XNetMon: Uma Arquitetura com Segurança para Redes Virtuais

Natalia Castro Fernandes e Otto Carlos Muniz Bandeira Duarte

[1]Grupo de Teleinformática e Automação (GTA/PEE/UFRJ)
Universidade Federal do Rio de Janeiro (UFRJ) – Rio de Janeiro, RJ – Brasil

{natalia,otto}@gta.ufrj.br

Abstract. *Isolation is essential to secure any virtualized environment sharing a common resource, and virtual networks are no different. Resource sharing with isolation prevents malicious virtual routers from consuming all physical resources and disturbing the performance of other virtual networks sharing the same machine. We propose a new architecture for Xen that provides isolation during the access of shared resources. A secure mechanism is designed to monitor the access of shared resources and to punish virtual routers that misbehave, guaranteeing an isolated operation of the virtual networks. In order to secure the control of virtual networks, we propose a communication protocol between the virtual routers and the administrative domain that prevents malicious virtual routers from affecting the forwarding table of other virtual routers. We developed a prototype and our experiments show that the proposed architecture guarantees the availability of the virtual-network control service, and provides a better resource sharing than known mechanisms, allowing complete isolation among virtual networks.*

Resumo. *O isolamento entre máquinas virtuais é uma propriedade essencial para a segurança dos ambientes virtualizados, em especial, na aplicação de redes virtuais. Um compartilhamento dos recursos provido com isolamento impede que roteadores virtuais com um comportamento malicioso utilizem todos os recursos da máquina física, prejudicando o desempenho das outras redes virtuais. Esse trabalho propõe uma nova arquitetura para o Xen, capaz de prover o isolamento das redes virtualizadas. É proposto um mecanismo de segurança que monitora o uso dos recursos compartilhados e pune os roteadores virtuais com comportamentos maliciosos, garantindo assim a operação isolada das redes virtuais. Para garantir também a segurança no controle do encaminhamento dos dados, é proposto um protocolo para comunicação entre o domínio de gerenciamento e os roteadores virtuais que impede que roteadores virtuais maliciosos modifiquem as tabelas de encaminhamento de outros roteadores virtuais. Foi desenvolvido um protótipo e os testes mostram que a arquitetura proposta garante a disponibilidade do serviço de controle das redes virtuais, além de promover o compartilhamento dos recursos com mais precisão que outros mecanismos da literatura, propiciando o isolamento das redes virtuais.*

1. Introdução

A tecnologia de virtualização permite que diversos ambientes virtuais sejam criados sobre o mesmo substrato físico. O Xen [Egi et al. 2007, Laureano and Maziero 2008] é uma plataforma de virtualização muito utilizada para criação de máquinas virtuais, que são ambientes virtuais que simulam uma máquina física, com *hardware*, sistema operacional e aplicações próprios. A virtualização tem sido proposta como base de um novo

paradigma de redes de computadores denominado redes virtuais. Neste modelo, um roteador físico é compartilhado por diferentes roteadores (por *software*) virtuais. Assim, um computador pessoal com Xen pode ser usado para compartilhar diferentes roteadores virtuais com os seus próprios sistemas operacionais e cada um rodando, em paralelo, a sua própria pilha de protocolos. Portanto, a virtualização viabiliza um novo conceito em redes que permite um amplo suporte a inovações. É possível criar e destruir, por demanda, redes virtuais com pilhas de protocolos diferentes, o que permite a coexistência de redes específicas que atendam sob medida as exigências dos usuários [Fernandes et al. 2010].

O isolamento e o desempenho são características fundamentais em um ambiente de redes virtuais. O isolamento garante que as redes operem de forma independente, o que é primordial para a segurança nos ambientes virtualizados, porque impede que algum roteador virtual malicioso ou com falhas interfira no funcionamento das demais redes virtuais. O desempenho no encaminhamento de pacotes também é importante, porque a maior flexibilidade provida pelos roteadores virtuais não deve implicar em uma redução nas taxas de encaminhamento. No entanto, a plataforma Xen apresenta deficiências tanto de isolamento quanto de desempenho em suas operações de Entrada/Saída (E/S) para comunicação em rede [Han et al. 2009, Egi et al. 2007, Fernandes et al. 2010]. Com isso, a segurança é comprometida, pois a troca de mensagens entre as máquinas virtuais e as máquinas externas não é feita de forma completamente isolada, e a flexibilidade dos ambientes virtuais causa um custo alto no desempenho da rede.

Este trabalho tem como objetivo propor uma nova arquitetura de virtualização no Xen, chamada de XNetMon, que garante o isolamento entre as redes virtuais e um alto desempenho no encaminhamento de pacotes. O XNetMon é baseado no paradigma de roteamento virtual com separação de planos [Wang et al. 2008, Pisa et al. 2010], no qual o encaminhamento de dados é desacoplado dos mecanismos de controle da rede. Assim, o controle do roteamento é realizado dentro do ambiente da máquina virtual, o que garante a flexibilidade nos mecanismos de controle, enquanto que o encaminhamento de pacotes é feito em um domínio privilegiado, chamado de Dom0, com acesso direto ao *hardware* físico, o que lhe confere um desempenho no encaminhamento similar a um Linux nativo. O XNetMon propõe mecanismos para tornar a separação de planos automática e segura dentro da plataforma Xen, como o controlador que monitora o uso dos recursos do domínio privilegiado por cada rede virtual, garantindo que nenhuma rede pode interferir no funcionamento das demais. O controlador proposto permite uma alocação de recursos diferenciada para cada rede virtual, possibilitando que o administrador atribua diferentes prioridades e quantidades de recursos de memória, CPU e banda passante do domínio privilegiado às redes virtuais.

Para analisar a proposta, foi desenvolvido um protótipo. Os testes realizados mostram que o XNetMon provê um compartilhamento adequado dos recursos, isolando as redes virtuais. De fato, o XNetMon consegue garantir alta disponibilidade no processo de separação de planos com segurança, além de reduzir atrasos no encaminhamento em situações de sobrecarga do domínio privilegiado em mais de oito vezes. A proposta é mais eficiente no controle do compartilhamento de enlaces do que outros mecanismos da literatura, reduzindo os erros na divisão dos recursos em mais de três vezes.

O restante do artigo está dividido da seguinte forma. Na Seção 2, é descrito o modelo de uma rede virtual utilizando-se o Xen e, na Seção 3, é descrito o modelo do atacante. Na Seção 4, é descrita a proposta e, na Seção 5, são descritos o protótipo desenvolvido e os resultados obtidos. Por fim, na Seção 6, são descritos trabalhos relacionados, enquanto que, na Seção 7, são apresentadas as conclusões do artigo.

2. A Arquitetura Xen e o Modelo de Redes Virtuais

O Xen é uma das ferramentas de virtualização mais utilizadas. A arquitetura do Xen (Figura 1) é composta pelo hipervisor, pelas máquinas virtuais, chamadas de domínios sem privilégios (DomU), e por uma máquina virtual privilegiada, chamada de Domínio 0 (Dom0). O hipervisor controla o acesso aos recursos físicos e trata as operações de entrada e saída (E/S) realizadas pelos domínios, isolando os ambientes virtualizados. O Dom0 é um domínio de *drivers* e um domínio administrativo do sistema Xen, que possui acesso direto ao *hardware*. Por ser um domínio de *drivers*, o Dom0 possui todos os *drivers* reais dos dispositivos físicos em uso. Dessa forma, o Dom0 é responsável por fazer o interfaceamento entre os *drivers* virtuais, localizados nos DomUs, e os dispositivos físicos. Além disso, o Dom0 também é a interface de gerenciamento entre o hipervisor e o usuário. Assim, é através do Dom0 que o usuário pode criar máquinas, modificar parâmetros e gerenciar o funcionamento do Xen.

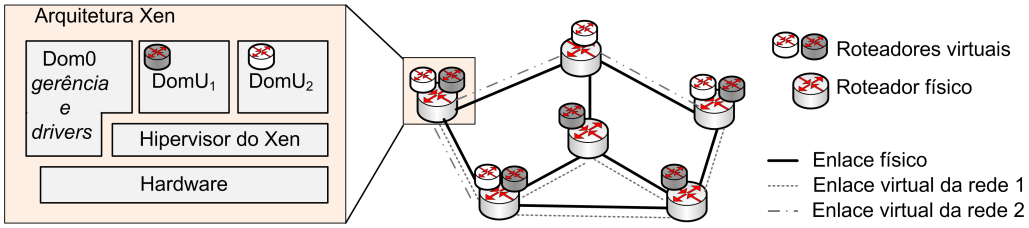


Figura 1. Arquitetura do Xen com duas redes virtuais. Cada máquina virtual é um roteador virtual e, nesse exemplo, cada roteador pertence a uma rede virtual diferente.

O envio e a recepção de pacotes são operações de E/S, o que demanda o uso de *drivers*, e, portanto, passam pelo Dom0. Dessa forma, todas as operações de rede dos DomUs geram uma sobrecarga tanto em memória quanto em CPU para o Dom0, tornando os recursos do Dom0 um possível ponto de estrangulamento quando sobrecarregado em uso de memória ou de CPU. Isto resulta em uma importante vulnerabilidade do Xen, pois, caso os recursos do Dom0 fiquem escassos, então todas as máquinas virtuais terão seu desempenho reduzido, uma vez que o Dom0 é um domínio compartilhado por todas as máquinas virtuais sem que exista algum mecanismo de isolamento eficiente. Esta falta de isolamento é uma falha grave de segurança, que pode ser explorada por redes (DomUs) maliciosas buscando afetar o desempenho de outras redes ao exaurir os recursos do Dom0.

A Tabela 1 mostra o consumo de CPU do Dom0 medido com a ferramenta Top ao se realizar operações de rede na plataforma Xen[1], assumindo um intervalo de confiança (IC) de 95% nas medidas. Observa-se que as operações de transferência de dados entre dois DomUs e também entre o DomU e o Dom0 são as mais custosas para o Dom0. Os resultados comprovam que uma ação maliciosa em um DomU pode facilmente exaurir os recursos do Dom0 e assim comprometer o desempenho de todos os outros domínios. Um dos objetivos da proposta é impedir que qualquer operação executada em uma rede virtual influencie o ambiente das outras redes virtuais, garantindo o isolamento entre as redes.

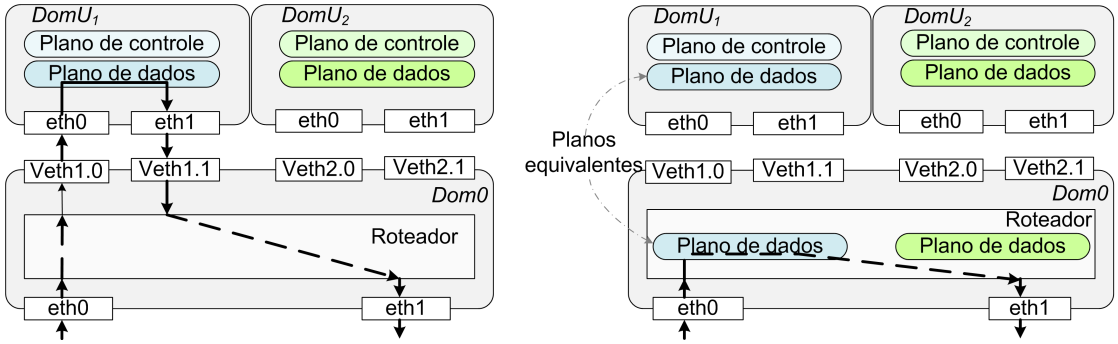
Uma alternativa para reduzir o volume de tráfego que passa pelas máquinas virtuais e para aumentar o desempenho no encaminhamento é o paradigma da separação de planos, que desacopla o plano de dados, responsável pelo encaminhamento, do plano de controle. A separação de planos é feita transferindo-se o encaminhamento de pacotes da

[1]Os testes são relativos a uma máquina com processador Intel core 2 quad com 4GB de memória RAM e Xen 3.4-amd64. As duas máquinas virtuais são configuradas com uma CPU virtual cada e 128 MB de memória, enquanto o Dom0 é configurado com uma CPU virtual e sem restrições de memória. Além disso, cada CPU virtual é associada a uma única CPU física de uso exclusivo. Os testes foram feitos utilizando-se a ferramenta Iperf (http://sourceforge.net/projects/iperf/).

Tabela 1. Consumo de CPU no Dom0 devido a operações nas máquinas virtuais

Uso de CPU (%)	IC (95%)	Descrição
0,71	0,60	Consumo básico de CPU do $Dom0$
66,43	8,93	Transferência de dados com TCP de $DomU$ para $Dom0$
85,49	5,91	Transferência de dados com TCP de $DomU_1$ para $DomU_2$
2,52	0,85	Transferência de dados com TCP de máq. externa para $Dom0$
1,79	1,01	Transferência de dados com TCP de máq. externa para $DomU$
1,20	0,65	Transferência de dados com TCP de $DomU$ para máq. externa

máquina virtual para o Dom0, que possui acesso direto ao *hardware*. Sem a separação de planos, quando um pacote chega ao Dom0, ele é encaminhado para o DomU, o qual consulta a sua tabela de encaminhamento e retorna o pacote para o Dom0, que o encaminha para fora da máquina física. Com a separação de planos, o pacote não precisa ser encaminhado até o DomU, pois uma cópia atualizada da tabela de encaminhamento do DomU está no Dom0, como mostra a Figura 2(b). É importante observar que apenas os pacotes de dados são encaminhados pela tabela do Dom0. Os pacotes de controle devem ser sempre encaminhados para o DomU, para que o plano de controle possa ser atualizado. O Xen, nativamente, não disponibiliza a separação de planos. Uma das funções do XNetMon é prover essa separação com confiabilidade.



(a) Sem a separação de planos: todos os pacotes passam pela máquina virtual e pelo Dom0.

(b) Com separação de planos: os pacotes passam apenas pelo Dom0, reduzindo o processamento e o atraso por pacote.

Figura 2. Modos de encaminhamento de um pacote no Xen assumindo que o fluxo de dados pertence à rede do roteador virtual em $DomU_1$.

3. Modelo do Atacante

Uma vez que os DomUs podem ter diferentes administradores, não é possível assumir que todos os domínios são confiáveis e que não prejudicariam intencional ou não intencionalmente outros domínios. Assim, assume-se um modelo no qual os DomUs podem ser maliciosos ou podem ter um comportamento não desejado. O comportamento malicioso é caracterizado quando um DomU intencionalmente executa atividades para quebrar o isolamento e, consequentemente, prejudicar o desempenho dos demais domínios. Por comportamento não desejado, define-se toda tentativa de um DomU de exceder a sua reserva de recursos que ocupe todos os recursos da máquina física, o que pode impedir a operação dos demais domínios. Ambos os comportamentos são prejudiciais e, por simplicidade, os domínios que os executam são definidos como domínios adversários. Os demais domínios são denominados domínios comuns.

4. Descrição da Proposta

O principal objetivo do XNetMon é prover o isolamento das redes virtuais, impedindo que domínios adversários possam prejudicar o desempenho de rede de qualquer

outro domínio. O XNetMon alcança este objetivo controlando o uso dos recursos do Dom0. Outra função do XNetMon é alocar os recursos físicos entre os DomUs de acordo com os parâmetros configurados pelo administrador da máquina física. Assim, é possível especificar o quanto cada roteador virtual pode utilizar de cada recurso do Dom0, bem como atribuir aos roteadores virtuais prioridades no uso dos recursos ociosos. O XNetMon também possui funções auxiliares que incluem: a definição das redes virtuais, a separação de planos e a comunicação segura entre máquinas virtuais e o Dom0. A definição das redes virtuais descreve as características do tráfego de cada rede, garantindo que cada rede virtual não se sobrepõe a outra. A separação de planos mantém as tabelas de encaminhamento e os filtros de pacote atualizados entre máquinas virtuais e Dom0. Por fim, a comunicação segura garante que a troca de mensagens para a sincronização dos planos de dados entre máquinas virtuais e o Dom0 é feita de forma confiável.

A arquitetura do XNetMon, ilustrada na Figura 3, é do tipo cliente-servidor, com o servidor no Dom0 e um cliente em cada roteador virtual (DomU). O servidor XNetMon, que roda no Dom0, possui um módulo 'controlador' responsável por monitorar o uso dos recursos do Dom0 pelos roteadores virtuais (DomUs) e punir os DomUs adversários, que possuem comportamentos maliciosos ou não desejados. Os recursos controlados do Dom0 são a banda utilizada, a CPU e a memória. O servidor é composto, também, pelos módulos auxiliares 'separação de planos', que recebe os pedidos de atualização das tabelas de encaminhamento no Dom0, 'comunicação segura', que permite a comunicação entre o Dom0 e os roteadores virtuais (DomUs) de forma segura, e 'definição de redes virtuais'. O cliente XNetMon, executado em cada DomU, é composto por dois módulos: 'separação de planos' e 'comunicação segura'.

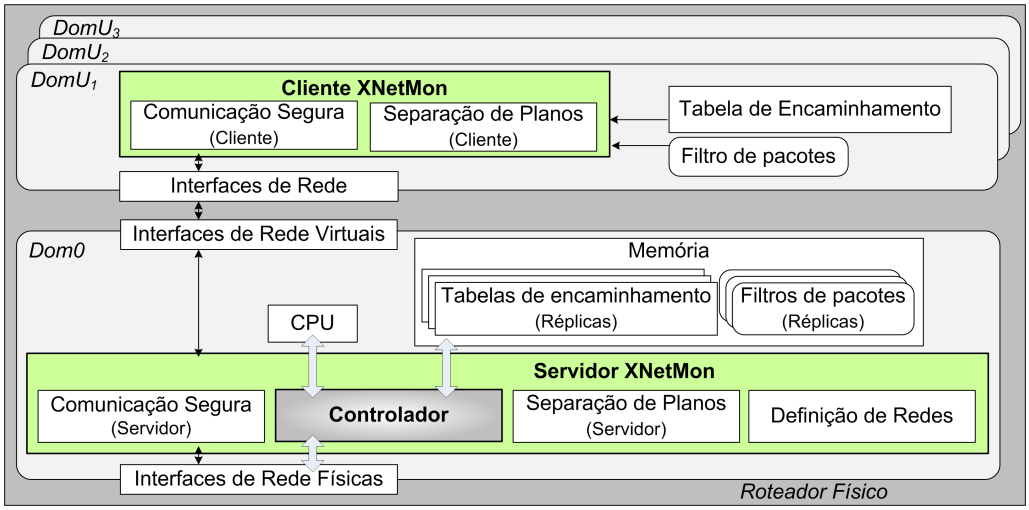


Figura 3. Arquitetura do XNetMon, na qual são representados os fluxos de dados e os recursos do Dom0 que são monitorados, assumindo três roteadores virtuais.

4.1. Módulo do Controlador

O controlador é responsável por monitorar o uso e alocar os recursos do Dom0 para os DomUs a cada período de tempo T. Portanto, o controlador monitora o uso total dos recursos do Dom0, $U(t)$, assim como o uso de recursos por cada roteador virtual i, dado por $U_i(t)$, e pune os DomUs adversários. A alocação de recursos do Dom0 às redes virtuais se processa de duas formas: por reserva fixa e por demanda. A alocação por reserva fixa é feita de acordo com os parâmetros configurados pelo administrador da máquina física. Assim, já na configuração, especifica-se e reserva-se uma quantidade fixa de cada recurso do Dom0 para cada DomU e, por consequência, garante-se uma

"qualidade mínima" para cada rede virtual. A alocação por demanda do XNetMon garante a alta eficiência, porque disponibiliza os recursos não utilizados evitando a ociosidade dos recursos do Dom0. Portanto, são disponibilizados para alocação por demanda os recursos que não foram reservados de forma fixa e também os recursos reservados que não foram usados. Desta forma, uma premissa do controlador é disponibilizar os recursos fixos para uma rede i, representados como um percentual, $Perc_i$, dos recursos totais do Dom0, $R(t)$, sempre que houver demanda. Outra premissa é que todos os recursos que não são usados são alocados por demanda aos roteadores virtuais segundo uma prioridade pré-estabelecida na configuração do controlador. A prioridade é dada por um parâmetro, chamado de peso, dado por $W_i \in$ Z, onde $1 \leq W_i \leq 1000$. Quanto maior o peso de uma rede virtual, maior o acesso que ela tem aos recursos ociosos. Assim, a alocação por demanda dos recursos com base nos pesos provê uma "qualidade adicional diferenciada" para cada rede virtual.

Para cada rede virtual, o XNetMon monitora o uso da banda dos enlaces físicos e também o uso de CPU e de memória no Dom0. A monitoração do uso da banda é feita pela observação do volume de bits usado em cada enlace de saída físico em cada período de tempo T. Se uma rede exceder o uso de um enlace, apenas os pacotes destinados àquele enlace são punidos através de descartes com base em uma probabilidade proporcional ao gasto de recursos "não-utilizados" (ociosos) e do peso da rede.

O uso de CPU no Dom0 é monitorado com base no volume de pacotes que transitam pelo Dom0. O custo de processamento atribuído a cada pacote é diferenciado pela origem e pelo destino do pacote, pois, como indicado na Tabela 1, o processamento de cada pacote tem um impacto diferente sobre a CPU do Dom0 dependendo de se o pacote vai para ou vem de um DomU, ou se vai para ou vem de uma máquina externa. Assim, a estimativa do uso de CPU é feita ponderando-se o número de pacotes pelo custo de processamento de cada pacote. Além disso, em transferências entre DomUs, o DomU de origem se responsabilizará por todos os custos de uso de CPU, para impedir que um domínio adversário consiga exaurir os recursos de CPU de um domínio comum através de uma transmissão de dados com alta taxa. Em transferências entre DomU e Dom0, os custos de uso de CPU são sempre contabilizados para o DomU. Se uma rede exceder o uso de CPU no Dom0 pelo excesso de pacotes processados ou por excesso de atualizações do plano de dados, então todos os pacotes provenientes ou com destino para aquele roteador virtual são descartados com uma probabilidade proporcional ao uso de recursos não-utilizados (ociosos) de CPU e ao peso da rede.

Com isso, o uso de cada recurso do Dom0 por uma rede i além da sua reserva fixa, seja ele CPU ou a banda de algum enlace físico, pode gerar uma probabilidade de descarte de pacotes. Os descartes devido a CPU são feitos na chegada do pacote ao Dom0, para evitar o processamento de pacote, enquanto que os descartes devido ao enlace são feitos apenas após o processamento do pacote, para que se descubra para qual interface de saída o pacote está sendo direcionado.

O controle de memória é feito pela observação do tamanho da tabela de encaminhamento. Caso a memória do Dom0 atinja limites críticos, as redes virtuais cujas tabelas/filtros ocupem mais memória do que é permitido são punidas, através do descarte de um percentual de rotas. Para evitar a perda de pacotes, uma rota *default* para o roteador virtual é adicionada, forçando os pacotes correspondentes as rotas descartadas a serem encaminhados pelo roteador virtual ao invés de serem descartados pelo Dom0. Portanto, a redução do tamanho da tabela de encaminhamento nunca implicará em descarte de pacotes, mas apenas em uma redução do desempenho no encaminhamento da rede virtual, pois o encaminhamento passa a ser efetuado no DomU correspondente ao invés de ser feito pelo Dom0.

4.1.1. Cálculo da Punição

A punição dos domínios adversários (maliciosos ou de comportamento não-desejável) é feita pelo descarte de pacotes. Busca-se um valor de probabilidade de descarte de pacotes que equilibre o uso dos recursos do Dom0 pelos roteadores virtuais. Os descartes são feitos apenas se o roteador virtual usou mais do que os recursos reservados para ele e se o uso dos recursos atingir um nível crítico, dado por um percentual N dos recursos totais $R(t)$, que possa impedir outros roteadores de usarem os seus recursos reservados. A taxa de descarte é atualizada a cada intervalo T, para que a punição possa acompanhar a taxa de chegada de pacotes de cada rede virtual. Assumindo que o total de recursos não reservados seja $D = R(t) - \sum_{\forall i} Perc_i \cdot R(t)$ e C seja uma constante de ajuste, então a atualização da probabilidade de descarte da rede virtual i no intervalo $t + T$, dada por $P_{ui}(t + T)$, é feita segundo o Algoritmo 1.

Algoritmo 1: Cálculo da punição para cada rede virtual.

Entrada: $P_{ui}(t)$, W_i, $Perc_i$, $R(t)$, $U(t)$, $U_i(t)$, $D(t)$, N, C
Saída: $P_{ui}(t + T)$
se *(*$Perc_i \cdot R(t) < U_i(t)$*) ou (*$P_{ui}(t) > 0$*)* **então**
    % Calcula uso de recursos ociosos
    $U_{oi} = (U_i(t) - Perc_i \cdot R(t))/D(t)$
    se *(*$Perc_i \cdot R(t) < U_i(t)$*)* **então**
        se *(*$N \cdot R(t) \leq U(t)$*)* **então**
            se *(*$P_{ui}(t) > 0$*)* **então**
                % Como não existem recursos ociosos, alguma rede pode estar sendo prejudicada e a punição é aumentada
                $P_{ui}(t + T) = min(P_{ui}(t) + (1 + U_{oi}) \cdot (1 + \frac{1}{W_i}) \cdot \frac{P_{ui}(t)}{(3 - \frac{1}{W_i})}, 1)$
            senão
                $P_{ui}(t + T) = P_{uinicial}$ % Seta um valor inicial de punição
            fim
        senão
            % Como existem recursos ociosos que podem ser distribuídos, a punição é reduzida
            $P_{ui}(t + T) = max(P_{ui}(t) - (1 + (1 - U_{oi})) \cdot (1 - \frac{1}{W_i}) \cdot \frac{P_{ui}(t)}{(3 + \frac{1}{W_i})}, 0)$
        fim
    senão
        % Como utilizou apenas os seus recursos fixos, a punição é diminuida rapidamente
        $P_{ui}(t + T) = max(P_{ui}(t) - C \cdot (1 - \frac{1}{W_i}) \cdot \frac{P_{ui}(t)}{3}, 0)$
    fim
senão
    $P_{ui}(t + T) = 0$
fim

É importante observar que mesmo que o DomU consuma menos do que sua reserva fixa, a punição não é imediatamente zerada para evitar instabilidades. O algoritmo busca a punição ideal para o DomU e essa busca pode ter instantes nos quais a punição vai estar acima ou abaixo do valor correto até a estabilização do valor da probabilidade de punição. Outro ponto importante é que, para impedir que as operações de E/S (comunicação de rede) geradas pelos roteadores virtuais interrompam os demais serviços do Dom0 por sobrecarga de CPU, uma punição residual pode ser aplicada constantemente nas interfaces de saída das máquinas virtuais. Tal punição deve ser pequena o suficiente para não impactar transmissões de baixo volume entre roteadores virtuais, mas deve impedir que o DomU consuma todos os recursos do Dom0, aumentando o seu tempo de resposta.

4.2. Módulo de Separação de Planos

O módulo da separação de planos tem como objetivo separar as funções de controle da rede da função de encaminhamento de dados. A principal função de controle da rede é criar a tabela de encaminhamento a partir das mensagens do protocolo de rotea-

mento que cada roteador virtual recebe. Como mencionado anteriormente, para um maior desempenho no encaminhamento dos dados, é fundamental que esta função de encaminhamento seja executada no Dom0. No entanto, ao separar a função de encaminhamento da função de controle, os roteadores virtuais ficam impedidos de atualizarem suas tabelas de encaminhamento e seus filtros de pacote, porque eles não têm permissão de acesso à memória do Dom0. A solução no XNetMon foi manter tabelas e filtros nos DomUs e fazer uma réplica no Dom0. Portanto, os XNetMon clientes (nos DomUs) possuem um módulo que faz o monitoramento das modificações na tabela de encaminhamento e nos filtros de pacote, e o XNetMon servidor (no Dom0) faz o mapeamento tanto da tabela de encaminhamento quanto do filtro de pacotes construídos em cada DomU em uma tabela de encaminhamento e um filtro de pacotes no Dom0.

Toda mudança na tabela de encaminhamento ou filtro de pacotes deve ser atualizada imediatamente na sua réplica no Dom0. Por essa razão, o módulo de monitoração de modificações na tabela de encaminhamento e nos filtros observa a interface de entrada do DomU e verifica as mudanças quando uma nova mensagem de controle chega ao domínio. Em seguida, sempre que for encontrada uma diferença na tabela ou no filtro, apenas a diferença é transmitida para o Dom0, através do módulo de comunicação segura. Do lado do servidor do XNetMon, o módulo de separação de planos, ao receber uma mensagem notificando uma modificação na tabela ou filtro, busca pelas definições da rede para descobrir em qual tabela do Dom0 aquelas modificações devem ser inseridas.

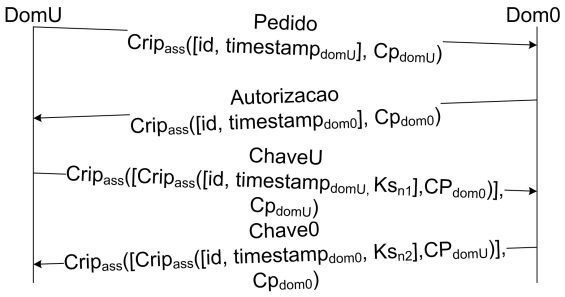
4.3. Módulo de Comunicação Segura

O objetivo do módulo de comunicação segura é permitir a comunicação entre o Dom0 e os DomUs utilizando autenticação mútua e privacidade na transferência dos dados. Esse deve ser um módulo leve, pois é usado com frequência na atualização do plano de dados no Dom0. A autenticação mútua é necessária para garantir que nenhuma máquina adversária tentará se passar por uma máquina comum ou pelo Dom0 para gerar dados falsos no plano de dados do domínio comum atacado no Dom0.

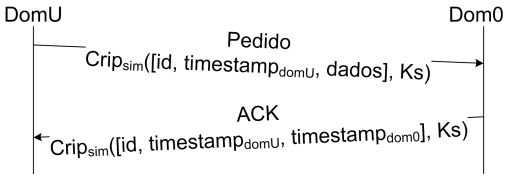
A comunicação segura é composta por dois protocolos: um para troca de chaves de sessão, descrito na Figura 4(a) e um para troca de dados entre as máquinas, descrito na Figura 4(b), nos quais se assume que Cp é a chave privada, CP é a chave pública, $crip([M], K)$ é a cifragem da mensagem M com a chave K e id é a identificação do nó fonte da mensagem. Uma vez que o custo computacional para o estabelecimento do canal seguro deve ser baixo, optou-se pela criação de um protocolo para autenticar e estabelecer uma chave de sessão e um protocolo para transmitir dados com segurança. Com isso, a autenticação pode ser feita utilizando criptografia assimétrica, enquanto que a comunicação é feita utilizando-se criptografia simétrica. Além disso, sempre que existe troca de mensagens de controle, é necessário impedir os ataques de *replay*, no qual o domínio adversário repete transmissões antigas de mensagens de controle de um domínio comum ou do Dom0. Um domínio adversário poderia, por exemplo, repetir uma mensagem de atualização antiga para adicionar rotas que acabaram de ser excluídas no plano de dados do domínio comum. Com isso, seria possível gerar inconsistências na tabela de encaminhamento no Dom0 com mensagens autenticadas. Para evitar ataques de *replay*, o Dom0 e as demais máquinas virtuais executam uma troca de *timestamps* durante o estabelecimento da chave de sessão. Com a troca dos *timestamps*, tanto o Dom0 quanto o domínio comum sabem a diferença entre os seus relógios, T_{d0} e T_{du}, respectivamente, de acordo com a equação $\delta_{est} = |T_{d0} - T_{du}|$. Assim, sempre que uma mensagem de atualização for enviada, ela deve conter o *timestamp* da fonte, o qual é utilizado para verificar se a mensagem não é uma repetição. Essa verificação é feita se o tempo atual no Dom0, T_{d0_n}, e o tempo atual no DomU, T_{du_n}, obedecerem à seguinte inequação, assumindo que e_{max}

é o atraso máximo de transmissão,

$$|T_{d0_n} - T_{du_n}| - e_{max} < \delta_{est} < |T_{d0_n} - T_{du_n}| + e_{max}. \quad (1)$$



(a) Sequência de mensagens para estabelecimento de chaves de sessão $K_s = f(K_{s1}, K_{s2})$.



(b) Sequência de mensagens para sincronização de dados entre o Dom0 e o DomU.

Figura 4. Diagramas de tempo das mensagens para comunicação segura.

Dessa forma, ao se garantir a autenticidade, a privacidade e não-reprodutibilidade dos dados, é possível afirmar que a comunicação entre o Dom0 e os DomUs ocorre de forma segura.

5. Descrição do Protótipo e Resultados Obtidos

Foi desenvolvido um protótipo para realizar uma prova de conceito sobre como fazer a comunicação segura entre o Dom0 e as máquinas virtuais, medir a efetividade do controlador na presença de domínios adversários e verificar a eficiência do controlador na divisão dos recursos. O protótipo foi implementado em C++ e Python e disponibiliza a separação de planos, a comunicação segura entre domínios e o controle de rede e de CPU no Xen modo *router*. Os planos de dados foram criados utilizando-se as diversas tabelas de encaminhamento do *kernel* do Linux. A monitoração dos pacotes e a punição com descarte probabilístico foram implementadas com o Iptables[2]. Foram escolhidos para a implementação da comunicação segura os protocolos Blowfish[3], por ser considerado um protocolo de criptografia simétrica leve e seguro, e RSA[4], por ser um dos protocolos de criptografia assimétrica mais utilizados. Para computar o uso de CPU no XNetMon, estimou-se o custo de cada operação de rede no Dom0, incluindo a comunicação entre DomUs, de DomU para Dom0, de Dom0 para DomU, de DomU para máquina externa, de máquina externa para DomU e de máquina externa para máquina externa. Para tanto, mediu-se o consumo de CPU no Dom0 ao se realizar transmissões com diferentes taxas de pacote. Em seguida, ajustes foram feitos com base nos testes do protótipo desenvolvido. O descarte residual nas interfaces virtuais, descrito na Seção 4.1, foi estimado com base na taxa de pacotes que causava danos ao funcionamento do Dom0, chegando-se a uma probabilidade de descarte de $0,0009$.

Os testes foram realizados em uma máquina, doravante denominada roteador, com processador Intel core2 quad com 4GB de memória RAM, utilizando o Xen 3.4-amd64 no modo *router*. Esse roteador possui cinco interfaces físicas de rede Ethernet com banda de 1Gb/s cada. São instanciadas quatro máquinas virtuais, cada uma delas com uma CPU virtual e 128 MB de memória e executando um sistema operacional Debian com *kernel* Linux 2.6-26-2. As máquinas virtuais são configuradas com cinco interfaces de rede, para um mapeamento simples das interfaces virtuais nas interfaces reais. O número de

[2]http://www.netfilter.org/

[3]http://www.schneier.com/blowfish.html

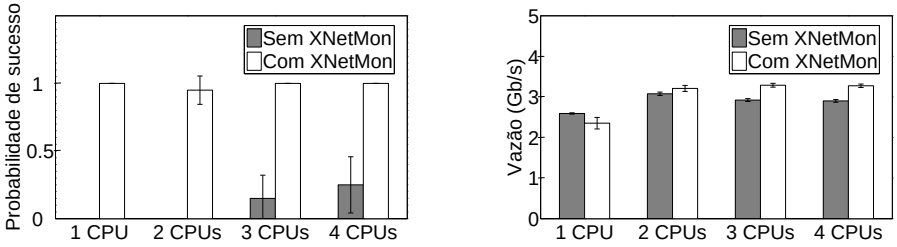
[4]http://www.rsa.com/rsalabs/node.asp?id=2146

CPUs virtuais no Dom0 varia de acordo com o teste e não existem restrições de memória para esse domínio. As CPUs físicas estão compartilhadas por todas as CPUs virtuais, cabendo ao hipervisor o escalonamento das CPUs virtuais nas CPUs reais. Os testes usam duas máquinas externas, que geram ou recebem pacotes, cada uma com uma interface de rede de 1Gb/s. Essas máquinas estão ligadas ao roteador e se comunicam apenas por intermédio do roteador.

O primeiro teste realizado objetiva medir a disponibilidade do procedimento de atualização do plano de dados realizado com segurança. A segurança é importante para impedir que DomUs maliciosos se passem por DomUs comuns para inserir informações falsas nos planos de dados dos domínios atacados. Para tornar a separação de planos convencional segura, adicionou-se a ela o protocolo de comunicação segura do XNetMon. O teste de disponibilidade do procedimento de atualização é considerado bem sucedido se o DomU consegue atualizar o seu plano de dados usando o protocolo de comunicação segura descrito na Seção 4.3 independente das demais operações que o Dom0 esteja executando. Isto significa que nenhuma operação consegue impedir a atualização do plano de dados e, portanto, é inviável qualquer possibilidade de ataque deste tipo. Assim, comparou-se o mecanismo de separação de planos convencional com a separação de planos no XNetMon que se serve do protocolo de comunicação segura e tem o controlador de rede. O teste consiste de um máximo de três tentativas do $DomU_1$ para atualizar o plano de dados, enquanto o $DomU_2$ envia dados em alta taxa para o $DomU_1$, usando a ferramenta Iperf com comunicação via TCP. O cenário procura simular a ação de um roteador virtual, o $DomU_2$, como um domínio adversário tentando impedir que um roteador comum, o $DomU_1$, opere normalmente. No XNetMon, o $DomU_1$, que tenta fazer a atualização, tem $Perc_1 = 0,5$ de reserva de recursos, enquanto que o domínio adversário, chamado de $DomU_2$, possui $Perc_2 = 0,3$ de reserva de recursos. As demais máquinas virtuais não possuem recursos reservados. Todos os DomUs possuem peso $W = 500$.

A Figura 5(a) mostra a probabilidade de sucesso de atualização do plano de dados, enquanto que a Figura 5(b) mostra o volume de dados transmitido entre as máquinas virtuais, ambas assumindo um intervalo de confiança de 95%. Observa-se que o ataque para impedir a atualização do plano de dados é efetivo quando não se usa o XNetMon. Sem o uso do XNetMon, o ataque continua efetivo mesmo quando se disponibiliza um maior número de CPUs, procurando oferecer ao Dom0 maior capacidade de processamento para tratar os pacotes trocados entre as máquinas virtuais e os cálculos criptográficos. Os resultados com o XNetMon mostram um aumento de até 100% na probabilidade de sucesso de atualização do plano de dados, mostrando sua eficácia em prover disponibilidade para atualizações do plano de dados. O XNetMon é eficaz porque o controlador limita o tráfego de ataque proveniente do $DomU_2$ quando este usa excessivamente a CPU para enviar dados em alta taxa e, ao mesmo tempo, o controlador do XNetMon reserva a CPU necessária para o envio das mensagens de atualização e os cálculos criptográficos no $DomU_1$. A Figura 5(b) mostra que o XNetMon gera uma punição no tráfego do $DomU_2$ para o $DomU_1$, para garantir que os recursos de CPU não sejam exauridos, e por isto a vazão com XNetMon com uma CPU é menor que a vazão da separação de planos convencional. Ao se aumentar o número de CPUs, essa punição é naturalmente relaxada. Com isso, o volume de tráfego com o XNetMon passa a ser maior do que o obtido com a separação de planos convencional, sem o controlador do XNetMon. Nesta condição, devido à CPU do $DomU_1$ ser reservada, garantindo que tanto as atualizações de plano de dados quanto os ACKs da conexão TCP iniciada pelo $DomU_2$ são entregues, a perda de ACKs quando não se utiliza o controlador XNetMon causa um impacto maior no tráfego do que as limitações de envio impostas pelo controle proposto ao $DomU_2$ devido ao consumo de CPU. Assim, o XNetMon garante uma separação de planos segura e sempre disponível, ao mesmo tempo em que assegura uma conexão com alto desempenho entre as máquinas

virtuais, devido à arquitetura proposta com o módulo controlador. De fato, o teste mostra que a separação de planos com segurança só pode ser considerada disponível quando existe algum tipo de limitador do uso de CPU do Dom0 pelos roteadores virtuais.

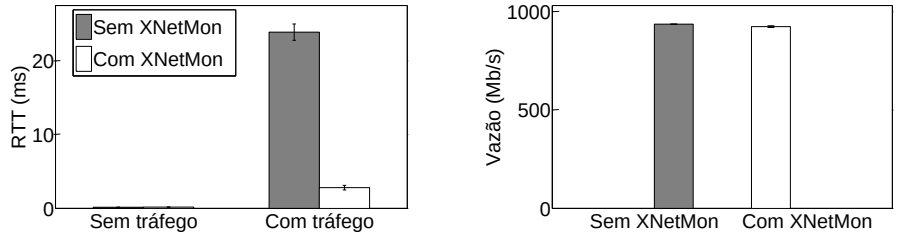


(a) Probabilidade de sucesso na atualização do plano de dados.

(b) Volume de dados transmitidos na conexão entre DomUs.

Figura 5. Disponibilidade do procedimento de atualização do plano de dados com segurança pelo $DomU_1$ quando existe um tráfego entre o $DomU_2$ e o $DomU_1$.

O segundo teste realizado avalia o atraso inserido na transmissão de dados pelo XNetMon, quando comparado a separação de planos convencional, assumindo um intervalo de confiança de 95% nos resultados. O objetivo é verificar o quanto a sobrecarga do XNetMon afeta o atraso de acordo com a carga de trabalho do Dom0. Uma vez que não está se medindo a justiça na divisão dos recursos, foi criada apenas uma rede virtual com 100% dos recursos do Dom0 no teste do XNetMon. O teste consiste de dois experimentos que medem o *Round Trip Time* (RTT) na comunicação entre duas máquinas externas. No primeiro experimento, não existe tráfego de fundo, enquanto que, no segundo experimento, um tráfego de fundo TCP foi gerado com a ferramenta Iperf entre as duas máquinas externas. Os resultados de ambos os experimentos são mostrados na Figura 6(a). Observa-se que, sem tráfego de fundo, a transmissão de dados apresenta um valor de RTT muito baixo para as duas configurações. No entanto, quando existe um tráfego de fundo, a CPU do Dom0 é sobrecarregada, aumentando o tempo de resposta do sistema, o que implica em criação de filas no Dom0 e, consequentemente, um aumento do RTT. Os resultados mostram que o controle de CPU e banda provido pelo XNetMon impede que a CPU seja sobrecarregada e, com isso, o desempenho com XNetMon é bem superior, resultando em uma redução do RTT, em mais de oito vezes. É importante observar que, embora o controle do XNetMon implique em descarte de pacotes, esses descartes não causam um grande impacto sobre o tráfego, como mostra a Figura 6(b).



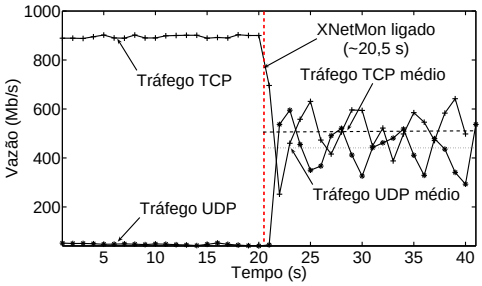
(a) Impacto do XNetMon sobre o RTT.

(b) Vazão do tráfego de fundo.

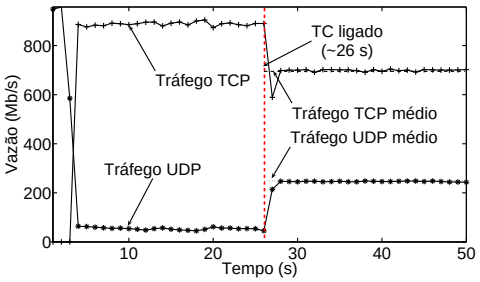
Figura 6. Atraso na comunicação entre máquinas externas em função do tráfego encaminhado pelo Dom0, assumindo o uso de uma CPU virtual no Dom0.

O terceiro experimento diz respeito ao compartilhamento dos enlaces de saída. As ações tomadas em uma rede virtual não podem interferir nos recursos reservados para outra rede virtual para que isto não se torne uma vulnerabilidade. Assim, é essencial que uma rede possa contar com os recursos que lhe foram reservados, independente das ações nas outras redes. Nesse experimento, um DomU e uma máquina externa em redes virtuais diferentes iniciam uma comunicação com a outra máquina externa. Dessa forma,

existe uma disputa pela banda do enlace para a máquina de destino. Ambas as redes virtuais foram configuradas com uma reserva $Perc = 0,5$ dos recursos do Dom0 e peso $W = 500$. Para avaliar o compartilhamento provido pelo XNetMon, realizou-se também um teste utilizando a ferramenta *Traffic Control* (TC), muito utilizada para controle de tráfego em máquinas Linux, para fazer o controle da interface de saída para a máquina externa de destino, assumindo-se o uso de *Hierarchy Token Bucket* (HTB). Assim, no teste com o TC, foram criadas duas filas de saída, cada uma com reserva mínima de 512Mb/s, podendo emprestar até 1Gb/s do enlace para simular a mesma política de uso de recursos do XNetMon. As Figuras 7(a) e 7(b) apresentam o resultado quando o DomU inicia uma comunicação com UDP com taxa máxima de 1,5 Gb/s e a máquina externa, uma comunicação com TCP, ambas utilizando o Iperf com pacotes de 1500 B. O teste foi realizado sem nenhum controle das comunicações no início e, a partir do instante marcado, as comunicações passam a ser controladas pelo XNetMon ou pelo TC.



(a) XNetMon: tráfego UDP no DomU e TCP na máquina externa.



(b) TC: tráfego UDP no DomU e TCP na máquina externa.

Figura 7. Justiça na divisão de recursos assumindo que as duas redes possuem recursos iguais no Dom0.

Os resultados deste experimento mostram que o tráfego da máquina externa tem prioridade sobre o tráfego do DomU quando não se utiliza nenhum tipo de controle[5]. Isso representa uma falha grave de isolamento ao se utilizar a separação de planos, pois o tráfego externo influencia no volume máximo do tráfego gerado por um DomU, gerando uma divisão de recursos sem justiça. Assim, uma máquina externa de uma rede adversária poderia gerar um tráfego de ataque contra a rede de um domínio comum dentro da máquina física. Uma vez que os recursos estão sendo divididos igualmente entre as duas redes, a situação que configura justiça levaria a uma divisão igualitária do enlace, ou seja, 512Mb/s para cada rede virtual. Embora a variação do tráfego com o XNetMon seja maior do que a do TC, a vazão média com o XNetMon possui um erro menor com relação à taxa ideal de 512Mb/s para cada rede virtual. De fato, se não houver um controle da vazão de entrada da máquina externa, o tráfego UDP da máquina virtual é desprivilegiado e não consegue atingir altas taxas. Com isso, o controle com o XNetMon apresentou um erro da vazão média com relação à vazão ideal por rede de $-14,2\%$ para a comunicação UDP e de $-0,62\%$ para a comunicação TCP, enquanto que o TC apresentou um erro de $-52,18\%$ para a comunicação UDP e $+35,68\%$ para a comunicação TCP. Dessa forma, o XNetMon apresentou uma justiça superior na divisão dos recursos do enlace por ser adaptado à arquitetura do Xen.

6. Trabalhos Relacionados

Os problemas de isolamento do Xen vêm sendo estudados sobre diversos aspectos [Han et al. 2009, Egi et al. 2008]. Jin *et al.* propuseram um mecanismo para garantir

[5]Foram realizados testes, cujos resultados não estão apresentados por razões de concisão, nos quais a máquina virtual iniciava uma comunicação com TCP e o mesmo comportamento foi observado.

justiça no uso das memórias caches L2 e L3 no Xen, cujo uso também não é contemplado pelos mecanismos de isolamento do hipervisor do Xen [Jin et al. 2009]. A proposta modifica o algoritmo de alocação de páginas de memória do hipervisor utilizando a técnica de coloração de páginas. Outra proposta faz uma abordagem baseada no estudo do desempenho de roteadores virtuais criados com Xen [Egi et al. 2008], observando os gargalos no encaminhamento causados por uso de CPU e de memória ao se encaminhar pacotes pelo Dom0. A proposta propõe a criação de diferentes planos de dados com eficiência e flexibilidade usando o Click[6], que é uma plataforma para criação de roteadores modulares. No entanto, não são apresentados mecanismos para diferenciar o uso dos recursos entre as máquinas, assim como também não é apresentada uma forma de fazer a separação de planos de forma segura. Dessa forma, ambas as propostas poderiam ser somadas ao XNetMon para a obtenção de maior desempenho, flexibilidade e segurança.

Mecanismos para o isolamento de ambientes virtuais também foram propostos para outras plataformas de virtualização. O Trellis [Bhatia et al. 2008] é um sistema para prover isolamento entre as redes virtuais na plataforma VINI[7]. Por ser baseado em virtualização em nível de sistema operacional, ou seja, todos os ambientes virtuais compartilham o mesmo *kernel*, o desempenho do encaminhamento de pacotes utilizando-se o Trellis é inferior ao do Xen com separação de planos [Egi et al. 2008]. O maior problema das abordagens baseadas em virtualização de sistema operacional [Bhatia et al. 2008, Wang et al. 2008, Zec 2003] é que todos os planos de controle são executados sobre o mesmo sistema operacional e, em geral, essas abordagens também não permitem a criação de planos de dados diferenciados por rede virtual, o que são características restritivas. Outra plataforma de virtualização é o OpenFlow [McKeown et al. 2008], o qual se baseia na premissa de uma rede com elementos encaminhadores simples e plano de controle centralizado. Para dividir os recursos dos elementos encaminhadores pelas redes virtuais, o OpenFlow utiliza a ferramenta FlowVisor [Sherwood et al. 2010], a qual funciona como um *proxy* entre os elementos encaminhadores e os planos de controle. O FlowVisor controla o uso de memória e CPU dos elementos encaminhadores, assim como faz o controle da divisão dos espaços de rede, ou seja, sobre quais características definem cada rede virtual.

Uma vez que a plataforma de virtualização do XNetMon é o Xen, é possível ter alta flexibilidade para o desenvolvimento dos planos de controle, já que cada ambiente virtual possui o seu próprio sistema operacional e conjunto de aplicações. O desempenho no encaminhamento é obtido com a separação de planos. O XNetMon diferencia-se por tratar de como criar uma separação de planos segura e como impedir que DomUs ou máquinas externas atrapalhem o funcionamento das demais redes virtuais.

7. Conclusões

O XNetMon é uma arquitetura para garantir, com segurança e eficácia, o isolamento e a divisão dos recursos compartilhados no Xen. A proposta permite uma separação de planos com segurança, devido ao uso de protocolos para comunicação segura. Os resultados mostram que a proposta faz um compartilhamento de recursos adequado, reduzindo o erro na divisão dos recursos entre os domínios em mais de três vezes, quando comparado a outro mecanismo. Além disso, o XNetMon reduz atrasos no encaminhamento, por impedir que o Dom0 seja sobrecarregado e aumente o seu tempo de resposta. De fato, os resultados mostram que a sobrecarga do Dom0 prejudica as operações de encaminhamento, de gerência e de controle. O XNetMon também apresenta alta disponibilidade para atualização de plano de dados, o que é essencial para uma separação de planos

[6]http://read.cs.ucla.edu/click/click
[7]http://www.vini-veritas.net/

segura. Tais resultados foram obtidos devido à eficiência do controlador do XNetMon, o qual é adaptado à arquitetura do Xen, impedindo a sobrecarga do Dom0 e provendo uma divisão adequada dos recursos. Portanto, o XNetMon provê um isolamento entre as redes virtuais, impedindo a ação de máquinas adversárias sob diferentes tipos de cenários.

Como trabalhos futuros, pretende-se implementar o controle de memória e estudar o ajuste de parâmetros no controlador para que a estabilização da punição seja mais rápida. Além disso, pretende-se integrar o XNetMon com outras propostas com um plano de dados mais flexível no Dom0.

Referências

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 GT-CS-07-10, Princeton University, Georgia Tech, and T-Labs/TU Berlim.

Egi, N., Greenhalgh, A., Handley, M., Hoerdt, M., Huici, F., and Mathy, L. (2008). Fairness issues in software virtual routers. In *PRESTO '08: Proceedings of the ACM workshop on Programmable routers for extensible services of tomorrow*, pages 33–38.

Egi, N., Greenhalgh, A., Handley, M., Hoerdt, M., Mathy, L., and Schooley, T. (2007). Evaluating Xen for router virtualization. In *ICCCN'07: International Conference on Computer Communications and Networks*, pages 1256–1261.

Fernandes, N. C., Moreira, M. D. D., Moraes, I. M., Ferraz, L. H. G., Couto, R. S., Carvalho, H. E. T., Campista, M. E. M., Costa, L. H. M. K., and Duarte, O. C. M. B. (2010). Virtual networks: Isolation, performance, and trends. *To be published in the Annals of Telecommunications*.

Han, S.-M., Hassan, M. M., Yoon, C.-W., and Huh, E.-N. (2009). Efficient service recommendation system for cloud computing market. In *ICIS'09: Proceedings of the 2nd International Conference on Interaction Sciences*, pages 839–845.

Jin, X., Chen, H., Wang, X., Wang, Z., Wen, X., Luo, Y., and Li, X. (2009). A simple cache partitioning approach in a virtualized environment. In *2009 IEEE International Symposium on Parallel and Distributed Processing with Applications*, pages 519–524.

Laureano, M. A. P. and Maziero, C. A. (2008). Virtualização: Conceitos e aplicações em segurança. In *Minicursos do VIII Simpósio Brasileiro em Segurança da Informação e de Sistemas Computacionais*, pages 1–49.

McKeown, N., Anderson, T., Balakrishnan, H., Parulkar, G., Peterson, L., Rexford, J., Shenker, S., and Turner, J. (2008). OpenFlow: Enabling innovation in campus networks. *ACM SIGCOMM Computer Communication Review*, 38(2):69–74.

Pisa, P. S., Fernandes, N. C., Carvalho, H. E. T., Moreira, M. D. D., Campista, M. E. M., Costa, L. H. M. K., and Duarte, O. C. M. B. (2010). Openflow and Xen-based virtual network migration. In *The World Computer Congress 2010 - Network of the Future Conference (a ser publicado)*.

Sherwood, R. et al. (2010). Carving research slices out of your production networks with OpenFlow. *ACM SIGCOMM Computer Communication Review*, 40(1):129–130.

Wang, Y., Keller, E., Biskeborn, B., der Merwe, J. V., and Rexford, J. (2008). Virtual routers on the move: Live router migration as a network-management primitive. In *ACM SIGCOMM*, pages 231–242.

Zec, M. (2003). Implementing a clonable network stack in the FreeBSD kernel. In *Proceedings of the 2003 USENIX Annual Technical Conference*, pages 137–150.