seu tráfego. Um exemplo de janela é apresentado na Figura 5.4. A área chamada *Topology* mostra as topologias e os mapeamentos das redes virtuais sobre a rede de substrato. Ao clicar sobre o nome de um nó nessa área, gráficos com a taxa de envio e recebimento de dados das interfaces desse nó são criados na área *Traffic*. Esses gráficos são atualizados em tempo real. No exemplo da figura estão abertos os gráficos das interfaces dos nós virtuais *horizon.atlas.a* e *horizon.cronos.b*. Os gráficos mostram o tráfego de um arquivo sendo transmitido através de um fluxo TCP pela rede virtual A, que contém o nó *horizon.atlas.a*.

Um agente especial executa um comportamento que gera a interface gráfica. Esse agente recebe as informações difundidas pelos agentes do sistema. Ele constrói em tempo de execução as imagens e os gráficos através das informações na sua base de conhecimento. O desenvolvimento desse comportamento foi feito através das bibliotecas Swing e JChart2D que são implementadas em Java. O posicionamento dos nós físicos na tela é feito manualmente através de um arquivo de configuração e os nós virtuais são posicionados acima dos nós físicos onde estão incorporados.

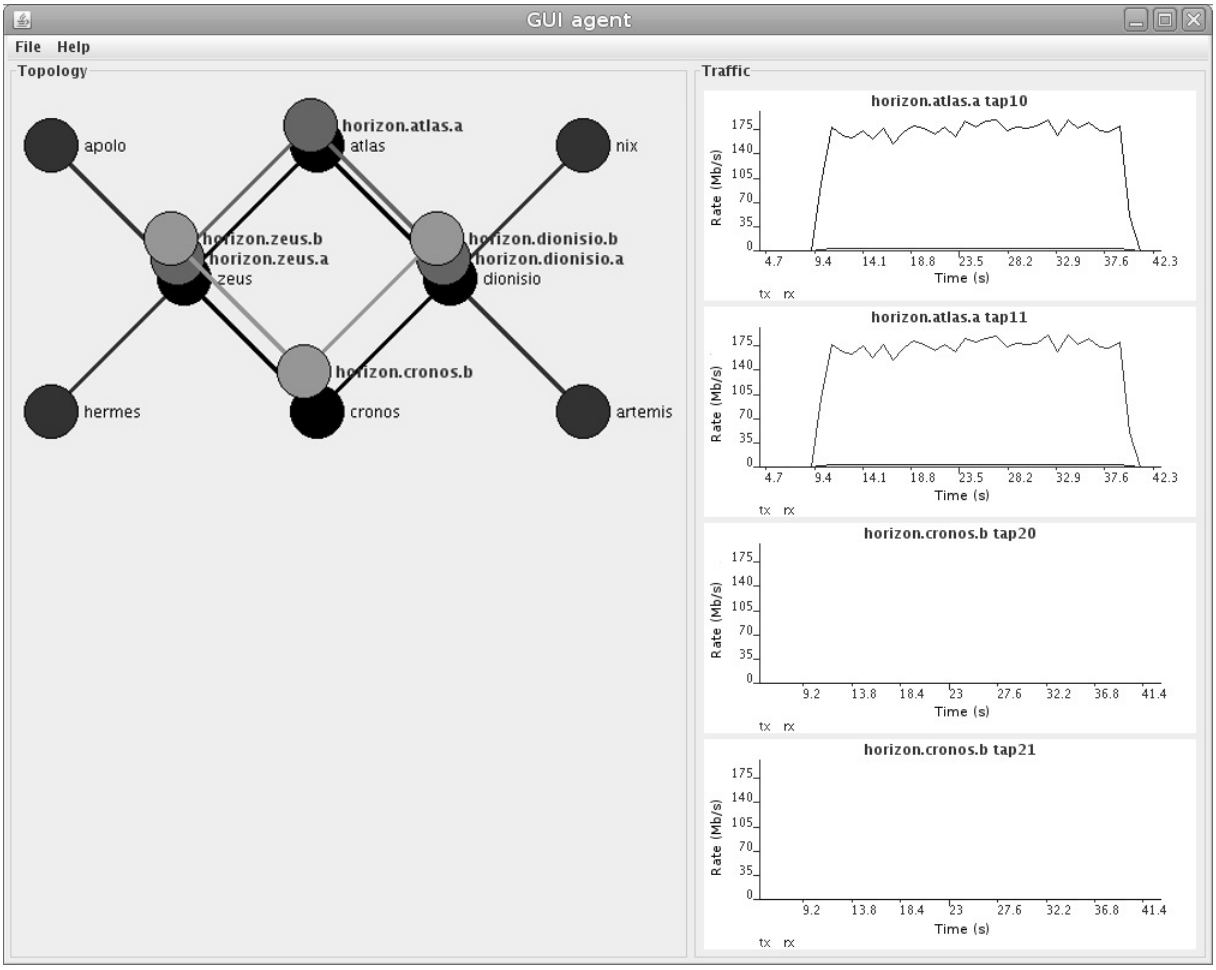


Figura 5.4: Interface gráfica do sistema de gerência autonômica de redes virtuais.

A implementação de uma interface gráfica no protótipo é interessante por dois motivos. O primeiro é que a visualização de dados faz parte da tarefa de gerência e contribui bastante para o desenvolvimento do sistema. O segundo é que o comportamento da interface gráfica mostra que agentes heterogêneos, com diferentes funções, são facilmente criados com a arquitetura proposta. Todos os agentes do protótipo possuem a implementação do comportamento da interface gráfica. Através do orquestrador dinâmico, é possível controlar os comportamentos dos agentes em tempo de execução. Assim, eles podem ser executados a partir do mesmo arquivo executável e modificados através do arquivo de políticas. Os diferentes tipos de agentes do sistema: agente de interface, agente de rede virtual e agente de nó, poderiam ser desenvolvidos de uma maneira semelhante.

Capítulo 6

Avaliação do sistema

Neste capítulo, as ideias propostas na arquitetura do sistema de gerência autonômica de redes virtuais serão avaliadas através de experimentação. Um protótipo com foco na autocura de redes virtuais, descrito no Capítulo 5, foi implementado e instalado na plataforma de experimentação para a execução dos experimentos. As características das redes virtuais desenvolvidas neste trabalho são descritas na Seção 6.1. Os experimentos de adaptação de redes virtuais pelo protótipo são apresentados na Seção 6.2. A Seção 6.3 contém os experimentos em ambientes reais e simulados que avaliam a função de autocura do sistema de gerência autonômica de redes virtuais.

6.1 Plataforma de experimentação

Na plataforma de experimentação, os nós virtuais são criados através de máquinas virtuais KVM e os enlaces virtuais através de fluxos do Open vSwitch. As imagens dos discos das máquinas virtuais são armazenadas em um repositório na rede acessado através do protocolo NFS. Portanto, ao migrar uma máquina apenas o conteúdo da memória é copiado enquanto o disco virtual permanece no repositório. Os nós virtuais utilizam o mecanismo padrão de encaminhamento de pacotes do sistema operacional GNU/Linux. Foram utilizados nos experimentos o roteamento estático, com as rotas inseridas manualmente nos nós virtuais, e também o roteamento dinâmico através do protocolo OSPF, utilizando a suíte de software de roteamento Quagga [Ishiguro, 2006].

A Figura 6.1 mostra um exemplo de criação de enlaces virtuais na plataforma de experimentação. A rede virtual possui dois nós virtuais (*NV1* e *NV2*) e dois enlaces virtuais (*NV1NV2* e *NV2NV1*), um em cada sentido da comunicação. *NV1* foi mapeado no nó físico *NF1* e *NV2* foi mapeado no nó físico *NF3*. As interfaces de rede dos nós virtuais estão conectadas às interfaces virtuais *tap0* dos nós físicos *NF1* e *NF3*. A configuração das tabelas de fluxos de cada nó da rede de substrato é apresentada abaixo dos nós físicos.

O endereço físico da interface virtual de origem é utilizado para especificar os fluxos. Um pacote sendo transmitido de *NV1* para *NV2* através do enlace virtual *NV1NV2* chega a *NF1* pela sua interface virtual *tap0*. O seu comutador Open vSwitch pesquisa através do campo endereço de origem no cabeçalho do protocolo da camada de enlace a qual fluxo aquele pacote pertence. Ao defini-lo como pertencente ao fluxo do enlace virtual NV1VN2, ele realiza a ação de encaminhar o pacote para a sua interface de rede eth1. NF4 recebe o pacote pela sua interface de rede *eth0* e realiza o mesmo procedimento de *NF1*. Por fim, *NF3* encaminha o pacote à *NV2* pela interface virtual *tap0*. O pacote chega a *NV2* sem sofrer modificações no caminho, portanto, é como se ele realmente tivesse sido enviado diretamente de *NV1*. Também não é inserido nenhum cabeçalho adicional no caminho, como é feito no tunelamento de pacotes.

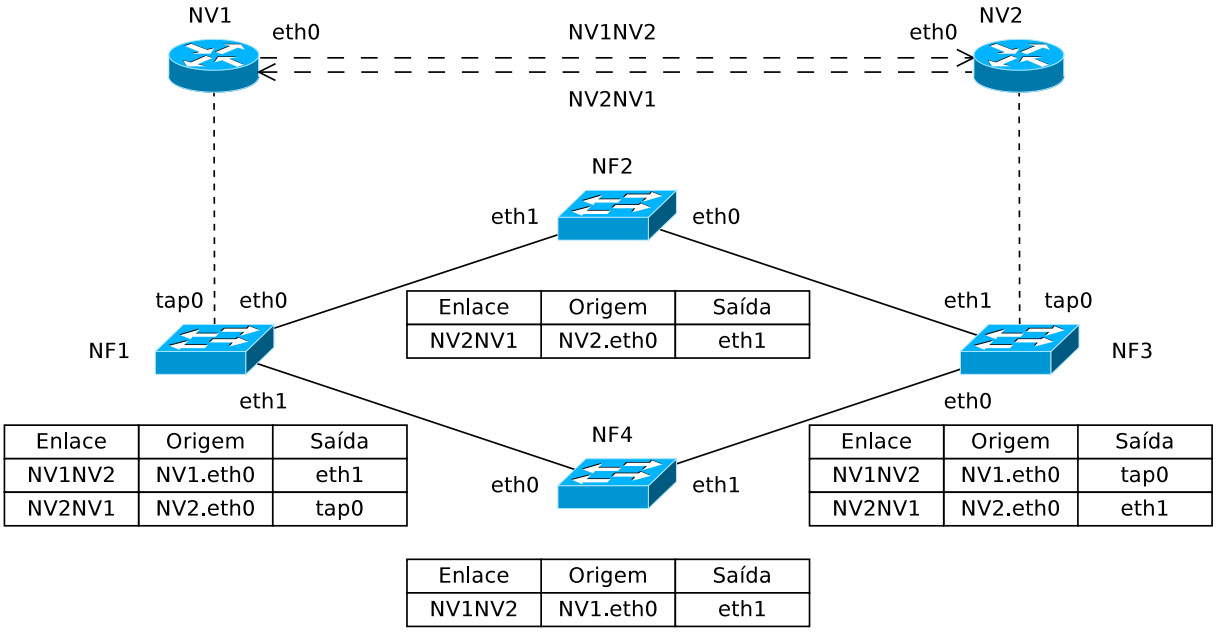


Figura 6.1: Exemplo da configuração de redes virtuais na plataforma de experimentação.

É possível observar na Figura 6.1 que os caminhos dos enlaces *NV1VN2* e *NV2NV1* são distintos. Um pacote enviado de *NV2* para *NV1* passa por *NF2* e não por *NF4*. A utilização de enlaces virtuais direcionados independentes é interessante pois facilita a criação dos fluxos e a implementação do sistema. Além disso, é possível ter situações em que apenas um sentido da comunicação entre dois nós de uma rede virtual é utilizado. Nesse caso, recursos seriam economizados pois apenas um enlace virtual seria criado. O uso de enlaces virtuais direcionados contribui também para o relaxamento do problema de mapeamento das redes virtuais, pois não há nesse caso nenhum tipo de restrição no problema definindo que a comunicação entre dois nós virtuais utilizem o mesmo caminho

em ambos os sentidos.

6.2 Adaptações em redes virtuais

Neste experimento [Senna et al., 2011a], o sistema de gerência autonômica de redes virtuais monitora os enlaces físicos da rede de substrato e, no caso de haver uma sobrecarga em um deles, realiza migrações de enlaces virtuais para evitar perdas de pacotes. O foco da avaliação é o mecanismo de ajuste dinâmico de frequência realizado para atingir um bom compromisso entre o desempenho e a sobrecarga do sistema. Esse mecanismo é realizado pelo orquestrador dinâmico.

A plataforma de experimentação, apresentada na Figura 6.2, foi montada com quatro servidores (*S1*, *S2*, *S3* e *S4*) e dois nós físicos no núcleo da rede (*zeus* e *dionisio*). Estes dois nós físicos foram interconectados por dois enlaces físicos (*EF1* e *EF2*). As interfaces de rede físicas foram configuradas para se comunicarem a uma taxa máxima de 100Mbps. Sobre a rede de substrato foram criadas duas redes virtuais, cada uma contendo dois nós virtuais interconectados por um enlace virtual.

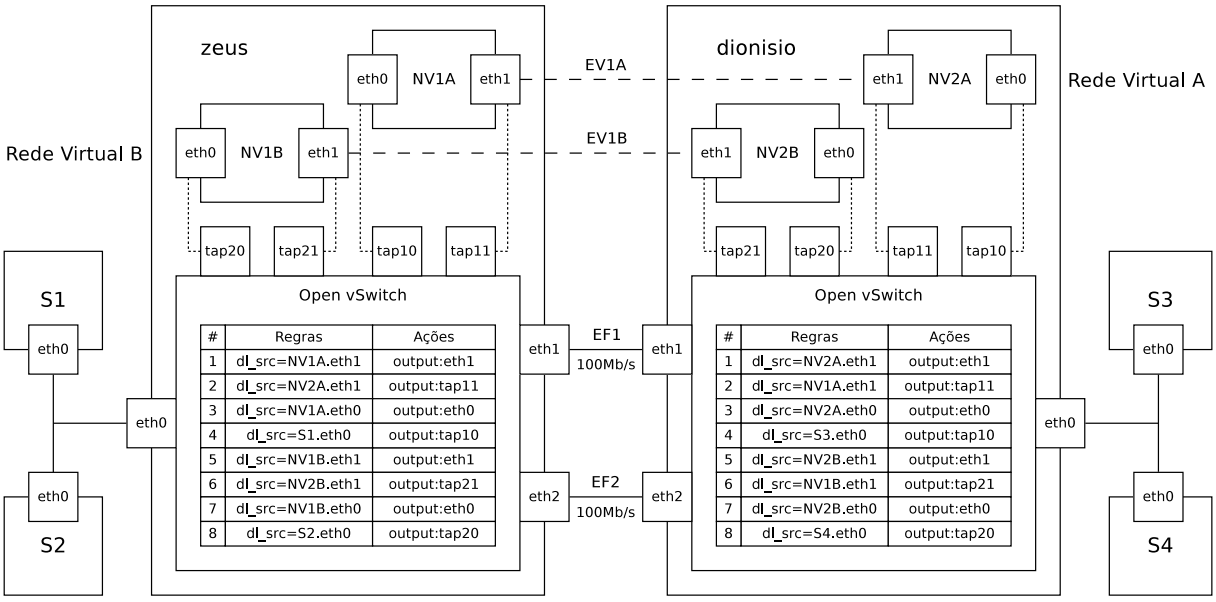


Figura 6.2: Montagem do experimento de adaptações em redes virtuais.

A Figura 6.2 detalha a configuração das tabelas de fluxos dos comutadores Open vSwitch. Os enlaces virtuais são definidos através de regras baseadas nos endereços de camada de enlace das interfaces, permitindo que as redes virtuais utilizem qualquer protocolo da camada de rede. O comutador Open vSwitch analisa o campo de endereço de origem do cabeçalho Ethernet, indicado através da palavra reservada *dl_src*, para definir a

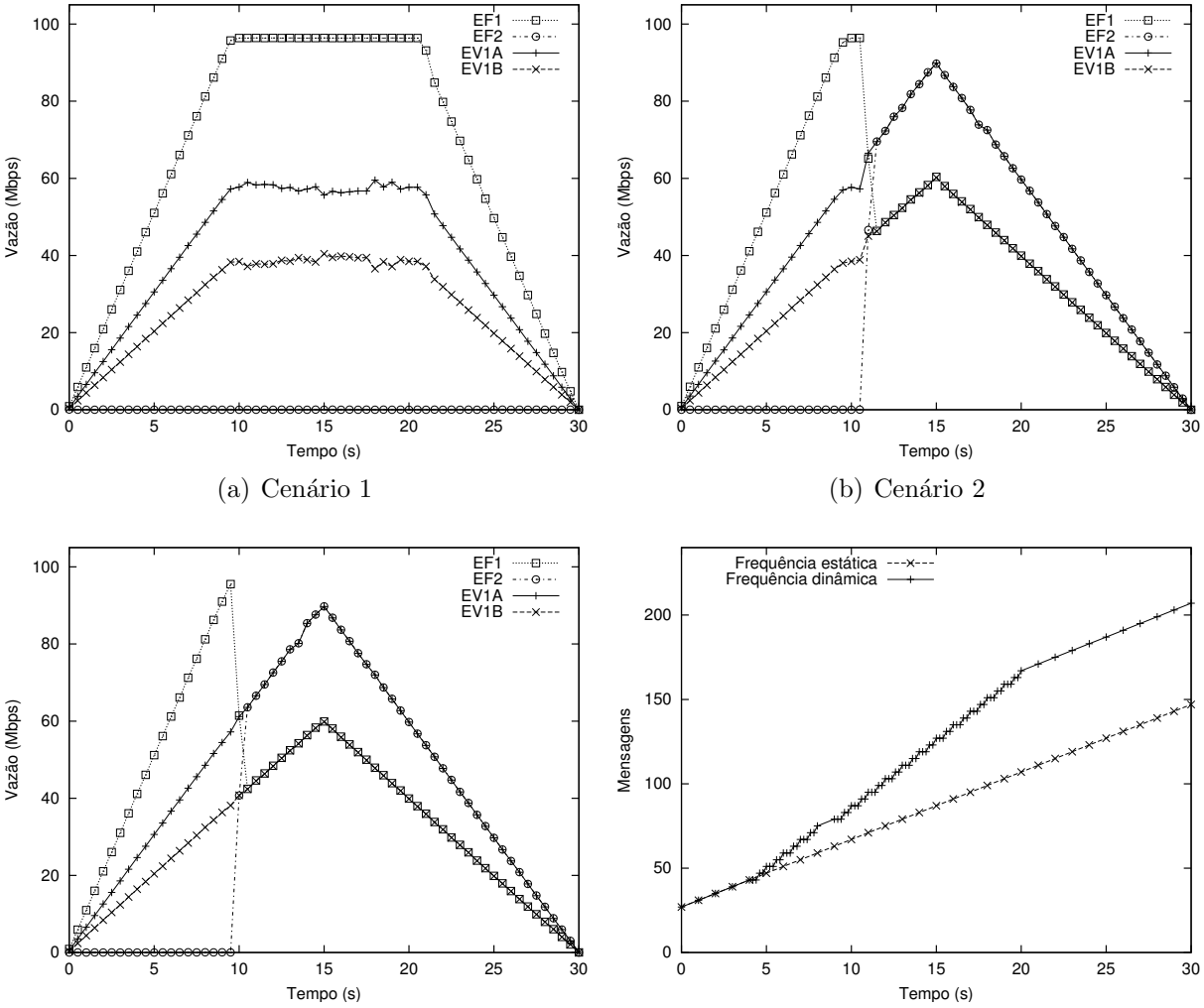
ação sobre cada pacote. O enlace virtual *EV1A* em ambas as direções é mapeado sobre o enlace físico *EF1* através das regras 1 e 2 de *zeus* e *dionisio*. Suas regras 3 e 4 configuram o encaminhamento de pacotes entre a rede virtual A e os servidores *S1* e *S3*. O enlace virtual *EV1B* em ambas as direções é mapeado também sobre o enlace físico *EF1* através das regras 5 e 6 de *zeus* e *dionisio*. E suas regras 7 e 8 configuram o encaminhamento de pacotes entre a rede virtual B e os servidores textitS2 e *S4*.

Nos experimentos realizados, cujos resultados são apresentados na Figura 6.3, dois fluxos UDP foram criados, um de *S1* para *S3* passando pela rede virtual A, e outro de *S2* para *S4* passando pela rede virtual B. No fluxo que passa pela rede virtual A, o tráfego foi modelado na origem para crescer a uma taxa constante de $6MB/s^2$ no intervalo de 0s a 15s, atingindo um pico de 90Mbps, e decrescer a mesma taxa no intervalo de 15s a 30s. No fluxo que passa pela rede virtual B, o tráfego foi modelado na origem para crescer a uma taxa constante de $4MB/s^2$ no intervalo de 0s a 15s, atingindo um pico de 60Mbps, e descrescer a mesma taxa no intervalo de 15s a 30s.

Foram utilizados 3 cenários nesta avaliação e, em todos eles, ambos os enlaces virtuais estão mapeados inicialmente no enlace físico EF1. No cenário 1, os agentes não estão em execução e nenhuma adaptação é realizada. No cenário 2, os agentes estão em execução, porém, seus orquestradores dinâmicos apenas definem o estado inicial dos comportamentos. No cenário 3, os agentes utilizam o mecanismo de ajuste dinâmico de frequência do laço de controle autonômico através dos seus orquestradores dinâmicos.

As curvas dos gráficos dos cenários 1, 2 e 3, na Figura 6.3, correspondem ao tráfego em cada um dos enlaces, dois físicos (*EF1* e *EF2*) e dois virtuais (*EV1A* e *EV1B*). No cenário 1, o enlace físico *EF1* é saturado após 10s e ocorrem perdas de pacotes de ambos os fluxos, enquanto o enlace físico *EF2* está livre (Figura 6.3(a)). O total de pacotes perdidos foi de 24.610. No cenário 2, quando o enlace físico *EF1* está sobrecarregado, o agente em *zeus* realiza uma adaptação, alterando o mapeamento do enlace virtual *EV1A* para o enlace físico *EF2*. Como a taxa de execução do laço de controle autonômico é de 1s, o tempo de reação do agente não é o suficiente para evitar uma perda considerável de pacotes, num total de 632 (Figura 6.3(b)). O tempo de 1s foi escolhido pois ele é o dobro do período de comportamento especial de difusão da base de conhecimento dos agentes, aumentando a probabilidade do laço de controle autonômico ter informações atualizadas dos demais agentes a cada iteração.

Simplesmente aumentar a frequência do laço dos agentes reduziria o tempo de resposta, mas aumentaria a sobrecarga das mensagens trocadas pelo sistema de gerência autonômica de redes virtuais. Para obter um compromisso entre o tempo de resposta e a sobrecarga, o mecanismo de ajuste dinâmico da frequência do laço de controle autonômico é utilizado no cenário 3. A constante *SLOW* (rever Figura 5.3) foi definida como 1s, o mesmo valor utilizado no cenário anterior. A constante *FAST* foi definida como 200ms, um valor



(a) Cenário 1

(b) Cenário 2

(c) Cenário 3

(d) Sobrecarga

Figura 6.3: Resultados da avaliação do protótipo no estudo de caso sobre adaptação de enlaces virtuais.

próximo ao mínimo suficiente para que todos os comportamentos conseguissem terminar sua execução. Apesar do período do laço de controle autonômico ser sido reduzido para 0,2s, o período do comportamento especial de difusão da base de conhecimento do agente foi mantido no seu valor padrão de 0,5s. A constante *THRESHOLD* foi definida como 50%, que foi um valor escolhido arbitrariamente, porém razoável, em função da largura de banda e da taxa de crescimento do tráfego do enlace físico. Portanto, quando o enlace físico *EF1* ultrapassa 50% de utilização, a frequência do laço de controle autonômico do agente passa a ser cinco vezes maior, permitindo uma resposta mais rápida do agente, que realiza a migração do enlace virtual *EV1A* para o enlace físico *EF2* (Figura 6.3(c)).

Ainda assim, houve uma perda de 58 pacotes no total, resultado aproximadamente onze vezes menor em relação ao cenário 2.

Também foi avaliada a sobrecarga introduzida pelo protótipo na rede de substrato nos cenários 2 e 3. A Figura 6.3(d) mostra o número acumulado de mensagens enviadas pelo agente em *zeus* em função do tempo. Durante o tempo em que a utilização do enlace físico *EF1* é maior que 50%, entre os instantes de 5s e 20s, a taxa de mensagens por segundo é apenas duas vezes maior no cenário 3 em relação ao cenário 2. Além disso, no restante do tempo a taxa é a mesma em ambos os cenários. Com isso podemos concluir que o mecanismo de controle dinâmico de frequência do laço autonômico, realizado pelo orquestrador dinâmico, ofereceu uma melhora significativa no tempo de resposta do agente com uma sobrecarga da rede de controle relativamente menor.

6.3 Autocura de redes virtuais

O experimentos dessa seção avaliam a função de autocura de redes virtuais do sistema de gerência autonômica. Foram realizados experimentos em ambiente real, com o objetivo de medir valores referentes ao desempenho do sistema, e experimentos em ambiente simulado, que avaliam a eficiência da alocação de recursos. Os experimentos também são úteis para validar o laço de controle autonômico, o mecanismo de recuperação de máquinas virtuais e comparar os algoritmos de seleção de nós físicos suplentes.

6.3.1 Experimentos em ambiente real

O objetivo deste experimento é avaliar o comportamento de cada tipo de migração de máquinas virtuais e medir o seu impacto em um cenário de redes virtuais. O protótipo do sistema de gerência autonômica de redes virtuais não foi utilizado nesse experimento. Na montagem da plataforma de experimentação foram utilizados dois servidores, dois nós físicos da rede de substrato, e um roteador virtual. No experimento, uma cópia de um arquivo de 1GB é feita de um servidor ao outro, através da aplicação SCP. Essa aplicação gera um fluxo TCP que foi configurado para passar através do roteador virtual, cujas rotas foram definidas manualmente. Aproximadamente 10s após o início da cópia, o roteador virtual é migrado para o outro nó da rede de substrato. Foi realizada uma medição de controle, com o arquivo sendo transmitido através do roteador virtual sem nenhum tipo de migração. Essa cópia levou cerca de 24s para ser concluída.

A Figura 6.4 mostra os tempos de execução e os períodos de inatividade da aplicação de cada execução, através das curvas do tráfego de chegada no servidor de destino. A curva "Migração em tempo real" mostra a execução do experimento com esse tipo de migração, em que o roteador virtual continua em execução durante o processo de cópia da

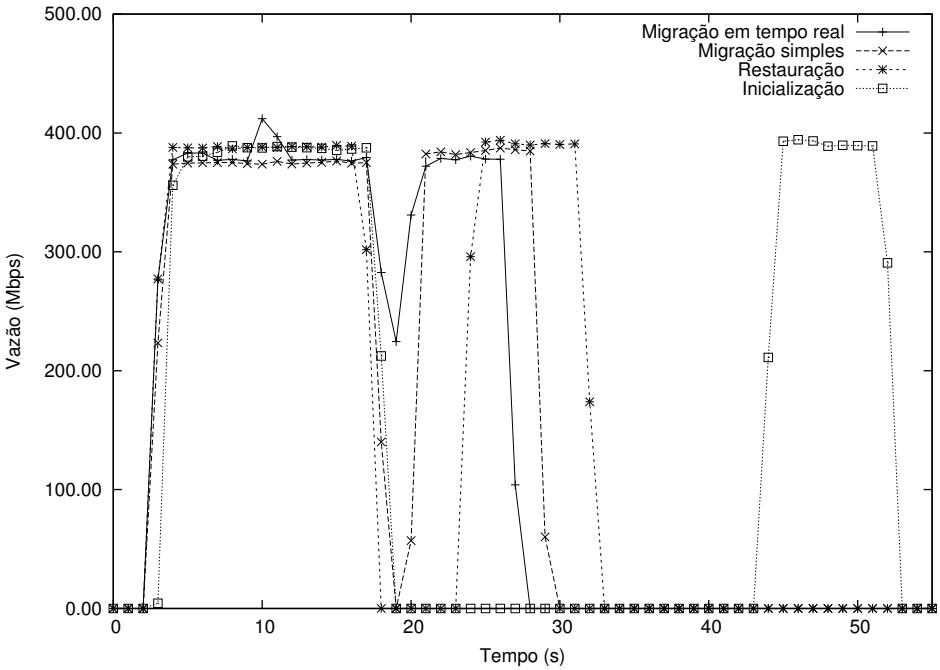


Figura 6.4: Resultados dos experimentos sobre impacto da migração de nós da rede virtual.

sua memória. Esse tipo de migração é a que resulta no menor período de inatividade e a cópia levou cerca de 25s para ser concluída, valor próximo ao da medição de controle. A curva "Migração simples" mostra a execução do experimento com esse tipo de migração, em que o roteador virtual é interrompido durante o processo de cópia da sua memória. Nesse caso, o período de inatividade é um pouco maior e a cópia levou cerca de 27s para ser concluída. A curva "Restauração" mostra a execução do experimento com o salvamento do roteador virtual na origem e restauração no destino. Esse caso é semelhante ao anterior, porém, a memória da máquina virtual é primeiramente salva em um repositório na rede e em seguida ela é restaurada no nó físico de destino. Como o acesso ao disco é mais lento e duas transmissões da memória virtual são realizadas, o período de inatividade aumenta e a cópia levou cerca de 30s para ser concluída. A curva "Inicialização" mostra a execução do experimento com com a liberação da máquina virtual na origem e a inicialização no destino. Essa execução teve um longo período de inatividade decorrente do processo de inicialização do sistema operacional da máquina virtual e a cópia levou cerca de 50s para ser realizada.

A Figura 6.5 ilustra a montagem da plataforma de experimentação construída para o próximo experimento [Soares e Madeira, 2011]. O núcleo da rede de substrato é composto por quatro máquinas: *zeus*, *atlas*, *dionisio*, e *cronos*. Elas possuem um comutador interno

Open vSwitch para a criação dos enlaces virtuais. As máquinas de borda (*zeus* e *dionisio*) estão conectadas aos servidores *apolo*, *hermes*, *nix*, e *artemis*, através de comutadores Giga Ethernet. Sobre a rede de substrato, duas redes virtuais, contendo três nós virtuais cada, foram instanciadas. Os nós virtuais foram criados através de máquinas virtuais KVM e possuem o roteamento IPv4 habilitado. As imagens das máquinas virtuais estão em um repositório acessível por todas as máquinas da rede de substrato via NFS. Todas as máquinas físicas e virtuais utilizam o sistema operacional Debian GNU/Linux, com a versão do kernel 2.6.32.

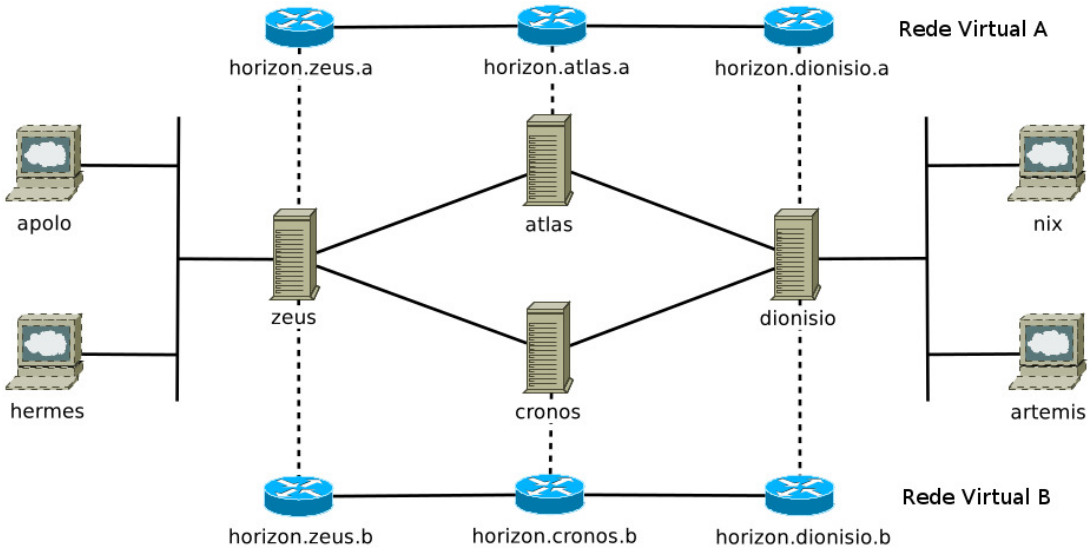


Figura 6.5: Cenário utilizado na avaliação da função de autocura de redes virtuais em ambiente real.

Este experimento têm o objetivo de avaliar a arquitetura da gerência autonômica de redes virtuais e testar diferentes abordagens para a recuperação de nós virtuais em caso de falhas nos nós da rede de substrato. O tempo de recuperação T_r de uma rede virtual pode ser definido como:

$$T_r = T_d + T_p + T_i + T_c \tag{6.1}$$

onde T_d é o tempo do diagnóstico da falha, T_p é o tempo gasto planejando as ações de correção, T_i é o tempo de instanciação da máquina virtual, e T_c é o tempo de convergência do protocolo de roteamento.

Dois métodos de recuperação de nós virtuais foram implementados nos agentes do protótipo. O Método 1 inicializa a máquina virtual a partir da imagem do seu disco contida no repositório da rede. O Método 2 restaura a máquina virtual a partir de uma cópia da sua memória, que está no mesmo repositório e que foi gerada em um

momento em que o nó virtual estava operacional. Neste último caso, não há o atraso decorrente da inicialização do sistema operacional e o tempo de convergência do protocolo de roteamento é menor. Dois cenários de configuração das redes virtuais foram avaliados neste experimento. O Cenário 1 utiliza um roteamento estático, em que as rotas são manualmente configuradas nos nós virtuais. O Cenário 2 utiliza um roteamento dinâmico, que utiliza o protocolo de roteamento OSPF fornecido pela suíte de roteamento Quagga.

A aplicação de cópia segura (SCP) gera um tráfego TCP, transferindo um arquivo de 1GB de *hermes* para *artemis* através da rede virtual B. Para cada combinação dos cenários com os métodos de recuperação dos nós virtuais, foram realizadas 10 repetições e foi calculado um intervalo de confiança de 95%. Todas as execuções foram realizadas em sequência, em um curto espaço de tempo, em que as condições da rede eram similares.

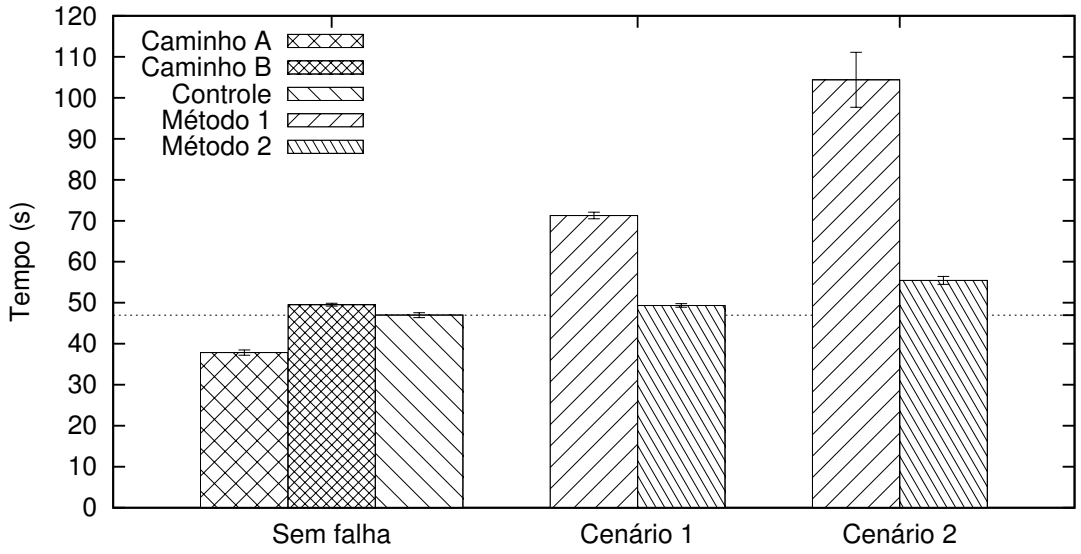


Figura 6.6: Desempenho da função de autocura de redes virtuais em ambiente real.

Os resultados são apresentados na Figura 6.6. Foram realizadas execuções na plataforma de experimentação sem os agentes para servirem de base de comparação. Para isso foram medidos os tempos para enviar o arquivo através do caminho A (*zeus* → *cronos* → *dionisio*) e do caminho B (*zeus* → *atlas* → *dionisio*). Como é possível observar, as taxas de transmissão do Caminho A e B são diferentes. Portanto, foram realizadas execuções do experimento com uma migração em tempo real do nó virtual *horizon.cronos.b* de *cronos* para *atlas* após 10s do início da cópia do arquivo, para a obtenção de uma medição de controle, pois a migração em tempo real resulta no menor período de inatividade, como mostra experimento anterior. Em seguida, foram executados os experimentos com o protótipo que realiza a autocura das redes virtuais. Uma falha no nó físico *cronos* ocorre também após 10s do início da aplicação e o protótipo recupera o nó virtual *horizon.cronos.b* em

atlas.

Utilizando o Método 1, que inicializa a máquina virtual a partir do arquivo de imagem do seu disco, o tempo para completar a cópia do arquivo foi 52% maior no Cenário 1, com roteamento estático, e 122% maior no Cenário 2, com roteamento dinâmico, em relação à medição de controle. Nota-se que nesse caso há uma diferença considerável no tempo de recuperação da rede virtual em função do cenário utilizado. Isso se deve principalmente ao tempo de convergência do algoritmo de roteamento. Com o Método 2, que restaura a máquina virtual a partir de um cópia da memória, o tempo para completar a cópia do arquivo foi de 5% e 18% maior em relação à medição de controle nos Cenários 1 e 2, respectivamente.

Os resultados são favoráveis à utilização do Método 2, que recupera a máquina virtual a partir de uma cópia da memória, especialmente no Cenário 2, em que as redes virtuais utilizam um roteamento dinâmico. Apesar desse método ser mais rápido, o salvamento, a transmissão e o armazenamento da cópia da memória dos nós virtuais podem causar um impacto significativo na rede de substrato. Para reduzi-los, o sistema pode fazer as cópias das memórias dos nós virtuais em períodos de inatividade da rede. Além disso, um armazenamento distribuído e a utilização de um mecanismo de expansão de memória sob demanda, provido por algumas plataformas de virtualização, também podem ser utilizados para reduzir a sobrecarga desse método.

6.3.2 Experimentos em ambiente simulado

O ambiente simulado foi utilizado para avaliar como as adaptações realizadas pela função de autocura impactam na alocação de recursos da rede de substrato. A plataforma de experimentação utilizada nos experimentos em ambiente real é pequena para esse tipo de avaliação. O protótipo do sistema de gerência autonômica de redes virtuais é o mesmo em ambos os casos. No cenário simulado, todos os agentes são executados em processos na mesma máquina. Portanto, o tempo não é simulado, embora existam algumas variações decorrentes do tempo de propagação das mensagens ser um pouco menor. A troca de informações entre os agentes é realizada da mesma maneira no cenário real e no simulado, apenas as primitivas de monitoramento e controle dos recursos foram alterados nesse último caso para executar as ações apenas na base de conhecimento, onde os recursos simulados estão definidos.

A função de autocura tem como objetivo realizar as ações de reparo nas redes virtuais de maneira rápida e ao mesmo tempo manter a eficiência da alocação de recursos físicos. Essa eficiência é definida na literatura através da taxa de aceitação das requisições de redes virtuais ou da receita do provedor de infraestrutura [Chowdhury et al., 2009; Yu et al., 2008]. Entretanto, no caso da função de autocura, o sistema de gerência autonômica

não lida com requisições de redes virtuais e sim com falhas. Portanto, serão utilizadas outras métricas que fornecem indicadores de eficiência da alocação de recursos em um dado instante.

A quantidade de recursos alocados para os nós virtuais é a mesma independente do mapeamento da rede virtual. Por outro lado, os enlaces virtuais são mapeados sobre caminhos no substrato e, portanto, quanto maior o número de enlaces no caminho, maior será o consumo desses recursos. Ao minimizar a utilização dos enlaces da rede de substrato é provável que a taxa de aceitação das requisições das redes virtuais aumente e, em consequência disso, as receitas do provedor de infraestrutura aumentem também. Assim, uma das métricas para a avaliação da eficiência da alocação de recursos é a utilização dos enlaces ρ, que também é utilizada no trabalho de Fajjari et al. [2011a], através da equação:

$$\rho = \sum_{e^S \in E^S} \sum_{e^V \in E^V} [M^E[e^S, e^V] \times bw(e^V)] \tag{6.2}$$

onde E^S é o conjunto de enlaces da rede de substrato, E^V é o conjunto de enlaces de todas as redes virtuais, M^E é a matriz de mapeamento dos enlaces virtuais sobre os enlaces físicos, e $bw(e^V)$ é a largura de banda consumida pelo enlace virtual.

A outra métrica utilizada representa o balanceamento da alocação de recursos. Se a alocação de recursos for desbalanceada, alguns recursos podem ficar saturados rapidamente, causando perdas de conectividade na rede de substrato e, assim, redução da taxa de aceitação das requisições de redes virtuais e das receitas do provedor de infraestrutura. Diferentemente do trabalho de Zhu e Ammar [2006], o balanceamento foi definido como desvio padrão σ dos recursos residuais da rede de substrato, e o recurso utilizado foi a capacidade de processamento do nós físico, conforme as equações abaixo:

$$\mu = \frac{1}{|N^S|} \sum_{n^S \in N^S} cpu'(n^S) \tag{6.3}$$

$$\sigma = \sqrt{\frac{1}{|N^S|} \sum_{n^S \in N^S} [cpu'(n^S) - \mu]^2} \tag{6.4}$$

onde N^S é o conjunto de nós da rede de substrato e $cpu'(n^S)$ é a quantidade residual de recursos de processamento do nó físico.

A rede de substrato simulada contém dez nós, como mostra a Figura 6.7. A topologia da rede de substrato é a mesma utilizada no trabalho de Houidi et al. [2010]. O nós físicos contêm processadores de quatro núcleos e são conectados através de enlaces de 1Gbps. Sobre a rede de substrato foram instanciadas três redes virtuais com cinco nós cada. Essas redes virtuais possuem topologias livres de escala criadas aleatoriamente. Todos os nós virtuais ocupam um núcleo de processamento e os enlaces virtuais utilizam 50Mbps

da taxa de transmissão. A incorporação dessas redes virtuais é realizada através de um mapeamento aleatório de nós virtuais seguido de um mapeamento de enlaces virtuais que considera o menor caminho. Dessa forma, a alocação inicial dos recursos do substrato para as redes virtuais não é ótima em relação às métricas utilizadas.

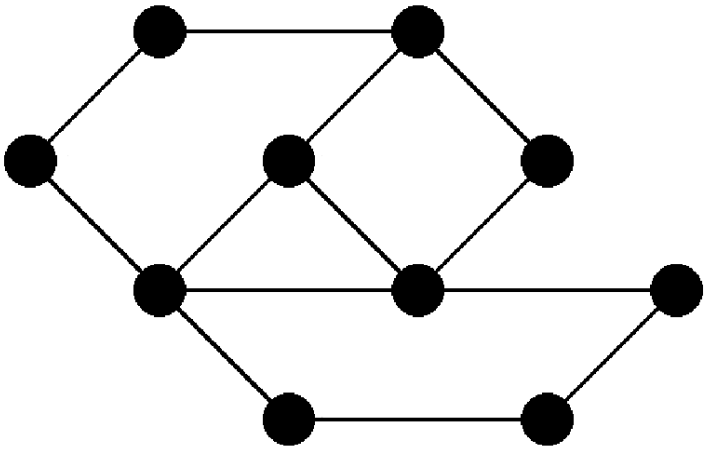
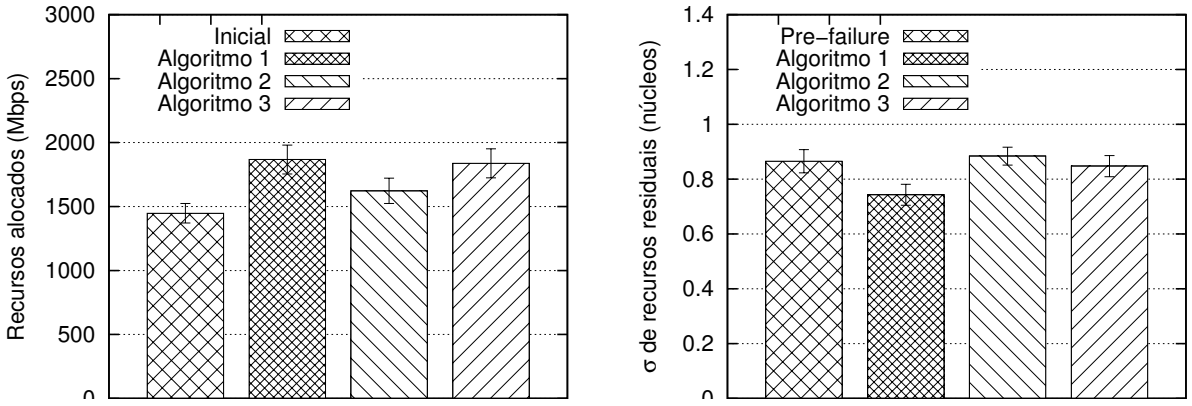


Figura 6.7: Cenário utilizado na avaliação da função de autocura em ambiente simulado.

Neste experimento [Soares e Madeira, 2012], um evento de falha ocorre em um nó aleatório da rede de substrato depois da incorporação de todas as redes virtuais. O protótipo realiza o diagnóstico da falha e repara as redes virtuais afetadas de acordo com o planejamento proativo. Foram utilizados os algoritmos apresentados no Capítulo 5 para a seleção de nós físicos suplentes. O Algoritmo 1 (*MostAvailableNode*) visa balancear os recursos residuais dos nós físicos. O Algoritmo 2 (*NerestNode*) tem o objetivo de minimizar a utilização dos enlaces físicos. Como base de comparação, foi criado o Algoritmo 3, que escolhe aleatoriamente o nó físico suplente para cada nó virtual entre os possíveis. Para cada algoritmo de seleção de nós físicos suplentes, 20 repetições foram realizadas. Como os valores iniciais das métricas, medidos antes do evento de falha, são bastante aleatórios, foi utilizado um intervalo de confiança de 70%.

A Figura 6.8 apresenta os resultados deste experimento. Para cada métrica utilizada, os seus valores iniciais foram comparados aos valores medidos após a reparação dos recursos virtuais pela função de autocura do sistema de gerência autonômica. A Figura 6.8(a) mostra que a utilização dos enlaces do substrato aumenta depois de um evento de falha. Isso é esperado, uma vez que falhas nos nós do substrato causam a perda de algumas conexões e, assim, os caminhos da rede de substrato utilizados pelos enlaces virtuais tendem a tornarem-se maiores. Entretanto, o Algoritmo 2 resulta em uma utilização menor dentre os algoritmos avaliados. Também observa-se que o Algoritmo 1 tem um comportamento semelhante e em média até pior em relação ao Algoritmo 3. Por outro lado o Algoritmo

1 melhora o balanceamento dos recursos residuais dos nós do substrato, mesmo quando comparado à situação inicial, como mostra a Figura 6.8(b). Vale a pena ressaltar que o mapeamento inicial dos nós virtuais é realizado de maneira aleatória, não sendo realizado nenhum tipo de otimização.



(a) Utilização dos enlaces físicos

(b) Balanceamento dos nós físicos

Figura 6.8: Avaliação da eficiência da alocação de recursos da função de autocura de redes virtuais em ambiente simulado.

Os resultados sugerem que é possível realizar uma autocura de redes virtuais através de adaptações simples, de forma a manter a eficiência da alocação de recursos sem a necessidade de remapear a rede virtual toda. Isso é importante no nosso sistema de gerência autonômica de redes virtuais, onde as ações são realizadas localmente através dos agentes distribuídos e, também, porque os nós do substrato podem não ter recursos computacionais suficientes para executar algoritmos complexos em tempo aceitável.

Capítulo 7

Estudo de caso: gerência de redes virtuais em nuvens

Nos capítulos anteriores foi apresentada uma gerência de redes virtuais agnóstica em relação às suas aplicações. Essa gerência monitora os recursos da infraestrutura da rede e realiza ações de remapeamento dos recursos virtuais. Neste capítulo, será apresentado um estudo de caso de gerência de redes virtuais para atender aplicações distribuídas executadas em uma nuvem. A monitoração não leva em consideração apenas o consumo dos recursos da infraestrutura, como processamento, memória e banda passante, mas também lida com informações das aplicações, como níveis de prioridades, ou o tempo consumido na execução de alguma tarefa. Para este estudo foram elaborados diversos cenários com o intuito de mostrar como a gerência de redes virtuais pode impactar na execução de aplicações em nuvens.

Uma nuvem computacional é um *pool* de recursos pertencentes a uma organização que pode ser compartilhado por diversos usuários. Esses usuários, sejam pertencentes à própria organização, no caso de uma nuvem privada, ou pertencentes a outras organizações, no caso de uma nuvem pública, são cobrados pela quantidade de recursos efetivamente utilizados, como um serviço. Os recursos providos podem estar no nível de infraestrutura, no modelo IaaS (*Infrastructure as a Service*), no nível de plataforma de desenvolvimento, no modelo PaaS (*Platform as a Service*), ou no nível de software, no modelo SaaS (*Software as a Service*). A virtualização é um conceito chave para nuvens, pois os recursos computacionais precisam ser rapidamente aprovisionados e facilmente gerenciáveis, para atender em um curto espaço de tempo as solicitações feitas pelos clientes. As nuvens permitem que as aplicações sejam elásticas, isto é, que elas sejam capazes de se expandir e se contrair no *pool* de recursos de maneira automática.

As aplicações colocadas em execução na nuvem podem ser sistemas distribuídos de larga escala. Nesse tipo de aplicação, o transporte de dados entre seus subsistemas impacta

diretamente no seu desempenho global. Atualmente em uma nuvem, apenas os servidores são virtualizados e os recursos de rede são compartilhados entre os usuários. O trabalho de Truong Huu et al. [2011] apresenta uma proposta de extensão do modelo de computação em nuvem com a adição de redes virtuais. Um provedor de serviço interessado em utilizar a nuvem requisita os recursos de rede (roteadores, enlaces, pontos de acesso) assim como os demais recursos para o processamento e armazenamento dos dados. As redes virtuais podem ser utilizadas pelos provedores de infraestrutura para isolar e garantir QoS nas comunicações de seus clientes.

Uma gerência única da nuvem nesse cenário pode não ser viável dada a complexidade do problema de controlar todos os recursos virtuais necessários para a execução de uma aplicação. Portanto, podemos utilizar um modelo que separa a gerência da nuvem em dois níveis, um responsável pela gerência de aplicações e outro, pela gerência de redes virtuais. No exemplo da Figura 7.1, um *workflow* com 5 tarefas é submetido pelo cliente à gerência de aplicações. Essa gerência utiliza um algoritmo de escalonamento para escolher os recursos computacionais onde as tarefas serão executadas, sem o conhecimento do núcleo da rede. A definição do *workflow* com suas tarefas mapeadas nos servidores é encaminhada em seguida à gerência de redes virtuais. Essa gerência utiliza um algoritmo de mapeamento que especifica uma rede virtual interligando os servidores previamente escolhidos para atender os fluxos de dados do *workflow*.

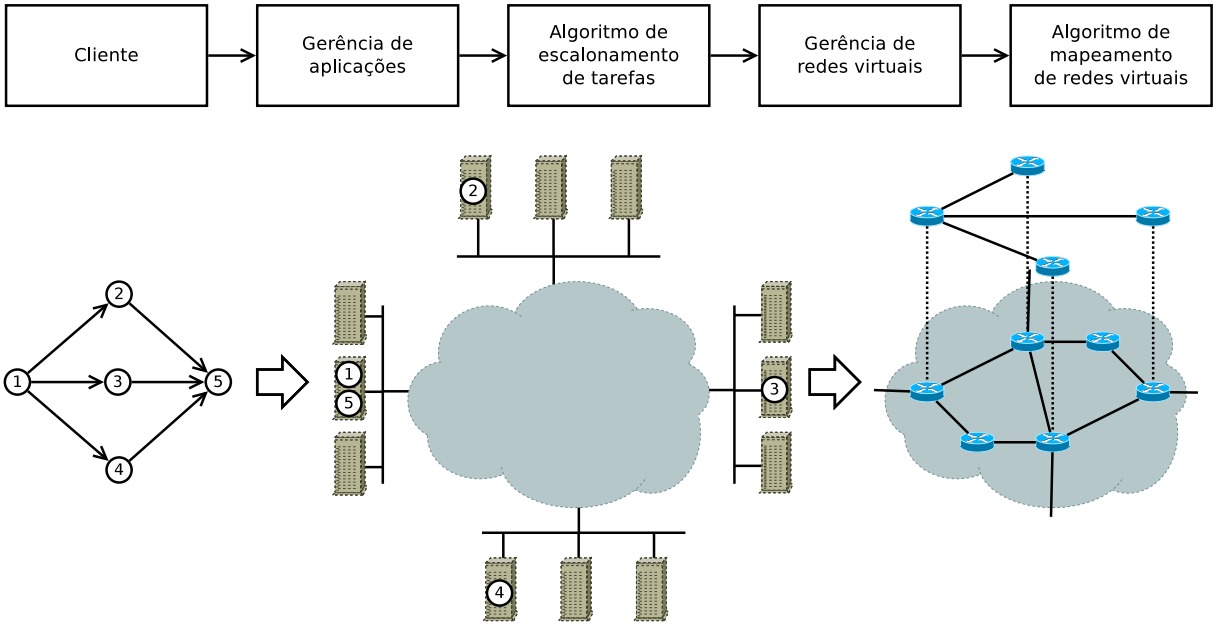


Figura 7.1: Gerência em dois níveis de *workflow* e rede virtual para computação em nuvem.

Essa divisão de papéis é interessante pela especialização da gerência, mas pode levar a uma alocação de recursos não ótima. É importante que existam interfaces bem definidas entre a gerência de aplicações e a gerência de redes virtuais para que possam trocar informações entre si e fazer solicitações visando uma otimização global. A integração da gerência de aplicações e das redes é conhecida na literatura como redes cientes de serviços e serviços cientes de redes [Galis et al., 2009].

Neste capítulo será apresentada uma arquitetura para a integração de um sistema de gerência de aplicações com um sistema de gerência de redes virtuais. Foram realizados experimentos que mostram o impacto das redes virtuais na execução de uma aplicação em relação a uma rede não virtualizada. Também foi avaliado através de experimentação como a flexibilidade das redes virtuais pode contribuir para a execução de uma única aplicação, ou de várias com diferentes níveis de prioridades.

O protótipo do sistema de gerência de redes virtuais apresentado nos capítulos anteriores não foi utilizado nos experimentos, pois seria necessário o sistema completo, com o agente de interface e os agentes de redes virtuais para este estudo de caso. Também teria que ser definido como a gerência de aplicações se comunicaria com a gerência de redes virtuais para realizar solicitações e sinalizar problemas. No entanto, o objetivo deste capítulo não é avaliar o funcionamento do sistema de gerência autonômica de redes virtuais, mas sim mostrar de que maneiras essa gerência pode contribuir na execução de aplicações em uma nuvem.

7.1 Arquitetura de gerência de integrada

A arquitetura do sistema de gerência integrada de aplicações e redes virtuais em nuvens é baseado no GPO (*Grid Process Orchestration*) [Senna et al., 2010], um *middleware* para execução de *workflows* baseado em serviços (Figura 7.2). Os *workflows* são definidos através da GPOL (*GPO Language*). A GPOL utiliza conceitos de orquestração de serviços do WS-BPEL, com a adição de diretivas específicas para nuvens, como manutenção de estado, serviços potencialmente transitórios, notificação, serviços orientados a dados, e grupos. Adicionalmente ela permite ao usuário iniciar a execução de serviços em série ou em paralelo. O escalonador é responsável pela distribuição dos serviços do *workflow* nos recursos disponíveis. Para isso, o escalonador pode ter implementado diferentes algoritmos com diferentes funções objetivo, e decidir qual utilizar dependendo da aplicação ou das características atuais do ambiente. Informações sobre os recursos disponíveis na nuvem podem ser obtidas através do monitor de recursos, que opera de maneira distribuída, com uma instância em cada recurso computacional, provendo informações sob demanda. O GPO monitora os tempos de execução de cada seção do *workflow*, especificado através da GPOL, incluindo os tempos gastos em cada operação invocada no processo, e salva em

um arquivo exclusivo para cada *workflow*.

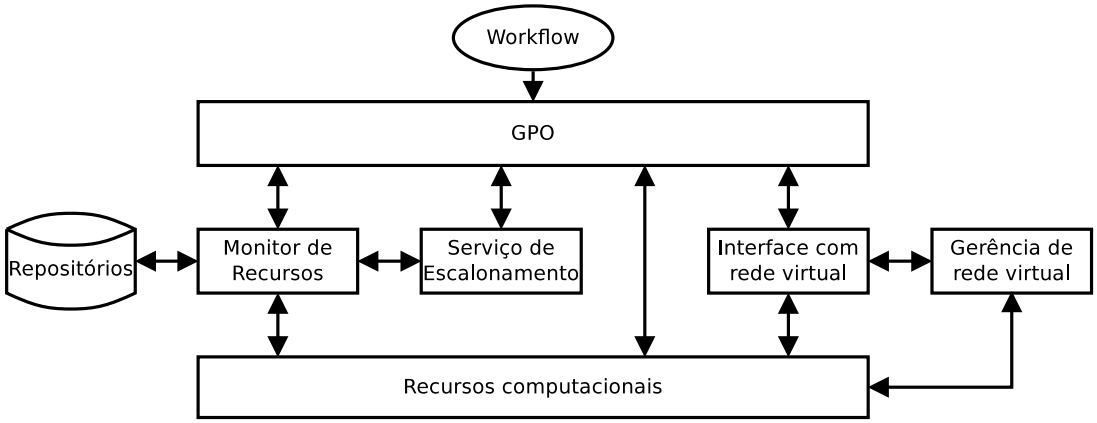


Figura 7.2: Arquitetura do sistema de gerência de *workflow*.

O GPO utiliza uma interface para a integração com o sistema de gerência de redes virtuais, como mostra a Figura 7.2. Através dessa interface, o GPO requisita uma rede virtual para a execução do *workflow*. O monitor de recursos monitora o desempenho da rede virtual podendo identificar problemas como falha ou degradação do serviço de comunicação. Nesses casos, o GPO notifica o sistema de gerência de redes virtuais, requisitando melhorias no desempenho do enlace ou da rede virtual como um todo. A utilização de uma interface bem definida entre o sistema de gerência de aplicações e o sistema de gerência de redes virtuais é interessante para torná-los independentes de implementação.

Neste capítulo, os *workflows* utilizados nos experimentos serão baseados na aplicação de filtro de mediana. Ela é uma aplicação de processamento de imagem que pode ser executada em paralelo pela divisão da imagem em fragmentos e em seguida com a fusão das partes resultantes novamente em uma única imagem. O filtro de mediana substitui o valor de um pixel pela mediana dos valores da vizinhança circundante.

7.2 Plataforma de experimentação

A plataforma de experimentação criada para execução dos experimentos é composta pela infraestrutura física, que inclui os servidores, os nós e os enlaces físicos; um *middleware* para execução de serviços em nuvem; um sistema de gerência de *workflow*; e um conjunto de software para criação e controle de redes virtuais [Senna et al., 2011b].

O processamento é realizado pelos quatro servidores: *apolo*, *artemis*, *nix* e *hermes*, cujas características são apresentadas na Tabela 7.1. Nessas máquinas foi instalado o Globus Toolkit[1] versão 4, um software de código aberto para a criação de grades compu-

[1]http://www.globus.org/toolkit

tacionais orientadas a serviços. Também foi instalado o sistema de gerência de *workflow* para controlar e monitorar os *workflows* em execução na nuvem.

Tabela 7.1: Configurações dos servidores da nuvem.

Nome	Processador	Frequência	Núcleos	Memória
apolo	Pentium 4	3.00GHz	2	2.5GB
nix	Core2 Quad Q6700	2.66GHz	4	8GB
hermes	Core2 Quad Q6700	2.66GHz	4	8GB
artemis	Xeon 3040	1.86GHz	2	1GB

O núcleo da rede contém duas máquinas: *zeus* e *dionisio*, que estão interconectados por dois enlaces físicos, com taxas de transmissão de 100Mbps e de 1Gbps respectivamente. Essas máquinas utilizam uma plataforma de virtualização para a criação de máquinas virtuais que executam o encaminhamento de pacotes e funcionam como roteadores virtuais. Foram criadas três redes virtuais, compostas por dois roteadores virtuais, cada uma interligando *apolo* a um dos demais servidores: *artemis*, *nix* e *hermes*. A montagem dessa plataforma de experimentação é representada na Figura 7.3.

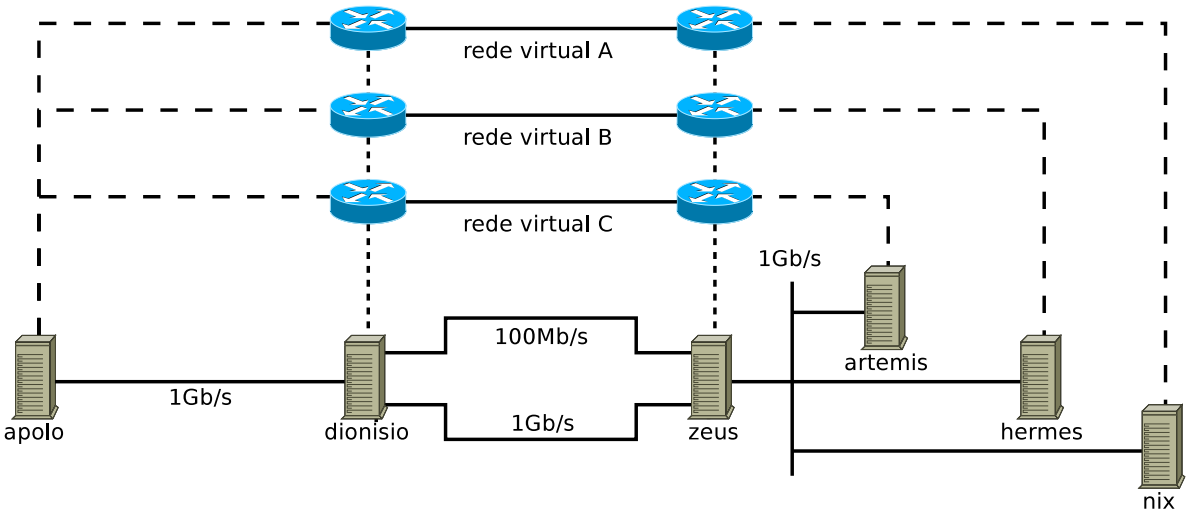


Figura 7.3: Montagem da plataforma de experimentação para o estudo de caso.

7.3 Resultados experimentais

Foram realizadas execuções de *workflows* na plataforma de experimentação para avaliar o desempenho da rede em três aspectos:

1. A sobrecarga envolvida quando roteadores virtuais são utilizados.
2. Como os enlaces da rede virtual se comportam com transmissões de fluxo de dados concorrentes.
3. Como as adaptações providas pelo sistema de gerência de redes virtuais pode beneficiar a execução de *workflows*.

7.3.1 Impacto das redes virtuais na execução de um *workflow*

A primeira parte dos experimentos visa avaliar o desempenho das redes virtuais criadas para transportar os dados entre os servidores. Um *workflow* simples que realiza o filtro de mediana em uma imagem foi executado utilizando cada uma das redes virtuais. O *workflow* utiliza três serviços e realiza duas transmissões de dados como mostra a Figura 7.4.

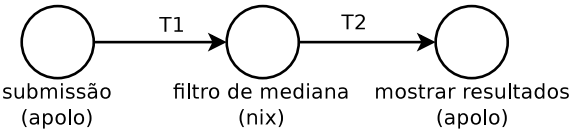


Figura 7.4: *Workflow* que aplica o filtro de mediana em uma imagem.

O *workflow* recebe uma submissão do usuário em *apolo* e envia a imagem para ser processada em *nix* (T1), onde o filtro de mediana é aplicado. Em seguida, a imagem é transferida de volta a *apolo* (T2), onde a imagem resultante é apresentada ao usuário. O *workflow* de filtro de mediana foi executado em imagens de 10.000 × 10.000 *pixels*, com 287MB. Foram comparadas as execuções na plataforma de experimentação com as redes virtuais utilizando o enlace de 1Gbps contra a execução que utiliza um comutador Giga Ethernet conectando os servidores. A Figura 7.5 apresenta a média de 5 execuções dos tempos de execução de cada passo do *workflow* com um intervalo de confiança de 95%.

Na Figura 7.5, é possível observar que o tempo das transferências T1 e T2 dobram quando são realizados através de uma rede virtual e o tempo de execução do filtro de mediana não é impactado por elas. Isso causa um impacto no tempo total de execução do *workflow*, que tem um aumento de 23%. Todas as redes virtuais apresentam um comportamento similar em relação à sobrecarga que elas geram no transporte de dados em relação a uma comunicação direta através de um comutador Giga Ethernet. É importante salientar que os serviços invocados durante a execução dos *workflows* foram executados em máquinas físicas.

A sobrecarga introduzida pelos roteadores virtuais nas transferências dos arquivos do *workflow* é causada principalmente pelo fato do encaminhamento dos dados nas redes virtuais serem executados pelos roteadores virtuais. O sistema operacional da máquina

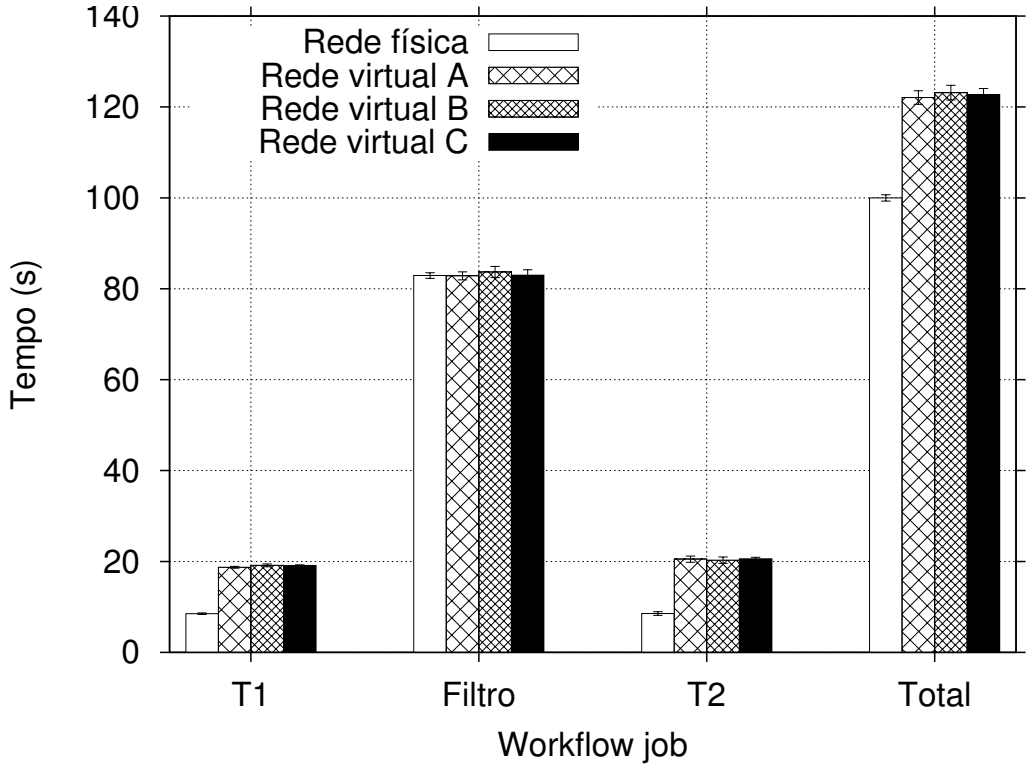


Figura 7.5: Avaliação de desempenho da aplicação utilizando redes virtuais em comparação com uma rede física.

física precisa fazer uma troca de contexto para o roteador virtual a cada pacote que é encaminhado. Os comutadores internos, que encaminham pacotes entre as interfaces de rede físicas e virtuais, também causam uma degradação do desempenho. Além disso, nesse experimento, o encaminhamento feito através das redes virtuais é comparado com o encaminhamento feito por um comutador Ethernet, que é realizado em hardware. Por fim, há um número maior de saltos nas redes virtuais, passando pelos roteadores *zeus* e *dionisio*, o que aumenta os tempos de processamento, transmissão e de filas.

O objetivo desse capítulo é mostrar o impacto da gerência das redes virtuais na execução de aplicações e não o desempenho da rede virtual em si. Essa primeira avaliação foi realizada para servir de base para os próximos experimentos, que mostrarão como a flexibilidade das redes virtuais e sua capacidade de migração na rede de substrato podem contribuir para a obtenção de um melhor desempenho das aplicações.

7.3.2 Redes virtuais concorrentes

Na seção anterior, não havia concorrência entre as redes virtuais, pois apenas uma delas foi executada por vez. Nesta seção, fluxos TCP passando pelas redes virtuais foram colocados

em execução ao mesmo tempo. Cada fluxo de dados corresponde a uma transferência de um arquivo de imagem de 15.000 × 15.000 *pixels*, com 644MB. As redes virtuais tiveram que concorrer pelos recursos dos enlaces físicos de 1Gbps e de 100Mbps. Foram utilizados quatro cenários (A, B, C e D) na avaliação, como mostra a Tabela 7.2.

Tabela 7.2: Cenários utilizados para avaliar o desempenho de redes virtuais concorrentes.

	100 Mbps			1 Gbps		
	Fluxo 1	Fluxo 2	Fluxo 3	Fluxo 1	Fluxo 2	Fluxo 3
A	×	×	×			
B		×	×	×		
C			×	×	×	
D				×	×	×

Cada cenário possui uma combinação diferente do mapeamento das redes virtuais sobre a rede de substrato. No cenário A, as três redes virtuais estão mapeadas sobre o enlace de 100Mbps. No cenário B, duas redes virtuais estão mapeadas no enlace de 100Mbps e uma no enlace de 1Gbps. No cenário C, uma rede virtual está mapeada no enlace de 100Mbps e duas no enlace de 1Gbps. E no cenário D, as três redes virtuais estão mapeadas no enlace de 1Gbps. O fluxos de dados concorrentes são descritos a seguir:

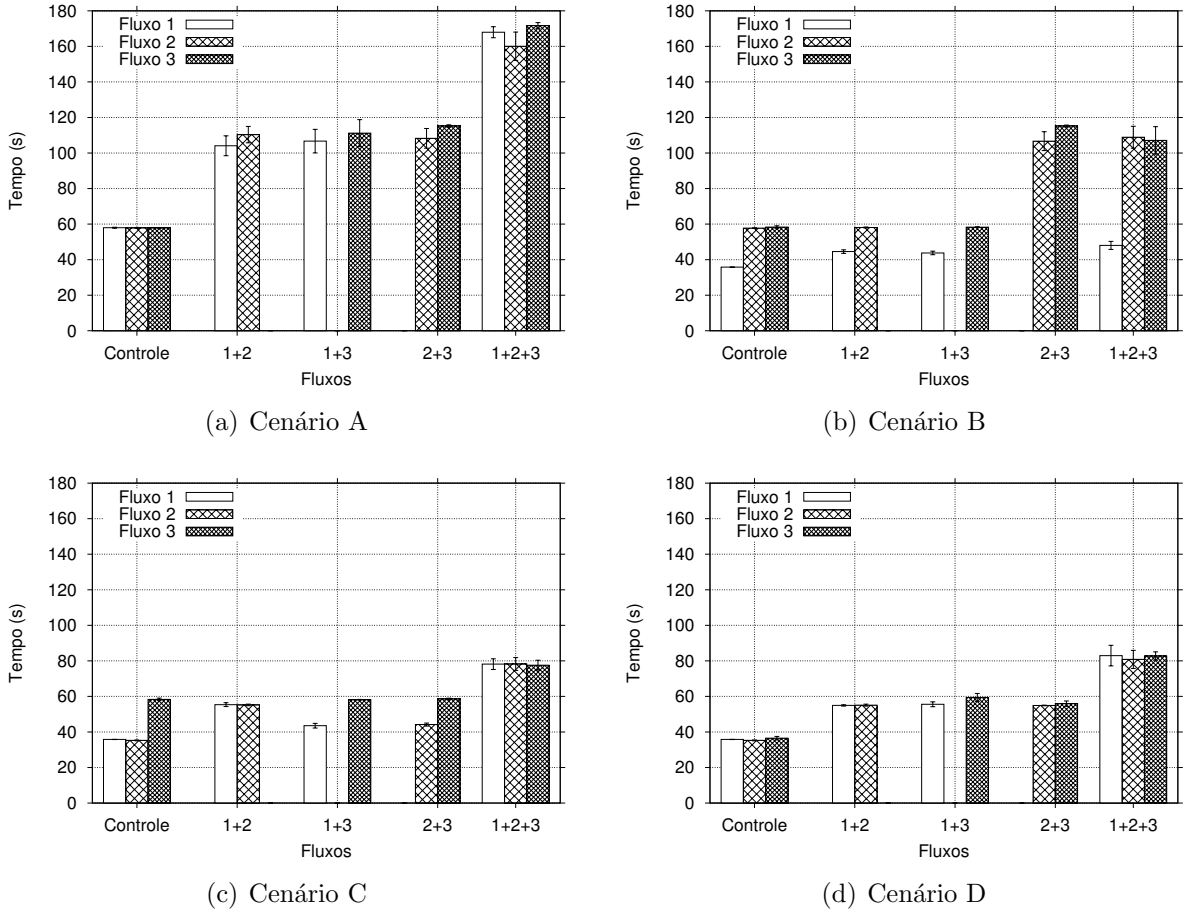
Fluxo 1: Transferência de dados de *nix* para *apolo* utilizando a rede virtual A;

Fluxo 2: Transferência de dados de *hermes* para *apolo* utilizando a rede virtual B;

Fluxo 3: Transferência de dados de *artemis* para *apolo* utilizando a rede virtual C.

A Figura 7.6 apresenta os resultados deste experimento. Os valores de "Controle" são os tempos de transferência cada fluxo separadamente, sem concorrência com os demais. No cenário A (Figura 7.6(a)), é possível observar que o tempo de transmissão dos fluxos 1 e 2 de maneira concorrente fica um pouco abaixo do dobro do tempo de controle. O mesmo acontece quando apenas os fluxos 1 e 3 e apenas os fluxos 2 e 3 são transmitidos. Quando os três fluxos são transmitidos ao mesmo tempo, todos eles têm o desempenho significativamente degradado pela concorrência do enlace.

No cenário B (Figura 7.6(b)), o valor da medição de controle do fluxo 1 cai, pois neste cenário ele utiliza o enlace de 1Gbps. O fluxo 1 tem um tempo de transferência similar para todas as combinações de fluxo, uma vez que ele sempre utiliza o enlace de 1Gbps sozinho. Entretanto, é possível observar um aumento em função número de fluxos do cenário devido à concorrência nas máquinas físicas, onde os roteadores virtuais estão instanciados. Os fluxos 2 e 3 também se comportam de maneira similar ao tempo de



(a) Cenário A

(b) Cenário B

(c) Cenário C

(d) Cenário D

Figura 7.6: Avaliação de desempenho das redes virtuais concorrendo pelos enlaces físicos de 1Gbps e 100Mbps.

controle quando transmitidos apenas com o fluxo 1 pelo mesmo motivo. Quando os fluxos 2 e 3 são transmitidos juntos, eles compartilham o enlace de 100Mbps, o que causa uma degradação no seu desempenho. Comparando os cenários A e B é possível notar que o encaminhamento do fluxo 1 por um enlace separado beneficia todos os fluxos.

No cenário C (Figura 7.6(c)), observa-se que o fluxo 3 não é afetado pelos fluxos 1 e 2, já que o fluxo 3 é o único no enlace de 100Mbps, exceto no caso em que todos os fluxos são enviados juntos. Isso mostra que a concorrência pelas máquinas físicas onde os roteadores virtuais estão instanciados também é um fator limitante para o desempenho global das redes virtuais. Quando os três fluxos são transmitidos ao mesmo tempo, os fluxos 1 e 2 concorrem entre si pelo enlace de 1Gbps, fazendo com que seu desempenho seja próximo ao do fluxo 3 sozinho no enlace de 100Mbps.

No cenário D (Figura 7.6(d)), a combinação de 2 fluxos resulta em um tempo de transferência maior quando comparado com o valor da medição de controle. Quanto

todos os fluxos são transmitidos ao mesmo tempo, a concorrência pelo enlace causa um tempo de transmissão ainda maior. Entretanto, é importante notar que os tempos de transmissão de todos os fluxos juntos é menor que a soma dos tempos de transmissão de todos os fluxos sozinhos. Além disso, o tempo total de transmissão dos três fluxos concorrentemente no cenário D é menor em relação aos cenários A e B, e similar quando comparado ao cenário C. Portanto, os cenários C e D são opções válidas para a transmissão dos três fluxos de maneira mais rápida na nossa plataforma de experimentação. Esses resultados contribuem para o desenvolvimento de estratégias de adaptação para execução de *workflows* sobre redes virtuais.

7.3.3 Adaptação das redes virtuais

Nesta seção será mostrado como o sistema de gerência de redes virtuais pode adaptar os enlaces virtuais durante a execução dos *workflows* para melhorar o seu desempenho. Serão utilizados os mesmos cenários da seção anterior para a distribuição dos fluxos sobre os enlaces físicos.

Adaptação durante a execução de um *workflow*

O primeiro experimento dessa seção utiliza um *workflow* que realiza o filtro de mediana dividindo uma imagem de 15.000 × 15.000 *pixels*, com 644MB, em três fragmentos, envia-os em paralelo para processamento em diferentes recursos, e recebe os fragmentos processados de volta para gerar a imagem resultante final (Figura 7.7). Os passos da execução do *workflow* são os seguintes:

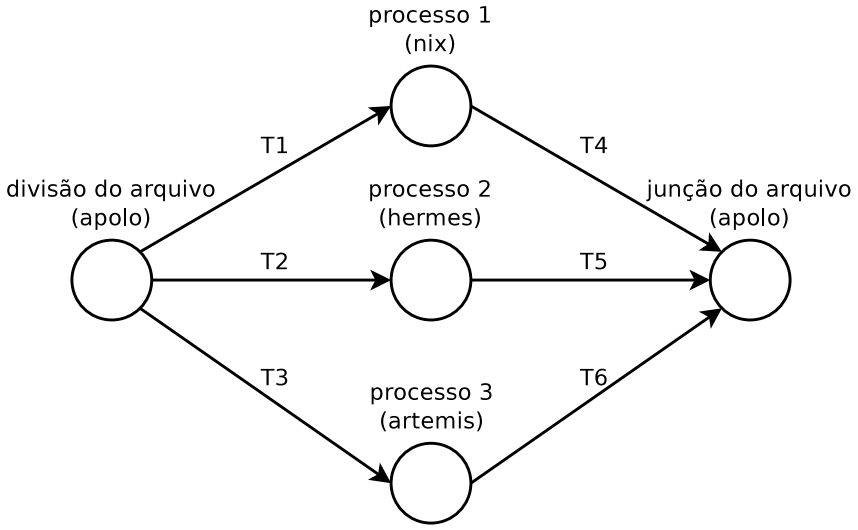


Figura 7.7: *Workflow* que aplica o filtro de mediana em paralelo.

1. *apolo* divide o arquivo da imagem em três fragmentos.

2. *apolo* transfere em paralelo um fragmento para *nix* através da rede virtual A, um fragmento para *hermes* através da rede virtual B, e um fragmento para *artemis* através da rede virtual C. Nessas transferências todos os fluxos são encaminhados pelo enlace de 100Mbps.

3. *nix*, *hermes*, and *artemis* executam o filtro de mediana sobre seus fragmentos do arquivo de imagem. Nesse instante o sistema de gerência realiza as adaptações nas redes virtuais.

4. *apolo* recebe todos os fragmentos processados de *nix*, *hermes* e *artemis* e junta-os. Os fluxos são encaminhados por diferentes caminhos, usando os cenários da seção anterior.

Durante a execução do *workflow*, o sistema de gerência de redes virtuais pode distribuir os fluxos nos caminhos disponíveis de acordo com os requisitos do *workflow*. Por exemplo, se os tempos de execução das primeiras três transferências através do enlace de 100Mbps estiverem abaixo dos requisitos da aplicação (discriminados no seu SLA), o sistema de gerência de redes virtuais pode migrar os enlaces virtuais para o enlace físico de 1Gbps.

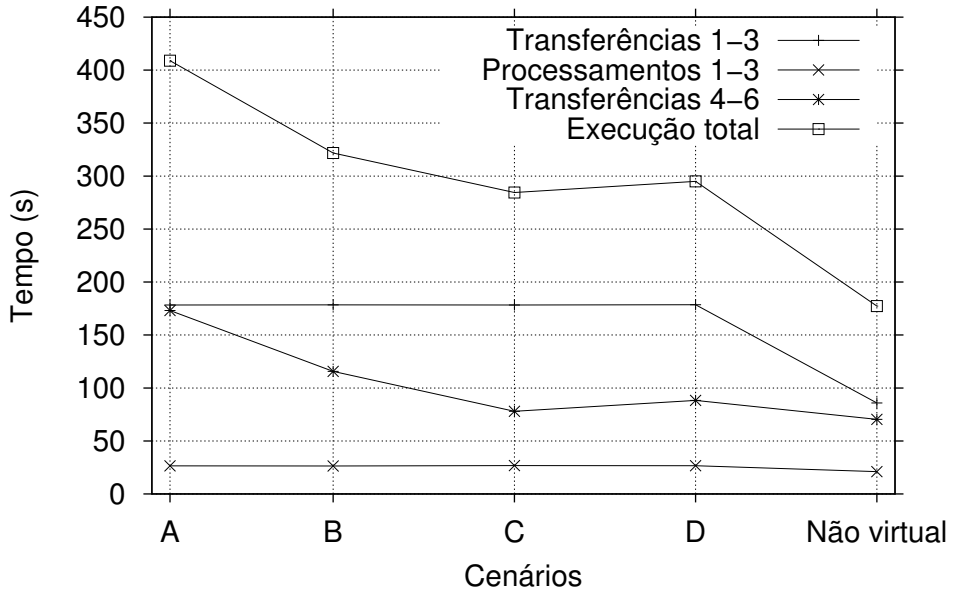


Figura 7.8: Avaliação de desempenho da aplicação em função de adaptações das redes virtuais durante a execução do *workflow*.

A Figura 7.8 mostra os ganhos potenciais desse tipo de adaptação na execução do *workflow*. Pode-se observar que quando o sistema de gerência de redes virtuais migra do

cenário A para o B, ele melhora os tempos de transferência no retorno dos fragmentos do arquivo de imagem para *apolo* (transferências 4-6). Como consequência, o tempo de execução do *workflow* também é reduzido. Se os requisitos forem mais rígidos, o sistema de gerência pode adaptar para os cenários C ou D, atingindo tempos de transferência de dados menores.

Workflows concorrentes

Neste estudo de caso serão apresentados os resultados da execução de *workflows* concorrentes com níveis de prioridade: (i) o *workflow* de menor prioridade W_1 da Figura 7.9, para o processamento em paralelo de uma imagem de 20.000 × 20.000 *pixels*, com 1.145MB; e (ii) o workflow de maior prioridade W_2 da Figura 7.4, para a aplicação do filtro de mediana em uma imagem de 10.000 × 10.000 *pixels*, com 287MB.

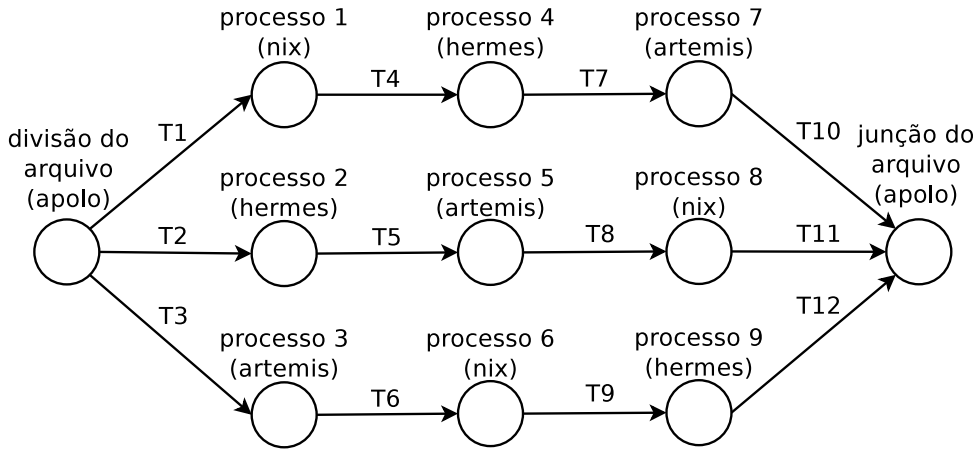


Figura 7.9: *Workflow* que aplica uma sequência de filtros em paralelo.

No experimento, W_1 começa sua execução, dividindo o arquivo de imagem e enviando os fragmentos para *nix*, *hermes* e *artemis*, através das redes virtuais A, B e C mapeadas no enlace físico de 1Gbps. Depois do processamento, W_2 começa sua execução enviando o arquivo de *apolo* para *hermes*, utilizando uma nova rede virtual também mapeada no enlace físico de 1Gbps. Nesse instante, quatro fluxos de dados concorrerão pelo enlace de 1Gbps entre *zeus* e *dionisio*: T4, T5 e T6 de W_1 e T1 de W_2. O GPO requisita uma adaptação através da interface com a gerência de redes virtuais para dar prioridade a W_2. O objetivo do sistema de gerência de redes virtuais é satisfazer os requisitos do W_2 que possui prioridade mais alta. Para alcançar isso, ele deve reconfigurar as redes virtuais para prover desempenho melhor à W_2.

Foram analisadas três ações possíveis a serem tomadas pelo sistema de gerência de redes virtuais:

Ação 1: Não realizar nenhuma adaptação. Deixar que todos os fluxos sejam encaminhados através do enlace de 1Gbps.

Ação 2: Reconfigurar as redes virtuais para que os fluxos de W_2 utilizem exclusivamente o enlace de 100Mbps.

Ação 3: Reconfigurar as redes virtuais para que os fluxos de W_2 utilizem exclusivamente o enlace de 1Gbps, i.e., migrar as redes virtuais A, B e C para o enlace físico de 100Mbps.

Os resultados das transferência de dados da execução concorrente dos *workflows* são apresentados na Figura 7.10. No caso da Ação 1, os tempos totais de execução dos *workflows* (incluindo o tempo de processamento que não é mostrado na figura por motivo de clareza) são de 206,038s para W_1 e 70,605s para W_2. A Ação 2 foi a pior opção para o *workflow* prioritário, uma vez que ela tem como consequência um aumento dos tempos de suas transferências de dados (T1 e T2 de W_2). Nesse caso, os tempos totais de execução para W_1 e W_2 são 216,825s e 81,368s, respectivamente. Por outro lado, quando a Ação 3 é tomada, os tempos de transferência de dados do *worfklow* prioritário W_2 são menores, fazendo o seu tempo de execução total cair para 50,598s.

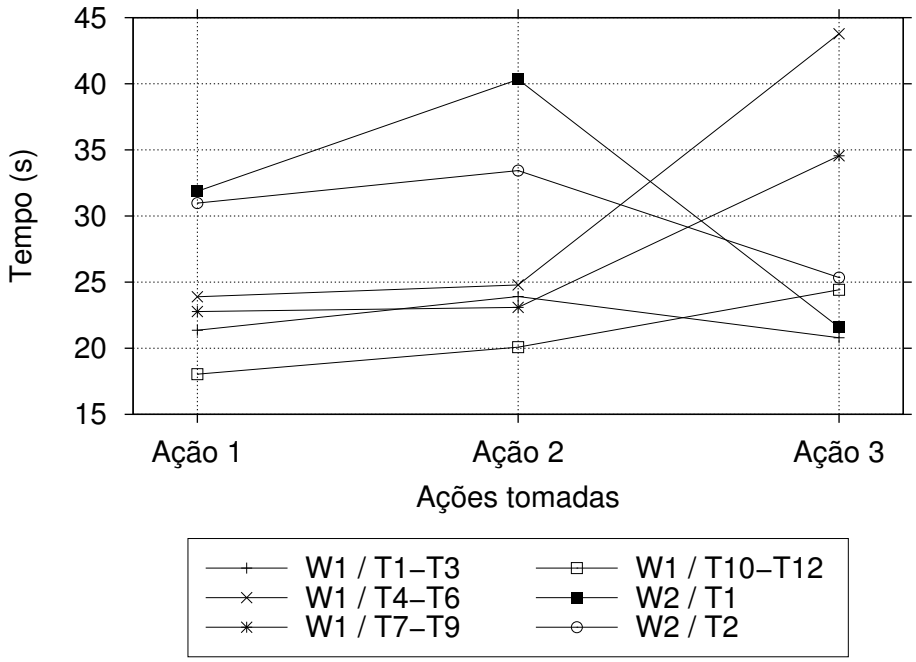


Figura 7.10: Avaliação de desempenho das transferências de dados dos *workflows* concorrentes em função das adaptações nas redes virtuais.

Esse estudo de caso mostra que, quando o *workflow* prioritário W_2 é colocado em execução na plataforma de experimentação, a melhor opção é encaminhar seus fluxos pelo enlace de 1Gbps sem concorrência. Entretanto, como uma segundo opção, encaminhar seus fluxos através do enlace de 1Gbps junto com os três fluxos do workflow W_1 pode ser melhor do que encaminhá-los sozinho pelo enlace de 100Mbps.

7.4 Análise do estudo de caso

Neste capítulo, diversos experimentos envolvendo a execução de *workflows* sobre redes virtuais foram realizados. Apesar desses experimentos serem bastante simples eles mostram como a gerência de redes virtuais pode contribuir para a execução de aplicações em uma nuvem. Os resultados obtidos neste estudo de caso podem servir de base para a definição de uma solução para redes virtuais cientes de serviço. Há um grande dinamismo nas aplicações em execução em uma nuvem, pois além da sua entrada e saída, elas se expandem e se contraem, consumindo uma quantidade variável de recursos ao longo do tempo. Nesse cenário o desempenho das aplicações pode cair devido a uma má distribuição dos recursos de rede.

A gerência das redes virtuais realiza migrações dos recursos virtuais visando melhorar o desempenho das aplicações. Um algoritmo de remapeamento de redes virtuais poderia ser utilizado, mas isso pode gerar problemas. Primeiramente, quanto mais adaptações tiverem que ser feitas, maior será o impacto das migrações para as aplicações. Em segundo lugar, isso poderia requerer um controle centralizado, que vai contra o que propusemos neste trabalho, uma arquitetura distribuída para a gerência das redes virtuais, para obter uma maior escalabilidade e robustez. Adaptações simples como as apresentadas aqui, mais localizadas, que privilegiam a migração de enlaces virtuais ao invés de nós virtuais, podem melhorar satisfatoriamente o desempenho das aplicações.

Capítulo 8

Conclusão

Duas novas tecnologias foram combinadas para o desenvolvimento de um novo modelo para a Internet do futuro neste trabalho. A primeira, das redes virtuais, contribui com o isolamento das comunicações de diferentes organizações e com a implantação de diversas arquiteturas de redes sobre a mesma infraestrutura física, de maneira a oferecer um serviço de transporte de dados específico para seu tipo de aplicação. A segunda, da gerência autonômica, contribui para um novo tipo de gerência de sistemas e também de redes de computadores com pouca ou nenhuma intervenção humana. Isso torna o sistema mais simples de administrar e de utilizar, pois funções menos complexas de configuração, otimização, cura e proteção são realizadas pelo próprio sistema.

Com as redes virtuais é criada uma separação da gerência em dois níveis. No nível de serviços, os protocolos das redes virtuais precisam ser gerenciados para operarem de maneira otimizada. No nível de infraestrutura, a gerência é responsável pela alocação de recursos físicos às redes virtuais. É neste nível, do provedor de infraestrutura, que a gerência autonômica de redes virtuais proposta neste trabalho opera. Ela utiliza os conceitos da computação autonômica: a gerência distribuída, laço de controle autonômico e base de conhecimento, para atingir seus objetivos. Esses conceitos foram implementados e avaliados na prática, com a implementação de um protótipo real, baseado na arquitetura multiagente desenvolvida, sobre uma plataforma de experimentação que utiliza as tecnologias atuais de virtualização de redes.

Os resultados experimentais mostraram nos cenários estudados o desempenho da função de autocura de redes virtuais e a eficiência na alocação de recursos dos algoritmos que atuam no planejamento das ações corretivas. A experimentação serve também para validar a arquitetura do sistema de gerência autonômica de redes virtuais com o modelo de informação proposto. Os resultados foram favoráveis ao mecanismo de cópia da memória virtual para recuperação de roteadores virtuais, especialmente em redes virtuais que utilizam algoritmos de roteamento dinâmico, e a uma escolha criteriosa das ações

corretivas a serem tomadas na presença de falhas para que a alocação de recursos não seja degradada no processo de autocura de redes virtuais.

Também foi estudada a integração da gerência das redes virtuais com a gerência de aplicações executadas em uma nuvem computacional. Nesse estudo de caso, foi mostrado como a flexibilidade das redes virtuais pode contribuir para um melhor desempenho das aplicações. Essa gerência de redes virtuais cientes de serviços requer mais desenvolvimento com a definição das interfaces de comunicação entre os módulos. A implementação do protótipo da arquitetura completa e a avaliação do sistema em cenários maiores, mais próximos da realidade das nuvens computacionais, será um dos trabalhos futuros propostos. Outro trabalho futuro é o detalhamento das funções de autoconfiguração, autootimização, autoproteção dentro da arquitetura de gerência autonômica de redes virtuais, assim como foi feita com a função de autocura neste trabalho.

Referências Bibliográficas

G. Alkmim, D. Batista, e N. Fonseca. Optimal Mapping of Virtual Networks. In *IEEE Global Telecommunications Conference (GLOBECOM)*, pp. 1–6, 2011.

T. Anderson, L. Peterson, S. Shenker, e J. Turner. Overcoming the Internet impasse through virtualization. *Computer*, 38(4):34–41, 2005.

P. Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, R. Neugebauer, I. Pratt, e A. Warfield. Xen and the art of virtualization. *SIGOPS Operating Systems Review*, 37:164–177, 2003.

F. Bellard. QEMU, a fast and portable dynamic translator. In *USENIX Annual Technical Conference (ATEC)*, pp. 41–46, Berkeley, EUA, 2005. USENIX Association.

M. Boucadair, P. Georgatsos, N. Wang, D. Griffin, G. Pavlou, M. Howarth, e A. Elizondo. The AGAVE approach for network virtualization: differentiated services delivery. *Annals of Telecommunications*, 64:277–288, 2009.

M. Bourguiba, K. Haddadou, e G. Pujolle. Evaluating and Enhancing Xen-Based Virtual Routers to Support Real-Time Applications. In *7th IEEE Consumer Communications and Networking Conference (CCNC)*, pp. 1–5, 2010.

T. Braga, F. Silva, L. Ruiz, e H. Assunção. Redes autonômicas. In *Minicursos do Simpósio Brasileiro de Redes de Computadores (SBRC)*, pp. 173–207, 2006.

A. Carissimi. *Minicursos do Simpósio Brasileiro de Redes de Computadores (SBRC)*, chapter Virtualização: da teoria a soluções, pp. 173–207. 2008.

Y. Cheng, R. Farha, M. Kim, A. Leon-Garcia, e J. Hong. A generic architecture for autonomic service and network management. *Computer Communications*, 29(18):3691–3709, 2006.

N. Chowdhury e R. Boutaba. Network virtualization: state of the art and research challenges. *IEEE Communications Magazine*, 47(7):20–26, 2009.

N. Chowdhury, M. Rahman, e R. Boutaba. Virtual Network Embedding with Coordinated Node and Link Mapping. In *IEEE INFOCOM 2009*, pp. 783–791, 2009.

B. Chun, D. Culler, T. Roscoe, A. Bavier, L. Peterson, M. Wawrzoniak, e M. Bowman. PlanetLab: an overlay testbed for broad-coverage services. *SIGCOMM Computer Communication Review*, 33(3):3–12, 2003.

D. Clark, C. Partridge, J. Ramming, e J. Wroclawski. A knowledge plane for the internet. In *Conference on applications, technologies, architectures, and protocols for computer communications*, pp. 3–10, New York, EUA, 2003. ACM.

D. Coulson, D. Berrange, D. Veillard, C. Lalancette, L. Stump, e D. Jorm. Libvirt 0.7.5: Application Development Guide, 2010. `http://libvirt.org/guide/pdf/Application_Development_Guide.pdf`.

R. Couto, M. Campista, e L. COSTA. XTC: Um Controlador de Vazão para Roteadores Virtuais Baseados em Xen. In *XXIX Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos (SBRC)*, pp. 17–30, 2011.

S. Davy, C. Fahy, L. Griffin, Z. Boudjemil, A. Berl, A. Fischer, H. de Meer, e J. Strassner. Towards a Policy-based Autonomic Virtual Network to support Differentiated Security Services. In *International Conference on Telecommunications & Multimedia (TEMU)*, pp. 1–8, 2008.

S. Davy, J. Serrat, A. Astorga, B. .Jenning, e J. Rubio-Loyola. Policy-Assisted Planning and Deployment of Virtual Networks. In *7th International Conference on Network and Service Management (CNSM)*, pp. 1–8, 2011.

S. Dobson, S. Denazis, A. Fernández, D. Gaïti, E. Gelenbe, F. Massacci, P. Nixon, F. Saffre, N. Schmidt, e F. Zambonelli. A survey of autonomic communications. *ACM Transactions on Autonomous and Adaptive Systems*, 1:223–259, 2006.

I. Fajjari, M. Ayari, e G. Pujolle. VN-SLA: A Virtual Network Specification Schema for Virtual Network Provisioning. In *Ninth International Conference on Networks (ICN)*, pp. 337–342, 2010.

I. Fajjari, N. Aitsaadi, G. Pujolle, e H. Zimmermann. VNE-AC: Virtual Network Embedding Algorithm Based on Ant Colony Metaheuristic. In *IEEE International Conference on Communications (ICC)*, pp. 1–6, 2011a.

I. Fajjari, M. Ayari, O. Braham, G. Pujolle, e H. Zimmermann. Towards an Autonomic Piloting Virtual Network Architecture. In *4th IFIP International Conference on New Technologies, Mobility and Security (NTMS)*, pp. 1–5, 2011b.

N. Feamster, L. Gao, e J. Rexford. How to lease the internet in your spare time. *SIGCOMM Computer Communication Review*, 37:61–64, 2007.

N. Fernandes e O. Duarte. Provendo Isolamento e Qualidade de Serviço em Redes Virtuais. In *XXIX Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos (SBRC)*, pp. 295–308, 2011.

N. Fernandes, M. Moreira, I. Moraes, L. Ferraz, R. Couto, H. Carvalho, M. Campista, L. Costa, e O. Duarte. Virtual networks: isolation, performance, and trends. *Annals of Telecommunications*, 66:339–355, 2011.

A. Galis, H. Abramowicz, M. Brunner, D. Raz, P. Chemouil, J. Butler, C. Polychronopoulos, S. Clayman, H. de Meer, T. Coupaye, A. Pras, K. Sabnani, P. Massonet, e S. Naqvi. Management and service-aware networking architectures (MANA) for future Internet: System functions, capabilities and requirements. In *Fourth International Conference on Communications and Networking in China (ChinaCOM)*, pp. 1–13, 2009.

A. Ganek e T. Corbi. The dawning of the autonomic computing era. *IBM Systems Journal*, 42(1):5–18, 2003.

Ginkgo Networks. Ginkgo Distributed Network Piloting System. White Paper, 2008. `http://www.ginkgo-networks.com/IMG/pdf/WP_Ginkgo_DNPS_v1_1.pdf`.

N. Gude, T. Koponen, J. Pettit, B. Pfaff, M. Casado, N. McKeown, e S. Shenker. NOX: towards an operating system for networks. *SIGCOMM Computer Communication Review*, 38:105–110, 2008.

J. He, R. Zhang-Shen, Y. Li, C.-Y. Lee, J. Rexford, e M. Chiang. DaVinci: dynamically adaptive virtual networks for a customized internet. In *ACM CoNEXT Conference*, pp. 15:1–15:12, New York, EUA, 2008. ACM.

I. Houidi, W. Louati, e D. Zeghlache. A Distributed Virtual Network Mapping Algorithm. In *IEEE International Conference on Communications (ICC)*, pp. 5634–5640, 2008.

I. Houidi, W. Louati, D. Zeghlache, e S. Baucke. Virtual Resource Description and Clustering for Virtual Network Discovery. In *IEEE International Conference on Communications Workshops (ICC Workshops)*, pp. 1–6, 2009.

I. Houidi, W. Louati, D. Zeghlache, P. Papadimitriou, e L. Mathy. Adaptive virtual network provisioning. In *Second ACM SIGCOMM Workshop on Virtualized Infrastructure Systems and Architectures (VISA)*, pp. 41–48, New York, EUA, 2010. ACM.

IBM. Autonomic Computing: IBM's Perspective on the State of Information Technology, 2001. `http://www.research.ibm.com/autonomic/manifesto/autonomic_computing.pdf`.

K. Ishiguro. Quagga: A routing software package for TCP/IP networks, 2006. `http://www.quagga.net/docs/quagga.pdf`.

R. Jain. Internet 3.0: Ten Problems with Current Internet Architecture and Solutions for the Next Generation. In *IEEE Military Communications Conference (MILCOM)*, pp. 1–9, 2006.

J. Kephart e D. Chess. The Vision of Autonomic Computing. *Computer*, 36:41–50, 2003.

A. Kivity, Y. Kamay, D. Laor, U. Lublin, e A. Liguori. KVM: the Linux Virtual Machine Monitor. *Linux Symposium*, 1:225–230, 2007.

J. Lu e J. Turner. Efficient mapping of virtual networks onto a shared substrate. *Department of Computer Science and Engineering Washington University in St Louis Technical Report (WUCSE)*, 35(2006-35):1–11, 2006.

Q. Mahmoud. *Cognitive Networks: Towards Self-Aware Networks*. Wiley-Interscience, 2007.

C. Marquezan, L. Granville, G. Nunzi, e M. Brunner. Distributed autonomic resource management for network virtualization. In *IEEE Network Operations and Management Symposium (NOMS)*, pp. 463–470, 2010.

D. Mattos, N. Fernandes, V. da Costa, L. Cardoso, M. Campista, L. Costa, e O. Duarte. OMNI: OpenFlow MaNagement Infrastructure. In *2nd IFIP International Conference Network of the Future (NoF)*, pp. 52–56, 2011.

D. Mattos, L. Mauricio, L. Cardoso, I. Alvarenga, L. Ferraz, e O. Duarte. Uma Rede de Testes Interuniversitária a com Técnicas de Virtualização Híbridas. In *XXIX Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos (SBRC)*, pp. 912–919, 2012.

N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, e J. Turner. OpenFlow: enabling innovation in campus networks. *SIGCOMM Computer Communication Review*, 38:69–74, 2008.

P. Molinero-Fernández, N. McKeown, e H. Zhang. Is IP going to take over the world (of communications)? *SIGCOMM Computer Communication Review*, 33:113–118, 2003.

B. Pfaff, J. Pettit, K. Amidon, M. Casado, T. Koponen, e S. Shenker. Extending Networking into the Virtualization Layer. In *Eighth ACM Workshop on Hot Topics in Networks (HotNets)*, pp. 1–6, 2009.

P. Pisa, R. Couto, H. Carvalho, D. Neto, N. Fernandes, M. Campista, L. Costa, O. Duarte, e G. Pujolle. VNEXT: Virtual NEtwork management for Xen-based Testbeds. In *2nd IFIP International Conference Network of the Future (NoF)*, pp. 41–45, 2011.

G. Pujolle. An Autonomic Architecture for Network Management and Control. *UPGRADE*, 9:36–40, 2008.

J. Rexford e C. Dovrolis. Future Internet architecture: clean-slate versus evolutionary research. *Communications of the ACM*, 53:36–40, 2010.

J. Roberts. The clean-slate approach to future Internet design: a survey of research initiatives. *Annals of Telecommunications*, 64:271–276, 2009.

F. Rodríguez-Haro, F. Freitag, e L. Navarro. Enhancing virtual environments with QoS aware resource management. *Annals of Telecommunications*, 64:289–303, 2009.

S. Sallent, A. Abelém, I. Machado, L. Bergesio, S. Fdida, J. Rezende, S. Azodolmolky, M. Salvador, L. Ciuffo, e L. Tassiulas. FIBRE Project: Brazil and Europe Unite Forces and Testbeds for the Internet of the Future. In T. Korakis, M. Zink, e M. Ott, editors, *Testbeds and Research Infrastructure, Development of Networks and Communities*, volume 44 of *Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering*, pp. 372–372. Springer Berlin Heidelberg, 2012.

C. Senna, L. Bittencourt, e E. Madeira. Execution of service workflows in grid environments. *International Journal of Communication Networks and Distributed Systems (IJCNDS)*, 5(1/2):88–108, 2010.

C. Senna, M. Soares, D. Batista, E. Madeira, e N. Fonseca. Experiments with a Self-Management System for Virtual Networks. In *II Workshop de Pesquisa Experimental da Internet do Futuro (WPEIF)*, pp. 7–10, 2011a.

C. Senna, M. Soares, L. Bittencourt, e E. Madeira. An Architecture for Adaptation of Virtual Networks on Clouds. In *7th Latin American Network Operations and Management Symposium (LANOMS)*, pp. 1–8, 2011b.

R. Sherwood, G. Gibb, K. Yap, G. Appenzeller, N. McKeown, e G. Parulkar. Flowvisor: A network virtualization layer. Technical report, Stanford, 2009.

M. Soares e E. Madeira. Sistema Multiagente para Autogerenciamento Distribuído de Falhas em Redes Virtuais. In *XVI Workshop de Gerência e Operação de Redes e Serviços (WGRS)*, pp. 17–30, 2011.

M. Soares e E. Madeira. A multi-agent architecture for autonomic management of virtual networks. In *IEEE Network Operations and Management Symposium (NOMS)*, pp. 1183–1186, 2012.

J. Strassner, N. Agoulmine, e E. Lehtihet. FOCALE: A Novel Autonomic Networking Architecture. In *Latin American Autonomic Computing Symposium (LAACS)*, pp. 48–60, 2006.

J. Strassner, D. Raymer, e S. Samudrala. Providing Seamless Mobility Using the FOCALE Autonomic Architecture. In Y. Koucheryavy, J. Harju, e A. Sayenko, editors, *Next Generation Teletraffic and Wired/Wireless Advanced Networking*, volume 4712 of *Lecture Notes in Computer Science*, pp. 330–341. Springer Berlin / Heidelberg, 2007.

T. Truong Huu, G. Koslovski, F. Anhalt, J. Montagnat, e P. Vicat-Blanc Primet. Joint Elastic Cloud and Virtual Network Framework for Application Performance-cost Optimization. *Journal of Grid Computing*, 9:27–47, 2011.

J. Turner e D. Taylor. Diversifying the Internet. In *IEEE Global Telecommunications Conference (GLOBECOM)*, volume 2, pp. 755–760, 2005.

S. Vaughan-Nichols. New Approach to Virtualization Is a Lightweight. *Computer*, 39(11): 12–14, 2006.

Y. Wang, E. Keller, B. Biskeborn, J. van der Merwe, e J. Rexford. Virtual routers on the move: live router migration as a network-management primitive. *SIGCOMM Computer Communication Review*, 38:231–242, 2008.

M. Yu, Y. Yi, J. Rexford, e M. Chiang. Rethinking virtual network embedding: substrate support for path splitting and migration. *SIGCOMM Computer Communication Review*, 38:17–29, 2008.

M. Yu, J. Rexford, M. J. Freedman, e J. Wang. Scalable flow-based networking with DIFANE. *SIGCOMM Computer Communication Review*, 41:351–362, 2010.

Y. Zhu e M. Ammar. Algorithms for Assigning Substrate Network Resources to Virtual Network Components. In *25th IEEE International Conference on Computer Communications (INFOCOM)*, pp. 1–12, 2006.

Y. Zhu, R. Zhang-Shen, S. Rangarajan, e J. Rexford. Cabernet: connectivity architecture for better network services. In *ACM CoNEXT Conference*, pp. 64:1–64:6, New York, EUA, 2008. ACM.