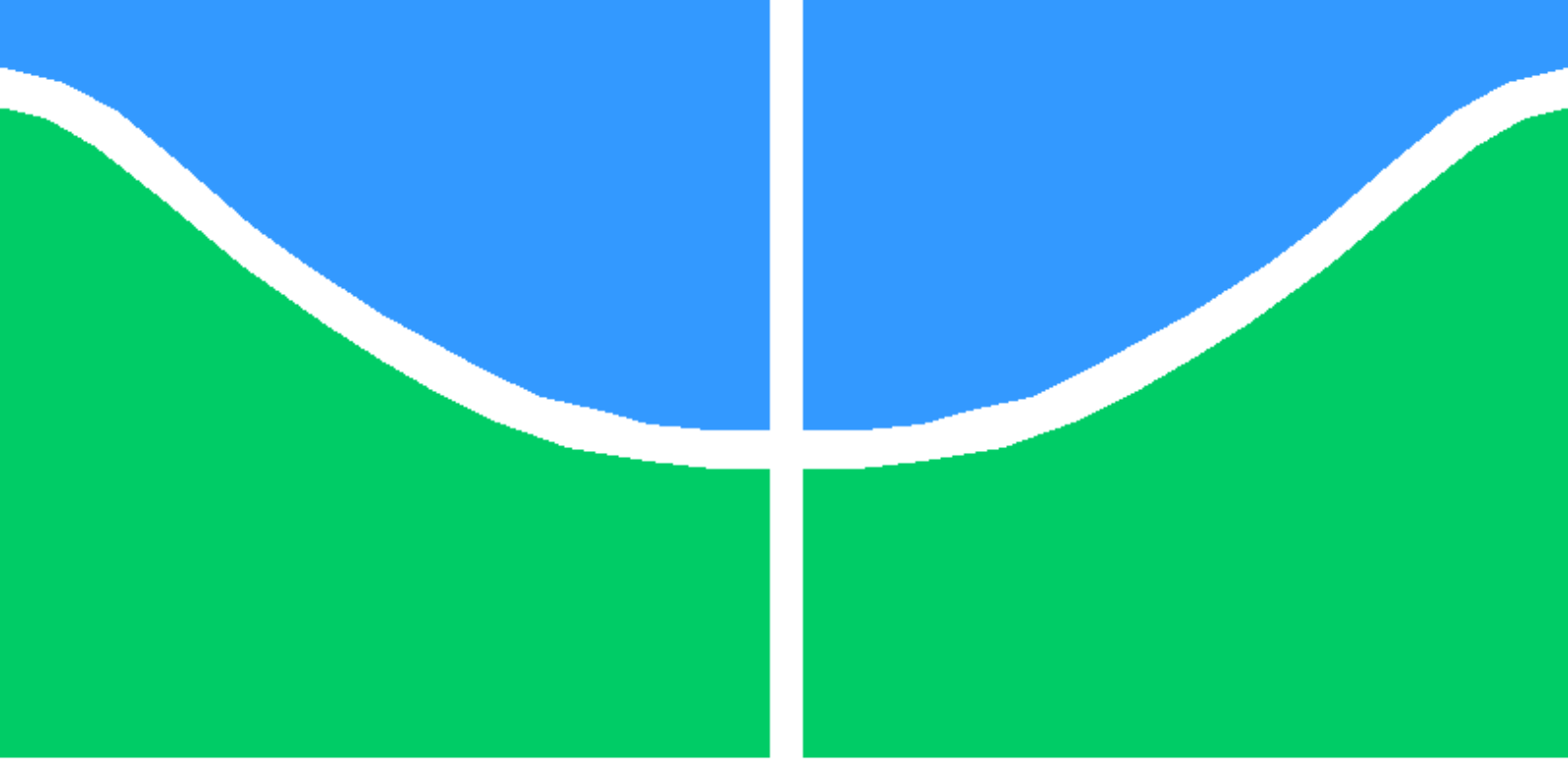


Proposta de arquitetura SDN para uma sala de Provas Eletrônicas

Caio Guilherme Lisboa de Oliveira

Paulo Vítor Souto Soares de Sousa

Brasília, Fevereiro de 2023

UNIVERSIDADE DE BRASÍLIA

FACULDADE DE TECNOLOGIA

UNIVERSIDADE DE BRASILIA
Faculdade de Tecnologia

Proposta de arquitetura SDN para uma sala de Provas Eletrônicas

Caio Guilherme Lisboa de Oliveira

Paulo Vítor Souto Soares de Sousa

Relatório submetido ao Departamento de Engenharia Elétrica como requisito parcial para obtenção do grau de Engenheiro de Redes de Comunicação

Banca Examinadora

Georges Daniel Amvame Nze, Dr., EnE/UnB _____
Orientador

Fábio Lúcio Lopes de Mendonça, Dr., EnE/UnB _____
Examinador Interno

Diego Martins de Oliveira, Esp., IFB/Brasília _____
Examinador Externo

Agradecimentos

Quero expressar minha profunda gratidão a todas as pessoas que me ajudaram a chegar até aqui. Em primeiro lugar, gostaria de agradecer à minha família por todo o amor e apoio incondicional que sempre me deram. Vocês são minha base, minha rocha e, sem vocês, eu não seria quem sou hoje.

Agradeço também aos meus professores e mentores, em especial o professor Dr. Georges Daniel Amvame Nze. Vocês me ensinaram tanto, não só sobre o assunto em questão, mas também sobre a vida. Sua sabedoria e experiência foram fundamentais para o meu crescimento pessoal e profissional.

Gostaria de agradecer também aos meus colegas de trabalho, colegas de estudo e amigos. Vocês me desafiaram e me inspiraram diariamente. Trabalhar e estudar com vocês foi um privilégio e estou grato pelas amizades e parcerias que criamos.

Por fim, quero agradecer a todas as pessoas que, de alguma forma, contribuíram para minha jornada. Se você já me deu um conselho, me deu uma oportunidade, me ajudou em um projeto ou simplesmente me deu um sorriso, obrigado.

Caio Guilherme Lisboa de Oliveira

Agradecimentos

Agradeço à minha família por ter me apoiado durante todo este trabalho e também durante o período de formação na graduação.

Ao meu orientador, professor Dr. Georges Daniel Amvame Nze, por ter nos guiado no processo de desenvolvimento deste trabalho.

Aos professores da UnB, por sempre exigerem o melhor de cada um de nós.

Também devo agradecimento aos desenvolvedores dos diversos softwares de código aberto usados neste trabalho.

Paulo Vitor Souto Soares de Sousa

Resumo

O crescente aumento no número de usuários do Moodle expõe os diversos problemas de segurança encontrados para realização de provas virtuais. Neste contexto, este trabalho apresenta uma proposta de arquitetura SDN (Software Defined Networking) para uma sala de provas eletrônicas. É criado um ambiente de provas seguro e com um sistema prático de administração. Para isso, algumas regras de controle de fluxo são propostas. Por fim, o trabalho é validado por meio da comprovação do funcionamento do ambiente.

Palavras-chave: GNS3, Open vSwitch, OpenFlow Manager, Moodle, SDN, Wireshark.

Abstract

The growing increase in the number of Moodle users exposes the various security issues found for conducting virtual exams. In this context, this work presents an SDN (Software Defined Networking) architecture proposal for an electronic exam room. A safe test environment with a practical management system is created. To this end, flow control rules are proposed. Finally, the environment is run and explained to validate the work.

Keywords: GNS3, Open vSwitch, OpenFlow Manager, Moodle, SDN, Wireshark.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS

API	Application Programming Interface
ARP	Address Resolution Protocol
BGP	Border Gateway Protocol
DHCP	Dynamic Host Configuration Protocol
DNS	Domain Name System
GUI	Graphic User Interface
HTTP	HyperText Transfer Protocol
HTTPS	HyperText Transfer Protocol Secure
ICMP	Internet Control Message Protocol
IP	Internet Protocol
LLDP	Link Layer Discovery Protocol
MAC	Media Access Control
MD-SAL	Model-Driven Service Abstraction Layer
MPLS	Multi-Protocol Label Switching
MTU	Maximum Transmission Unit
NAC	Network Access Control
NIC	Network Interface Card
Nmap	Network Mapper
ODL	OpenDaylight
OF	OpenFlow
OFM	OpenFlow Manager
OSPF	Open Shortest Path First
OVS	Open vSwitch
OVSDB	Open vSwitch Database
QR Code	Quick Response Code
RAM	Random Access Memory
SDN	Software-Defined Networking
SSH	Secure Shell
SSL	Secure Sockets Layer
TCP	Transmission Control Protocol
TLS	Transport Layer Security
VLAN	Virtual Local Area Network
VM	Virtual Machine
VPN	Virtual Private Network
WSL2	Windows Subsystem for Linux
XML	eXtensible Markup Language

1 Introdução

Como consequência da pandemia de COVID-19, os ambientes de ensino e provas tiveram que ser migrados para plataformas on-line. O aumento do número de usuários do Moodle — cerca de 70 milhões em 2015 para mais de 290 milhões em 2021 — comprova essa tendência, conforme é dito por [1]. Na Universidade de Brasília, esta plataforma já era usada antes mesmo do avanço da pandemia. E mesmo após o relaxamento das medidas sanitárias, vários cursos ainda utilizam o ambiente virtual como plataforma de ensino.

Além da evolução dos ambientes de prova, tem-se a evolução do gerenciamento de redes. A Rede Definida por Software (SDN) é uma abordagem capaz de simplificar e otimizar o gerenciamento em redes pela mudança do controle da rede para um dispositivo central, ou alguns dispositivos. É uma tecnologia importante, visto que grandes empresas — Cisco, Facebook, Google — colaboraram entre si para incentivar a adoção da arquitetura SDN de código aberto [2]. A grande vantagem é a mudança do controle em cada dispositivo para um controle central da rede.

Considerando tal situação, existe a possibilidade de integração de ambas as tecnologias para a criação de um ambiente seguro para instituições escolares. É possível criar uma solução segura sem utilizar o modelo de arquitetura tradicional de redes, em que cada dispositivo é configurado individualmente.

Por isso, este projeto tem como intenção apresentar uma proposta de arquitetura de uma sala de provas controlada de forma centralizada, utilizando-se Redes Definidas por Software (SDN). Nela são dispostos três ambientes distintos, isolados por meio de um Switch central, controlado pela interação entre o protocolo OpenFlow e um controlador de administração, permitindo a definição de regras de fluxo, orquestração e segmentação da rede.

1.1 Motivação

A ideia inicial deste trabalho era utilizar o software desenvolvido em [3] e complementar a parte de SDN. Porém, devido a diversos problemas, não foi possível usar o software. Como solução, a plataforma Moodle foi utilizada como ambiente de provas, além da criação de um ambiente seguro de provas e com sistema prático de administração com SDN.

Assim como o trabalho [3], este também possui uma preocupação com segurança. No entanto, a segurança é relacionada com quais dispositivos e quais sites os hosts estão tentando se conectar. A ideia é criar um ambiente seguro em horários de prova, ter controle do que está sendo feito na rede de forma geral.

O momento de realização de provas é um período importante para a vida acadêmica dos alunos. Por isso, é importante garantir que todos estão aplicando os conhecimentos adquiridos. Logo, outra preocupação é impedir o uso de meios ilícitos nos períodos de realização de provas. Por esse motivo, há a possibilidade de bloquear tráfego que não seja para o servidor de provas.

1.2 Objetivo Geral

O objetivo principal do trabalho é a disponibilização de um ambiente de provas seguro, à prova de fraudes e com um sistema prático de administração. As ferramentas de Redes Definidas por Software são utilizadas para construção da simulação.

O embasamento do trabalho contempla estudos sobre diversos temas apresentados no capítulo 2, como Docker, Open vSwitch, SDN, entre outros.

1.3 Objetivos Específicos

Para alcançar o objetivo geral, os seguintes objetivos específicos foram propostos:

1. Estudar e especificar funcionalidades para gerenciamento de um ambiente de provas;
2. Propor um ambiente baseado no Laboratório de Redes da UnB;
3. Definir quais ferramentas são necessárias;
4. Realizar a configuração de todos dispositivos presentes na simulação;
5. Estudar quais regras podem ser aplicadas no ambiente de simulação;

1.4 Metodologia

Tendo em vista os objetivos destacados, este trabalho foi dividido em nove momentos, conforme é mostrado na figura 1.1. Primeiramente, era necessário definir o tema do projeto. que foi realizada por meio de discussões com o orientador.

Com o tema definido, era primordial obter conhecimentos teóricos sobre o assunto e estudar projetos similares para se ter uma visão geral do projeto. Então, o trabalho [3] foi estudado e resumido para analisar como SDN poderia ser encaixado no trabalho já feito. Além disso, era preciso definir quais eram os objetivos finais.

Em seguida, foi criado, no GNS3 (Graphical Network Simulator 3), a topologia do projeto, conforme figura 3.1. A quantidade e tipos de dispositivos e softwares foram escolhidos para criar uma simulação do projeto. Além da topologia, foram analisadas quais especificações de hardware suportariam a simulação. O hardware primordial era a memória RAM (Random Access Memory). Para rodar a simulação, era necessário alocar, pelo menos, 7 gigabytes de memória RAM para o GNS3.

A partir de então, as máquinas da topologia foram criadas e configuradas. Os adaptadores dos dispositivos também foram ajustados para atender às demandas. Optou-se por inserir os dispositivos da simulação na rede física, com o fim de torná-los acessíveis por qualquer dispositivo na rede.

Alguns dispositivos foram criados em Docker. A escolha de contêineres Docker para montagem da topologia foi motivada pela agilidade, baixa utilização de recursos e velocidade de implantação, o que é fundamental para uma topologia com muitos dispositivos.

Então, alguns testes foram feitos para garantir que todos os dispositivos da topologia tinham comunicação entre si. Além disso, os softwares OpenDaylight e OpenFlow Manager passaram a ser usados para definição de regras de fluxo. Essas definições foram escolhidas para atender os objetivos iniciais.

No penúltimo momento, foi feita a validação dos resultados. O ambiente foi simulado para se obter os resultados. Para isso, algumas regras foram editadas e trocadas.

Por último, foi feita uma revisão geral do que foi desenvolvido. A possibilidade de criação de novas funcionalidades no projeto também foram pensadas. Então, foi feita a entrega do projeto com as configurações e especificações de cada passo.

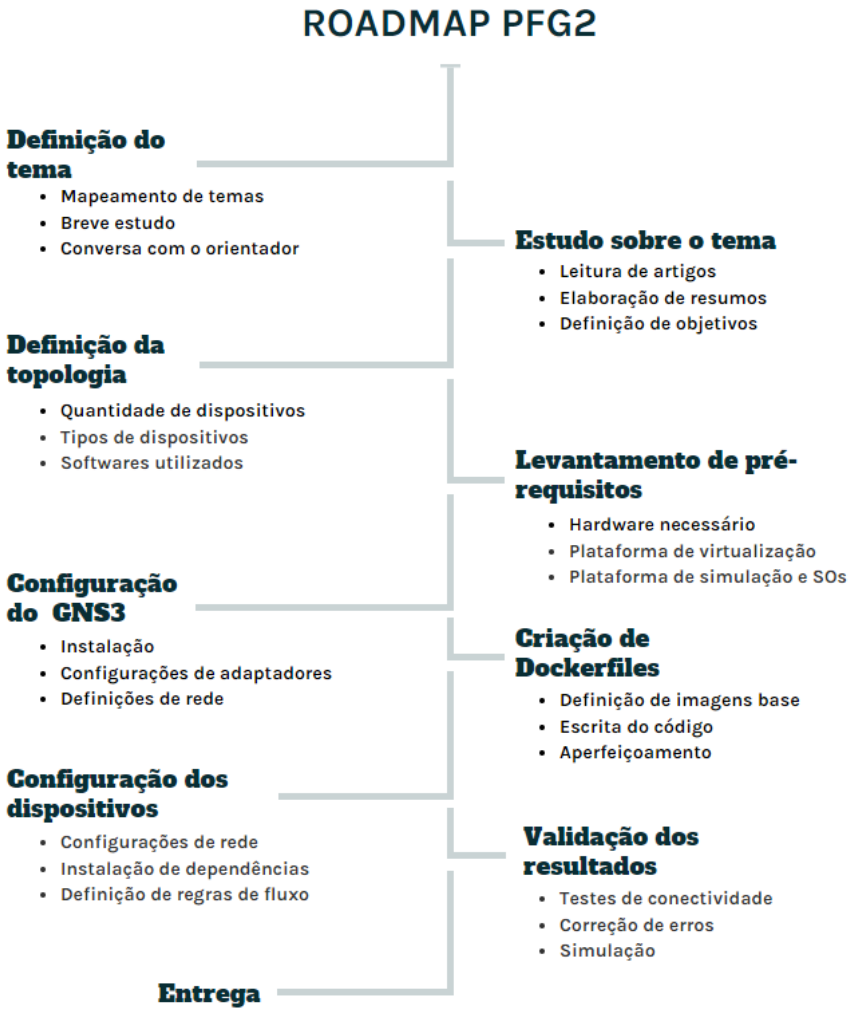


Figura 1.1 – Roadmap do projeto.

1.5 Organização do Trabalho

Este trabalho visa o estudo, implementação e análise de resultados do ambiente de provas. A divisão do conteúdo está feita em cinco capítulos mais o anexo.

- No primeiro, é feita uma breve introdução dos assuntos e descrição do problema, além da apresentação de motivações, objetivos e visão geral do trabalho.
- No capítulo 2, é feita uma revisão dos principais assuntos necessários para o desenvolvimento do projeto. Também apresenta alguns trabalhos relacionados que serviram como base.
- No terceiro capítulo, há a descrição de configuração das ferramentas e o do próprio ambiente construído para a realização dos testes. Por fim, como as regras foram definidas para criação do ambiente seguro.
- No quarto capítulo, alguns testes são feitos para validar as propostas iniciais e discussão de resultados, com base em dados adquiridos por ferramentas de análise de tráfego, como o Wireshark.
- No último capítulo, há a discussão final sobre o projeto e conclusões feitas. Além disso, são feitas proposições de trabalhos futuros.
- Por fim, o anexo apresenta configurações necessárias para execução de algumas ferramentas.

2 Revisão Bibliográfica e Trabalhos Relacionados

Neste capítulo, há o detalhamento das ferramentas e tecnologias utilizadas para o desenvolvimento deste projeto. O objetivo é explicar o funcionamento e principais funcionalidades de cada ferramenta individualmente, antes de detalhar o projeto. Por fim, alguns trabalhos relacionados são listados.

2.1 GNS3: Graphical Network Simulator 3

Antes da virtualização de equipamentos de rede, era necessário usar dispositivos físicos para simular ambientes de rede. Obviamente, era uma forma cara e pouco flexível para fazer testes. Por isso, alguns simuladores foram sendo desenvolvidos, em destaque o GNS3 [4].

O GNS3 é um software de simulação de redes que permite a criação e teste de configurações em redes reais e virtuais sem a necessidade de equipamentos físicos. A plataforma é baseada em software livre e é compatível com vários sistemas operacionais, incluindo Windows, Linux e macOS [5]. Nele, é possível customizar ambientes para criar topologias específicas com uma variedade de roteadores, switches e compurtadores [4]. Isso permite teste de configurações e aplicação de soluções sem afetar a rede real [5].

O simulador é composto, basicamente, por dois componentes: o GNS3-all-in-one software (GUI) e a GNS3 VM [5].

1. O GUI (Graphic User Interface) consiste da parte cliente e a interface gráfica do GNS3 no mesmo arquivo de instalação do hospedeiro local. Após instalado, o software usa o próprio hospedeiro como servidor. Então, a comunicação entre cliente e servidor passa a ser entre a interface gráfica do usuário - cliente - e o hospedeiro no qual se encontra - servidor.

2. A GNS3 VM é o servidor virtual que faz a comunicação com a interface gráfica do usuário. Normalmente, ele possui dois adaptadores de rede virtuais: um para realizar a comunicação entre o cliente e o servidor e outro para estabelecer comunicação com a internet pública.

A documentação [5] abrange múltiplos tópicos instalação, configuração e uso do software, solução de problemas, gerenciamento de projetos, etc.

2.2 Docker

Docker é uma plataforma aberta que permite desenvolvimento, compartilhamento e execução de aplicações. Faz a separação entre aplicação e infraestrutura para rápido desenvolvimento de software. Com ele, é possível gerenciar a infraestrutura de maneira semelhante ao gerenciamento

de aplicações. Ao tirar proveito da metodologia Docker, o tempo entre escrita e teste de códigos é reduzido de forma expressiva [6].

2.2.1 Arquitetura Docker

Docker usa uma arquitetura cliente-servidor. O cliente se comunica com o daemon, que faz o "build", controla o nível de isolamento, roda e distribui os contêineres. O daemon espera requisições API (Application Programming Interface) e faz o gerenciamento de objetos — imagens, contêineres e volumes [6].

Uma imagem é um template "read-only" com instruções para a criação do contêiner. Para isso, cria-se um Dockerfile com os passos necessários para montar a imagem e rodá-la (é possível construir uma imagem própria) [6]. Há um repositório on-line que permite o compartilhamento de imagens entre os usuários, chamado de Docker Hub.

O Docker empacota e roda a imagem num ambiente isolado, chamado de contêiner. O isolamento permite a execução de múltiplos contêineres ao mesmo tempo no host [6]. Eles são isolados por conta de dois aspectos de kernel: namespaces e control groups (cgroups). Os namespaces são usados para dividir a "visão" que os processos têm do sistema. Atualmente, o kernel possui seis namespaces: PID, PIC, NET, MNT, UTS e USER. Já os cgroups são um mecanismo de restrição de uso de recursos de um processo ou grupo de processos. Os recursos controlados são: CPU, RAM, largura de banda de rede e I/O de disco. O objetivo é prevenir que um processo utilize todos os recursos da máquina [7].

Os contêineres não incorporam o próprio kernel como máquinas virtuais, em vez disso rodam no kernel do host, o que diminui o caminho das chamadas de sistema e reduz a sobrecarga de software. Além disso, podem dividir recursos de software, como bibliotecas com o host, evitando, assim, duplicação de código. A ausência de kernel e algumas bibliotecas faz com que as imagens do contêiner sejam de apenas alguns megabytes e que o processo de inicialização seja muito rápido [7].

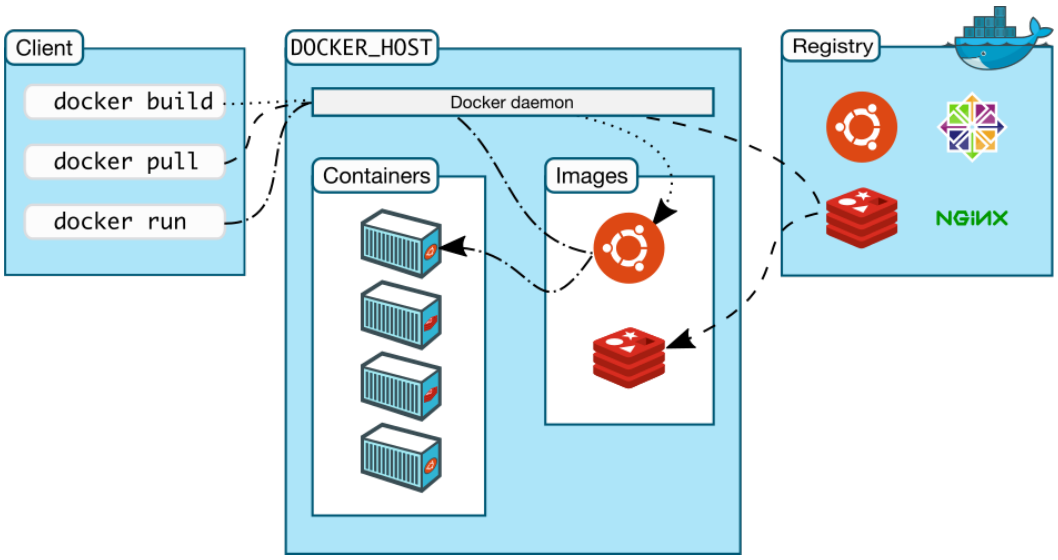


Figura 2.1 – Arquitetura Docker [6].

2.3 Open vSwitch

O OVS (Open vSwitch) é um switch virtual multicamada de código aberto. É projetado para permitir automação de rede, enquanto ainda suporta interfaces e protocolos de gerenciamento padrão — NetFlow, sFlow, LACP, 802.1ag, etc [8].

O OVS possui dois componentes que direcionam o encaminhamento de pacotes: ovs-vswitchd, um daemon; e um módulo kernel de "datapath", como mostrado na figura 2.2. Por exemplo, o kernel recebe um pacote da NIC (Network Interface Card). Ou o ovs-vswitchd já instruiu o módulo sobre o que fazer com o pacote, ou não. Se já tiver instruído, o módulo simplesmente segue as instruções, chamadas de ações, dadas pelo ovs-vswitchd. Caso contrário, entrega o pacote ao daemon. Então, o daemon decide o que fazer com o pacote, depois o devolve ao módulo kernel com as instruções [9].

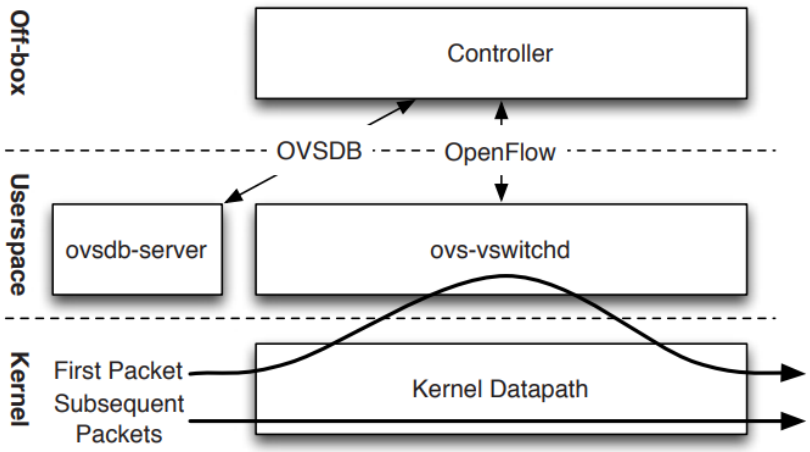


Figura 2.2 – Componentes do Open vSwitch [9].

Além do daemon, o OVS possui o ovsdb-server, que é o banco de dados de configuração. Para acessá-lo, o controlador SDN se conecta por meio do protocolo OVSDB (Open vSwitch Database) [9].

O Open vSwitch também é usado como um switch SDN. A principal forma de controlar encaminhamento é pelo OpenFlow [10]. O OpenFlow permite que o controlador adicione, remova, atualize, monitore e obtenha estatísticas de "flow tables" e dos "flows". Na prática, o ovs-vswitchd recebe as "flow tables" do controlador SDN, faz a correspondência entre os pacotes recebidos e as "flow tables", junta as ações e faz o cache do resultado no módulo kernel [9].

2.4 SDN: Software-Defined Networking

Software-Defined Networking é uma abordagem que utiliza controladores "software-based" ou APIs para realizar a comunicação com a infraestrutura de hardware e direcionar o tráfego na rede. SDN pode criar e controlar redes virtuais ou físicas por meio de software [11].

As redes SDN possuem algumas vantagens [11].

- Maior controle e flexibilidade: em vez de configurar hardwares específicos, é possível fazer o controle de fluxo de tráfego na rede pelo controlador "software-based".
- Infraestrutura de rede personalizável: é possível configurar serviços e alocar recursos virtuais para mudar a infraestrutura da rede em tempo real. Isso permite otimização do fluxo de dados pela rede.
- Segurança Robusta: administradores podem criar zonas separadas para dispositivos que requerem diferentes níveis de segurança, ou colocar dispositivos comprometidos em quarentena para não afetar o resto da rede.

2.4.1 Arquitetura SDN

Conforme é mostrado na figura 2.3, a arquitetura SDN contém seis componentes principais [12].

1. Camada de aplicações: gerenciam e implementam a lógica de controle da rede SDN [2].

2. Camada de controle: possui um ou múltiplos controladores, que direcionam regras, políticas e entradas de fluxo à camada de infraestrutura pela interface "southbound" [2].

3. Camada de dados (ou camada de infraestrutura): representa os dispositivos de encaminhamento, como roteadores, switches, etc. Utiliza APIs "southbound" para interagir com a camada de controle.

4. Interface "northbound": permite que haja comunicação entre a camada de controle e a camada de aplicação. Geralmente, são APIs "open source".

5. Interfaces leste-oeste: admite a comunicação entre vários controladores. Usam protocolo de roteamento, como BGP (Border Gateway Protocol) e OSPF (Open Shortest Path First).

6. Interfaces "southbound": permite comunicação entre camada de controle e camada de dados. Também instala as regras de fluxo nas tabelas de encaminhamento estabelecidas pelo controlador [2]. O OpenFlow é a API "southbound" mais utilizada.

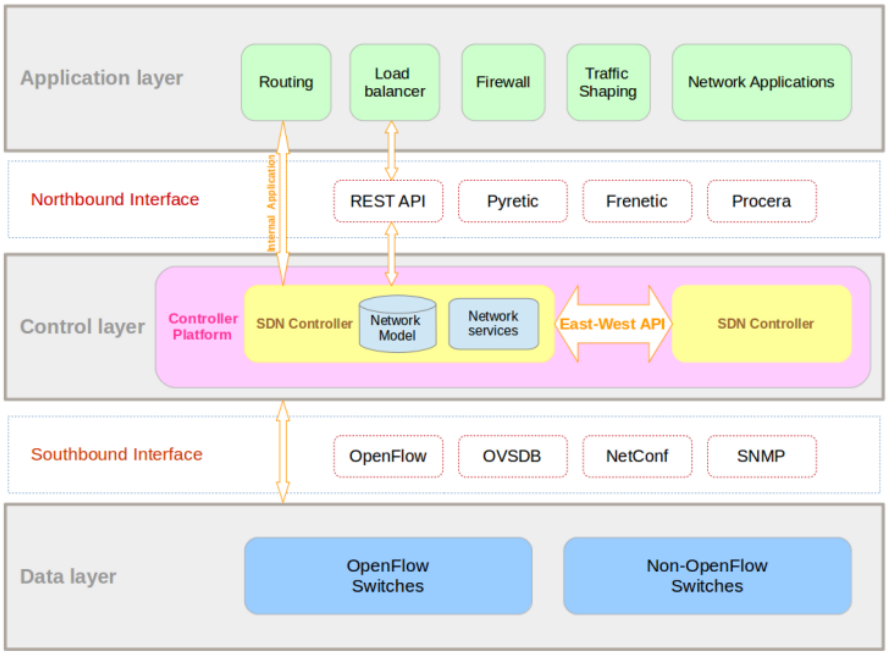


Figura 2.3 – Arquitetura SDN [13].

2.5 OpenFlow

O protocolo OF (OpenFlow) permite a implementação do conceito de SDN. Utiliza códigos "open source" para se comunicar com controladores SDN e switches. Ele permite o gerenciamento do encaminhamento de pacotes, por meio de um agente externo e o controle dos nós da rede a partir de um ponto central, o controlador [14]. O OF popularizou a ideia de separar o plano de controle do plano de dados da rede. O plano de controle gera a tabela de roteamento; e o plano de dados utiliza a tabela para determinar para qual porta encaminhar o pacote [15].

É um protocolo orientado por fluxo. Toda decisão de encaminhamento do OpenFlow se dá por parte das tabelas dos fluxos. Um fluxo é constituído pelos cabeçalhos do pacote, pode ser os campos da camada de enlace, rede ou transporte [16].

Quando um pacote chega num dispositivo com OpenFlow, os cabeçalhos são analisados e comparados com as entradas da "flow table". Se houver correspondência, ações são realizadas e os contadores são atualizados. Se houver mais de uma correspondência, a entrada com maior prioridade é escolhida. Mas, caso não haja, o pacote é enviado ao controlador [16].

Há duas maneiras de o controlador OpenFlow se comunicar com os switches: out-of-band e in-band. Na primeira, o controlador usa um caminho que não é usado pelo OF para comunicação com os switches. Na outra, o controlador usa o mesmo caminho que o OF [16].

Por último, vale ressaltar a vantagem de poder realizar balanceamento de carga. O controlador OF pode monitorar a rede e acomodar o tráfego por meio de alterações nas tabelas de fluxo.

2.5.1 Arquitetura OpenFlow

A arquitetura OpenFlow inclui alguns componentes [16].

1. Controlador: componente responsável pelas regras e ações que gerenciam o encaminhamento de pacotes. Faz a atualização das entradas nas "flow tables" dos switches. Abstrai a configuração.

2. "Flow Entry": representa um tipo de pacote que pode chegar no switch. As regras incluem uma ação para cada fluxo. As três ações mais comuns são enviar o fluxo para uma porta; encapsular e enviar ao controlador; e descartar os pacotes. E a configuração dos fluxos é baseada nos metadados e campos de cabeçalho dos pacotes.

3. "Flow Table": a entrada da tabela que define qual pacote será tratado. Esses campos guardam informações como a porta de entrada do fluxo, IP (Internet Protocol) de origem. E as ações são definidas nos cabeçalhos das tabelas de fluxo. Dados como pacotes transmitidos e tempo de transmissão são salvos pelos contadores como forma de obter dados estatísticos.

4. Switch: possui as "flow tables", que são gerenciadas pelo OF. Um switch OF deve possuir: "flow table", canal de comunicação e o próprio protocolo OpenFlow.

5. Canal Seguro: controlador se comunica com os switches por canais seguros. A conexão é do tipo TCP (Transmission Control Protocol), com possibilidade de se usar TLS (Transport Layer Security) ou SSL (Secure Sockets Layer). É por meio do canal seguro que o controlador distribui as regras de encaminhamento.

2.6 OpenDaylight

O ODL (OpenDaylight) é uma plataforma de código aberto feita para SDN. Utiliza protocolos "abertos" para prover monitoramento de dispositivos de rede e controle centralizado e programático. Escrito em Java, ele utiliza o Maven como ferramenta de "build" [17].

O ODL possui algumas vantagens em questão de segurança. Uma é que a separação do plano de controle e gerenciamento do plano de dados faz com que ameaças possuam uma superfície de ataque menor. A outra é que o controle central fornece melhor visão e controle da rede ao administrador, permitindo decisões rápidas. No entanto, é vantajoso apenas se o acesso ao controle é seguro [17].

2.6.1 Arquitetura OpenDaylight

A arquitetura do ODL é dividida em três camadas: camada de aplicações de rede e serviços; de controle, na qual as abstrações do SDN são encontradas; e a camada de interfaces "southbound" e plugins [18].

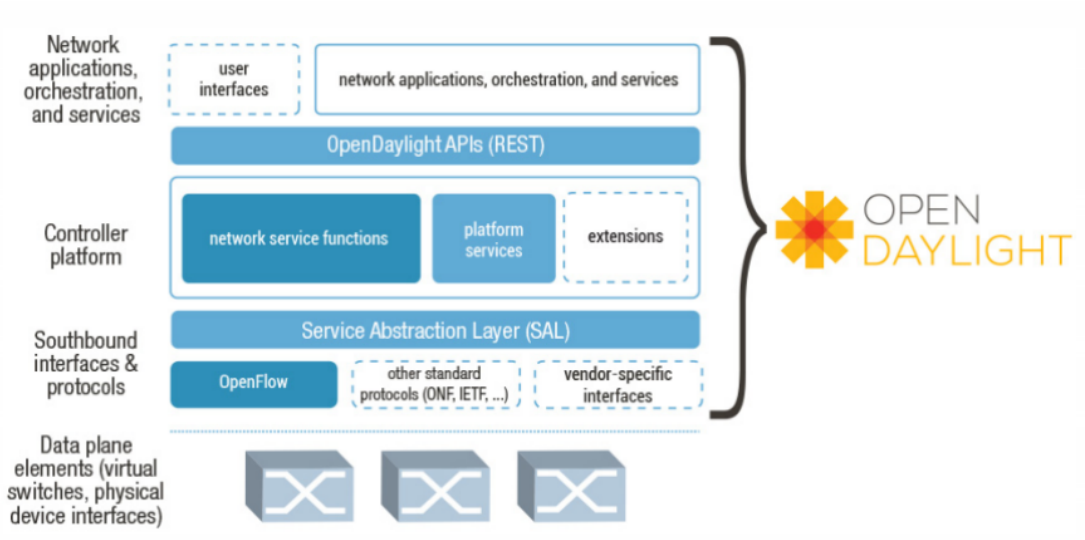


Figura 2.4 – Arquitetura do OpenDaylight [18].

A camada de aplicações de rede é a camada superior da arquitetura. Nela, há serviços e aplicações que controlam e monitoram o controlador e as diversas características da rede. Algumas aplicações são: SDNi Wrapper, que faz coleta e transporte de informações entre as camadas; DLUX, que busca facilitar a interação do usuário com o controlador [18].

A camada central, a de controle, viabiliza a abstração de SDN. Inclui o BNSFs (Serviços Básicos de Rede), a SAL (Service Layer Abstraction). Os BNSFs do ODL são responsáveis pelo monitoramento de topologia, switches, estatísticas e hosts. A SAL possui um gerenciador de plugins que permite a comunicação com diversos protocolos de rede. A SAL evoluiu, junto com o ODL, para MD-SAL (Model-Driven SAL). Esta nova arquitetura permite que múltiplos controladores se juntem num cluster em que cada controlador gerencia uma rede de serviços [19]. Além disso, a camada central possui APIs "Northbound", que fazem a comunicação entre controladores SDN e aplicações na rede [18].

Os protocolos "southbound" são encarregados de realizar a comunicação segura e eficiente entre o controlador e os elementos presentes na rede. O ODL possui suporte a esses protocolos por meio de plugins, por exemplo, o plugin OpenFlow [18].

2.7 OpenFlow Manager

O OFM (OpenFlow Manager) é uma aplicação desenvolvida para rodar "por cima" do ODL, como pode ser visto na figura 2.5. Possui as funções de mostrar topologias, programar caminhos e juntar estatísticas do OpenFlow.

O controlador OpenDaylight fica entre a aplicação e a rede. Interage com elementos de rede na direção "southbound" usando uma variedade de protocolos. Na direção "northbound", apresenta uma abstração da rede por meio de REST APIs. O OpenFlow Manager aproveita essas tecnologias para gerenciar a rede OpenFlow [20].

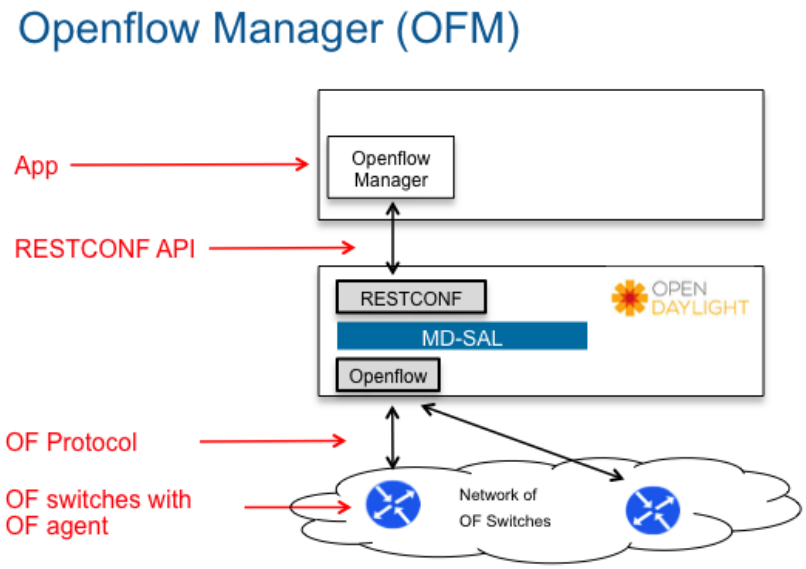


Figura 2.5 – Arquitetura do OpenFlow Manager [20].

Na camada inferior da figura 2.5, há os switches OpenFlow, que realizam operações de comutação com base no "Match/Action". Switches OF suportam agentes OF e usam MPLS (Multi-Protocol Label Switching) para rotular pacotes através da rede. Utilizam OSPF ou IS-IS para manter e distribuir a topologia entre os nós da rede. Um dos roteadores usa BGP-LS para mandar uma cópia da topologia ao controlador ODL [20].

O MD-SAL, na camada intermediária, gera um conjunto de REST APIs (chamados de RESTCONF), que informa quais aplicações podem "chamar". OFM é a aplicação que aciona as APIs RESTCONF para recolher dados de switches OF, programar fluxos e coletar estatísticas [20].

O OFM, camada superior da arquitetura, é acessado por meio de um navegador, como Google Chrome, Mozilla Firefox, etc. As principais funções do OFM são [20]:

1. Basic View: apresenta a topologia dos dispositivos OF na rede.

2. Flow Management: pode ser usado para escolher o número de fluxos para cada dispositivo OF na rede, mostrar a lista de fluxos configurados com as opções de adicionar, modificar e deletar fluxos.

3. Statistics: fornece estatísticas dos fluxos configurados na rede e as portas correspondentes.

4. Hosts: fornece um sumário dos dispositivos OpenFlow que o OFM gerencia.

2.8 Moodle

O ensino remoto é uma área com crescimento significativo, sobretudo desde a pandemia do COVID-19. Com isso, as instituições tiveram que se adaptar devido a todas as restrições. O Moodle se mostrou como uma ferramenta importante para contornar o problema. Prova disso é que número de usuários do Moodle subiu de 78 milhões em 2015 para 294 milhões em 2021 [1].

O Moodle é um sistema de gerenciamento de cursos "open source" usado para criar um ambiente virtual de aprendizagem. Foi criado em 2002 por Martin Dougiamas para auxiliar no ensino on-line. É escrito em PHP e pode ser rodado em diversos servidores web, incluindo Apache. Além disso, suporta vários bancos de dados, como MySQL, PostgreSQL, Oracle, SQLite [21].

Moodle significa Modular Object-Oriented Dynamic Learning Environment, que explica a estrutura de plug-in, pois é totalmente modular. Plug-ins são códigos que podem ser colocados no Moodle. Então, ele é detectado, instalado e disponibilizado como uma ferramenta na interface do Moodle [22]. Os plug-ins, ou módulos, são divididos em seis [23]:

1. Módulos de comunicação: "backbone" de toda comunicação interna e externa. Inclui troca de arquivos, fóruns de discussão e chats.

2. Módulos de produtividade: inclui os módulos de busca, calendário, progresso.

3. Módulo de envolvimento do aluno: inclui portfólio do aluno e autoavaliação.

4. Módulos de administração: módulo que possui acesso a todos os outros. Inclui autenticação, autorização de usuário.

5. Módulo de entrega do curso: ferramentas de avaliação on-line, módulo de gerenciamento de curso e acompanhamento de alunos.

6. Módulo de design de currículo: inclui módulos de personalização e modelos de curso.

Para instalar o Moodle, há duas opções: fazer o download do site oficial ou fazer o "pull" do código no repositório do github [22].

2.9 Wireshark

O Wireshark é uma ferramenta usada para capturar e analisar tráfego na rede. O tráfego é apresentado na forma de pacotes individuais, com o maior detalhamento possível [24].

O software captura os dados de uma interface de rede e apresenta no contexto de endereçamento, protocolos e dados [25]. Por exemplo, o GUI, mostrado na figura 2.6, apresenta três painéis: Packet List, que lista todos os pacotes capturados; Packet Details, que mostra informações do pacote selecionado do Packet List; e Packet Bytes, que apresenta os dados brutos do pacote [25].

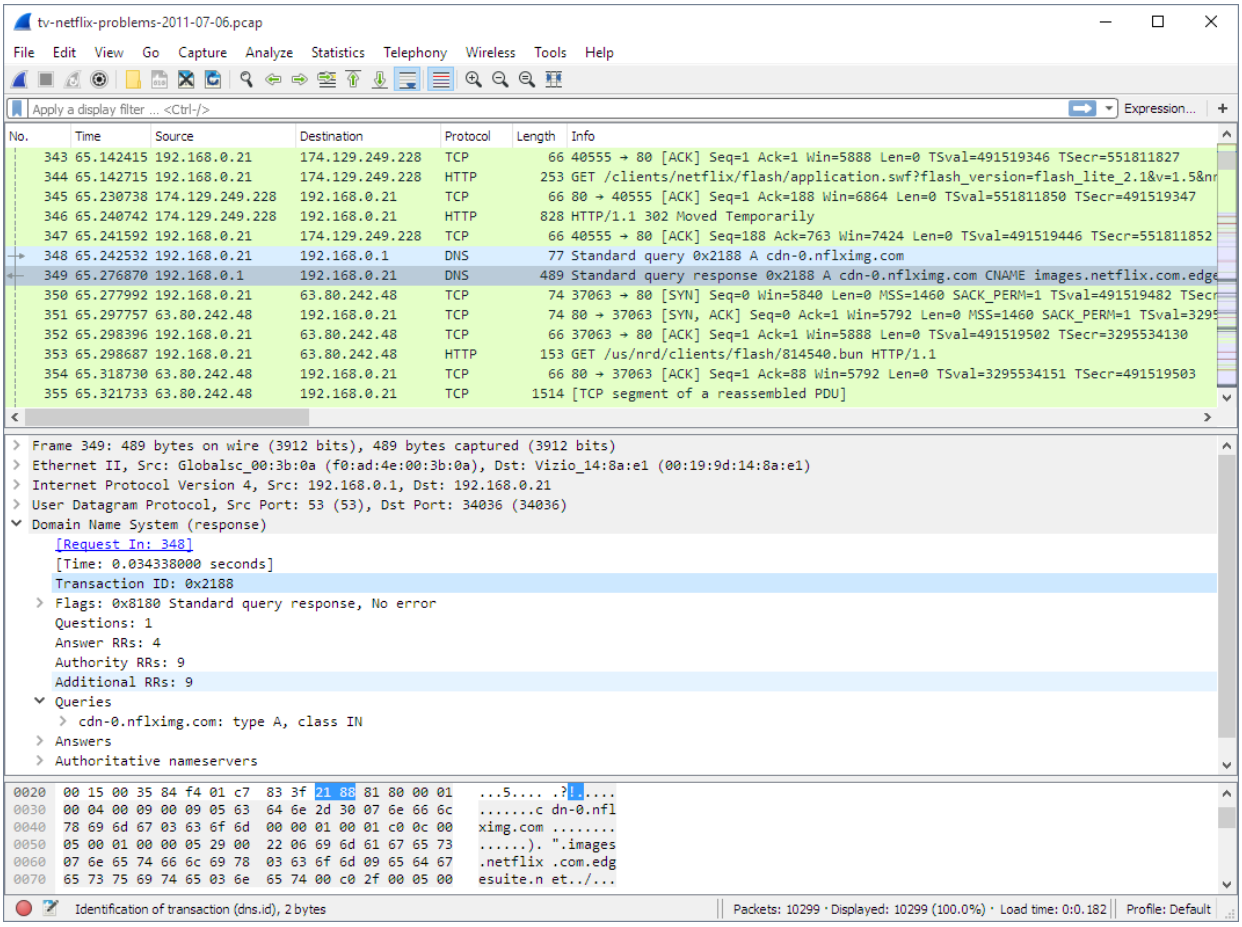


Figura 2.6 – GUI do Wireshark [24].

A capacidade de analisar pacotes no Wireshark se deve aos dissecadores (dissectors). Os dissecadores fazem "parse" de um protocolo e o decodificam para apresentá-lo na GUI. Eles permitem que o Wireshark apresente um contexto para o streaming de bytes transmitidos [25].

Uma ferramenta bastante usada é o filtro. A engine de filtragem do Wireshark permite restringir os pacotes capturados de modo que os fluxos de comunicação fiquem evidentes. Há dois tipos de filtro: filtros de exibição e filtros de captura. Os filtros de exibição dizem respeito apenas ao que se vê na lista de pacotes. Os filtros de captura capturam os pacotes correspondentes e descartam os outros [25]. Geralmente, o filtro de captura é usado para limitar o volume de dados e o filtro de exibição para visualizar apenas os os pacotes desejados [25].

Outra ferramenta importante é o Flow Graph. Ele mostra horário dos pacotes, direção, portas e outros dados para cada conexão capturada entre hosts. As conexões podem ser filtradas por fluxos ICMP (Internet Control Message Protocol), TCP, etc [24].

Uma vantagem do Wireshark é capacidade de salvar tráfego num arquivo (que pode ser compartilhado). Os formatos dos arquivos gerados são pcap ou pcapng [24].

2.10 Trabalhos Relacionados

No trabalho [3], o autor cria uma arquitetura de um portal seguro de provas eletrônicas utilizando SDN. O enfoque do trabalho é voltado para a criação do software de provas com vários recursos: opção de agendamento de provas; uso de QR Code para validar o agendamento; uso de VPN (Virtual Private Network) para realização de provas remotas; e outras funcionalidades. A parte de SDN não possui muito destaque. Por isso, este trabalho propõe explorar mais a parte de SDN a fim de criar um ambiente de provas seguro e com um sistema prático de administração de forma centralizada. Porém, utilizando o Moodle como software de provas, em vez do software criado em [3].

Este trabalho também está relacionado com [26], em que os autores criam uma aplicação web, o MoodleWatcher, para coletar as ações dos usuários registradas pelo Moodle. O MoodleWatcher agrupa os "logs" para permitir identificar quais ações e por quem foram realizadas em períodos de prova. Para isso, é apresentado um painel com uma lista de ações suspeitas ao professor ou responsável pela aplicação da prova. Então, as fraudes podem analisadas após a realização das avaliações. Já a proposta SDN deste trabalho visa evitar as ações fraudulentas antes que elas aconteçam.

O trabalho [27], similar ao [26], em que os autores propõem o MoodleGate. Nessa proposta, cria-se Listas de Controle de Acesso (ACLs) para determinar quem, quando e o que pode ser feito no ambiente Moodle. Para isso, um firewall consulta as ACLs para cada requisição feita pelos usuários e decide qual ação deve ser tomada. A interface do MoodleGate se integra com os exames do Moodle para simplificar a administração de ACLs para as provas. Já nesta proposta de SDN, a parte de controle das ações dos usuários fica para as regras criadas no controlador SDN.

Outro trabalho relacionado é o [28]. Nele, os autores tentam aprimorar a alocação de largura de banda para o Moodle. Tem-se a meta de melhorar a qualidade de serviço. No entanto, neste projeto, há um enfoque no controle da largura de banda de cada usuário utilizando o Moodle. Já neste trabalho, o foco é voltado para criação de um ambiente de provas que dificulte fraudes em horários de prova.

3 Arquitetura do Projeto

Este capítulo descreve o processo de construção do projeto de maneira geral. Logo após, a partir da seção 3.2, faz um detalhamento das configurações de cada dispositivo.

3.1 Apresentação da topologia

A figura 3.1 apresenta quatro regiões interconectadas pelo switch central: sala de provas, administração, servidor de prova e cloud. A sala de provas possui quatro computadores firefox ligados ao switch central. Cada um deles é uma imagem Docker de um computador com apenas funções necessárias para rodar os testes do projeto.

A administração também é uma imagem Docker. Tem a função de simular o controlador OpenDaylight junto com o OpenFlow Manager. O detalhamento da criação desse dispositivo está explicado na subseção 3.4.

O servidor de provas é uma imagem Docker com o software Moodle incluso. A criação desse dispositivo está explicada na subseção 3.5.

A figura da nuvem na figura 3.1 representa o acesso à internet pública. Além disso, possui a funcionalidade de incluir os dispositivos conectados na mesma rede local em que o GNS3 está rodando.

Por último, tem-se, no centro, o switch Open vSwitch, que é responsável pela aplicação das regras SDN. A subseção 3.6 possui um detalhamento da sua criação.

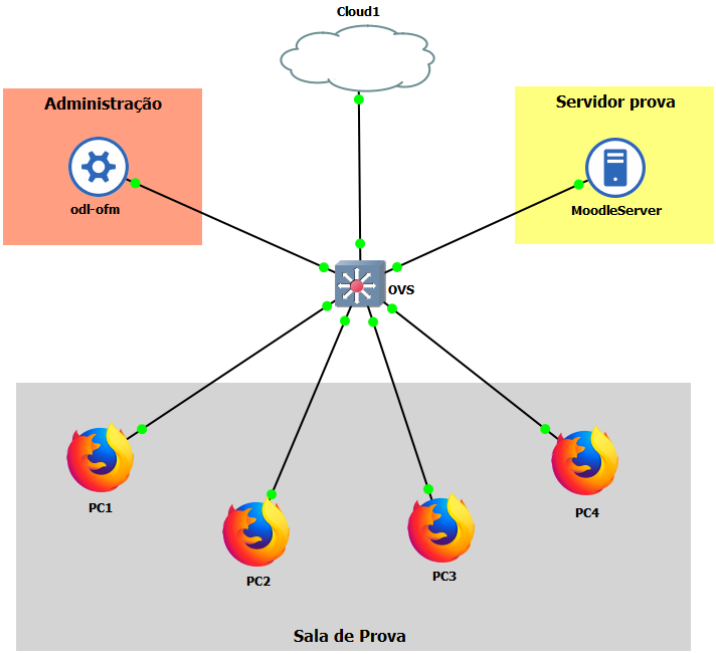


Figura 3.1 – Topologia construída no GNS3.

Como dito anteriormente, optou-se por inserir a simulação na rede física, com o objetivo de tornar os dispositivos acessíveis por qualquer aparelho da rede. Mesmo assim, decidiu-se pela não utilização de DHCP (Dynamic Host Configuration Protocol) para distribuição dos endereços IPs, para não ocorrerem conflitos na rede e facilitar o gerenciamento. Dito isso, os IPs definidos são apresentados na tabela 3.1.

Tabela 3.1 – Descrição dos endereços IP definidos

Dispositivo	Endereço IP	Máscara de Sub-rede	Gateway
Controlador	192.168.1.39	255.255.255.0	192.168.1.254
OVS eth 1	192.168.1.40	255.255.255.0	192.168.1.254
OVS eth 0	192.168.1.41	255.255.255.0	192.168.1.254
Moodle Server	192.168.1.100	255.255.255.0	192.168.1.254
PC1	192.168.1.49	255.255.255.0	192.168.1.254
PC2	192.168.1.50	255.255.255.0	192.168.1.254
PC3	192.168.1.51	255.255.255.0	192.168.1.254
PC4	192.168.1.52	255.255.255.0	192.168.1.254

O endereço IP 192.168.1.254 pertence ao roteador presente na rede física.

3.2 Configuração do GNS3

Após a criação das imagens Docker (explicadas nas subseções 3.4 e 3.5), as máquinas foram importadas para o ambiente virtual do GNS3. O trabalho [29] foi usado como guia para montar a topologia e realizar as configurações.

Como plataforma de virtualização utilizou-se o VMWare Workstation Pro, que apresenta maior compatibilidade com o GNS3 e também melhor desempenho quando comparado com o Oracle Virtual Box, além de oferecer melhor customização das redes virtuais por meio da ferramenta *Virtual Network Editor*, que foi fundamental para a configuração em modo *Bridge*.

Para o GNS3 VM, máquina virtual responsável pela virtualização das redes no GNS3, definiu-se três interfaces: Host Only, Nat e Bridged. Além disso, disponibilizou-se 2 VCPUS e 7 gigabytes de memória RAM. Este alto valor de memória RAM se justifica pela utilização de muitos contêineres Docker, uma vez que os mesmos são executados dentro do próprio GNS3 VM.

3.3 Configuração dos Computadores de Acesso

Os computadores de acesso possuem como base a imagem Webterm, que é uma imagem Docker que fornece uma interface web baseada em terminal para acesso ao shell de um container Docker. Para sua utilização, definiu-se persistência em sua raiz e instalou-se alguns pacotes necessários para a simulação, como ping, curl, wget, nmap, iputils, netcat e netstat, além definir, em seu arquivo de hosts, um mapemento para o endereço IP do servidor Moodle.

3.4 Configuração do Controlador OpenDaylight

Para configurar o controlador ODL e o OFM, é necessário rodar uma série de comandos no prompt de comando de uma máquina Linux. Para facilitar a sua criação e tornar a implantação mais rápida e prática, criou-se uma Dockerfile de geração da imagem composta tanto pelo ODL quanto pelo OFM.

O Dockerfile de criação da imagem docker contendo OpenDaylight e OFM é apresentado abaixo, sendo que um pequeno tutorial de sua execução é apresentado no Anexo A:

```
1
2  FROM ubuntu
3
4  # instalacao de pacotes necessarios
5  RUN apt update \
6      && DEBIAN_FRONTEND=noninteractive apt install -y --no-install-recommends \
7          bash-completion \
8          software-properties-common \
9          sudo \
10         curl \
11         ssh \
12         git \
13         vim \
14         unzip \
15         wget
16
17 # permitir login ssh sem senha
18 RUN sed -i 's/#PermitRootLogin prohibit-password/PermitRootLogin
19     prohibit-password/g' /etc/ssh/sshd_config
20
21 # Configuracao java 8
22 RUN apt update \
23     && add-apt-repository ppa:openjdk-r/ppa -y \
24     && apt update \
25     && apt install openjdk-8-jdk -y
26
27 # Instalacao JAVA_HOME
28 ENV JAVA_HOME=/usr/lib/jvm/java-8-openjdk-amd64 \
29     PATH="$JAVA_HOME:$PATH"
30
31 # instalacao OpenDaylight e OFM
32 RUN wget https://nexus.opendaylight.org/content/groups/public/org/opendaylight/
33         integration/distribution-karaf/0.3.0-Lithium/distribution-karaf-0.3.0-
34         Lithium.tar.gz \
35     && tar -xvzf distribution-karaf-0.3.0-Lithium.tar.gz \
36     && add-apt-repository ppa:chris-lea/node.js -y \
37     && apt update \
38     && apt install nodejs -y \
39     && git clone https://github.com/CiscoDevNet/OpenDayLight-OpenFlow-App.git \
40     && apt install npm  -y \
41     && npm install -g grunt-cli \
42     && rm - rf /var/lib/apt/lists/*
```

```
43
44  # Definicao de variaveis de ambiente
45  ENV ODL_HOME=/root/distribution-karaf-0.3.0-Lithium \
46      OFM_HOME=/root/ofm
47
48  EXPOSE 8181 6633 6653 6640 8101 9000
```

Utilizando como base uma imagem Ubuntu, primeiramente atualiza-se o repositório de dependências e instala-se todos os pacotes necessários, sendo eles: sudo, curl, ssh, git, vim, unzip e wget.

Feito isso, configura-se o SSH (Secure Shell) para permitir login sem senha, para facilitar a configuração e acesso. Isso é feito inserindo a linha "PermitRootLogin prohibit - password" no arquivo de configuração do SSH.

Depois, instala-se o java JDK em sua versão 8 e configura-se a variável de ambiente JAVA_HOME, que é uma variável de ambiente que aponta para o diretório de instalação do Java no sistema. Ela é usada por várias ferramentas para localizar o Java instalado no sistema, sendo fundamental para garantir que as ferramentas Java funcionem corretamente.

Configurado o Java, pode-se enfim realizar a instalação do OpenDaylight e do OFM. O ODL é instalado simplesmente fazendo o dowload de seu arquivo no próprio site do OpenDaylight, enquanto o OFM é instalado clonando o seu repositório do Github e instalando-se o nodejs, o npm e o grunt-cli.

Por fim, são definidas variáveis de ambiente para facilitar a locomoção no terminal e são apontadas as portas a serem expostas, que são: 8181, 6633, 6653, 6640, 8101 e 9000, sendo as cinco primeiras para utilização do ODL, enquanto a última para utilização do OFM.

3.5 Configuração do Servidor de provas

Assim como no caso do ODL e OFM, optou-se pela criação de uma imagem Docker para o servidor de provas Moodle [30], com o objetivo de facilitar a implantação. No caso, há três serviços utilizando o mesmo contêiner, que são o MySQL, Apache2 e o próprio Moodle.

O Dockerfile de criação da imagem docker contendo a instalação do Moodle é apresentado abaixo, sendo que no Anexo B é apresentado um exemplo de sua execução:

```
1
2    FROM ubuntu
3
4    ARG MOODLE_HOME=/var/www/html/moodle
5
6    # instalacao de pacotes necessarios
7    RUN apt update \
8        && DEBIAN_FRONTEND=noninteractive apt install -y --no-install-recommends
    \
9            bash-completion \
10           software-properties-common \
```

```
            git \
            vim \
            unzip \
            wget \
            apache2 \
            cron \

    # Instalacao PHP
    RUN apt update \
        && add-apt-repository ppa:ondrej/php -y \
        && apt update \
        && apt-get install -y php7.4 php7.4-cli php7.4-json php7.4-common \
            php7.4-mysql php7.4-zip php7.4-gd php7.4-mbstring php7.4-curl \
            php7.4-xml php7.4-bcmath php7.4-intl php7.4-xmlrpc php7.4-soap

    # Copia script que cria tabelas utilizadas pelo moodle para dentro da imagem
    COPY --chown=root:root mysql-moodle.sql /tmp

    # Instalacao MySQL
    RUN \
        wget https://dev.mysql.com/get/mysql-apt-config_0.8.12-1_all.deb \
        && dpkg -i mysql-apt-config_0.8.12-1_all.deb \
        && apt-key adv --keyserver keyserver.ubuntu.com --recv-keys 467
    B942D3A79BD29 \
        && apt update \
        && apt-cache policy mysql-server \
        && apt install -y -f mysql-client=5.7* mysql-community-server=5.7* mysql-
    server=5.7* \
        && mysql -u root < '/tmp/mysql-moodle.sql'; \
        && echo "service mysql start" >> /root/.bashrc \
        && echo "service apache2 start" >> /root/.bashrc

    # Instalacao Moodle
    RUN \
        git clone -b MOODLE_310_STABLE git://git.moodle.org/moodle.git \
        && mv /root/moodle /var/www/html/ \
        && mkdir -p /home/moodle/files \
        && chmod 777 -R /home/moodle/files \
        && rm -rf /var/lib/apt/lists/*

    # Configuracao do Crontab
    RUN \
        echo "* * * * * /usr/bin/php  /var/www/html/moodle/admin/cli/cron.php >/
    dev/null" >> /var/spool/cron/crontabs/root

    # Copia arquivo de configuracao do moodle para dentro da imagem.
    COPY --chown=root:root config.php ${MOODLE_HOME}/

    # Definicao de variaveis de ambiente
    ENV MOODLE_HOME=${MOODLE_HOME}

    EXPOSE 80 8080 443
```

Novamente, a imagem base escolhida é o Ubuntu em sua versão mais atual. Inicia-se a instalação definindo uma variável cujo valor é o local de instalação do Moodle, para evitar repetições e simplificar o código.

Em seguida, atualiza-se o repositório de pacotes e instala-se os pacotes a serem utilizados, sendo eles: bash-completition, software-properties-common, git, vim, unzip, wget, apache2 e cron.

Feito isso, instala-se o PHP 7.4, que é um dos pré-requisitos para o funcionamento do Moodle. No caso, além do próprio PHP, são instalados diversos pacotes necessários para a utilização do moodle.

Instalado o PHP, copia-se para dentro do contêiner o script MySQL a ser executado para criação das tabelas e usuários a serem utilizados pelo Moodle, seguido da instalação do próprio MySQL, cuja versão escolhida é a 5.7.

Finalizada esta instalação, configura-se o MySQL e o Apache2 para inicializarem assim que a máquina for ligada, por meio da edição do arquivo *bashrc* do usuário *root*.

Instalados todos os pré-requisitos, o Moodle pode enfim ser instalado. Isso é feito clonando seu repositório do Github, movendo-o para diretório /var/www/html para o tornar acessível e criando-se um diretório de arquivos que deve ser acessível para o usuário do serviço. Vale destacar que, para o Moodle ser executado corretamente, é necessário ter o crontab pré-instalado no sistema, para que ele possa executar algumas de suas tarefas regularmente, em períodos agendados.

Por fim, configura-se o crontab para executar, a cada minuto, o *script cron.php*; copia-se o arquivo de configuração do Moodle - responsável por definir urls, portas e diretórios - para dentro do container; define-se variáveis de ambiente; e se expõe as portas 80, 8080 e 443.

3.6 Configuração do Open vSwitch

A configuração do Open vSwitch é simples, pois os passos estão descritos no trabalho [29]. Basicamente, definiu-se os endereços IPs para as interfaces eth0 e eth1 - são as interfaces que ligam, respectivamente, o OVS ao controlador e o OVS à rede externa - e, por meio dos comandos apresentados abaixo, configurou-se a conexão.

```
1  ---
2  ovs-vsctl add-port br0 eth0
3  ovs-vsctl set bridge br0 protocols=OpenFlow13
4  ovs-vsctl set-controller br0 tcp:192.168.1.39:6633
5  ---
```

O primeiro comando é utilizado para adicionar uma porta virtual a um switch virtual, que no caso é o OVS. Ele permite que dispositivos virtuais, como máquinas virtuais, sejam conectados ao switch virtual e possam trocar pacotes de rede. No caso, adicionou-se a interface eth0 à br0.

O segundo comando é usado para configurar propriedades de um bridge (switch) virtual no Open vSwitch. Ele permite alterar configurações como o nome da bridge, a configuração de firewall, as políticas de fluxo, entre outras configurações. Essa capacidade de configuração permite que o

administrador da rede personalize a operação do switch virtual de acordo com as necessidades da rede. No caso apresentado, simplesmente definiu-se a utilização do protocolo OpenFlow13 para a *bridge* br0.

Por fim, o terceiro comando é usado para configurar o controlador para um switch virtual no Open vSwitch. O controlador é responsável por controlar o comportamento do switch virtual, incluindo a instalação de regras de fluxo e a garantia de que os pacotes são roteados de maneira eficiente. O comando "ovs-vsctl set-controller" permite especificar quais controladores estão conectados ao switch virtual, assim como qual é o controlador principal. Dessa forma, é possível garantir que o switch virtual funcione de maneira eficiente e que a rede seja gerenciada de maneira centralizada. Neste caso, definiu-se o dispositivo de endereço IP 192.168.1.39 como controlador da *bridge* br0.

3.7 Regras definidas no OpenDaylight-Openflow-App (OFM)

Foram definidas cinco regras ao controlador para criar o ambiente seguro, como descrito anteriormente. Cada uma está descrita nas próximas subseções, tomando como base o apresentado em [31].

3.7.1 Primeira Regra

A primeira regra definida tem o intuito de bloquear todo tráfego oriundo da sub-rede do laboratório para rede pública. Como é mostrado na figura 3.2, dez campos são preenchidos para realizar tal configuração. Primeiro, escolhe-se qual dispositivo receberá as alterações. No caso, a imagem mostra que é o Open vSwitch.

Em seguida, escolhe-se em qual tabela o novo fluxo será colocado. Foi escolhida a tabela 0.

Outra configuração necessária é escolher a identificação para o fluxo. O ID 2 foi escolhido, conforme mostra a figura 3.2.

O quarto campo é da prioridade do fluxo. Esse valor estabelece a ordem de preferência das regras de fluxo. Quanto maior o valor, maior a prioridade. Foi escolhido o valor 100 para ser uma das primeiras regras a serem atendidas.

O nome do fluxo também é escolhido para que seja legível para seres humanos. Foi escolhido o nome "Bloquear todo tráfego originado da sub-rede do laboratório" para simplificar o entendimento.

Em seguida, o hard timeout e idle timeout foram simplesmente definidos como 0. Isso significa que a entrada de fluxo é considerada permanente e não expira. Ela pode ser removida com uma mensagem de modificação da tabela de fluxo do tipo OFPFC_DELETE [32].

O oitavo campo define qual tipo de Ethernet é usada. O número 2048 representa a conexão física com cabo Ethernet utilizando o Internet Protocol version 4 (IPv4).

O endereço IPv4 da fonte também deve ser escolhido. No caso deste trabalho, configurou-se os endereços IP dos computadores de prova como parte da sub-rede 192.168.1.48/28. Com isso, define-se como sub-rede de origem desta regra, a sub-rede 192.168.1.48/28.

Por último, escolhe-se qual ação é realizada com os pacotes provenientes desse fluxo. A figura 3.2 mostra que a ação escolhida foi "Drop". Com isso, todos os pacotes recebidos que se encaixarem nessas definições listadas serão descartados.

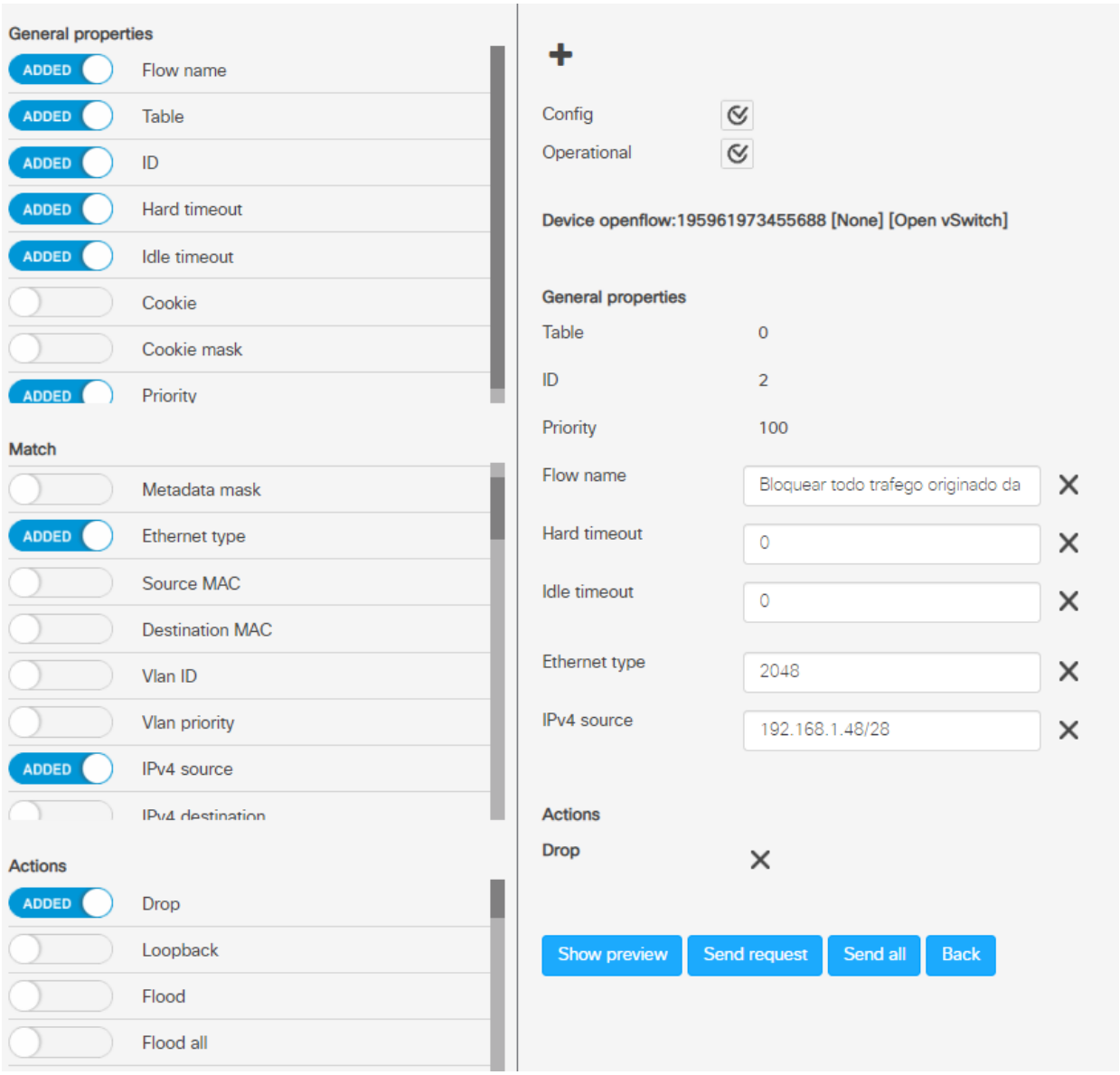


Figura 3.2 – Regra OFM para bloquear todo tráfego oriundo da sub-rede do laboratório.

3.7.2 Segunda Regra

A segunda regra definida tem a finalidade de bloquear todo tráfego entre computadores no laboratório. Como é mostrado na figura 3.3, onze campos são preenchidos para realizar tal configuração. Primeiro, escolhe-se qual dispositivo receberá as alterações. No caso, a imagem mostra que é o Open vSwitch.

Em seguida, escolhe-se em qual tabela o novo fluxo será colocado. Foi escolhido a tabela 0.

Outra configuração necessária é escolher a identificação para o fluxo. O ID 1 foi escolhido, conforme mostra a figura 3.3.

O quarto campo é da prioridade do fluxo. Esse valor estabelece a ordem de preferência das regras de fluxo. Quanto maior o valor, maior a prioridade. Foi escolhido o valor 100 para ser uma das primeiras regras a ser atendidas.

O nome do fluxo também é escolhido para que seja legível para seres humanos. Foi escolhido o nome "Impedir tráfego entre PCs" para simplificar o entendimento.

Em seguida, o hard timeout e idle timeout foram simplesmente definidos como 0. Isso significa que a entrada de fluxo é considerada permanente e não expira. Ela pode ser removida com uma mensagem de modificação da tabela de fluxo do tipo OFPFC_DELETE [32].

O oitavo campo define qual tipo de Ethernet é usada. O número 2048 representa a conexão física com cabo Ethernet, utilizando o Internet Protocol version 4 (IPv4).

O endereço IPv4 da fonte e destino são definidos. O endereço "192.168.1.48/28" engloba todos os computadores presentes na simulação do laboratório. Então, foi designado como fonte e destino. Ou seja, esta regra se aplica a todos os pacotes que partirem da sub-rede do laboratório e que também estão destinadas à mesma sub-rede.

Por último, escolhe-se qual ação é realizada com os pacotes provenientes desse fluxo. A figura 3.3 mostra que a ação escolhida foi "Drop". Com isso, todos os pacotes recebidos que se encaixarem nessas definições listadas serão descartados.

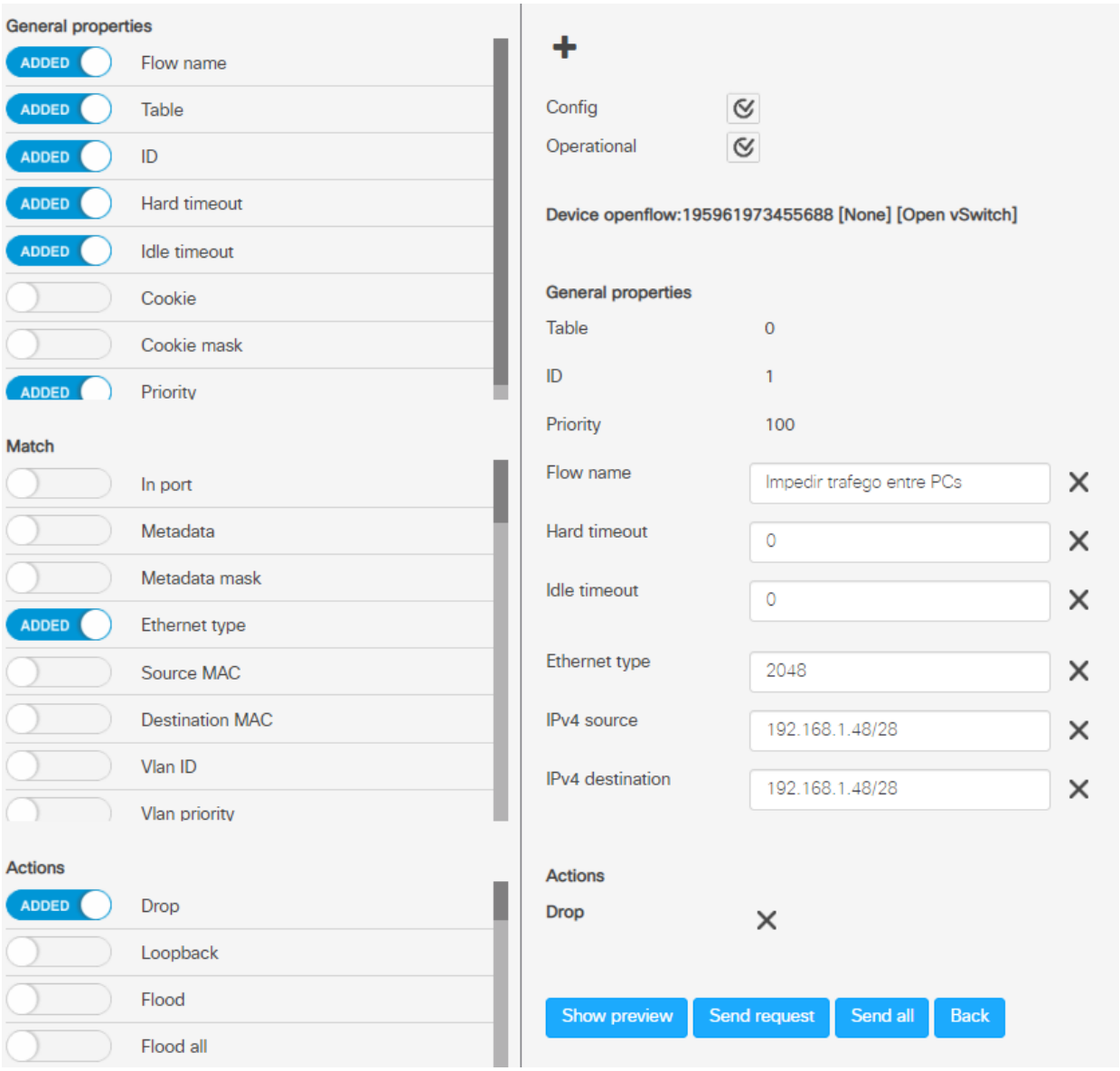


Figura 3.3 – Regra OFM para bloquear todo tráfego entre PCs do laboratório.

3.7.3 Terceira Regra

A terceira regra definida tem a finalidade de permitir apenas tráfego na porta 80 ao servidor Moodle da topologia. Como é mostrado na figura 3.4, quatorze campos são preenchidos para realizar tal configuração. Primeiro, escolhe-se qual dispositivo receberá as alterações. No caso, a imagem mostra que é o Open vSwitch.

Em seguida, escolhe-se em qual tabela o novo fluxo será colocado. Foi escolhida a tabela 0.

Outra configuração necessária é escolher a identificação para o fluxo. O ID 3 foi escolhido, conforme mostra a figura 3.4.

O quarto campo é da prioridade do fluxo. Esse valor estabelece a ordem de preferência das regras de fluxo. Quanto maior o valor, maior a prioridade. Foi escolhido o valor 110 para ser uma das primeiras regras a serem atendidas.

O nome do fluxo também é escolhido para que seja legível para seres humanos. Foi escolhido o nome "Permitir acesso Moodle" para simplificar o entendimento.

Em seguida, o hard timeout e idle timeout foram simplesmente definidos como 0. Isso significa que a entrada de fluxo é considerada permanente e não expira. Ela pode ser removida com uma mensagem de modificação da tabela de fluxo do tipo OFPFC_DELETE [32].

O oitavo campo define qual tipo de Ethernet é usada. O número 2048 representa a conexão física com cabo Ethernet, utilizando o Internet Protocol version 4 (IPv4).

O campo de endereço MAC (Media Access Control) de origem também é preenchido. No caso, escolheu-se o endereço MAC dos computadores do laboratório, presentes na figura 3.1. A definição de endereço MAC de fonte, neste caso, foi pensado para evitar que o usuário conecte qualquer dispositivo na rede e se comunique com o servidor de provas. Com isso, apenas dispositivos com endereços MAC cadastrados poderão ter acesso ao Moodle, o que incrementa a segurança da rede e reforça a ideia de ser uma solução o mais "à prova de colas" possível. Esta regra foi configurada para cada um dos computadores do laboratório, sendo que a regra apresentada na figura 3.4 é aplicada para o PC1.

O endereço IPv4 da fonte e destino são definidos. O endereço "192.168.1.48/28" engloba todos os computadores presentes na simulação do laboratório. E o endereço "192.168.1.100/32" representa o servidor de provas (Moodle). Optou-se por cadastrar os endereços IP de fonte e destino também para limitar ainda mais o acesso ao Moodle, desta forma, esta regra se aplica apenas aos dispositivos presentes na rede "192.168.1.48/28", cujo destino seja o dispositivo de IP 192.168.1.100, que é o servidor de provas.

O décimo segundo campo a ser definido é o IP protocol. Foi atribuido o valor 6, que representa o protocolo TCP. Isto significa que essa regra se aplica apenas a pacotes utilizando tal protocolo.

Então, o campo "TCP destination port" foi definido como 80 para permitir tráfego apenas na porta de destino 80 do protocolo TCP. No caso deste trabalho, como se trata de um ambiente fechado de testes, não se configurou SSL/TLS com o servidor de provas, portanto, toda a comunicação ocorre pela porta 80. Vale destacar que, em um ambiente real de produção, esta comunicação deve ser feita obrigatoriamente por HTTPS (HyperText Transfer Protocol Secure), o que alteraria esta regra para a porta 443 ou então a porta HTTPS de preferência.

Por último, escolhe-se qual ação é realizada com os pacotes provenientes desse fluxo. A figura 3.4 mostra que a ação escolhida foi "Normal". Nesse caso, nenhuma ação é realizada nos pacotes que se encaixam nessa regra, isto é, pacotes que se encaixam nesta regra poderão passar sem nenhum tipo de limitação.

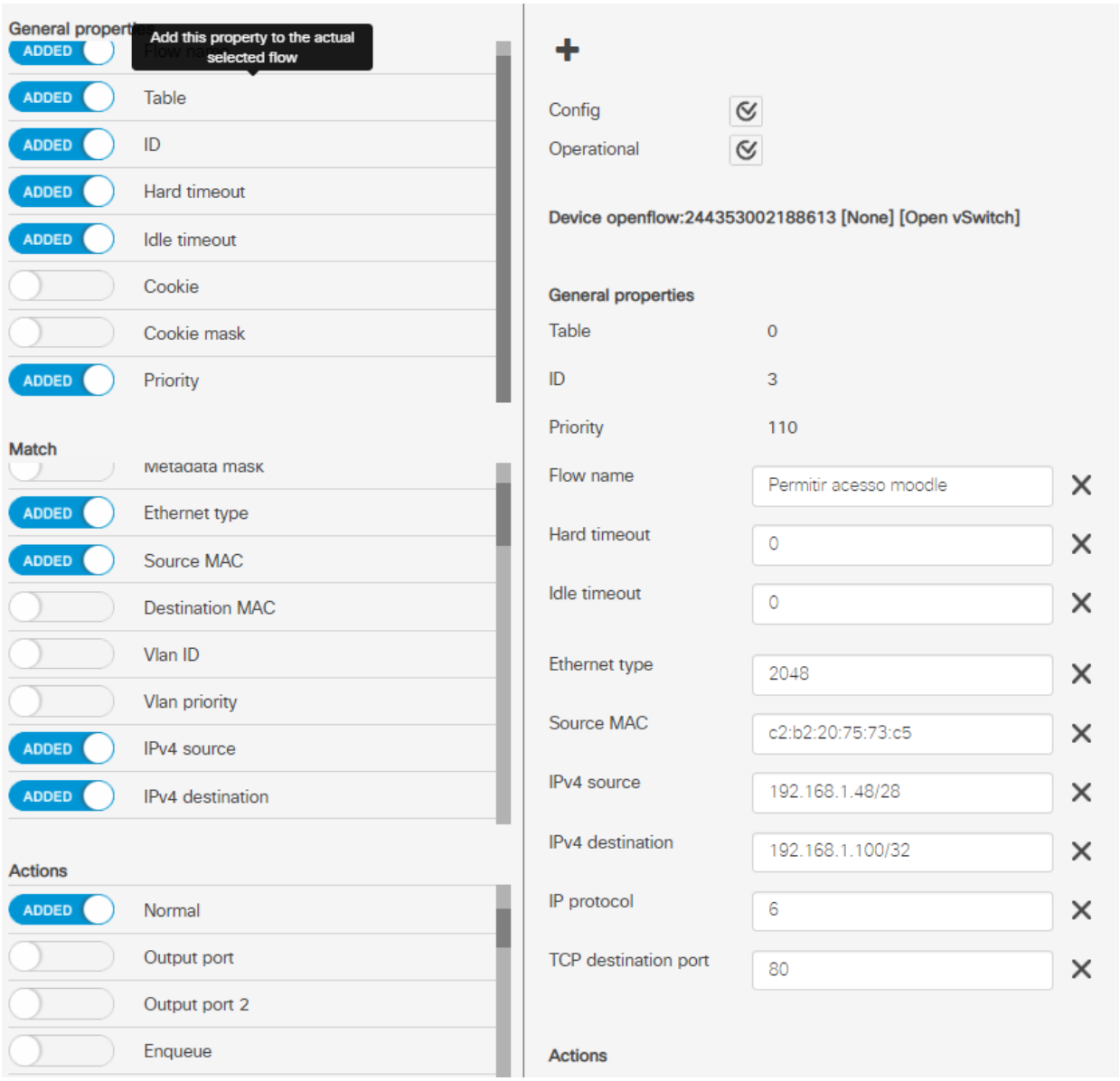


Figura 3.4 – Regra OFM para permitir tráfego na porta 80 ao MoodleServer.

3.7.4 Quarta Regra

A quarta regra definida tem a finalidade de autorizar o acesso ao controlador pelo computador administrador da rede. Como é mostrado na figura 3.5, nove campos são preenchidos para realizar tal configuração. Primeiro, escolhe-se qual dispositivo receberá as alterações. No caso, a imagem mostra que é o Open vSwitch.

Em seguida, escolhe-se em qual tabela o novo fluxo será colocado. Foi escolhida a tabela 0.

Outra configuração necessária é escolher a identificação para o fluxo. O ID 4 foi escolhido, conforme mostra a figura 3.5.

O quarto campo é da prioridade do fluxo. Esse valor estabelece a ordem de preferência das regras de fluxo. Quanto maior o valor, maior a prioridade. Foi escolhido o valor 1000 para ser a

primeira regra atendida. A definição deste alto valor de prioridade é feito para prevenir o caso do administrador da rede "se trancar do lado de fora", isto é, impede que o mesmo crie acidentalmente uma regra que o impeça de ter acesso ao controlador.

O nome do fluxo também é escolhido para que seja legível para seres humanos. Foi escolhido o nome "Permitir administração" para simplificar o entendimento.

Em seguida, o hard timeout e idle timeout foram simplesmente definidos como 0. Isso significa que a entrada de fluxo é considerada permanente e não expira. Ela pode ser removida com uma mensagem de modificação da tabela de fluxo do tipo OFPFC_DELETE [32].

O oitavo campo define o endereço MAC de origem. No caso, escolheu-se o endereço MAC da máquina utilizada pelo administrador da rede para acessar o controlador. Ou seja, apenas o dispositivo cujo endereço MAC for o indicado neste campo terá acesso a essa regra.

Por último, escolhe-se qual ação é realizada com os pacotes provenientes desse fluxo. A figura 3.5 mostra que a ação escolhida foi "Normal". Nesse caso, nenhuma ação é realizada nos pacotes que se encaixam nessa regra, isto é, pacotes que se encaixam nesta regra poderão passar sem nenhum tipo de limitação.

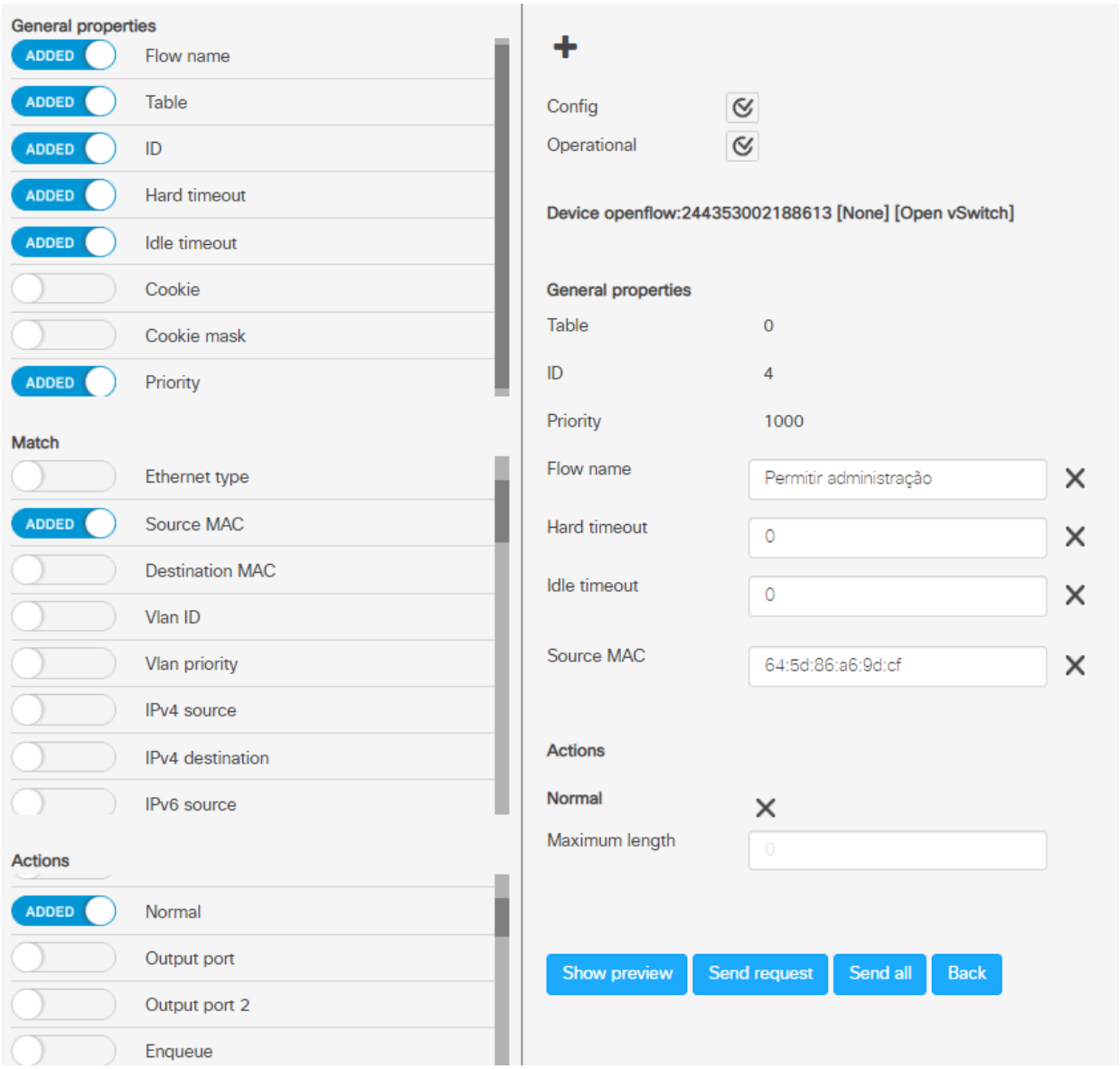


Figura 3.5 – Regra OFM para permitir tráfego originado do PC de administração.

3.7.5 Quinta Regra

A quinta regra definida tem a função de bloquear tráfego externo para o interior do laboratório. Como é mostrado na figura 3.6, sete campos são preenchidos para realizar tal configuração. Primeiro, escolhe-se qual dispositivo receberá as alterações. No caso, a imagem mostra que é o Open vSwitch.

Em seguida, escolhe-se em qual tabela o novo fluxo será colocado. Foi escolhida a tabela 0.

Outra configuração necessária é escolher a identificação para o fluxo. O ID 5 foi escolhido, conforme mostra a figura 3.6.

O quarto campo é da prioridade do fluxo. Esse valor estabelece a ordem de preferência das regras de fluxo. Quanto maior o valor, maior a prioridade. Foi escolhido o valor 100 para ser uma das primeiras regras a serem atendidas.

O nome do fluxo também é escolhido para que seja legível para seres humanos. Foi escolhido o nome "Impedir tráfego externo" para simplificar o entendimento.

Em seguida, o hard timeout e idle timeout foram simplesmente definidos como 0. Isso significa que a entrada de fluxo é considerada permanente e não expira. Ela pode ser removida com uma mensagem de modificação da tabela de fluxo do tipo OFPFC_DELETE [32].

Em seguida, o sexto campo, "In port", seleciona uma porta do dispositivo que receberá a regra. No caso, foi escolhida a porta do Open vSwitch ligada à internet pública.

Por último, escolhe-se qual ação é realizada com os pacotes provenientes desse fluxo. A figura 3.6 mostra que a ação escolhida foi "Drop". Com isso, todos os pacotes recebidos na porta especificada que se encaixarem nas definições serão descartados.

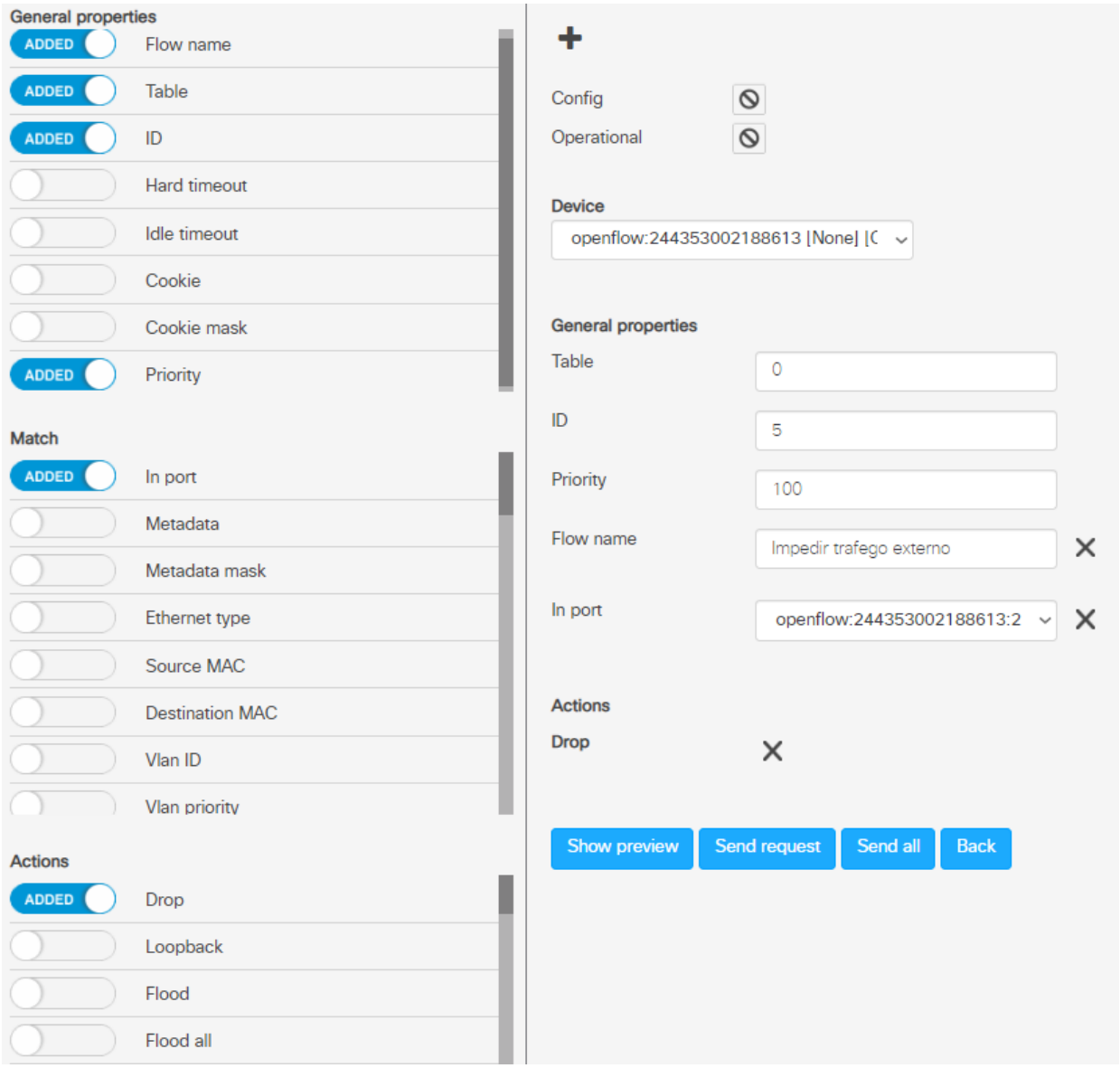


Figura 3.6 – Regra OFM para bloquear tráfego externo destinado a rede interna.

4 Validação e Discussão dos Resultados

Realizadas as configurações e criações de regras, foi possível testar o ambiente proposto. Portanto, este capítulo é dedicado à apresentação dos resultados obtidos.

Vale ressaltar que o endereço IP do controlador SDN é 192.168.1.39; e do Open vSwitch, 192.168.1.41. Ambos os endereços são usados para todas as seções e subseções deste capítulo.

4.1 Comunicação Entre Controlador e OVS

As figuras a seguir demonstram, por meio do Wireshark, algumas das principais formas de troca de mensagem entre Switch e controlador SDN, desde o estabelecimento da conexão até a mudança de regras de fluxo ou encerramento da comunicação.

Para estabelecer uma conexão OpenFlow, o primeiro passo é o envio da mensagem do tipo OFPT_HELLO, com o campo de versão apontando para a versão mais alta do protocolo suportada pelo remetente. A partir do recebimento desta mensagem, os dispositivos podem entrar em consenso quanto à versão a ser utilizada e então dar início à comunicação [33]. Isto é evidenciado nas figuras 4.1 e 4.2, em que a primeira mostra, além de todo o processo de handshake, o envio da mensagem OFPT_HELLO, enquanto a segunda detalha tal mensagem.

openflow_v4

No.	Time	Source	Destination	Protocol	Length	Info
32	1846.863693	192.168.1.41	192.168.1.39	OpenFlow	82	Type: OFPT_HELLO
34	1847.039675	192.168.1.39	192.168.1.41	OpenFlow	90	Type: OFPT_FEATURES_REQUEST
36	1847.043153	192.168.1.41	192.168.1.39	OpenFlow	98	Type: OFPT_FEATURES_REPLY
38	1847.115419	192.168.1.39	192.168.1.41	OpenFlow	98	Type: OFPT_MULTIPART_REQUEST, OFPMP_DESC
39	1847.117889	192.168.1.41	192.168.1.39	OpenFlow	1170	Type: OFPT_MULTIPART_REPLY, OFPMP_PORT_DESC
40	1847.117963	192.168.1.41	192.168.1.39	OpenFlow	1138	Type: OFPT_MULTIPART_REPLY, OFPMP_DESC
43	1847.307121	192.168.1.39	192.168.1.41	OpenFlow	194	Type: OFPT_FLOW_MOD
44	1847.335348	192.168.1.39	192.168.1.41	OpenFlow	242	Type: OFPT_FLOW_MOD
46	1847.596894	192.168.1.39	192.168.1.41	OpenFlow	94	Type: OFPT_SET_CONFIG
47	1847.599069	192.168.1.41	192.168.1.39	OpenFlow	122	Type: OFPT_MULTIPART_REPLY, OFPMP_GROUP_FEATURES
49	1847.639268	192.168.1.39	192.168.1.41	OpenFlow	82	Type: OFPT_MULTIPART_REQUEST, OFPMP_METER_FEATURES
50	1847.641194	192.168.1.41	192.168.1.39	OpenFlow	98	Type: OFPT_MULTIPART_REPLY, OFPMP_METER_FEATURES

Figura 4.1 – Handshake entre switch e controlador SDN.

Como destacado, o OVS diz ao controlador que a versão mais alta suportada é o OpenFlow 1.3. Nota-se que tal mensagem não possui *body*, isto é, trata-se apenas do *Header* do OpenFlow.

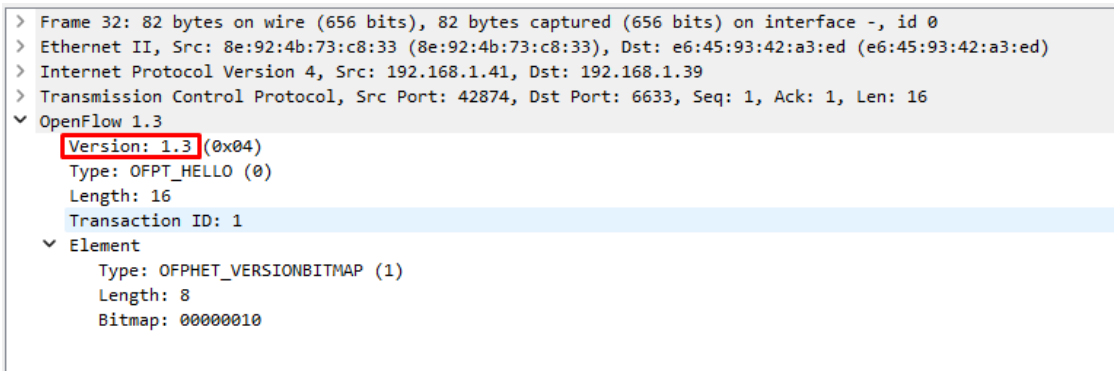


Figura 4.2 – Detalhes da mensagem OFPT_HELLO do OVS.

Passada a primeira etapa do estabelecimento da conexão, o controlador envia ao switch uma mensagem do tipo OFPT_FEATURES_REQUEST, a qual requisita informações a respeito das configurações do switch em geral, como portas operacionais e endereço MAC. O Switch, então, responde com uma mensagem do tipo OFTP_FEATURES_REPLY, a qual contém algumas das informações pedidas. A figura 4.3 mostra a mensagem enviada pelo switch no caso estudado.

O campo *datapath_id* identifica unicamente um *datapath*, sendo os 48 bits à direita o endereço MAC do switch (b2-39-f3-d1-3b-48) e os 16 bits à esquerda dependente da implementação. O campo *n_buffers*, por sua vez, especifica o número máximo de pacotes que o switch pode armazenar em buffer ao enviar pacotes para o controlador - neste caso, igual a zero -, utilizando mensagens do tipo *packet_in*, que serão explicadas mais adiante. O campo *n_tables*, como o próprio nome já diz, representa o número de tabelas suportadas pelo switch (254), as quais podem ser consultadas mais detalhadamente pelo controlador, por meio do envio de mensagens do tipo OFPMP_TABLE_FEATURES. Logo abaixo, o campo *auxiliary_id* identifica o tipo de conexão entre o switch e o controlador. No caso estudado, trata-se da principal conexão entre os mesmos, uma vez que seu campo é igual a zero. Por fim, o campo *capabilities* indica as funções suportadas pelo *datapath* [33]. Nota-se que o mesmo suporta o envio de informações de estado de fluxo, tabelas, portas, grupos e filas, além de não estarem ativadas as funções de rearranjo de fragmentos de IP e bloqueio de portas em *loop*.

```
> Frame 36: 98 bytes on wire (784 bits), 98 bytes captured (784 bits) on interface -, id 0
> Ethernet II, Src: 8e:92:4b:73:c8:33 (8e:92:4b:73:c8:33), Dst: e6:45:93:42:a3:ed (e6:45:93:42:a3:ed)
> Internet Protocol Version 4, Src: 192.168.1.41, Dst: 192.168.1.39
> Transmission Control Protocol, Src Port: 42874, Dst Port: 6633, Seq: 17, Ack: 25, Len: 32
v OpenFlow 1.3
    Version: 1.3 (0x04)
    Type: OFPT_FEATURES_REPLY (6)
    Length: 32
    Transaction ID: 2
    datapath_id: 0x0000b239f3d13b48
    n_buffers: 0
    n_tables: 254
    auxiliary_id: 0
    Pad: 0
  v capabilities: 0x0000004f
      .... .... .... .... .... .... .... ...1 = OFPC_FLOW_STATS: True
      .... .... .... .... .... .... .... ..1. = OFPC_TABLE_STATS: True
      .... .... .... .... .... .... .... .1.. = OFPC_PORT_STATS: True
      .... .... .... .... .... .... .... 1... = OFPC_GROUP_STATS: True
      .... .... .... .... .... .... ..0. .... = OFPC_IP_REASM: False
      .... .... .... .... .... .... .1.. .... = OFPC_QUEUE_STATS: True
      .... .... .... .... .... ...0 .... .... = OFPC_PORT_BLOCKED: False
    Reserved: 0x00000000
```

Figura 4.3 – Detalhes da mensagem OFPT_FEATURES_REPLY do OVS.

Concluídas as configurações iniciais, o controlador envia então uma mensagem do tipo OFPT_SET_CONFIG, representado pelo pacote 46 na figura 4.1, que pode ser utilizada para definir parâmetros de configuração no switch. Vale destacar que o switch não responde a este tipo de mensagem. Como visto na figura 4.4, destacam-se essencialmente dois campos: *Miss send length* e *IP Fragments*. O primeiro, igual a 65535, indica o número máximo de bytes que o *datapath* pode enviar ao controlador, enquanto o segundo indica que não se deve realizar nenhum tipo de tratamento especial nos pacotes IP fragmentados, ou seja, o manuseio "normal" de fragmentos significa que deve ser feita uma tentativa de passar os fragmentos pelas tabelas OpenFlow [33].

```
> Frame 46: 94 bytes on wire (752 bits), 94 bytes captured (752 bits) on interface -, id 0
> Ethernet II, Src: e6:45:93:42:a3:ed (e6:45:93:42:a3:ed), Dst: 8e:92:4b:73:c8:33 (8e:92:4b:73:c8:33)
> Internet Protocol Version 4, Src: 192.168.1.39, Dst: 192.168.1.41
> Transmission Control Protocol, Src Port: 6633, Dst Port: 42874, Seq: 361, Ack: 2225, Len: 28
> OpenFlow 1.3
v OpenFlow 1.3
    Version: 1.3 (0x04)
    Type: OFPT_SET_CONFIG (9)
    Length: 12
    Transaction ID: 8
  v Flags: 0x0000
      .... ..00 = IP Fragments: OFPC_FRAG_NORMAL (0)
    Miss send length: OFPCML_NO_BUFFER (65535)
```

Figura 4.4 – Detalhes da mensagem OFPT_SET_CONFIG do controlador.

Outra mensagem que se pode realçar é a OFPT_FLOW_MOD, a qual é utilizada pelo controlador para executar modificações, configurar ou encerrar regras em uma determinada *flow table*. Tal mensagem é expandida na figura 4.5, em que se têm a inserção de duas regras de fluxo nas tabelas do switch.

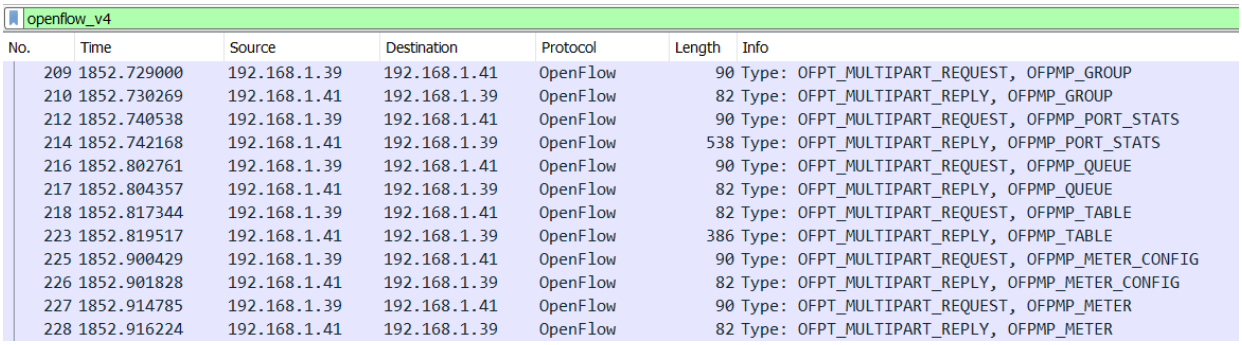
A primeira informação que pode ser obtida desta mensagem é a versão do OpenFlow a ser utilizada, sendo igual a 1.3. Em seguida, destaca-se o campo *cookie*, que se trata de um valor de dados opacos escolhido pelo controlador e pode ser utilizado para identificar unicamente e filtrar fluxos, sendo igual a 0x2b00000000000001 na primeira regra. Outro campo importante é o *Table ID*, que especifica a tabela na qual a entrada do fluxo deve ser inserida, modificada ou excluída, sendo neste caso igual a 0, isto é, a regra deve ser inserida na primeira tabela da *pipeline*. Em seguida, tem o campo *Command*, que especifica o comando a ser efetivamente executado e, como se pode ver, é igual a OFPFC_ADD. Este valor indica que a regra inexiste nas tabelas do switch e que, portanto, deve ser adicionada. Os campos Out port e Out group filtram opcionalmente o escopo das mensagens OFPFC_DELETE e OFPFC_DELETE_STRICT por porta de saída e de grupo. Como neste caso ambos possuem o valor de OFPP_ANY e OFPG_ANY especificados, a filtragem de saída está desabilitada para estes fluxos [33].

```
> Frame 43: 194 bytes on wire (1552 bits), 194 bytes captured (1552 bits) on interface -, id 0
> Ethernet II, Src: e6:45:93:42:a3:ed (e6:45:93:42:a3:ed), Dst: 8e:92:4b:73:c8:33 (8e:92:4b:73:c8:33)
> Internet Protocol Version 4, Src: 192.168.1.39, Dst: 192.168.1.41
> Transmission Control Protocol, Src Port: 6633, Dst Port: 42874, Seq: 57, Ack: 2225, Len: 128
v OpenFlow 1.3
    Version: 1.3 (0x04)
    Type: OFPT_FLOW_MOD (14)
    Length: 64
    Transaction ID: 3
    Cookie: 0x2b00000000000001
    Cookie mask: 0x0000000000000000
    Table ID: 0
    Command: OFPFC_ADD (0)
    Idle timeout: 0
    Hard timeout: 0
    Priority: 0
    Buffer ID: OFP_NO_BUFFER (4294967295)
    Out port: OFPP_ANY (4294967295)
    Out group: OFPG_ANY (4294967295)
  > Flags: 0x0000
    Pad: 0000
  > Match
  > Instruction
v OpenFlow 1.3
    Version: 1.3 (0x04)
    Type: OFPT_FLOW_MOD (14)
    Length: 64
    Transaction ID: 4
    Cookie: 0x2b00000000000000
    Cookie mask: 0x0000000000000000
    Table ID: 0
    Command: OFPFC_ADD (0)
    Idle timeout: 0
    Hard timeout: 0
    Priority: 0
    Buffer ID: OFP_NO_BUFFER (4294967295)
    Out port: OFPP_ANY (4294967295)
    Out group: OFPG_ANY (4294967295)
  > Flags: 0x0000
    Pad: 0000
  > Match
  > Instruction
```

Figura 4.5 – Detalhes da mensagem OFPT_FLOW_MOD do controlador.

A figura 4.6 apresenta outra troca de mensagens muito importante principalmente para a manu-

tenção da conexão entre controlador e switch, que se tratam das mensagens de OFPT_MULTIPART_REQUEST e OFPT_MULTIPART_REPLY. Enquanto o sistema está em funcionamento, o controlador pode requisitar informações de estado ao *datapath* utilizando a primeira, enquanto o switch responde utilizando a segunda. Tanto na requisição quanto na resposta, o campo responsável por especificar o tipo de informação a ser passada e como o *body* deve ser interpretado é o *type* [33]. Como é visto na figura, essencialmente seis tipos de informação foram trocadas: OFPMP_GROUP, OFPMP_PORT_STATS, OFPMP_QUEUE, OFPMP_TABLE, OFPMP_METER_CONFIG e OFPMP_METER. O primeiro trata de informações relacionadas ao estado do grupo, o segundo do estado de portas, o terceiro lida com informações de filas, o quarto com estado das tabelas e os dois últimos tratam informações e configurações de métricas.



openflow_v4

No.	Time	Source	Destination	Protocol	Length	Info
209	1852.729000	192.168.1.39	192.168.1.41	OpenFlow	90	Type: OFPT_MULTIPART_REQUEST, OFPMP_GROUP
210	1852.730269	192.168.1.41	192.168.1.39	OpenFlow	82	Type: OFPT_MULTIPART_REPLY, OFPMP_GROUP
212	1852.740538	192.168.1.39	192.168.1.41	OpenFlow	90	Type: OFPT_MULTIPART_REQUEST, OFPMP_PORT_STATS
214	1852.742168	192.168.1.41	192.168.1.39	OpenFlow	538	Type: OFPT_MULTIPART_REPLY, OFPMP_PORT_STATS
216	1852.802761	192.168.1.39	192.168.1.41	OpenFlow	90	Type: OFPT_MULTIPART_REQUEST, OFPMP_QUEUE
217	1852.804357	192.168.1.41	192.168.1.39	OpenFlow	82	Type: OFPT_MULTIPART_REPLY, OFPMP_QUEUE
218	1852.817344	192.168.1.39	192.168.1.41	OpenFlow	82	Type: OFPT_MULTIPART_REQUEST, OFPMP_TABLE
223	1852.819517	192.168.1.41	192.168.1.39	OpenFlow	386	Type: OFPT_MULTIPART_REPLY, OFPMP_TABLE
225	1852.900429	192.168.1.39	192.168.1.41	OpenFlow	90	Type: OFPT_MULTIPART_REQUEST, OFPMP_METER_CONFIG
226	1852.901828	192.168.1.41	192.168.1.39	OpenFlow	82	Type: OFPT_MULTIPART_REPLY, OFPMP_METER_CONFIG
227	1852.914785	192.168.1.39	192.168.1.41	OpenFlow	90	Type: OFPT_MULTIPART_REQUEST, OFPMP_METER
228	1852.916224	192.168.1.41	192.168.1.39	OpenFlow	82	Type: OFPT_MULTIPART_REPLY, OFPMP_METER

Figura 4.6 – Mensagens OFPT_MULTIPART_REQUEST/REPLY.

A figura 4.7 mostra a resposta do OVS para a requisição OFPMP_PORT_STATS. Nota-se que, além das informações de versão e tipo já conhecidos, estão no *body* da resposta as informações pedidas. Além disso, observa-se que são fornecidos dados como número de pacotes recebidos, número de pacotes enviados, número de bytes trocados, número de pacotes derrubados, número de erros, número de colisões e tempo de atividade, para cada porta do switch.

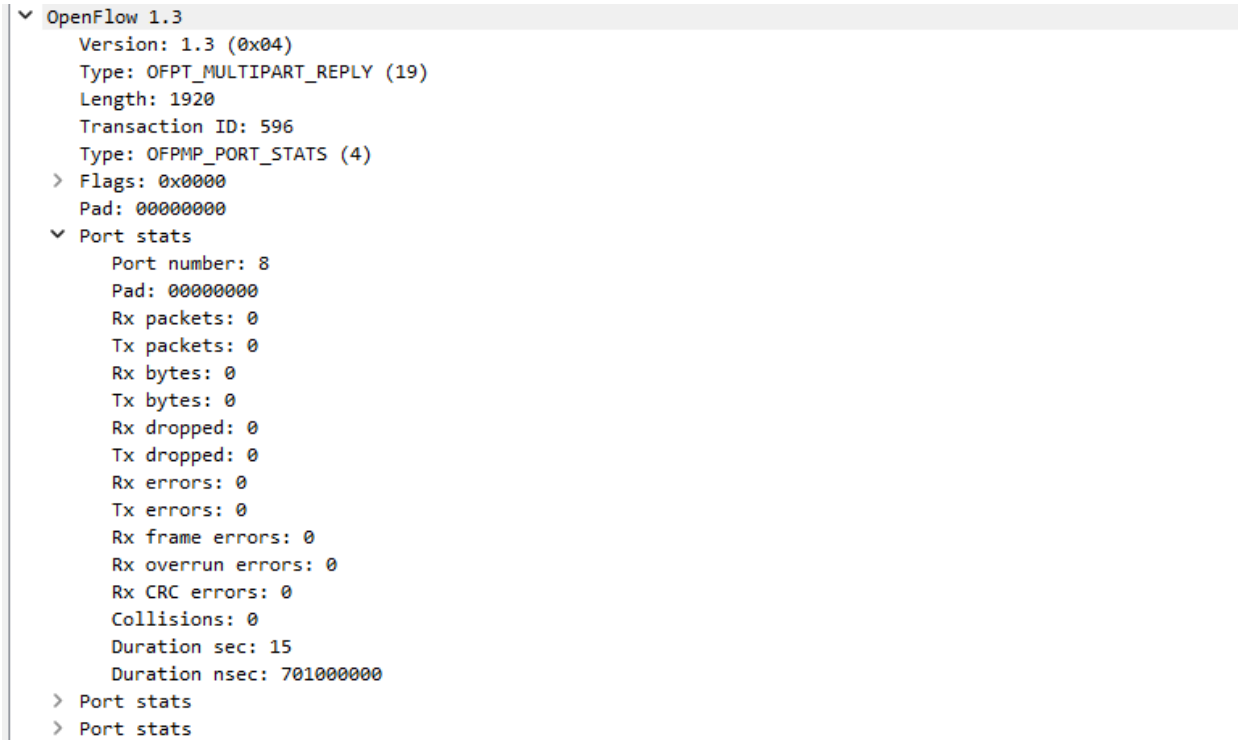


Figura 4.7 – Detalhes da mensagem OFPMP_PORT_STATS do OVS.

A figura 4.8 mostra a resposta do OVS para a requisição OFPMP_TABLE. São apresentadas estatísticas de número de entradas ativas, número de pacotes pesquisados na tabela e número de pacotes que foram interceptados pelas regras definidas. Esses dados podem ser muito úteis tanto para o controle da rede quanto para auditorias, sejam elas internas ou externas.

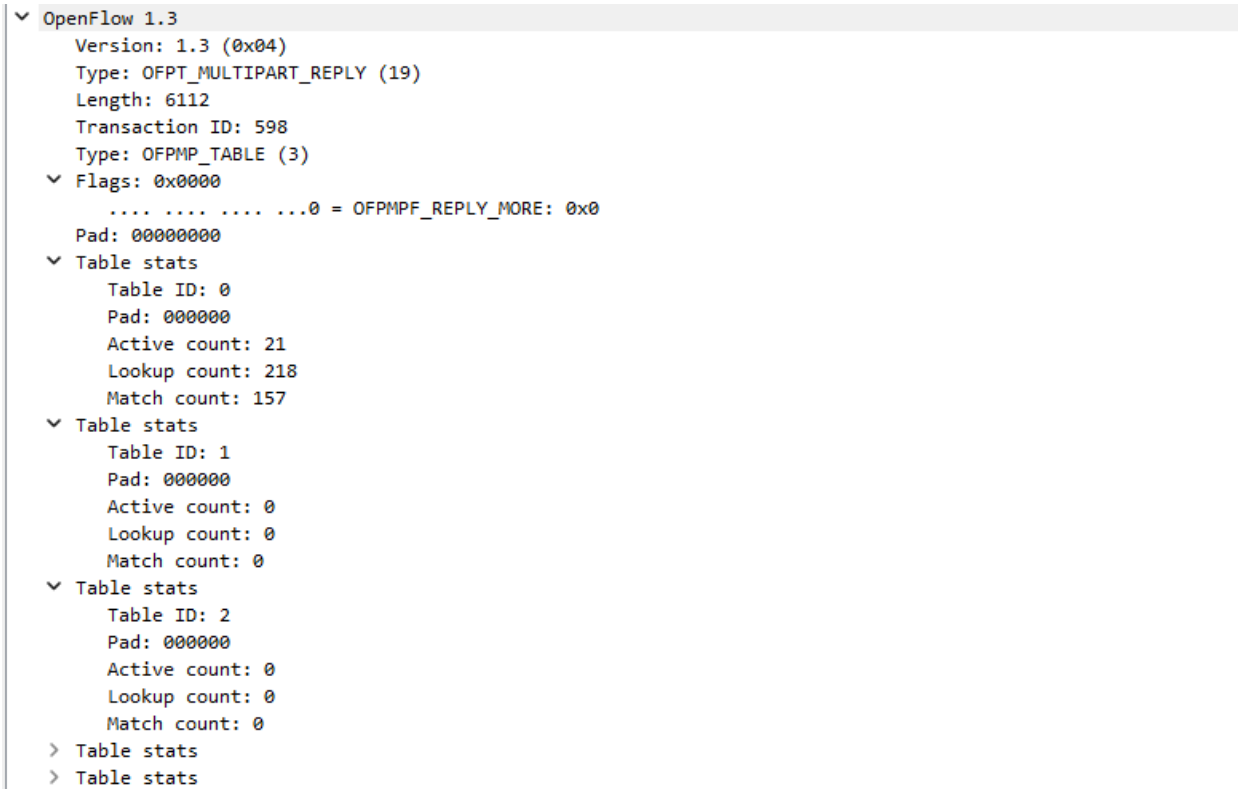


Figura 4.8 – Detalhes da resposta OFPMP_TABLE_STATS do OVS.

A figura 4.9 mostra a troca de mensagens OFPMP_AGGREGATE entre o controlador e o OVS. Inicialmente, o controlador SDN envia uma mensagem OFPMP_AGGREGATE do tipo OFPT_MULTIPART_REQUEST. Nesse caso, o controlador está requisitando informações agregadas sobre múltiplas entradas [33]. Em seguida, o OVS responde por meio das mensagens OFPT_MULTIPART_REPLY.

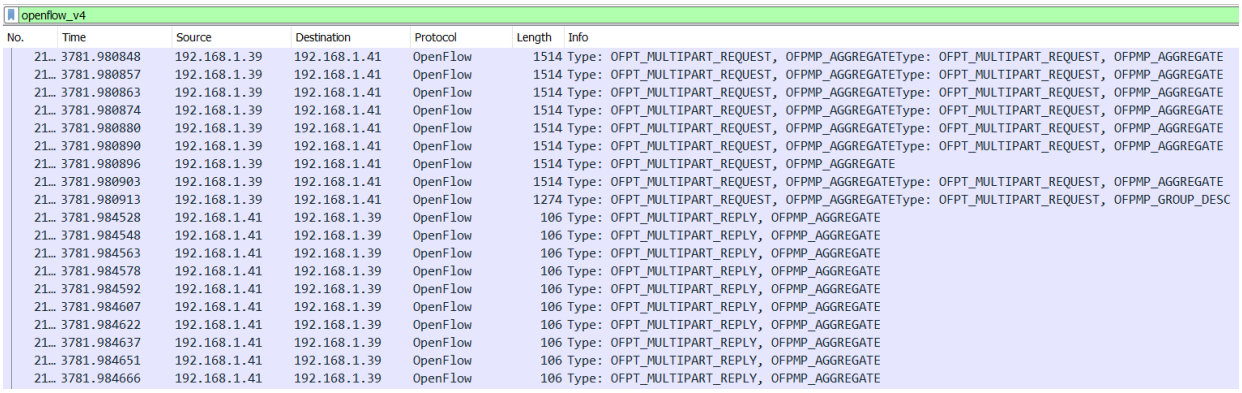


Figura 4.9 – Mensagens OFPMP_AGGREGATE.

Com o objetivo de manter o serviço, o controlador SDN deve continuamente estar atualizando-se sobre a rede e sua topologia. Após o estabelecimento da conexão entre os dispositivos, apesar de já ter conhecimento das portas do Switch e seus respectivos endereços, o controlador ainda não está completamente ciente dos seus links operacionais e como eles estão conectados. Para resolver

isso, o controlador envia, periodicamente, mensagens do tipo OFTP_PACKET_OUT para todas as portas anunciadas pelo Switch. Utilizando o protocolo LLDP (Link Layer Discovery Protocol), este tipo de mensagem é fundamental para redes SDN, onde o descobrimento de topologia é mais crítico e tal função deve ser exercida pelo controlador quase que em tempo real. [34]. Esse envio de pacotes pode ser visto na figura 4.10, enquanto na figura 4.11 um desses pacotes é detalhado.

openflow_v4

No.	Time	Source	Destination	Protocol	Length	Info
61	1847.880041	192.168.1.39	192.168.1.41	OpenFlow	219	Type: OFPT_PACKET_OUT
62	1847.883128	192.168.1.39	192.168.1.41	OpenFlow	219	Type: OFPT_PACKET_OUT
65	1847.886925	192.168.1.39	192.168.1.41	OpenFlow	219	Type: OFPT_PACKET_OUT
66	1847.890677	192.168.1.39	192.168.1.41	OpenFlow	219	Type: OFPT_PACKET_OUT
68	1847.892822	192.168.1.39	192.168.1.41	OpenFlow	221	Type: OFPT_PACKET_OUT
69	1847.895093	192.168.1.39	192.168.1.41	OpenFlow	220	Type: OFPT_PACKET_OUT
71	1847.897528	192.168.1.39	192.168.1.41	OpenFlow	219	Type: OFPT_PACKET_OUT
72	1847.899854	192.168.1.39	192.168.1.41	OpenFlow	220	Type: OFPT_PACKET_OUT
74	1847.903124	192.168.1.39	192.168.1.41	OpenFlow	220	Type: OFPT_PACKET_OUT
75	1847.906215	192.168.1.39	192.168.1.41	OpenFlow	220	Type: OFPT_PACKET_OUT
77	1848.226291	192.168.1.39	192.168.1.41	OpenFlow	1514	Type: OFPT_PACKET_OUT

Figura 4.10 – Mensagem OFPT_PACKET_OUT do controlador.

O campo *Buffer ID* indica se os dados do pacote estão incluídos no *data array*, isto é, diz se alguns bytes da mensagem estão incluídos na porção de dados da mensagem. Neste caso, como seu valor é igual a OFP_NO_BUFFER, conclui-se que tais dados estão de fato incluídos no mesmo. O campo *In Port*, por sua vez, indica a porta de entrada que deve ser associada com o pacote para processamento do OpenFlow. Nota-se que, no caso estudado, este campo é igual a OFPP_CONTROLLER, que é uma constante reponsável por indicar o endereço de destino de um pacote de fluxo para o controlador OpenFlow. Quando um pacote é enviado para este endereço, o controlador é responsável por decidir qual ação tomar em relação ao pacote, como por exemplo, redirecioná-lo para outra porta ou descartá-lo. Vale destacar que este tipo de mensagem também pode ser utilizada para o controlador realizar o envio de um pacote de dados a partir do controlador para uma das portas de um switch OpenFlow [33].

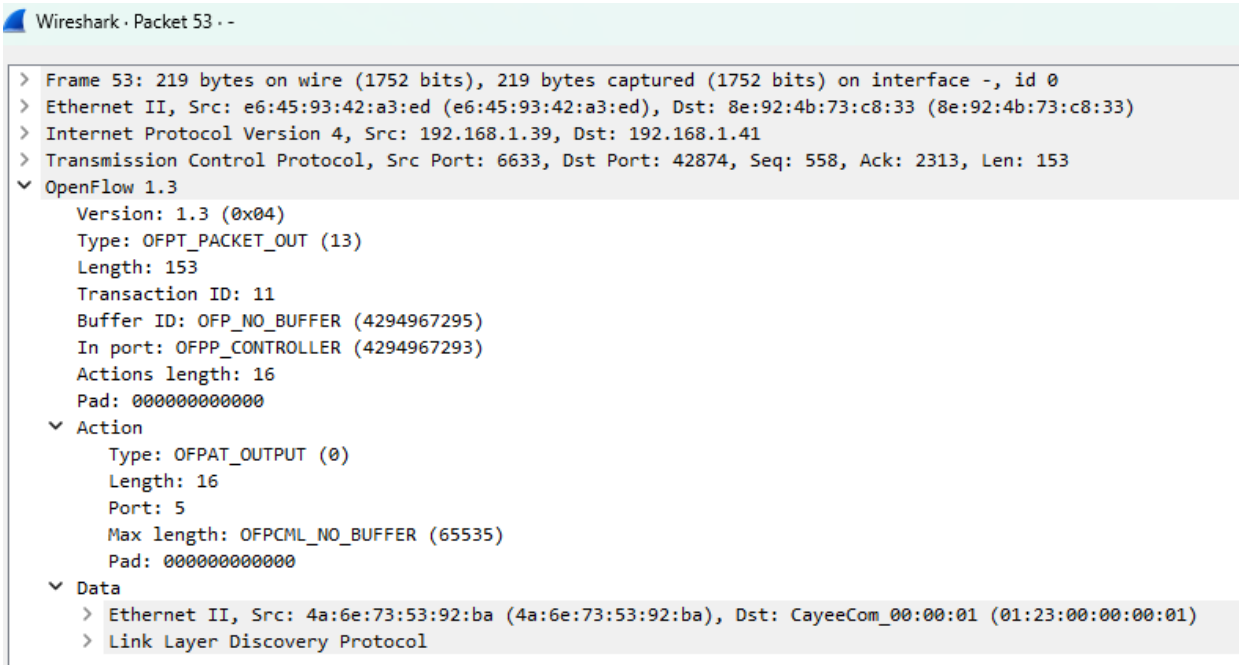
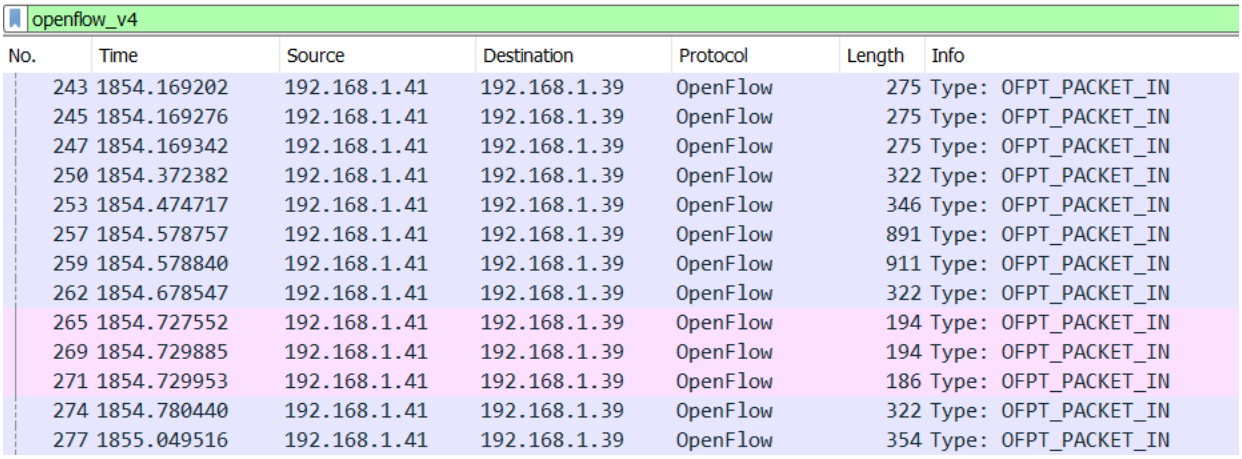


Figura 4.11 – Detalhes da mensagem PACKET_OUT.

Visto na figura 4.12, as mensagens do tipo PACKET_IN costumam ser utilizadas quando os pacotes são recebidos pelo *datapath* e deve-se notificar o controlador da mesma. Ou seja, este tipo de mensagem é utilizado no protocolo OpenFlow para notificar o controlador sobre o recebimento de um pacote de dados em uma das portas de um switch OpenFlow. Quando um pacote é recebido em uma porta e não há um fluxo configurado para lidar com ele, o switch envia uma mensagem Packet In ao controlador para que ele decida a ação a ser tomada [33]. Este pacote inclui o Switch ID e a porta da entrada na qual o pacote foi recebido e, devido a essas informações, o controlador é capaz de determinar a presença do link operacional no Switch e atualizar seu banco de dados a respeito da topologia da rede [34].



No.	Time	Source	Destination	Protocol	Length	Info
243	1854.169202	192.168.1.41	192.168.1.39	OpenFlow	275	Type: OFPT_PACKET_IN
245	1854.169276	192.168.1.41	192.168.1.39	OpenFlow	275	Type: OFPT_PACKET_IN
247	1854.169342	192.168.1.41	192.168.1.39	OpenFlow	275	Type: OFPT_PACKET_IN
250	1854.372382	192.168.1.41	192.168.1.39	OpenFlow	322	Type: OFPT_PACKET_IN
253	1854.474717	192.168.1.41	192.168.1.39	OpenFlow	346	Type: OFPT_PACKET_IN
257	1854.578757	192.168.1.41	192.168.1.39	OpenFlow	891	Type: OFPT_PACKET_IN
259	1854.578840	192.168.1.41	192.168.1.39	OpenFlow	911	Type: OFPT_PACKET_IN
262	1854.678547	192.168.1.41	192.168.1.39	OpenFlow	322	Type: OFPT_PACKET_IN
265	1854.727552	192.168.1.41	192.168.1.39	OpenFlow	194	Type: OFPT_PACKET_IN
269	1854.729885	192.168.1.41	192.168.1.39	OpenFlow	194	Type: OFPT_PACKET_IN
271	1854.729953	192.168.1.41	192.168.1.39	OpenFlow	186	Type: OFPT_PACKET_IN
274	1854.780440	192.168.1.41	192.168.1.39	OpenFlow	322	Type: OFPT_PACKET_IN
277	1855.049516	192.168.1.41	192.168.1.39	OpenFlow	354	Type: OFPT_PACKET_IN

Figura 4.12 – Mensagem OFPT_PACKET_IN do OVS.

A figura 4.13 detalha um desses pacotes *Packet In*. O campo "Buffer ID" realiza a mesma função que nas mensagens do tipo *Packet Out*, ou seja, indica se parte do pacote está ou não em

buffer. Neste caso, percebe-se que de fato ocorre. O campo “Reason”, por sua vez, indica a razão do envio do pacote. Tal campo é igual a OFPR_ACTION na figura 4.13, anunciando que foi uma ação expressa explicitamente para o controlador, o que pode indicar ao mesmo que determinada ação especificada foi executada com sucesso. Por fim, o campo “Match” reflete os headers do pacote e contextualiza o controlador para quando o pacote de entrada possui o campo de OXM TLVs, que inclui quaisquer alterações aplicadas ao pacote no processamento anterior, incluindo ações já executadas, se houver, mas não quaisquer alterações no conjunto de ações. Os TLVs OXM devem incluir campos de contexto, ou seja, campos cujos valores não podem ser determinados a partir dos dados do pacote [33].

```
▾ OpenFlow 1.3
    Version: 1.3 (0x04)
    Type: OFPT_PACKET_IN (10)
    Length: 280
    Transaction ID: 0
    Buffer ID: OFP_NO_BUFFER (4294967295)
    Total length: 238
    Reason: OFPR_ACTION (1)
    Table ID: 0
    Cookie: 0x2b00000000000006
  ▾ Match
      Type: OFPMT_OXM (1)
      Length: 12
    › OXM field
      Pad: 00000000
    Pad: 0000
  ▾ Data
    ▾ Ethernet II, Src: SamsungE_2d:78:c6 (38:68:a4:2d:78:c6), Dst: IPv4mcast_07 (01:00:5e:00:00:07)
      › Destination: IPv4mcast_07 (01:00:5e:00:00:07)
      › Source: SamsungE_2d:78:c6 (38:68:a4:2d:78:c6)
        Type: IPv4 (0x0800)
    ▾ Internet Protocol Version 4, Src: 192.168.1.4, Dst: 224.0.0.7
        0100 .... = Version: 4
        .... 0101 = Header Length: 20 bytes (5)
      › Differentiated Services Field: 0x88 (DSCP: AF41, ECN: Not-ECT)
        Total Length: 224
        Identification: 0x7af1 (31473)
      › 010. .... = Flags: 0x2, Don't fragment
        ...0 0000 0000 0000 = Fragment Offset: 0
        Time to Live: 1
        Protocol: UDP (17)
        Header Checksum: 0x5be0 [validation disabled]
        [Header checksum status: Unverified]
        Source Address: 192.168.1.4
        Destination Address: 224.0.0.7
    ▾ User Datagram Protocol, Src Port: 8001, Dst Port: 8001
        Source Port: 8001
        Destination Port: 8001
        Length: 204
        Checksum: 0xf6f0 [unverified]
        [Checksum Status: Unverified]
        [Stream index: 0]
      ▾ [Timestamps]
          [Time since first frame: 1.940827000 seconds]
          [Time since previous frame: 0.000562000 seconds]
        UDP payload (196 bytes)
▾ Data (196 bytes)
    Data: 7b2264617461223a7b227631223a7b22757269223a22687474703a2f2f3139322e313638…
    [Length: 196]
```

Figura 4.13 – Detalhes da mensagem OFPT_PACKET_IN.

A figura 4.14, que se trata do *flow graph* gerado no Wireshark para esta troca de mensagens, ilustra todo este processo de handshake e controle. Primeiramente, o Switch envia um pacote do tipo OFPT_HELLO, seguido dos pacotes OFPT_FEATURES_REQUEST e OFPT_FEATURES_REPLY. A partir desta comunicação inicial, mensagens do tipo MULTIPART também começam a ser trocadas, os fluxos são criados pela mensagem de OFPT_FLOW_MOD e, então, são configurados outros parâmetros da comunicação por meio da mensagem de OFPT_SET_CONFIG. Após isso, o controlador começa a enviar mensagens do tipo OFPT_PACKET_OUT para conhecer o estado das portas do OVS.

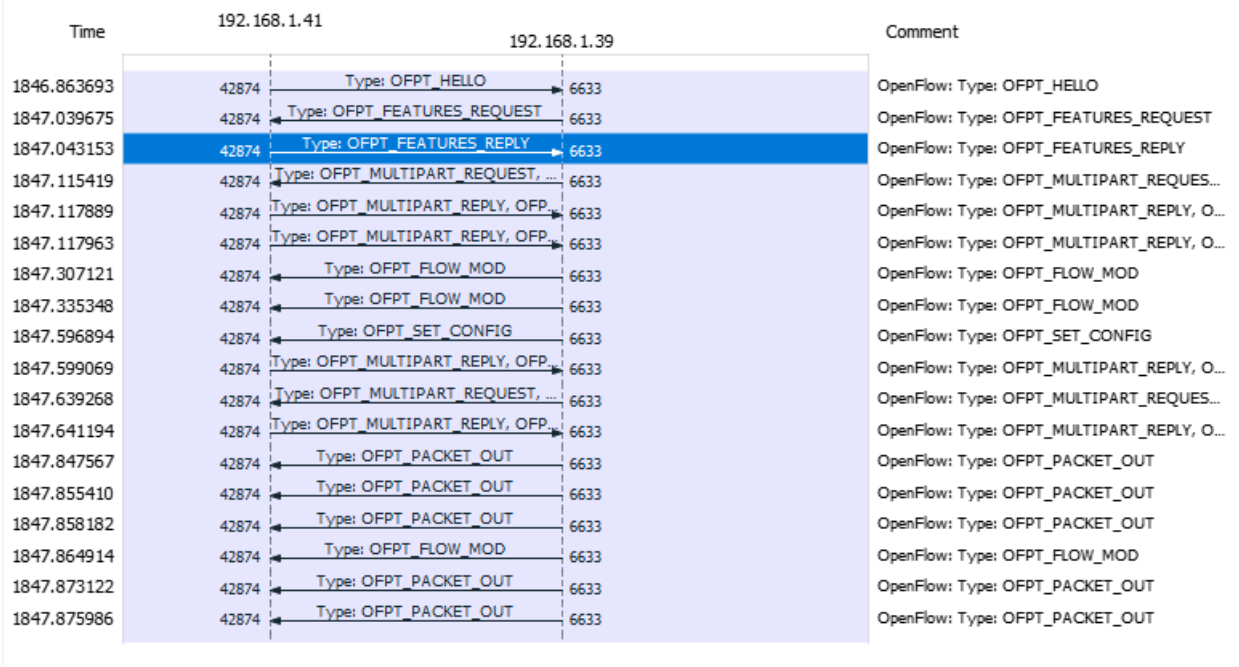


Figura 4.14 – Flow graph entre controlador SDN e OVS.

4.2 Comunicação Entre ODL e OFM

Como dito na seção 2, a comunicação entre a camada de controle e a camada de gerenciamento em redes SDN é feita geralmente por meio de Northbound APIs. No caso deste trabalho, isto não é diferente. A comunicação entre o ODL e o OFM é feita inteiramente por meio destas APIs.

Como no caso deste trabalho tanto ODL quanto OFM estão na mesma máquina, esta comunicação não pode ser rastreada por meio do Wireshark, portanto, as figuras 4.15 e 4.16 apresentam um exemplo desta comunicação por meio do elemento de inspeção do navegador. A primeira figura apresenta o *header* da requisição, enquanto a segunda mostra o *payload* da mesma, sendo que o objetivo desta requisição é a criação de uma nova regra na tabela de fluxos do OVS.

Como dito, a figura 4.15 representa o processo de requisição por parte do OFM à Northbound API disponibilizada por padrão pelo OpenDaylight. Desta forma, o OFM primeiro comunica o ODL sobre a criação de uma nova regra, e em seguida o ODL envia uma mensagem do tipo OFPT_FLOW_MOD para o switch semelhante à mensagem apresentada na figura 4.4, com o fim de se criar uma nova regra na tabela de fluxo. Esta mensagem deve indicar, ao switch de destino,

sua respectiva tabela a ser modificada e o ID do fluxo.

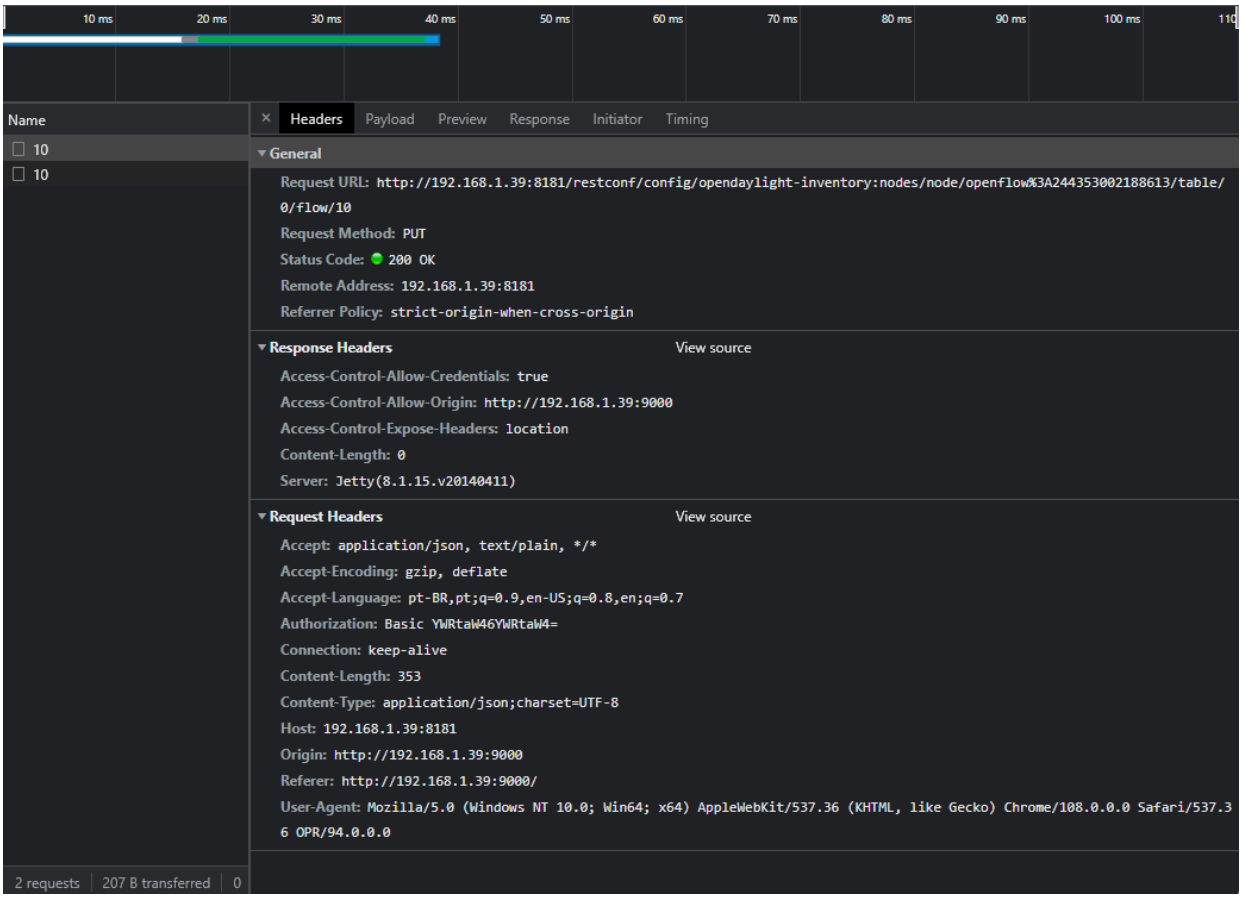


Figura 4.15 – Requisição de criação de regra de fluxo do OFM ao OVS.

Como visto na figura 4.15, a URL de requisição é "http://192.168.1.39:8181/restconf/config/opendaylight-inventory:nodes/node/openflow%3A244353002188613/table/0/flow/10". Isto quer dizer que, para o nó identificado por "openflow244353002188613", que é o OVS, deve-se criar, na tabela 0, o fluxo cujo ID é igual a 10. Para isso, é utilizado o método PUT e é usada também uma política de referência do tipo "strict-origin-when-cross-origin", que é uma política de referência cruzada (CORS), que define como um site é tratado quando faz uma solicitação a uma origem diferente. Esse modo específico permite acesso aos recursos compartilhados apenas se a solicitação for feita a partir da mesma origem (mesmo protocolo, mesmo host e mesma porta).

Além disso, são passados outros parâmetros relacionados ao cabeçalho de resposta e cabeçalho de requisição, como o tipo de metadado aceito, o tipo de controle de acesso, linguagem, método de autorização e os endereços IP de fonte e destino. Um campo que se pode destacar é o de "Authorization" que, no caso, utiliza o método Basic Authentication [35]. Este é um mecanismo de autenticação baseado em cabeçalho HTTP (HyperText Transfer Protocol) que permite que o servidor verifique a identidade de um usuário antes de permitir o acesso a recursos protegidos. Isso é realizado enviando as credenciais de usuário (nome de usuário e senha) codificadas em base64 no cabeçalho "Authorization" da solicitação HTTP. Esse mecanismo é considerado básico pois não oferece proteção contra interceptação ou modificação de credenciais durante a transmissão.

Ou seja, para um ambiente de produção, é recomendada a troca deste mecanismo para um mais avançado, como o "Authorization Bearer Token".

Como exemplo desta fragilidade, pegando o valor codificado apresentado em tal figura, que é igual a "YWRtaW46YWRtaW4=" e, inserindo-o em um decodificador de Base64, obtém-se o resultado "admin:admin", que são justamente as credenciais de acesso padrão do OpenDaylight.

Tratando-se agora da figura 4.16, tem-se efetivamente o fluxo a ser cadastrado nas tabelas do OVS, originalmente representado no formato XML (eXtensible Markup Language). Ele descreve as correspondências do OpenFlow e é determinado pelo modelo yang opendaylight-match-types [31].

Neste caso, criou-se uma regra de fluxo demonstrativa cujo nome é "Block 8.8.8.8", com hard-timout igual a 0, ID igual a 10, idle-timout igual a 0, prioridade de 100 e tabela destino igual a 0. O objetivo é derrubar pacotes IPv4 destinados ao IP 8.8.8.8, que é um endereço IP público do Google — frequentemente usado como um servidor DNS (Domain Name System) padrão.

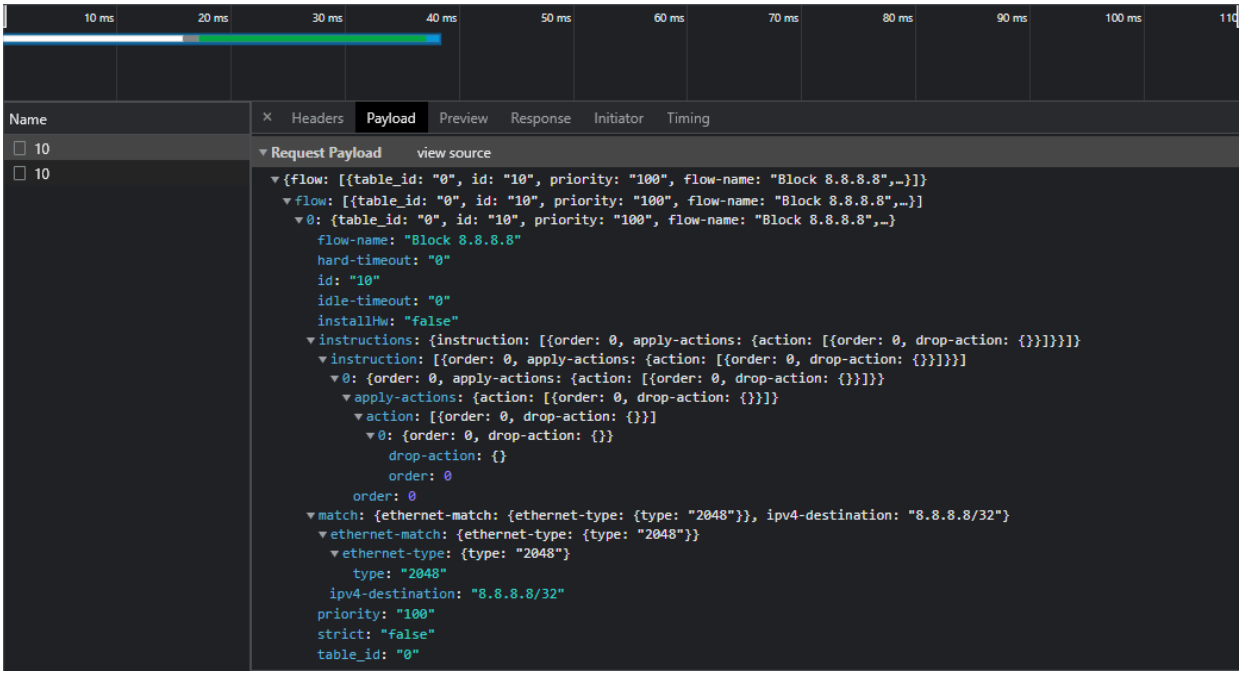


Figura 4.16 – Payload da requisição de criação de regra de fluxo do OFM ao OVS.

4.3 Validação das Regras de Controle de Fluxo

4.3.1 Resultados OVS, Open vSwitch e OFM

Finalmente, após a realização de toda a comunicação e estabelecimento do enlace entre OVS e controlador, o switch constrói todos os fluxos encaminhados pelo controlador em suas tabelas. A figura 4.17 apresenta a resposta do switch à execução do comando "ovs-ofctl -O OpenFlow13 dump-flows br0", que é usado para exibir as regras de fluxo (flows) atualmente instaladas no controlador OpenFlow na interface de bridge (br0). O "ovs-ofctl" é a ferramenta de linha de comando para controlar o Open vSwitch, "-O OpenFlow13" especifica que o protocolo OpenFlow 1.3 será usado,

e “dump-flows br0” é o comando para exibir as regras de fluxo na interface de bridge especificada (br0).

Este comando retorna uma listagem de regras de fluxo, cada uma com seus respectivos detalhes, incluindo a fonte, o destino, o endereço MAC, endereço IP, porta, ações (por exemplo, encaminhar para uma porta específica, drop) e contador de fluxo (como número de pacotes e bytes que correspondem a esta regra).

Cada uma das regras definidas na seção 3.7 foram identificadas e ressaltadas na figura 4.17. Por exemplo, a primeira regra marcada, de número 4, corresponde à quarta regra definida anteriormente. Nota-se que ela está ativa a aproximadamente 16747 segundos, 5871 pacotes foram compatíveis com ela e 1247293 bytes foram trafegados. Cada uma das outras regras apresentadas na mesma figura representa simples regras de encaminhamento de pacotes entre as interfaces do dispositivo.

Figura 4.17 – dumpFlowsOVS.

Prosseguindo, a figura 4.18 apresenta o retorno do OVS para o comando “ovs-vsctl show”, que é usado para exibir as configurações atuais do switch virtual. O “ovs-vsctl” é a ferramenta de linha de comando para controlar o Open vSwitch, e “show” é o comando para exibir a configuração atual.

A saída deste comando exibe informações detalhadas sobre as bridges, interfaces, portas e regras de fluxo configuradas no Open vSwitch, incluindo o nome, endereço MAC, endereço IP, VLAN (Virtual Local Area Network), MTU (Maximum Transmission Unit), dentre outros.

Observando a figura 4.18, nota-se que existem quatro bridges configuradas no OVS: br0, br1, br2 e br3. Entretanto, dentre elas apenas a br0 possui portas atribuídas, enquanto as outras permanecem em espera. Para o protocolo OpenFlow, *bridge* é uma entidade virtual que age como

um switch lógico, permitindo que as redes lógicas sejam conectadas entre si. A bridge mantém uma tabela de fluxo, que contém regras para determinar como os pacotes são encaminhados entre as interfaces conectadas [36].

Nota-se que todas as interfaces disponíveis no switch estão atribuídas a esta *bridge* (br0), da eth0 até a eth15. Além disso, percebe-se que é indicado o controlador ao qual a *bridge* está conectada, 192.168.1.39, o estado da conexão com o mesmo, e o *datapath_ type*, que no caso é igual a *netdev*, indicando que a *bridge* está sendo implementada como uma bridge de rede. Isso significa que a *bridge* está operando diretamente na camada de rede, em vez de em uma camada mais alta, como a camada de transporte.

```
/ # ovs-vsctl show
76bb728c-b9ec-4441-88c4-e1fd5b890900
    Bridge br2
        datapath_type: netdev
        Port br2
            Interface br2
                type: internal
    Bridge br3
        datapath_type: netdev
        Port br3
            Interface br3
                type: internal
    Bridge br0
        Controller "tcp:192.168.1.39:6633"
            is_connected: true
        datapath_type: netdev
        Port eth6
            Interface eth6
        Port eth5
            Interface eth5
        Port eth12
            Interface eth12
        Port eth0
            Interface eth0
        Port eth15
            Interface eth15
        Port eth8
            Interface eth8
        Port eth14
            Interface eth14
        Port eth3
            Interface eth3
        Port br0
            Interface br0
                type: internal
        Port eth2
            Interface eth2
        Port eth9
            Interface eth9
        Port eth13
            Interface eth13
        Port eth11
            Interface eth11
        Port eth1
            Interface eth1
        Port eth7
            Interface eth7
        Port eth4
            Interface eth4
        Port eth10
            Interface eth10
    Bridge br1
        datapath_type: netdev
        Port br1
            Interface br1
                type: internal
```

Figura 4.18 – vsctlShow.

As figuras 4.19 e 4.20 apresentam a visão da topologia gerada respectivamente pelo ODL e pelo OFM. Como esperado, são apresentados, além dos dispositivos presentes na topologia, outros dispositivos presentes na rede física local, dentre eles o roteador da rede, identificado pelo endereço

MAC “e0:1f:ed:ba:8d:88”. Esta ilustação é montada de acordo com a descoberta de topologia descrita na seção 4.1.

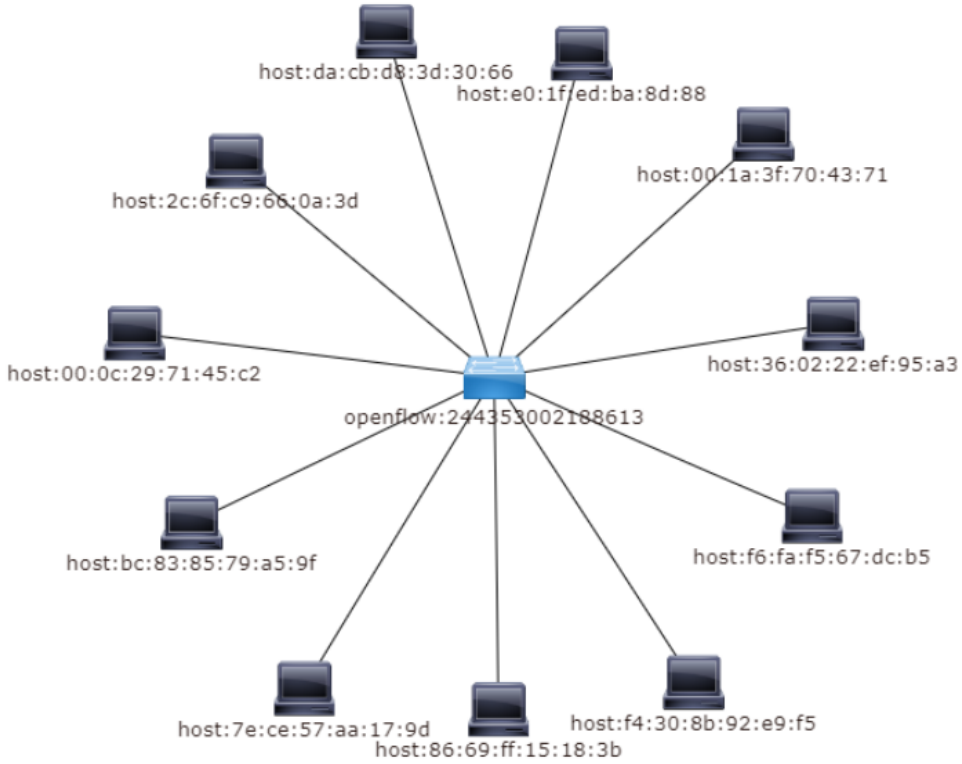


Figura 4.19 – Topologia do ODL.

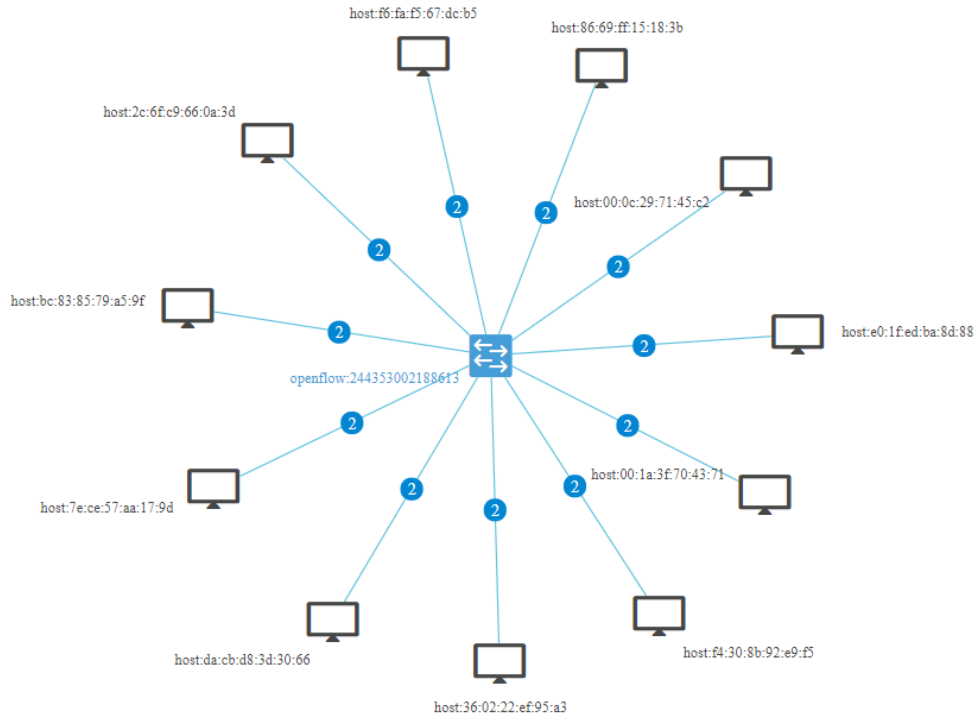


Figura 4.20 – Topologia do OFM.

Além de apresentar uma ilustração da topologia, o OFM também exibe tabelas com listagens de regras de fluxos e dispositivos cadastrados no sistema. A figura 4.21 apresenta algumas das regras configuradas, enquanto a figura 4.22 mostra alguns dos dispositivos presentes na mesma.

Analisando a figura 4.21, observa-se que são disponibilizados 7 (sete) campos de informação. O primeiro é o *Flow Name*, que indica o nome fluxo configurado. O segundo é o *ID*, que é o identificador único que reconhece unicamente um fluxo. O terceiro campo é o *Table ID*, responsável por determinar a tabela à qual a regra faz parte. O quarto campo é o *Device*, que indica a qual switch a mesma está presente. O quinto campo, *Device type*, por sua vez, mostra qual o tipo de dispositivo no qual a regra se encontra. O campo *Device name* indica qual o nome do switch, sendo neste caso igual a "None", isto é, o aparelho não possui nome definido. O penúltimo campo, *Operational*, é responsável por dizer o estado operacional da regra apresentada. Se este campo for igual a "ON DEVICE", quer dizer que a regra está funcionando normalmente e está em execução. Caso este valor seja "NOT ON DEVICE", significa que a regra não está operacional ou ainda não foi aplicada. Por fim, o campo *Actions* indica algumas das operações que podem ser feitas nesta regra, como editar, visualizar ou excluir.

Flows

Flow name	ID	Table ID	Device	Device type	Device name	Operational	Actions
Impedir trafego externo	5	0	openflow:244353002188613	Open vSwitch	None	ON DEVICE	
Permitir administracao	4	0	openflow:244353002188613	Open vSwitch	None	ON DEVICE	
Permitir acesso moodle	3	0	openflow:244353002188613	Open vSwitch	None	ON DEVICE	
Impedir trafego entre PCs	2	0	openflow:244353002188613	Open vSwitch	None	ON DEVICE	
Bloquear trafego sub-rede prova	1	0	openflow:244353002188613	Open vSwitch	None	ON DEVICE	
[id:CtrlGen 0-9, table:0]	#UF$TABLE*0-9	0	openflow:244353002188613	Open vSwitch	None	ON DEVICE	
[id:CtrlGen 0-8, table:0]	#UF$TABLE*0-8	0	openflow:244353002188613	Open vSwitch	None	ON DEVICE	

Figura 4.21 – Flow Table no OFM.

Partindo agora para a figura 4.22, percebe-se que a tabela de Hosts do OFM detalha 6 (seis) campos referentes a cada dispositivo conectado. O primeiro, *Host ID*, é uma identificação única atribuída a cada dispositivo conectado à rede SDN. É usado para rastrear e gerenciar dispositivos na rede, alocar recursos de rede e fornecer segurança e políticas de rede apropriadas para cada dispositivo.

O segundo campo, *Attachment Point ID*, é um identificador único de uma porta de switch específica na rede SDN. Ele é utilizado para determinar a posição de um dispositivo na rede e para determinar como o tráfego de rede é roteado em relação a esse dispositivo. Este campo é

importante para o controle e gerenciamento da rede, incluindo a alocação de recursos de rede, a implementação de políticas de segurança e o gerenciamento de tráfego.

O terceiro campo, *Attachment Point ID status*, representa o estado atual de um *Attachment Point ID*, isto é, de uma porta específica do switch. Pode incluir informações sobre o estado da porta (ativa ou inativa), se está operando corretamente ou se há algum erro, se está com congestionamento, dentre outras informações. Neste caso, como seu valor é igual a *true*, significa que o enlace está funcionando normalmente. Este campo é muito usado para monitorar a saúde da rede e identificar potenciais problemas.

O quarto e quinto campos, *HTS address IP* e *HTS address MAC*, representam respectivamente o endereço IP atribuído a um dispositivo da rede SDN e o endereço MAC referente a este mesmo dispositivo.

Por fim, o campo *HTS address last seen* é uma informação que indica o momento em que um dispositivo com um determinado endereço HTS (seja ele IP ou MAC) foi visto pela última vez na rede. É usado para monitorar a presença de dispositivos na rede e para identificar dispositivos que não estão mais ativos. Esse campo é importante para o gerenciamento da rede, pois permite que o administrador de rede saiba quando um dispositivo sai da rede e, assim, tomar medidas para corrigir o problema.

Hosts

Host ID	Attachment point ID	Attachment point status	HTS address IP	HTS address MAC	HTS address last seen
host:da:cb:d8:3d:30:66	openflow:244353002188613:4	true	192.168.1.50	da:cb:d8:3d:30:66	5 Feb 2023 14:15:12
host:86:69:ff:15:18:3b	openflow:244353002188613:6	true	192.168.1.52	86:69:ff:15:18:3b	5 Feb 2023 14:30:21
host:bc:83:85:79:a5:9f	openflow:244353002188613:2	true	192.168.1.23	bc:83:85:79:a5:9f	5 Feb 2023 11:20:58
host:7e:ce:57:aa:17:9d	openflow:244353002188613:3	true	192.168.1.49	7e:ce:57:aa:17:9d	5 Feb 2023 15:3:41
host:f6:fa:f5:67:dc:b5	openflow:244353002188613:5	true	192.168.1.51	f6:fa:f5:67:dc:b5	5 Feb 2023 14:33:40
host:e0:1f:ed:ba:8d:88	openflow:244353002188613:2	true	192.168.1.254	e0:1f:ed:ba:8d:88	5 Feb 2023 11:20:13
host:f4:30:8b:92:e9:f5	openflow:244353002188613:2	true	192.168.1.12	f4:30:8b:92:e9:f5	5 Feb 2023 11:29:0

Figura 4.22 – Hosts do OFM.

4.4 Validação do funcionamento das regras

As próximas subseções fazem validação das regras propostas na seção 3.7.

4.4.1 Primeira Regra

Para validação da primeira regra configurada, responsável por bloquear todo o tráfego originado da rede interna para rede pública, foi executado um simples ping tanto no PC3 quanto no PC2. O primeiro direcionado ao DNS padrão do Google, 8.8.8.8, para testar conectividade com a internet pública; e o segundo, ao controlador.

Como mostram as figuras 4.23 e 4.24, tais tentativas foram frustradas. Isso mostra que, mesmo que o usuário tente, ele não poderá, em momento nenhum, ter acesso à internet para pesquisar a

resposta de determinada pergunta em sua prova, ou então se comunicar com um colega e também não conseguirá ter acesso ao controlador para modificar alguma regra de fluxo. Isso ajuda na prevenção de fraudes e garante a integridade das avaliações.

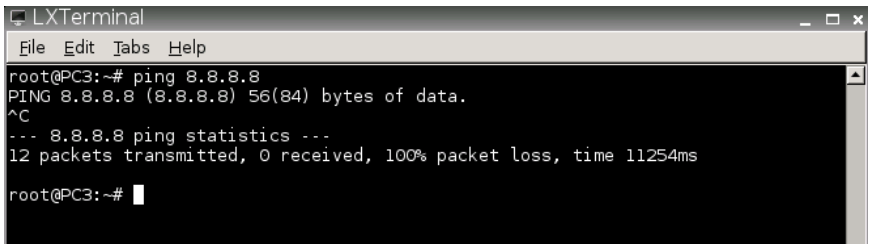


Figura 4.23 – Tentativa de ping do PC3 à internet pública.

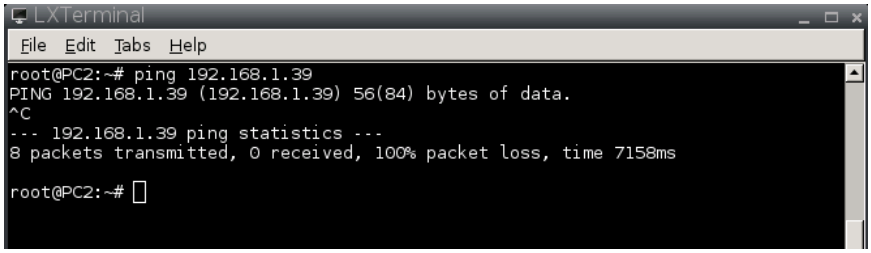


Figura 4.24 – Tentativa de ping do PC2 ao controlador.

4.4.2 Segunda Regra

A segunda regra, responsável por impedir a comunicação entre dispositivos dentro da sala de provas, foi validada por meio da utilização de três ferramentas: Ping, Nmap e SSH.

Primeiramente, como visto na figura 4.25, foi executado um ping entre PC1 e PC3, que foi corretamente bloqueado. Isso aponta um indício de que a comunicação entre dispositivos internos foi de fato bloqueada.

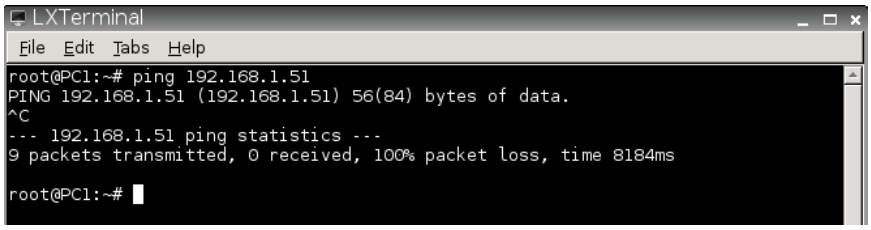


Figura 4.25 – Tentativa de ping entre PC1 e PC3.

Tendo a tentativa de ping frustada, o próximo teste realizado foi um Nmap destinado à rede da sala de provas. Para isso, executou-se o comando "# nmap -A -sV -sT 100 192.168.1.48/28". O primeiro parâmetro, "-A", indica que deve ser feita detecção de serviço e sistema operacional. O segundo parâmetro, "-sV", diz que, além da descoberta de serviço, deve também ser feita a descoberta de *daemon*. O último parâmetro, "-sT", habilita o escaneamento de saídas TCP.

Como resultado deste comando, visto na figura 4.26, nota-se que, apesar do usuário ser capaz de identificar os endereços MAC e IP dos dispositivos da rede, ele não consegue identificar nem

serviços ativos, nem sistemas operacionais. A regra impede a comunicação a nível da camada de rede, mas não impede a nível de camada de enlace. Então, por meio do protocolo ARP (Address Resolution Protocol), a descoberta é possível. O dispositivo envia pacotes ARP de descoberta para rede, e recebe as informações pedidas.

Isso mostra que, apesar de a regra criada ter sucesso no impedimento da comunicação, ela não impede a descoberta de dispositivos. Então, ainda é necessário criar uma regra que impeça esse tipo de comunicação ARP.

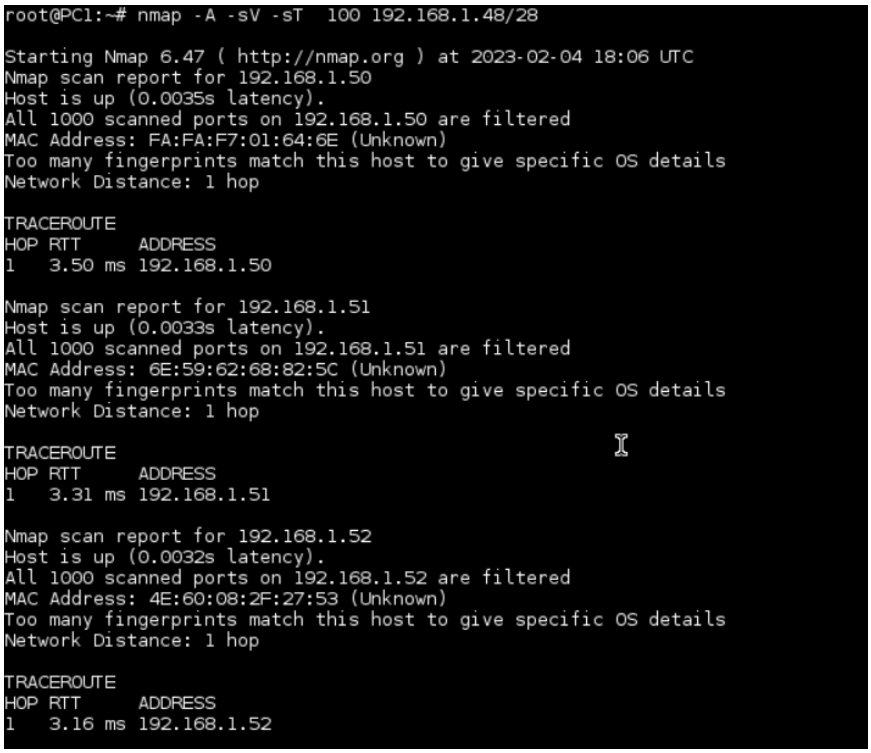


Figura 4.26 – Tentativa de Nmap do PC1 para a sub-rede da sala de provas.

Então, como mostra a figura 4.27, executou-se uma tentativa de conexão SSH entre o PC2 e PC4. Novamente, o teste foi corretamente bloqueado, sendo apresentada uma mensagem de *timeout* na tela, isto é, o PC2 nem mesmo conseguiu encontrar rota disponível para o PC4, como esperado.

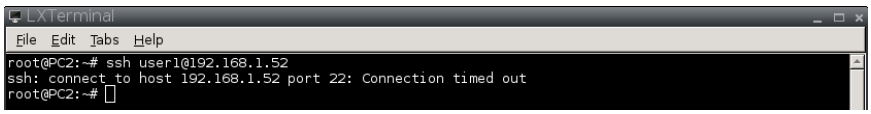


Figura 4.27 – Tentativa de conexão SSH entre PC2 e PC4.

Assim como a primeira regra, esta regra reforça a ideia de manter a integridade e a validade das avaliações, impedindo principalmente a comunicação entre alunos que estão realizando a mesma prova.

4.4.3 Terceira Regra

A validação da terceira regra foi relativamente mais complexa do que as demais, uma vez que esta regra é responsável tanto por permitir que apenas dispositivos com endereços MAC selecionados acessem o servidor de provas, como também limitar este acesso única e exclusivamente à porta 80.

O primeiro teste realizado foi o acesso do PC1 ao Moodle e, simultaneamente, a execução de um ping destinado ao mesmo servidor. Como mostrado na figura 4.28, o acesso ao Moodle foi realizado com êxito, enquanto o ping foi corretamente bloqueado. Isto indica que, de fato, o acesso está sendo permitido apenas à porta 80, impedindo que o usuário tenha sucesso em qualquer tipo de tentativa de enviar tráfego destinado a outras portas. Isso também reduz consideravelmente a superfície de ataque do servidor, uma vez que os únicos tipos de ataques que podem ser direcionados ao mesmo são os que se aproveitam de vulnerabilidades inerentes ao protocolo HTTP ou à porta 80.

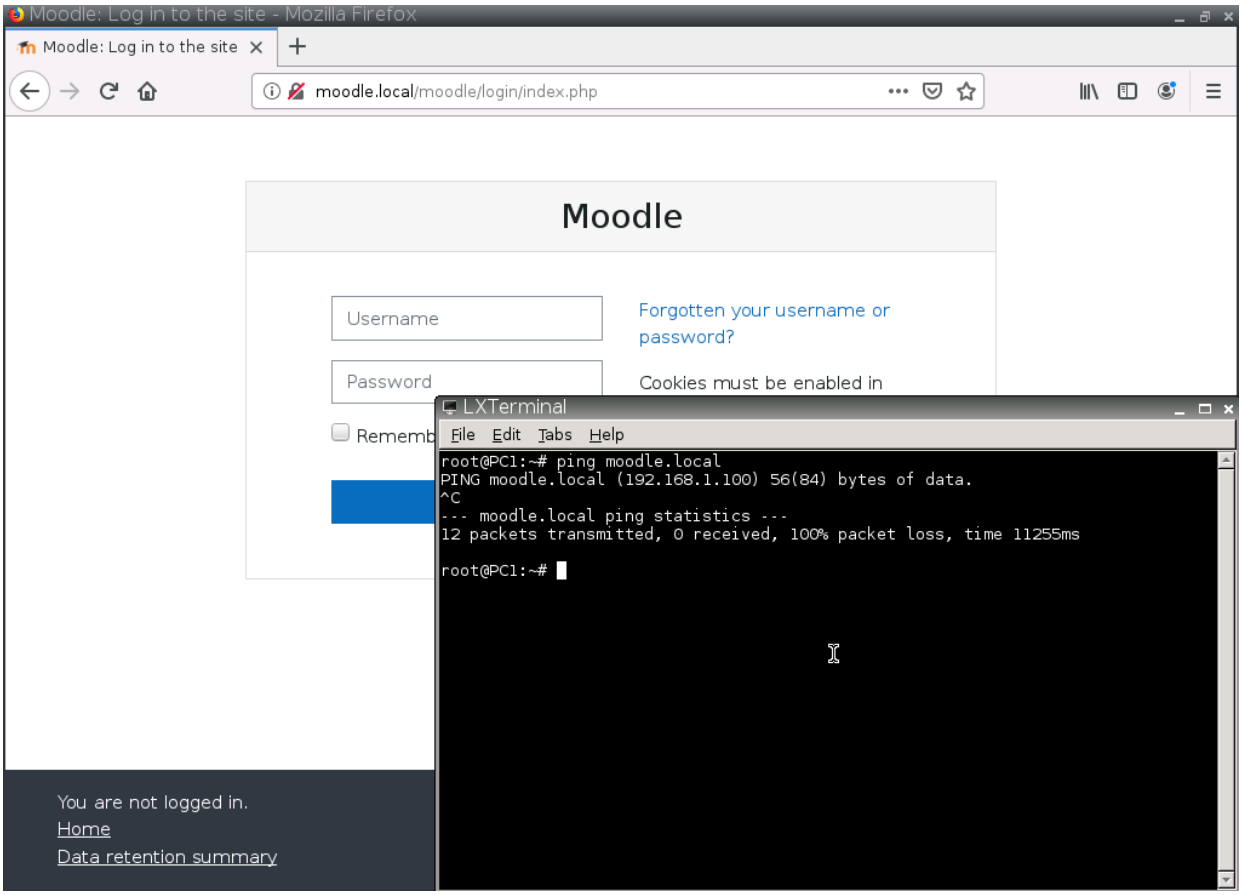


Figura 4.28 – Acesso ao Moodle realizado pelo PC1.

As figuras 4.29 e 4.30 reforçam o apresentado na imagem anterior. A primeira mostra os pacotes do ping entre PC1 e Moodle capturados pelo Wireshark; e a segunda, os pacotes do acesso à página Web da plataforma. Nas duas imagens, a janela à esquerda apresenta a captura do tráfego do enlace entre PC1 e OVS, enquanto a segunda janela mostra o tráfego entre OVS e Moodle.

A imagem 4.29, referente à tentativa de ping entre os dispositivos, mostra que o fluxo de

pacotes ocorreu normalmente entre PC1 e switch, entretanto, nota-se que os pacotes foram enviados mas não obtiveram resposta. Isso se justifica pela porção à direita desta mesma imagem, onde é apresentada uma ausência de comunicação. Mostra que, de fato, a comunicação está sendo interceptada pelo switch e o mesmo está tomando a decisão correta quanto ao seu encaminhamento.

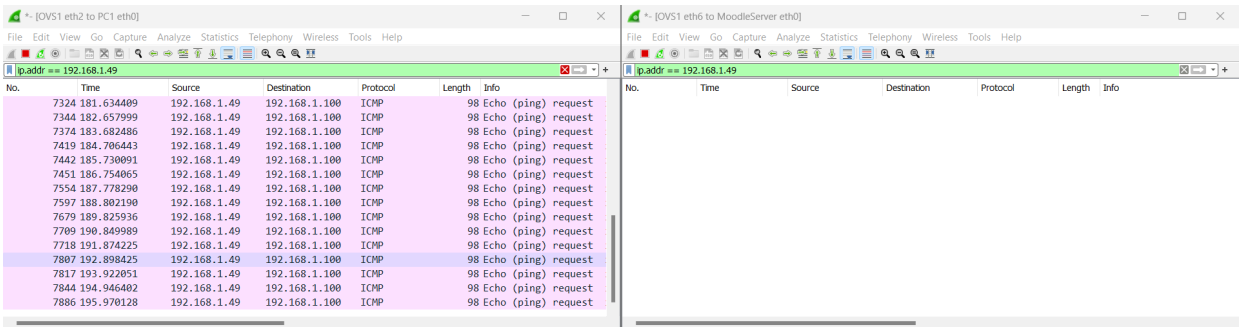


Figura 4.29 – Tentativa de ping entre PC1 e Moodle.

A imagem 4.30, referente à comunicação via HTTP entre os dispositivos, mostra que o fluxo de pacotes também ocorreu normalmente entre PC1 e OVS. Porém, desta vez, além de pacotes de requisição, há pacotes de resposta, o que indica que aqueles atingiram com sucesso o destino final. Isso é provado pela captura do tráfego ocorrido entre OVS e Moodle, em que os pacotes apresentados são exatamente os mesmos observados entre PC1 e OVS.

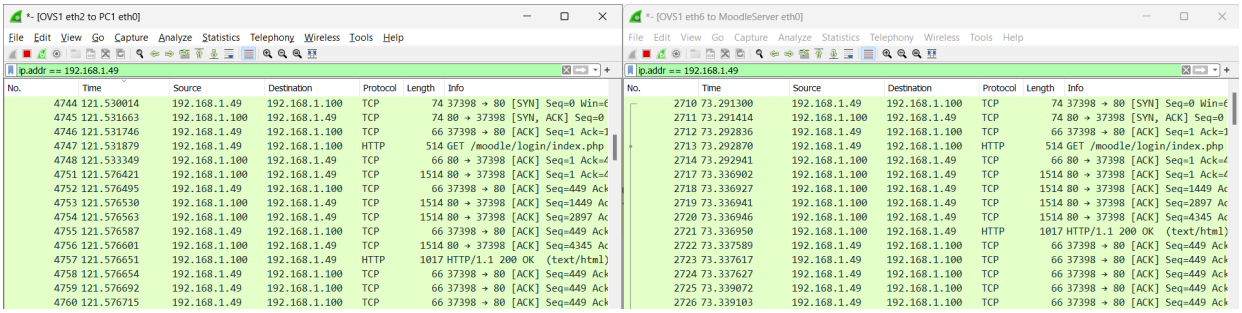


Figura 4.30 – Tráfego do PC1 para o Moodle, porém destinado à porta 80 (HTTP).

Por fim, o último teste realizado para comprovação desta regra foi a tentativa de acesso ao Moodle por um dispositivo cujo endereço MAC não está cadastrado nas tabelas de fluxo do Open vSwitch. Para isso, instalou-se um novo dispositivo Webterm na topologia, conectado ao OVS. Em seguida, o navegador deste dispositivo foi aberto e foi realizada a tentativa de acesso ao Moodle. Como mostrado na figura 4.31, esta ação foi corretamente bloqueada pelo switch, uma vez que ocorreu um time-out na comunicação.

Isso mostra que a execução de NAC (Network Access Control), por mais simples que seja, é fundamental para manter a segurança da rede, evitando que o indivíduo conecte qualquer dispositivo na rede e acesse as porções críticas. A implementação desta regra impede, por exemplo, que um aluno adentre à sala de avaliações, conecte seu computador pessoal à rede local e realize a prova, tendo acesso a seus arquivos pessoais e possíveis fontes de fraude.

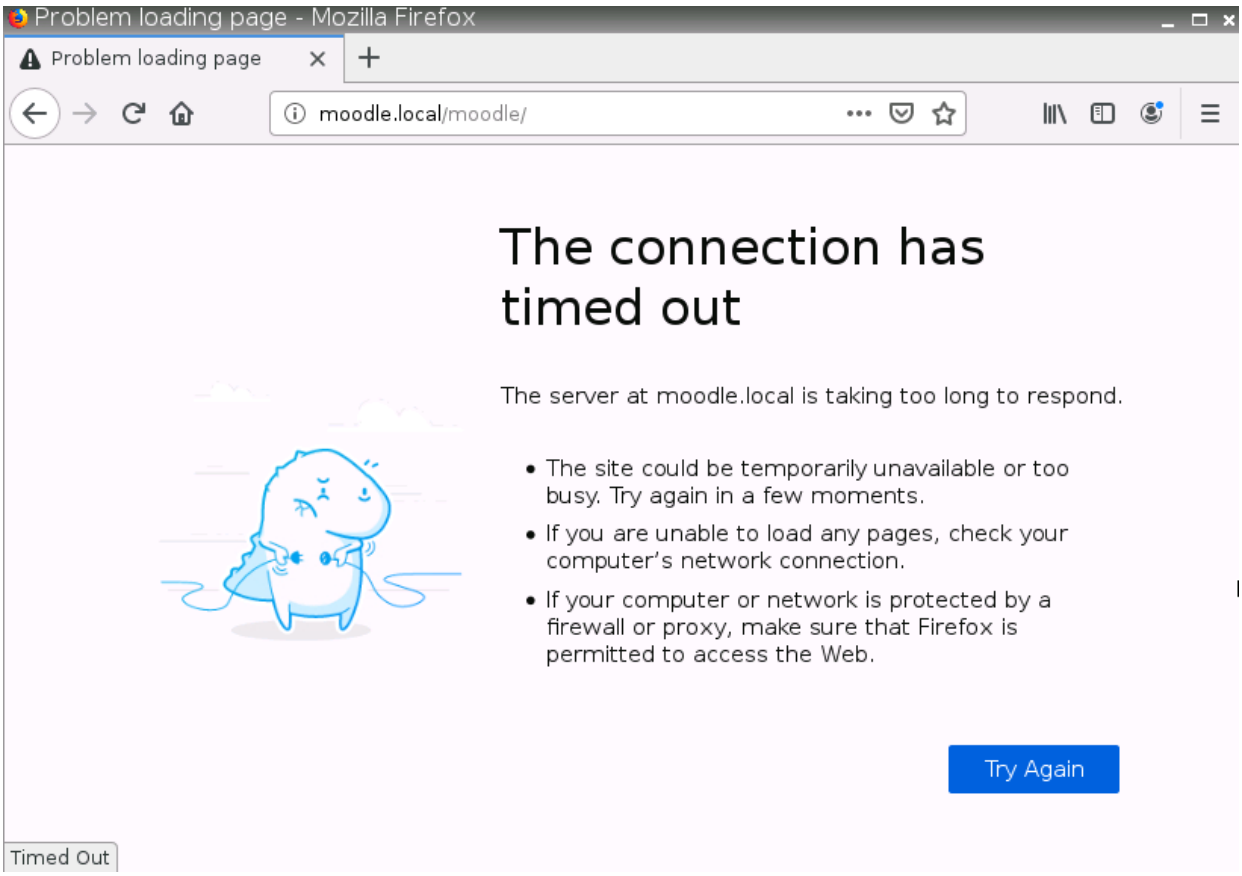


Figura 4.31 – Tentativa de acesso ao Moodle por dispositivo sem endereço MAC cadastrado.

4.4.4 Quarta Regra

O processo de teste da quarta regra, que objetiva liberar o acesso do PC de administração ao controlador, foi relativamente simples. Para isso, simplesmente criou-se uma nova regra pelo OFM, de prioridade 999, que bloqueia o endereço MAC do PC da administração ao controlador. Ou seja, criou-se um regra contrária à mesma, porém com prioridade inferior em uma unidade. Como esperado, o acesso do PC de administração ao controlador continuou ocorrendo normalmente, o que prova a efetividade dessa regra.

4.4.5 Quinta Regra

O teste da quinta regra, que tem a função de bloquear todo acesso vindo da rede externa, também foi bastante simples. Como a simulação é acessível a partir da rede física, acessou-se a página HTTP do OFM de um outro dispositivo da rede - outro que não seja o de administração -, e verificou-se que o acesso ocorreu normalmente.

Após isso, ativou-se a regra mostrada na figura 3.6 e testou-se novamente o acesso ao OFM, constatando-se que o acesso foi bloqueado de fato. Da mesma forma, também verificou-se a conexão do computador de administração - que também está localizado na rede externa à topologia - com o OFM e notou-se que o acesso continuou ocorrendo normalmente, o que mais uma vez prova a eficácia da quinta regra.

Esta regra incrementa a segurança da rede, uma vez que impede a entrada de ataques externos ou malwares na sala de provas, melhorando a segurança dos sistemas e dados. Além disso, auxilia na proteção de informações confidenciais, isto é, algumas provas eletrônicas podem conter informações confidenciais ou sensíveis. E a utilização de uma regra que impeça a entrada de dados ajuda a proteger essas informações contra vazamentos acidentais ou intencionais.

5 Conclusão e Trabalhos Futuros

Este trabalho propôs uma arquitetura SDN para disponibilização de um ambiente de provas seguro, à prova de fraudes e com um prático sistema de administração. Para isso, foram apresentados: os conceitos teóricos das tecnologias utilizadas; a arquitetura do projeto, junto com ilustrações e configurações do ambiente e de regras; e, por fim, a validação dos resultados.

Diversos softwares de código aberto foram utilizados para realização da proposta inicial, como o Moodle, Docker, SDN, Wireshark, entre outros. Eles foram utilizados de maneira integrada, então, o detalhamento de cada um foi importante para se ter uma compreensão de cada processo e etapa.

Para criar e validar a arquitetura, diversas configurações e escolhas de regras tiveram que ser feitas. No caso deste trabalho, cinco regras foram criadas para validar a proposta. Todas foram usadas juntas objetivando criar o ambiente seguro para realização de provas. As regras bloqueiam: a comunicação entre os computadores da rede interna do laboratório; o tráfego interno destinado à internet pública; e tráfego externo destinado à rede interna. E as outras regras autorizam: o acesso ao controlador pelo administrador; e o tráfego para o servidor Moodle.

Para analisar o tráfego e validar as regras propostas, prints do Wireshark, prompts de comando e navegadores web foram usados como ferramentas de comprovação das hipóteses. Os resultados mostraram que todas as regras funcionaram. Com isso, restringiu-se os acessos indevidos oriundos de fontes externas e uso de meios ilícitos - pelo bloqueio à internet pública e da comunicação entre alunos. Além disso, criou-se um ambiente configurável apenas por quem possui autorização.

Portanto, o objetivo inicial de gerenciamento e controle de acesso do laboratório foi atingido e validado por meio dos testes e análises de tráfego da rede. Como o trabalho pode sofrer complementações, algumas sugestões são feitas na seção 5.1.

5.1 Trabalhos Futuros

Uma sugestão de trabalho futuro é utilizar o software criado no trabalho [3] como o portal de provas, em vez de usar o Moodle. Então, utilizar as ferramentas de SDN e as outras tecnologias complementares para realizar o gerenciamento da rede. Essa era a ideia inicial deste documento, porém foi descartada por conta de alguns empecilhos.

Outro ponto que pode ser aprimorado é quanto ao tratamento de algumas das principais vulnerabilidades conhecidas do SDN, como ataques à camada de controle, ataques à virtualização e ataques direcionados à disponibilidade do controlador e do switch, não descartando também os ataques direcionados ao Moodle.

Pode-se também trabalhar com foco na segregação da rede por meio da implantação de Firewalls, VLANS e outras tecnologias, além de poder ser instalada uma ferramenta de monitoramento da rede, como o Zabbix.

Além disso, também é possível implementar algum tipo de solução NAC, tornando a autorização de acesso à rede melhor gerenciável, automatizando seu gerenciamento de acesso, compliance com regulamentos, melhoria da visibilidade e proteção contra ameaças.

Outra sugestão é a implementação das regras criadas neste trabalho em algum ambiente real. E, se possível, a adição de novas regras no ambiente.

Bibliografia

[1] Sithara HPW Gamage, Jennifer R Ayres e Monica B Behrend. "A systematic review on trends in using Moodle for teaching and learning". Em: *International Journal of STEM Education* 9.1 (2022), pp. 1–24.

[2] Manish Paliwal, Deepti Shrimankar e Omprakash Tembhurne. "Controllers in SDN: A review report". Em: *IEEE access* 6 (2018), pp. 36256–36270.

[3] João Paulo Pimentel. "Proposta de arquitetura de um portal seguro de provas eletrônicas utilizando SDN como solução de gerenciamento dos usuários da rede". Em: (2022).

[4] Jason C Neumann. *The book of GNS3: build virtual network labs using Cisco, Juniper, and more*. No Starch Press, 2015.

[5] GNS3. *Getting Started with GNS3*. <https://docs.gns3.com/docs/>. Acesso em 25/01/2023.

[6] Docker. *Docker Overview*. <https://docs.docker.com/get-started/overview/>. Acesso em 25/01/2023.

[7] Theo Combe, Antony Martin e Roberto Di Pietro. "To docker or not to docker: A security perspective". Em: *IEEE Cloud Computing* 3.5 (2016), pp. 54–62.

[8] Open vSwitch. *Open vSwitch Overview*. <https://www.openvswitch.org/>. Acesso em 26/01/2023.

[9] Ben Pfaff et al. "The design and implementation of open vswitch". Em: *12th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 15)*. 2015, pp. 117–130.

[10] ONF. *OpenFlow*. <http://www.opennetworking.org/sdn-resources/onf-specifications/openflow>. Acesso em 26/01/2023.

[11] VMWare. *What is Software-Defined Networking (SDN)?* <https://www.vmware.com/topics/glossary/content/software-defined-networking.html#:~:text=Software%2DDefined%20Networking-,What%20is%20Software%2DDefined%20Networking%20(SDN)%3F,direct%20traffic%20on%20a%20network.>. Acesso em 28/01/2023.

[12] Othmane Blial, Mouad Ben Mamoun e Redouane Benaini. "An overview on SDN architectures with multiple controllers". Em: *Journal of Computer Networks and Communications* 2016 (2016).

[13] Fetia Bannour, Sami Souihi e Abdelhamid Mellouk. "Distributed SDN control: Survey, taxonomy, and challenges". Em: *IEEE Communications Surveys & Tutorials* 20.1 (2017), pp. 333–354.

[14] Eder Leao Fernandes e Christian Esteve Rothenberg. "OpenFlow 1.3 software switch". Em: *Salao de Ferramentas do XXXII Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuıdos SBRC* (2014), pp. 1021–1028.

[15] Fei Hu, Qi Hao e Ke Bao. "A survey on software-defined network and openflow: From concept to implementation". Em: *IEEE Communications Surveys & Tutorials* 16.4 (2014), pp. 2181–2206.

[16] UFRJ. *OpenFlow*. <https://www.gta.ufrj.br/ensino/eel879/vf/openflow/>. Acesso em 26/01/2023.

[17] OpenDaylight. *Getting Started Guide*. <https://docs.opendaylight.org/en/latest/getting-started-guide/index.html>. Acesso em 27/01/2023.

[18] UFRJ. *Controlador OpenDaylight*. <https://www.gta.ufrj.br/ensino/eel879/trabalhos_vf_2018_2/opendaylight/>. Acesso em 27/01/2023.

[19] Jan Medved et al. "Opendaylight: Towards a model-driven sdn controller architecture". Em: *Proceeding of IEEE international symposium on a world of wireless, mobile and multimedia networks 2014*. IEEE. 2014, pp. 1–6.

[20] Cisco. *OpenDaylight-Openflow-App*. <https://developer.cisco.com/codeexchange/github/repo/CiscoDevNet/OpenDaylight-Openflow-App/>. Acesso em 28/01/2023.

[21] G. Henrick e K. Holland. *Moodle Administration Essentials*. Packt Publishing, 2015. ISBN: 9781784393182. URL: <https://books.google.com.br/books?id=h-8-CgAAQBAJ>.

[22] Moodle. *Moodle Docs 4.1*. <https://docs.moodle.org/401/en/Main_page>. Acesso em 28/01/2023.

[23] Sheo Kumar, Anil Kumar Gankotiya e Kamlesh Dutta. "A comparative study of moodle with other e-learning systems". Em: *2011 3rd International Conference on Electronics Computer Technology*. Vol. 5. IEEE. 2011, pp. 414–418.

[24] Wireshark. *Wireshark User's Guide*. <https://www.wireshark.org/docs/wsug_html/>. Acesso em 30/01/2023.

[25] J. Bullock e J.T. Parker. *Wireshark para profissionais de segurança: Usando Wireshark e o Metasploit Framework*. Novatec Editora, 2017. ISBN: 9788575225998. URL: <https://books.google.com.br/books?id=cSAuDwAAQBAJ>.

[26] Rodolfo Matos, Sofia Torrão e Tito Carlos S Vieira. "Moodlewatcher: Detection and prevention of fraud when using Moodle quizzes". Em: *Proceedings of INTED2012 Conference*. 2012.

[27] Rodolfo Matos e Jonathan Barber. "Moodlegate: Securing computer driven exam environments". Em: *INTED2013* (2013).

[28] Anderson H da Silva Marcondes et al. "SDN4Moodle: an SDN-based Toolset to Enhance QoS of Moodle Platform". Em: *2018 IEEE Symposium on Computers and Communications (ISCC)*. IEEE. 2018, pp. 00627–00632.

[29] Georges Daniel Amvame Nze. *SDN no GNS3 com MiniNet*. Departamento de Engenharia Elétrica, Universidade de Brasília. Acesso em 28/01/2023.

[30] Moodle. *Installing Moodle*. <https://docs.moodle.org/401/en/Installing_Moodle>. Acesso em 20/01/2023.

[31] ODL OpenFlowPlugin. *Flow Examples*. <https://docs.opendaylight.org/projects/openflowplugin/en/latest/users/flow-examples.html>. Acesso em 20/01/2023.

[32] Juniper Networks. *OpenFlow Flow Entry Timer Overview*. <https://www.juniper.net/documentation/us/en/software/junos/sdn-openflow/topics/concept/junos-sdn-openflow-flow-entry-timers-overview.html>. Acesso em 04/02/2023.

[33] ONF. *OpenFlow Switch Specification*. <https://opennetworking.org/wp-content/uploads/2014/10/openflow-spec-v1.3.0.pdf>. Acesso em 29/01/2023.

[34] Dana Hasan e Mohamed Othman. "Efficient Topology Discovery in Software Defined Networks: Revisited". Em: *2nd International Conference on Computer Science and Computational Intelligence 2017* (2017), pp. 1–9.

[35] PayZen. *Fase de autenticação*. <https://payzen.io/pt-BR/rest/V4.0/api/kb/authentication.html>. Acesso em 05/02/2023.

[36] Matt Conran. *OVS Bridge and Open vSwitch (OVS) Basics*. <https://network-insight.net/2015/11/21/open-vswitch-ovs-basics/>. Acesso em 05/02/2023.

Anexo A - Execução do contêiner Docker do Controlador.

Pré-requisitos (ambiente Windows):

- WSL2 (Windows Subsystem for Linux 2) e Docker Desktop instalados.
- Mininet rodando em Virtual Box, network do tipo Host-only.

Para criar o container:

```
# docker run -t -d -p “6633:6633” -p “6653:6653” -p “6640:6640” -p “8181:8181” -p “8101:8101” -p “9000:9000” –name odl-ofm caiocglo/opendaylight-ofm
```

Após usar o comando acima, o container já inicia, e o shell dele pode ser acessado. Após ligar ou reiniciar a máquina local, o container Docker deve ser iniciado pelo comando:

```
# docker start odl-ofm
```

Para abrir o shell do container:

```
# docker exec -it odl-ofm bash
```

Ao iniciar o container, o ODL e o OFM devem ser ligados manualmente — eles ficam na pasta /home.

Reiniciar o container:

```
# docker restart odl-ofm
```

Ver containers em execução:

```
# docker ps
```

Rodar mininet:

- Colocar como controlador: <IP do host>, porta 6633.

  Exemplo: sudo mn –controller=remote,ip=192.168.56.1,port=6633 –switch=ovsk, protocols=OpenFlow13 –mac –topolinear,4

Anexo B - Execução do contêiner Docker do Moodle.

Pré-requisitos (ambiente Windows):

- WSL2 e Docker Desktop instalados.

Para criar o container:

```
# docker run -t -d -p 80:80 -p 3306:3306 –name moodle caiocglo/moodlev2
```

Deve-se configurar o arquivo de Hosts da máquina de acesso ao Moodle para mapear o endereço IP do mesmo para o nome “moodle.local”. Esta configuração é necessária porque esta imagem do Moodle já foi pré-configurada para utilizar este nome. Chamar o servidor diretamente pelo endereço IP não irá funcionar.

Após usar o comando acima, o container já inicia, e o shell dele pode ser acessado. Após ligar ou reiniciar a máquina local, o container Docker deve ser iniciado pelo comando:

```
# docker start moodle
```

Para abrir o shell do container:

```
# docker exec -it moodle bash
```

Ao iniciar o container, o Moodle já inicia automaticamente e pode ser acessado.

Para acessar o Moodle, basta acessar no navegador:

```
http://moodle.local:80/moodle
```

Reiniciar o container:

```
# docker restart moodle
```

Ver containers em execução:

```
# docker ps
```