

Felipe Belsholff Pina

Implementação de Funções de Rede Virtuais usando ClickOS

Vitória, Espírito Santo, Brasil

Julho 2018

Felipe Belsholff Pina

Implementação de Funções de Rede Virtuais usando ClickOS

Projeto final de graduação apresentado ao Departamento de Informática do Centro Tecnológico da Universidade Federal do Espírito Santo, como requisito parcial para obter o título de Bacharel em Ciência da Computação

Universidade Federal do Espírito Santo - UFES

Centro Tecnológico - CT

Graduação em Ciência da Computação

Orientador: Prof. Dr. Rodolfo da Silva Villaça

Coorientador: Prof. Dr. Magnos Martinello

Vitória, Espírito Santo, Brasil

Julho 2018

Felipe Belsholff Pina

Implementação de Funções de Rede Virtuais usando ClickOS

Projeto final de graduação apresentado ao Departamento de Informática do Centro Tecnológico da Universidade Federal do Espírito Santo, como requisito parcial para obter o título de Bacharel em Ciência da Computação

Trabalho aprovado. Vitória, Espírito Santo, Brasil, 13 de julho de 2018:

Prof. Dr. Rodolfo da Silva Villaça
Orientador

Prof. Dr. Magnos Martinello
Co-orientador

Professor
Convidado Interno

Profa. Dra. Cristina Klippel
Dominicini
Convidado Externo

Vitória, Espírito Santo, Brasil
Julho 2018

*Em memória de Leonina. Dedicado àqueles que diante das
negativas do mundo, contrariaram-no na busca pelo sim.*

Agradecimientos

A

*“Você toma a pílula azul — a história acaba, e
você acorda em sua cama acreditando no que
quiser acreditar. Você toma a pílula vermelha —
fica no País da Maravilhas, e lhe mostro quão
fundo a toca do coelho vai.”*

(Morpheus, The Matrix)

Resumo

Com o advento da virtualização e o surgimento do modelo de computação em nuvem (*Cloud Computing*), uma das consequências imediatas para as empresas e instituições foi a redução da aquisição de novos equipamentos. Além disso, ainda no ponto de vista de infraestrutura, desacoplar funções que estavam embarcadas em hardware específico e fechado, trazendo-as até camadas de abstração superiores, habilita um menor tempo de implantação e de manutenção, acrescido de um grande salto com relação a escalabilidade e gerenciamento das mesmas.

Neste cenário, o objetivo do paradigma NFV é prover funções de rede como: *firewalls*, balanceadores de carga, NATs, e IDSs, etc, através de uma linguagem de marcação, colaborando com a flexibilidade do provisionamento e a programabilidade de tais funções, até então também engessadas pelo uso dos *middleboxes* de prateleira.

Neste quesito, ClickOS propõe uma plataforma de virtualização para criação de funções de rede virtuais de alto desempenho, a serem amplamente utilizados por desenvolvedores de VNFs. Ele provê como ferramenta uma máquina virtual minimalista, escalável e otimizada para executar o *software* Click no *hypervisor* do Projeto Xen, seguido de uma sequência de otimizações focadas na vazão de pacotes.

No entanto, a pouca quantidade de trabalhos ou artigos detalhando o uso de ClickOS diante da atual demanda por NFV, até mesmo em língua inglesa, dificulta sua adoção, difusão, e por fim, o desenvolvimento da plataforma e seus conceitos por pesquisadores, professores, empresas e outros atores interessados, mesmo que este seja em software livre e possua uma documentação online, já que o conhecimento sobre ele está disperso.

Assim sendo, este trabalho tem como objetivo apresentar detalhadamente ClickOS como uma alternativa para a implementação de **funções de rede virtuais** dentro de NFV, usando como cenário o balanceamento de carga em serviços de hospedagem *web*.

Palavras-chave: NFV; virtualização de funções de rede; VNF; função de rede virtual; middleboxes virtuais; Click; ClickOS.

Abstract

With the advent of virtualization and the Cloud Computing model arising, one of the fresh consequences for companies and institutions was new equipment acquisition reduction. In addition, still from an infrastructure viewpoint, decoupling functions that were embedded into specific and closed hardware up to higher abstraction layers enables a shorter deployment and maintenance time, plus a big increase related to scalability and management of them.

In this scenario, the NFV paradigm goal is provide network functions as: firewalls, load balancers, NATs, IDSes, etc, through a markup language, collaborating with the provisioning flexibility and programmability of such functions, until then also plastered by the use of shelf middleboxes.

In this regard, ClickOS proposes a virtualization platform for high performance virtual network functions construction by developers. It provides as a tool a minimalist, scalable, and optimized virtual machine to run Click software in the Xen Project hypervisor, followed by some optimizations focused on packet throughput.

However, a lack of works, articles or papers detailing ClickOS use in face of current NFV demand, even in english language, makes it difficult to adopt, disseminate, and finally, the development of the platform and its concepts by researchers, professors, companies and other stakeholders, even if it is in free software and has an online documentation, since the knowledge about it is dispersed.

Thus, this work aims to present in detail ClickOS as an alternative to the virtual network functions implementation within NFV, using a load balancing scenario in web hosting services.

Keywords: NFV; network functions virtualization; VNF; virtual network function; virtual middleboxes; Click; ClickOS.

Lista de Figuras

| | |
|---|----|
| Figura 1 – Representação do modelo NFV | 24 |
| Figura 2 – Estrutura das interfaces virtuais de rede no Linux Bridge | 26 |
| Figura 3 – Descrição gráfica de um elemento em Click | 30 |
| Figura 4 – Condicionador de tráfego em Click | 32 |
| Figura 5 – Uma aplicação dos elementos mínimos para ações sobre o protocolo IP | 35 |
| Figura 6 – Interface gráfica e grafo padrão da Figura 5 em Clicky | 37 |
| Figura 7 – Arquitetura inicial presente no ClickOS | 39 |
| Figura 8 – Antes e depois das alterações mencionadas | 40 |
| Figura 9 – Exemplo de arquivo de configuração de uma VNF | 41 |
| Figura 10 – Comportamento esperado das VNFs propostas | 46 |
| Figura 11 – Código para o firewall em Click | 50 |
| Figura 12 – Código para o balanceador de carga por endereço IP de origem no Click | 55 |
| Figura 13 – Protótipo para execução dos testes | 60 |
| Figura 14 – Fluxo de pacotes TCP 80 com foco no firewall | 63 |
| Figura 15 – Fluxo de pacotes TCP 8080 com foco no firewall | 64 |
| Figura 16 – Fluxo de pacotes no balanceador de carga por endereço IP | 66 |
| Figura 17 – Fluxo de pacotes no balanceador de carga por Round Robin | 68 |
| Figura 18 – Média de transações por segundo de acordo com o número de usuários | 70 |
| Figura 19 – Números absolutos de erros de acordo com o número de usuários | 71 |
| Figura 20 – Tempo médio de resposta de acordo com o número de usuários | 72 |

Lista de Tabelas


| | |
|--|----|
| Tabela 1 – Exemplo de fluxo de pacotes que vai ao firewall | 47 |
| Tabela 2 – Exemplo de regras no firewall | 47 |
| Tabela 3 – Resultado do processamento no firewall | 47 |
| Tabela 4 – Divisão do fluxo de pacotes vindos do cliente com Round Robin | 53 |
| Tabela 5 – Divisão do fluxo de pacotes vindos de 3 clientes com endereço IP de origem | 53 |
| Tabela 6 – Especificações de hardware para os testes | 57 |
| Tabela 7 – Softwares para a VM hospedeira | 57 |
| Tabela 8 – Softwares para as VMs clientes | 58 |
| Tabela 9 – Softwares para as VMs servidoras | 58 |
| Tabela 10 – Recursos do hospedeiro (dom0) | 59 |
| Tabela 11 – Recursos das três VMs clientes | 59 |
| Tabela 12 – Recursos das três VMs servidoras | 59 |
| Tabela 13 – Recursos das duas VMs para as VNFs | 59 |

Lista de Acrônimos e Siglas

| | |
|--------|---|
| API | Sigla em língua inglesa para Interface de Programação de Aplicação |
| ARP | Acrônimo em língua inglesa para Protocolo de Resolução de Endereços |
| BSS | Sigla em língua inglesa para Sistemas de Suporte aos Negócios |
| BWM-NG | Sigla em língua inglesa para Monitor de Largura de Banda NG |
| CAPEX | Acrônimo em língua inglesa para Capital Investido |
| CSS | Sigla em língua inglesa para Folhas de Estilo em Cascata |
| CSV | Sigla em língua inglesa para Valores Separados por Vírgula |
| DHCP | Sigla em língua inglesa para Protocolo de Configuração Dinâmica de Dispositivos |
| DNAT | Acrônimo em língua inglesa para Tradução de Endereços de Rede de Destino. Não confundir neste trabalho com Tradução de Endereços de Rede Dinâmicos. |
| DPDK | Sigla em língua inglesa para Kit de Desenvolvimento em Plano de Dados da Intel |
| ETSI | Sigla em língua inglesa para o Instituto de Padronização de Telecomunicações Europeu |
| GNU | Acrônimo recursivo em língua inglesa para GNU não é Unix |
| HTTP | Sigla em língua inglesa para Protocolo de Transmissão de Hipertexto |
| IDS | Sigla em língua inglesa para Sistema de Detecção de Intrusão |
| IETF | Sigla em língua inglesa para Força-Tarefa de Engenharia da Internet |
| IRTF | Sigla em língua inglesa para Força-Tarefa de Pesquisa da Internet |
| IP | Sigla em língua inglesa para Protocolo da Internet |
| KVM | Sigla em língua inglesa para Máquina Virtual baseada em Núcleo |
| LAMP | Sigla em língua inglesa para a junção dos nomes de Linux, Apache, MySQL e PHP |
| MAC | Acrônimo em língua inglesa para Controle de Acesso ao Meio |

| | |
|----------|--|
| MIT | Acrônimo em língua inglesa para Instituto Tecnológico de Massachusetts, Estados Unidos |
| MSS | Sigla em língua inglesa para Tamanho de Segmento Máximo |
| NAT | Acrônimo em língua inglesa para Tradução de Endereços de Rede |
| NEC | Acrônimo em língua inglesa para Companhia Elétrica Nipônica |
| NFV | Sigla em língua inglesa para Virtualização de Funções de Rede |
| NFV MANO | Combinação de sigla e acrônimo em língua inglesa para Orquestração e Gerenciamento da Virtualização de Funções de Rede |
| NFVI | Sigla em língua inglesa para Infraestrutura de Virtualização de Funções de Rede |
| NIC.BR | Sigla em língua portuguesa para Núcleo de Informação e Coordenação do Ponto BR |
| OPEX | Acrônimo em língua inglesa para Despesa Operacional |
| OSI | Acrônimo em língua inglesa para Interconexão de Sistemas Abertos |
| PHP | Sigla em língua inglesa para Personal Home Page |
| OSS | Sigla em língua inglesa para Sistemas de Suporte às Operações |
| RED | Acrônimo em língua inglesa para Detecção Adiantada Randômica |
| RFC | Sigla em língua inglesa para Requisição Para Comentários |
| SDN | Sigla em língua inglesa para Rede Definida por Software |
| SNMP | Sigla em língua inglesa para Protocolo Simples de Gerência de Rede |
| SQL | Sigla em língua inglesa para Linguagem de Consulta Estruturada |
| SSH | Sigla em língua inglesa para o protocolo Shell Seguro |
| STP | Sigla em língua inglesa para Protocolo de Árvore Geradora |
| TCP | Sigla em língua inglesa Protocolo de Controle de Transmissão |
| UDP | Sigla em língua inglesa Protocolo de Datagramas de Usuário |
| VIM | Acrônimo em língua inglesa para Gerente de Infraestrutura Virtual |
| VM | Sigla em língua inglesa para Máquina Virtual |
| VNF | Sigla em língua inglesa para Função de Rede Virtual |

Sumário

| | | |
|-------|---|----|
| | Lista de Figuras | 13 |
| | Lista de Tabelas | 15 |
| 1 | INTRODUÇÃO | 21 |
| 1.1 | OBJETIVO | 22 |
| 1.2 | ESTRUTURA DO TRABALHO | 22 |
| 2 | CONCEITOS  | 23 |
| 2.1 | Virtualização de Funções de Rede – NFV | 23 |
| 2.2 | Linux Bridge | 25 |
| 3 | ENTENDENDO CLICKOS | 29 |
| 3.1 | Click | 29 |
| 3.2 | ClickOS | 38 |
| 4 | PROJETO E IMPLEMENTAÇÃO | 43 |
| 4.1 | Um dia dentro das funções propostas | 45 |
| 4.2 | Firewall | 47 |
| 4.3 | Balanceador de Carga | 52 |
| 5 | TESTES | 57 |
| 5.1 | Infraestrutura e softwares | 57 |
| 5.2 | Alocação de recursos | 58 |
| 5.3 | Protótipo | 60 |
| 5.4 | Proposta, execução e resultados | 61 |
| 5.4.1 | Firewall | 62 |
| 5.4.2 | Balanceador de carga por endereço IP de origem | 65 |
| 5.4.3 | Balanceador de carga por Round Robin | 67 |
| 5.4.4 | Desempenho | 69 |
| 6 | CONCLUSÃO | 73 |
| | Referências Bibliográficas | 75 |
| | REFERÊNCIAS BIBLIOGRÁFICAS | 75 |

1 INTRODUÇÃO

Os *middleboxes* são dispositivos de propósito específico que implantam serviços ao longo de redes físicas, promovendo segurança, isolamento, compartilhamento, otimização, gerenciamento, etc. São exemplos os *switches*, roteadores, *firewalls*, balanceadores de carga, priorizadores de pacotes, limitadores de banda, *proxies*, filtros de conteúdo, dentre outros (1). Durante muito tempo foram exclusivamente dispositivos de *hardware* com *software* embarcado, com alto custo e ciclo de vida reduzido, diante do rápido desenvolvimento das redes de computadores. No âmbito de atualizações e manutenções, não são equipamentos modulares, o que somado ao fim do **suporte ocasiona** a troca dos aparelhos, prejudicando o capital investido (CAPEX) e a despesa operacional (OPEX) das empresas usuárias.

No entanto, este modelo não está mais sozinho. Esforços para a construção de *middleboxes* em *hardware* de propósito geral já eram discutidos desde o fim da década de noventa, com Click (2), por exemplo. O início deste século trouxe o processamento em ~~altas frequências~~, junto com a capacidade de paralelizá-lo, e mais a frente, a viabilidade da camada de virtualização de maneira geral, com menor perda de desempenho, grande eficiência energética e econômica ~~para os investidores~~. Desta forma, os *middleboxes* dissociados de *hardware* se tornaram viáveis, na forma de **funções de rede virtuais** (VNFs) com o uso de tecnologias como *containers* e paravirtualização.

Um dos padrões em desenvolvimento nesse sentido é o de Virtualização de Funções de Rede (NFV), que sob responsabilidade do Gerente de Infraestrutura Virtual (VIM) aloca recursos de um hardware de propósito geral para prover funções personalizadas, habilitando escalabilidade, manutenibilidade, maior ciclo de vida de hardware e software, e economia de recursos e investimentos – justamente os pontos fracos do modelo acoplado ao hardware. Citado anteriormente como exemplo, Click foi um dos casos de sucesso em matéria de desacoplar código, sendo o equivalente de sua época a uma função de rede virtual. E mais de uma década depois, sua existência **e vantagens** se tornaram parte de um dos projetos de virtualização de dispositivos **de rede: chama-se** ClickOS (1).

ClickOS otimiza a trinca Click, Projeto Xen (3) e MiniOS (4) em uma plataforma integrável a soluções NFV como mostra (5), para:

- a) uma fácil implementação de VNFs com uma linguagem de marcação voltada para visão de fluxos, sendo de fácil legibilidade;
- b) ~~para o~~ provimento da camada de virtualização ~~necessária~~, incluindo a gerência dos recursos das máquinas virtuais;

- c) ~~para~~ prover um *micro-kernel* suficientemente robusto e pequeno para que as VNFs sejam executadas de maneira escalável com o devido isolamento em um ambiente de paravirtualização.

No entanto, até pela questão das disputas do nicho e pelo ainda recente desenvolvimento de soluções para provimento de VNFs, a plataforma sofre com a falta de materiais didáticos, sejam em inglês ou português, mesmo aqueles que tratam apenas de Click, desestimulando seu uso em detrimento de outros semelhantes como ~~atualmente acontece com~~ P4 (6). Como exemplo dessa questão, é possível citar a falta de documentações sobre: a integração de Click com os *switches* VALE (7) e OpenVSwitch (8) através de ClickOS; as documentações de sua instalação; a falta de exemplos sobre o correto uso dos elementos; exemplos implementados e comentados; e documentação oficial incompleta.

1.1 OBJETIVO

Este trabalho apresenta a plataforma para implementação de VNFs ClickOS, destacando sua importância e atuação dentro de NFV, entregando um rápido desenvolvimento e manutenção de funções, e consolidando a virtualização de dispositivos de redes como alternativa viável aos dispositivos físicos semelhantes.

Como objetivo específico, no trabalho implementa-se duas funções de rede virtuais em ClickOS detalhadamente, com um cenário comum ao balanceamento de carga em serviços de hospedagem *web*, e avalia a prova de conceito da plataforma a partir de testes.

1.2 ESTRUTURA DO TRABALHO

Para além deste, o [Capítulo 2](#) trata dos conceitos principais associados ao trabalho; o [Capítulo 3](#) visa detalhar os mecanismos de Click e ClickOS para suas aplicações no trabalho; o [Capítulo 4](#) projeta e executa a realização do objetivo específico, percorrendo sobre as funções de rede e o comportamento esperado, exemplificando passo a passo os rastros de uma comunicação dentro do cenário, destacando vantagens adquiridas com o uso das ferramentas e plataformas; o [Capítulo 5](#) versa sobre a realização e os resultados dos testes no cenário proposto, buscando confirmar a posição de ClickOS como alternativa aos dispositivos físicos de rede; o [Capítulo 6](#) caminha no sentido de analisar o esforço realizado, enfatizar pontos positivos e negativos da plataforma e da proposta, e encaminhar possíveis trabalhos futuros.

2 CONCEITOS

Este capítulo expõe rapidamente dois conceitos necessários para entender o ecossistema onde o trabalho proposto se insere. Na Seção §2.1, uma rápida introdução ao contexto de NFV é exposta, com suas motivações e realizações que possibilitam a discussão da virtualização dos caros dispositivos de rede.

Já na Seção §2.2, o *switch* virtual é posicionado dentro do cenário de virtualização, com o cuidado de explicar ao máximo seu mecanismo básico de uso, como insumo para o restante do trabalho.

~~Mais conceitos são tratados no Capítulo 3, no entanto a riqueza de detalhes faz deste um capítulo com destaque.~~

2.1 Virtualização de Funções de Rede – NFV

O modelo NFV foi proposto em 2012 pelas empresas de telecomunicações em conjunto com o Instituto de Padronização de Telecomunicações Europeu (ETSI), tendo a redução de CAPEX e de OPEX como principal objetivo. Para tal, a proposta é desacoplar as funções em software dos equipamentos de rede hospedeiros (9).

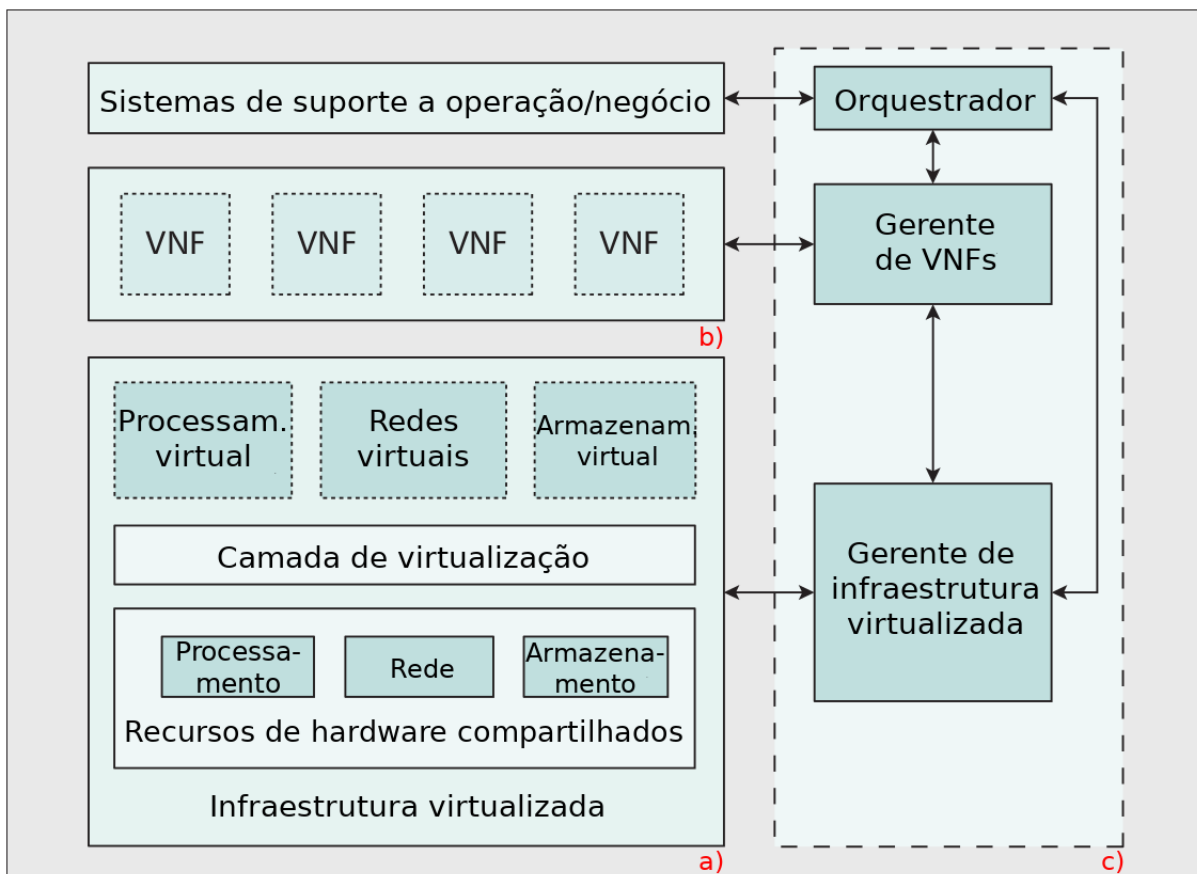
Para o ETSI, paradigma já possui uma arquitetura especificada, embora ainda não fechada ou padronizada. Os itens mais importantes são descritos da seguinte forma:

- a) Infraestrutura NFV (NFVI): combinação de recursos de *hardware* e *software* sobre os quais as funções de rede virtualizadas são desenvolvidas. As abstrações dos recursos são atingidas ao inserir sobre elas uma camada de virtualização usando *hypervisors*, habilitando então a separação entre as funções de rede e os recursos físicos. Máquinas Virtuais (VMs) representam o uso de recursos de processamento e armazenamento, enquanto elos e nós virtuais (sistemas operacionais em VMs) representam o uso de recursos de rede (9). São exemplos de *hypervisors* o Projeto Xen, KVM (10), HyperV (11), VMware vSphere (12), VirtualBox (13).
- b) Serviços e Funções de Rede Virtuais: funções de rede são normalmente dispositivos dentro de uma infraestrutura de rede com interfaces externas e comportamento bem definidas (14). Alguns exemplos mais comuns são: firewall, roteadores, balanceadores de carga. Boa parte delas também é conhecida pelo termo *middlebox* (15). Já as VNFs são representações implementadas e implantadas em recursos virtuais, que podem conter diversos elementos internos de rede, e ser instalada em diversas soluções virtualizadas como VMs e containers

Docker (16) ou Linux (17), seja de forma a replicar funcionalidades ou de forma a dividir tarefas de uma dada função (9). Do ponto de vista do usuário, as VNFs devem ser idênticas as funções de rede, inclusive em termos de desempenho (9).

- c) Orquestração e Gerenciamento NFV (NFV MANO): funcionalidades para prover as VNFs e suas operações relacionadas, tais como as configurações de recursos, de ciclo de vida, e o comportamento das funções virtuais, sob responsabilidade do VIM. Isso inclui a orquestração e o gerenciamento do ciclo de vida de recursos de software e hardware que dão suporte a virtualização. NFV MANO foca em tarefas específicas para gerenciamento de virtualização no framework NFV. Já o **framework** é responsável por definir as interfaces para comunicação de componentes do NFV MANO, tal qual a coordenação de sistemas tradicionais de gerenciamento como os Sistemas de Suporte às Operações (OSS) e os Sistemas de Suporte aos Negócios (BSS) (9).

Figura 1 – Representação do modelo NFV



Fonte: Adaptado de (18)

Com a arquitetura descrita, várias implementações tiveram seu desenvolvimento iniciado, como OPNFV (19) e OpenStack (20) e órgãos de padronização já discutem formas de universalizá-lo mesmo que questões técnicas e de negócios estejam abertas, como em NFVRG (21) da Força-Tarefa de Pesquisa da Internet (IRTF), associada a Força-Tarefa de

Engenharia da Internet (IETF). Segundo (9), algumas dessas questões são: a) controle e gerência de equipamentos legados; b) quais funções de rede virtuais devem ser implantadas em *datacenters* ou operadoras; c) quais devem estar dentro de VMs ou *containers*; d) quais recursos do NFVI são necessários para desenvolver uma determinada função virtual; e) quais os requerimentos operacionais de ambientes que operam VNFs e equipamentos legados.

O modelo ~~projetado~~ traz diversas vantagens em relação ao modelo de software acoplado a hardware além das economias financeiras por si só. A programabilidade das funções de rede permitem um desenvolvimento, manutenção e implantação de acordo com os anseios do cliente, de maneira mais rápida e estimulando um novo mercado de serviços de software para um novo extrato de empresas. Já o advento da virtualização traz a tona o uso de recursos de *hardware* sob demanda, agrupando o que antes eram diversos equipamentos com recursos mal utilizados em um conjunto de *hardwares* potentes, capaz de escalar recursos conforme o uso da rede se torne mais intenso, com custo mínimo para o desempenho, e menor para o investidor.

2.2 Linux Bridge

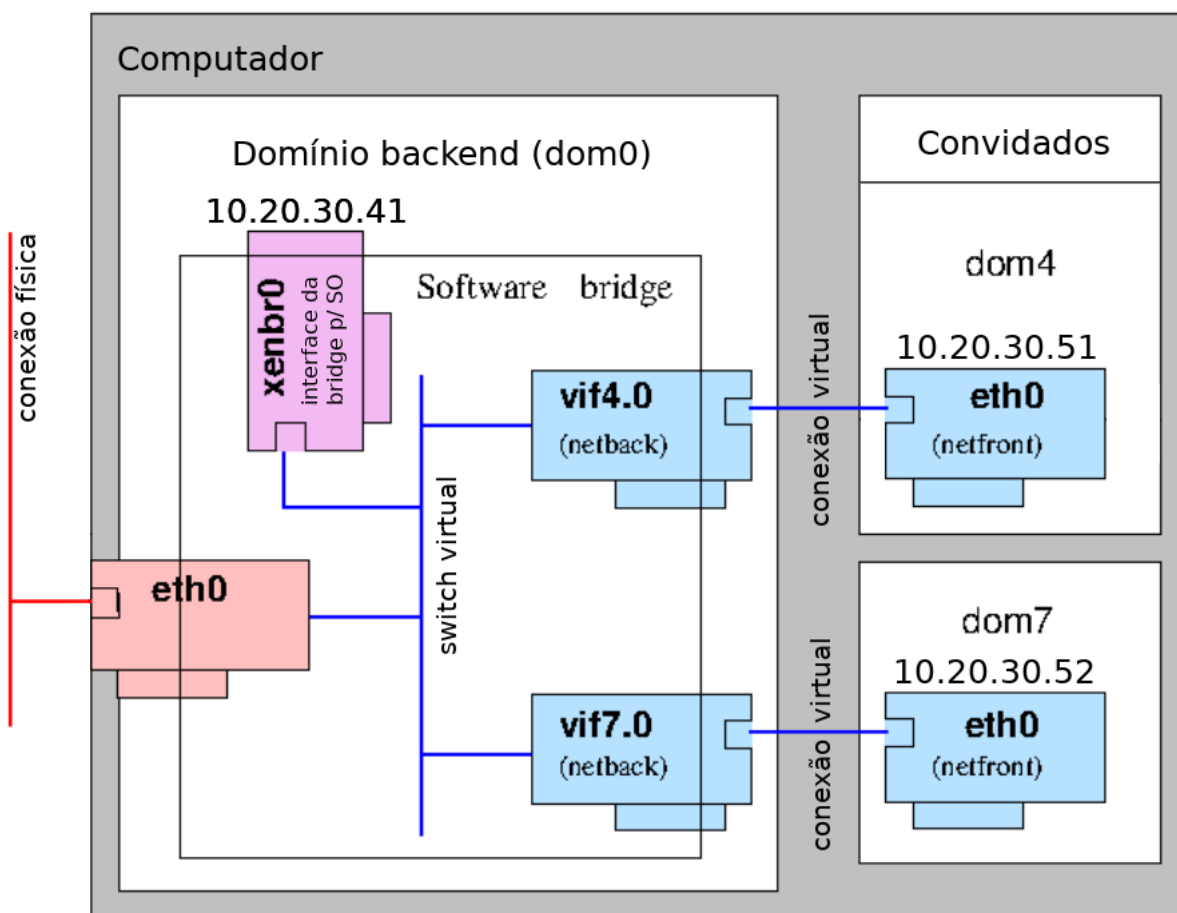
Os dispositivos de rede chamados comutadores (*switches*) são responsáveis por conectar diversos nós entre si para assim facilitar uma rede local (22). O objetivo da versão virtualizada desses equipamentos é basicamente igual, embora guarde diferenças de implementação como visto em (8). Os *switches* virtuais provêem conectividade entre VMs e destas com o mundo externo, envolvendo muitas das mesmas funções providas pelas versões físicas (8) para centenas de dispositivos virtualizados por hospedeiro.

Um dos expoentes dessa classe é o OpenVSwitch, que leva programabilidade para as redes ao abordar a questão de Redes Definidas por Software (SDNs). No entanto outros mais simples podem ser úteis em determinadas situações. É o caso do VALE (7) e do Ethernet Linux Bridge (22), ~~que é o *switch* deste trabalho~~. O Linux Bridge faz parte do *kernel* Linux (23) como um módulo e é dividido de forma grosseira em quatro componentes como a maioria desses dispositivos (22):

- a) O conjunto de portas usado para encaminhar tráfego entre os dispositivos ligados a ele, reais ou virtuais;
- b) O plano de controle usado para executar o Protocolo de Árvore Geradora (STP), que calcula a árvore geradora mínima para a rede local, prevenindo que repetições infinitas quebrem a rede;
- c) O plano de encaminhamento responsável por processar quadros que entram pela porta de rede, fazendo uma decisão de encaminhamento sobre qual porta deve recebê-los;

- d) O aprendizado de endereços de Controle de Acesso ao Meio (MAC) (24) mantém registrado em um banco de dados (tabela) a porta de rede ~~(no caso, virtual)~~ na qual o endereço MAC está conectado, e portanto, servindo de consulta para encaminhar quadros para seu destinatário. Caso não conste o endereço, o quadro é enviado para todas as portas exceto a de origem.

Figura 2 – Estrutura das interfaces virtuais de rede no Linux Bridge



Fonte: Adaptado de (25)

A título de descrição, a Figura 2 retrata o conjunto de portas de um *switch* virtual. Ele possui quatro portas ocupadas, sendo duas com interfaces virtuais de duas VMs convidadas (*vif4.0* e *vif7.0*), uma porta com a interface física do **hospedeiro para** comunicação externa a camada de virtualização, e por último uma porta conectada ao elemento de rede que identifica o computador conectado. Este último recebe um endereço do Protocolo de Internet (IP) (26) caso queira que o hospedeiro esteja conectado a rede dos convidados.

Dada o comutador virtual criado e configurado por um assistente como o *brctl* no GNU/Linux, um conjunto de quadros pode ser lançado da VM *dom4* para a *dom7* através da camada de enlace. Dado o conjunto de portas, os quadros trafegam como esperado a partir do momento que: o plano de controle não detecta caminhos circulares; o plano

de encaminhamento solicita a informação sobre qual porta enviar o fluxo para a `dom7`, e recebe essa informação do plano de aprendizado de MACs ~~através de consulta ao seu banco de dados ou fazendo uma busca.~~

Após realizar as configurações, a comunicação entre as duas máquinas virtuais se dá da seguinte forma: `dom4` envia quadros através da sua interface de rede de saída, entrando então na interface virtual `vif4.0`, que por sua vez os encaminha automaticamente para o processamento dos componentes do *switch* citados anteriormente. Tendo o destino sido detectado na porta a qual se conecta a `vif7.0`, o *switch* encaminha os quadros até a sua entrada, que por sua vez encaminha automaticamente em saída para a interface de rede da VM. Um detalhe importante a se comentar é que todo esse tráfego é transparente para o hospedeiro, fazendo de sua segurança e isolamento algo primordial.

Uma das formas de anexar VMs ao *switch* virtual criado é através dos arquivos de configuração de recursos ~~das próprias~~. No caso do Xen, com as entradas `vif = []`. A Figura 5 demonstra um exemplo mais concreto em seu topo, porém um exemplo simplificado é `vif=['bridge=mybridge']`, onde o Xen atribui aleatoriamente um endereço MAC e conecta a máquina ao comutador virtual `mybridge` (25).

3 ENTENDENDO CLICKOS

Este capítulo expõe de forma aprofundada as duas principais tecnologias utilizadas neste trabalho. Na Seção §3.1, Click é dissecado com detalhes sobre seu funcionamento, entregando uma literatura básica para que seja possível compreender e posteriormente desenvolver funções de rede a partir do zero. Alguns pequenos problemas encontrados durante o desenvolvimento deste trabalho também recebem destaque, junto com seu principal caso de uso atual no âmbito de NFV, e sua ferramenta de geração de grafos para maior visibilidade quando necessário.

Já na Seção §3.2, o mesmo detalhamento com relação a tecnologia ClickOS e seu uso é feito, mantendo o cuidado com a função de uma literatura de consulta para o leitor. Todo o processo para a instanciação de uma máquina virtual é citado, mesmo de forma ampla, deixando a implementação para capítulos posteriores.

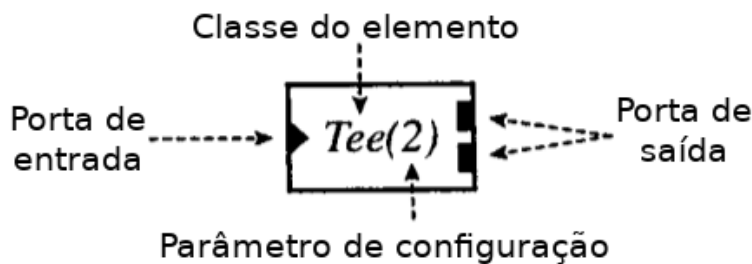


3.1 Click

A arquitetura Click foi idealizada por Robert Morris, Eddie Kohler e a equipe do Laboratório de Ciências da Computação do Instituto Tecnológico de Massachusetts (MIT) em 1999, que propuseram e desenvolveram uma plataforma em *hardware* geral e em *software* livre, para permitir o desenvolvimento de funções de rede sob demanda, e sem as limitações corriqueiramente impostas por soluções proprietárias em redes. Não há restrição de camadas ou protocolos de transmissão, bastando adicionar seu suporte, porém o foco está na camada 2 e 3 do modelo OSI (27), com diversas operações na camada 4.

Click trabalha com uma unidade chamada elemento. Ele representa a execução de uma atividade esperada no pacote recebido e deve ser conectado a outros como em um grafo, permitindo o fluxo de pacotes. A junção de vários desses obedecendo uma lógica cria uma VNF como um *firewall*, balanceador de carga, ou qualquer outro processador de pacotes. Fazem parte do projeto cerca de trezentos elementos pré-implementados pelos desenvolvedores e parceiros, em código aberto, e podem ser alterados conforme desejado desde que o editor tenha conhecimentos em C++ (2), (28).

Figura 3 – Descrição gráfica de um elemento em Click



Fonte: Adaptado de (2)

Uma lista não extensa desses elementos com base na documentação do Click disponível (29):

- a) Protocolo Ethernet: `EtherEncap`, `EtherSwitch`, `EtherVLANEncap`, `SetEtherAddress`;
- b) Protocolo ARP: `ARPQuerier`, `ARPResponder`; `ARPTTable`; `CheckARPHeader`;
- c) Protocolo IPv4: `CheckIPHeader`, `IPEncap`, `IPFragmenter`, `SetIPAddress`, `SetIPChecksum`;
- d) Protocolo IPv6: `CheckIP6Header`, `SetIP6Address`, `ProtocolTranslator64`, `ProtocolTranslator46`;
- e) Classificadores: `Classifier`, `IPClassifier`;
- f) Roteamento: `StaticIPLookup`, `LinearIPLookup`, `DirectIPLookup`, `IPRouteTable`;
- g) NAT: `IPRewriter`, `ICMPRewriter`, `TCPRewriter`, `RoundRobinIPMapper`, `SouceIPHashMapper`;
- h) Modelagem de tráfego (*traffic shaping*): `BandwidthShaper`, `RatedSplitter`, `BandwidthRatedSplitter`, `Shaper`;
- i) Escalonamento (*scheduling*): `PrioSched`, `RoundRobinSched`;
- j) Gerenciamento de filas: RED (Detecção Adiantada Randômica), CoDel (Controlled Delay);
- k) Redes sem Fio: `WifiEncap`, `WifiDecap`, `WirelessInfo`, `BeaconScanner`, `OpenAuthRequester`;
- ~~l) e demais elementos de controle, de testes e de procura de erros.~~

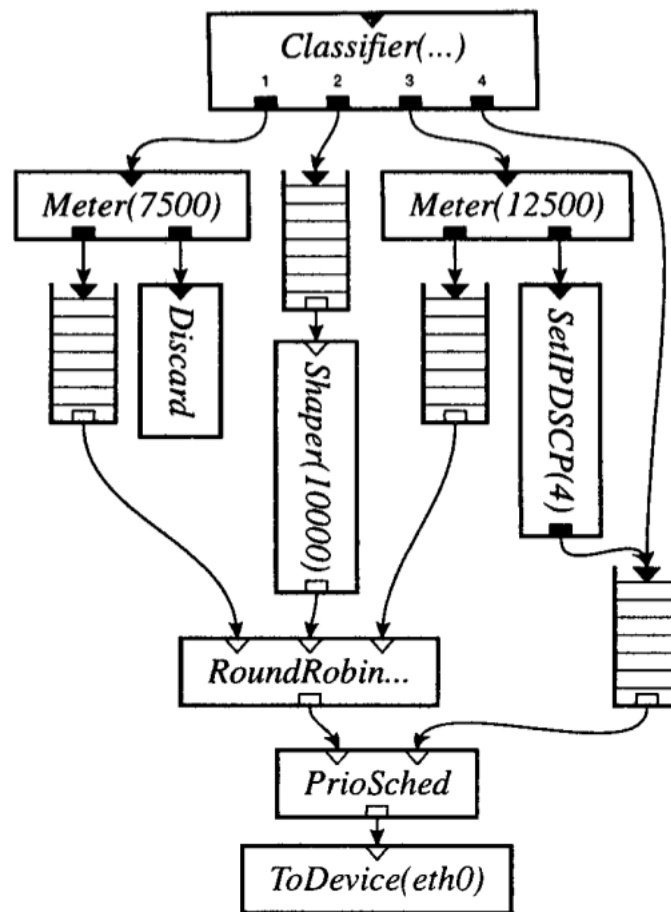
A reutilização de código é visível, porém a legibilidade é tão importante quanto. Para isso, a ferramenta dispõe de uma linguagem marcação própria que esconde sua linguagem de programação base. Ela evidencia uma visão em grafos de fluxo, onde os elementos são nós e as arestas são as conexões por onde fluem os pacotes. Cada elemento possui especificações sobre o número de portas de entrada e saída, que devem ser conectadas no decorrer da implementação, no arquivo da função virtual. Veja a Figura 3e a Figura 5 para exemplos.

Falando sobre portas, elas podem ser do tipo de fluxo *pull* ou *push*, imitando a relação produtor-consumidor presente em sistemas distribuídos. Os elementos com portas *pull* aguardam um sinal para enviar quadros ou requisitam através de um para receber outros do seu elemento vizinho. Já as portas *push* possuem um comportamento de envio indiscriminado, e portanto, os elementos que as utilizam devem estar sempre disponíveis para receber pacotes ou eles se perdem. Por fim, há portas chamadas agnósticas, que assumem o comportamento do fluxo de processamento predominante(2). Veja a Figura 4 para uma melhor visualização.

A existência de ambos se dá pela natureza do uso de uma rede de alta velocidade compartilhada como as redes *Ethernet* (24): esvazie seu *buffer* de entrada de quadros o mais rápido possível para que possa continuar recebendo-os sem perdas; aguarde a sua vez para enviar os seus quadros à rede o mais rápido possível sem degradar outros pacotes. Desta forma, percebe-se que o mecanismo *push* compõe o recebimento e o mecanismo *pull* o envio. Já as filas, que são o único elemento capaz de converter o fluxo de *push* para *pull*, são responsáveis por armazenar os pacotes processados mas impedidos de ir ao *buffer* de saída do equipamento (rede bastante congestionada, com muito recebimento de quadros pelos dispositivos e por outro lado pouca saída).

Ainda em relação as portas, algumas regras devem ser obedecidas: olhando a partir de um elemento, uma dada porta de saída não pode ser conectada a várias portas de entrada de outros elementos, pois geraria efeito de duplicidade e inconsistência na rede; as portas de entrada do tipo *pull* não podem se conectar a mais de uma porta de saída; e nenhum outro elemento é capaz de converter tipos de fluxo exceto as filas, como comentado.

Figura 4 – Condicionador de tráfego em Click



Fonte: (2)

Dado os parâmetros do elemento classificador (*Classifier*), os pacotes que entram nele com diferentes características possuem quatro caminhos:

- Um chaveador de fluxo (*Meter*) que limita em no máximo 7500 quadros por segundo o fluxo, sendo então despejados em uma fila caso respeitem essa taxa ou então descartados pela outra porta de saída;
- Uma fila acoplada a um elemento limitador de fluxo (*Shaper*), que solicita pacotes a uma taxa de no máximo 10000 quadros por segundo através de porta *pull*;
- Outro chaveador de fluxo com 12500 quadros por segundo em outra fila, ou a passagem no elemento que marca o campo de serviços diferenciados do cabeçalho IP para o valor 4 antes de despejar em uma quarta fila;
- Quarta fila aliás, que também recebe o fluxo que sai pela porta 4 do classificador.

O percurso dos três primeiros caminhos terminam em um escalonador do tipo *Round Robin* (30), que após seu processamento, ~~este~~ ainda sofre o processamento de outro es-

calonador baseado em prioridade ordenada, onde a primeira entrada tem precedência sobre as seguintes e assim segue da esquerda para a direita. Ao fim, os pacotes saem pela interface escolhida no elemento de envio (`ToDevice`).

Ainda sobre a Figura 4, ao passar pelas filas a coloração das portas muda, indicando a mudança de fluxo de processamento de *push* para *pull*. Outra observação interessante é quanto a fila do quarto **caminho que** recebe as saídas de processamento de dois elementos, evidenciando que as portas de entrada *push*, ao contrário das portas *pull*, podem receber fluxos de múltiplas portas de saída.

O exemplo acima possui uma visão em grafo. Já ~~o presente na~~ Figura 5 demonstra através da linguagem de marcação do Click uma estrutura mínima para atuar sobre o protocolo IP:

- a) Todos os momentos onde `::` é utilizado significa uma definição de nome para o elemento criado. Toda vez que uma `->` é utilizada significa uma ligação entre a porta de saída X e a porta de entrada Y de dois elementos;
- b) O `AdressInfo` é como um banco de dados rústico, **para** guardar dados sobre um dispositivo e usar **a** macro ~~atribuída~~ para representá-los;
- c) `FromDevice` e `ToDevice` fazem respectivamente o recebimento dos quadros do *buffer* de rede de entrada e o envio de outros para o *buffer* de rede de saída;
- d) ~~Já~~ o `Classifier` classifica os pacotes com base na comparação de valores presentes nos cabeçalhos, **onde os dois primeiros parâmetros do exemplo são códigos referentes aos dados do protocolo ARP (31) (consulta e resposta, respectivamente), o terceiro parâmetro contém o código para os pacotes com cabeçalho IP e o quarto para todos os outros.** A análise do cabeçalho é feita de forma que o elemento procura o **código** do cabeçalho em determinado **byte** do quadro, formando o par *byte/código*. Em caso de mais de um par, a procura se torna mais específica, buscando casar **mais um valor** no *byte* indicado, o que nesse caso é a diferença entre consulta e resposta ARP;
- e) `ARPQuerier` e `ARPResponder` realizam o papel de preencher a tabela ARP de um dispositivo. O primeiro, diante da não existência de uma entrada para o destino solicitado por um pacote, consulta ao `ARPResponder` de todos os dispositivos na rede acerca do endereço MAC associado a um dado endereço IP, e ao receber a resposta, encapsula o pacote em um quadro. O mesmo vale quando o portão (*gateway*) se torna o destinatário temporário, o que acontece quando o `SetIPAddress` é usado (detalhes no [Capítulo 4](#)), por exemplo. Como **adiantado**, ~~o segundo elemento desta alínea~~ responde às consultas feitas por outros dispositivos com `ARPQuerier` e pode ser utilizado para realizar um *Proxy ARP* (32);

- f) Queue define uma fila de 200 entradas para receber os pacotes do fluxo *push* para entregar quando solicitado pelos elementos do fluxo *pull*;
- g) Conexões entre: FromDevice e o Classifier; Queue e ToDevice; as requisições ARP vindas do Classifier e o ARPResponder; deste último e a Queue; as respostas ARP vindas do Classifier e a porta de entrada 1 do ARPQuerier; de um elemento de controle de erro (Idle) e a entrada 0 do ARPQuerier para evitar mensagens de erro, e a conexão deste mesmo elemento a uma fila de tamanho 200 para escoar os pacotes encapsulados;
- h) Os pacotes IP classificados no Classifier saem pela porta 2 e então podem ser modificados como desejado. ~~Inclusive~~, não é necessário inserir todos os códigos no bloco indicado para preenchimento, desde que ao fim se conectem desse ponto até o último elemento. Esse aliás, deve se conectar ao ARPQuerier para que os pacotes sejam encapsulados e sigam o caminho até o elemento que proporciona a saída da rede;
- i) Os demais pacotes saem pela porta 3 e são descartados ao ligar tal saída do Classifier ao Discard.

Figura 5 – Uma aplicação dos elementos mínimos para ações sobre o protocolo IP

```

1 // Rede no vnf.cfg:-
2 // vif          = ['ip=10.0.0.1,mac=00:19:85:11:00:54,bridge=suaBridge',~
3 //              'ip=outroIP,mac=outroMAC,bridge=outraBridge']~
4 |
5 //              name          ip          ipnet          mac (def em vnf.cfg)~
6 AddressInfo(net0          10.0.0.1          10.0.0.0/8          00:19:85:11:00:54, ...);~
7 |
8 entrada :: FromDevice(0); //0->1ª vif declarada no vnf.cfg~
9 saida :: ToDevice(0);~
10 |
11 classificador :: Classifier(12/0806 20/0001, // saida 0. pacotes consulta ARP~
12 |.....|.....|.....|.....|.....|.....12/0806 20/0002, // saida 1. pacotes resposta ARP~
13 |.....|.....|.....|.....|.....|.....12/0800,          // saida 2. pacotes IP~
14 |.....|.....|.....|.....|.....|.....-);          // saida 3. outros~
15 |
16 // arpQ :: ARPQuerier(net0:ip, net0:mac); //ou~
17 arpQ :: ARPQuerier(net0);~
18 |
19 // arpR :: ARPResponder(net0:ip, net0:mac); //ou~
20 arpR :: ARPResponder(net0);~
21 |
22 fila :: Queue(200);~
23 |
24 entrada -> classificador;~
25 fila -> saida;~
26 classificador[0] -> arpR -> fila; //ou assim: arpR[0], [0]arpR, [0]arpR[0]~
27 classificador[1] -> [1]arpQ;~
28 Idle -> [0]arpQ;~
29 arpQ -> fila;~
30 |
31 classificador[2] -> /SEUS ELEMENTOS/~
32 |.....|.....|.....|.....|.....|.....-> /PARA MANIPULAR PACOTES IP/~
33 |.....|.....|.....|.....|.....|.....-> /OPERAM NO FLUXO A PARTIR DAQUI/~
34 |.....|.....|.....|.....|.....|.....-> arpQ; //em uma linha também funciona (veja linha 27)~
35 |
36 classificador[3] -> Discard;~

```

Fonte: Elaborada pelo autor

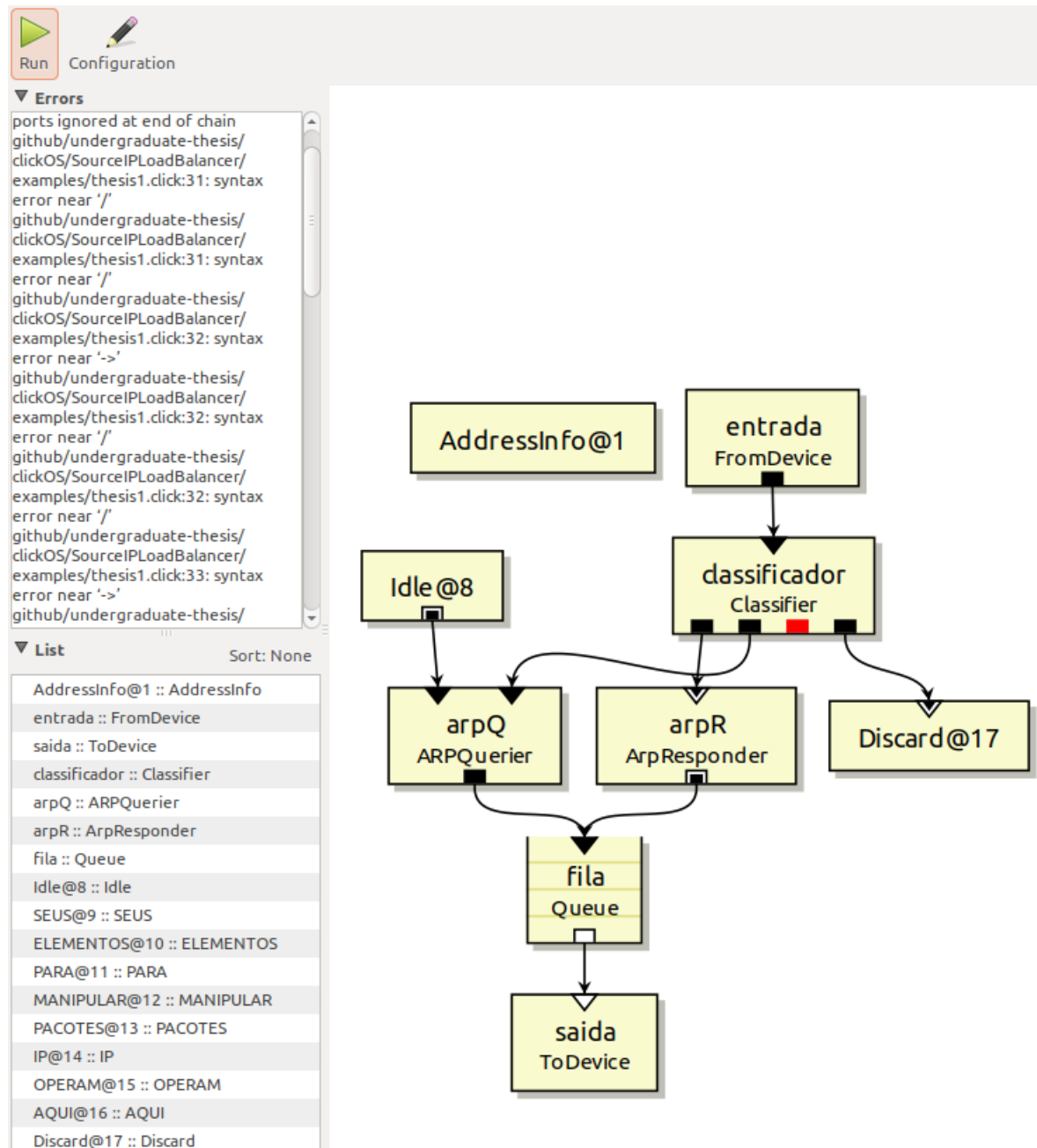
Todos os elementos podem ser consultados na documentação oficial do Click em (29) para um maior entendimento ~~de detalhes que fogem este trabalho~~. Outra fonte interessante são os livros, artigos, padronizações e textos técnicos sobre o uso dos protocolos de rede como por exemplo as Requisições Para Comentários (RFCs) do IETF.

A linguagem de marcação de Click possui um analisador que encontra erros de sintaxe e semântica em tempo de execução para auxílio no desenvolvimento. Na mesma linha, a ferramenta disponibiliza manipuladores de elementos (*element handlers*) básicos para que seja possível analisar comportamentos indesejados em tempo de execução. Um exemplo é a tabela do protocolo TCP (33) disponibilizada pelo elemento de NAT (IPRe-

writer), que pode indicar o correto funcionamento ou não de uma VNF que a utilize. Outros manipuladores são capazes de alterar determinadas variáveis em tempo de execução, mas não são muito comuns. De qualquer forma, Click permite a implementação de outros manipuladores para os elementos dado um conhecimento em C++. Para consultar quais manipuladores estão implementados, basta ~~consultar~~ a documentação citada.

Outra forma de facilitar a correção de erros, e também facilitar o entendimento de uma implementação em Click é através do uso da ferramenta de geração de grafos distribuída com Click chamada **Clicky**. Ela transforma elementos e portas em uma representação gráfica de um grafo, baseado em um arquivo de Folhas de Estilo em Cascata (CSS), e também é capaz de analisar erros e retorná-los. Na Figura 6, pode-se ver o grafo em Clicky, representando o código descrito na Figura 5, exceto o ramo da porta 2 que encontra erros devido ao bloco fora dos padrões e por isso é ignorado. É possível exportar os grafos para arquivos.

Figura 6 – Interface gráfica e grafo padrão da Figura 5 em Clicky



Fonte: Elaborada pelo autor

Durante o desenvolvimento deste trabalho, algumas limitações nos elementos foram observadas:

- Para criar balanceadores de carga, os elementos existentes são limitados ao mecanismo de *Round Robin* (`RoundRobinIPMapper`) e ao mecanismo de distribuição por IP de origem (`SourceIPHashMapper`), sem uma análise de carga dos servidores que recebem o balanceamento, através do SNMP (34), por exemplo;
- A tabela TCP criada no NAT através dos elementos acima não se apaga como deveria (35) devido a um erro de implementação que reinicia o contador res-

ponsável por sua expiração, e pode assim criar instabilidade quanto aos balanceamentos. Desta forma os balanceadores só garantem o balanceamento em tempo de criação da regra, e uma vez que elas permaneçam indefinidamente, determinadas conexões perenes podem desbalancear o sistema, já que não seriam rebalanceadas. Este erro foi reportado a comunidade, mas não a tempo de ser corrigido para este trabalho;

- c) A manipulação de mensagens do Protocolo de Datagrama do Usuário (UDP) (36) maiores que o Tamanho de Seguimento Máximo (MSS) deste é um problema quando se desenvolve um NAT em Click, já que para o recálculo da soma de verificação (*checksum*) usando o `UDPChecksum`, seria necessário remontar a mensagem UDP (36), o que Click não disponibiliza, comprometendo o desenvolvimento de alguma função de rede nessa situação.

Click possui um módulo para operar em espaço de *kernel* do Linux, em substituição ao módulo do sistema operacional. Isso significa um desempenho maior em relação a sua execução em espaço de usuário. É em espaço de *kernel* que os desenvolvedores, à época, conseguiram desempenho próximo ao módulo padrão do Linux (2). No entanto, atualmente o módulo do Click, encontra-se bastante defasado, compatibilizando apenas com a versão do *kernel* Linux 2.6, em face da versão 4.16 atual. Esta defasagem faz com que seu uso em um sistema operacional tradicional seja desestimulado, devido a complexidade em se programar para essa parte de um sistema. Seu uso atual possui escopo em *middleboxes* virtualizados, como em ESCAPE (37), CliMB (38), ClickOS (1) (ver Seção §3.2) e outros não relacionados a um *kernel* como para o Kit de Desenvolvimento em Plano de Dados (DPDK) com FastClick (39).

~~Desde o título este trabalho se propõe a trabalhar com uma dessas aplicações que cresceram em torno de Click. ClickOS especializou uma micro VM, desenvolvendo novos elementos, adaptando seu *micro kernel* para executar Click com seu módulo *kernel* de forma eficiente atingindo 10Gb/s para a maioria dos tamanhos de pacotes (1).~~

~~OBS: Parei aqui <~~

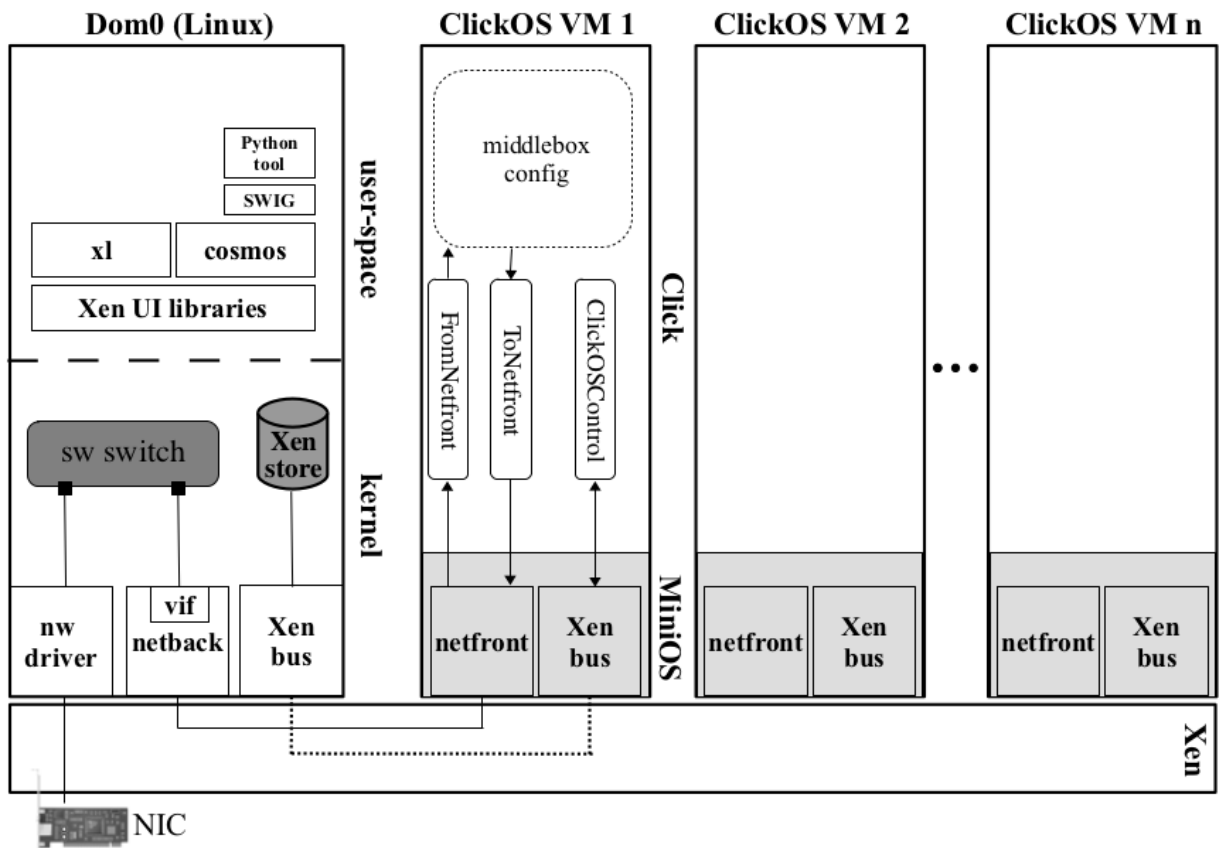
3.2 ClickOS

ClickOS é parte de um projeto de pesquisa da Universidade politécnica de Bucarest junto com a Companhia Elétrica Nipônica na Europa – NEC Europe – para o provisionamento de VNFs que possam substituir middleboxes de hardware. Ele se encontra representado no modelo NFV pela camada de virtualização, gerenciando localmente as funções de rede virtuais. Trata-se primeiramente de um sistema operacional baseado em Click, focado em alta vazão, baixo atraso e isolamento (40). Porém seus criadores objetivam também uma plataforma de alta performance com flexibilidade, escalabilidade

e reúso de código no topo de hardwares de propósito geral (1).

Para a alcinha de plataforma fazer sentido, os limites de um sistema operacional para micro-máquinas virtuais devem ser superados. ClickOS se apoia no projeto Xen e no VALE Switch (7) e propõe mudanças significativas as estruturas de ambos ao inserí-los em sua solução.

Figura 7 – Arquitetura inicial presente no ClickOS



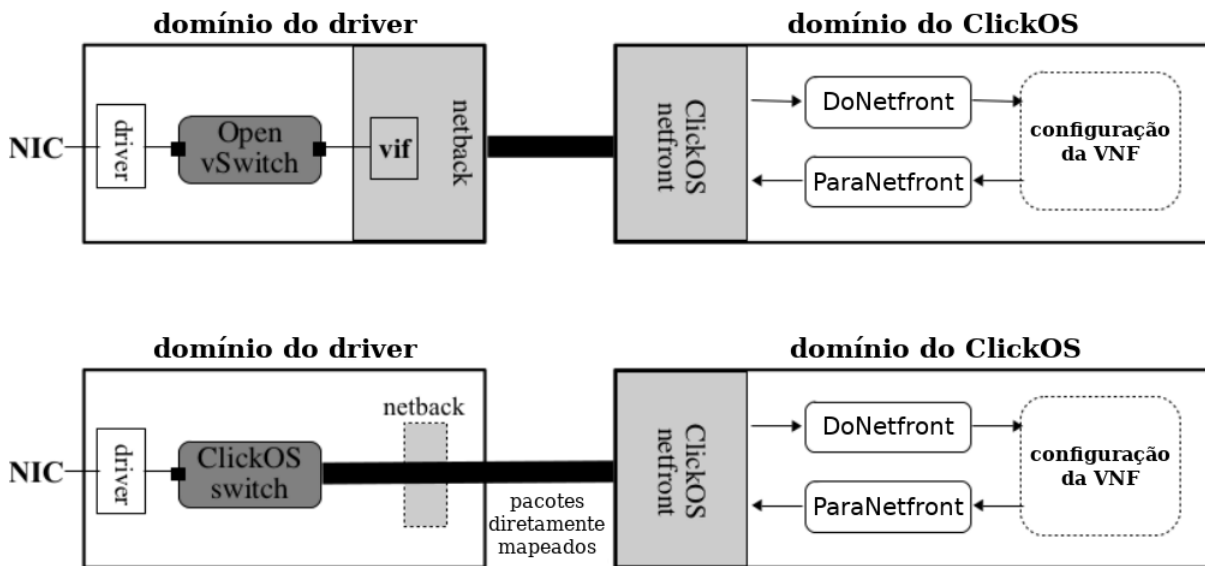
Fonte: (1)

No Xen, **hypervisor** de virtualização desenvolvido pela comunidade de software livre, várias otimizações foram realizadas segundo (1). dentre elas, se destacam: o acesso aos pacotes recebeu uma atenção especial ao remover intermediários da comunicação entre as máquinas virtuais e o **switch** virtual como o **driver netback**, que passou a ter um papel de controle apenas; já o **driver** netfront associado as VMs foi reprogramado para colocar na memória de sua VMs o **buffer** da porta do switch; por fim, as interfaces virtuais do Xen (vifs) foram ignoradas e então removidas.

Já o VALE Switch, dispositivo de comutação de pacotes virtual (switch virtual) apresentado por Luigi Rizzo e Giuseppe Lettieri da Universidade de Pisa, usado para interconectar máquinas virtuais em redes a partir da API Netmap (41), entra na plataforma substituindo o pouco otimizado OpenVSwitch e ao fim é alterado para que forneça acesso

direto aos buffers das portas do seu switch para o netfront, e para que possuía mais portas, para citar as principais modificações(1). Ao fim é renomeado para ClickOS Switch (1).

Figura 8 – Antes e depois das alterações mencionadas



Fonte: Adaptado de (1)

A escolha por Click para integrar a plataforma deve-se a flexibilidade e o reuso de código mencionados nos objetivos dos criadores e abordados na ~~seção~~ Seção §3.1. ClickOS constrói seu sistema operacional a partir do MiniOS: um micro-kernel de sistema operacional minimalista voltado para um processamento quase monoprocessoado, escalável, e de baixa carga de CPU, criado pelos desenvolvedores do projeto Xen. A adaptação de Click ao micro-kernel do MiniOS, inclusive portando as bibliotecas necessárias sua construção neste sistema, resultou em um kernel enxuto, que dentro de uma micro-VM consome 5MB de memória ao executar, sendo instanciável em 30 ms, provocando um atraso na rede de 45µs, não necessitando de disco e alcançando uma vazão de 10Gbps (1). Este desempenho confirma as características atribuídas a ele no início desta seção.

Para instanciar as VNFs, ClickOS usa o método disponibilizado pelo projeto Xen, através de um arquivo de configuração, onde é possível descrever itens como quantidade de processadores, memória, interfaces virtuais de rede, comportamento de falhas e o nome da função virtual.



Figura 9 – Exemplo de arquivo de configuração de uma VNF

```

1 kernel» » = './build/clickos_x86_64'↵
2 vcpus» » = '1'↵
3 memory» » = '12'↵
4 #juntar as 2 linhas abaixo e retirar as reticências se copiar.↵
5 vif      = ['ip=10.0.0.1,mac=00:19:85:11:00:54,bridge=suaBridge',↵
6 |...|...|...|...'ip=outroIP,mac=outroMAC,bridge=outraBridge', ...]↵
7 name» » = 'vnf'↵
8 on_crash» = 'preserve'↵

```

Fonte: Elaborada pelo autor

Baseando-se na Figura 9, é importante dizer que os dados sobre as redes conectadas devem estar de acordo com os dados atribuídos a VNF no código de Click para não haver enganos e erros. Muito cuidado ao gerar endereços MAC ou IP, pois existem classes e restrições em ambos.

O segundo passo da criação de VNFs em ClickOS passa pelo uso do gerente de pilha Cosmos. Ele recebe o código Click e a identificação da VM criada com o kernel ClickOS para inserir o primeiro no segundo, transformando-o em um middlebox virtual.

As VNFs não possuem linha de comandos para seu gerenciamento próprio, sendo esse um papel da máquina hospedeira. No momento da implantação das funções a partir do hospedeiro, não há passagem de parâmetros em tempo de inicialização. Apenas algumas operações de leitura e escrita em tempo de execução são permitidas, através dos manipuladores de elementos descritos na Seção §3.1, que podem ser acionados através do hospedeiro com o Cosmos, mas muito longe de ser um suporte consistente a configurações em tempo de execução como uma adição de regras em um firewall ou uma remoção de servidor no balanceador de carga. Para contornar essa limitação, scripts de marcação de texto podem ser utilizados a parte para gerar os arquivos na linguagem de Click e recarregar a função de rede, mesmo que signifique uma queda repentina do serviço associado.

Apesar de não possuir linha de comandos, as máquinas virtuais com o núcleo do ClickOS exibem um registro de alertas e erros exibido pelo comando `xl console`, que pode reportar erros de codificação de todos os níveis, erros de inicialização, ~~máscaras configurações e elementos de análise de erros para informar estados através deste.~~

~~P4, NetBricks(42), Elastic Edge(43), OpenNetVM (44)
(45).~~

Por fim, ClickOS pode trabalhar em conjunto com soluções NFV através de uma integração com o VIM da solução. São exemplos já implementados o OpenStack e o Nomad(5).

Comandos para criação e execução, monitoramento e análise de erros estão no

apêndice .

4 PROJETO E IMPLEMENTAÇÃO

A partir dos conceitos apresentados, este capítulo propõe, projeta e implementa duas funções de rede virtuais, motivado pelos benefícios da camada de virtualização no âmbito de servidores e redes com seus dispositivos. As características atribuídas ao ClickOS na Seção §3.2, incluindo-se Click na Seção §3.1, pavimentam a intenção de demonstrar o desenvolvimento de VNFs de maneira mais clara, a partir de uma visão macro em relação àquela vista em linguagens de programação, sem se desenhar apenas como uma plataforma de ensino, já que em determinado momento é capaz de entregar desempenho satisfatório, inclusive sendo habilitável ao NFV.

As funções escolhidas são um firewall (Seção §4.2) e um balanceador de carga (Seção §4.3) em uma rede com Ethernet, TCP/IP. A escolha se deve pela boa legibilidade e assimilação de suas funções mais importantes, já que são facilmente encontrados nas infraestruturas da Tecnologia da Informação. Nesta proposta, o firewall precede o balanceador de carga sem nós intermediários. ~~No Capítulo 5 tais funções são testados justamente em um cenário de serviço de hospedagem web.~~

Espera-se que a curva de aprendizado da ferramenta seja melhor a partir dessas funções ~~do que com funções simples porém desintegradas. Inclusive, foge o escopo deste trabalho demonstrar VNFs complexas como as aplicadas hoje em middleboxes físicos, que signifiquem recursos superestimados e subutilizados na maioria dos casos, o que justamente é um dos pontos de combate do NFV e suas funções virtuais.~~ Um resumo do proposto pode ser visto na Seção §4.1.

Tanto o firewall quanto o balanceador de carga possuem elementos mínimos para seu funcionamento, como é o caso da estrutura de camada física e de enlace, composta pelos elementos FromDevice, ToDevice, Classifier, ARPQuerier e ARPResponder (ver um uso delas na Figura 5). Já outros como IPFilter, SetIPAddress, RoundRobinIPMapper, SourceIPHashMapper, IPRewriter SetTCPChecksum estão associados a tarefa de cada VNF de forma específica. Segue um detalhamento deles com base na documentação (29):

- IPFilter: possui uma porta de entrada por onde passam pacotes IP, e duas de saída por onde saem fluxos permitidos e fluxos bloqueados. Ao invés do último caso, há a possibilidade de ignorá-los, o que faz com que tal fluxo não saia pela respectiva porta. O que define o caminho dos pacotes são as regras adicionadas nos parâmetros do elemento, com base na ação descrita. A comparação pacote-regra é feita de forma ordenada, logo as regras mais específicas devem estar antes das mais genéricas. Em um firewall de produção, é comum que o padrão seja bloquear por padrão e aceitar as

exceções, sendo que para reproduzir este comportamento, basta adicionar as regras de permissão antes da regra **deny all ou drop all**, ao fim;

- **SetIPAddress**: possui uma porta de entrada e outra de saída. É o elemento utilizado para configurar o endereço do gateway da rede, usado para os pacotes cujo destino está a mais de um salto a partir da rede atual. Quando configurado, realiza uma anotação no pacote recebido que é utilizada pelos elementos ARP ao encapsular o pacote IP. A anotação tem precedência sobre o endereço IP de destino presente no pacote no momento de busca pelo MAC associado na tabela ARP. Dessa forma, um pacote consegue saltar para as redes do **gateway** em busca do seu destino, que por sua vez pode repetir a operação sucessivamente até que encontre-o;
- **RoundRobinIPMapper**: não possui portas. Serve como um anexo ao IPRewriter para determinar qual regra deve ser utilizada no momento da reescritção de endereço IP e portas a ser feita por este. A determinação acontece seguindo um padrão de intervalo de tempo sem prioridade como é o escalonamento Round Robin. Ele permite que o IPRewriter tente distribuir de maneira uniforme os pacotes que recebe;
- **SourceIPHashMapper**: na mesma linha do anterior, no entanto a regra utilizada nesse caso depende de uma função hash para fazer a escolha dentre aquelas configuradas em seus parâmetros, fixando-a através do endereço IP de origem para que todos os pacotes provenientes do endereço tenham a mesma ação, independentemente de outras variáveis como endereço IP de destino ou portas;
- **IPRewriter**: Um dos elementos mais complexos estudados por esse trabalho, pode possuir quantas portas de entrada e saída for necessário. De forma geral, este elemento faz o papel de reescrever os cabeçalhos com relação aos endereços IP e portas de transporte, com base nas ações configuradas em uma tabela pelas regras inseridas nos parâmetros do elemento. Estas, por sua vez, são combinadas com portas, associando uma porta de entrada X ao índice **X** na lista de parâmetros configurados no elemento. Caso os dados do pacote não casem com uma ação previamente estabelecida por outra regra, o elemento a cria e a aplica com base na regra da associação, atribuindo a porta de saída Y descrita na regra para o pacote alterado sair do elemento. Inclusive, o NAT costuma receber pacotes-resposta para aqueles que sofreram **reescritção** anteriormente, e por isso o IPRewriter prontamente adiciona uma ação de retorno para que ele também reescreva o pacote-resposta, e este volte ao requerente através de outra porta de saída Z do elemento, configurada também nos parâmetros. Em suma, no IPRewriter, o desenvolvedor é responsável por gerir o fluxo e dizer onde eles devem entrar e sair, trazendo flexibilidade para a solução. Além das regras de **reescritção**, há outras regras que permitem ações padrão caso não haja casamento padrão-pacote naquela porta, como ignorar o pacote e simplesmente

passar o pacote adiante em uma porta de saída desejada, inclusive nas portas já existentes. Há também a possibilidade de usar anexos como os disponibilizados pelos dois elementos acima descritos, que armazenam suas regras com o mesmo padrão utilizado aqui. Para um exemplo concreto, veja a implementação do balanceador de carga na Seção §4.3.

- SetTCPChecksum: com uma porta de entrada e outra de saída, este elemento é responsável por corrigir a soma de verificação do cabeçalho TCP, e garantir que os pacotes sejam considerados válidos pelas camadas de transporte que os recebem. Muito associado as saídas de um IPRewriter para validar os pacotes alterados por ele.

4.1 Um dia dentro das funções propostas

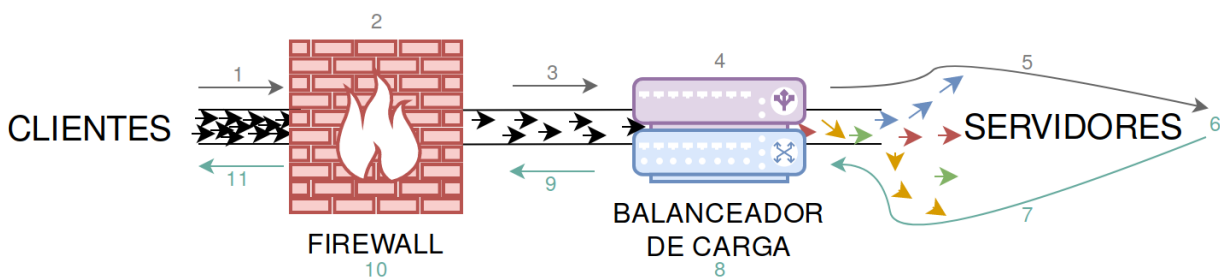
Um pacote criado por um cliente irá solicitar alguma ação para o servidor. Dada a abstração para que o dado chegue até o primeiro ponto de parada (firewall), ele chega pela rede, entra na interface virtual do switch virtual, que encaminha o mesmo para a interface contida na VNF. Nesse ponto o pacote é encaminhado para o buffer de entrada presente no kernel do ClickOS, que por sua vez permite que o Click receba os pacotes na interface de rede 0 através do elemento FromDevice. O elemento citado transmite o dado para o classificador (Classifier), que diante da análise o passa pela saída 2 ao verificar que se trata de um pacote Ethernet com cabeçalho IP na camada mais interna. Logo a frente, seu cabeçalho Ethernet é removido (Strip), seu cabeçalho IP é checado (CheckIPHeader), para que posteriormente possa ser analisado pelo elemento base do Firewall (IPFilter). Ao ser autorizado, o pacote inicia o caminho de saída para o próximo nó do cenário entrando no elemento responsável por consultar o MAC do próximo e encapsulá-lo (ARPQuerier), logo após chegando a fila usada para armazenar os pacotes enquanto o dispositivo de saída não está pronto (Queue), que quando o buffer de saída estiver apto, é removido de lá pelo elemento ToDevice enviando-o pela interface de rede 1, notabilizando uma possível rede diferente daquela de entrada. Então o pacote vai ao buffer de saída da VNF e de lá para a interface virtual e para o switch virtual que encaminha para a porta do próximo nó. Passagens: 1, 2 e parte do 3 na Figura 10.

O processo se repete para a entrada do pacote no balanceador de carga até a etapa de checagem do endereço IP, onde ele é entregue ao elemento responsável pelo processo de troca de endereços de destino no DNAT (IPRewriter). Aqui a troca é feita conforme o elemento configurado para balancear a carga (SourceIPHashMapper ou RoundRobinIPMapper). O pacote com IP alterado para o servidor escolhido entra pela porta 0 de entrada e pela configuração sai pela porta 0 de saída, onde a correção da soma de verificação do protocolo TCP o aguarda e logo após o pacote sai em direção ao seu encapsulamento do

cabeçalho Ethernet, repetindo o todo o processo de saída da VNF pela interface de rede 1, e o processo para sua entrega ao próximo nó. Passagens: parte de 3, 4 e parte do 5 na Figura 10.

O próximo nó é o servidor, que assim como os outros recebe em sua interface de rede o pacote da interface virtual, **adicionando** ao buffer de entrada que em um momento analisa o protocolo de transporte e entrega-o de acordo com o serviço conectado a porta de destino definida. **O serviço destinado analisa o pacote recebido**, e supondo que **seja respondível**, o faz, o que ao fim gera um pacote resposta na rede em sentido contrário, que inicialmente sai do buffer de saída da interface de rede, passa pela interface virtual e chega ao switch virtual, que reencaminha o pacote para a porta onde está a interface com o balanceador de carga. Passagens: parte do 5, 6 e parte do 7 na Figura 10.

Figura 10 – Comportamento esperado das VNFs propostas



Fonte: Elaborada pelo autor

No sentido contrário, o pacote resposta faz um percurso parecido que no sentido normal. As principais diferenças estão na chegada a partir da interface de rede 1 e saída para o firewall na interface de saída 0, e no elemento de troca de endereços. Este último recebe o dado do elemento de checagem de endereços pela porta 1 ao invés da porta 0, para que possa descartar o pacote caso não haja regra compatível. Pela regra gravada na fase de envio, o pacote resposta ao pacote original deve sair pela porta 1 do elemento após corretamente convertido para os IPs externos. Assim que sai pela porta 1, o pacote tem seu reprocessamento da soma de verificação como anteriormente, e depois recebe a anotação (**SetIPAddress**) indicando que o pacote precisa passar por um gateway no próximo nó para rumar ao seu destino final. Todo o restante do processo no balanceador de carga é comum aos passos já mencionados. Passagens: parte do 7, 8 e parte de 9 na Figura 10.

Chegando ao firewall, o pacote resposta passa pelo mesmo procedimento mencionado no início, trocando as interfaces de rede em que entram e saem os pacotes: agora na porta 1 e porta 0, respectivamente. A partir deste ponto, retoma-se a abstração com relação a chegada do pacote resposta ao cliente. Passagens: parte do 9, 10, e 11.

4.2 Firewall

A proposta aqui passa por realizar os bloqueios no nível da camada de roteamento e transporte. O elemento de Click responsável deve avaliar a quádrupla de endereços e portas de origem e destino para decidir com base nas regras implantadas a permissão ou não para seguir fluxo, em modo sem estado (**stateless**). Dada a lista de pacotes no fluxo exibida na Tabela 1 e a lista de regras na Tabela 2, o resultado esperado segue na Tabela 3:

Tabela 1 – Exemplo de fluxo de pacotes que vai ao firewall

| #Pct | IP origem | Porta envio | IP destino | Porta alvo |
|------|---------------|-------------|---------------|------------|
| 1 | 198.51.100.55 | - | 203.0.113.71 | 443 |
| 2 | 203.0.113.71 | 443 | 198.51.100.55 | 45045 |
| 3 | 198.51.100.55 | 60365 | 203.0.113.4 | 21 |
| 4 | 192.0.2.237 | 53385 | 203.0.113.71 | 22222 |
| 5 | 203.0.113.71 | 22222 | 192.0.2.237 | 53385 |

Tabela 2 – Exemplo de regras no firewall

| #Regra | IP origem | Porta envio | IP destino | Porta alvo | Ação |
|--------|----------------|-------------|----------------|------------|----------|
| 1 | - | - | - | 22222 | Aceitar |
| 2 | 203.0.113.0/24 | 1024-65536 | 203.0.113.0/24 | 21 | Aceitar |
| 3 | 203.0.113.0/24 | 22222 | 203.0.113.0/24 | 1024-65536 | Aceitar |
| 4 | - | - | 203.0.113.71 | 443 | Aceitar |
| 5 | 203.0.113.71 | 443 | - | - | Aceitar |
| 6 | - | - | - | - | Rejeitar |

Tabela 3 – Resultado do processamento no firewall

| #Pct | IP origem | Porta envio | IP destino | Porta alvo | Situação |
|------|---------------|-------------|---------------|------------|----------------|
| 1 | 198.51.100.55 | - | 203.0.113.71 | 443 | ACEITO em 4 |
| 2 | 203.0.113.71 | 443 | 198.51.100.55 | 45045 | ACEITO em 5 |
| 3 | 198.51.100.55 | 60365 | 203.0.113.4 | 21 | REJEITADO em 6 |
| 4 | 192.0.2.237 | 53385 | 203.0.113.71 | 22222 | ACEITO em 1 |
| 5 | 203.0.113.71 | 22222 | 192.0.2.237 | 53385 | REJEITADO em 6 |

Os pacotes aceitos são transmitidos ao próximo nó da rede: o balanceador de carga.

Para que o firewall funcione de acordo com a Seção §4.1, são necessárias duas interfaces de rede para diferenciar os fluxos ~~do~~ para o mundo externo, e o fluxo ~~do~~ para o balanceador de carga. Essa separação naturalmente exige dois conjuntos mínimos de elementos para tratar os pacotes na camada de enlace, respondendo e realizando requisições ARP, e classificando os pacotes conforme seus cabeçalhos. Exige também que o arquivo

de configuração da máquina virtual a abrigar o firewall ativo as placas de rede e atribua endereços IP, endereços MAC e os switches virtuais nos quais a interface se conecta.

Com relação a memória e processamento, as máquinas virtuais não exigem muitos recursos, a ponto de 12MB de memória e uma CPU virtual fazer a maioria das tarefas, inclusive um firewall, como demonstra (1). Em caso de problemas, ClickOS dá sinais sobre sua parada no registro de erros e alertas, e portanto, uma heurística pode definir melhor quais recursos ajustar para a função sem que haja desperdício. O tamanho das filas de saída de pacotes deve ser estudado.

Uma vez que os conjuntos de elementos são corretamente configurados, é preciso tratar os pacotes de dados. O uso do elemento Strip, passando o número de bytes usados pelo cabeçalho Ethernet para que os remova é o primeiro passo. Verificar a validade do cabeçalho IP extraído, para garantir que o pacote ainda é válido, é o segundo. Isto é providenciado pelo elemento CheckIPHeader, que também pode filtrar pacotes com base em IPs indesejados, principalmente IPs de comunicação abrangente (broadcasts). Depois desse tratamento e checagem, é possível pensar no bloqueio e autorização de pacotes de maneira geral.

IPFilter, a partir de cabeçalhos, realiza filtrações de pacotes usando valores de endereço IP origem e destino, portas de origem e destino, e o tipo de protocolo de transporte, inclusive limitando pelos tipos diferentes de pacotes dentro destes e valores padrão de variáveis. As regras de autorização ou bloqueio devem ser construídas com essas bases, atentando-se para não limitar ou autorizar acesso em excesso. Para conhecimento de como as regras devem ser construídas, uma consulta aos elementos IPFilter e IPClassifier na documentação é essencial (29).

Dois filtros devem ser configurados: o primeiro trabalha no sentido de entrada do fluxo externo, aceitando apenas pacotes cujos cabeçalhos cumpram requisitos de endereço de rede IP de origem, endereço único IP de destino, número de porta de origem dentro de um intervalo e por fim, o número único de porta de destino, limitando não só os recursos e serviços acessados, como para quem estão disponíveis, e ignorando os demais; já o segundo trabalha no sentido de saída do fluxo, portanto, vindo do balanceador. Este aceita que apenas os pacotes-resposta retornem ao mundo externo de onde vieram suas requisições, e visa limitar a comunicação indesejada dos dispositivos balanceados, ignorando os demais pacotes e servindo de camada de proteção contra as invasões, principalmente. Uma análise de tráfego é mandatória para que nenhuma comunicação crítica dos dispositivos seja bloqueada nas redes envolvidas, no caso de firewall e balanceador em sistemas já implantados.

Uma vez que os pacotes de dados fluem pelas duas interfaces de rede, eles são classificados, extraídos de seus cabeçalhos Ethernet, e na sequência checados quanto a suas ~~validades~~. Daí então cada fluxo entra em seu respectivo filtro e, se autorizados,

devem se encaminhados para o reencapsulamento do cabeçalho Ethernet com destino ao balanceador ou ao mundo externo, sendo dispensado na respectiva fila e depois no respectivo elemento de saída do ClickOS correspondente para chegar ao seu destino.

Por fim, pacotes não classificados são descartados.

Da linha 2 à 5 há uma definição de macros no AddressInfo, que nos dois primeiros parâmetros descrevem para net0 e net1 endereços IP próprios, endereços IP das redes e endereços MAC respectivamente. Estes são os endereços atribuídos as interfaces de rede ao inserí-los nos elementos ARP. O terceiro parâmetro guarda apenas o endereço IP do balanceador de carga para onde são enviados os pacotes autorizados.

De 7 à 11 há a definição e declaração de dois classificadores, sendo um para cada rede. Ambos, idênticos, são definidos para filtrar pacotes de consulta do protocolo ARP na porta 0, pacotes de respostas do protocolo ARP na porta 1, pacotes do protocolo IP quaisquer na porta 2, e por fim os demais pacotes na porta 3.

Nas linhas 13 e 14, acontece a conexão entre as entradas de pacotes das interfaces de rede e a porta 0 de seus respectivos classificadores.

Nas linhas 16 e 17 há a declaração e definição das duas filas necessárias para a conversão do fluxo push em pull no momento do despacho dos pacotes já filtrados, sendo uma para cada interface novamente. A porta 0 de saída de cada fila se conecta com a porta 0 de entrada do respectivo elemento de saída de pacotes à respectiva interface de rede.

Logo abaixo, na linha 19 e 20, duas novas declarações e definições são realizadas, com o intuito de inserir o par de endereços IP e MAC de cada interface da VNF (net0 e net1) na respectiva tabela ARP, para que o elemento de consulta desse protocolo realize o correto encapsulamento no nível da camada de enlace, principalmente o que diz respeito ao MAC de origem. Após isso, a porta de saída 0 de tais elementos é conectada a porta de entrada da respectiva fila.

Continuando nas linhas 22 e 23, as portas de saída 1 dos classificadores contém as respostas de consultas ARP, e são conectadas as portas de entrada 1 dos respectivos elementos de consulta para que as receba corretamente e possa encapsular pacotes IP com o correto endereço MAC de destino consultado.

Nas linhas 25 e 26, as portas de saída 0 de ambos classificadores se conectam as respectivas portas de entradas 0 do elemento responsável por responder as consultas ARP de outros dispositivos com o correto endereço MAC para o endereço seu IP consultado. Tal elemento é responsável por atribuir o endereço IP daquela interface de rede na VNF, podendo responder por mais de um deles caso seja desejado. Perceba que os endereços configurados aqui são os mesmos que foram configurados nas linhas 19 e 20, para que haja não só o recebimento dos pacotes mas também a correta indicação ao par da comunicação de para quem enviar os seus.

Nas linhas 28 e 29, ocorre uma declaração e definição do elemento de filtragem que está relacionado a entrada do fluxo externo. Ele possui apenas duas regras definidas em seus parâmetros: uma para permitir pacotes com endereço IP de origem local à interface

de rede 0, que tenham como endereço IP de destino o balanceador de carga, cujo número da porta de origem esteja entre 32768 e 61000, e a porta de destino seja a porta 80; e outra para ignorar todos os demais pacotes que cheguem.

Com a mesma intenção, nas linhas 31 e 32 há uma declaração e definição semelhante a anterior, mudando apenas o foco de seu uso: a intenção agora é filtrar os pacotes que chegam do balanceador e que estão de saída para redes externas. Para isso, os endereços e números de portas que eram de origem se tornam de destino e vice-versa.

Entre as linhas 34 e 37, a porta de saída 2 do classificador da rede 0 entrega os pacotes para a extração do pacote IP no elemento, que por sua vez despacha os pacotes IP para o elemento de checagem do cabeçalho IP, que por sua vez passa os pacotes checados para o filtro de entrada, que realizando sua tarefa, encaminha-os para o encapsulamento da camada de enlace na rede 1, fluindo em direção a respectiva interface de rede, onde está o balanceador de carga. Todas as portas não mencionadas, sejam de entrada ou saída, são de índice 0.

De maneira semelhante, entre as linhas 39 e 42 o processo se repete, e os pacotes-resposta vindos do balanceador pela interface de rede 1, que são autorizados, são entregues para encapsulamento na rede 0, fluindo em direção a sua respectiva interface.

Por último, nas linhas 44 e 45, os pacotes que saem pela porta 3 do classificador entram em um elemento que determina o descarte deles.

4.3 Balanceador de Carga

A proposta para esta VNF é dividir um determinado fluxo entre servidores de uma função semelhante, diminuindo a possibilidade de um servidor específico sobrecarregar ou ter vida útil reduzida, conferindo maior disponibilidade e durabilidade dos equipamentos. Podem haver duas versões, entre o método conhecido como Round Robin e o método por endereço IP de origem. O primeiro divide os pacotes para os servidores com base em um tempo de concessão fixo e o segundo envia todos os pacotes com um mesmo endereço IP de origem para o mesmo servidor destino, fazendo a seleção com base em **uma hash**. **Ele** não avalia a carga atual e a situação atual dos servidores para tomar as decisões, recebendo como entrada apenas o número de servidores disponibilizados.

Dado 1500 pacotes emitido pelo cliente, **espera-se que** os servidores recebam os pacotes aproximadamente da seguinte forma com o método Round Robin:

Tabela 4 – Divisão do fluxo de pacotes vindos do cliente com Round Robin

| Dispositivo | Pacotes transmitidos | Pacotes recebidos |
|-------------|----------------------|-------------------|
| Cliente | 1500 | - |
| Servidor 1 | - | 1500/n |
| Servidor 2 | - | 1500/n |
| ⋮ | ⋮ | ⋮ |
| Servidor n | - | 1500/n |

Já no método por endereço IP de origem, com três diferentes clientes e 1500 pacotes cada, espera-se que:

Tabela 5 – Divisão do fluxo de pacotes vindos de 3 clientes com endereço IP de origem

| Dispositivo | Pacotes transmitidos | Pacotes recebidos |
|-------------|----------------------|-------------------|
| Cliente 1 | 1500 | - |
| Cliente 2 | 1500 | - |
| Cliente 3 | 1500 | - |
| Cliente 1 | 1500 | - |
| Cliente 1 | 1500 | - |
| Servidor 1 | - | 1500 |
| Servidor 2 | - | 1500 |
| Servidor 3 | - | 4500 |

Para que o balanceador de carga funcione de acordo com a Seção §4.1 e a Tabela 4, são necessárias duas interfaces de rede para diferenciar os fluxos originados e destinados ao mundo externo, dos fluxos originados e destinados aos servidores. Assim como no firewall, esta separação exige dois conjuntos mínimos de elementos na camada de enlace para tratar a entrada e saída na rede, gerenciar os pacotes de controle do protocolo ARP e separar os pacotes de acordo com o protocolo. A semelhança segue com os requisitos de memória e processamento, a correta configuração dos elementos mínimos e os elementos Strip de desencapsulamento e CheckIPHeader de checagem do protocolo IP.

Um elemento sem portas de entrada ou saída deve ser configurado para cada método descrito. Aquele que cobre o método Round Robin (RoundRobinIPMapper), deve conter um parâmetro para cada servidor disponível, num formato que indique qual a modificação para cada um dos itens dentre endereço IP de origem, número de porta de origem, endereço IP de destino e número de porta de destino, nessa ordem, sendo que os não indicados permanecem intactos. Após tais indicações, os parâmetros são finalizados com o informe de uma porta de saída do elemento para a regra criada a partir dessas indicações, e o informe de uma porta de saída para a regra de conversão do seu pacote-resposta, já em fluxo de retorno. Já que este elemento não possui portas, elas são alocadas no elemento de NAT no IPRewriter, pois a função deste é fornecer novas funcionalidades ao NAT disponível por Click, tal qual uma variação no momento de criação de suas regras.

Já o elemento para o método de balanceamento por endereço IP de origem (SourceIPHashMapper) precisa de mais parâmetros. Como trabalha com uma estrutura de tabela hash, é preciso indicar uma semente e um tamanho de tabela como primeiro parâmetro, e só após iniciar com os parâmetro de modificação de cabeçalho. Estes, por sua vez, possuem apenas uma modificação inserida ao final de cada parâmetro: um identificador para comparações na tabela. Assim como o elemento anterior, ele não possui portas, servindo como uma nova funcionalidade ao NAT no IPRewriter.

O elemento de NAT (IPRewriter) dessa proposta é simples e passa como parâmetro o conjunto de indicações dos elementos acima, para que sejam usados no caso dos pacotes de entrada de sua respectiva porta não combinarem com as regras existentes; e também uma outra determinação para ignorar os pacotes que entram por um outra porta na mesma situação.

Uma vez que os pacotes de dados fluem pelas duas interfaces de rede, eles são classificados, extraídos de seus cabeçalhos Ethernet, e na sequência checados quanto a suas validades. Dai então o fluxo vindo do mundo externo entra por sua porta 0 e o fluxo vindo dos servidores pela sua porta 1. A saída do primeiro fluxo, com as modificações deve acontecer pela porta 0, e a saída do outro fluxo pela porta 1, permitindo então que a soma de verificação do protocolo TCP seja redefinida separadamente em ambos, através de elementos instanciados.

Após esse processo, os pacotes são encaminhados para o reencapsulamento com destino ao servidor definido pelo NAT e ao cliente, respectivamente. Uma observação importante é que no último caso, o destino só é alcançável com mais de um salto na rede, o que significa que o reencapsulamento da camada de enlace deve ser atribuído ao gateway da rede, no caso o próprio firewall, para que ele possa dar destino ao pacote. Um elemento deve realizar a tarefa de indicá-lo.

Figura 12 – Código para o balanceador de carga por endereço IP de origem no Click

```

1 //      nome      ip      ipnet      mac
2 AddressInfo(net0 192.0.2.61      192.0.2.0/24      00:01:92:00:02:61,
3 .....net1 172.16.40.21      172.16.0.0/12      00:01:72:16:40:21,
4 .....fw0 192.0.2.11,
5 .....ws1 172.16.40.55,
6 .....ws2 172.16.80.147,
7 .....ws3 172.16.37.181
8 );
9 classificador0, classificador1 :: Classifier(12/0806 20/0001, // 0.consultas ARP
10 .....12/0806 20/0002, // 1.respostas ARP
11 .....12/0800, // 2.pacotes IP
12 .....- // 3.outros
13 );
14 FromDevice(0) -> [0]classificador0;
15 FromDevice(1) -> [0]classificador1;
16
17 saida0 :: Queue(1024) -> ToDevice(0);
18 saida1 :: Queue(1024) -> ToDevice(1);
19
20 arp0 :: ARPQuerier(net0) -> saida0;
21 arp1 :: ARPQuerier(net1) -> saida1;
22
23 classificador0[1] -> [1]arp0;
24 classificador1[1] -> [1]arp1;
25
26 classificador0[0] -> ARPResponder(net0) -> saida0;
27 classificador1[0] -> ARPResponder(net1) -> saida1;
28
29 mapeamento :: SourceIPHashMapper(129 0xbadbeef,
30 .....- - ws1 - 0 1 4055,
31 .....- - ws2 - 0 1 80147,
32 .....- - ws3 - 0 1 37181
33 );
34 nat :: IPRewriter(mapeamento,
35 .....drop
36 );
37
38 classificador0[2] -> Strip(14)
39 .....-> CheckIPHeader()
40 .....-> [0]nat;
41
42 classificador1[2] -> Strip(14)
43 .....-> CheckIPHeader()
44 .....-> [1]nat;
45
46 nat[0] -> SetTCPChecksum()
47 .....-> [0]arp1;
48
49 nat[1] -> SetTCPChecksum()
50 .....-> SetIPAddress(fw0)
51 .....-> [0]arp0;
52
53 classificador0[3] -> Discard;
54 classificador1[3] -> Discard;

```

Fonte: Elaborada pelo autor

As descrições para as ações programadas entre as linhas 2 e 27 são idênticas para aquelas feitas na implementação da Figura 11. O mesmo vale para as linhas 53 e 54.

Nas linhas de 29 à 33 há uma declaração e definição do elemento de balanceamento por endereço IP de origem, com seu tamanho de tabela *hash* e semente como primeiro parâmetro e nos seguintes, as modificações necessárias para cada servidor, a serem mapeadas no próximo elemento como regras de ida e volta. Pode-se ver também os identificadores ao final de cada parâmetro para efeitos de comparação.


No caso do elemento de balanceamento por Round Robin, as linhas descritas acima são substituídas pela declaração e definição do referido elemento, onde os parâmetros se limitam às modificações necessárias para cada servidor disponível, sem os identificadores naturais a tabela *hash*.

Nas linhas 34 à 36 acontece uma declaração e definição do elemento que reescreve os cabeçalhos conforme as regras determinadas pelos elementos de balanceamento. Ele reescreve desta forma apenas os pacotes que entram pela sua porta 0 e não possuem um regra **de reescritção** anteriormente definida. Já os pacotes que entram pela porta 1, são ignorados caso não possuam regra predefinida.

Entre as linhas 38 e 44, dado que as portas não mencionadas de entrada e saída são portas 0, os classificadores das diferentes interfaces de rede enviam os pacotes IP por suas portas de saída 2, onde entram nos elementos de extração de cabeçalho Ethernet, saindo direto para a entrada nos elementos de checagem do cabeçalho IP, e após esse processo, entram cada um em uma porta de entrada diferente do elemento responsável pelo NAT. O primeiro fluxo entra pela porta 0 e se depara com o elemento de balanceamento, que nesse caso sorteia um servidor e **executa a reescritção**, enviando o pacote pela sua porta de saída 0. O segundo fluxo entra pela porta 1 e espera encontrar a regra inversa a alteração efetuada para que possa retornar para aquele dispositivo que o requisitou. A porta de saída nesse caso é a de número 1. Caso não encontrem a regra, os pacotes nessa situação são simplesmente ignorados.

As saídas do NAT entre as linhas 46 e 51 têm como destino o encapsulamento no respectivo elemento de consulta ARP. No caso, o encapsulamento dos pacotes que chegam de fora, é feito pelo elemento ARP associado a rede do balanceador, e vice versa. Porém antes precisam ter suas somas de verificação do protocolo TCP refeitas para evitar falsos erros. O caso dos pacotes-resposta requer um elemento que anota o endereço do *gateway* ao pacote para que o encapsulamento seja feito com ele, mesmo sem alterar o cabeçalho IP. **O *gateway* nesse caso é invocado através do parâmetro.**

5 TESTES

Dadas as implementações de firewall e balanceador de carga ~~com DNAT~~, os testes  demonstram suas validades. Este capítulo discorre sobre os recursos de hardware e software necessários para a execução dos testes (Seção §5.1 e Seção §5.2); encaminha a configuração de um ambiente de testes com simulação em um serviço de hospedagem de páginas web na internet, tomando como base os elementos construídos no Capítulo 4 (Seção §5.3); e propõe os testes, descreve a execução e apresenta os resultados encontrados, pontuando seus detalhes (Seção §5.4).

5.1 Infraestrutura e softwares

A estrutura utilizada para os testes consiste em um servidor de virtualização com VMware, onde uma máquina virtual com o sistema operacional Debian é alocada para todo o experimento. As especificações são:

Tabela 6 – Especificações de hardware para os testes

| Hospedeiro - Dell PowerEdge R820 | Máquina Virtual - dom0 |
|---|--------------------------------------|
| 8 CPUs x Intel Xeon CPU E5-4603 @ 2.00GHz | 4 vCPUs |
| 383.96 GB de RAM DDR3 | 6GB de RAM |
| 2 HDs x 2TB | 20GB de HD |
| VMware ESXi 6.5.0 | Debian Server 8.7 - kernel 3.16.51-3 |

Como dito na Seção §3.2, ClickOS introduz uma nova camada de virtualização ao sistema que o hospeda através do Xen, e isso ocorre dentro de outra camada de virtualização provida pelo VMware, o que não parece ideal. Como o foco deste trabalho é a implementação, o excesso de camadas de virtualização não se mostrou um problema da maior relevância. Reserva-se para trabalhos futuros a otimização deste ambiente de testes com foco em desempenho.

Ao que se refere aos softwares instalados na máquina virtual criada pelo VMware, uma lista não extensiva deles é:

Tabela 7 – Softwares para a VM hospedeira


| Tipo | Software | Versão |
|---------------------|-------------|---------|
| Provedor de VNFs | ClickOS | 0.2 |
| Hypervisor | Xen Project | 4.4.1-9 |
| Manipulador de VNFs | Cosmos | 0.1 |

| | | |
|--------------------------|------------------|-----------|
| Switch virtual | Linux Bridge | 3.16.51-3 |
| Gerente do Switch | Bridge utilities | 1.5-9 |
| Monitor de rede | BWM-NG | 0.6-3.1 |
| Analizador de tráfego | Tcpdump | 4.9.2-1 |
| Sincronizador de relógio | Ntpdate | 4.2.6 |

Importante mencionar que a criação do ClickOSSwitch pelos autores do ClickOS não exclui a possibilidade de uso de outros comutadores como o próprio OpenVSwitch e o Linux Bridge. A consequência do uso destes outros é a perda de desempenho, mas para testes e aplicações que não exijam desempenho prontamente, a plataforma facilita sua aplicação nesses ambientes. **Este trecho é importante no contexto deste trabalho, uma vez que o foco é a implementação e o Linux Bridge entrega o mínimo para a troca de fluxos.**

Já os **softwares** instalados nas máquinas virtualizadas pelo Xen que fazem papel de clientes (cli[1-3]) do serviço de hospedagem web são:

Tabela 8 – Softwares para as VMs clientes

| Tipo | Software | Versão |
|---|---------------|------------------------|
| Sistema Operacional | Debian Server | 8.7 - kernel 3.16.51-3 |
| Gerador de tráfego HTTP | Siege | 3.0.8-1 |
| Navegador  b | Lynx | 2.8.9 |
| Sincronizador de relógio | Ntpdate | 4.2.6 |
| Gerente de interfaces de rede | ifconfig | 1.60-26 |
| Gerente de rotas de rede | route | 1.60-26 |

Por fim, os softwares instalados nas máquinas virtualizadas pelo Xen que fazem o papel de servidores web (ws[1-3]) do serviço em questão são:

Tabela 9 – Softwares para as VMs servidoras

| Tipo | Software | Versão |
|-------------------------------|---------------|------------------------|
| Sistema Operacional | Debian Server | 8.7 - kernel 3.16.51-3 |
| Serviço de hospedagem web | Apache | Worker - 2.4.10-10 |
| Gerente de interfaces de rede | ifconfig | 1.60-26 |
| Gerente de rotas de rede | route | 1.60-26 |

5.2 Alocação de recursos

Mesmo com as modificações feitas pelo ClickOS, o Xen ainda sim é capaz de entregar virtualização de máquinas com sistemas operacionais gerais como os GNU/Linux. Isso possibilita que o ambiente de teste seja todo desenvolvido em um mesmo local, facili-

tando o processo de implantação dos testes. Além das máquinas virtualizadas citadas na Seção §5.1, aqui o Xen também virtualiza duas micro-VMs para o firewall e o balanceador de carga respectivamente. Seguem os recursos alocados para cada uma delas através de virtualização:

Tabela 10 – Recursos do hospedeiro (dom0)

| Recurso | Alocado |
|---------------|-------------------------------|
| Processador | 1 CPU dedicado |
| Memória | 1GB dedicado |
| Disco | 20GB |
| Swap em disco | 1280MB alocados dinamicamente |

Esta alocação dedicada retira tais recursos do total disponível para a virtualização, assim restando 5GB de memória e três CPUs das fornecidas pelo VMware para toda a estrutura do teste. Os recursos restantes são todos compartilhados com os dispositivos abaixo:

Tabela 11 – Recursos das três VMs clientes

| Recurso | Alocado |
|-------------------|------------------------------|
| Processador | 1 CPU compartilhado |
| Memória | 256MB compartilhado |
| Disco | 6GB alocados dinamicamente |
| Swap em disco | 512MB alocados dinamicamente |
| Interface de Rede | 2x em modo bridge |

Tabela 12 – Recursos das três VMs servidoras

| Recurso | Alocado |
|-------------------|------------------------------|
| Processador | 1 CPU compartilhado |
| Memória | 768MB compartilhado |
| Disco | 6GB alocados dinamicamente |
| Swap em disco | 768MB alocados dinamicamente |
| Interface de Rede | 2x em modo bridge |

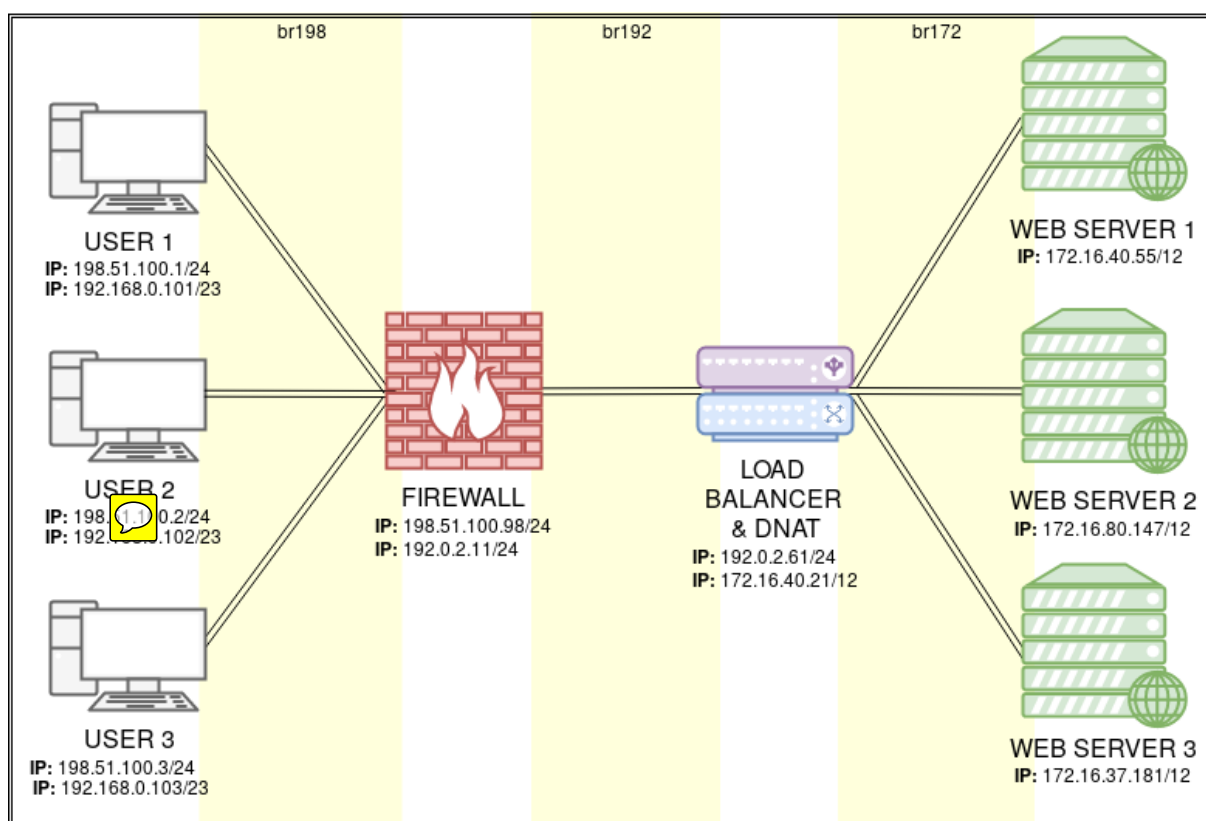
Tabela 13 – Recursos das duas VMs para as VNFs

| Recurso | Alocado |
|-------------------|---------------------|
| Processador | 1 CPU compartilhado |
| Memória | 24MB compartilhado |
| Interface de Rede | 2x em modo bridge |

5.3 Protótipo

Os recursos mencionados na Seção §5.2 foram alocados pensando em um ambiente de validação baseado em hospedagem web. Nesse ambiente é provável encontrar equipamentos que desempenham as funções de rede implementadas no capítulo anterior, sendo as empresas desse setor possíveis interessadas neste protótipo de solução virtualizada. No entanto, o ambiente proposto **abstrai** certa complexidade quando conecta o cliente diretamente ao firewall, sem saltos e sem que ao invés disso haja um dispositivo que redirecione as requisições ao mesmo. Mas é importante notar que esta adaptação não prejudica os testes que são propostos na Seção §5.4. O protótipo planejado **segue abaixo e** esta presente no :

Figura 13 – Protótipo para execução dos testes



Fonte: Elaborada pelo autor

Três pontes (bridges) ethernet foram criadas no switch virtual Linux Bridge para prover as conexões entre as máquinas virtuais, e uma entre estas e as demais redes físicas para fins de administração dos testes.

Após criadas e configuradas com seus softwares, as máquinas usuárias têm suas interfaces de rede configuradas com um IP e máscara de sub-rede usados para a comunicação entre VMs (198.51.100.0/24) e outro para que o controle das operações seja realizado de outra máquina (192.168.0.0/23) via SSH. Tal configuração é feita na linha de comando

das VMs clientes para cada interface de rede usando o `ifconfig`, e também no arquivo de configuração das VMs no hospedeiro. Neste arquivo também se determina a qual rede do switch virtual as interfaces de rede virtuais das VMs estão conectadas. No caso do protótipo, trata-se das redes dentro das áreas amarelas na Figura 13 e a rede de controle que não é exibida na figura. Por fim, as interfaces usadas para o teste em si recebem através do route a informação de um gateway. Isto é essencial para que os pacotes cheguem até o serviço web, tendo o firewall como intermediário.

O firewall e o balanceador de carga recebem boa parte das suas configurações na fase de implementação, de modo estático, tais como IPs e máscaras de sub-rede como os vistos na Figura 13. Para uma melhor visualização da configuração de rede dessas VNFs, o ideal é consultar a e a . No mais, os arquivos de configuração das VMs no hospedeiro recebem as informações sobre as redes do switch virtual usadas e as suas configurações para cada uma delas.

Idênticos aos clientes com relação a configuração de rede, os servidores web abdicam de interfaces de rede virtuais para seu gerenciamento, já que se configurados uma única vez se tornam dispositivos passivos nesse quesito, com modificações esporádicas e suportáveis através do hospedeiro. Antes de inseridos a rede local alimentada pelo balanceador de carga (172.16.0.0/12), seus softwares são configurados e recebem a página web que será usada para os testes, que possui cerca de 200 bytes. Trata-se de uma página HTML minimalista contendo apenas a identificação exclusiva de cada um dos três servidores. Nenhum ajuste extra foi feito no Apache Worker além do bloqueio de escrita de registros apenas para os testes da seção 5.4.4, já que as operações estavam afetando sensivelmente a análise dos resultados.

Alguns detalhes comuns a alguns ou todos os dispositivos:

- A unidade de transporte máxima (MTU) é padronizada em 1500 bytes para todos do protótipo através do uso do *ifconfig* nas interfaces de rede virtuais e nas interfaces do switch virtual;
- Os relógios dos clientes, servidores web, e hospedeiro foram sincronizados antes dos testes para facilitar a análise dos resultados, usando o *Ntpdate* com o servidor b.ntp.br do NIC.BR;

5.4 Proposta, execução e resultados

Nesta seção, dois testes principais são descritos dentro do protótipo descrito na Seção §5.3. O primeiro testa a função do firewall em bloquear o acesso a determinadas portas, como por exemplo, a capacidade da função de rede programada de exercer o bloqueio ao acesso por máquinas indesejáveis. De forma idêntica, o segundo testa a função

do balanceador de carga em dividir o fluxo de entrada entre vários servidores com base no endereço IP e na n^o da porta de origem do pacote.

5.4.1 Firewall

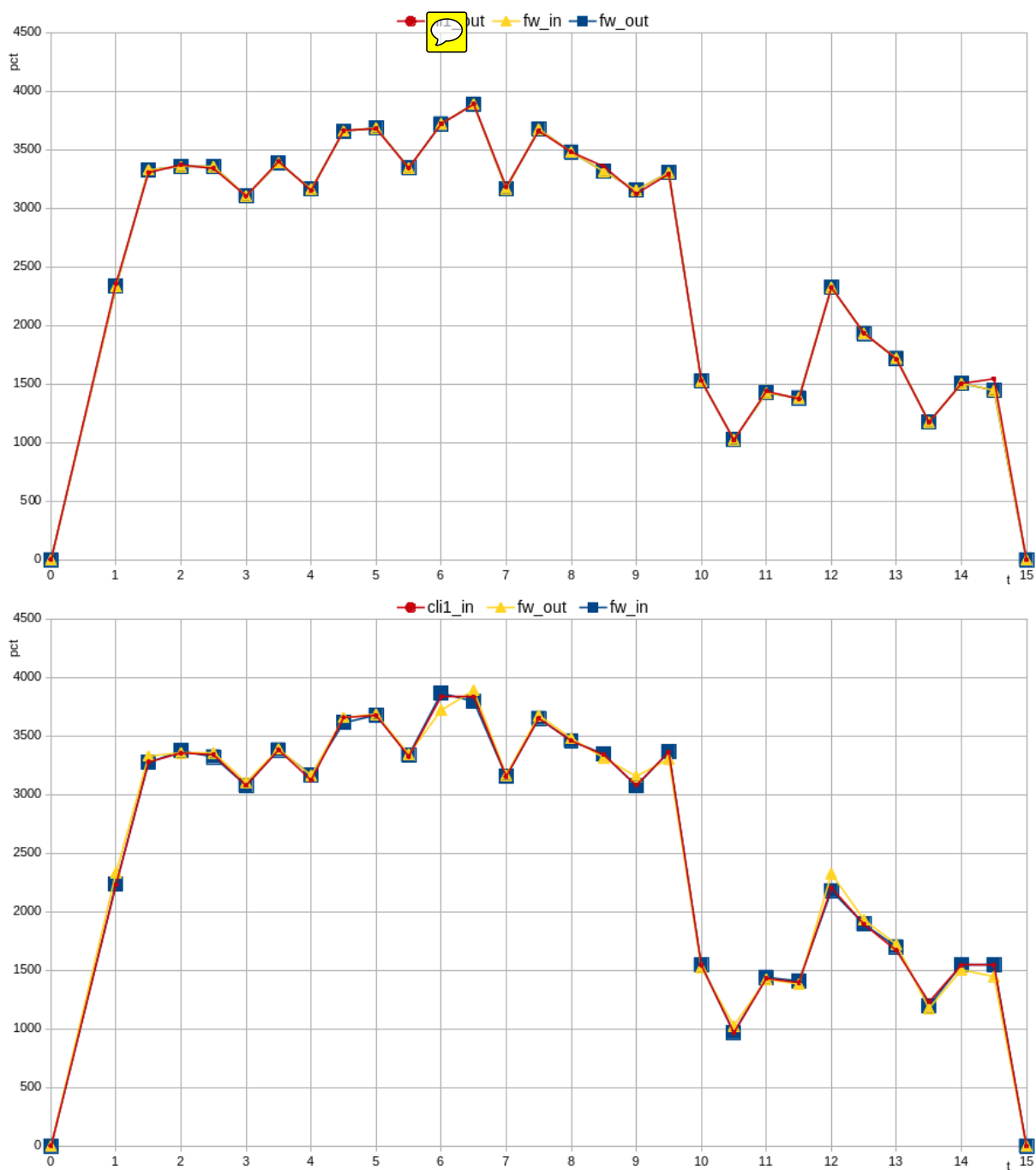
Usando o **Siege** a partir de um cliente qualquer da rede, emite-se tráfego tendo como destino a borda do serviço de hospedagem como na figura Figura 13. O tráfego HTTP gerado passa obrigatoriamente pelo firewall, permitindo uma análise detalhada do fluxo, realizando concessões e bloqueios conforme a proposta.

As decisões tomadas pela função afetam a execução do Siege. Para monitorá-las, aplica-se o **BWM-NG** nas interfaces virtuais relacionadas ao firewall e ao cliente em questão, a procura de padrões que demonstrem o comportamento do fluxo. Não trafegam quaisquer outras conexões pelas interfaces testadas.

Para a execução, executa-se os softwares em paralelo e sincronizadamente, com o Siege no cliente e o BWM-NG no hospedeiro da plataforma ClickOS. O primeiro é configurado para uma execução em no máximo quinze segundos com cem processos concorrentes emitindo requisições para `http://192.0.2.61:X/index2.html`, onde X é o número da porta a ser acessada. Já o segundo guarda uma leitura do fluxo dos pacotes a cada meio segundo nas interfaces selecionadas, de acordo com padrão CSV.

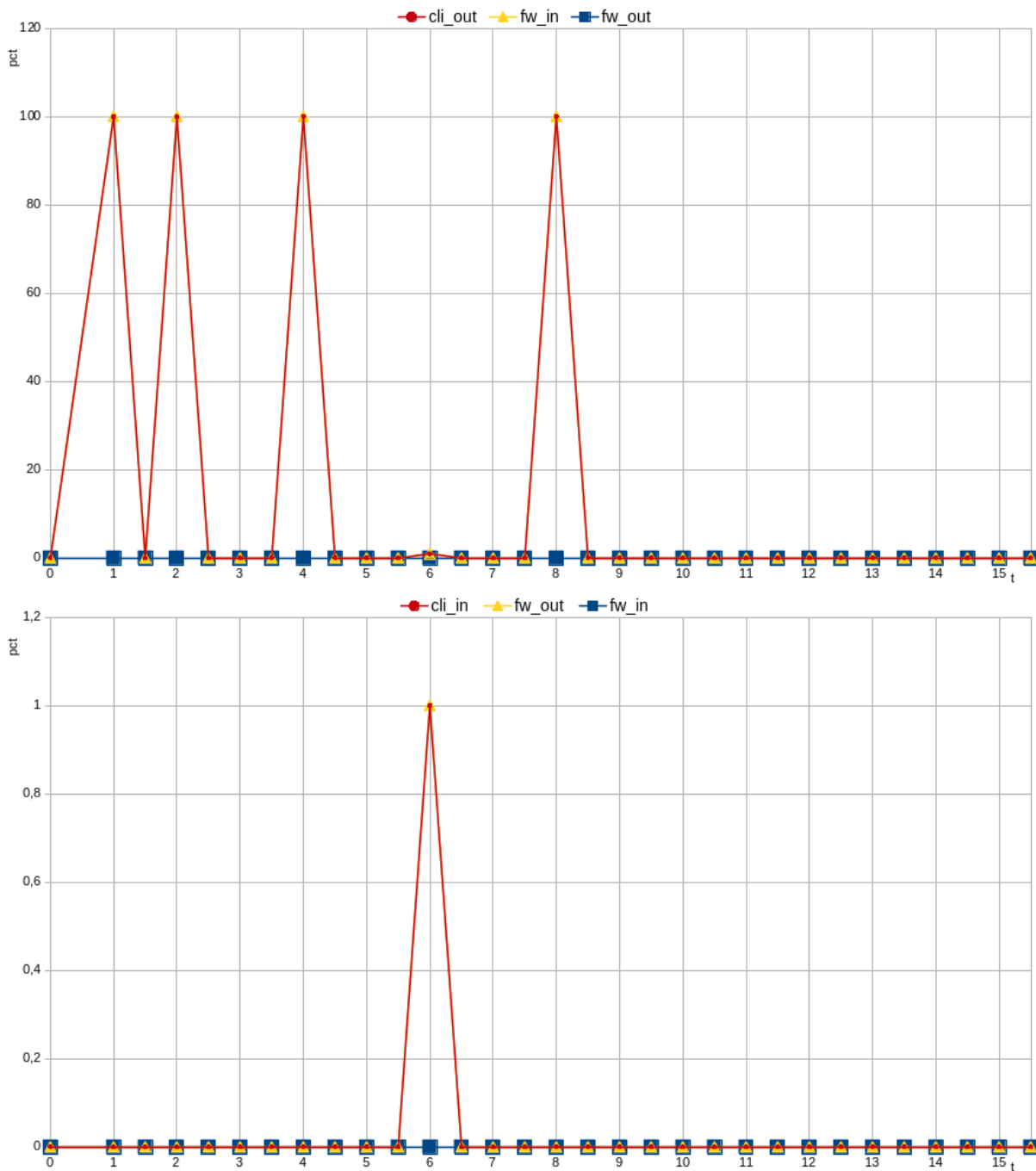
Para auxílio na geração dos gráficos e análise dos resultados, os arquivos resultantes de ambos comandos **armazenam os segundos** do relógio do sistema.

Figura 14 – Fluxo de pacotes TCP 80 com foco no firewall



Fonte: Elaborada pelo autor

Figura 15 – Fluxo de pacotes TCP 8080 com foco no firewall



Fonte: elaborada pelo autor

Para tornar os gráficos mais limpos, foram excluídas as representações dos clientes não utilizados no momento da execução.

Como se pode observar na parte superior da Figura 14, o cliente configurado para enviar o fluxo de requisições através da porta TCP 80 possui um padrão condizente com o esperado em uma conexão normal, respeitando o controle de fluxo nativo do protocolo de transporte em questão, e sem perdas significantes. O tráfego gerado por ele é notado nas interfaces de entrada e de saída do firewall, demonstrando aceitação deste pelas regras.

No outro sentido do tráfego e na parte inferior da figura, as respostas mantêm proporções semelhantes se comparadas ao gráfico de envio, salvo variações mínimas nas quantidades a serem interpretadas como naturais ao procedimento de medição, e demonstrando inexistência de condições adversas.

Já o cliente configurado para enviar requisições HTTP pela porta TCP 8080, na parte superior da Figura 15, demonstra problemas para utilizar o protocolo para estabelecer transporte para o servidor web, enviando apenas cem pacotes de sincronia em intervalos de tempo binários, e que não aparecem na interface virtual de saída do firewall, sendo então bloqueados pela função de rede virtual. Este número não é em vão, já que são cem os processos concorrentes que requisitam conexões com o serviço. Se o fluxo de saída não chega ao serviço, logo é improvável que seja haja resposta em forma de tráfego de entrada representado na parte inferior, o que de fato não ocorre exceto por um único pacote, que por exclusão, é uma resposta ARP do próprio firewall a uma requisição do cliente (veja no **segundo seis** de ambos gráficos). Importante lembrar que este firewall trabalha apenas com filtragem de pacotes IP e não responde pelos pacotes ARP, que seguem outros fluxos na árvore de execução do Click como visto na Figura 5.

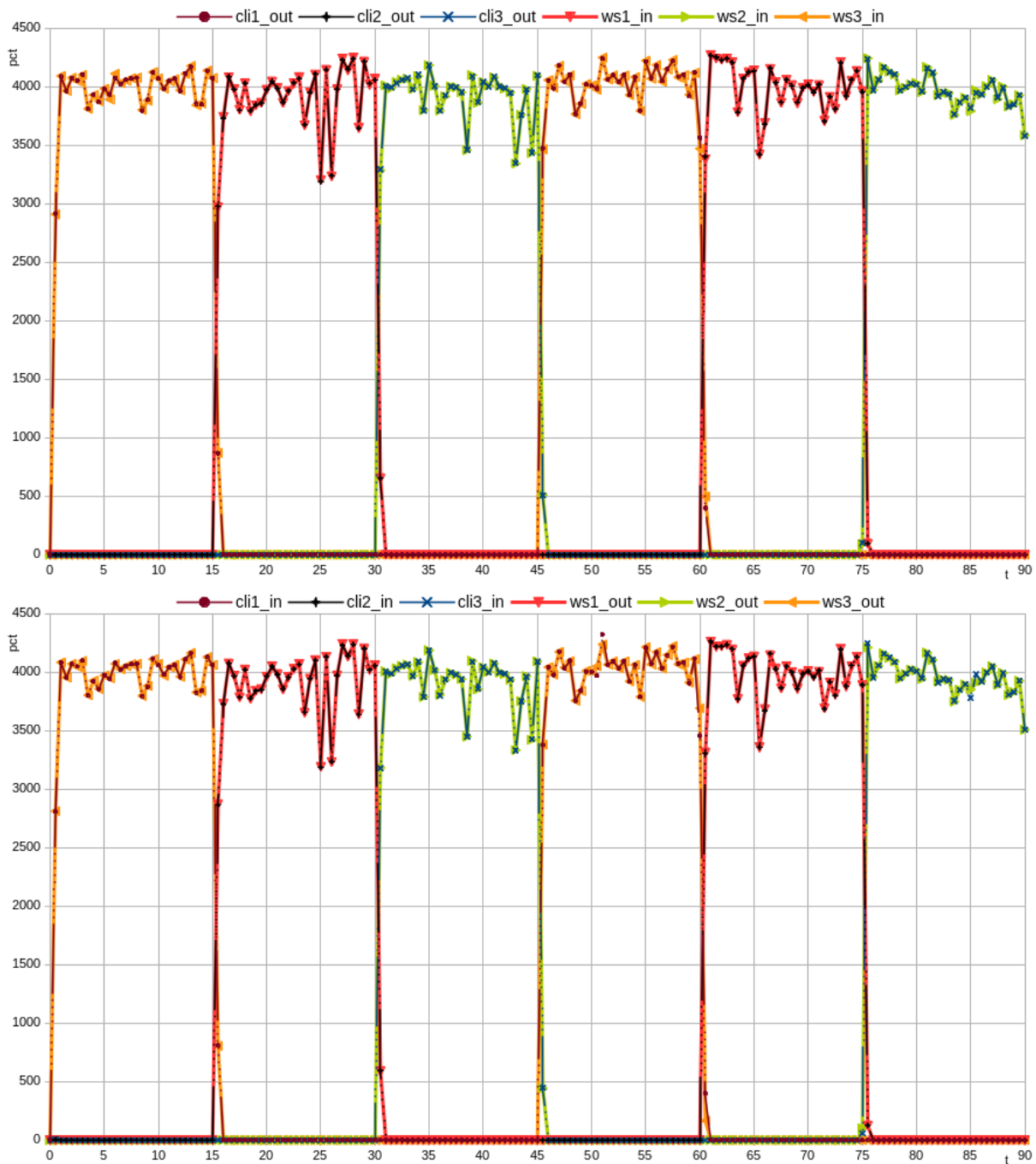
5.4.2 Balanceador de carga por endereço IP de origem

Neste teste, três clientes emitem fluxos para a borda do serviço de hospedagem duas vezes cada, de forma que as emissões não se sobrepõem. O tráfego gerado passa obrigatoriamente pelo firewall, e uma vez filtrado chega ao balanceador, que o distribui aos servidores web configurados na VNF através da aplicação de um DNAT.

Novamente o BWM-NG é usado para monitorar as interfaces virtuais relacionadas ao cliente e aos servidores, que são respectivamente origem e destino dos pacotes, a procura de padrões que demonstrem o comportamento do fluxo. Não existem quaisquer outras conexões vigentes no ambiente de teste.

A execução é feita em sequência, partindo do primeiro cliente ao terceiro, duas vezes. São seis execuções sequenciais e alternadas de no máximo quinze segundos cada para `http://192.0.2.61:80/index2.html`, gerando o arquivo CSV, e armazenando os segundos do relógio do sistema para auxílio na geração dos gráficos e análise dos resultados.

Figura 16 – Fluxo de pacotes no balanceador de carga por endereço IP



Fonte: Elaborada pelo autor

Nos gráficos da [Figura 16](#) primeiro, acima, demonstra a saída dos pacotes dos clientes e a chegada deles nos servidores web e o segundo, abaixo, demonstra o caminho contrário, onde os pacotes-resposta retornam do servidor para o cliente. Em todos os seis momentos do primeiro gráfico há um cliente enviando e seus pacotes chegando a um único servidor, inclusive mantendo a relação cliente servidor definida nas segundas tentativas. Cada cliente foi balanceado para um servidor diferente, embora nesse caso não seja um resultado ruim que dois deles fossem encaminhados para o mesmo servidor, já

que o elemento não analisa o estado atual dos nós balanceadores. Claro que ao se tratar de uma conexão TCP, os pacotes-resposta também precisam retornar corretamente para que não haja uma queda na transmissão por parte do cliente, o que acontece e valida o bom funcionamento da VNF.

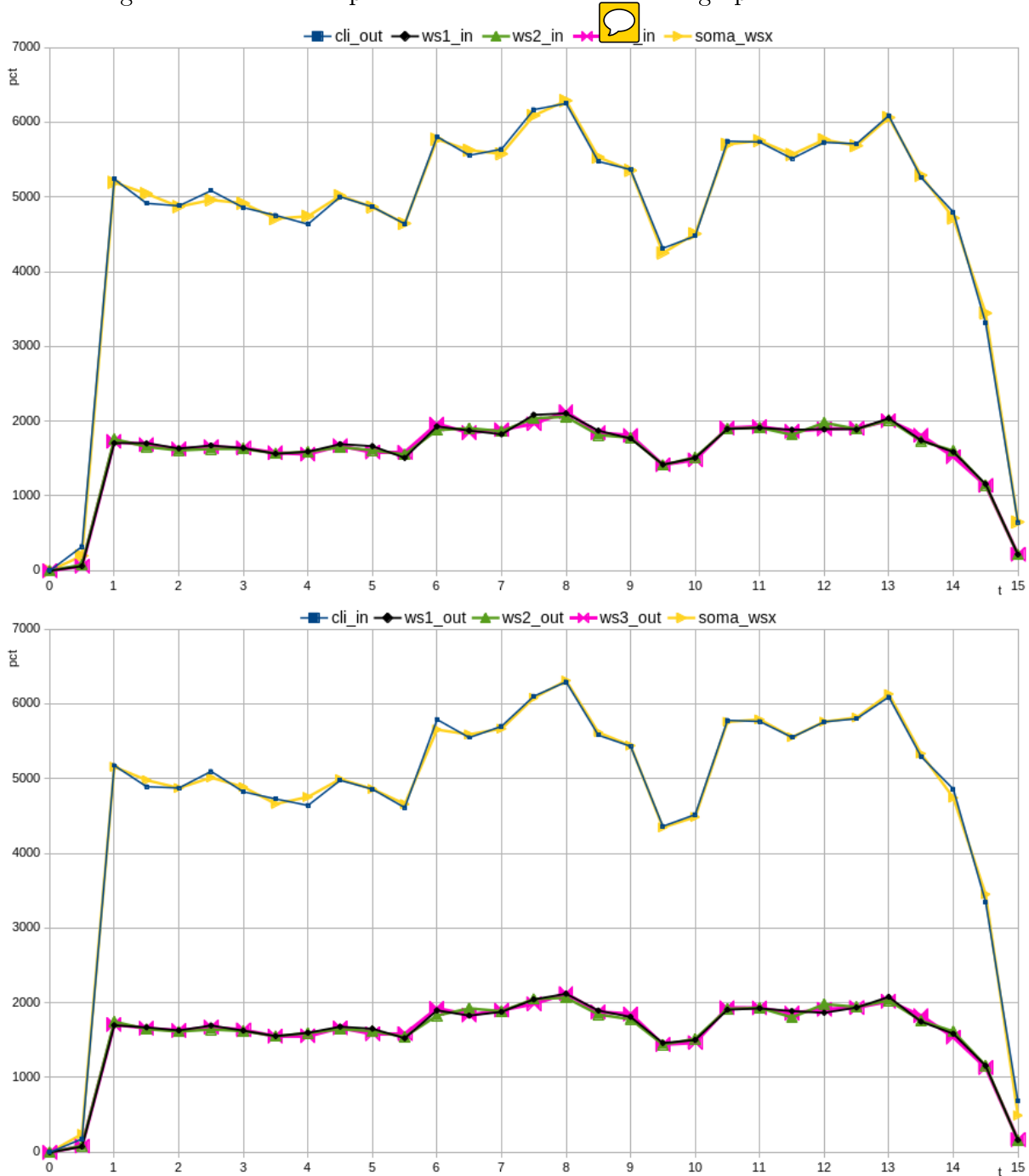
5.4.3 Balanceador de carga por Round Robin

O Siege emite o mesmo tipo de fluxo do firewall partir de um cliente para a borda do serviço de hospedagem. O tráfego HTTP na porta 80 passa obrigatoriamente pelo firewall, e uma vez filtrado chega ao balanceador de carga, que o distribui aos servidores web configurados na VNF, através da aplicação de um DNAT.

O BWM-NG é usado novamente para monitorar as interfaces virtuais relacionadas ao cliente e aos servidores web, que são origem e destino dos pacotes, a procura de padrões que demonstrem o comportamento do fluxo. Não trafegam quaisquer outras conexões pelas interfaces testadas.

A execução ocorre de forma parecida com a do teste anterior, porém sem a necessidade de múltiplas execuções. Desta forma, apenas um cliente é necessário.

Figura 17 – Fluxo de pacotes no balanceador de carga por Round Robin



Fonte: Elaborada pelo autor

Na parte superior da Figura 17, está o fluxo de pacotes enviados pelo cliente. Observa-se uma semelhança entre as quantidades de pacotes enviados pelo cliente e recebidos por todos os servidores. Os servidores web recebem individualmente cerca de um terço do total de pacotes, como é possível ver nas três linhas que os representam, praticamente sobrepostas, e corroboram com a expectativa despejada em cima do balanceador de carga desenvolvido.

Na parte inferior da figura está o fluxo de pacotes recebido pelo cliente, vindo dos

servidores, e em resposta as requisições HTTP. Este gráfico se assemelha ao anterior em suas linhas e nas quantidades de pacotes enviados a cada intervalo, o que é de se esperar de uma rede neutra, e demonstrando que o teste não sofreu reações adversas, tampouco alta carga de processamento com os parâmetros usados.

Para tornar os gráficos mais limpos, foram excluídas as representações dos clientes não utilizados no momento da execução.

5.4.4 Desempenho

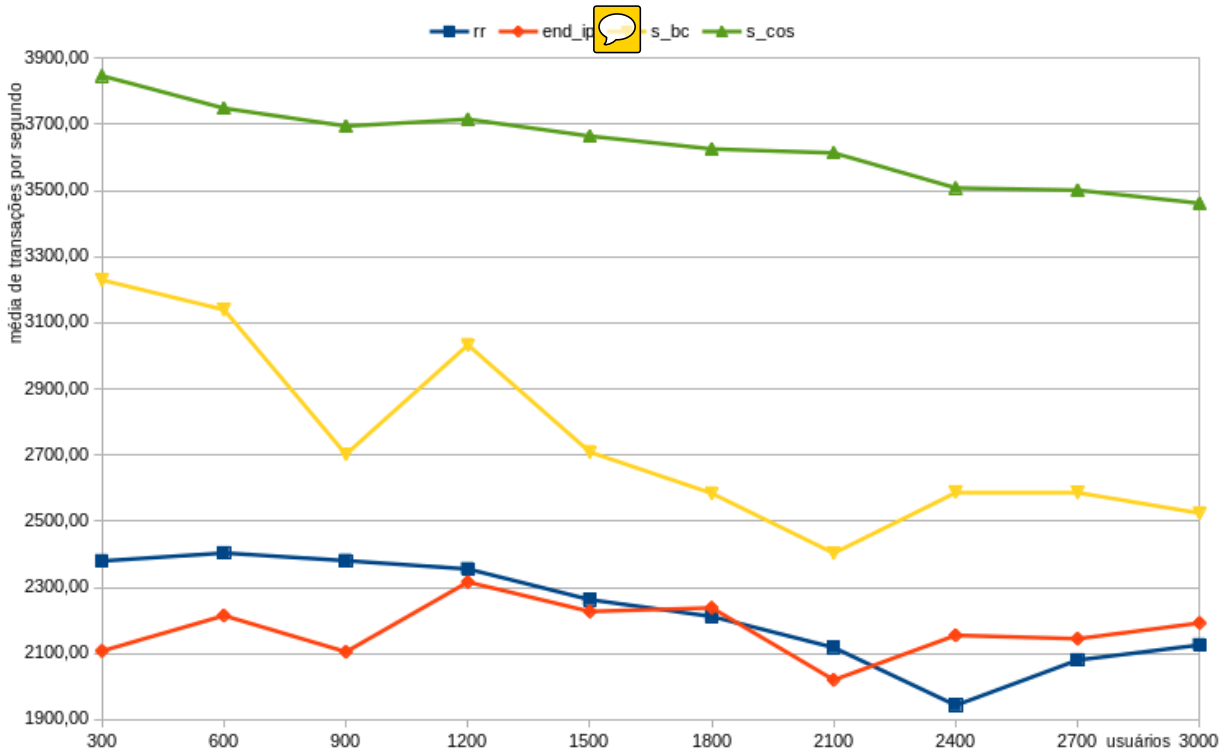
Embora o desempenho não seja primordial para este trabalho, é possível fazer algumas análises sobre a adição de ClickOS ao ambiente virtualizado, mesmo com uma camada extra de virtualização. A intenção é analisar o tráfego, os erros e o tempo de resposta médio do protótipo e de seus recortes, onde as VNFs são retiradas gradativamente começando pelo balanceador, passando pelo firewall e assim chegando a uma conexão direta entre cliente e servidor.

Apenas o Siege é utilizado, já que em seu relatório indica os dados necessários e com a facilidade do formato CSV. Ele é configurado para enviar pacotes por sessenta segundos aos endereços, que por sua vez variam de acordo com o recorte do protótipo. Outra variável é o número de usuários concorrentes em cada máquina cliente. Durante os pré-testes, verificou-se uma limitação de mil usuários simultâneos para cada cliente, e números acima deste causam vazamentos de memória nas máquinas. Dado que o cenário é todo virtualizado, optou-se por manter três clientes ao invés de buscar mais recursos e clientes com novas VMs.

Para gerar uma amostragem, um passo de cem clientes foi escolhido, totalizando dez execuções crescentes para o protótipo e os recortes. Uma pausa de cinco minutos é dada entre uma execução de amostra e outra, totalizando cerca de sessenta minutos de execução para cada situação testada, para eliminar qualquer conexão ainda estabelecida nos servidores web que estejam aguardando resposta, e portanto ocupando-o.

Dessa forma, os gráficos representam quatro situações: com balanceador Round Robin, com balanceador por endereço IP de origem, apenas com o firewall, e sem VNFs. Todas foram medidas separadamente, de forma que as interferências inerentes ao processamento compartilhado das máquinas e do ambiente de virtualização podem afetar uma mas não a outra. Uma tentativa de minimizar este problema reside em anular o registro de acessos e erros do Apache Worker no caso do processamento nas máquinas, e restringir o uso das máquinas virtuais em meio aos testes, inclusive fazendo os testes durante horários de pouca carga no ambiente do VMware.

Figura 18 – Média de transações por segundo de acordo com o número de usuários



Fonte: Elaborada pelo autor

Na Figura 18, fica clara a influência das VNFs na quantidade de transações efetuadas. É de se esperar que uma ligação direta entre as VMs clientes e as servidoras traga melhores resultados; que logo em seguida viria o desempenho de um elemento que, neste trabalho, é considerado somente leitura, adicionando mais um nó na comunicação cliente-servidor; e por último, também adicionando mais um nó, estariam os balanceadores de carga, que efetuam altas cargas de escrita e afetam o tempo de resposta como demonstrado na Figura 20.

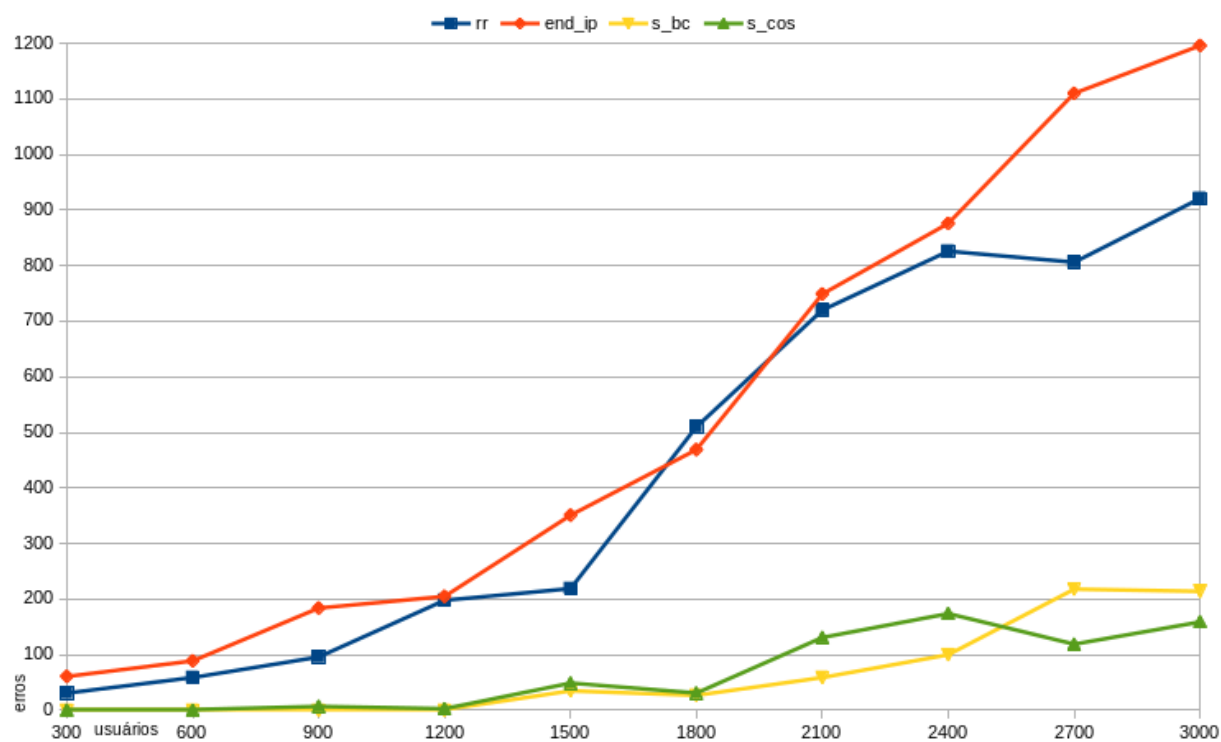
Entre os balanceadores cabe analisar a eficiência de cada um deles dentro do protótipo. Essencialmente, o IPRewriter realiza quatro operações de escrita nos pacotes associados a uma transação, independente do método de balanceamento: as alterações de cabeçalho IP, e o recálculo da soma de verificação TCP, em cada sentido. A adição dessas operações ao sistema trazem os benefícios desejados, mas também uma queda de desempenho notável, aliada a uma falha de implementação que impede as regras com conexões realizadas sejam eliminadas da tabela de regras, causando nesta um crescimento proporcional ao fluxo de entrada, e tornando sua consulta cada vez mais lenta até que o balanceador seja reiniciado.

OBS: Favor ignorar daqui pra frente

Com os dados gerados dos vários pré-testes, não foi possível atribuir diferença significativa de desempenho entre os métodos de balanceamento, e portanto, qualquer

avaliação dos códigos de ambos não faz efeito. A alta variação da média de transações do balanceamento por IP de origem não encontra explicações nos estudos feitos sobre a ferramenta, e passa a assumir questões da carga do sistema como um todo.

Figura 19 – Números absolutos de erros de acordo com o número de usuários

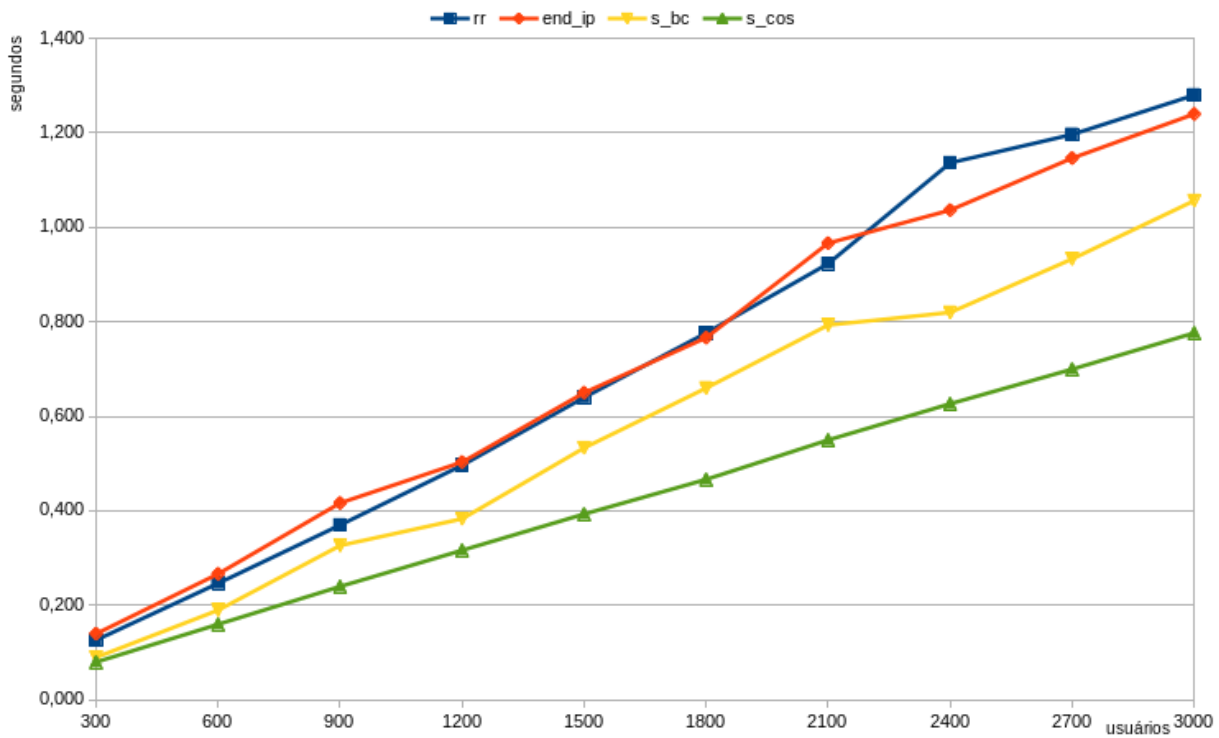


Fonte: Elaborada pelo autor

Nesta parte do teste, é preciso diferenciar os dois principais tipos de erros que acontecem. O primeiro é o erro de tempo limite atingido, que cancela uma conexão dado uma ausência de pacotes de dados ou de controle trocados em determinado tempo. O Siege configura um tempo limite de trinta segundos para estabelecimento de conexão e quando atingido libera o usuário para uma nova tentativa. Já o Apache, configura vários destes no âmbito de dados, podendo o tempo variar de vinte a sessenta segundos por padrão. Já o segundo trata-se de uma conexão reiniciada pelo par, que pode acontecer tanto da parte do cliente quanto do servidor...

A grande parte dos erros apresentados para as execuções sem balanceador são do tipo tempo limite atingido, onde o servidor web reporta com o código 408 o fim do tempo de espera para o recebimento de algum pacote de dados. No caso das execuções com o balanceador, o tipo predominante de erro é o de conexão reiniciada. É de se esperar que o número de erros fosse de uma grandeza próxima na primeira situação devido a exclusiva leitura do firewall, e que crescesse na segunda situação devido as operações de escrita, mas a mudança no tipo de erro merece atenção.

Figura 20 – Tempo médio de resposta de acordo com o número de usuários



Fonte: Elaborada pelo autor

O tempo de resposta na Figura 20 acompanha quase que rigorosamente o comportamento da média da taxa de transações da Figura 18.

6 CONCLUSÃO

Reafirmar a porra toda

Referências Bibliográficas

- 1 MARTINS, J. et al. ClickOS and the Art of Network Function Virtualization. In: *Proceedings of the 11th USENIX Conference on Networked Systems Design and Implementation*. Berkeley, CA, USA: USENIX Association, 2014. (NSDI'14), p. 459–473. ISBN 978-1-931971-09-6. Disponível em: <<http://dl.acm.org/citation.cfm?id=2616448.2616491>>. Citado 5 vezes nas páginas 21, 38, 39, 40 e 48.
- 2 MORRIS, R. et al. The Click Modular Router. In: *Proceedings of the Seventeenth ACM Symposium on Operating Systems Principles*. New York, NY, USA: ACM, 1999. (SOSP '99), p. 217–231. ISBN 978-1-58113-140-6. Disponível em: <<http://doi.acm.org/10.1145/319151.319166>>. Citado 6 vezes nas páginas 21, 29, 30, 31, 32 e 38.
- 3 BARHAM, P. et al. Xen and the Art of Virtualization. In: *Proceedings of the Nineteenth ACM Symposium on Operating Systems Principles*. New York, NY, USA: ACM, 2003. (SOSP '03), p. 164–177. ISBN 978-1-58113-757-6. Disponível em: <<http://doi.acm.org/10.1145/945445.945462>>. Citado na página 21.
- 4 POPURI, S. *A Tour of Mini-OS Kernel*. 200–. Disponível em: <<https://www.cs.uic.edu/~spopuri/minios.html>>. Citado na página 21.
- 5 VENTRE, P. L. et al. Performance evaluation and tuning of Virtual Infrastructure Managers for (Micro) Virtual Network Functions. In: . IEEE, 2016. p. 141–147. ISBN 978-1-5090-0933-6. Disponível em: <<http://ieeexplore.ieee.org/document/7919489/>>. Citado 2 vezes nas páginas 21 e 41.
- 6 BOSSHART, P. et al. P4: Programming protocol-independent packet processors. *SIGCOMM Comput. Commun. Rev.*, ACM, New York, NY, USA, v. 44, n. 3, p. 87–95, jul. 2014. ISSN 0146-4833. Disponível em: <<http://doi.acm.org/10.1145/2656877.2656890>>. Citado na página 22.
- 7 RIZZO, L.; LETTIERI, G. VALE, a Switched Ethernet for Virtual Machines. In: *Proceedings of the 8th International Conference on Emerging Networking Experiments and Technologies*. New York, NY, USA: ACM, 2012. (CoNEXT '12), p. 61–72. ISBN 978-1-4503-1775-7. Disponível em: <<http://doi.acm.org/10.1145/2413176.2413185>>. Citado 3 vezes nas páginas 22, 25 e 39.
- 8 PFAFF, B. et al. Extending Networking into the Virtualization Layer. jan. 2009. Citado 2 vezes nas páginas 22 e 25.
- 9 MIJUMBI, R. et al. Network Function Virtualization: State-of-the-Art and Research Challenges. *IEEE Communications Surveys Tutorials*, v. 18, n. 1, p. 236–262, 2016. ISSN 1553-877X. Citado 3 vezes nas páginas 23, 24 e 25.
- 10 QUMRANET, A. K. et al. Kvm: The linux virtual machine monitor. v. 15, p. 225–230, 06 2007. Citado na página 23.

- 11 COOLEY, S. *Introdução ao Hyper-V no Windows 10*. 2017. Disponível em: <https://docs.microsoft.com/pt-br/virtualization/hyper-v-on-windows/about/>. Citado na página 23.
- 12 SERVER Virtualization Software | vSphere | VMware. 2018. Disponível em: <https://www.vmware.com/products/vsphere.html>. Citado na página 23.
- 13 WATSON, J. VirtualBox: Bits and Bytes Masquerading As Machines. *Linux J.*, v. 2008, n. 166, fev. 2008. ISSN 1075-3583. Disponível em: <http://dl.acm.org/citation.cfm?id=1344209.1344210>. Citado na página 23.
- 14 ETSI, N. Gs nfv 003-v1. 2.1-network function virtualisation (nfv): Terminology for main concepts in nfv. *publishing December*, 2014. Citado na página 23.
- 15 BRIM, S. W.; CARPENTER, B. E. *Middleboxes: Taxonomy and Issues*. RFC Editor, 2002. RFC 3234. (Request for Comments, 3234). Disponível em: <https://rfc-editor.org/rfc/rfc3234.txt>. Citado na página 23.
- 16 MERKEL, D. Docker: Lightweight linux containers for consistent development and deployment. *Linux J.*, Belltown Media, Houston, TX, v. 2014, n. 239, mar. 2014. ISSN 1075-3583. Disponível em: <http://dl.acm.org/citation.cfm?id=2600239.2600241>. Citado na página 24.
- 17 BIEDERMAN, E. W.; NETWORK, L. Multiple instances of the global linux namespaces. In: CITESEER. *Proceedings of the Linux Symposium*. [S.l.], 2006. v. 1, p. 101–112. Citado na página 24.
- 18 HAN, B. et al. Network function virtualization: Challenges and opportunities for innovations. *IEEE Communications Magazine*, v. 53, n. 2, p. 90–97, fev. 2015. ISSN 0163-6804. Citado na página 24.
- 19 PRICE, C.; RIVERA, S. et al. Opnfv: An open platform to accelerate nfv. *White Paper. A Linux Foundation Collaborative Project*, 2012. Disponível em: https://networkbuilders.intel.com/docs/OPNFV_WhitePaper_Final.pdf. Citado na página 24.
- 20 SEFRAOUI, O.; AISSAOUI, M.; ELEULDJ, M. OpenStack: Toward an Open-Source Solution for Cloud Computing. *International Journal of Computer Applications*, v. 55, p. 38–42, out. 2012. Citado na página 24.
- 21 IRTF Network Function Virtualization Research Group (NFVRG). 2015. Disponível em: <https://irtf.org/nfvrg>. Citado na página 24.
- 22 VARIS, N. Anatomy of a linux bridge. In: *Proceedings of Seminar on Network Protocols in Operating Systems*. [S.l.: s.n.], 2012. p. 58. Citado na página 25.
- 23 LINUX - The Linux Foundation. 2018. Disponível em: <https://www.linuxfoundation.org/projects/linux/>. Citado na página 25.
- 24 IEEE Standard for Ethernet. *IEEE Std 802.3-2015 (Revision of IEEE Std 802.3-2012)*, p. 1–4017, mar. 2016. Citado 2 vezes nas páginas 26 e 31.
- 25 XEN Networking - Xen. 2018. Disponível em: https://wiki.xen.org/wiki/Xen_Networking. Citado 2 vezes nas páginas 26 e 27.

- 26 POSTEL, J. et al. *Internet Protocol*. RFC Editor, 1981. RFC 791. (Request for Comments, 791). Disponível em: <<https://rfc-editor.org/rfc/rfc791.txt>>. Citado na página 26.
- 27 DAY, J. D.; ZIMMERMANN, H. The OSI reference model. *Proceedings of the IEEE*, v. 71, n. 12, p. 1334–1340, dez. 1983. ISSN 0018-9219. Citado na página 29.
- 28 STROUSTRUP, B. *The C++ Programming Language*. 4th. ed. [S.l.]: Addison-Wesley Professional, 2013. ISBN 0321563840, 9780321563842. Citado na página 29.
- 29 KOHLER, E. *click: The Click modular router: fast modular packet processing and analysis*. 2018. Original-date: 2010-09-21T22:45:25Z. Disponível em: <<https://github.com/kohler/click>>. Citado 4 vezes nas páginas 30, 35, 43 e 48.
- 30 RASMUSSEN, R. V.; TRICK, M. A. Round robin scheduling – a survey. *European Journal of Operational Research*, v. 188, n. 3, p. 617 – 636, 2008. ISSN 0377-2217. Disponível em: <<http://www.sciencedirect.com/science/article/pii/S0377221707005309>>. Citado na página 32.
- 31 PLUMMER, D. C. *An Ethernet Address Resolution Protocol: Or Converting Network Protocol Addresses to 48.bit Ethernet Address for Transmission on Ethernet Hardware*. RFC Editor, 1982. RFC 826. (Request for Comments, 826). Disponível em: <<https://rfc-editor.org/rfc/rfc826.txt>>. Citado na página 33.
- 32 SMOOT, C.-M.; QUARTERMAN, J. S. *Using ARP to implement transparent subnet gateways*. RFC Editor, 1987. RFC 1027. (Request for Comments, 1027). Disponível em: <<https://rfc-editor.org/rfc/rfc1027.txt>>. Citado na página 33.
- 33 POSTEL, J. et al. *Transmission Control Protocol*. RFC Editor, 1981. RFC 793. (Request for Comments, 793). Disponível em: <<https://rfc-editor.org/rfc/rfc793.txt>>. Citado na página 35.
- 34 FEDOR, M. et al. *Simple Network Management Protocol (SNMP)*. RFC Editor, 1990. RFC 1157. (Request for Comments, 1157). Disponível em: <<https://rfc-editor.org/rfc/rfc1157.txt>>. Citado na página 37.
- 35 KOHLER, E.; MORRIS, R.; POLETTTO, M. Modular components for network address translation. In: . IEEE, 2002. p. 39–50. ISBN 978-0-7803-7457-7. Disponível em: <<http://ieeexplore.ieee.org/document/1019227/>>. Citado na página 37.
- 36 POSTEL, J. *User Datagram Protocol*. RFC Editor, 1980. RFC 768. (Request for Comments, 768). Disponível em: <<https://rfc-editor.org/rfc/rfc768.txt>>. Citado na página 38.
- 37 CSOMA, A. et al. ESCAPE: Extensible Service Chain Prototyping Environment Using Mininet, Click, NETCONF and POX. In: *Proceedings of the 2014 ACM Conference on SIGCOMM*. New York, NY, USA: ACM, 2014. (SIGCOMM '14), p. 125–126. ISBN 978-1-4503-2836-4. Disponível em: <<http://doi.acm.org/10.1145/2619239.2631448>>. Citado na página 38.
- 38 LAUFER, R. et al. CliMB: Enabling Network Function Composition with Click Middleboxes. In: *Proceedings of the 2016 Workshop on Hot Topics in Middleboxes and Network Function Virtualization*. New York, NY, USA: ACM,

2016. (HotMiddlebox '16), p. 50–55. ISBN 978-1-4503-4424-1. Disponível em: <http://doi.acm.org/2940147.2940152>. Citado na página 38.
- 39 BARBETTE, T.; SOLDANI, C.; MATHY, L. Fast Userspace Packet Processing. In: *Proceedings of the Eleventh ACM/IEEE Symposium on Architectures for Networking and Communications Systems*. Washington, DC, USA: IEEE Computer Society, 2015. (ANCS '15), p. 5–16. ISBN 978-1-4673-6632-8. Disponível em: <http://dl.acm.org/citation.cfm?id=2772722.2772727>. Citado na página 38.
- 40 BONDAN, L.; SANTOS, C. R. P. d.; GRANVILLE, L. Z. Management requirements for ClickOS-based Network Function Virtualization. In: *10th International Conference on Network and Service Management (CNSM) and Workshop*. [S.l.: s.n.], 2014. p. 447–450. Citado na página 38.
- 41 RIZZO, L. Revisiting Network I/O APIs: The Netmap Framework. *Commun. ACM*, v. 55, n. 3, p. 45–51, mar. 2012. ISSN 0001-0782. Disponível em: <http://doi.acm.org/10.1145/2093548.2093565>. Citado na página 39.
- 42 PANDA, A. et al. NetBricks: Taking the V out of NFV. In: *Proceedings of the 12th USENIX Conference on Operating Systems Design and Implementation*. Berkeley, CA, USA: USENIX Association, 2016. (OSDI'16), p. 203–216. ISBN 978-1-931971-33-1. Disponível em: <http://dl.acm.org/citation.cfm?id=3026877.3026894>. Citado na página 41.
- 43 PALKAR, S. et al. E2: A Framework for NFV Applications. In: *Proceedings of the 25th Symposium on Operating Systems Principles*. New York, NY, USA: ACM, 2015. (SOSP '15), p. 121–136. ISBN 978-1-4503-3834-9. Disponível em: <http://doi.acm.org/10.1145/2815400.2815423>. Citado na página 41.
- 44 ZHANG, W. et al. OpenNetVM: A Platform for High Performance Network Service Chains. In: *Proceedings of the 2016 Workshop on Hot Topics in Middleboxes and Network Function Virtualization*. New York, NY, USA: ACM, 2016. (HotMiddlebox '16), p. 26–31. ISBN 978-1-4503-4424-1. Disponível em: <http://doi.acm.org/2940147.2940155>. Citado na página 41.
- 45 YURCHENKO, M. et al. OpenNetVM: A Platform for High Performance NFV Service Chains. In: *Proceedings of the Symposium on SDN Research*. New York, NY, USA: ACM, 2018. (SOSR '18), p. 21:1–21:2. ISBN 978-1-4503-5664-0. Disponível em: <http://doi.acm.org/10.1145/3185467.3190786>. Citado na página 41.