Sistema Multiagentes para Autogerenciamento Distribuído de Falhas em Redes Virtuais

Milton A. Soares Jr., Edmundo R. M. Madeira

[1]Instituto de Computação – Universidade Estadual de Campinas (Unicamp)
Campinas – SP – Brasil

milton@lrc.ic.unicamp.br, edmundo@ic.unicamp.br

Abstract. *New proposals for the evolution of the Internet, as diversification of architectures and providers, may increase the complexity of a network whose proper operation is essential, given its importance to society and to global economy. A new theoretical framework has emerged to deal with the complexity of the networks through self-management. In this paper, the concepts of autonomic networks were applied in a virtualized environment through a multi-agent system, and experiments were performed with a focus on self-management failure, or self-healing, of virtual networks.*

Resumo. *Novas propostas para a evolução da Internet, como a diversificação de arquiteturas e provedores, podem aumentar a complexidade de uma rede cujo bom funcionamento é fundamental, dada sua importância para a sociedade e para a economia global. Um novo arcabouço teórico tem surgido para lidar com a complexidade das redes através do autogerenciamento. Neste trabalho, os conceitos das redes autonômicas foram aplicados em um ambiente virtualizado através de um sistema multiagentes e foram realizados experimentos com foco no autogerenciamento de falhas, ou autocura, de redes virtuais.*

1. Introdução

Com o avanço das plataformas de virtualização e o aumento do poder computacional do *hardware* foi possível imaginar o uso de virtualização na Internet para habilitar o pluralismo de arquiteturas e a separação de provedores de serviços e de infraestrutura [Turner and Taylor 2005]. Essa é uma estratégia *clean slate* para a Internet do futuro com o intuito de superar problemas na arquitetura atual [Anderson et al. 2005]. Embora a abordagem *one-size-fits-all* do TCP/IP, que mantem a complexidade nas extremidades da rede, tenha sido responsável pelo sucesso da Internet nos últimos 30 anos, muitos serviços atuais demandam garantias de banda, atraso, *jitter*, segurança, e até mesmo de funcionamento, que a arquitetura não é capaz de oferecer. Com as redes virtuais é possível implementar arquiteturas de rede especializadas para atender a um determinado tipo de serviço sobre uma mesma infraestrutura física.

O plano de gerenciamento tem um papel fundamental nesse cenário. O provedor de infraestrutura precisa gerenciar as redes virtuais tanto para garantir os serviços contratados pelo provedor de serviços, quanto para obter uma boa utilização dos recursos físicos. A segmentação da rede de maneira ótima, robusta e segura é um desafio devido à complexidade do problema [Zhu and Ammar 2006, Yu et al. 2008, Houidi et al. 2008]. As ferramentas de gerenciamento atuais são centralizadas e dependentes de intervenção

humana, o que são fontes de defeitos, e não são capazes de lidar com a heterogeneidade de tecnologias.

Novas abordagens de gerenciamento baseadas em ferramentas da inteligência artificial e dos sistemas cognitivos tem surgido em resposta à crescente complexidade das telecomunicações. Na Internet, é proposta a inclusão de novos planos no modelo de referência para acomodá-las [Clark et al. 2003, Gaïti et al. 2006]. Entre as novas abordagens, a das redes autonômicas [Dobson et al. 2006] propõe lidar com a complexidade habilitando as redes a se autogerenciarem. Tarefas mais simples de configuração, otimização, recuperação de falhas e segurança podem ser realizadas pela própria rede, sem intervenção humana, liberando os administradores para tarefas mais complexas como a definição das políticas e objetivos da rede.

A utilização de máquinas virtuais em *data-centers* para consolidação de servidores, que utilizam técnicas de migração e *backup*, é uma prática comum nos dias atuais. Entretanto, roteadores possuem necessidades específicas que devem ser levadas em consideração. No plano de dados, os roteadores são responsáveis pelo transporte de dados, que é uma aplicação sensível a falhas. Problemas dessa natureza geram perdas que podem afetar os serviços que utilizam esse transporte. No plano de controle, os roteadores são responsáveis pelos algoritmos de roteamento que autoconfiguram as rotas ao custo de alguma latência e sobrecarga.

A proposta deste trabalho é o desenvolvimento de um sistema multiagentes para o autogerenciamento distribuído de redes virtuais. Sua arquitetura é baseada no ciclo autonômico e na base de conhecimento. Nós aplicamos o sistema em um cenário de falhas em que ele deve realizar o diagnóstico e o reparo de maneira autonômica. Também estudamos diferentes mecanismos de recuperação e seu impacto no tempo de reparo e consequentemente nas perdas de pacotes. Este trabalho está inserido no contexto do projeto Horizon [Horizon 2010]. O objetivo do projeto é a definição de uma nova arquitetura para a Internet do futuro baseada no pluralismo de arquiteturas, através de redes virtuais, e no plano de pilotagem, que inclui mecanismos inteligentes para adaptar protocolos a mudanças no ambiente.

Sistemas multiagentes também são utilizados em [Houidi et al. 2010] para manter os contratos e SLAs (*service level agreements*) 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. A função de dissimilaridade também é utilizada para escolher onde serão instanciados os nós virtuais que tiveram problemas. Todas as ações dos agentes são executadas após o diagnóstico da falha. Nós implementamos e testamos situações em que o planejamento é antecipado, através de cálculos e difusões de informações realizados periodicamente. Nós também avaliamos a recuperação de roteadores virtuais através de arquivos que armazenam um estado anterior da memória do roteador para reduzir a latência da inicialização do sistema e da configuração das rotas.

Em [Marquezan et al. 2010] uma arquitetura de gerenciamento distribuída é apresentada 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 ciclo de controle autonômico. Ele monitora os enlaces e busca minimizar a carga de tráfego na rede através da migração de nós virtuais. Nós implementamos o ciclo de controle au-

tonômico nos agentes do sistema de autogerenciamento distribuído para a autocura das redes virtuais.

O restante deste texto foi organizado da seguinte forma: a Seção 2 contém os conceitos básicos de redes virtuais e redes autonômicas que são utilizados neste trabalho. Na Seção 3, a arquitetura do sistema multiagentes desenvolvida neste trabalho é apresentada. Os detalhes de implementação do sistema no *testbed* para a execução dos experimentos são descritos na Seção 4. A Seção 5 apresenta resultados dos experimentos e os discute. A conclusão e os trabalhos futuros são mostrados na Seção 6.

2. Conceitos básicos

Esta seção aborda os conceitos básicos utilizados neste trabalho. O primeiro é o conceito de redes virtuais, as quais o sistema de autogerenciamento distribuído se destina. Outro conceito importante é o de redes autonômicas que possuem os fundamentos para o desenvolvimento do sistema multiagentes.

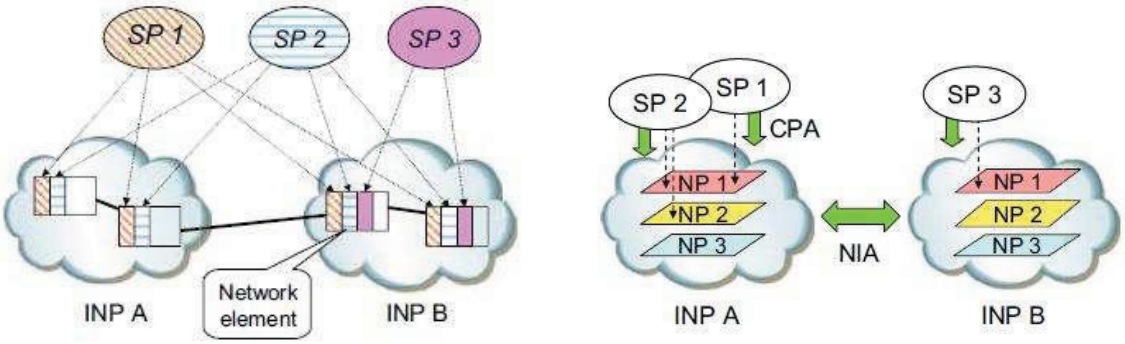
2.1. Redes virtuais

Virtualização é uma técnica empregada em vários cenários de redes de computadores. Nas VLANs (*virtual local area network*) é possível a criação de enlaces virtuais em uma rede local. VPNs (*virtual private network*) são utilizadas para inserir máquinas remotas dentro de uma rede através de um circuito virtual seguro. Com a técnica de tunelamento é possível encaminhar pacotes por redes de diferentes tecnologias. Em todos esses cenários, apenas o enlace está sendo virtualizado. Existem também as redes *overlay* que interligam aplicações e usuários, como as redes P2P (*peer-to-peer*).

As redes virtuais possuem uma abordagem diferente. A ideia por trás delas é a separação lógica dos recursos físicos de uma rede, incluindo dispositivos de comutação. Cada rede virtual pode ter seus próprios protocolos, algoritmos e configurações, de acordo com os objetivos dos serviços que estão sendo executados sobre ela, e deve ter isolamento, isto é, a operação das redes virtuais não deve causar interferência entre si, embora elas estejam sobre a mesma infraestrutura. As redes virtuais podem ser implementadas através de roteadores virtuais ou de separação dos planos de dados e de controle.

No primeiro caso, o roteador físico tem que ser capaz de fornecer e isolar *fatias* de seus recursos aos roteadores virtuais. Ele também tem que multiplexar e demultiplexar pacotes entre o mundo físico e o virtual. Roteadores virtuais podem ser criados via software através de plataformas conhecidas para virtualização de máquinas como o KVM [Habib 2008] e o Xen [Barham et al. 2003]. O KVM transforma o Linux em um *hypervisor* de virtualização total através de um módulo do kernel. Ele é baseado no emulador de máquinas Qemu e roda em arquiteturas x86 com a tecnologia Intel VT ou AMD-V. O Xen é um *hypervisor* de paravirtualização, o que, em geral, oferece um desempenho melhor, mas requer sistemas operacionais modificados ou específicos para ele nas máquinas virtuais.

Na segunda abordagem, há a separação do plano de dados e do plano de controle. Um controlador centralizado é responsável por funções de roteamento, controle de acesso, gerenciamento de tráfego etc. O controlador edita as tabelas de encaminhamento dos comutadores da rede e faz a inspeção dos fluxos que chegam. O OpenFlow [McKeown et al. 2008] é o principal produto dessa abordagem. Ele fornece o *software* do



(a) CABO (b) AGAVE

Figura 1. Propostas para o pluralismo de arquiteturas na Internet do futuro com base na virtualização [Boucadair et al. 2009].

controlador e um protocolo para comunicação segura entre o controlador e os comutadores.

As redes virtuais podem habilitar o pluralismo de arquiteturas na Internet do futuro. Uma das propostas é a do projeto CABO (*concurrent architectures are better then one*) [Feamster et al. 2007]. Ela visa a reestruturação da Internet através da separação de provedores de infraestrutura e de serviços. Nesse contexto, um provedor de serviços contrataria provedores de infraestrutura ao longo da comunicação fim-a-fim para o fornecimento de redes virtuais, que ficariam sob seu controle. Essa proposta está representada na Figura 1(a).

Outra proposta 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*). Os provedores de Internet criariam NPs internamente para atender determinada classe de tráfego, tendo o controle sobre eles. Os serviços com requisitos de QoS semelhantes seriam atendidos por este NP de uma maneira agregada. Através de interconexões horizontais com outros provedores que possuem NPs semelhantes formariam as PIs e, assim, o serviço poderia ser diferenciado de fim-a-fim. A Figura 1(b) apresenta a proposta do projeto AGAVE.

2.2. Redes Autonômicas

O manifesto da computação autonômica [IBM 2001] enfatiza que a complexidade tem sido um grande obstáculo para o desenvolvimento da área de TI, pois ela está crescendo além da habilidade humana de gerenciá-la. A computação autonômica é bio-inspirada no sistema nervoso autônomo que é responsável pela regulação do corpo de acordo com mudanças ambientais sem a necessidade do controle consciente. O objetivo dela é reduzir ou eliminar a intervenção humana no gerenciamento através das propriedades de autoconfiguração, auto-otimização, autocura e autoproteção.

A arquitetura de [Kephart and Chess 2003] é baseada em gerentes autonômicos distribuídos. O gerente autonômico realiza um ciclo de controle sobre o elemento gerenciado e utiliza uma base de conhecimento para armazenar as informações coletadas. A cada laço ele realiza as atividades de monitoramento, análise, planejamento e execução, o que alimenta o próximo ciclo através da base de conhecimento.

O cenário do aumento da complexidade, heterogeneidade, ubiquidade, conectividade e integração é o motivo que leva à necessidade de desenvolvimento de redes autonômicas [Braga et al. 2006]. As redes autonômicas são baseadas nos princípios da computação autonômica. Elas devem ser capazes de realizar o autogerenciamento a partir de políticas de alto-nível definidas pelos administradores ou inferidas através do conhecimento da aplicação. Essa segunda abordagem merece destaque pela propriedade de autoconhecimento baseado em técnicas de aprendizagem de máquina e ciência do contexto.

Para o autogerenciamento é preciso que a rede autonômica seja capaz de se autoconfigurar, auto-otimizar, autocurar e autodefender. Essas propriedades podem ser obtidas com a inclusão de gerentes autonômicos nos elementos de rede para executarem essas tarefas. O sistema de autogerenciamento deve ser distribuído, com cada gerente autonômico responsável por uma parte dos recursos da rede que é o seu elemento gerenciado. Isso evita a criação de um ponto único de falhas e permite maior escalabilidade.

Os gerentes devem atuar de maneira autonômica, mas compartilhar objetivos comuns. A atuação individual dos gerentes autonômicos sobre seus elementos gerenciados deve propiciar um ciclo autonômico maior para o autogerenciamento da rede de acordo com suas políticas e seus objetivos.

O FOCALE [Strassner 2007] é uma arquitetura autonômica para gerenciamento de redes. Ela também baseia-se em gerentes autonômicos sobre os recursos gerenciados, porém, eles não possuem um ciclo de controle estático. O ciclo pode ser definido pelo agente em tempo de execução, baseado no contexto e nas políticas de alto-nível. A arquitetura também define o uso de uma camada MTBL (*model-based translation layer*) para possibilitar a utilização de equipamentos legados no sistema de gerenciamento. Para a adaptação, a arquitetura utiliza técnicas de aprendizado e cognição para comparar, baseado 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*.

3. Proposta de Arquitetura do Sistema Multiagentes

A modelagem orientada a agentes é um paradigma interessante para o desenvolvimento de sistemas distribuídos de autogerenciamento. Agentes são entidades autônomas que observam o ambiente e atuam sobre ele, podendo ter algum nível de cognição e se comunicar com outros agentes. Eles podem fazer o papel do gerente autonômico de um recurso da rede, seu elemento gerenciado. Portanto, o sistema para autogerenciamento de redes virtuais proposto neste trabalho será baseado em uma arquitetura multiagentes.

Esse sistema deve realizar a recuperação de falhas das redes virtuais. As falhas nos roteadores virtuais podem ocorrer por problemas neles próprios, nos nós físicos ou até mesmo nos enlaces físicos. No primeiro caso, o agente responsável pelo roteador virtual deve diagnosticar e avisar os demais sobre a falha. Nos outros casos, o agente também irá falhar e, portanto, os agentes vizinhos devem diagnosticar a falha.

No sistema multiagentes para autogerenciamento de redes viruais há apenas um tipo de agente que atua nos nós da rede física. Os agentes são responsáveis pelo monitoramento dos recursos dos nós físicos e virtuais: cpu, memória, armazenamento, interfaces

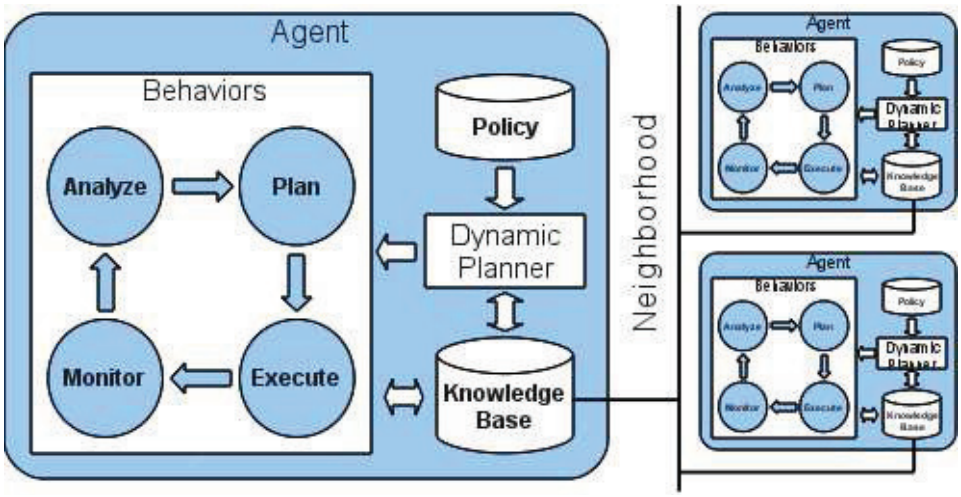


Figura 2. Arquitetura do gerente autonômico.

de rede etc. Eles também devem controlar os roteadores virtuais e instanciá-los em caso de falhas.

A arquitetura dos agentes é baseada nos comportamentos, na base de conhecimento, nas políticas e no DP (*dynamic planner*), como mostra a Figura 2. O sensoriamento, a cognição e a atuação dos agentes são realizados pelos comportamentos. Foram definidos quatro comportamentos: Monitoramento, Análise, Planejamento e Execução, para realizar atividades distintas da tarefa de autogerenciamento de falhas. Nossos agentes executam esses comportamentos periodicamente em sequência formando um ciclo de controle autonômico. Eles estão descritos a seguir:

Monitoramento: Coleta dados dos roteadores virtuais e alimenta a base de conhecimento.

Análise: Realiza o diagnóstico de falhas nos roteadores virtuais ou em nós físicos da vizinhança.

Planejamento: Calcula o custo do nó físico a partir da utilização de seus recursos. Quanto mais ocupado estiver, maior será o custo. E difunde essa informação para os outros agentes.

Execução: Verifica se todos os agentes da vizinhança já enviaram suas informações. Caso sim, o agente do nó físico de menor custo instancia os roteadores virtuais neste.

Nós definimos no arquivo de políticas a ordem de execução e os parâmetros dos comportamentos, como a taxa de execução do laço e o limite máximo de tempo que um agente pode ficar sem difundir informações para que seja considerada uma falha na rede física. O DP é responsável por interpretar o arquivo de políticas, alterar parâmetros dos comportamentos, e controlar o ciclo de vida do agente. Ele também tem acesso à base de conhecimento e pode atuar de acordo com informações contidas nela.

Para que o sistema multiagentes funcione de maneira correta é preciso que os agentes sejam sincronizados para executar algumas de suas ações, por exemplo: a execução do reparo da rede virtual só deve ser feita depois que todos os custos das

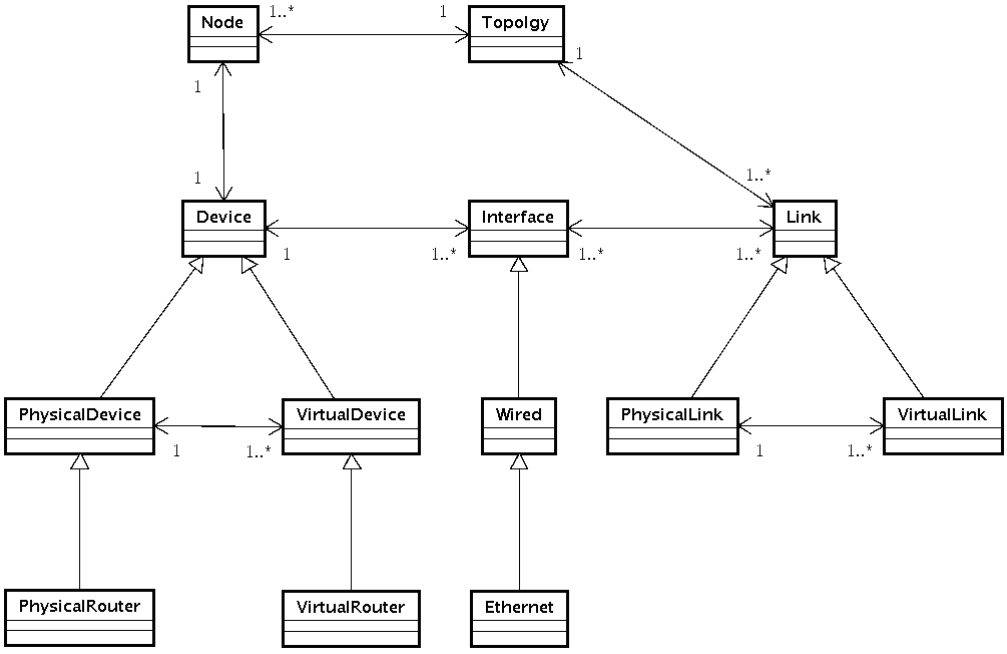


Figura 3. Modelo de informação que descreve a infraestrutura e as redes virtuais.

máquinas foram calculados, para garantir que apenas um agente irá executar a ação de instanciar o roteador virtual. Nós criamos essa sincronização através de informações na base de conhecimento que controlam os comportamentos.

A base de conhecimento serve de repositório para todas as informações do agente, que podem ser coletadas localmente, pelo monitoramento, ou remotamente, através da comunicação com outros agentes. A base de conhecimento possui um modelo de informação comum entre os agentes vizinhos para permitir a comunicação e a interpretação dos dados. O modelo de informação apresentado na Figura 3 representa as topologias das redes virtuais e seus mapeamentos na rede física. Esse modelo foi desenvolvido em trabalhos do projeto Horizon e é baseado em [Fajjari et al. 2010].

Os agentes podem se organizar em vizinhanças, o que limita o escopo de difusão da base de conhecimento. Através dessa comunicação seletiva os agentes formam visões situadas da rede. Um agente pode fazer parte de diversas vizinhanças e difundir diferentes informações para cada uma delas. Na arquitetura do sistema de autogerenciamento de redes virtuais existe apenas uma vizinhança com todos os agentes da rede, e todas a informações geradas pelo agente são difundidas para os vizinhos.

4. Implementação

Antes de aplicar o sistema em um ambiente real é importante testá-lo em um cenário controlado. A Figura 4 ilustra o *testbed* construído com essa finalidade.

A infraestrutura da rede é composta por três máquinas: *zeus*, *cronos* e *dionisio*, que são interligadas por dois comutadores, um de 100Mbps e um de 1Gbps. Sobre ela foi criada uma rede virtual com dois roteadores: *vrouter1A* e *vrouter2A*. As máquinas *artemis* e *apolo* são os *hosts* que farão uso da rede virtual para se comunicarem. Os caminhos foram definidos por rotas estáticas nos *hosts* e interfaces virtuais, que utilizam

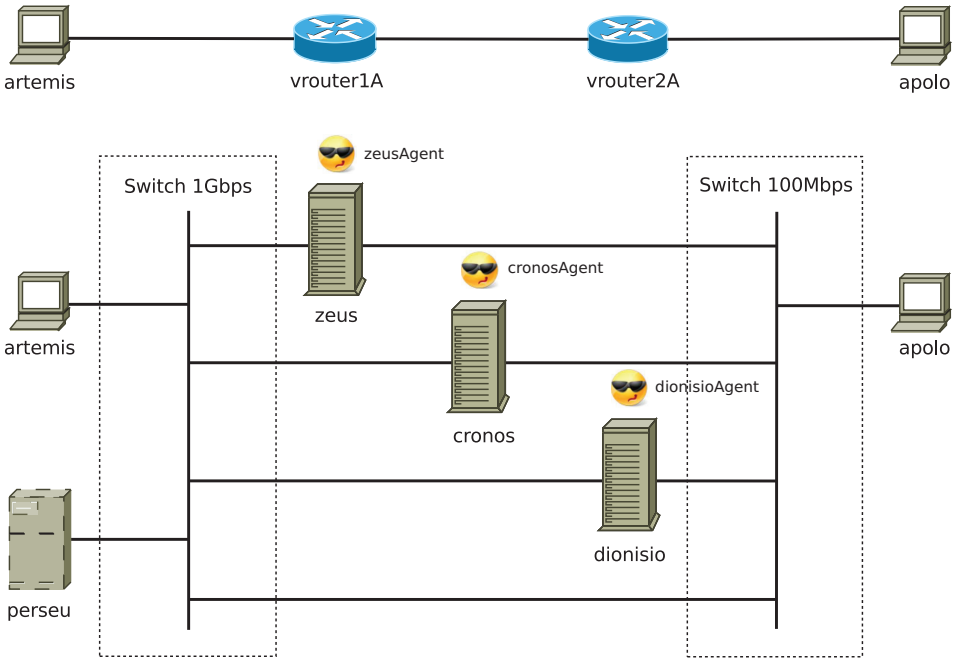


Figura 4. *Testbed* criado para validação do sistema de gerenciamento distribuído.

um endereçamento diferente da rede de controle. Na máquina *perseu* fica o repositório de arquivos, onde estão armazenados as imagens das máquinas virtuais. Tanto as máquinas físicas quanto as virtuais possuem o sistema operacional Debian GNU/Linux com a versão 2.6.32 do kernel.

Os roteadores virtuais foram criados com o KVM. Ele possui um bom desempenho e oferece virtualização total. O KVM possui suporte a SMP, *ballooning* de memória, interligação de interfaces físicas e virtuais através de comutação ou roteamento, e migração *live*, em que toda a memória da máquina virtual é copiada em tempo de execução antes do controle ser transmitido da origem para o destino.

O gerenciamento das máquinas virtuais tanto para a montagem do *testbed* quanto pelos agentes utiliza a biblioteca Libvirt [Coulson et al. 2010]. Ela fornece uma API para monitoramento e controle de diversas plataformas de virtualização, entre elas KVM, Xen e VMware. O uso da Libvirt no sistema de autogerenciamento distribuído de redes virtuais é importante por torná-lo independente de tecnologia.

Os agentes rodam nas máquinas físicas do núcleo da rede: *zeus*, *cronos* e *dionisio*, como mostra a Figura 4. Nós utilizamos a plataforma Ginkgo [Ginkgo Networks 2008] para a implementação dos agentes. Ela permite a criação de agentes leves e portáveis, o que facilita sua implementação em ambientes heterogêneos: roteadores, comutadores, de redes guiadas ou sem-fio. O Ginkgo é um *framework* que possui os blocos de construção básicos dos agentes e os mecanismos para comunicação intra-agentes e para o controle remoto dos agentes através de serviços web.

O experimento tem como objetivo validar o sistema de autogerenciamento distribuído e testar diferentes abordagens para o reparo das redes virtuais. O tempo de recuperação de um roteador virtual pode ser definido segundo a fórmula:

$$T_r = T_d + T_p + T_i + T_c$$

onde T_d é o tempo de diagnóstico da falha; T_p é o tempo gasto no planejamento da ação, que envolve trocas de informações entre os agentes; T_i é o tempo de instanciação da máquina virtual e T_c é o tempo de configuração do roteador através do algoritmo de roteamento.

Nos experimentos nós utilizamos duas abordagens para o planejamento da execução, uma pré e uma pós-diagnóstico. Na primeira, o comportamento Planejamento é executado a cada ciclo, calculando o custo da máquina e difundindo essa informação através da vizinhança. Na segunda, ele é executado apenas quando é diagnosticada uma falha pelo comportamento Análise do agente.

Também foram estudadas duas formas para a instanciação do roteador virtual. Uma delas inicializa o sistema da máquina virtual utilizando a imagem que está no repositório de arquivos na máquina *perseu*. A outra utiliza um arquivo com um estado anterior da memória, que também está no repositório de arquivos, gerado em um momento em que o roteador virtual já estava em funcionamento. Nesse caso, não é preciso inicializar ou reconfigurar as rotas.

As execuções foram realizadas com roteamento estático e dinâmico na rede virtual. O roteamento estático foi configurado manualmente nos roteadores virtuais. Para o dinâmico foi utilizada a suíte de roteamento Quagga [Ishiguro 2006], rodando o algoritmo de roteamento OSPFv2.

5. Resultados e discussão

Um experimento inicial foi realizado no *testbed* antes da implementação do sistema multiagentes. Nesse experimento foram realizadas migrações de um roteador virtual com roteamento estático enquanto um fluxo UDP de taxa constante de 500Kbps de *artemis* a *apolo* passava pela rede virtual. O tráfego foi gerado com a ferramenta Iperf. As curvas do gráfico da Figura 5 mostram a taxa de chegada em *apolo* e quando ela está em zero indica perdas na rede. Todas as migrações iniciaram próximas ao instante 8s.

Na primeira execução, a máquina virtual foi destruída na origem e inicializada no destino. Nela, a recuperação demorou 27s devido principalmente ao tempo de inicialização do sistema. Na segunda execução o estado do roteador virtual foi salvo no repositório de arquivos, ele foi interrompido e restaurado no destino. Mesmo gastando mais tempo com o salvamento, essa execução levou apenas 45% do tempo da anterior. A terceira e a quarta utilizam serviços de migração oferecidos pelo KVM. Na migração normal a máquina virtual é interrompida, a memória é copiada pela rede da origem ao destino onde, por fim, a máquina é reativada. Essa situação é parecida com a anterior, exceto pelo fato que a cópia é feita diretamente e não através de uma máquina intermediária e portanto ela é um pouco mais rápida, 37% do tempo da primeira execução. Nessas duas situações, no momento que as máquinas foram restauradas ocorreu um pico na rede. Isso ocorreu provavelmente porque no instante em que a memória da máquina virtual foi salva, havia pacotes no *buffer* que se somaram aos que estavam chegando no instante que ela foi reativada. A última execução foi feita com migração live, em que a memória é copiada da origem ao destino enquanto a máquina virtual está em execução e somente quando não houver mais nenhuma página modificada, o controle é transferido. Essa é

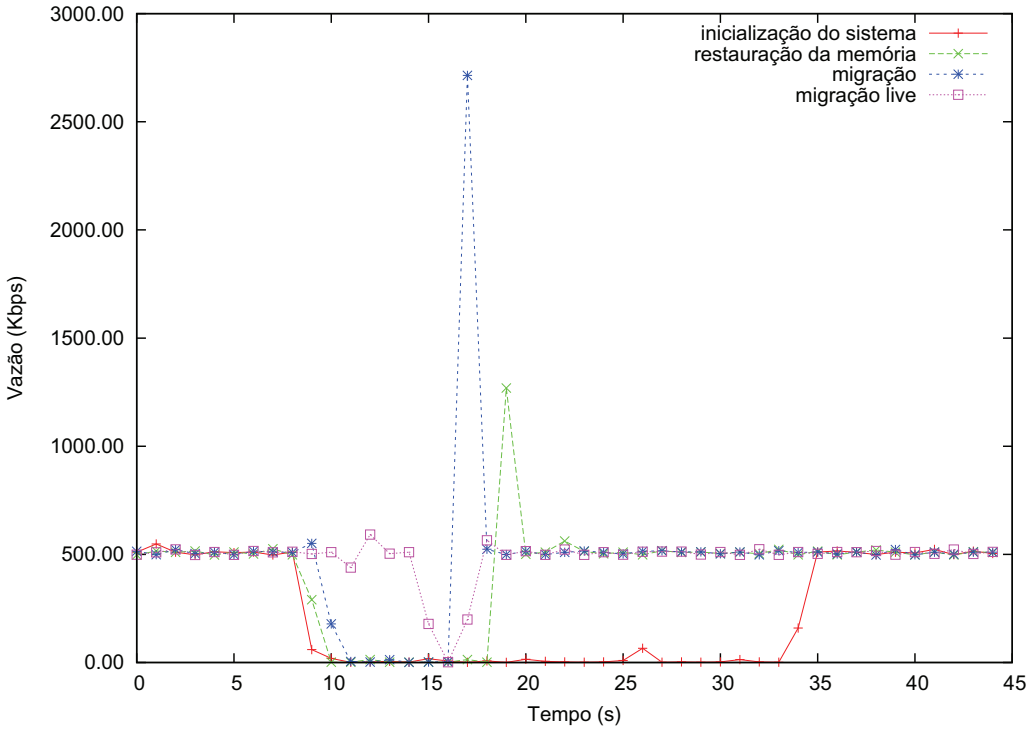
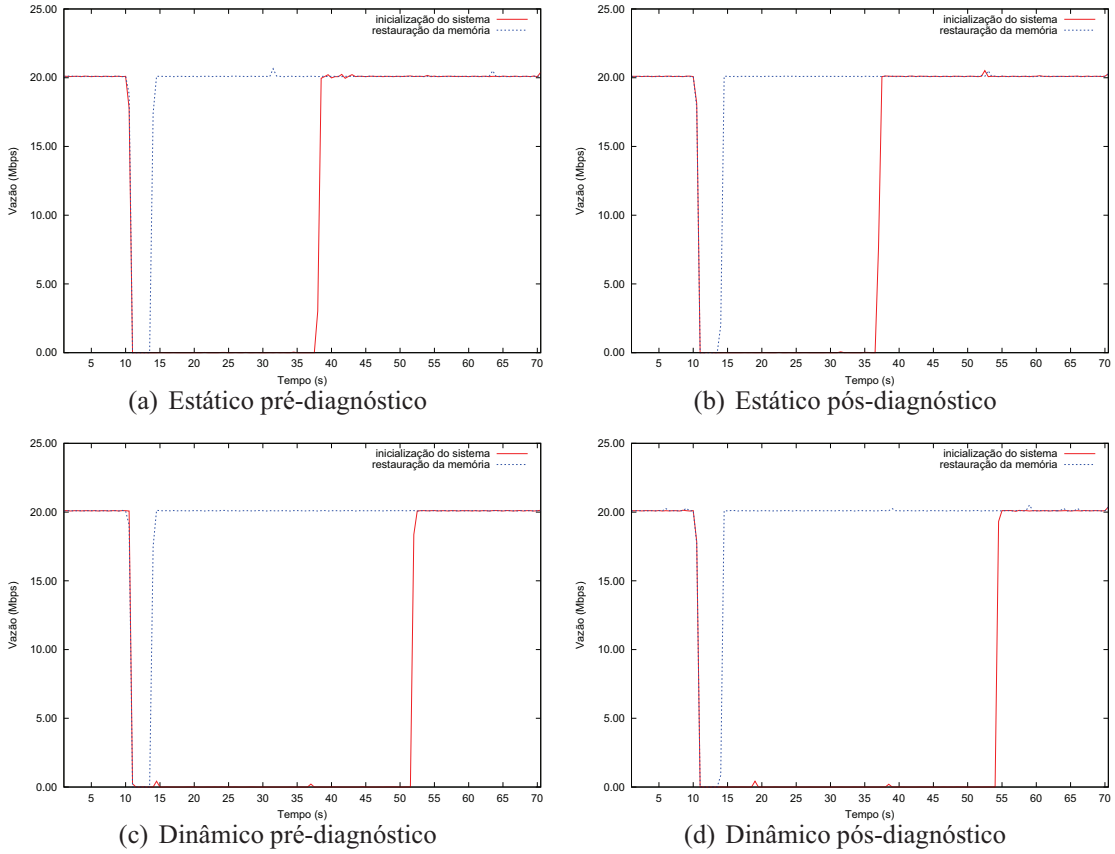


Figura 5. Gráfico do experimento sem o sistema multiagentes.

a melhor forma de migrar um roteador virtual, pois a transmissão foi interrompida por menos tempo, menos de 15% em relação à primeira execução.

Os gráficos da Figura 6 mostram os resultados do experimento com o sistema de gerenciamento distribuído. Os gráficos (a) e (b) apresentam as execuções com roteamento estático, e (c) e (d), com roteamento dinâmico. Em (a) e (c) estão os gráficos das execuções que utilizaram o planejamento pré-diagnóstico e em (b) e (d), os de planejamento pós-diagnóstico. Em cada gráfico estão sendo comparadas as abordagens de inicialização do sistema da máquina virtual e de restauração a partir de um arquivo com um estado anterior da memória. Em todas as execuções um fluxo UDP de taxa constante de 20Mbps de *artemis* a *apolo*, gerado pela ferramenta Iperf, está passando pela rede virtual. As curvas representam as taxas de chegada de pacotes ao destino ao longo do tempo.

Após 10s a máquina *zeus* é desconectada da rede. Os agentes em *cronos* e *dionisio* percebem a falha de *zeus* porque param de receber a difusão da sua base de conhecimento. No experimento os agentes realizam a difusão a cada 0,5s e quando eles percebem que a informação de algum nó físico está desatualizada a mais de 1s o comportamento Análise gera o diagnóstico de falha. Portanto, em todos os experimentos, o tempo de diagnóstico da falha varia entre 1 e 1,5s. No caso do planejamento pré-diagnóstico, os agentes iniciam imediatamente a execução, e no pós-diagnóstico, o comportamento Planejamento realiza o cálculo do custo que será enviado aos demais agentes na próxima difusão. O comportamento Execução verifica se a informação do custo de todos os nós físicos, com exceção do que falhou, está atualizada. O custo de *dionísio* é o maior porque ele já possui um roteador virtual em execução. Portanto, quem irá recuperar *vrouter1A* é o agente em *cronos*, que inicializa o sistema do roteador virtual ou recupera o estado da memória se



(a) Estático pré-diagnóstico (b) Estático pós-diagnóstico

(c) Dinâmico pré-diagnóstico (d) Dinâmico pós-diagnóstico

Figura 6. Gráfico do experimento com os agentes

encontrar o arquivo de *backup*. Ele difunde a informação de término da execução através da base de conhecimento e todos os agentes limpam as informações de planejamento.

As maiores variações estão na forma que a máquina virtual é recuperada, como pode ser visto na diferença do tempo de restabelecimento da conexão, que chega a ser quase 9 vezes maior quando o sistema tem que ser carregado. A diferença é maior nos casos em que um algoritmo de roteamento está sendo executado, pois o roteador virtual precisa reconfigurar suas rotas para restabelecer a conexão da rede. A recuperação através da inicialização do sistema demorou 41,1s com o planejamento pré-diagnóstico e 43,6s com o pós-diagnóstico. Com o roteamento estático a diferença é menor. Os tempos de recuperação foram de 27,5s com pré-diagnóstico e 26,4s com pós-diagnóstico.

Na abordagem em que o estado da memória é restaurado, o tipo de roteamento praticamente não interferiu, pois a máquina virtual2A não chegou a fazer alterações na tabela de roteamento. Tanto no cenário estático quanto no dinâmico, o tempo de recuperação foi de 3,1s com o planejamento pré-diagnóstico e de 3,5s com o pós-diagnóstico. Embora essa abordagem seja mais rápida, o salvamento, a transmissão e o armazenamento do estado da memória causam impactos significativos. É interessante a utilização de *ballooning* para reduzir a memória da máquina virtual.

As estratégias de planejamento pré e pós-diagnóstico possuem pouca diferença, e é mais perceptível na abordagem de restauração da memória, pois na outra abordagem o tempo é mais variável devido à inicialização do sistema. Porém, devemos levar em

consideração que o nosso *testbed* é pequeno e que em cenários maiores a sobrecarga na rede para o planejamento da execução pode aumentar bastante, causando uma latência maior. Nesse sentido é interessante a separação da rede em vizinhanças menores e mais locais para reduzir os impactos da difusão. A agregação de dados como é feita no cálculo local do custo também é uma estratégia para reduzir a sobrecarga.

Também seria possível combinar as estratégias anteriores e salvar o estado da memória do roteador virtual na máquina de menor custo calculado no planejamento prévio. Isso seria bom para evitar o armazenamento centralizado, o que gera um ponto único de falhas e aumenta a sobrecarga da rede.

6. Conclusão

Sistemas baseados em computação bio-inspirada geralmente não possuem desempenho ótimo, mas podem apresentar características interessantes como robustez e escalabilidade. Nosso sistema multiagentes de autogerenciamento distribuído de redes virtuais utiliza os conceitos da computação autonômica para lidar com a complexidade. O sistema apresentou-se funcional no cenário estudado, cujo foco é a autocura das redes virtuais. Também pudemos analisar diferentes abordagens para a recuperação dos roteadores virtuais, como a restauração do estado da memória e o planejamento de ações pré-diagnóstico.

Os resultados se mostraram favoráveis à utilização de técnicas de backup de memória, em especial se tratando de roteadores virtuais para ganhar o tempo de configuração das rotas. Esse backup poderia ocorrer em momentos de ociosidade da rede para não impactar no funcionamento da mesma e os sistemas de virtualização poderiam oferecer recursos para manter a cópia da memória atualizada de maneira eficiente.

Para os trabalhos futuros pretendemos validar o sistema multiagentes de autogerenciamento de redes virtuais em um cenário maior. Também desejamos estudar algumas questões levantadas na discussão, como a utilização de técnicas de *ballooning* de memória e a manutenção dos *backups* do estado dos roteadores virtuais através do sistema multiagentes. Outro assunto de interesse é o autogerenciamento de enlaces virtuais. Alguns problemas poderiam ser corrigidos apenas com a migração do enlace virtual, evitando a migração de roteadores virtuais, que possui um custo maior.

Referências

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

Barham, P., Dragovic, B., Fraser, K., Hand, S., Harris, T., Ho, A., Neugebauer, R., Pratt, I., and Warfield, A. (2003). Xen and the art of virtualization. *SIGOPS Oper. Syst. Rev.*, 37:164–177.

Boucadair, M., Georgatsos, P., Wang, N., Griffin, D., Pavlou, G., Howarth, M., and Elizondo, A. (2009). The agave approach for network virtualization: differentiated services delivery. *Annals of Telecommunications*, 64:277–288. 10.1007/s12243-009-0103-4.

Braga, T. R. M., Silva, F. A., Ruiz, L. B., and Assunção, H. P. (2006). Redes autonômicas. In *Minicursos do Simpósio Brasileiro de Redes de Computadores, SBRC'2006*.

Clark, D. D., Partridge, C., Ramming, J. C., and Wroclawski, J. T. (2003). A knowledge plane for the internet. In *Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications*, SIGCOMM '03, pages 3–10, New York, NY, USA. ACM.

Coulson, D., Berrange, D., Veillard, D., Lalancette, C., Stump, L., and Jorm, D. (2010). Libvirt 0.7.5: Application development guide. disponível em: http://libvirt.org/guide/pdf/application development guide.pdf.

Dobson, S., Denazis, S., Fernández, A., Gaïti, D., Gelenbe, E., Massacci, F., Nixon, P., Saffre, F., Schmidt, N., and Zambonelli, F. (2006). A survey of autonomic communications. *ACM Trans. Auton. Adapt. Syst.*, 1:223–259.

Fajjari, I., Ayari, M., and Pujolle, G. (2010). Vn-sla: A virtual network specification schema for virtual network provisioning. *International Conference on Networking*, 0:337–342.

Feamster, N., Gao, L., and Rexford, J. (2007). How to lease the internet in your spare time. *SIGCOMM Comput. Commun. Rev.*, 37:61–64.

Gaïti, D., Pujolle, G., Salaun, M., and Zimmermann, H. (2006). Autonomous network equipments. In *Autonomic Communication*, pages 177–185.

Ginkgo Networks (2008). Ginkgo distributed network piloting system. white paper.

Habib, I. (2008). Virtualization with kvm. *Linux J.*, 2008.

Horizon (2010). Horizon project: A new horizon to the internet. disponível em: http://www.gta.ufrj.br/horizon/.

Houidi, I., Louati, W., and Zeghlache, D. (2008). A distributed and autonomic virtual network mapping framework. *Autonomic and Autonomous Systems, International Conference on*, 0:241–247.

Houidi, I., Louati, W., Zeghlache, D., Papadimitriou, P., and Mathy, L. (2010). Adaptive virtual network provisioning. In *Proceedings of the second ACM SIGCOMM workshop on Virtualized infrastructure systems and architectures*, VISA '10, pages 41–48, New York, NY, USA. ACM.

IBM (2001). Autonomic computing: Ibm's perspective on the state of information technology.

Ishiguro, K. (2006). Quagga: A routing software package for tcp/ip networks. disponível em: http://www.quagga.net/docs/quagga.pdf.

Kephart, J. O. and Chess, D. M. (2003). The vision of autonomic computing. *Computer*, 36:41–50.

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

McKeown, N., Anderson, T., Balakrishnan, H., Parulkar, G., Peterson, L., Rexford, J., Shenker, S., and Turner, J. (2008). Openflow: enabling innovation in campus networks. *SIGCOMM Comput. Commun. Rev.*, 38:69–74.

Strassner, J. (2007). *Cognitive networks: towards self-aware networks*, chapter The role of autonomic networking in cognitive networks, pages 23–52. John Wiley and Sons.

Turner, J. and Taylor, D. (2005). Diversifying the internet. In *Global Telecommunications Conference, 2005. GLOBECOM '05. IEEE*, volume 2, pages 6 pp. –760.

Yu, M., Yi, Y., Rexford, J., and Chiang, M. (2008). Rethinking virtual network embedding: substrate support for path splitting and migration. *SIGCOMM Comput. Commun. Rev.*, 38:17–29.

Zhu, Y. and Ammar, M. (2006). Algorithms for assigning substrate network resources to virtual network components. In *INFOCOM 2006. 25th IEEE International Conference on Computer Communications. Proceedings*, pages 1 –12.