Uma arquitetura baseada em Redes Definidas por Software para isolamento de redes em datacenters virtualizados

Rogério V. Nunes[1], **Raphael L. Pontes**[1], **Dorgival Guedes**[1]

[1] Departamento de Ciência da Computação
Universidade Federal de Minas Gerais – Belo Horizonte, MG – Brazil

{rogervn,raphaelluciano,dorgival}@dcc.ufmg.br

Abstract. *The increasing interest in Cloud Computing has brought new demands to providers of "Infrastructure-as-a-service" solutions. To host a large number of clients in the same datacenter, they require multi-tenant networks that can guarantee traffic isolation and scalability, with low costs. This paper describes a solution for this problem that uses Software Defined Networks (SDN). By using SDN, we can program the virtual switches at the physical servers to meet all those requirements, without demanding special hardware in the network. Experiments show good results with very little overhead, even preventing DoS attacks between tenants.*

Resumo. *O interesse crescente em Computação em Nuvem cria novas exigências para os provedores de soluções de "Infraestrutura-como-serviço". Para abrigar um grande número de clientes em um mesmo* datacenter*, eles requerem redes multi-cliente que possam garantir isolamento de tráfego e escalabilidade, preferencialmente a baixo custo. Este artigo apresenta uma solução para esse problema que usa Redes Definidas por Software (RDS). Com RDS, podemos programar os* switches *virtuais dos servidores físicos de uma forma elegante e flexível para atender a esses requisitos, sem exigir hardware especial na rede. Experimentos mostram bons resultados com muito pouco* overhead*, inclusive prevenindo ataques de negação de serviço entre usuários.*

1. Introdução

Os últimos anos têm presenciado a poplularização da computação em nuvem como um paradigma importante para a implementação de serviços na Internet, o qual exige recursos e soluções organizadas ao redor de sistemas distribuídos. Entre as diferentes formas de computação em nuvem, soluções de infraestrutura-como-serviço (IaaS) estão entre as mais bem sucedidas até o momento. Em IaaS, recursos computacionais são controlados por um modelo pague-pelo-que-usa, viabilizando a criação de soluções elásticas que se adaptam às demandas dos usuários de forma transparente.

Em um cenário como esse, grandes *datacenters* que usam técnicas de virtualização se mostraram a melhor forma de obter alta utilização de recursos mantendo um ambiente flexível. Nessas instalações, clientes do *datacenter* (em inglês denominados *tenants*, inquilinos) podem contratar máquinas virtuais (VMs) que são interconectadas para criar uma rede virtual, sobre a qual os serviços executam. Dois importantes requisitos para que esse tipo de solução funcione são o isolamento dos recursos dos inquilinos e a capacidade do sistema de escalar para acomodar um grande número deles. A primeira condição

é necessária para garantir que os serviços de cada inquilino não sejam afetados pelas ações dos demais (acidental ou maliciosamente). A segunda é essencial para tornar o modelo viável economicamente. Entretanto, apesar da virtualização de máquinas oferecer bom isolamento de CPU, memória e espaço de armazenamento, o acesso à rede é uma outra estória, especialmente quando escalabilidade e facilidade de gerência também são desejáveis [Greenberg et al. 2009].

Alguns novos protocolos e arquiteturas de redes vêm sendo propostos para endereçar esses problemas, mas eles exigem hardware novo para funcionar. Na maioria dos casos, atualizar todo o hardware de rede não é uma opção e soluções mais fáceis de gerenciar, escaláveis e eficientes, ainda não existem.

Neste trabalho propomos um novo sistema, *DCPortals*, que endereça aqueles fatores sem exigir novo hardware. Nossa solução é baseada na re-escrita de campos do cabeçalho do pacote, de forma a esconder a origem e o destino reais do hardware no núcleo da rede, ao mesmo tempo que também esconde o tráfego de cada rede virtual de quaisquer VMs que não pertençam ao mesmo inquilino.

Tendo isso em mente, o restante deste trabalho está organizado da seguinte forma: a seção 2 apresenta alguns conceitos básicos relacionados; a seção 3 descreve a arquitetura adotada e detalha a operação do sistema, cujo comportamento é avaliado na seção 4. Em seguida, a seção 5 coloca o *DCPortals* no contexto de outros trabalhos relacionados e a seção 6 apresenta nossas considerações finais, incluindo a discussão de trabalhos futuros.

2. Conceitos básicos

Alguns *datacenters* se valem do uso de VPNs sobre redes Ethernet para isolar o tráfego de cada inquilino. Apesar de ser uma solução relativamente simples, VPNs são limitadas pela definição do protocolo, que reserva apenas 12 bits para a identificação de diferentes redes virtuais. Outros usam soluções de roteamento da camada 3 para isolar tráfego, o que limita a capacidade do inquilino de configurar seu próprio esquema de endereçamento, exigindo configurações complexas à medida que o número de máquinas cresce. Em ambas as soluções, ainda há o problema de manipular os endereços de um grande número de VMs em uma única rede de *datacenter*. Em alguns casos, cada máquina física pode hospedar até centenas de VMs, cada uma com seu próprio endereço da camada 2 (MAC). Tabelas de encaminhamento na maioria dos *switches* Ethernet têm espaço limitado e o desempenho pode cair significativamente com a necessidade de mais *broadcasts*, já que os *switches* não conseguem apreender todos os endereços de todos os destinos possíveis. Soluções baseadas no empilhamento de protocolos (tunelamento) podem reduzir a necessidade dos *switches* da rede aprenderem todos os endereços, já que eles passam a precisar apenas de identificadores de túneis, mas têm limitações de desempenho e gerência. Uma boa revisão dessas técnicas é o trabalho de Cabuk *et al.* [Cabuk et al. 2007, Cabuk et al. 2008].

Para atingir o objetivo deste trabalho, utilizamos os *switches* virtuais presentes nos monitores de virtualização (hipervisores) presentes em cada máquina física. Esses *switches* podem ser facilmente atualizados para versões já disponíveis que implementam o modelo OpenFlow [McKeown et al. 2008]. Uma dessas versões é a Open vSwitch [Pfaff et al. 2009], que já se tornou padrão para hipervisores como o Xen e o KVM. Com OpenFlow, cada *switch* virtual exporta uma interface de programação para acesso às suas tabelas de encaminhamento. Usando essa interface, um controla-

dor pode informar ao *switch* como tratar pacotes que não casam com nenhum dos fluxos já identificados anteriormente. Para esses pacotes, ele pode determinar como aqueles pacotes (bem como os demais daquele fluxo) devem ser roteados, pode instruí-lo a re-escrever campos dos cabeçalhos baseado em parâmetros do fluxo, ou descartá-los. Enfim, é possível para o administrador do *datacenter* controlar como a rede encaminha cada fluxo de pacotes que a atravessa. Isso levou à criação do conceito de hipervisor de rede, um controlador (software) é capaz de isolar o tráfego dos inquilinos na rede assim como hipervisores de máquinas isolam VMs que compartilham CPU e memória. O sistema resultante é chamado de Rede Definida por Software (ou *Software Defined Network*, SDN) [Casado et al. 2010]. A abstração de hipervisores de redes definidas por software provê uma representação logicamente centralizada da rede, onde a configuração e o controle da rede podem ser realizados facilmente, enquanto mantém a escalabilidade da solução. Há benefícios importantes nesse enfoque:

- **Redução do uso da memória de *switches*:** ao re-escrever os cabeçalhos Ethernet podemos ocultar os endereços de camada 2 das VMs dos *switches* do núcleo da rede, reduzindo o número de entradas usadas em suas tabelas internas.
- **Criação de redes virtuais isoladas para cada inquilino:** ao controlar o encaminhamento de pacotes nos *switches* de borda podemos garantir que o tráfego de cada inquilino nunca alcançará VMs de outros inquilinos. Isso oferece tanto privacidade (VMs de outros inquilinos não podem acessar tráfego daquele inquilino) quanto proteção contra ataques (um inquilino malicioso não é capaz de direcionar seu tráfego para máquinas em outras redes virtuais).
- **Integração da configuração e gerência de VMs e da rede virtual:** *DCPortals* integra o hipervisor de rede com um controlador de virtualização bem conhecido, o OpenStack[1], de forma que a instanciação de máquinas virtuais possa ser integrada à configuração e operação da rede virtual. Se uma VM migra de um hospedeiro para outro, o hipervisor pode identificar isso automaticamente e reconfigurar os *switches* para tratar o tráfego daquele inquilino apropriadamente.
- **Implementação fácil das funcionalidades necessárias:** Por usar o paradigma SDN, podemos desenvolver algoritmos de controle que usam os recursos da rede da melhor forma. A re-escrita de pacotes se torna um procedimento simples e automatizado, em que *DCPortals* pode interagir com OpenStack para obter os parâmetros de configuração de que necessita. Isso simplifica as demandas sobre o hardware convencional no núcleo da rede, ao mesmo tempo que garante o isolamento de tráfego no nível das VMs. Além do mais, isso pode ser feito sem usar os rótulos de VLANs, que os libera para serem usados na implementação de um solução de roteamento multicaminhos baseado em VLANs, por exemplo, o que poderia ser facilmente integrado à nossa solução em *datacenters* que ofereçam redes Ethernet com canais redundantes [Mudigonda et al. 2010].

3. Detalhamento da solução

DCPortals foi implementado como um módulo construído sobre o hipervisor de SDN POX[2]. Ele consulta a base de dados do *OpenStack* diretamente para extrair a informação de que necessita sobre as máquinas virtuais e suas redes virtuais, tais como identificação

[1]http://www.openstack.org

[2]https://openflow.stanford.edu/display/ONL/POX+Wiki

do inquilino, outras VMs em uma dada rede virtual e, especialmente, a localização de cada VM. Com aquela informação, ele monta as mensagens OpenFlow para informar às Open vSwitches como tratar os fluxos de pacotes de/para uma dada VM. OpenStack controla o hipervisor Xen em cada hospedeiro, que configura sua Open vSwitch adequadamente. A figura 1 ilustra as relações entre os módulos.

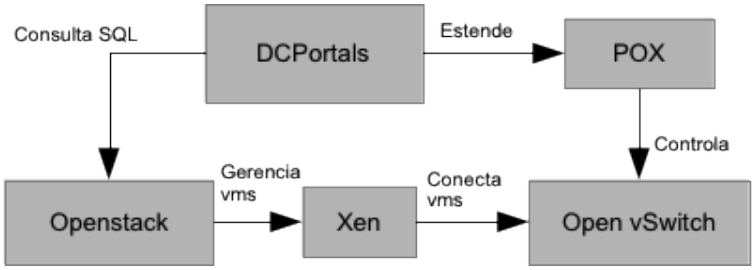

Figura 1. Arquitetura do sistema *DCPortals*.

O administrador do sistema usa a API OpenStack para disparar cada máquina virtual e indicar a rede virtual a que ela pertence. OpenStack seleciona o hospedeiro físico que executará a VM e envia a informação que ela precisa para ser disparada. O hipervisor Xen conecta a nova VM ao Open vSwitch da máquina física. Quando a nova VM envia seu primeiro pacote pela rede, o Open vSwitch identifica um novo fluxo e usa o protocolo OpenFlow para informar ao controlador POX, que extrai a informação sobre o fluxo a partir do pacote recebido. POX notifica o *DCPortals*, que inspeciona o pacote, consulta a base de dados OpenStack e envia outra mensagem OpenFlow de volta para o switch apropriado, dizendo como os demais pacotes daquele fluxo devem ser tratados. Como o fluxo é direcionado a uma VM em algum ponto da rede, *DCPortals* também já programa o Open vSwitch ao qual a VM de destino está conectado para também estar preparado para tratar os pacotes do fluxo. A seção a seguir fornece mais detalhes sobre esse processo e cada um dos aspectos principais do sistema.

3.1. A abstração de rede virtual

No *DCPortals*, a abstração de rede virtual oferece uma representação clara de como é definido o isolamento lógico entre as múltiplas redes de inquilinos que compartilham a mesma infraestrutura, independente da sua organização física. Cada inquilino vê suas máquinas como conectadas a um único *switch* virtual, separado dos demais, como ilustrado na figura 2. Máquinas conectadas àquela rede podem se comunicar livremente e todo seu tráfego é limitado às VMs conectadas ao *switch*, independente da estrutura que as interliga na topologia física. Essa abstração garante ao menos a segurança de uma rede local, sem forçar os inquilinos a se preocupar com a infraestrutura física, compartilhada.

A identificação das máquinas que forma uma dada rede é derivada da chave RSA de cada VM que é parte da configuração do OpenStack. Essa chave é usada para configurar o acesso do administrador a todas as VMs de um usuário por SSH. *DCPortals* assume que todas as VMs que compartilham uma mesma chave RSA pertencem a uma mesma rede (isolada). Tal premissa é razoável, considerando-se que todas as máquinas na rede local de um inquilino deveriam ser acessíveis para aquele inquilino. A chave RSA pode ser considerada um identificador opaco para o *switch* virtual daquela rede. Ao adotar essa

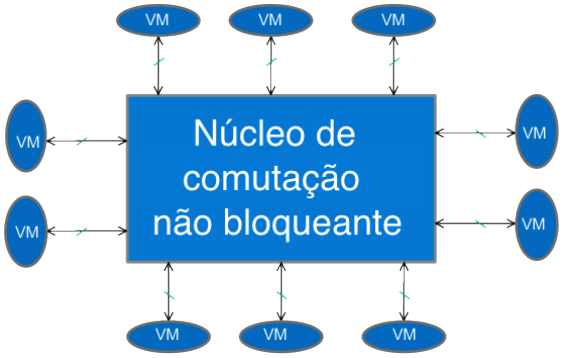

Figura 2. Abstração de rede virtual de cada inquilino.

solução, evitamos forçar cada inquilino a explicitamente fornecer uma descrição da rede, o que exigiria uma extensão do *esquema* da base de dados OpenStack[3]

Para verificar se os pacotes de uma VM A podem alcançar uma outra VM B, tudo que *DCPortals* tem que fazer é consultar a base OpenStack e verificar se as chaves RSA de A e B são iguais. Se esse for o caso, as máquinas estão na mesma rede virtual e tráfego de uma pode alcançar a outra, o que significa que o controlador de rede deve programar as Open vSwitches nas máquinas físicas que hospedam A e B de forma a garantir que pacotes possam fluir entre elas. Caso contrário, se A e B estão em redes diferentes, um pacote de A para B seria descartado pela primeira Open vSwitch no seu caminho.

3.2. Integração dos módulos

DCPortals usa a API do POX para manipular (receber, interpretar, construir e enviar) mensagens OpenFlow que controlam as Open vSwitches na rede, baseado nos fluxos identificados e na localização das VMs associadas. Ele basicamente reage a dois eventos definidos na interface POX, *ConnectionUp*, que é disparado quando um novo *switch* é conectado à rede do *datacenter*, e *PacketIn*, que é lançado quando um novo pacote chega ao controlador vindo de algum *switch*, indicando que um novo fluxo foi identificado e para o qual as regras de encaminhamento ainda não foram definidas. O primeiro evento é usado para construir as estruturas de dados necessárias para acompanhar o estado dos dispositivos, enquanto o segundo é essencial para fornecer as informações necessárias para se implementar o roteamento e isolamento de tráfego.

Como mencionado anteriormente, o sistema consulta a base de dados MySQL do OpenStack para recuperar informações sobre as VMs existentes cada vez que um novo fluxo é observado. Isso inclui, além da informação sobre a chave RSA da VM, a sua localização física e seus endereços IP e MAC. Além disso, *DCPortals* dispara consultas para obter informações sobre os hospedeiros físicos no sistema quando ele precisa identificar o destino de um novo fluxo.

Além da informação obtida do OpenStack, *DCPortals* também tem sua própria base de configuração. Ela é usada para armazenar a informação necessária para acessar a base MySQL, bem como os endereços MAC de todas as máquinas físicas que fazem parte da rede do *datacenter*. Isso é necessário para identificar a interface de controle de

[3]Desde que iniciamos este trabalho, o OpenStack já foi estendido com um módulo especial criado para descrever configurações de rede, o Quantum. Integrar o *DCPortals* a ele é um trabalho futuro previsto.

cada máquina, através da qual as mensagens OpenFlow são enviadas, separadamente da rede que contém o tráfego das VMs dos inquilinos.

3.3. Re-escrita de pacotes para isolamento das redes

Como discutido anteriormente, o uso de re-escrita de pacotes para implementar o isolamento de redes tem dois benefícios principais: garante que o tráfego de um inquilino fique fora do alcance dos demais e reduz a pressão sobre a memória dos dispositivos de rede, que não precisam lidar com os endereços MAC de todas as VMs no *datacenter*. Para conseguir isso, nós re-escrevemos os endereços de enlace (MAC) em todos os pacotes que atravessam uma Open vSwitch na borda da rede para remover a informação das VMs.

Para fazer isso, *DCPortals* considera que cada VM no *datacenter* é criada com um endereço IP único. Essa não é uma restrição séria para os inquilinos já que, por definição, todos os endereços IP válidos devem ser únicos. Se algum inquilino usa endereços IP restritos, é fácil assinalar para ele um segmento em uma faixa como a região 10/8. Nosso objetivo com isso é garantir que, dado um endereço IP na rede do *datacenter*, seja sempre possível identificar a VM exata associada a ele. Fazendo isso, deve ser sempre possível encontrar o endereço MAC correto (origem ou destino) para qualquer pacote, simplesmente verificando o endereço IP correspondente no pacote.

Quando uma máquina virtual VM_1, operando no hospedeiro físico $Host_1$, envia um pacote para outra máquina virtual VM_3 na mesma rede virtual, mas fisicamente localizada no hospedeiro físico $Host_2$, o pacote original enviado por VM_1 que alcança o *switch* virtual no $Host_1$ conterá os endereços MAC de VM_1 e VM_3, bem como os endereços IP de ambos. Para simplificar, consideremos que *DCPortals* já identificou o fluxo associado e programou os *switches* de borda apropriadamente. O Open vSwitch no $Host_1$, então, vai substituir os endereços MAC de VM_1 e VM_3 no cabeçalho Ethernet do pacote com os endereços MAC de $Host_1$ e $Host_2$, respectivamente. O pacote resultante é que atravessará o núcleo da rede, ainda carregando os endereços IP de VM_1 e VM_3. Entretanto, os endereços MAC que serão usados nas tabelas de aprendizado e encaminhamento dos *switches* no núcleo serão apenas aqueles dos hospedeiros físicos. Quando o pacote alcança $Host_2$, ele atravessará o *switch* virtual ali configurado, Nesse momento, ele verificará os endereços IP no pacote e re-escreverá os endereços MAC de VM_1 e VM_3 no cabeçalho. Esse é o pacote que será entregue a VM_3 nesse ponto. Note-se que que os endereços MAC das máquinas virtuais nunca cruzaram a rede física; mesmo assim, os pacotes só alcançaram destinos para os quais o sistema pode atestar sua conectividade segundo alguma rede virtual.

A figura 3 ilustra o processo para um pacote em um fluxo que já foi identificado e teve regras de re-escrita propagadas para os *switches* de borda apropriados, exatamente como discutido no parágrafo anterior. É importante lembrar que esse processo usa os endereços IP no pacote para garantir que os cabeçalhos originais possam ser recompostos. Isso significa que *DCPortals* só funciona com tráfego IP no momento. Não há, entretanto, aplicações relevantes em *datacenters* que não sejam baseadas em IP.

3.4. Tratamento de mensagens ARP e DHCP

A técnica de re-escrita de MACs descrita substitui endereços Ethernet (MAC) das máquinas virtuais em pacotes que já haviam sido construídos com aqueles endereços. Entretanto, para as VMs construírem aqueles pacotes pela primeira vez, elas devem aprender

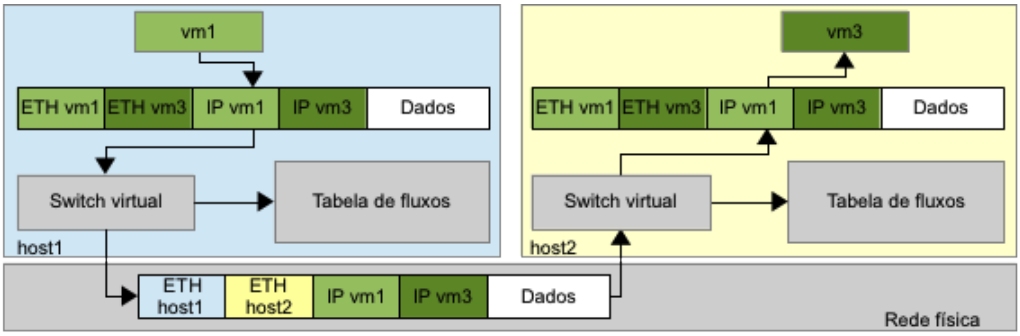

Figura 3. Re-escrita de cabeçalhos no *DCPortals*

o endereço MAC do destinatário. Em uma rede tradicional, isso seria conseguido através de uma mensagem de *broadcast* usando o protocolo ARP (*Address Resolution Protocol*). O protocolo se baseia em dois tipos de mensagens: *ARP request* e *ARP reply*. O primeiro é enviado para o endereço de *broadcast* da rede quando se precisa descobrir o endereço MAC associado a um certo endereço IP que se deseja contactar. O segundo é a resposta, enviada pela máquina alvo, informando seu endereço MAC.

Para o isolamento da rede virtual funcionar, não é aceitável que essas mensagens de *broadcast* viagem pela rede carregando os endereços de enlace das máquinas virtuais. *DCPortals* corrige esse problema interceptando toda comunicação ARP nos *switches* de borda e tratando-as diretamente. Já que o sistema tem acesso à base de dados do OpenStack, ele sabe como responder qualquer consulta ARP na rede. Tudo que ele precisa fazer é construir a mensagem de *ARP reply* apropriada e entregá-la diretamente à VM que iniciou uma consulta. Como as consultas são interceptadas no *switch* mais próximo do transmissor, as mensagens ARP nunca atravessam o núcleo da rede.

O protocolo DCHP, usado durante a configuração das máquinas virtuais na rede, também segue um padrão de funcionamento semelhante. Da mesma forma que no ARP, suas mensagens podem ser interceptadas e tratadas diretamente pelo *DCPortals* quando isso for interessante para o sistema.

3.5. Tratamento de outros *broadcasts*

Apesar da maioria das mensagens de *broadcast* em redes locais ser relacionada ao ARP ou ao DCHP, ainda devemos considerar como outras mensagens desse tipo serão tratadas (por exemplo, algum *broadcast* UDP gerado por uma aplicação). Quando um grupo de VMs é configurada em uma rede virtual, quaisquer pacotes desse tipo que ainda existam na rede devem ser entregues a todas as máquinas configuradas na mesma rede virtual — e apenas a elas. Entretanto, em um ambiente compartilhado complexo como uma rede de *datacenter*, esse não é o caso, já que tais pacotes seriam entregues a todas as máquinas conectadas à rede Ethernet física.

A figura 4 mostra a diferença entre o que acontece nesse caso em uma rede com e sem *DCPortals*. Na figura, máquinas virtuais de mesma cor representam máquinas em uma mesma rede virtual. VM_2 envia uma mensagem de *broadcast*. Idealmente, aquela mensagem deveria ser entregue apenas às outras máquinas na mesma rede virtual, VM_1, VM_4 e VM_8. Sem *DCPortals*, entretanto, todas as máquinas, independente de sua rede virtual, receberiam o pacote.

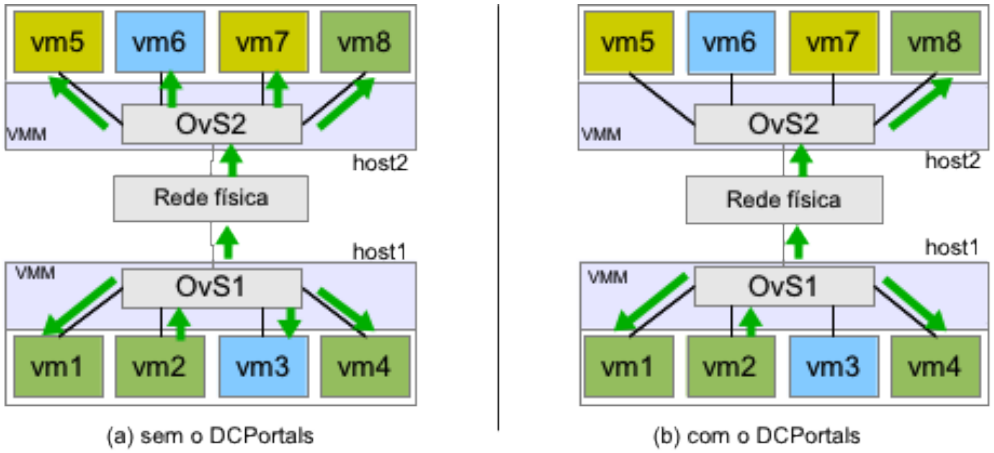

Figura 4. Diferença no tratamento de um *broadcast*.

Para conseguir o efeito desejado, primeiro o pacote de *broadcast* é inspecionado pelo *switch* virtual local ao hospedeiro que executa a VM transmissora. Ele é entregue, então, aos portos do *switch* que estejam conectados a outras VMs da mesma rede virtual e é enviado também pela interface externa real, se há outras VMs da mesma rede em outros hospedeiros físicos. Essa última mensagem alcançará a interface física de todos os hospedeiros da rede, como qualquer mensagem de *broadcast*. Dessa forma, ele atingirá todas as Open vSwitches na borda da rede. Cada uma delas verificará se existem outras VMs locais pertencentes à rede virtual de origem do pacote e entregarão o pacote apenas a tais VMs. No caso do exemplo da figura, o *switch* OvS_1 só entregará o pacote pelas portas conectadas às máquinas VM_1 VM_4, e à rede física.

Quando o pacote deixar a máquina hospedeira do transmissor, o processo de re-escrita de MAC age como anteriormente descrito, para remover o endereço MAC do transmissor, mas mantém o endereço de *broadcast* Ethernet como destino. Isso garantirá que todos os *switches* convencionais da rede física propagarão o pacote para toda a borda. Quando a mensagem de *broadcast* alcança um *switch* virtual da borda, o controlador recompõe o pacote original antes de entregá-lo às máquinas da mesma rede virtual. No nosso exemplo, o controlador comandaria o *switch* OvS_2 para entregar o pacote apenas pela porta conectada à VM_8.

Agindo assim, as mensagens continuam a ter o efeito de *broadcast*, mas serão entregues apenas às máquinas que realmente tenham sido configuradas como fazendo parte da rede virtual do transmissor, garantindo que nenhuma outra VM terá acesso a elas.

3.6. Ligação da rede virtual interna ao *datacenter* com a Internet

Na maioria dos casos, é necessário para cada máquina de uma aplicação em nuvem ter acesso ao mundo externo (à Internet), às vezes apenas para que o inquilino possa acessar e controlar suas máquinas, outras vezes para que todas possam oferecer uma interface para um serviço distribuído entre elas, como uma aplicação Web replicada, por exemplo. Os detalhes de como tais conexões são definidas dependem pesadamente da estrutura de cada *datacenter* e da sua política de serviço. Por exemplo, elas podem ser possíveis apenas

Máquina virtual	IP	Rede Virtual	Host físico
vm2	10.0.20.2	lan1	host1
vm3	10.0.20.3		
vm4	10.0.20.4		host2
vm5	10.0.20.5	lan2	
vm6	10.0.20.6		

Tabela 1. Distribuição de máquinas virtuais e redes entre as máquinas físicas usadas nos experimentos.

pela definição de uma segunda interface de rede em uma das máquinas virtuais, que seria a única máquina visível externamente e que passaria a agir como *gateway* entre as redes, responsável por rotear as mensagens entre a rede interna, virtual, e a rede externa.

Em sua implementação atual, *DCPortals* não requer que os inquilinos configurem uma máquina para lidar com o roteamento entre os dois domínios: ele seleciona uma máquina do *datacenter*, que pode ser a mesma responsável por executar o controlador, para rotear o tráfego entre a rede virtual e o restante da Internet. Essa máquina é capaz de "ver" todas as redes internas e tem regras específicas para controlar o tráfego que flui por ela (por exemplo, permitindo comunicação entre as redes de dois inquilinos de uma forma bem controlada). Dada a natureza flexível do paradigma SDN, esse *gateway* pode ser facilmente estendido para implementar regras de *firewall*, por exemplo.

4. Avaliação

Para realizar os experimentos de validação e confirmar a operação correta do sistema, utilizamos três máquinas, cada uma com duas interfaces de rede, conectadas a dois *switches* diferentes. Uma das redes resultantes foi usada para o tráfego de gerência (comandos OpenStack e OpenFlow) e a outra foi usada para a comunicação entre as VMs (rede operacional). Esse tipo de configuração é bastante usual em *datacenters* comerciais [Greenberg et al. 2009].

A rede de gerência usava endereços IP na faixa 10.0.254/24. As máquinas virtuais foram configuradas com duas redes virtuais isoladas, cada uma cobrindo duas máquinas físicas. Uma mesma faixa de endereços foi usada para todas as VMs, para acentuar a necessidade de isolamento de tráfego: sendo configuradas com endereços na mesma faixa, a não ser que algum mecanismo externo de isolamento esteja ativo, tráfego de cada VM poderia ser direcionado para qualquer outra máquina virtual. A distribuição de endereços entre as máquinas e de máquinas entre as redes é ilustrada na tabela 1.

Hospedeiros foram configurados com o sistema operacional Ubuntu versão 11.10 e os pacotes OpenStack, Xen e Open vSwitch obtidos dos repositórios oficiais. As máquinas virtuais foram configuradas com uma das imagens padrão do OpenStack, executando Ubuntu 10.10. Mais detalhes sobre a configuração desses sistemas está disponível em outro documento [Nunes 2012].

Usando esse ambiente nós executamos dois experimentos: o primeiro foi um teste de isolamento simples usando `ping` para o endereço de *broadcast* da rede e avaliação da interferência do sistema na latência de comunicação entre as VMs; o segundo testou o sistema sob um ataque de negação de serviço.

4.1. Verificação de isolamento e latência

O experimento consistiu em usar o comando `ping` para enviar uma mensagem *ICMP Request* para o endereço de *broadcast* da rede local a partir de uma das máquinas virtuais. O comportamento esperado para esse uso do `ping` é que cada máquina na mesma rede do transmissor deve responder com uma mensagem *ICMP Reply*, mesmo o transmissor[4]. Na saída gerada pelo `ping`, as múltiplas respostas para a mesma consulta devem aparecer com uma observação "DUP", já que no comportamento usual do programa elas são consideradas duplicatas (apesar das origens diversas). Com o *DCPortals*, mesmo com todas as máquinas configuradas na mesma faixa de endereços e compartilhando a mesma rede física, apenas VMs da mesma rede virtual do transmissor devem receber uma mensagem de consulta e, consequentemente, apenas elas devem responder. Nós iniciamos o comando `ping` de VM_2, na rede 1, e de VM_5, na rede 2. Sem *DCPortals*, ambas receberam respostas de todas as máquinas:

```
vm2:
    PING 10.0.20.255 (10.0.20.255) 56(84) bytes of data.
    64 bytes from 10.0.20.2: icmp_req=1 ttl=64 time=0.027 ms
    64 bytes from 10.0.20.4: icmp_req=1 ttl=64 time=2.60 ms (DUP!)
    64 bytes from 10.0.20.3: icmp_req=1 ttl=64 time=3.21 ms (DUP!)
    64 bytes from 10.0.20.6: icmp_req=1 ttl=64 time=3.22 ms (DUP!)
    64 bytes from 10.0.20.5: icmp_req=1 ttl=64 time=8.47 ms (DUP!)

vm5:
    PING 10.0.20.255 (10.0.20.255) 56(84) bytes of data.
    64 bytes from 10.0.20.5: icmp_req=1 ttl=64 time=0.027 ms
    64 bytes from 10.0.20.6: icmp_req=1 ttl=64 time=1.00 ms (DUP!)
    64 bytes from 10.0.20.3: icmp_req=1 ttl=64 time=1.86 ms (DUP!)
    64 bytes from 10.0.20.2: icmp_req=1 ttl=64 time=5.91 ms (DUP!)
    64 bytes from 10.0.20.4: icmp_req=1 ttl=64 time=10.5 ms (DUP!)
```

Isso confirma que realmente todas as máquinas são alcançáveis a partir das demais, apesar da configuração desejada em redes isoladas. Cada máquina recebeu uma cópia da mensagem de *broadcast* através da rede Ethernet compartilhada e respondeu diretamente ao transmissor.

Quando ativamos o *DCPortals* os resultados mudam, deixando claro que nem todas as máquinas foram alcançadas por cada mensagem de *broadcast*. A VM_2 recebeu apenas respostas das máquinas configuradas em sua própria rede virtual (inclusive de si própria), o mesmo acontecendo com VM_5. Isso confirma que, apesar de ocuparem a mesma faixa de endereços e compartilharem a mesma rede física, as máquinas em cada rede virtual foram isoladas corretamente.

```
vm2:
    PING 10.0.20.255 (10.0.20.255) 56(84) bytes of data.
    64 bytes from 10.0.20.2: icmp_req=1 ttl=64 time=0.026 ms
    64 bytes from 10.0.20.3: icmp_req=1 ttl=64 time=66.7 ms (DUP!)
    64 bytes from 10.0.20.4: icmp_req=1 ttl=64 time=88.5 ms (DUP!)

vm5:
    PING 10.0.20.255 (10.0.20.255) 56(84) bytes of data.
    64 bytes from 10.0.20.5: icmp_req=1 ttl=64 time=0.032 ms
    64 bytes from 10.0.20.6: icmp_req=1 ttl=64 time=103 ms (DUP!)
```

Traces de pacotes coletados em algumas das VMs (por consequência, dentro das suas redes virtuais) e na interface de rede das máquinas físicas (já no ponto em que pacotes

[4]Para esse experimento, o sistema operacional das VMs foi configurado para permitir mensagens ICMP para endereços de *broadcast*.

entram no núcleo da rede, após os *switches* virtuais da borda) confirmam o isolamento (detalhes omitidos por limitações de espaço).

O aumento dos tempos do ping com o *DCPortals* representam o custo de instalação do fluxo entre as duas VMs pelo sistema. Realizamos testes para avaliar o *overhead* da operação do controlador SDN, tanto durante o estabelecimento de um fluxo quando durante a transmissão de dados por um fluxo estabelecido. Houve, em média, um *overhead* de 7 ms em consultas ARP e 47 ms para o estabelecimento de um fluxo. Tais overheads estão limitados aos primeiros pacotes de um fluxo apenas. Uma vez estabelecida a entrada na tabela de encaminhamento, não houve diferenças significativas entre os dois sistemas.

4.2. Proteção contra ataques de negação de serviço

Uma motivação comum para a demanda de isolamento entre redes virtuais é a ameaça de que um inquilino possa iniciar um ataque de negação de serviço direcionado às máquinas de um outro inquilino. Em um ambiente de *datacenter* onde não há esse isolamento, um fluxo UDP criado de uma máquina atacante para uma rede alvo pode consumir banda da rede a ponto de fazer com que o serviço do inquilino atacado deixe de ser acessível. Um caso reportado semelhante, apesar da origem ter sido externa, ocorreu com o serviço BitBucket, enquanto ele usava a infraestrutura Amazon EC2 [BitBucket Attack 2012].

Tal problema não deveria ocorrer se as redes virtuais dos inquilinos fossem adequadamente isoladas, como é nosso objetivo com *DCPortals*. Para comprovar isso, neste experimento criamos um ataque UDP, direcionado a outra rede virtual. Para tornar o problema ainda mais grave, o fluxo UDP foi criado para o endereço de *broadcast* da rede destino, visando atingir todas as máquinas. A figura 5 mostra a configuração usada neste caso. Máquinas virtuais VM_2 e VM_3, localizadas no hospedeiro físico $host_1$ estabelecem uma transferência TCP entre eles. Ao mesmo tempo, VM_5, o atacante, inicia um fluxo UDP endereçado para o endereço de *broadcast* da rede. Usamos `iperf` para gerar os dois fluxos e medimos a vazão efetiva da conexão TCP entre VM_2 e VM_3. Limitamos a banda máxima de cada VM a 1 Gbps, um valor usual, equivalente à tecnologia da rede física mais comum atualmente.

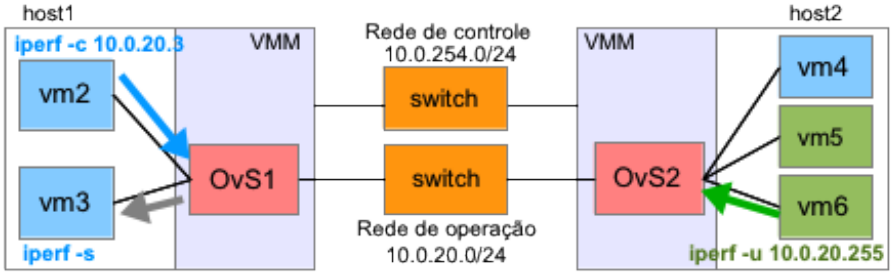

Figura 5. Experimento de negação de serviço. VM_6 inicia um alagamento UDP para o endereço de *broadcast* da rede, enquanto VM_2 e VM_3 (em outra rede virtual) estabelecem um fluxo TCP entre elas.

Cada teste durou 1.000 segundos e a vazão TCP foi medida a cada 3 segundos. Os primeiros 3 segundos foram descartados para eliminar ruídos devido a variações no disparo dos fluxos. Resultados são mostrados na tabela 2.

Cenário	Banda (Mbps)
Sem isolamento, sem ataque	928,29 ± 0,16
DCPortals, sem ataque	928,21 ± 0,22
Sem isolamento, sob ataque	41,21 ± 12,99
DCPortals, sob ataque	910,11 ± 0,28

Tabela 2. Vazão média do fluxo TCP observada em diferentes condições; erro considera intervalo de confiança de 99%.

Claramente, podemos ver a diferença entre os dois casos. O sistema sem isolamento sofre uma perda de aproximadamente 95% da vazão observada. Por outro lado, *DCPortals* sofre apenas uma perda de 5%. Com o isolamento, o fluxo UDP é bloqueado nos *switches* de borda, não sendo entregue às outras redes. A perda nesse caso é devida ao *overhead* no *switch* de borda para descartar os pacotes UDP que chegam. Considerando o sistema sem ataques, verificamos que não há diferença estatística entre os dois casos. Isso é o esperado, considerando-se a observação anterior sobre o baixo *overhead* durante a duração dos fluxos.

4.3. Escalabilidade

Durante os experimentos não tivemos recursos para realizar testes de escalabilidade extensos. Entretanto, acreditamos que a solução proposta tem boa escalabilidade. Em primeiro lugar, a operação do sistema e o encaminhamento dos fluxos já identificados reside nas Open vSwitches em cada máquina física que, por construção, estão entre os comutadores virtuais mais eficientes na atualidade [Pfaff et al. 2009]. Por outro lado, o controlador POX é capaz de processar mais de 30.000 fluxos por segundo, mais que suficiente para as demandas do *DCPortals* em um grande *datacenter*, com até dezenas de milhares de máquinas[5].

5. Trabalhos relacionados

Greenberg *et al.* [Greenberg et al. 2009] apresentam um interessante estudo sobre custos de um *datacenter* para computação em nuvem. Entre outras observações, eles identificam a rede como um dos principais desafios nesse contexto. Em particular, mencionam explicitamente a dependência as soluções atuais em relação a VLANs e os problemas associados a essa prática.

Uma das primeiras iniciativas visando uma técnica de isolamento de redes virtuais foi desenvolvida por um grupo do HP Labs [Cabuk et al. 2007], começando com a definição de domínios virtuais confiáveis (*Trusted Virtual Domains*, TVDs). Estes seriam seções de rede logicamente isoladas, de forma independente da topologia da infraestrutura. Para implementar esse isolamento, os autores criaram um módulo interno às máquinas virtuais que é responsável por todo o processamento relacionado ao isolamento de rede. Duas técnicas de isolamento, rótulos de VLANs e encapsulamento EtherIP, são comparadas. Aquele trabalho tem um objetivo semelhante ao *DCPortals*, mas as soluções consideradas têm limitações de escalabilidade e exigem alterações intrusivas no hipervisor (Xen). Um estudo comparativo mais longo do mesmo grupo menciona a técnica de re-escrita de endereços MAC no mesmo contexto das duas outras [Cabuk et al. 2008].

[5]http://www.noxrepo.org/pox/about-pox/

DCPortals usa o paradigma de redes definidas por software para resolver o problema de isolamento. Pettit *et al.* [Pettit et al. 2010] já discutiram a viabilidade desse tipo de enfoque para redes de *datacenters*, mas não apresentaram uma solução concreta. Duas aplicações do controlador SDN NOX, a *datacenters* foram apresentadas anteriormente, mas elas foram focadas na implementação de novas arquiteturas de rede [Tavakoli et al. 2009, Heller et al. 2010]. Diferente dessas soluções, *DCPortals* não exigem hardware com recursos OpenFlow no núcleo da rede e foca especificamente no isolamento de tráfego.

Um trabalho com motivações bastante similares ao aqui apresentado é certamente Netlord, desenvolvido por Mudigonda *et. al.* [Mudigonda et al. 2011]. Naquele trabalho, os autores usam uma solução baseada em tunelamento para obter isolamento sem exigir hardware especial na rede. Entretanto, a forma como aquela solução é implementada é bem diferente, ao usar uma extensão do hipervisor Xen especialmente desenvolvida para esse fim. Nós acreditamos que o uso de redes definidas por software é um enfoque mais elegante e flexível, sendo uma característica determinante do nosso trabalho. Isso simplifica a implementação e oferece um leque mais amplo de possibilidades de aplicação. *DCPortals*, por exemplo, funciona diretamente não apenas com Xen, mas com qualquer outro hipervisor que use a biblioteca libvirt e Open vSwitch, como é o caso do KVM.

6. Conclusão

Este trabalho descreve *DCPortals*, um sistema desenvolvido para fornecer isolamento de tráfego a redes virtuais em um ambiente de *datacenter* virtualizado. A arquitetura do sistema e os detalhes de implementação foram apresentados, bem como resultados que confirmam o isolamento oferecido, inclusive no contexto de um ataque de negação de serviço entre inquilinos. Avaliações também quantizaram o *overhead* durante o estabelecimento de fluxos, que são visíveis mas raros, e mostraram que durante operação normal os custos por fluxo são desprezíveis.

Como trabalhos futuros, continuamos melhorando o sistema; considerando sua integração com OpenStack, pretendemos estudar sua integração com Quantum, o componente de OpenStack recentemente anunciado para integração com controladores OpenFlow. Pretendemos também trabalhar na integração de *DCPortals*, que oferece isolamento de redes, com *Gatekeeper*, um sistema projetado para oferecer garantias de qualidade de serviço em uma rede de *datacenter* [Rodrigues et al. 2011].

Agradecimentos

Este trabalho foi parcialmente financiado pelo UOL (www.uol.com.br), através do programa UOL Bolsa Pesquisa, Fapemig, CNPq e Instituto Nacional de Ciência e Tecnologia da Web, InWeb (MCT/CNPq 573871/2008-6).

Referências

BitBucket Attack (2012). Bitbucket amazon ddos attack. `http://blog.bitbucket.org/2009/10/04/on-our-extended-downtime-amazon-and-whats-coming/`. Acessado em julho de 2012.

Cabuk, S., Dalton, C. I., Eduards, A. e Fischer, A. (2008). A comparative study on secure network virtualization. Technical Report HPL-2008-57, HP Laboratories.

Cabuk, S., Dalton, C. I., Ramasamy, H. e Schunter, M. (2007). Towards automated provisioning of secure virtualized networks. In *Proceedings of the 14th ACM conference on Computer and communications security*, CCS '07, págs. 235–245, New York, NY, USA. ACM.

Casado, M., Koponen, T., Ramanathan, R. e Shenker, S. (2010). Virtualizing the network forwarding plane. In *Proceedings of the Workshop on Programmable Routers for Extensible Services of Tomorrow*, PRESTO '10, págs. 8:1–8:6, New York, NY, USA. ACM.

Greenberg, A., Hamilton, J., Maltz, D. A. e Patel, P. (2009). The cost of a cloud: research problems in data center networks. *SIGCOMM Computer Communication Review*, 39(1):68–73.

Heller, B., Erickson, D., McKeown, N., Griffith, R., Ganichev, I., Whyte, S., Zarifis, K., Moon, D., Shenker, S. e Stuart, S. (2010). Ripcord: a modular platform for data center networking. *SIGCOMM Comput. Commun. Rev.*, 40:457–458.

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

Mudigonda, J., Yalagandula, P., Al-Fares, M. e Mogul, J. C. (2010). Spain: Cots data-center ethernet for multipathing over arbitrary topologies. In *Proceedings of the 7th USENIX conference on Networked systems design and implementation*, NSDI'10, págs. 1–16, Berkeley, CA, USA. USENIX Association.

Mudigonda, J., Yalagandula, P., Mogul, J., Stiekes, B. e Pouffary, Y. (2011). Netlord: a scalable multi-tenant network architecture for virtualized datacenters. In *Proceedings of the ACM SIGCOMM 2011 conference*, SIGCOMM '11, págs. 62–73, New York, NY, USA. ACM.

Nunes, R. V. (2012). Uma aplicação de redes definidas por software para a gerência de redes de datacenters virtualizados. Master's thesis, DCC/UFMG.

Pettit, J., Gross, J., Pfaff, B., Casado, M. e Crosby, S. (2010). Virtual switching in an era of advanced edges. In *Proceedings of the 2nd Workshop on Data Center - Converged and Virtual Ethernet Switching (DC CAVES)*, DC CAVES, págs. 1–7, Amsterdam, The Netherlands. ITC.

Pfaff, B., Pettit, J., Koponen, T., Amidon, K., Casado, M. e Shenker, S. (2009). Extending networking into the virtualization layer. In *8th ACM Workshop on Hot Topics in Networks (HotNets-VIII)*.

Rodrigues, H., Soares, P., Santos, J. R., Turner, Y. e Guedes, D. (2011). Isolamento de tráfego em ambientes virtualizados. In *Anais do XXIX Simpósio Brasileiro de Redes de Computadores e Sistemas DIstribuídos (SBRC)*, págs. 1–14. SBC.

Tavakoli, A., Casado, M., Koponen, T. e Shenker, S. (2009). Applying NOX to the datacenter. In *Proceedings of workshop on Hot Topics in Networks (HotNets-VIII)*.