# Uma Avaliação Experimental de Soluções de Virtualização para o Plano de Controle de Roteamento de Redes Virtuais

**Carlos Corrêa[1], Sidney Lucena[1], Christian Rothenberg[2], Marcos Salvador[2]**

[1] Universidade Federal do Estado do Rio de Janeiro (UNIRIO) –
Rio de Janeiro - RJ

{carlos.correa,sidney}@uniriotec.br

[2]Centro de Pesquisa e Desenvolvimento em Telecomunicações (CPqD) –
Campinas - SP

{esteve,marcosrs}@cpqd.com.br

***Resumo.*** *A virtualização de redes é considerada um dos fundamentos para a construção da Internet do Futuro e capaz de catalisar tecnologias para a solução de problemas clássicos da camada IP. O presente trabalho faz um avanço nesta direção, avaliando o desempenho de diferentes tecnologias de redes virtuais, sob a perspectiva de um plano de controle de roteamento IP virtualizado. Foram testadas as ferramentas Xen, OpenVZ e LXC executando roteadores virtuais rodando Quagga, tanto com conectividade baseada em bridges nativas Linux quanto OpenVSwitch. Os resultados obtidos permitem uma análise comparativa com relação a funcionalidades disponíveis, consumo de recursos e tempos de convergência do protocolo de roteamento associado.*

***Abstract.*** *Network virtualization is one of the foundations for the Future Internet and able to catalyze technologies for the solution of classical problems in the IP layer. This work makes a step forward in that direction, evaluating the performance of different virtual network technologies, under the perspective of a virtualized IP routing control plane. The tools Xen, OpenVZ and LXC were evaluated in the context of virtual routers running Quagga, as the connectivity provided to them by both Linux bridges and OpenVSwitch. The results allow a comparative analysis of the tools, in respect available features, resource utilization and convergence of the associated routing protocol.*

## 1. Introdução

A virtualização de sistemas é uma técnica que permite que múltiplos processos em execução compartilhem o mesmo hardware, enquanto oferece a eles a ilusão de executarem sobre recursos dedicados. Inicialmente um mecanismo de isolamento, ela representa um fator de uso eficiente da crescente capacidade computacional disponível [Egi et al. 2007]. Mais tarde, um conceito análogo surgiria no contexto das redes de computadores, dando origem à *virtualização de redes* [Chowdhury and Boutaba 2010]. A virtualização de redes permite particionar a capacidade dos componentes de uma rede física. Torna-se possível então estabelecer múltiplas infraestruturas lógicas distintas sobre os mesmos componentes.

A virtualização de redes também concorre para a eficiência das arquiteturas de redes. Funções tradicionalmente gerenciadas de forma distribuída passaram a ser projetadas para uma execução e administração centralizadas. Este é o caso das arquiteturas virtuais de roteamento IP propostas em [Bolla et al. 2009] e [Nascimento et al. 2010]. Nestas arquiteturas as decisões de roteamento de tráfego, originalmente realizado por nós roteadores, são encaminhadas pelos comutadores a um sistema controlador. O controlador executa uma versão virtualizada da rede, a partir da qual obtém informações de roteamento e as utiliza para instruir aos comutadores como os fluxos devem ser transmitidos.

É possível supor que as características dos componentes do plano de virtualização influenciem o desempenho destas arquiteturas. Porém, não foi possível identificar na literatura uma caracterização destes elementos sob o ponto de vista de sua aplicabilidade. A ausência destas informações leva à necessidade de se construir múltiplos protótipos, combinando diferentes componentes, até a obtenção de uma solução satisfatória [Wang et al. 2008].

O presente trabalho tem por objetivo avaliar o desempenho de uma seleção de ferramentas de virtualização que seja representativa do estado-da-arte nesta área, contribuindo com informações quantitativas em relação aos requisitos funcionais para a construção de uma arquitetura virtual de serviços de roteamento IP. Para tal, os componentes de virtualização de cada ferramenta foram analisados de maneira a se determinar um conjunto de requisitos comuns que sejam relevantes no contexto dessas soluções. A partir desta análise, algumas métricas foram definidas objetivando evidenciar a adequação de cada solução. Os resultados apresentados caracterizam quantitativa e qualitativamente cada ferramenta. Assim, podem apoiar a escolha dos componentes para uma arquitetura de plano de controle virtual de roteamento.

O texto das próximas seções encontra-se assim organizado: a Seção 2 apresenta uma revisão da literatura de virtualização de sistemas e de redes, além das arquiteturas virtuais de serviços de roteamento IP; na Seção 3 são elencados os requisitos para os componentes das arquiteturas, além de selecionados aqueles a serem avaliados; na Seção 4 está documentada a metodologia de avaliação dos componentes; na Seção 5 os resultados dos experimentos realizados são apresentados, sendo reservada a Seção 6 para apresentação das conclusões e trabalhos futuros.

## 2. Virtualização

Freqüentemente uma solução de virtualização fornece aos sistemas executando sob sua supervisão a abstração de recursos computacionais de uso exclusivo. De fato, porém, tem-se um computador (“anfitrião”) cujos recursos são partilhados entre estes sistemas.

De maneira similar, a virtualização de redes (ou VR) trata-se de um método para que múltiplas arquiteturas de rede heterogêneas compartilhem o mesmo substrato físico - neste caso, componentes de uma rede como roteadores, comutadores, etc.

A seguir, são apresentados em detalhes cada um dos conceitos, bem como os trabalhos realizados em ambos os segmentos e que se relacionam a este.

### 2.1. Virtualização de Sistemas

Pode-se realizar a virtualização de sistemas de duas formas: a *virtualização completa*, em que cada convidado executa seu sistema operacional, e a virtualização baseada em *containers*, em que o sistema operacional do anfitrião distribui e isola os recursos disponíveis entre os sistemas convidados [Bhatia et al. 2008].

Ainda segundo [Bhatia et al. 2008], na virtualização completa o anfitrião executa um Monitor de Máquinas Virtuais (MMV, ou *hipervisor*). É responsabilidade do MMV prover uma abstração de *hardware* (chamada *máquina virtual*) para que seja possível executar o sistema operacional convidado. Ele também deve mapear cada requisição dos convidados a seus respectivos dispositivos virtuais para o elemento físico correspondente. Existe uma variação desta técnica chamada *paravirtualização* [Bhatia et al. 2008], que consiste em otimizar a emulação de *hardware* provida pelo MMV com vistas a um ganho de desempenho. Nesta abordagem, porém, os sistemas operacionais convidados precisam ser modificados - o que não acontece na virtualização completa.

Em um virtualizador de *containers*, uma imagem de sistema operacional virtualizada é compartilhada entre os nós convidados. Isto pode ser mais eficiente quando é preciso apenas executar de forma isolada diversas cópias do mesmo *software*. Ainda assim os nós virtuais podem ser individualmente gerenciados, como na virtualização completa.

### 2.2. Virtualização de redes e conectividade virtual

Segundo [Chowdhury and Boutaba 2010], existem quatro abordagens para a implementação de VR: as Redes Locais Virtuais (VLANs, na sigla em inglês), as Redes Virtuais Privadas (VPNs, também na sigla da língua inglesa), as redes ativas e as redes de sobreposição. Em [Sherwood et al. 2010] apresenta-se uma quinta, a partir do conceito de redes programáveis via *software*.

Para constituir uma rede programável via *software* é necessário dividir os elementos de uma rede em duas categorias. Há aqueles dedicados a encaminhar o tráfego entre os sistemas finais conectados, os quais designamos elementos do *plano de dados*. Complementarmente, há elementos que atuam no *plano de controle* (controladores), sendo responsáveis por decidir de que forma o plano de dados deverá realizar seu trabalho. Acrescenta-se então ao plano de controle o suporte a uma interface programática, o que permite que aplicações de alto nível sejam criadas para instrumentalizá-lo.

Independentemente do uso de VR, existe o desafio de se estabelecer conectividade na camada de enlace entre MVs de uma mesma subrede (ou domínio de *broadcast*), especialmente quando estas executam sobre o mesmo anfitrião. Nestes cenários surge a necessidade de um mecanismo de *conectividade virtual*. Tal conectividade é geralmente estabelecida por dispositivos virtuais baseados em *software* ([Bohme and Buytenhenk 2001], [Pfaff et al. 2009]), que são essenciais para as plataformas de roteamento aqui estudadas.

### 2.3. Plataformas virtuais de roteamento IP

Enquanto a tecnologia programável torna o encaminhamento de tráfego dentro da mesma sub-rede uma decisão trivial por parte dos elementos do plano de controle, ferramentas de virtualização são aplicadas às plataformas virtuais de roteamento IP para permitir que um fluxo de dados seja transmitido entre domínios de *broadcast* distintos.

Nestas arquiteturas, o plano de controle é representado por um sistema controlador, que possui em sua memória uma representação da topologia da rede controlada. Isto pode ser visto na Figura 1, que mostra uma rede atendida por uma plataforma virtual de roteamento IP. Os $n$ comutadores da rede estão interligados em série, tornando-a plenamente conexa no nível de enlaces. Cada comutador físico é representado na memória do controlador por uma instância de um mecanismo de conectividade virtual.

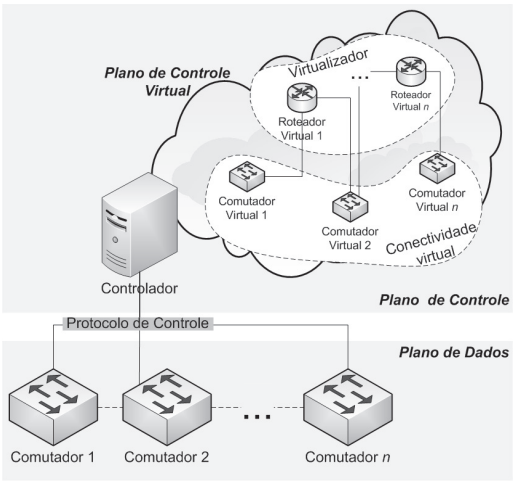


**Figura 1. Visão esquemática de uma plataforma virtual de roteamento IP.**

Em uma arquitetura tradicional, para permitir a comunicação entre clientes desta rede configurados em diferentes domínios de *broadcast* seria preciso empregar um dispositivo com função de roteamento (um roteador, por exemplo). No cenário apresentado, porém, MVs executando processos de roteamento são suficientes para que esta função seja desempenhada.

Para conhecer a rota a ser seguida por um fluxo do plano de dados entre domínios de *broadcast* distintos, o controlador consulta as bases de informações de roteamento dos processos do plano de controle virtual. Assim, é possível determinar por quais enlaces os dados devem ser transmitidos para chegar a seu destino, bem como comandar os elementos do plano de dados de acordo.

Foram pesquisadas na literatura propostas de plataformas virtuais de roteamento IP, descritas a seguir.

O trabalho em [Wang et al. 2008] propõe um *hipervisor de plano de dados*. Ao operar em conjunto com um plano de encaminhamento IP baseado em roteadores virtuais, esta solução permitiria a migração de processos de roteamento entre diferentes nós físicos "a quente", isto é, sem que fosse necessário interromper seu funcionamento. As ferramentas adotadas são o virtualizador OpenVZ e o *framework* de conectividade virtual TUN/TAP. Não obstante, o autor do trabalho também relata haver testado o uso do virtualizador Xen, mas que os recursos de migração de MVs disponíveis para esta ferramenta não permitiam uma migração bem-sucedida do plano de controle.

Apesar do uso intensivo de virtualização, a arquitetura de [Wang et al. 2008] não prevê a segregação do tráfego do plano de controle na memória de um controlador.

Em [Bolla et al. 2009] se propõe uma arquitetura chamada DROP, em que a

virtualização de processos roteadores é substituída pela execução de múltiplos processos de roteamento modificados para operar através de *links* virtuais. Tais canais de comunicação são estabelecidos a partir de uma API de comunicação inter-processos do sistema Linux, e se destinam à troca de informações do plano de controle. Os processos modificados também podem executar sob diferentes nós físicos. Apesar disto, a arquitetura depende da existência de um segmento de rede dedicado ao plano de controle, o que limita a flexibilidade e a escalabilidade da arquitetura.

A arquitetura introduzida em [Nascimento et al. 2010] propõe o uso de um virtualizador, mas sugere-se que a especificação da ferramenta a ser utilizada seja objeto de estudo posterior. A conectividade virtual neste caso é provida pelo *framework* TUN/TAP [Krasnyansky 2000].

Uma inovação ainda a ser alcançada pelas arquiteturas existentes é o estabelecimento de um plano virtual de roteamento IP que execute de forma inteiramente distribuída. Apesar desta funcionalidade ser colocada como um dos objetivos de [Bolla et al. 2009] e [Nascimento et al. 2010], nos dois casos ela fica reservada a pesquisas futuras.

As funções desempenhadas pelos virtualizadores e pela conectividade virtual no contexto das arquiteturas estudadas são bem conhecidas. Ainda assim, não foram encontradas análises que visassem justificar a escolha das ferramentas para compor o plano de controle virtual das mesmas, quer sob o ponto de vista funcional, quer de desempenho. Tal condição limita sua aplicação fora do âmbito experimental.

A literatura também fornece poucos detalhes a respeito de como refletir no plano de controle eventos ocorridos no plano de dados, como a indisponibilidade de um enlace. A solução em [Wang et al. 2008] parte do princípio de que tais eventos são previsíveis, enquanto em [Bolla et al. 2009] processos e protocolos específicos para o tratamento dos mesmos são propostos. Em [Nascimento et al. 2010] nenhuma abordagem específica é relatada, mas fica documentada a necessidade de uma avaliação aprofundada do tema.

A próxima seção descreve requisitos para os componentes do plano de controle de uma plataforma virtual de roteamento IP, de maneira genérica o suficiente para que atendam a uma ampla gama de abordagens.

## 3. Requisitos de uma plataforma virtual de roteamento IP

Em variados graus, todas as plataformas virtuais de roteamento IP encontradas na literatura contam com dois tipos de componentes, com funções claramente definidas: um virtualizador de sistemas e um mecanismo de conectividade virtual.

A partir das características que motivaram as escolhas das ferramentas utilizadas em cada uma dessas plataformas, este trabalho propõe os seguintes requisitos para o componente virtualizador de uma plataforma virtual de roteamento IP:

- *Isolamento*. O virtualizador deve ser capaz de segregar tanto os recursos quanto a pilha de rede dos sistemas executando sob sua supervisão. De outra forma, não seria possível executar versões concorrentes de processos de roteamento.
- *Eficiência*. O instrumento deve impor a menor sobrecarga possível, mantendo o máximo de recursos à disposição das MVs, que têm responsabilidade direta sobre

o desempenho da plataforma. Isto sugere que soluções baseadas em virtualização completa têm menores chances de atender ao requisito: a definição da técnica em si exclui a possibilidade de compartilhamento de código e dados entre anfitrião e convidados.

- *Escalabilidade*. A relação entre o consumo de recursos em um sistema e o número de MVs executadas deve ser aproximadamente linear, permitindo o planejamento flexível da capacidade do nó anfitrião e conferindo elasticidade do número de nós no plano virtual. Um virtualizador pouco escalável pode se tornar um fator limitante para as topologias que podem ser representadas por uma plataforma.
- *Flexibilidade*. A ferramenta de virtualização deve suportar a vinculação de múltiplos adaptadores de rede virtuais a cada MV, independentemente da existência de um adaptador físico ao qual eles possam ser mapeados. Isto é necessário para que possam ser representados, no plano de controle, nós que contam com um número de interfaces de rede diferente daquele disponível no nó anfitrião. Idealmente, o virtualizador também deve suportar, dinamicamente, o acréscimo, modificação e supressão das características dos adaptadores de rede vinculados a uma MV, o que é necessário para que a topologia virtual possa refletir tempestivamente a mudanças na conectividade do plano de dados.

Da mesma forma, este trabalho propõe os seguintes requisitos para o componente de conectividade virtual:

- *Acessos múltiplos*. Deve ser possível conectar múltiplos adaptadores de rede virtuais por meio de um único dispositivo virtual de conectividade. Isto é essencial tanto para a fidelidade da reprodução da conectividade do plano de dados na topologia virtual, quanto para a eficiência das arquiteturas em si. Enquanto a abstração oferecida por um *switch* virtual permite interligar $n$ MVs por meio de $n$ instruções de configuração, alcançar o mesmo efeito por meio de conectividade ponto-a-ponto exigiria a configuração de $n * (n - 1)$ interfaces virtuais.
- *Isolamento*. Deve ser possível executar simultaneamente múltiplas instâncias de dispositivos virtuais de conectividade, sem que estas, no entanto, compartilhem seu tráfego virtual de dados. De outro modo a infraestrutura virtual poderia não refletir fielmente a conectividade física, além de possivelmente trazer resultados inesperados do ponto de vista dos processos de roteamento executados.
- *Eficiência*. A troca de tráfego entre os dispositivos de rede virtuais deve ser a mais rápida possível. Além disso, um instrumento de conectividade virtual deve impor a menor sobrecarga (processamento, memória) possível, não trazendo impacto negativo à execução dos nós do plano virtual. Este aspecto é importante porque o objetivo final da execução de um protocolo de roteamento, sua convergência, é um processo sensível a tempo.
- *Escalabilidade*. A relação entre o consumo de recursos em um sistema e o número de conexões efetuadas entre as MVs deve ser aproximadamente linear, o que viabiliza o planejamento flexível da capacidade do nó anfitrião e concorre para a elasticidade da topologia representada pelo plano virtual.
- *Flexibilidade*. O mecanismo de conectividade deve oferecer primitivas que permitam a associação dinâmica de interfaces de rede virtuais aos dispositivos em execução, permitindo a rápida propagação de eventos do plano físico para o plano

virtual. Supondo-se uma topologia virtual com $n$ roteadores ligados a um *backbone*, cada qual por um adaptador de rede, desconectar um dos nós por meio da abstração de um *switch* virtual exige uma única instrução de configuração, quando realizar a mesma ação em uma topologia construída sobre conectividade ponto-a-ponto exige a desmobilização de $n - 1$ dispositivos virtuais.

- *Extensibilidade*. Preferencialmente, deve ser possível estender a conectividade virtual fornecida pelo mecanismo de maneira a suportar um plano de controle distribuído. Tal extensibilidade poderia ser provida através do suporte a estruturas lógicas externas à plataforma virtual de roteamento IP, como VLANs ou túneis, que seriam utilizados então para que os dados do plano de controle cruzassem transparentemente uma infraestrutura física.

Outros estudos incluem propostas de requisitos para infraestruturas de rede virtualizadas, se não para plataformas virtuais de roteamento IP.

Em [Bhatia et al. 2008], uma série de requisitos são elencados para uma plataforma que possa executar múltiplas redes virtuais programáveis sobre uma infraestrutura física de rede compartilhada: velocidade, isolamento, flexibilidade, escalabilidade e baixo custo. Neste trabalho, duas soluções de virtualização de código aberto são adotadas e combinadas com um mecanismo de conectividade virtual desenvolvido pelos autores, para ter em seguida seu comportamento comparado com o de outras soluções. Não leva em conta, porém, a separação entre os planos de controle e de dados das redes estudadas.

O estudo encontrado em [Fernandes et al. 2010] procura definir primitivas fundamentais para a virtualização de redes - *instanciar*, *deletar* e *monitorar* elementos, além de *migrá-los* entre nós físicos e *definir* seus parâmetros de alocação de recursos. Enquanto todas as primitivas relatadas são claramente importantes para as redes virtuais em geral, aqui assume-se que, no âmbito das plataformas virtuais de roteamento IP, as primitivas de migração e monitoramento podem ser providas com sucesso por instrumentos complementares aos virtualizadores e à conectividade virtual, ou pela combinação de funcionalidades de ambos.

Colocados os requisitos para os componentes das plataformas virtuais de roteamento IP, foram inicialmente selecionadas, dentre os elementos em uso nas plataformas estudadas, aqueles que melhor pudessem atendê-los.

Foram identificados na literatura, como sendo usados para a função de virtualizador, o hipervisor Xen [Egi et al. 2007] e as ferramentas de virtualização baseada em *containers* Linux-VServer [Bhatia et al. 2008] e OpenVZ [Wang et al. 2008]. O Linux-VServer, porém, não suporta plenamente a virtualização da pilha IP do sistema. Isto significa que os containers compartilham de estruturas de rede do anfitrião, como por exemplo sua tabela de rotas. Tal condição afeta o requisito de isolamento e limita severamente seu uso em uma arquitetura de serviços de roteamento IP.

Também foram identificados e considerados para avaliação virtualizadores que, segundo os estudos realizados, ainda não haviam sido aplicados àquelas arquiteturas: os hipervisores KVM [Habib 2008] e VMWare vSphere [Laverick 2010], além do mecanismo de *containers* LXC .

O hipervisor KVM realiza apenas virtualização completa, o que concorre con-

tra o requisito de eficiência e possivelmente contra a escalabilidade e a flexibilidade das arquiteturas que venham a adotá-lo.

O VMWare vSphere é um produto comercial, de código fechado, o que de alguma maneira limita as possibilidades de extensão e adaptação da ferramenta em direção a uma arquitetura especializada, além de, assim como o KVM, não suportar paravirtualização.

O LXC é um virtualizador de *container* recentemente integrado ao núcleo do Linux que inclui o recurso de virtualização da pilha de rede - o que o qualifica, no aspecto de isolamento, a operar como um roteador virtual. Ele também inclui suporte à suspensão e retomada da execução de um *container*, o que pode representar um facilitador para a implementação de um mecanismo de migração entre nós físicos.

Assim, sob o ponto de vista funcional reúnem características compatíveis com os requisitos propostos para a função de virtualizador o Xen, o OpenVZ e o LXC.

Para a função de conectividade virtual, além do *framework* TUN/TAP, amplamente utilizado pelas propostas encontradas na literatura, foram identificados como possíveis candidatos as *bridges* Linux [Benvenuti 2005] e o *switch* baseado em *software* OpenVSwitch, ou simplesmente OVS [Pfaff et al. 2009].

O *framework* TUN/TAP, porém, aplica-se à criação de interfaces virtuais para comunicação ponto-a-ponto, o que contraria o requisito de acesso múltiplo e o desqualifica para efeito desta avaliação. Em contrapartida, as demais alternativas atendem aos aspectos funcionais dos requisitos de isolamento, flexibilidade e extensibilidade estabelecidos para este tipo de componente.

Na próxima seção, as ferramentas de virtualização e conectividade virtual até aqui consideradas adequadas ao estabelecimento de uma plataforma de serviços de roteamento IP são quantitativamente comparadas.

## 4. Plano de experimentação

Para realização da avaliação das ferramentas nos termos inicialmente propostos por este trabalho, elas foram alternadamente utilizadas para operar planos de controle virtuais. Estes representavam diferentes cenários de conectividade, a partir dos quais informações de desempenho puderam ser extraídas para as várias dimensões de requisitos previamente elencadas.

Os experimentos foram executados em um equipamento com uma CPU Intel Core 2 Duo 2.93GHz e 3GB de memória RAM. O computador executava o Debian GNU/Linux 5.0 e estava dedicado à execução dos experimentos. As versões dos sistemas de virtualização testadas foram o Xen 4.0 e as últimas versões estáveis do OpenVZ e do LXC, todas sobre o núcleo Linux 2.6.32.

Duas topologias de rede distintas foram testadas: uma rede com nós dispostos em uma grade 3x3, em seguida numa grade 4x4, executando processos de roteamento. Esta disposição foi adotada como uma simulação de crescimento tanto de nós como de conexões virtuais de rede, além do número destes elementos se aproximar com o encontrado em *backbones* de produção do mundo real [Alderson et al. 2005].

Cada nó do plano de controle correspondia a um roteador virtual baseado na mesma versão de sistema operacional utilizada no anfitrião (Debian GNU/Linux 5.0),

executando a suíte de roteamento Quagga, onde apenas o serviço OSPF se encontrava ativado. Com o intuito de conferir maior generalidade aos dados coletados, foram extraídas informações de desempenho para diferentes intervalos de anúncio (intervalo de *Hello*) do protocolo: 1, 5 e 10 segundos. Sempre que o intervalo de anúncio foi modificado, o intervalo de detecção de enlace indisponível foi consistentemente ajustado - respectivamente para 4, 20 e 40 segundos.

A conectividade entre os roteadores virtuais também foi testada através de *bridges* Linux e do OVS. Dados de interesse que apoiassem a avaliação das ferramentas foram obtidos a partir da captura do tráfego passante nestes dispositivos de *software*.

Cada segmento de conectividade estabelecido entre dois nós contava com duas interfaces de rede virtuais endereçadas em uma sub-rede exclusiva, selecionada a partir do bloco privativo 10.0.0.0/8. Nenhuma das interfaces utilizadas pelo protocolo de roteamento possuía sumarização ativada (todas as sub-redes foram atribuídas à área 0 do protocolo OSPF).

Dentre todos os cenários possíveis, não foi avaliado o uso do virtualizador OpenVZ com conectividade OVS. Isto porque ambos eram incompatíveis até o momento do experimento, sem perspectiva recente de mudança deste aspecto.

Dados foram extraídos em diferentes etapas da operação de um plano de controle virtual, com o objetivo de compor diferentes métricas a partir das quais foi possível comparar os desempenhos relativos das ferramentas de virtualização testadas.

Para a etapa de inicialização das topologias em teste, um momento computacionalmente custoso em que normalmente as estruturas de dados e recursos de controle específicos de cada nó precisam ser alocados, as seguintes métricas e dados foram medidos:

- *Tempo de inicialização dos nós*. Trata-se do intervalo de tempo desde a solicitação de instanciamento dos nós de uma topologia virtual e sua plena operação. Dado o grande número de tarefas envolvidas no processo de carga de uma MV, esta medição oferece um termo de comparação de eficiência entre as ferramentas virtualizadoras.
- *Uso de CPU na inicialização*. Permite inferir, em um momento de intensa demanda e através da comparação dos resultados obtidos através dos diferentes sistemas de virtualização, o grau de sobrecarga imposta por cada solução.
- *Uso de memória na inicialização*. Também aplica-se à *eficiência* dos mecanismos de virtualização em teste durante a carga de uma MV - neste caso, avalia-se sua sobrecarga espacial.
- *Tempo até a convergência inicial*. Medição do intervalo de tempo desde a inicialização da topologia até a convergência do protocolo OSPF - aqui definida como o momento em que mensagens de anúncio de enlaces não são mais trocadas entre os nós.

Com o intuito de caracterizar o consumo de recursos das ferramentas durante a operação regular, sem a ocorrência de eventos que pudessem desencadear troca de mensagens de controle OSPF, após dez minutos da inicialização de cada topologia virtual as seguintes métricas foram coletadas:

- *Uso de CPU em operação regular*. Objetiva conhecer a sobrecarga de processamento imposta pelo uso das ferramentas aplicadas a cada um dos cenários testados

quando a topologia encontra-se plenamente operacional.
- *Uso de memória em operação regular*. Também busca conhecer a sobrecarga espacial imposta pelo uso das ferramentas de virtualização aplicadas a cada um dos cenários durante operação plena e livre de falhas.

Por fim, para caracterizar a eficiência dos componentes de virtualização experimentados durante a representação de eventos ocorridos no plano de dados, que motivam a troca e o processamento de mensagens de anúncio de enlaces do protocolo OSPF, foram introduzidas falhas simultâneas nos enlaces das topologias virtuais de forma a reduzir o grafo de conectividade de cada uma delas a uma árvore geradora mínima. Os mesmos enlaces foram reintegrados à topologia 120 segundos após sua desconexão, de maneira que categorias de eventos distintos foram observadas nos experimentos.

Na introdução de tais falhas e durante as atividades de resposta das topologias em execução, foram avaliadas as seguintes métricas:

- *Uso de CPU durante evento*. O consumo de recursos de processamento foi mensurado, para cada etapa de falha aplicada.
- *Uso de memória durante evento*. Durante as etapas de falha, o uso de memória principal do sistema também foi medido.
- *Convergência pós-evento*. Avaliação da evolução do tráfego OSPF desde a introdução de uma falha até a conclusão da troca de mensagens de anúncio do estado de enlaces do protocolo.

Não obstante a afinidade exclusiva das métricas propostas com o requisito de *eficiência* estabelecido na seção anterior, a realização de medidas para duas escalas distintas de topologia - com nove e dezesseis nós - permite a averiguação de linearidade na evolução da demanda por recursos entre os dois cenários e, portanto, a análise do grau de escalabilidade das soluções em teste.

## 5. Análise dos resultados

Com o objetivo de conferir relevância estatística aos resultados obtidos, cada uma das diferentes métricas teve seus dados coletados em 30 execuções distintas de cada topologia de rede para cada um dos cenários possíveis (combinação de ferramentas). Isto permitiu uma avaliação dos resultados em uma escala de confiança de 90% para todos os indicadores estabelecidos.

A Tabela 1 apresenta os resultados para o tempo de inicialização das topologias. O LXC é o mais rápido com a conectividade baseada em *bridges*, realizando a tarefa em um tempo em média 20% menor que o OpenVZ, no caso das grades 3x3. Para grades 4x4, a diferença se mantêm em 16%. Ambas as relações se mantém respectivamente em 18 e 15% ao comparar MVs LXC conectadas via OVS. O Xen é o mais lento, com tempos médios de inicialização de 43 e 78s, para as topologias 3x3 e 4x4, respectivamente, sob qualquer conectividade virtual.

A Figura 2 traz o consumo médio de CPU durante a inicialização das MVs. Conforme é possível visualizar, tanto o LXC quanto o OpenVZ levam a um consumo de 100% durante o processo de inicialização. Não obstante, a ocupação da CPU no LXC e no OpenVZ decaem aos valores de operação regular em menos de 50% do tempo verificado com o Xen. Os resultados apresentados se referem à conectividade do tipo *bridge*.

**Tabela 1. Tempos Médios para Inicialização das Redes Virtuais**

| Virtualizador | Conectividade | Topologia | Tempos para Inicialização (s) |
|---|---|---|---|
| LXC | Bridge | 3x3 | 31 |
| | | 4x4 | 62 |
| | OpenVSwitch | 3x3 | 32 |
| | | 4x4 | 63 |
| OpenVZ | Bridge | 3x3 | 39 |
| | | 4x4 | 74 |
| Xen | Qualquer | 3x3 | 43 |
| | | 4x4 | 78 |

Porém as médias de consumo para o Xen e o LXC se conservam quando a conectividade OVS é utilizada, não estando tais resultados, portanto, representados.

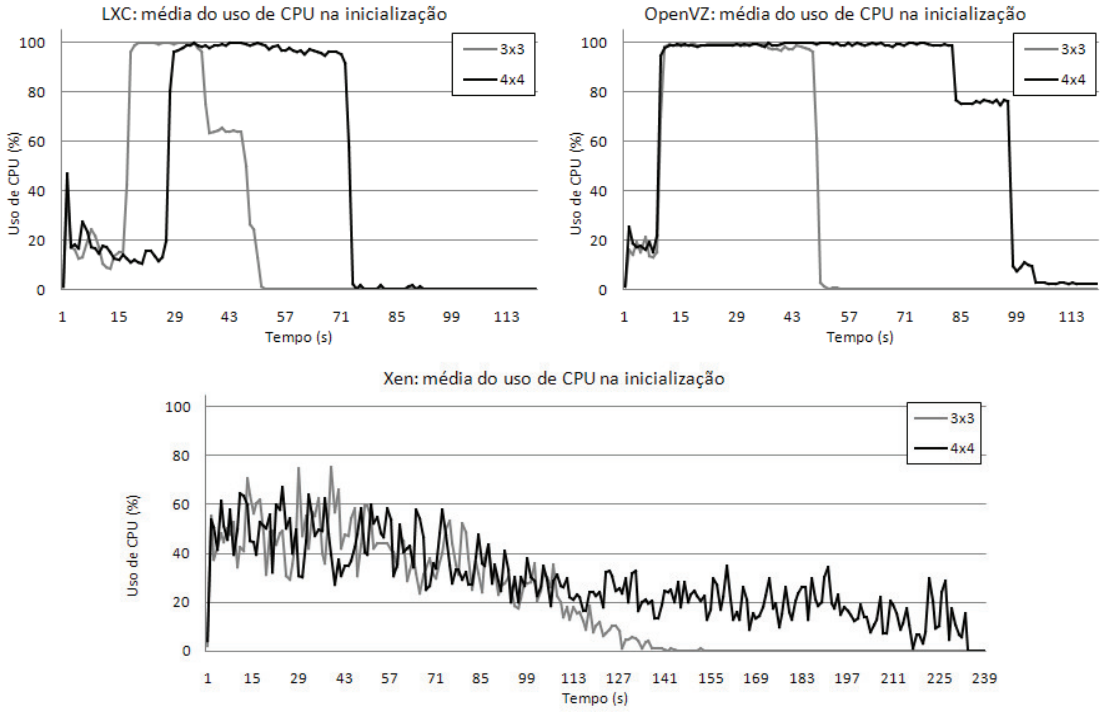


**Figura 2. Consumo de CPU durante a inicialização das topologias virtuais.**

A Tabela 2 apresenta o consumo médio de memória das ferramentas avaliadas, para todos os cenários. A partir da média de uso da RAM durante a inicialização das MVs é possível constatar que as ferramentas LXC e OpenVZ apresentaram evolução de consumo aproximadamente linear com o aumento da escala da topologia, com vantagem significativa para a demanda do LXC, que é menor. O uso do OVS no caso do LXC representa uma demanda adicional de apenas 2,8 a 4% de RAM. No caso do virtualizador Xen, uma quantidade fixa de memória deve ser reservada para cada MV no momento de sua criação (neste cenário 128MB foram reservados). Assim, o uso de RAM variou muito pouco ao longo de seus testes (menos que 5%).

As medições do tempo para convergência inicial do protocolo OSPF apresentaram

**Tabela 2. Utilização Média de Memória ao Longo da Operação das Redes Virtuais**

| Virtual. | Conec. | Topol. | Uso de RAM (MB), durante: | | |
|---|---|---|---|---|---|
| | | | Inicialização | Op. Regular | Evento Falha |
| LXC | Bridge | 3x3 | 169 | 180 | 206 |
| | | 4x4 | 259,50 | 276 | 314 |
| | OpenVSwitch | 3x3 | 175 | 185 | 228 |
| | | 4x4 | 269,75 | 286 | 369 |
| OpenVZ | Bridge | 3x3 | 180 | 226 | 268 |
| | | 4x4 | 337,25 | 350,5 | 693 |
| Xen | Qualquer | 3x3 | 1152 | 1152 | 1152 |
| | | 4x4 | 2048 | 2048 | 2048 |

resultado diverso daquele apontado pelo tempo de inicialização das topologias virtuais. Na Figura 3 estão representados os tempos de convergência inicial para o LXC e o OpenVZ, que obtiveram o melhor desempenho, e também para o Xen.

Há atraso entre o instanciamento da rede virtual LXC e o início da troca de dados OSPF. Mesmo operando a grade mais rápida, carregada em 31s, essa topologia só troca anúncios OSPF 50s após o início do experimento. Seu desempenho é superado pelo do OpenVZ, cujas MVs iniciam a comunicação OSPF imediatamente. Os resultados da ferramenta Xen acompanham seu tempo de carregamento, com troca de anúncios só após 130s do ínicio das topologias virtuais, sob qualquer conectividade.

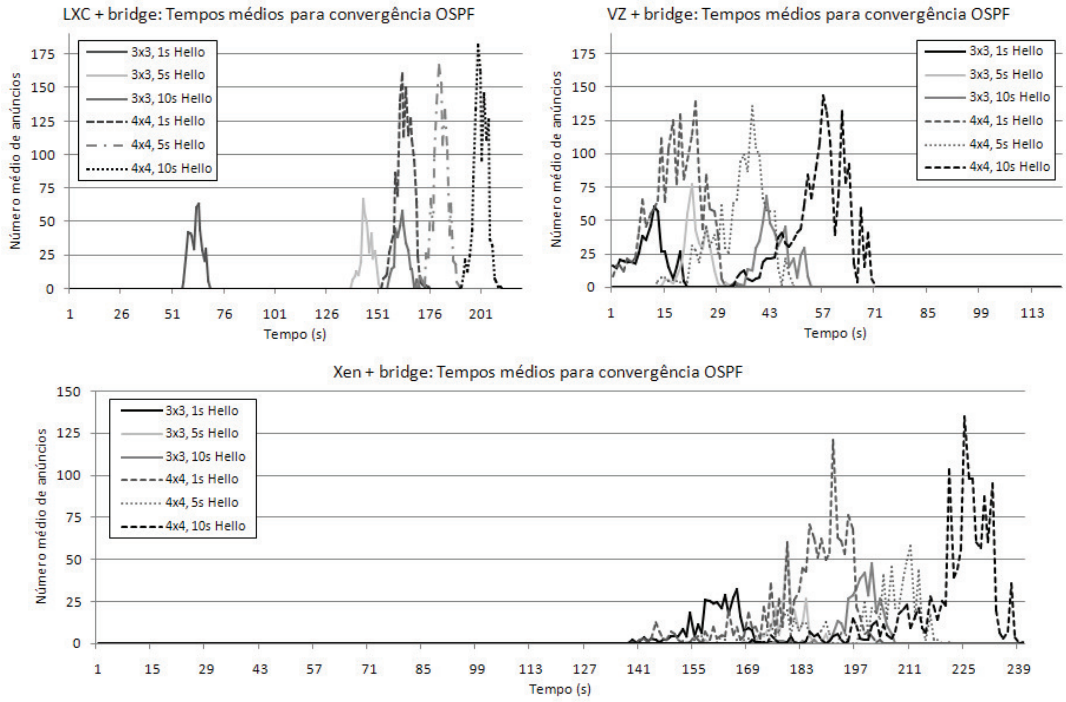


**Figura 3. Variação da convergência OSPF inicial com LXC, OpenVZ e Xen usando conectividade via bridge**

Após a sobrecarga inicial, todas as ferramentas de virtualização apresentam consumo de CPU médio em torno de 1% durante a operação de um plano de controle onde

não estão acontecendo desconexões. O consumo médio de memória é ampliado em 6,4% no LXC, enquanto o OpenVZ apresenta aumento médio de 4%.

Por fim, aplicados os testes de convergência após eventos de enlaces, foi possível verificar que seus resultados não são afetados por atrasos verificados no início da operação das redes virtuais, apesar do desempenho do Xen ainda ficar aquém das demais ferramentas, sob qualquer conectividade utilizada.

À guisa de concisão, a Figura 4 detalha os tempos médios de convergência após eventos de enlaces para as combinações de melhor desempenho. Tanto a convergência do LXC + OVS quanto a do OpenVZ + *bridges* se aproximam do valor ótimo do OSPF (o quádruplo do tempo de Hello), com leve vantagem para o LXC.

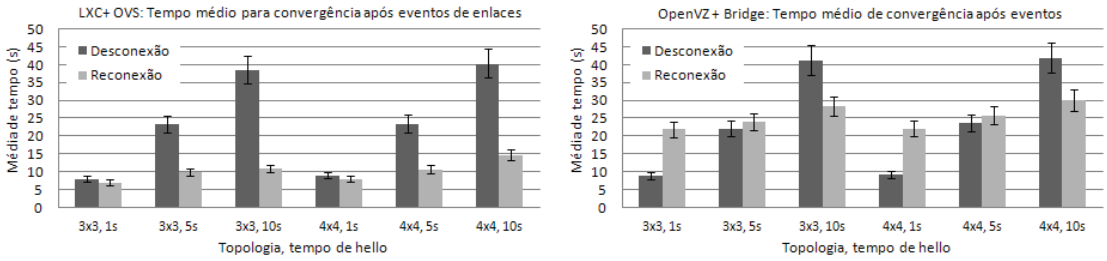


**Figura 4. Convergência LXC + OVS e OpenVZ + bridges após eventos de enlaces.**

O consumo de CPU não alcançou mais que 3% durante a realização dos testes com eventos de enlaces. Conforme pode ser visto na Tabela 2, o consumo de memória cresceu no período. No caso do LXC este crescimento foi aproximadamente linear para os cenários experimentados. Na topologia 4x4 OpenVZ, porém, o uso de RAM se acentua na ocorrência de falhas, o que pode indicar um problema de escalabilidade na ferramenta. Na ausência de novos eventos, o consumo de RAM observado retornou aos valores médios de operação.

## 6. Conclusão e trabalhos futuros

As medições realizadas sugerem que as ferramentas de virtualização baseadas em *containers* reúnem as características mais adequadas à execução de um plano de controle virtual, em oposição à paravirtualização utilizada via Xen. Enquanto os *containers* permitem um uso flexível e escalável da memória, o Xen se baseia em uma alocação fixa.

A virtualização oferecida pela ferramenta LXC exige mais tempo que aquela baseada no OpenVZ para que uma topologia inicie e o protocolo de roteamento OSPF convirja. Ainda assim, a convergência ao longo da operação regular é mais veloz na combinação LXC + OVS. Esta conjunção de ferramentas também apresentou consumo de recursos linear nos experimentos realizados - em oposição ao percebido no OpenVZ. Tais fatores, aliados aos aspectos de flexibilidade e extensibilidade do LXC e do OVS, permitem concluir que ambas são as mais adequadas a planos de controle virtuais de roteamento.

Perspectivas de trabalhos futuros incluem a implementação de uma das arquiteturas virtuais de roteamento IP estudadas usando tais ferramentas, além da implementação da primitiva de migração de nós virtuais LXC a partir das funcionalidades de suspensão e retomada desta solução.

## Referências

Alderson, D., Li, L., Willinger, W., and Doyle, J. C. (2005). Understanding internet topology: principles, models, and validation. *IEEE/ACM Trans. Netw.*, 13:1205–1218.

Benvenuti, C. (2005). *Understanding Linux Network Internals*. O'Reilly Media, Inc.

Bhatia, S., Motiwala, M., Muhlbauer, W., Valancius, V., Bavier, A., Feamster, N., Peterson, L., and Rexford, J. (2008). Hosting virtual networks on commodity hardware. Technical report.

Bohme, U. and Buytenhenk, L. (2001). Linux bridge-stp-howto. http://www.faqs.org/docs/Linux-HOWTO/BRIDGE-STP-HOWTO.html. Acessado em: 20 de novembro de 2010.

Bolla, R., Bruschi, R., Lamanna, G., and Ranieri, A. (2009). Drop: An open-source project towards distributed sw router architectures. In *Global Telecommunications Conference, 2009. GLOBECOM 2009. IEEE*, pages 1 –6.

Chowdhury, N. M. K. and Boutaba, R. (2010). A survey of network virtualization. *Comput. Netw.*, 54:862–876.

Egi, N., Greenhalgh, A., Handley, M., Hoerdt, M., Mathy, L., and Schooley, T. (2007). Evaluating xen for router virtualization. In *16th International Conference on Computer Communications and Networks, 2007. ICCCN 2007.*, pages 1256 – 1261.

Fernandes, N., Moreira, M., Moraes, I., Ferraz, L., Couto, R., Carvalho, H., Campista, M., Costa, L., and Duarte, O. (2010). Virtual networks: isolation, performance, and trends. *Annals of Telecommunications*, pages 1–17.

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

Krasnyansky, M. (2000). Universal tun/tap driver. http://vtun.sourceforge.net/tun/index.html. Acessado em: 21 de novembro de 2010.

Laverick, M. (2010). *VMware vSphere 4 Implementation*. McGraw-Hill, Inc., New York, NY, USA, 1 edition.

Nascimento, M. R., Rothenberg, C. E., Salvador, M. R., and Magalhães, M. F. (2010). Quagflow: partnering quagga with openflow. *SIGCOMM Comput. Commun. Rev.*, 40:441–442.

Pfaff, B., Pettit, J., Koponen, T., Amidon, K., Casado, M., and Shenker, S. (2009). Extending networking into the virtualization layer. In *Proc. of workshop on Hot Topics in Networks (HotNets-VIII)*.

Sherwood, R., Chan, M., Covington, A., Gibb, G., Flajslik, M., Handigol, N., Huang, T.-Y., Kazemian, P., Kobayashi, M., Naous, J., Seetharaman, S., Underhill, D., Yabe, T., Yap, K.-K., Yiakoumis, Y., Zeng, H., Appenzeller, G., Johari, R., McKeown, N., and Parulkar, G. (2010). Carving research slices out of your production networks with openflow. *SIGCOMM Comput. Commun. Rev.*, 40:129–130.

Wang, Y., Keller, E., Biskeborn, B., van der Merwe, J., and Rexford, J. (2008). Virtual routers on the move: live router migration as a network-management primitive. *SIGCOMM Comput. Commun. Rev.*, 38:231–242.