UNIVERSIDADE FEDERAL DE OURO PRETO

INSTITUTO DE CIENCIAS EXATAS E APLICADAS

DEPARTAMENTO DE ENGENHARIA DE COMPUTAÇÃO

RAQUEL MARINA DA CONCEIÇÃO

SISTEMA FIREWALL PARA O AMBIENTE ACADÊMICO DO INSTITUTO DE CIÊNCIAS EXATAS E APLICADAS – ICEA UTILIZANDO O PROTOCOLO OPENFLOW

Julho, 2018

João Monlevade – MG

RAQUEL MARINA DA CONCEIÇÃO

SISTEMA FIREWALL PARA O AMBIENTE ACADÊMICO DO INSTITUTO DE CIÊNCIAS EXATAS E APLICADAS – ICEA UTILIZANDO O PROTOCOLO OPENFLOW

Monografia apresentada ao curso Engenharia de Computação do Instituto de Ciências Exatas e Aplicadas, da Universidade Federal de Ouro Preto, como requisito parcial para aprovação na Disciplina “Trabalho de Conclusão de Curso II”.

Orientador: Erik de Britto e Silva

Co-orientador: Theo Silva Lins

Julho, 2018

João Monlevade – MG

C744s Conceição, Raquel Marina.
Sistema Firewall para o ambiente acadêmico do Instituto de Ciências Exatas e Aplicadas / ICEA utilizando o protocolo Openflow [manuscrito] / Raquel Marina Conceição. - 2018.

74f.: il.: color.

Orientador: Prof. MSc. Erik de Britto Silva.
Coorientador: Prof. MSc. Theo Silva Lins.

Monografia (Graduação). Universidade Federal de Ouro Preto. Instituto de Ciências Exatas e Aplicadas. Departamento de Computação e Sistemas de Informação.

1. Redes de computadores. 2. Firewalls (Medidas de segurança para computadores). I. Silva, Erik de Britto. II. Lins, Theo Silva . III. Universidade Federal de Ouro Preto. IV. Titulo.

Catalogação: ficha@sisbin.ufop.br CDU: 004.056.52

UNIVERSIDADE FEDERAL DE OURO PRETO
INSTITUTO DE CIÊNCIAS EXATAS E APLICADAS
COLEGIADO DO CURSO DE ENGENHARIA DE COMPUTAÇÃO

FOLHA DE APROVAÇÃO DA BANCA EXAMINADORA

SISTEMA FIREWALL PARA O AMBIENTE ACADÊMICO DO INSTITUTO DE CIÊNCIAS EXATAS E APLICADAS – ICEA UTILIZANDO O PROTOCOLO OPENFLOQ

Raquel Marina Conceição

Monografia apresentada ao Instituto de Ciências Exatas e Aplicadas da Universidade Federal de Ouro Preto como requisito parcial da disciplina CSI496 – Trabalho de Conclusão de Curso II do curso de Bacharelado em Engenharia de Computação e aprovada pela Banca Examinadora abaixo assinada:

Prof. MSc. Erik de Britto e Silva
DECSI – UFOP
Professor Orientador

Prof. MSc. Theo da Silva Lins
DECSI – UFOP
Professor Coorientador

Prof. MSc. Bruno Cerqueira Hott
DECSI – UFOP
Professor Convidado

MSc. Vinícius Fonseca e Silva
Pesquisador Convidado

João Monlevade, 13 de Julho de 2018

UNIVERSIDADE FEDERAL DE OURO PRETO
INSTITUTO DE CIÊNCIAS EXATAS E APLICADAS
COLEGIADO DE ENGENHARIA DA COMPUTAÇÃO

ATA DE DEFESA

Aos 13 dias do mês de Julho de 2018, às 16 horas e 20 minutos, na sala C 304 do ICEA, foi realizada a defesa de Monografia pela aluna Raquel Marina Conceição, sendo a Comissão Examinadora constituída pelos professores: Prof. MSc. Erik de Britto e Silva, Prof. MSc. Theo da Silva Lins, Prof. MSc. Bruno Cerqueira Hott e MSc. Vinícius Fonseca e Silva. A candidata apresentou a monografia intitulada: *"SISTEMA FIREWALL PARA O AMBIENTE ACADÊMICO DO INSTITUTO DE CIÊNCIAS EXATAS E APLICADAS – ICEA UTILIZANDO O PROTOCOLO OPENFLOW"*. A comissão examinadora deliberou, por unanimidade, pela aprovação do candidato, concedendo-lhe o prazo de 15 dias para incorporação no texto final das alterações sugeridas. Na forma regulamentar, foi lavrada a presente ata que é assinada pelos membros da Comissão Examinadora e pelo formando.

João Monlevade, 13 de Julho de 2018.

Prof. MSc. Erik de Britto e Silva
Professor Orientador/Presidente

Theo Silva Lins
Prof. MSc. Theo da Silva Lins
Professor Coorientador

Prof. MSc. Bruno Cerqueira Hott

Professor Convidado

MSc. Vinícius Fonseca e Silva

Pesquisador Convidado

Raquel Marina Conceição

Formanda

“*É possível encontrar a felicidade mesmo nas horas mais sombrias, basta se lembrar de procurar pela luz.”*
- J. K. Rowling

AGRADECIMENTOS

Agradeço primeiramente a Deus por não ter me deixado desanimar nos momentos mais difíceis que passei em minha vida.

Agradeço em especial aos meus pais por serem meu porto seguro, a minha fonte de força e sempre estarem ao meu lado me mostrando que a vida não é fácil, mas pode ser mais bonita de se viver. Dedico essa vitória a vocês, obrigada pela paciência.

Ao meu irmão Bruno, minha cunhada Nayara e meu primo Marcos, por terem a disponibilidade de me ajudar nos momentos mais difíceis no desenvolvimento deste trabalho.

Ao meu irmão, por Eletromagnetismo, por Análise e todas as outras matérias que fizemos juntos, obrigada por essa força.

A minha irmã Fabiana, por ter paciência e compreensão, obrigada você é um grande exemplo de força e persistência em minha vida.

Ao meu namorado Francismar, agradeço por estar presente em minha vida, por me compreender nos momentos difíceis e por fazer os meus dias mais felizes. Eu te amo.

Agradeço ao Erik meu Orientador e ao Theo meu Co-orientador, por me auxiliarem no desenvolvimento deste trabalho e me proporcionar uma nova experiência de conhecimento através do tema abordado.

Agradeço a todos os professores que passaram por minha vida escolar, aos professores do ensino fundamental, do ensino médio, do curso do pré-vestibular e principalmente aos professores da UFOP que me mostraram os caminhos básicos para ser uma profissional de qualidade.

Aos amigos do ICEA, por sempre estarem me incentivando a continuar, a dar sempre o meu melhor, a mostrar que nem tudo pode ser medido utilizando notas. Em especial aos meus grandes amigos da Visão Jr. vocês foram primordiais na minha vida, obrigada pelo carinho, as conversas e principalmente pelo aprendizado.

Aos meus amigos e familiares, pelas orações, conselhos, a paciência e principalmente a motivação dos últimos tempos.

A todos que de alguma forma contribuíram direta ou indiretamente para a conclusão deste trabalho, o meu muito obrigada!

RESUMO

A segurança de uma rede vai muito além das informações que trafegam pela mesma, é preciso garantir a confiabilidade nos dados através do software e do hardware que a compõe. Através do paradigma Redes Definidas por Software pode-se gerenciar melhor os switches, fazendo com que todas as tomadas de decisões sejam realizadas em um controlador e que os switches sejam responsáveis apenas pelo encaminhamento de dados. Esse trabalho utiliza-se de um sistema firewall para ser o sistema de controle responsável por gerenciar o fluxo de dados. Utilizou-se uma abordagem sistêmica da área, com aspectos teóricos e práticos. Na parte teórica, discutiram-se os componentes necessários para o desenvolvimento do sistema firewall. Na parte prática apresentou-se os resultados através do desenvolvimento de um protótipo do sistema firewall. Este trabalho possui o intuito apenas de desenvolver um protótipo acadêmico para a aplicação de regras no gerenciamento de fluxo. Utilizou-se simulações das aplicações das regras para ilustrar a utilidade do sistema firewall como um gerenciador e sendo assim pode-se desenvolver novas métricas de acordo com as necessidades da rede.

Palavras-chaves: Redes Definidas por Software, OpenFlow, Sistemas Firewall.

.

ABSTRACT

The security of a network goes far beyond the information that travels through it, it is necessary to guarantee the reliability in the data through the software and the hardware that composes it. Through the Software Defined Networks paradigm, you can better manage the switches, all decisions are taken by a SDN controller, and the SDN switches are responsible only for data routing. This work uses a firewall system as the control system responsible for managing the data flow. We used a systemic approach, with theoretical and practical aspects. In the theoretical part, the necessary components for the development of the firewall system were discussed. In the practical part the results were presented through the development of a prototype of the firewall system. This work intends only to develop an academic prototype for the application of rules in flow management. It was used simulations of the applications of rules to illustrate the usefulness of the firewall system as a manager, thus is possible to develop new metrics according to the needs of the network.

Keywords: Software Defined Networks, OpenFlow, Firewall Systems.

LISTA DE FIGURAS

LISTA DE ABREVIATURAS

ARP – Protocolo de resolução de endereços

DHCP – Protocolo de Configuração de Host Dinâmico

ICEA – Instituto de Ciências Exatas e Aplicadas

IP – Internet Protocol

ISPs – Internet Service Providers

LAN – Local Area Networks

MAC – Media access control

MPLS – Multi-protocol Label Switching

NTI – Núcleo de tecnologia da informação

OVS – Open vSwitch

RADIUS – Remote Authentication Dial In User Service

RFC – Request for Comments

ROM – Read-Only memory

SDN – Redes Definidas por Software

SDN – Software Defined Networking

SQL – Linguagem de Consulta Estruturada

TCAM – Ternary Content Addressable Memory

TCP – Protocolo de controle de transmissão

UDP – User Datagram Protocol

UFOP – Universidade Federal de Ouro Preto

WAN – Wide Area Network

SUMÁRIO

1. INTRODUÇÃO

A criação de instrumentos que proporcionam a maior facilidade na troca de informações sempre esteve em pauta ao longo da história, pois a comunicação é um elemento primordial das relações sociais. Superar fronteiras espaciais em benefício da maior velocidade de envio destas informações e a segurança das mesmas é objeto de constante busca por aperfeiçoamento, em função das inúmeras prerrogativas que ela nos proporciona. Quanto a isto, o período histórico que compreende a definitiva evolução tecnológica no campo da transmissão de informações ocorreu ao final do século XIX, transformações estas que influenciaram práticas cotidianas e ações políticas, econômicas e comerciais das nações de todo o globo.

Como exemplo, em meados do século XX, na Segunda Guerra Mundial (1939-1945), a segurança da transmissão de informações se tornou primordial para estratégia de Guerra. Neste período, grandes revoluções ocorreram principalmente na tentativa de quebra criptográfica de informações militares. Durante as décadas seguintes, a guerra fria e também os programas espaciais levaram as comunicações novamente a um novo nível, que hoje estão presentes em todas as transmissões de dados sejam elas financeiras, militares, governamentais e pessoais (TANENBAUM, 2013).

O crescente desenvolvimento destas tecnologias e a acessibilidade a elas fizeram com que nos dias atuais se discutisse a respeito dos usos e limites das tecnologias digitais, pois estão intrinsicamente relacionadas as atividades mais rotineiras. A maior ameaça à segurança virtual vem da tentativa de usuários que desejam enganar, roubar informações, até mesmo obter lucro financeiro ou manchar a credibilidade de meios considerados seguros, além de realizar ataques de cunho político, ideológico, religioso, entre outros (ZWICKY; COOPER; CHAPMAN, 2009). Devido a isso, é fundamental saber se as informações são confiáveis ou em quais plataformas podemos garantir a segurança desses dados (SOFKA, 2009).

Tanenbaum (2013) elucida que é praticamente impossível não analisarmos as redes quando o assunto a ser abordado é a tecnologia e a sua segurança, visto que, sempre se está em contato com vários tipos de usuários em maior ou menor grau de utilização, além do fato de que a Internet é um dos maiores sistemas desenvolvidos pelo homem, onde milhões de dispositivos estão conectados, tendo acesso a uma gama de informações em tempo real (KUROSE, 2010).

Desta forma, desenvolveu-se o sistema Firewall que é uma barreira de proteção que auxilia no bloqueio ao acesso a conteúdos maliciosos sem que os dados sejam impedidos de trafegar na rede (TORRES, 2013). Segundo Tanenbaum (2013) os Firewalls são uma adaptação digital para o esquema de segurança medieval, ou seja, são criadas barreiras ao redor do recurso desejado de modo que só exista um único ponto de acesso a tais recursos. Dessa forma consegue-se controlar todos os fluxos de entrada e saída, utilizando esse ponto de monitoramento.

Outros autores apresentam definição de Firewall como um dispositivo de proteção, com três pontos de risco: os dados, os recursos e a reputação do usuário ou proprietário do hardware ou software (COMER, 2016). Assim, a segurança é de interesse de todos os usuários, não importando qual é o seu nível de uso, visto que é primordial garantir a confiabilidade de todo o sistema (ZWICKY; COOPER; CHAPMAN, 2009).

1.1. Problema

O presente trabalho baseia-se na rede Ethernet do Instituto de Ciências Exatas e Aplicadas (ICEA) da Universidade Federal de Ouro Preto (UFOP), em João Monlevade. Este sistema é utilizado por servidores, professores e estudantes em funções administrativas e atividades de pesquisa desta instituição. O ICEA possui dois tipos principais de redes: a rede WI-FI e a rede cabeada. As redes WI-FI funcionam através de ondas de rádio transmitidas por meio de um adaptador, o roteador, que recebe os sinais, decodifica e os emite

a partir de uma antena (TORRES, 2013). As redes cabeadas ou redes Ethernet possuem como padrão a transmissão de dados utilizando a conexão de cabos cilíndricos que são conectados aos dispositivos (COMER, 2016). A rede via cabo é utilizada por um grupo restrito de pessoas que possuem um nível de acesso específico para utilizar esse recurso. Todos os procedimentos administrativos do campus são realizados por meio desse enlace e, dessa forma, percebe-se a necessidade de aumentar o nível de proteção desse sistema. Observou-se que tal rede possui fragilidades e deficiências que podem acarretar na invasão da rede, o que resultaria no comprometimento dos dados administrativos, acadêmicos e pessoais. Compreendendo a necessidade de suprir esta fragilidade, propõe-se por meio do sistema Firewall do protocolo OpenFlow (MCKEOWN, 2008) controlar uma rede SDN (Rede Definida por Software – Software Defined Networking) (NADEAU, GRAY, 2013) que gerencie os dados de forma a garantir a segurança de todos os usuários do sistema.

1.2. Objetivos

Analisando o problema apresentado em contraposição aos fins dados por esta pesquisa com a finalidade de desenvolver um protótipo Firewall para a proteção da rede Ethernet do ICEA, propõem-se os seguintes objetivos:

1.2.1 Objetivo geral

Criar regras para o gerenciamento de uma rede SDN no ICEA.

1.2.2. Objetivos específicos

Utilizar o software Firewall para aplicar as regras, com o intuito de gerenciar o fluxo de dados, desenvolvendo políticas de segurança, utilizando o

protocolo OpenFlow, propiciando aos usuários do sistema segurança e confiabilidade. No modelo atual em uso só existe o controle do fluxo da rede utilizando regras para o endereçamento MAC. Na arquitetura Ethernet cada porta de rede recebe um endereço físico que é gravado em uma memória ROM dentro do dispositivo. O endereço MAC consiste em seis bytes, os três primeiros identificam o fabricante da placa de rede e os três últimos são definidos pelo fabricante para seus dispositivos (TORRES, 2013). Os usuários que são cadastrados devem ter o número MAC do seu dispositivo inserido na tabela de autorização e sendo assim estarão aptos a utilizar a rede. Esse procedimento é pouco seguro para ser aplicado em uma rede onde há acesso à todas as informações administrativas.

Esse trabalho possui os seguintes objetivos específicos:

1- Revisar a literatura sobre as estruturas de Redes de Computadores, Redes Definidas por Software e protocolo *OpenFlow*.
2- Desenvolver um protótipo do sistema Firewall para gerenciar ao aplicar as regras de gerenciamento de fluxo.

1.3. Motivação

A rede Ethernet do campus do ICEA pode tornar-se mais segura e a utilização do sistema *OpenFlow* viabiliza não apenas o tráfego de dados seguros na rede, mas também o gerenciamento ativo da rede. O ambiente SDN pode futuramente servir como um analisador de dados e gerenciador de estatística. Por possuir em alguns casos código aberto, fornece liberdade para a construção de novos sistemas de gerenciamento. Assim, ele não depende de estrutura física para seu desenvolvimento e também não se limita a uma única e exclusiva linguagem de programação. Durante as pesquisas utilizou-se como ambiente de desenvolvimento para a aplicação proposta a plataforma Mininet (BHATIA, 2015). O Mininet é um emulador que emprega a virtualização em nível de processo. Essa ferramenta possui o compartilhamento de recursos e alcança uma maior escalabilidade em uma emulação completa. É necessária a

utilização de plataformas de simulação para situações reais que serão enfrentadas na rede, desta forma a rede não é afetada por problemas de segurança que podem aparecer durante os testes de funcionamento.

1.4. Metodologia

O estudo utilizará a base metodológica descritiva, que tende a apresentar, de forma detalhada, o desenvolvimento de um protótipo de sistema firewall utilizando o protocolo *OpenFlow* em um ambiente simulado. A metodologia será divida em três etapas, conforme apresentado a seguir:

Etapa 01: Busca e análise literária, nas seguintes áreas: Redes Definidas por Software, protocolo *OpenFlow* e Sistemas Firewall.

Etapa 02: Definição de requisitos para a implementação do protótipo.

Etapa 03: Simulação e avaliação dos resultados obtidos através do sistema desenvolvido.

1.5. Estrutura do trabalho

No capítulo 2 apresentado a seguir, é realizada uma revisão sobre todos os temas e ferramentas que serão utilizados, permitindo melhor entendimento do presente trabalho, além de servir como base para os resultados obtidos.

No capítulo 3 são apresentados os procedimentos básicos para a realização dos experimentos realizados na ferramenta MININET e no autenticador.

O capítulo 4 apresenta análises dos resultados, bem como as informações técnicas que foram utilizadas no desenvolvimento do protótipo apresentado. O capítulo 5 apresenta a conclusão do trabalho e também projeções e sugestões para trabalhos futuros.

2. CONCEITOS GERAIS E REVISÃO DA LITERATURA

Para o desenvolvimento do protótipo deste estudo, faz-se necessária uma análise das condições, protocolos, plataformas de simulação e linguagens, que permitirão o desenvolvimento de forma coesa e funcional. Um dos pontos centrais do projeto é a arquitetura Software *Defined Networking* – SDN (Redes Definidas por Software), o protocolo *OpenFlow*, o protocolo de configuração de host dinâmico – (DHCP) e o protocolo de resolução de endereços – (ARP) (PLUMMER, 1982). Utilizar-se-á um servidor *Radius* (RIGNEY, 2000) para realizar a conexão com o banco de dados e a autenticação do usuário. O banco de dados possui as seguintes informações: o nome do usuário, o seu número de CPF, o número MAC do dispositivo e a senha utilizada para efetuar o *login* na rede. O padrão de informações solicitadas condiz com as informações solicitadas no *login* da rede MINHA UFOP WIFI e com as informações atualmente solicitadas para a rede Ethernet.

2.1. Address resolution protocol (ARP) e media access control (MAC)

As redes utilizam o protocolo TCP/IP e o endereço IP para individualizar cada equipamento da rede. O endereço IP é atribuído à placa de rede e é alterado de acordo com as necessidades da rede onde o dispositivo está inserido, ou seja, o IP é o endereço que as aplicações utilizam para acessar a Internet ou outras redes (TORRES, 2013). As placas de rede utilizam o endereço MAC (endereço físico). Este endereçamento físico é associado à interface de comunicação, que conecta um dispositivo à rede, ou seja, esse endereço é individual para cada um dos dispositivos e é utilizado para controle do acesso à rede de computadores. O endereço MAC é gravado no hardware, na memória ROM da placa de rede dos equipamentos (TORRES, 2013).

Segundo Kurose (2010) o protocolo ARP é responsável por realizar a conversão entre os endereços IP e MAC da rede. Nos pacotes que transitam a

rede têm-se o endereço IP de destino, mas não o endereço MAC, sendo assim é necessário utilizarmos o ARP para realizar as devidas conversões.

2.2. Protocolo de configuração de host dinâmico – DHCP

O Protocolo de Configuração de Host Dinâmico (DHCP – Dynamic Host Configuration Protocol) fornece novos ajustes de parâmetros para a Internet. Esse protocolo possui dois principais componentes que são: um servidor DHCP para proporcionar configurações para um host e um protocolo para atribuição de endereços. Tal protocolo é constituído por um modelo cliente – servidor, onde o servidor esta encarregado de alocar endereços de rede para hosts e são realizadas entregas de parâmetros de configurações dinâmicas (DROMS, 1997).

Ainda segundo o autor acima, a definição para “servidor” refere-se a uma máquina ou dispositivo que inicializa os parâmetros utilizando o DHCP e o “cliente” refere-se ao solicitante de tais parâmetros. Em ambos os casos as definições estão relacionadas ao contexto descrito (DROMS, 1997).

2.2.1. Aplicação do DHCP na Universidade Federal De Ouro Preto – UFOP, Campus João Monlevade

O presente estudo teve como base a seguinte monografia: Desenvolvimento de um Sistema Web de Apoio ao Gerenciamento de Dispositivos da Rede do ICEA (CORDEIRO, 2015), onde se desenvolveu um software de um sistema Web de código aberto que realiza a automatização do servidor e a atualização de serviços ARP e DHCP. Utilizar-se-á a base do sistema desenvolvido que permite o cadastro de dispositivos computacionais utilizando o IP, MAC, tipo (Dinâmico ou Estático) e outros itens para a realização de tal tarefa.

Atualmente o sistema utilizado pelo núcleo de tecnologia da informação, entidade responsável pelo desenvolvimento e manutenção da rede do ICEA – NTI é o *Manager*. Este sistema realiza o cadastro de novos usuários fornecendo o acesso à rede cabeada do campus. Para cada tipo de usuário existem as permissões, ou seja, qual o tipo de acesso à rede que o usuário obterá.

2.3. Redes definidas por software – SDN

Segundo Guedes *et. al.* (2012), as redes definidas por software são um novo paradigma para desenvolvimento de novas pesquisas em redes de computadores. A arquitetura SDN faz a separação física do plano de controle de rede do plano de encaminhamento. Essa é uma arquitetura dinâmica, gerenciável, rentável e adaptável, ela desacopla as funções de controle e encaminhamento de rede, permitindo assim que o controle se torne diretamente programável e a infraestrutura seja abstraída para aplicativos e serviços de rede.

O intuito inicial das SDN era realizar um controle centralizado de regras utilizando um firewall para gerenciar o fluxo de dados em uma rede. Dessa maneira, qualquer switch/roteador passa a ser capaz de gerenciar e controlar os fluxos de dados que passa por ele (PETERSON, 2006; TENNENHOUSE, 2007). Além disto, o plano de dados consiste no caminho que cada pacote deve fazer, portanto, cada novo pacote que chega é analisado, se existir alguma regra na tabela de encaminhamento o dispositivo encaminha o pacote para a porta correspondente, se não existir uma requisição é enviada ao controlador, que pode adicionar uma nova regra na tabela de encaminhamento (GUEDES *et. al.*, 2012).

Desta forma, o plano de controle é o gerenciador das regras, onde se realizam as atualizações na tabela de encaminhamento e fazem as manutenções das estatísticas. Assim, ele é responsável por todo gerenciamento dos dados, de modo a analisar cada fluxo e gerencia o tempo

que a regra deve permanecer na tabela, ou quando um pacote não pode ser repassado para rede, seja internamente ou externamente (GUEDES *et. al.*, 2012).

2.3.1. Redes definidas por software: origem

A necessidade de desenvolver cada vez mais os dispositivos como: TVs, smartphones, computadores, desktops, tablets, entre outros componentes eletrônicos que estão interligados a Internet, aumentou-se a complexidade de problemas relacionados à área de redes de comunicação entre os dispositivos e a rede de computadores. Os problemas dessa área do conhecimento envolvem tanto a parte física quanto a parte lógica do equipamento (KUROSE, 2010).

O desenvolvimento de novas técnicas surge com as limitações físicas de recursos, por exemplo, no princípio do desenvolvimento da informática os computadores não necessitavam de conexões rápidas. Isso levou à necessidade do aprimoramento dos hardwares devido a um trafego de dados cada vez maior. Isto está ligado ao desenvolvimento das plataformas de software mais eficazes (TANENBAUM, 2003).

À medida em que se desenvolve novas plataformas de software a demanda de recursos físicos também aumenta. Assim, é necessário desenvolver o hardware para que seja possível realizar a utilização da plataforma criada. (TORRES, 2013).

Desta forma, o desenvolvimento da parte física da rede de computadores está em um processo de saturação, ou seja, têm-se limitações físicas nos dispositivos empregados atualmente e isso leva ao desenvolvimento de novas tecnologias para atender os requisitos necessários. Com o surgimento de novos dispositivos se obtém uma melhora no desempenho do ambiente, levando o melhor gerenciamento da rede (DIAS, 2014).

Entretanto, alguns equipamentos como *Bridges* e *Switches* estão sendo cada vez mais aprimorados para atender às demandas da rede (DIAS, 2014). A rede cabeada do ICEA utiliza os *Switches Ethernet,* também chamados de switches, desta forma, analisar-se-á este modelo de forma mais aprofundada, focando no encaminhamento de pacotes baseado no endereço MAC de cada dispositivo, ou seja, em uma rede de computadores cada *Switch* contém todos os endereços de todos os dispositivos ativos que estão ligados a ele.

Podemos comparar uma rede interna de computadores ligados a um *Switch* com uma transportadora de cartas. O remetente é o endereço MAC do dispositivo que enviou o pacote, o destinatário é o endereço MAC do equipamento que vai receber o pacote e o *Switch* é o "prestador do serviço" de entrega (TENNENHOUSE; WETHERALL, 2007).

Os Switches possuem um número fixo de portas onde se conectam todos os dispositivos presentes na rede. Cada endereço de cada máquina conectada ao switch é armazenado em uma tabela (DIAS, 2014). Na Figura 1, tem-se a tabela MAC de um *Switch Cisco 3560* (DIAS, 2014), onde são informados endereço MAC do computador e qual é a porta à qual os dispositivos estão fisicamente conectados.

Figura 1: Exemplo de uma tabela MAC de *Switch Cisco 3560*.

```
Switch# show mac address-table

          Mac Address Table
-------------------------------------------

Vlan    Mac Address       Type        Ports
----    -----------       --------    -----

   1    0002.16db.7b69    DYNAMIC     Fa0/3
   1    00d0.9725.4a8e    DYNAMIC     Fa0/2
   1    00d0.9725.4aee    DYNAMIC     Fa0/1
```

Extraído de: DIAS, 2014, p.9.

Para os *Switches* encaminharem pacotes com endereços utilizando IP, utiliza-se o protocolo ARP, realizando a comunicação entre as máquinas, para vincular o endereço da subrede, Rede Ethernet da mesma subrede IPv4, com o endereço MAC. A principal função do ARP é a tradução de endereço IP em endereço MAC (DIAS, 2014). A Figura 2 apresenta como as informações da tabela ARP ficam dispostas em um *Switch*, o que facilita o mapeamento dinâmico de endereços MAC para um IP.

Figura 2: Exemplo de uma tabela ARP de *Switch Cisco 3560*.

```
Switch# show ip arp

Protocol  Address          Age (min)  Hardware Addr   Type   Interface
Internet  192.168.1.1            -   0001.C7AC.67B8  ARPA   Vlan1
Internet  192.168.1.10           0   00D0.9725.4A8E  ARPA   Vlan1
Internet  192.168.1.11           0   0030.F246.0BC4  ARPA   Vlan1
Internet  192.168.1.30           0   0002.16DB.7B69  ARPA   Vlan1
```

Extraído de: DIAS, 2014, p.11.

As tabelas ARP e MAC podem ser consultadas quando necessário para a identificação do switch e/ou porta do dispositivo que recebeu ou enviou os pacotes. Podem ser utilizadas em diversos cenários, como por exemplo, identificar um servidor problemático ao utilizar a função *ping* nos endereços IP para rastrear a porta do equipamento com problemas (DIAS, 2014).

Para realizar qualquer tipo de configuração nos s*witches* comerciais é necessário aprender a linguagem vinculada ao equipamento, ou seja, cada fabricante de dispositivo fornece uma programação e/ ou linguagem utilizada, então cabe ao usuário identificar e familiarizar as peculiaridades, facilidades e dificuldades de cada linguagem (GUEDES *et. al.*, 2012).

Em uma grande sub-rede de máquinas, onde é necessária a utilização de mais de um *switch,* têm-se duas opções. A primeira é utilizar uma única marca de dispositivo, facilitando o aprendizado de apenas uma linguagem. A

segunda opção é a utilização de equipamentos de diversas empresas, e assim, será necessário aprender diversas linguagens. Entretanto, mesmo que se use equipamentos de um mesmo fabricante, podem existir diferenças nas linguagens utilizadas (DIAS, 2014).

Além da heterogeneidade na rede, existem outras dificuldades de se manter o encaminhamento de pacotes em alto desempenho, devido à limitação do hardware que influencia diretamente no direcionamento dos pacotes. Existem algumas iniciativas que propõem a conservação das operações, assim se mantém a viabilidade do desenvolvimento do hardware, porém com uma possibilidade de maior controle para os administradores da rede (GUEDES *et. al.*, 2012).

A proposta das redes SDN é inspirada na tecnologia de encaminhamento baseado em rótulos programáveis, popularizado pelo MPLS (*Multi-protocol Label Switching*). O MPLS propõe que o controle fino sobre o tráfego de rede se torne possível atribuindo um rótulo (*Label*) em cada pacote, e que essa informação exerça um controle diferenciado sobre o tráfego de rede (DAVIE & FARREL, 2008; KEMPF *et. al.*, 2011).

Seguindo essa proposta, a iniciativa mais bem sucedida foi a definição de interface e do protocolo *OpenFlow* (MCKEOWN *et. al.*, 2008). Com este protocolo, os elementos de encaminhamento oferecem uma interface simples de programação que possibilita estender o acesso e o controle da tabela de consulta utilizada pelo hardware, gerenciando o encaminhamento de pacotes, sendo assim possível determinar os próximos passos do dispositivo. A eficiência no encaminhamento de pacotes não é alterada, pois o switch continua utilizando a tabela de encaminhamento para realizar as consultas de endereços, porém, nesse modelo, a decisão sobre cada pacote deve ser processada por um controlador em um nível superior ao hardware (KEMPF *et. al.*, 2011).

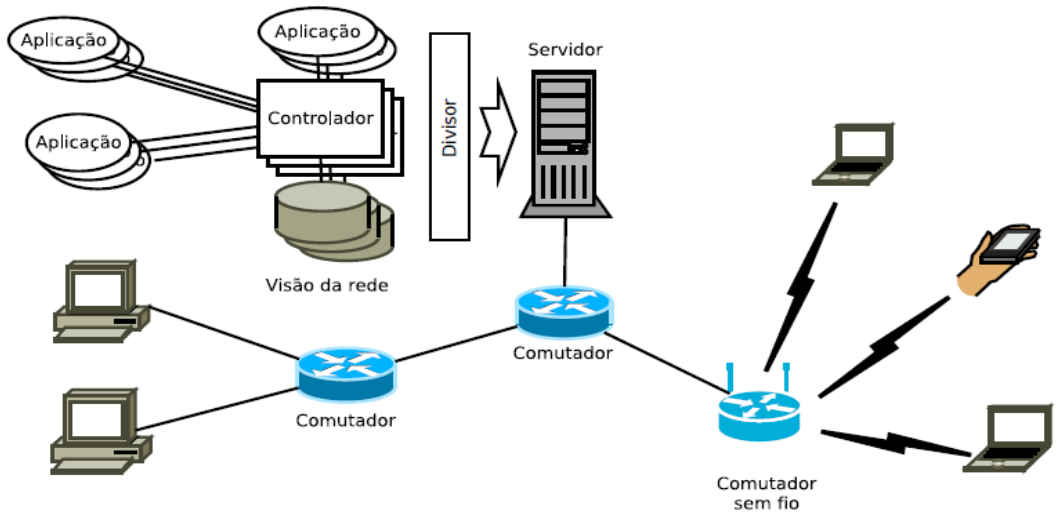
A vantagem da utilização do controlador sobre essas condições é que diferentes funcionalidades podem ser implementadas neste modelo. Essa

estrutura permite que a rede seja controlada de modo extensível através de aplicações expressas em software. A esse novo modelo deu-se o nome de Redes Definidas por Software (GUEDES *et al*, 2012).

2.3.2. Componentes de um sistema em SDN

Podemos dizer que uma Rede SDN é identificada pela existência de um controlador (software) que possui a função de gerenciamento do mecanismo de encaminhamento dos switches da rede, utilizando uma programação bem definida. O software utilizado é organizado com base em um controlador de aplicação geral, em torno do qual se constroem aplicações específicas para o fim de cada rede (GUEDES *et al*, 2012). Na Figura 3 é apresentada a estrutura básica de uma rede SDN.

Figura 3: Estrutura de uma Rede Definida por Software.



Extraído de: GUEDES *et. al*., 2012, p. 6.

2.3.3. Elementos de comutação programáveis

A programação dos elementos de rede é restrita à manipulação simples de pacotes baseados no conceito de fluxo, ou seja, na sequência de pacotes

que compartilham atributos com valores bem definidos. Além disto, a definição do que constitui o fluxo está diretamente relacionado aos recursos oferecidos pela interface de desenvolvimento (GUEDES *et al*, 2012).

O encaminhamento de pacotes utilizando os switches aplica um princípio simples: todo pacote é recebido pelo switch, e o dispositivo realiza uma consulta na tabela de encaminhamento. Se existirem regras referentes àquele tipo de pacote, ele realiza a ação determinada pela tabela. Se o pacote recebido não possuir referências na tabela o switch deverá repassar o pacote para o controlador (MCKEOWN *et. al*, 2008).

A aplicação de controle inspeciona e gera uma consulta à tabela de encaminhamento do switch, sendo assim, os próximos pacotes que possuírem atributos equivalentes ao avaliado serão encaminhados pelo switch somente utilizando a tabela de encaminhamento, sem que seja necessário o gerenciamento do controlador. A otimização da tabela de encaminhamentos é realizada pela programação das validades das regras, sendo assim possível determinar seus prazos de vencimento (ROTHENBERG *et. al*, 2011).

2.3.4. Controlador

Definindo a interface de programação para os elementos de chaveamento é possível desenvolver a aplicação que vai gerenciar cada switch separadamente. O problema dessa abordagem são as limitações associadas ao desenvolvimento de software, o qual é diretamente ligado a um dispositivo de hardware. Essa métrica exige uma programação de tarefas de baixo nível e o desempenho é dependente do hardware adotado (GUEDES *et. al*, 2012).

Para solucionar as limitações, foi identificada a necessidade de concentrar as tarefas de manipulação direta dos dispositivos de rede e oferecer uma abstração dos detalhes de baixo nível para que o programador possa desenvolver funções mais genéricas (PATEL, 2016).

Segundo Casado *et. al*. (2010), o controlador de rede pode concentrar a comunicação com todos os elementos programáveis que compõem a rede, oferecendo assim uma visão unificada do estado da rede. Além disto, esta metodologia de desenvolvimento abre uma nova possibilidade para o desenvolvimento de controladores descentralizados, para que, dessa forma, os problemas de centralização de funções sejam amenizados.

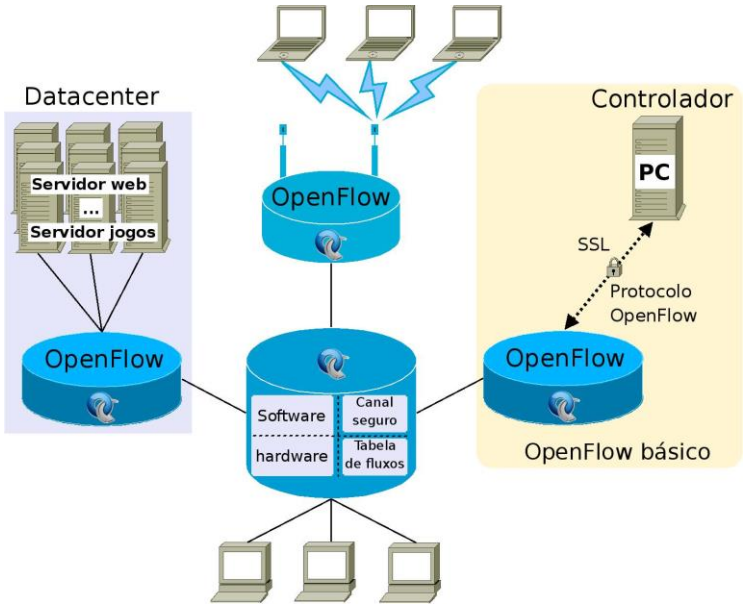
2.4. Openflow

O *OpenFlow* é um padrão ainda em desenvolvimento para a administração de redes LAN e WAN, com foco em equipamentos comerciais como *switches*, roteadores, *Access Points*, entre outros. O protocolo *OpenFlow* é responsável por incluir novas regras ou protocolos aos equipamentos a partir do plano de controle, deixando que a rede seja genérica a qualquer hardware ou software do *switch* (GUEDES *et. al*, 2012).

Segundo Rothenberg et al (2011), o *OpenFlow* foi inicialmente proposto pela Universidade de Stanford para atender à demanda de validação de novas propostas de arquiteturas e protocolos de rede. O *OpenFlow* define um protocolo-padrão para definir as ações de encaminhamento de pacotes em equipamentos de rede. As regras e ações são instaladas no hardware e são gerenciadas por um elemento externo denominado controlador (MCKEOWN et al, 2008), conforme ilustra a Figura 4.

A especificação *OpenFlow* busca reutilizar as funcionalidades do hardware existentes, através da definição de um conjunto de regras que serão realizadas por uma analise de cada ação, referentes ao comportamento e destino de cada pacote associado. Alguns exemplos de regras são: encaminhar, descartar, enviar para o controlador, reescrever campos do cabeçalho, entre outros (MCKEOWN et al, 2008)

Figura 4: Exemplo de uma rede com *OpenFlow* habilitado.



Extraído de: Rothenberg *et. al.*, 2011, p. 2.

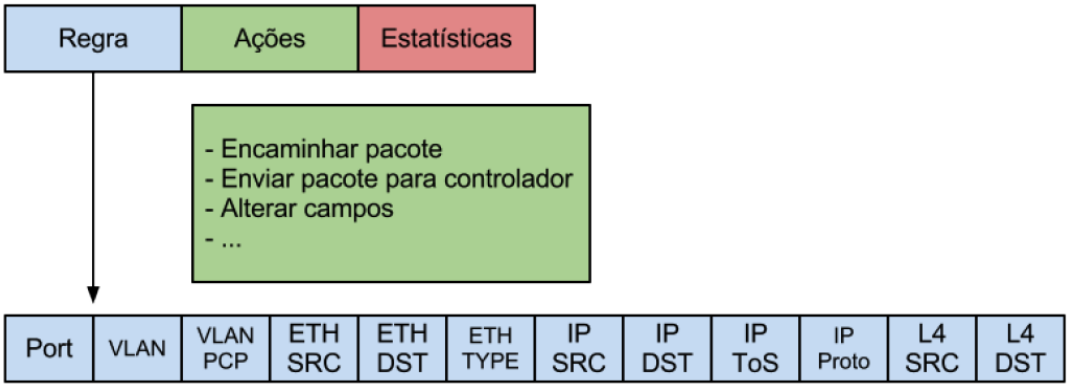
2.4.1. Estrutura do pacote de dados

A utilização do protocolo *OpenFlow* nos fornece possibilidades de programar novas regras para o tratamento do fluxo de dados. Segundo McKeown *et. al.* (2008), o fluxo pode ser definido como um conjunto de ações que formam uma entrada para a tabela de fluxos. Nos *switches* que utilizam o protocolo *OpenFlow*, adota-se o padrão de bits a cada entrada da tabela de fluxos armazenando em uma memória TCAM (*Ternary Content Addressable Memory*). Nas memórias TCAM convenciona-se os bits para representar se a função ou propriedade está ativa (bit um), se a função ou propriedade está desativada (bit zero) ou se a informação é irrelevante (*Don't care*) (MCKEOWN et al, 2008).

O padrão adotado é desenvolvido utilizando o plano de controle, que por sua vez é implementado com base nas informações escolhidas pelos programadores. A Figura 5 apresenta o padrão que pode ser produzido. As

informações em azul estão relacionadas às regras e às informações básicas de para a tabela de fluxo, em verde têm-se exemplos de ações que podem ser realizadas em cada pacote e em vermelho as estatísticas que podem ser obtidas através da captura de informações durante o processo de verificação dos pacotes.

Figura 5: Entrada na tabela de fluxos *OpenFlow*.



Extraído de: GUEDES *et. al*., 2012, p. 9.

A Figura 6 traz um exemplo de uma tabela de encaminhamento implementada em um firewall para realizar direcionamento dos pacotes da camada de enlace utilizando *VLANs*.

Temos presente na tabela de fluxo três exemplos de aplicação:

1. Caso o switch receba qualquer pacote com o endereço de saída 00:4F... deve ser encaminhado para a porta 4.
2. Caso o switch receba qualquer pacote TCP destinado à porta 22 exclua o mesmo do switch sem realizar o encaminhamento.
3. Caso o switch receba qualquer pacote da rede VLAN e com o endereço de saída 00:4F... deve ser encaminhado para as portas 4, 6 e 9.

Figura 6: Exemplo de uma tabela de fluxos em *Switch OpenFlow*.

Port	VLAN	ETH SRC	ETH DST	IP SRC	IP DST	IP Proto	L4 SRC	L4 DST	Ações
*	*	*	00:4F:...	*	*	*	*	*	**Porto 4**
*	*	*	*	*	*	**TCP**	*	**22**	**DROP**
*	**1**	*	00:4F:...	*	*	*	*	*	**Porto 4, 6, 9**

Extraído de: GUEDES *et. al*., 2012, p. 10

2.5. MININET

O *Mininet* é uma ferramenta de virtualização para simulação de topologias de redes de computadores. É um ambiente que já foi inserido o modelo SDN, sendo que o modelo básico é o *switch OpenFlow*. O *Mininet* permite a rápida prototipação de grandes redes utilizando apenas um computador para realizar as simulações, além de ser um emulador de rede, que cria uma rede de hosts virtuais, *switches*, controladores e links, sendo muito útil para testar as aplicações de uma SDN (BHATIA, 2015).

2.5.1. POX

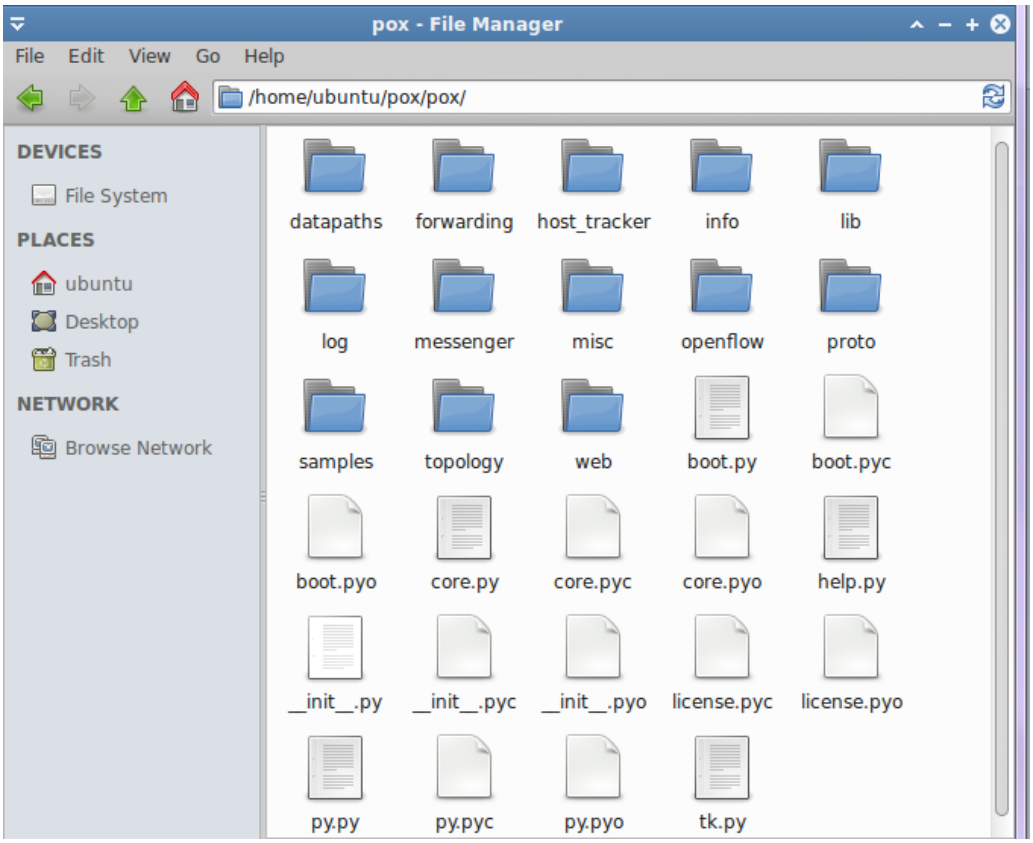
O POX, desenvolvido em *Python*, tem como objetivo utilizar as abstrações de alto nível de orientação a objeto oferecida pela linguagem, para obter uma interface mais simples para o programador. Para a utilização do POX faz-se necessário realizar a instalação de alguns recursos disponibilizados por desenvolvedores, encontrados em sites especializados (GUEDES *et. al*, 2012).

O POX é um dos controladores que podem ser utilizados nas SDN, na qual eles serão os responsáveis pela lógica de encaminhamento, através das regras que vão ser configuradas nos *switches OpenFlow*. A programação do controlador é orientada a eventos, ou seja, o programador determina o que vai ser realizado quando um novo pacote chegar ao *switch* (GUEDES *et. al*, 2012).

Portanto, têm-se dois modelos de programação. O primeiro deles tem código reativo, e quando recebe o pacote, envia o mesmo para o controlador tratar o conteúdo e assim é criado o evento. E o segundo possui código proativo, todas as regras são instaladas no *switch* assim que o mesmo é iniciado (KEMPF, 2011). Neste estudo, será empregado o modelo de conteúdo reativo.

O controlador já possui vários programas construídos dentro do diretório *forwarding* para serem utilizados como controlador de *switches OpenFlow*. Os diretórios são pastas existentes na máquina virtual onde todos os códigos que são executados no MININET estão guardados como é demonstrado na Figura 7. O POX pode ser utilizado para distribuição de protótipos, depuração de SDN, virtualização de rede, projeto do controlador e os modelos de programação (STANFORD, 2010).

Figura 7: Diretórios existentes no controlador POX do MININET.



2.6. PYTHON

Python é uma linguagem de programação desenvolvida por Guido Van Rossum em 1991, com o intuito de produzir códigos simples que tenham um fácil entendimento e que possam ser implementados de modo rápido e eficiente. Entre algumas características da linguagem pode-se ressaltar: o baixo uso de caracteres especiais, o que torna a linguagem muito parecida com *pseudocódigo executável*, a utilização de indentação para marcar blocos, quase nenhum uso de palavras-chave voltadas para a compilação, possui um coletor de lixo para gerenciar automaticamente o uso da memória, entre outras (PYTHON SOFTWARE FOUNDATION).

2.7. Servidor RADIUS

Um servidor RADIUS – *Remote Authentication Dial In User Service* – é um autenticador responsável por responder as requisições da rede, realizando a autorização ou não da inserção do suplicante, utilizando um banco de dados que irá fornecer as credenciais necessárias para realizar tal liberação de acesso (HASSEL, 2002).

O protocolo RADIUS consiste na autenticação, autorização e *accounting*, através do protocolo de transporte UDP (portas 1812 e 1813). A autenticação verifica a autenticidade do usuário, a autorização garante os recursos disponíveis apenas para os usuários permitidos e o *accounting* refere-se à coleta de informações sobre a utilização dos recursos disponibilizado. (RIGNEY, 2000).

2.7.1. Autenticação

Processo realizado para identificar um solicitante utilizando uma declaração de requisitos e fazendo a conferência de informações no banco de

dados para constatar a veracidade de tais métodos. Os métodos a serem utilizados no processo serão o MAC, IP e a senha da plataforma Minha UFOP.

2.7.2. Autorização

Procedimento aplicado para realizar a autorização e definir os tipos de permissões de cada solicitante. Existem políticas de segurança que devem ser documentadas para impedir que usuários acessem informações ou recursos inapropriados para suas respectivas categorias.

2.7.3. Accounting

Procedimento que realiza coletas de informações a respeito da utilização dos recursos. Este recurso é utilizado por ISP's (*Internet Service Providers*) que fornecem serviços de transferência de dados sob demanda para contabilizar o uso do serviço por seus usuários, possibilitando cobrança (HASSEL, 2002).

2.7.4. Banco de dados *MySQL*

Os bancos de dados são conjuntos de informações que possuem relações particulares entre si. Tem como principal finalidade reunir um conjunto de dados, de modo eficiente a fornecer tais métodos durante consultas, sendo operados utilizando sistemas gerenciadores de banco de dados. O *MySQL* é um sistema gerenciador, que utiliza a linguagem SQL (Structured Query Language) – *Linguagem de Consulta Estruturada* – como principal interface (ORACLE, 2017).

2.7.5. Protocolo *RADIUS*

Como descrito nos itens anteriores o protocolo *RADIUS* provê serviços de Autenticação, Autorização e Recenseamento utilizando uma base de dados centralizada de acordo com as definições estabelecidas na RFC 2866. O servidor *RADIUS* utiliza o protocolo UDP da camada de transporte para a troca de mensagens, com as aplicações clientes através da porta 1812 para Autenticação e 1813 para Autorização (HASSEL, 2002).

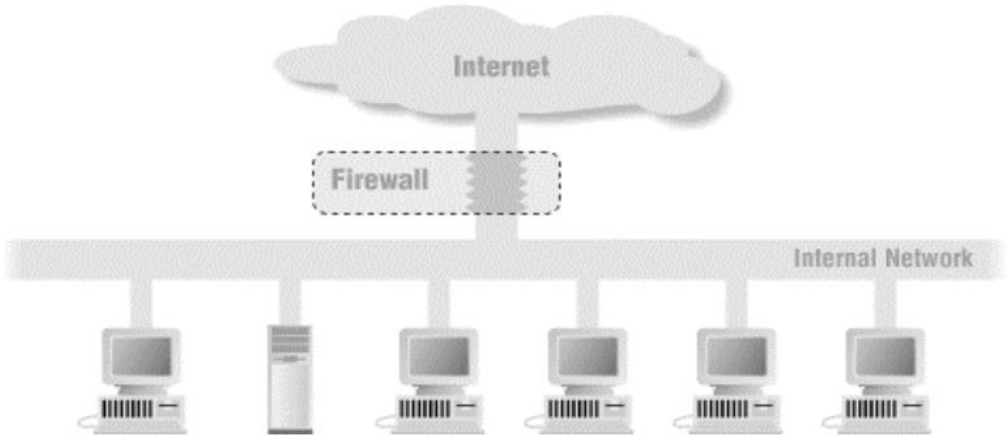
2.8. Sistema de Controle: Firewall

Firewall é um método de coibir o acesso de invasores externos à rede interna. Usualmente essa ferramenta é instalada em um ponto onde a rede interna é conectada à Internet. A existência de um firewall reduz consideravelmente as probabilidades de ataques de invasores externos, podendo impedir que os próprios usuários comprometam a segurança do sistema enviando informações confidenciais (ZWICKY; COOPER; CHAPMAN, 2009).

Além disto, este sistema de controle foi desenvolvido com o propósito de servir de barreira, através da inserção de diversas regras no sistema para garantir o máximo de segurança. Entretanto, nenhum firewall é totalmente impenetrável contra ataques de qualquer natureza, por isso são necessários a agregação de novas regras de segurança que visam aumentar a proteção, pois novas formas de burlar o sistema são desenvolvidas constantemente, fazendo com que a proteção também necessite de melhoras (ZWICKY; COOPER; CHAPMAN, 2009).

Pode-se utilizar este método de monitoramento para gerenciar o tráfego de dados da rede Ethernet do campus ICEA, realizando também uma pesquisa em cima dos pacotes de dados e filtrando as informações sobre o tráfego de rede, necessárias para criar estatísticas em plataformas de aplicações. A Figura 8 ilustra o posicionamento do firewall na rede.

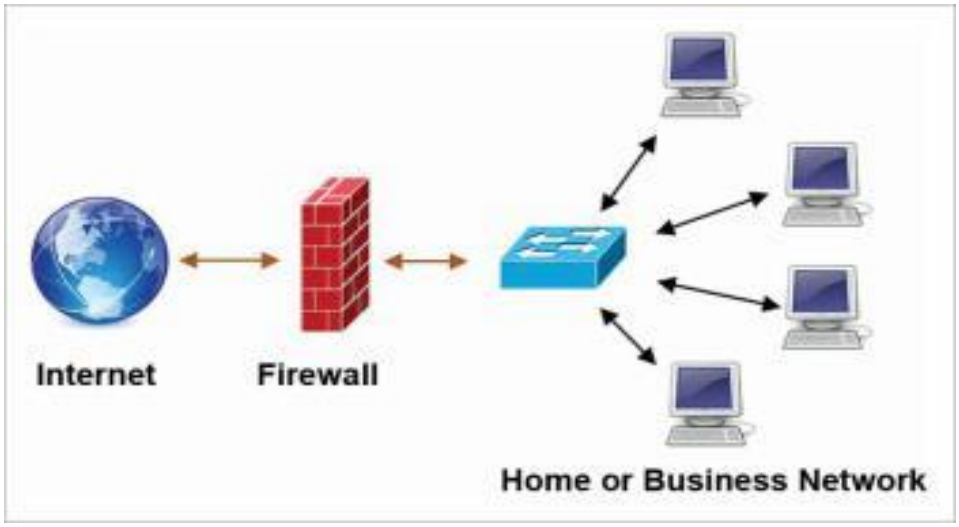
Figura 8: Firewall segurança entre duas redes.



Extraído de: ZWICKY; COOPER; CHAPMAN, 2009, p. 20.

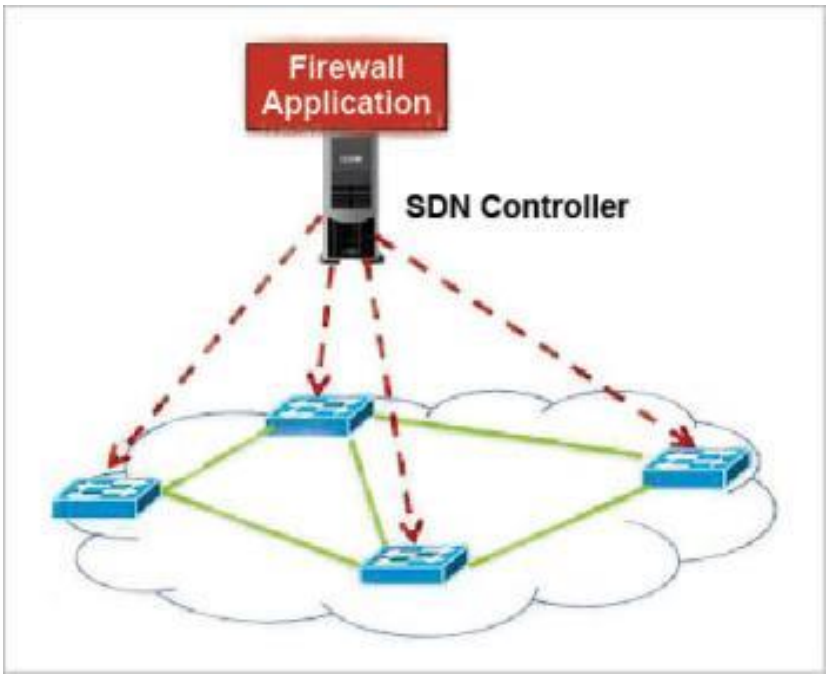
A arquitetura SDN é orientada a permitir aos administradores de rede gerenciar e controlar todo o sistema, através de um programa de software. Tendo, como propósito separar o plano de dados e o plano de controle, o que simplifica os serviços de rede (TANENBAUM, 2003). Nas Figura 9.a e Figura 9.b é apresentado um esboço de um sistema firewall em SDN.

Figura 9.a: Firewall baseado em uma arquitetura sem a utilização do SDN.



Extraído de: *Open Source*[1], 2017.

Figura 9.b: Firewall baseado em SDN.



Extraído de: *Open Source*[1], 2017.

[1] Disponível em: <http://opensourceforu.com/2016/07/implementing-a-software-defined-network-sdn-based-firewall/>

3. PROCEDIMENTOS BASICOS PARA OS EXPERIMENTOS

O protótipo elaborado consiste em um autenticador de usuário e um sistema firewall desenvolvido na arquitetura SDN. Existem protocolos de funcionamento que devem ser observados para a execução correta do sistema.

3.1. Procedimentos executados no MININET

Para o funcionamento correto do sistema de virtualização de rede desenvolvido na ferramenta MININET existem alguns procedimentos básicos que são realizados automaticamente, como a inicialização da topologia, e outros que são realizados manualmente no controlador, como a inserção das regras no controlador.

Para inicializar a topologia simples com quatro hosts utilizamos o comando:

$ sudo mn --topo=single,4 --mac --arp --controller=remote --switch=ovsk

Onde *topo=single,4* é uma topologia simples com apenas um switch e quatro hosts, *mac* é a solicitação do MAC, *arp* é a solicitação ARP, *controller* é o tipo de controlador e *switch* é o tipo de switch utilizado.

O POX suporta apenas a versão *OpenFlow 1.0* do OVS, então deve-se configurar estaticamente essa versão no switch utilizando o segundo terminal. O comando responsável para realizarmos essa configuração encontra-se abaixo:

$ sudo ovs-vsctl set bridge s1 protocols=OpenFlow10

É necessário inserir também as regras no controlador utilizando um novo terminal. Para inserir novas regras de controle utiliza-se o seguinte comando:

$ cd ./pox.py log.level --DEBUG REGRA_DO_FIREWALL

3.2. Autenticação dos Hosts

Após iniciar-se a topologia deve-se realizar a autenticação de cada um dos hosts utilizados, através do seguinte comando:

xterm h1 h2 h3 h4

Para abrir o terminal dos hosts 01, 02, 03 e 04. Com todos os terminais abertos utilizou-se o comando abaixo, para inicializar o servidor de autenticação de fluxo.

firefox http://192.168.56.101/autenticador/login.php

O identificador utilizado para o *host01=12345678901* e a *senha=host01*, o identificador utilizado para o *host02=12345678902* e a *senha=host02,* o identificador utilizado para o *host03=12345678903* e a *senha=host03* e o identificador utilizado para o *host04=12345678904* e a *senha=host04.* Com todos os hosts conectados pode-se realizar os procedimentos de verificação da regra.

4. APRESENTAÇÃO E ANÁLISE DOS RESULTADOS

O protótipo elaborado é um sistema firewall desenvolvido na arquitetura SDN para o ambiente acadêmico no campus do Instituto de Ciências Exatas e Aplicadas em João Monlevade, da Universidade Federal de Ouro Preto, utilizando o protocolo *OpenFlow*. Utilizou-se uma máquina virtual para simular a rede interna de computadores e o sistema firewall.

Como máquina virtual utilizou-se a ferramenta *Mininet*, que é uma plataforma que disponibiliza recursos para simular uma rede completa, ou seja, utilizando esse recurso é possível simular os hosts, switches, firewall, hubs, entre outros componentes de uma rede de computadores. Utilizou-se também o POX, que analisa cada novo fluxo na rede local e reconstrói o fluxo de aplicação para protocolos identificados como relevantes. Dentre eles estão: o SMTP (Simple Mail Transfer Protocol), HTTP (Hypertext Transfer protocol) e DNS (Domain Name System). A reconstrução só é encerrada quando o controlador possui informações suficientes para avaliar e aplicar o algoritmo para inserir uma nova regra na tabela de pacotes do switch (MEHD *et. al.*, 2011).

A rede Ethernet do ICEA possui como métricas de segurança a utilização de bloqueios de conteúdos pornográficos e/ ou páginas de conteúdo extremista e download de jogos (Plataforma Steam) para garantir que o tráfego de informações seja protegido de ataques externos. Porém, como toda grande rede de computadores, possui alguns pontos de fragilidade, como a baixa segurança na autenticação dos usuários na rede do campus, não existe uma atualização periódica automática das páginas de conteúdo inapropriado, entre outras fragilidades que devem ser melhoradas. O ponto de maior fragilidade em uma rede propicia a invasão de todo o sistema de segurança. Sendo assim, é necessário ter um cuidado maior com os pontos mais vulneráveis, pois através deles os invasores podem realizar grandes estragos em toda a estrutura de fluxo (ZWICKY; COOPER; CHAPMAN, 2009).

Visando aprimorar a segurança da rede e fortalecer os pontos de vulnerabilidade, foi desenvolvido o projeto de um Sistema Web de Apoio ao Gerenciamento de Dispositivos da Rede do ICEA (Cordeiro, 2015), que tinha como objetivo criar um sistema para automatizar o servidor e realizar as atualizações de serviços como ARP e o DHCP. Atualmente o sistema de internet é gerenciado por uma aplicação *Manager*, que automatizou o sistema de gerenciamento de informações aumentando assim a segurança da rede.

Apesar das contribuições adquiridas com o estudo, a rede interna do ICEA ainda possui alguns requisitos que podem ser melhorados para propiciar mais segurança a seus usuários. A segurança da rede Ethernet é baseada apenas no endereço MAC do usuário, ou seja, para conseguir utilizar a rede a cabeada do ICEA devemos realizar uma solicitação junto aos técnicos do NTI para que o número do MAC do host desejado seja incluído na tabela de MACs autorizados no sistema *Manager*.

Uma deficiência observada é que se um MAC autorizado for “clonado” qualquer usuário que tenha essa informação e que consiga configurar o MAC de sua interface com este endereço MAC já autorizado pelo sistema, pode ter acesso à rede mesmo que não esteja autorizado a utilizar esse tráfego. Recomenda-se que seja utilizado o serviço do servidor RADIUS, que já é utilizada na rede MINHA UFOP WIFI, para que ao realizar o cadastro do MAC do host do usuário seja cadastrado também número do CPF e a senha de acesso. Dessa maneira o usuário só vai ter acesso à rede Ethernet se também for autorizado através de autenticação pelo servidor. Na Figura 10 temos um esboço da página de *login* para a rede Ethernet.

Atualmente não existe um controle totalmente automatizado e centralizado na rede Ethernet do ICEA, o que dificulta a detecção e solução de problemas simples. Por exemplo, todos os laboratórios, por padrão, estão conectados a rede cabeada e sendo assim possuem acesso a Internet, quando é necessário, por algum motivo, desativar a rede para que os computadores não tenham acesso a Internet esse procedimento é realizado manualmente. Desse modo, analisando esse procedimento simples, podemos apenas

desenvolver regras para deixar a internet inativa em todos os computadores do laboratório por um tempo pré-determinado.

Figura 10: Página de *Login* para acesso a Rede Ethernet do ICEA.

Extraído e modificado de: zeppelin10.ufop.br/minhaUfop/

Para a realização dos experimentos na máquina virtual desenvolveu-se um servidor autenticador que possui um banco de dados onde se têm as principais informações da rede MINHA UFOP. Criou-se um banco com o nome “cadastro” onde se tem a tabela “usuários”. Os atributos que desejamos salvar neste banco de dados são: nome, CPF, MAC, a senha e o identificador. A Figura 11 apresenta-se a página de cadastro de novos usuários.

A Figura 12 ilustra a página de *Login*, inserindo o número de CPF e a senha é realizada a autenticação do usuário. Se as informações forem válidas é exibida a mensagem de sucesso, caso contrário é exibido à mensagem de falha na autenticação e o fluxo na rede não é autorizado.

Caso ocorra sucesso na autenticação do usuário, o mesmo é redirecionado para a página de navegação do Google e o usuário estará autorizado para utilizar à rede normalmente.

Figura 11: Página de Cadastro de novos usuários.

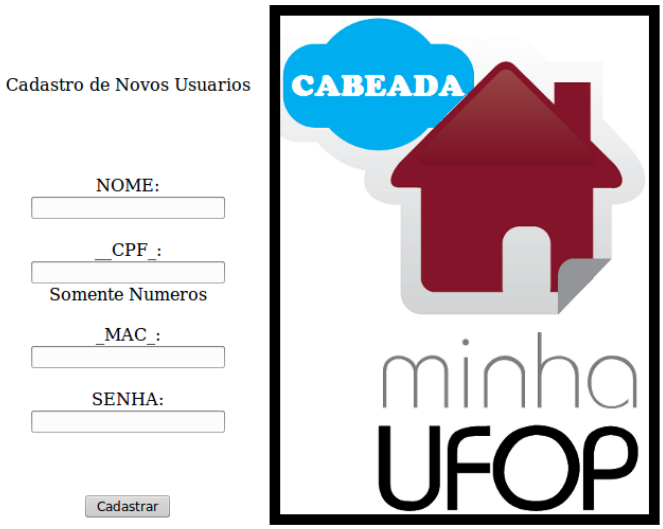


Figura 12: Página de *Login dos usuários*.

O sistema de gerenciamento também poderá fornecer informações sobre o tipo de fluxo de dados que é mais utilizado, o tempo de acesso médio, os horários de pico para a utilização da rede, entre outras informações que podem ser úteis de acordo com a área de pesquisa desejada.

A UFOP não possui regras definidas quanto às listas negras de conteúdos, ou seja, conteúdos que não devem ser acessados utilizando a rede da universidade. Porém, para um melhor gerenciamento do fluxo de dados e filtragem de conteúdos acessados, os técnicos do NTI sugerem que fossem implementadas regras de bloqueios para grupos específicos de conteúdos, como por exemplo, Torrents, arquivos de jogos da plataforma Steam, conteúdos pornográficos, entre outros itens que não condizem com um ambiente acadêmico.

Anexou-se ao sistema Firewall a *Black List* com os principais sites com conteúdos inapropriados. A Black List utilizada realiza o bloqueio de sites de pornografia, anúncios e outros sites de *malware* apontando os endereços de IP dos respectivos sites para 127.0.0.1 no arquivo hosts. A lista foi atualizada no dia 15 maio de 2018.

Outro ponto importante são os períodos de tempo em que há muitos usuários utilizando a rede. Durante esse intervalo de tempo, o fluxo de informações é constante e se a rede não for bem administrada o alto tráfego causa lentidão e/ ou perda de informações. Para tentar minimizar os efeitos de um tráfego intenso pode-se anexar regras ao sistema firewall para que os fluxos de dados dos usuários possuam um tempo de vencimento, ou seja, se o usuário não estiver utilizando o recurso por um período de tempo determinado ele será removido do sistema e suas regras de tráfego de dados nos switches deverão ser apagadas para diminuir o cache das tabelas de pacotes.

Além disto, pode-se ainda, com o auxílio do POX, gerar relatórios, experimentos e/ou estatísticas analisando o fluxo da rede local. Os dados obtidos através da análise do tráfego de dados serão confidenciais e de total responsabilidade da UFOP. Eles serão administrados apenas pelos gerenciadores responsáveis pela rede. Assim, o desenvolvimento um sistema protótipo de firewall realizará o gerenciamento do fluxo de dados na rede do Campus ICEA em João Monlevade.

O software desenvolvido no presente trabalho terá como função realizar o controle do tráfego de dados em uma rede simulada, o que auxiliará para

uma melhor compreensão do conteúdo, contribuindo assim para a realização de algumas alterações para a adaptação da rede convencional ao padrão requerido – protocolo *OpenFlow*. Além disso, altera-se o gerenciamento das conexões entre os *switches* e as tabelas implementadas de fluxo do *OpenFlow*.

Cada novo fluxo capturado e processado é inserido em uma regra de conexão na tabela dos switches. As tabelas foram atualizadas para receber informações de novos fluxos e inicializar o contador para a verificação de usabilidade do fluxo, ou seja, enquanto existir tráfego constante vai ser mantida a regra na tabela. Assim, quando houver um tempo maior que o definido e não existir fluxo de dados a regra é removida da tabela.

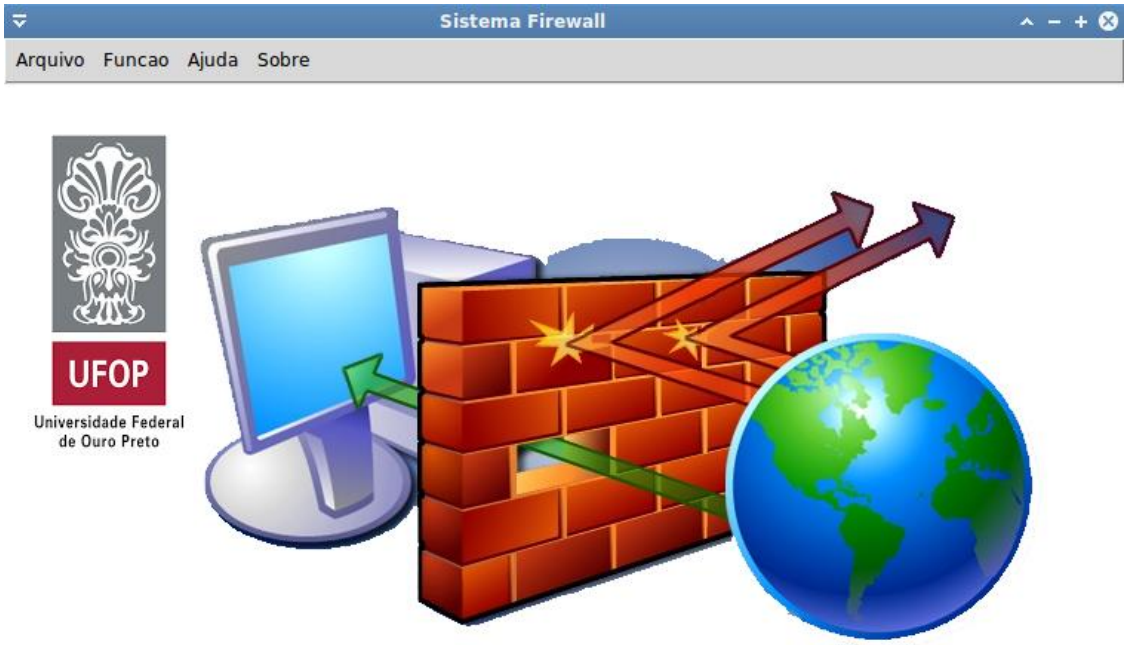
4.1. Desenvolvimento do sistema Firewall em python.

Para facilitar o gerenciamento das funções desenvolvidas utilizando a ferramenta POX, criou-se uma interface para auxiliar na aplicação das regras de controle do sistema Firewall utilizando a linguagem python. Na Figura 13 temos a janela de inicialização da interface do sistema Firewall.

Para realizar as simulações devemos utilizar a opção “Função” do sistema Firewall, nesse menu temos as quatro funções que foram desenvolvidas utilizando a plataforma POX, são elas: Bloqueio MAC, Gerenciamento por Tabela, Encaminhamento por IP e Gerenciador de Fluxo. Ao selecionar a opção desejada, é iniciado um terminal onde é gerada a topologia da rede e deve-se abrir um novo terminal de comando para inserir as regras no controlador.

[2] Disponível em: <http://someonewhocares.org/hosts/>

Figura 13: Janela Inicial do Sistema Firewall.



O Mininet gera uma nova topologia de rede informando os seguintes comandos: tipo de topologia (simples, linear ou árvore), número de *switches* e número de hosts, solicitação do MAC de cada elemento, solicitação do protocolo ARP, tipo de controlador que será utilizado e por fim qual será o tipo de *switch*.

O gerenciamento de conteúdo na rede Ethernet é realizado através de ações que vão ser executadas no controlador. O controlador utilizado é o firewall, logo tais ações devem ser executadas e desenvolvidas no mesmo. Este presente estudo vai apresentar algumas ações práticas de gerenciamento, como: bloqueador de MAC, gerenciamento de endereços de entrada e saída utilizando duas tabelas, encaminhador de pacotes com base em endereços IP e o gerenciador de fluxo.

4.2. Ação de bloqueio individual para endereços MAC.

$ cd ./pox

$ python pox.py --verbose regra1 py log --no-default --file=/tmp/mylog.log

Nas Figura 14.a, Figura 14.b, Figura 14.c e Figura 14.d, a seguir, temos a simulação da ação de bloqueio individual para endereços MAC. Na Figura 14.a temos a utilização da função Bloqueio MAC responsável por inicializar o comando para a geração da topologia no terminal de controle. Na Figura 14.b tem-se a topologia adequada com um switch e quatro hosts sendo apresentada no terminal de comando, depois inseriu-se as regras no controlador fazendo a conexão com a topologia criada Figura 14.c. Na Figura 14.d utilizou-se o comando *help* para exibir a lista de comandos disponíveis para serem realizados nos testes.

Figura 14.a: Utilização da função Bloqueio MAC.

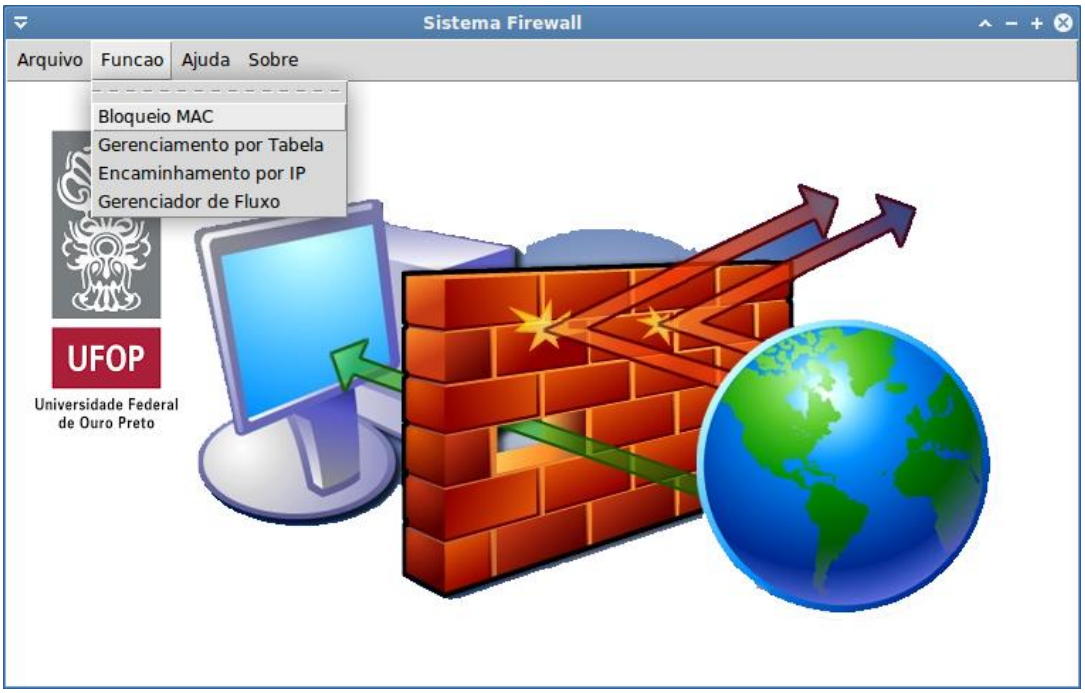


Figura 14.b: Topologia com um switch e quatro hosts. Terminal 1 – Topologia da Rede.

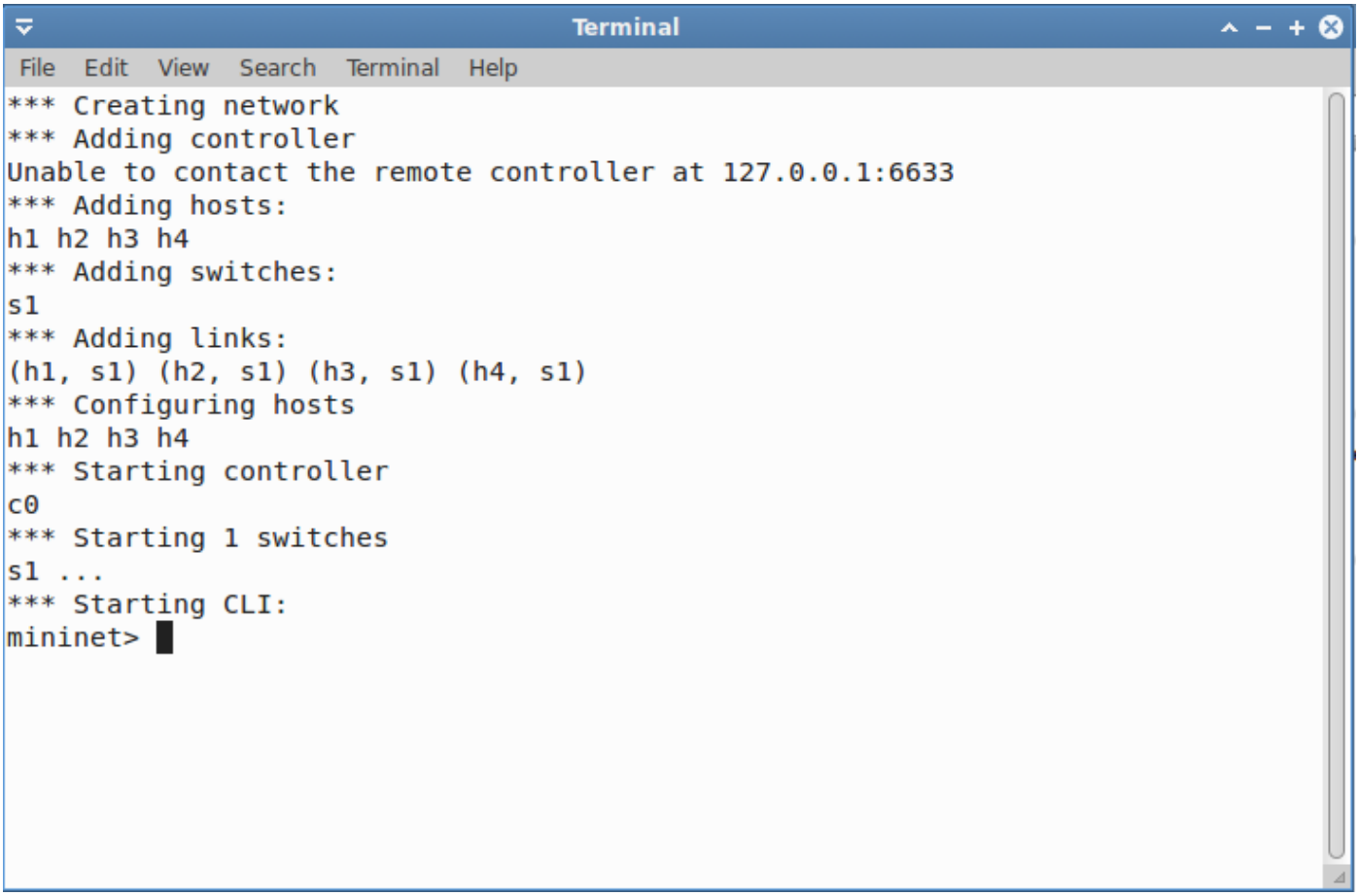
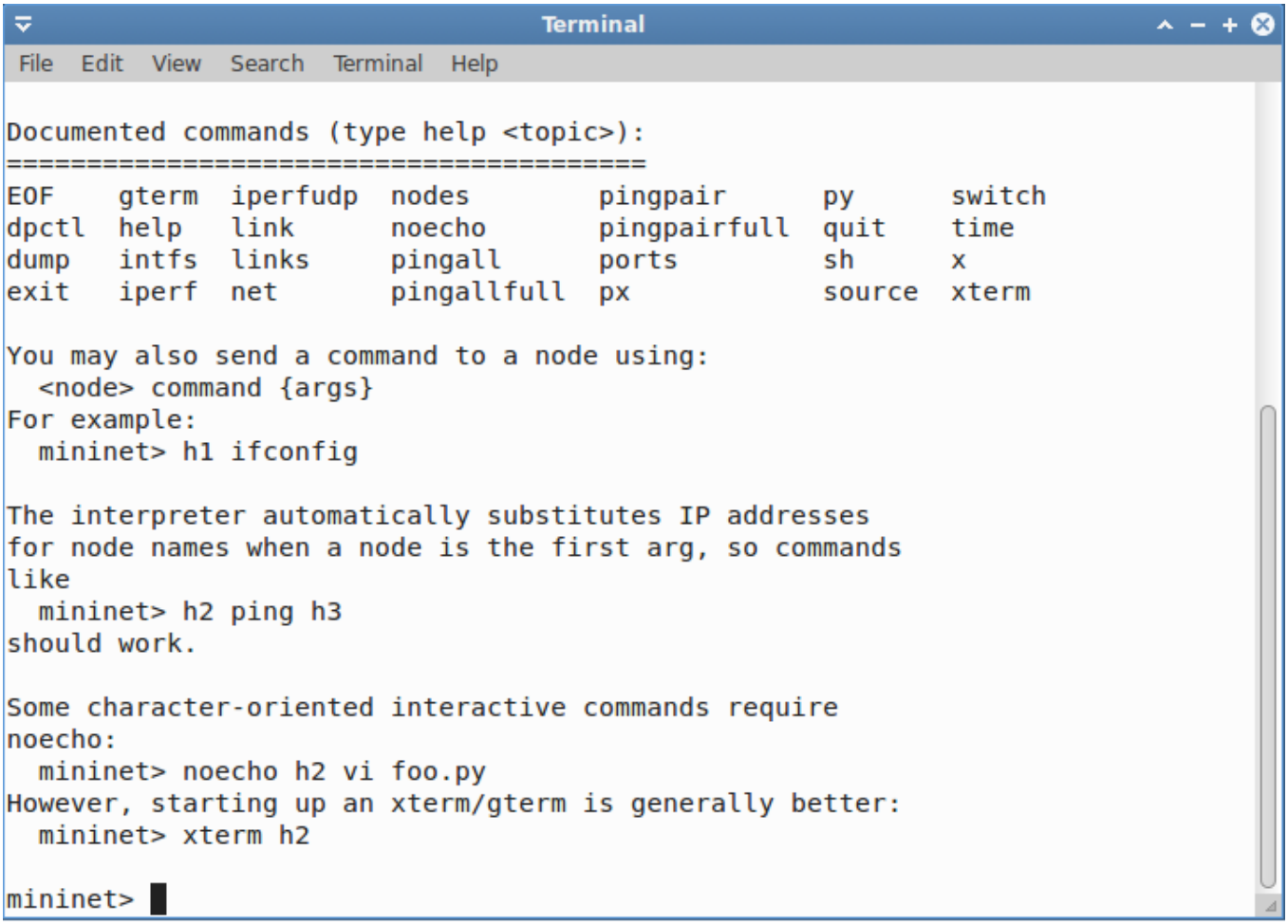


Figura 14.c: Regras do controlador. Terminal 2 – Configurações do controlador.

```
ubuntu@sdnhubvm:~$ sudo ovs-vsctl set bridge s1 protocols=OpenFlow10
ubuntu@sdnhubvm:~$ cd pox
ubuntu@sdnhubvm:~/pox$ python pox.py --verbose regra1 py log --no-default --fil
e=/tmp/mylog.log
POX 0.5.0 (eel) / Copyright 2011-2014 James McCauley, et al.
Ready.
POX>
```

Figura 14.d: Aplicação do comando *help*. Terminal 1 – Funções disponíveis para teste.



Nas Figura 15.a, Figura 15.b.1 e Figura 15.b.2 temos a conferência da função aplicada pelo controlador. Neste passo, utilizou-se o comando *pingall* para verificar se existe uma comunicação entre os hosts da rede. Na Figura 14.a, utilizou-se a regra do controlador de modo que nenhum host está conectado. Em seguida enviou-se *pings* para os demais hosts, entretanto nenhum vai responder.

Figura 15.a: Aplicação da regra de bloqueio de MAC em todos os hosts

```
mininet> pingallfull
*** Ping: testing ping reachability
h1 -> X X X
h2 -> X X X
h3 -> X X X
h4 -> X X X
*** Results:
 h1->h2: 1/0, rtt min/avg/max/mdev 0.000/0.000/0.000/0.000 ms
 h1->h3: 1/0, rtt min/avg/max/mdev 0.000/0.000/0.000/0.000 ms
 h1->h4: 1/0, rtt min/avg/max/mdev 0.000/0.000/0.000/0.000 ms
 h2->h1: 1/0, rtt min/avg/max/mdev 0.000/0.000/0.000/0.000 ms
 h2->h3: 1/0, rtt min/avg/max/mdev 0.000/0.000/0.000/0.000 ms
 h2->h4: 1/0, rtt min/avg/max/mdev 0.000/0.000/0.000/0.000 ms
 h3->h1: 1/0, rtt min/avg/max/mdev 0.000/0.000/0.000/0.000 ms
 h3->h2: 1/0, rtt min/avg/max/mdev 0.000/0.000/0.000/0.000 ms
 h3->h4: 1/0, rtt min/avg/max/mdev 0.000/0.000/0.000/0.000 ms
 h4->h1: 1/0, rtt min/avg/max/mdev 0.000/0.000/0.000/0.000 ms
 h4->h2: 1/0, rtt min/avg/max/mdev 0.000/0.000/0.000/0.000 ms
 h4->h3: 1/0, rtt min/avg/max/mdev 0.000/0.000/0.000/0.000 ms
```

Realizou-se a autenticação dos host 01 e host 02 utilizando os procedimentos descritos no capítulo 3.2.

Na Figura 15.b.1 é mostrada a tabela de fluxos atualizada, ou seja, com o fluxo entre os hosts autorizados e por fim, na Figura 15.b.2 temos a confirmação do fluxo apenas entre os hosts autenticados. A inserção das regras para os hosts autenticados ocorre no controlador e sendo assim o fluxo é liberado apenas para os hosts autenticados.

Figura 15.b.1: Tabela de Fluxo

```
mininet> sh ovs-ofctl dump-flows s1
NXST_FLOW reply (xid=0x4):
 cookie=0x0, duration=4319.928s, table=0, n_packets=5, n_bytes=450, idle_age=53
, in_port=4,dl_src=00:00:00:00:00:04,dl_dst=33:33:00:00:00:16 actions=drop
 cookie=0x0, duration=4319.891s, table=0, n_packets=1, n_bytes=78, idle_age=55,
 in_port=3,dl_src=00:00:00:00:00:03,dl_dst=33:33:ff:00:00:03 actions=drop
 cookie=0x0, duration=4319.667s, table=0, n_packets=1, n_bytes=78, idle_age=54,
 in_port=4,dl_src=00:00:00:00:00:04,dl_dst=33:33:ff:00:00:04 actions=drop
 cookie=0x0, duration=4319.630s, table=0, n_packets=5, n_bytes=450, idle_age=53
, in_port=3,dl_src=00:00:00:00:00:03,dl_dst=33:33:00:00:00:16 actions=drop
 cookie=0x0, duration=4319.511s, table=0, n_packets=0, n_bytes=0, idle_age=4319
, in_port=2,dl_src=00:00:00:00:00:02,dl_dst=33:33:ff:00:00:02 actions=drop
 cookie=0x0, duration=4319.225s, table=0, n_packets=1, n_bytes=78, idle_age=55,
 in_port=1,dl_src=00:00:00:00:00:01,dl_dst=33:33:ff:00:00:01 actions=drop
 cookie=0x0, duration=4319.222s, table=0, n_packets=5, n_bytes=450, idle_age=54
, in_port=2,dl_src=00:00:00:00:00:02,dl_dst=33:33:00:00:00:16 actions=drop
 cookie=0x0, duration=4319.204s, table=0, n_packets=5, n_bytes=450, idle_age=53
, in_port=1,dl_src=00:00:00:00:00:01,dl_dst=33:33:00:00:00:16 actions=drop
 cookie=0x0, duration=4318.880s, table=0, n_packets=5, n_bytes=350, idle_age=45
, in_port=3,dl_src=00:00:00:00:00:03,dl_dst=33:33:00:00:00:02 actions=drop
 cookie=0x0, duration=4318.709s, table=0, n_packets=5, n_bytes=350, idle_age=45
, in_port=4,dl_src=00:00:00:00:00:04,dl_dst=33:33:00:00:00:02 actions=drop
 cookie=0x0, duration=4318.486s, table=0, n_packets=5, n_bytes=350, idle_age=46
, in_port=2,dl_src=00:00:00:00:00:02,dl_dst=33:33:00:00:00:02 actions=drop
 cookie=0x0, duration=4318.212s, table=0, n_packets=5, n_bytes=350, idle_age=46
, in_port=1,dl_src=00:00:00:00:00:01,dl_dst=33:33:00:00:00:02 actions=drop
 cookie=0x0, duration=9.478s, table=0, n_packets=0, n_bytes=0, idle_age=9, in_p
ort=1,dl_src=00:00:00:00:00:01,dl_dst=00:00:00:00:00:02 actions=output:2
 cookie=0x0, duration=8.148s, table=0, n_packets=0, n_bytes=0, idle_age=8, in_p
ort=2,dl_src=00:00:00:00:00:02,dl_dst=00:00:00:00:00:01 actions=output:1
```

Figura 15.b.2: Aplicação da regra de fluxo entre o host 01 e host 02

```
mininet> pingallfull
*** Ping: testing ping reachability
h1 -> h2 X X
h2 -> h1 X X
h3 -> X X X
h4 -> X X X
*** Results:
 h1->h2: 1/1, rtt min/avg/max/mdev 0.306/0.306/0.306/0.000 ms
 h1->h3: 1/0, rtt min/avg/max/mdev 0.000/0.000/0.000/0.000 ms
 h1->h4: 1/0, rtt min/avg/max/mdev 0.000/0.000/0.000/0.000 ms
 h2->h1: 1/1, rtt min/avg/max/mdev 0.294/0.294/0.294/0.000 ms
 h2->h3: 1/0, rtt min/avg/max/mdev 0.000/0.000/0.000/0.000 ms
 h2->h4: 1/0, rtt min/avg/max/mdev 0.000/0.000/0.000/0.000 ms
 h3->h1: 1/0, rtt min/avg/max/mdev 0.000/0.000/0.000/0.000 ms
 h3->h2: 1/0, rtt min/avg/max/mdev 0.000/0.000/0.000/0.000 ms
 h3->h4: 1/0, rtt min/avg/max/mdev 0.000/0.000/0.000/0.000 ms
 h4->h1: 1/0, rtt min/avg/max/mdev 0.000/0.000/0.000/0.000 ms
 h4->h2: 1/0, rtt min/avg/max/mdev 0.000/0.000/0.000/0.000 ms
 h4->h3: 1/0, rtt min/avg/max/mdev 0.000/0.000/0.000/0.000 ms
```

Nas Figura 15.c.1 e Figura 15.c.2 realizou-se a autenticação do host 03 desta forma o controlador rejeita apenas os fluxos destinados ao host 04. Na Figura 15.c.1 tem-se a tabela de fluxos atualizada, ou seja, com o fluxo entre os hosts autorizados e na Figura 15.c.2 temos a confirmação do fluxo apenas entre os hosts autenticados. Utiliza-se o mesmo procedimento descrito no capítulo 3.2, para realizar a autenticação do host 3.

Figura 15.c.1: Tabela de Fluxo

```
mininet> sh ovs-ofctl dump-flows s1
NXST_FLOW reply (xid=0x4):
 cookie=0x0, duration=14.788s, table=0, n_packets=2, n_bytes=180, idle_age=13,
in_port=4,dl_src=00:00:00:00:00:04,dl_dst=33:33:00:00:00:16 actions=drop
 cookie=0x0, duration=14.718s, table=0, n_packets=0, n_bytes=0, idle_age=14, i
_port=3,dl_src=00:00:00:00:00:03,dl_dst=33:33:ff:00:00:03 actions=drop
 cookie=0x0, duration=14.648s, table=0, n_packets=0, n_bytes=0, idle_age=14, i
_port=4,dl_src=00:00:00:00:00:04,dl_dst=33:33:ff:00:00:04 actions=drop
 cookie=0x0, duration=14.426s, table=0, n_packets=2, n_bytes=180, idle_age=13,
in_port=1,dl_src=00:00:00:00:00:01,dl_dst=33:33:00:00:00:16 actions=drop
 cookie=0x0, duration=14.391s, table=0, n_packets=0, n_bytes=0, idle_age=14, i
_port=2,dl_src=00:00:00:00:00:02,dl_dst=33:33:ff:00:00:02 actions=drop
 cookie=0x0, duration=14.062s, table=0, n_packets=2, n_bytes=140, idle_age=6,
n_port=1,dl_src=00:00:00:00:00:01,dl_dst=33:33:00:00:00:02 actions=drop
 cookie=0x0, duration=13.683s, table=0, n_packets=1, n_bytes=90, idle_age=12,
n_port=3,dl_src=00:00:00:00:00:03,dl_dst=33:33:00:00:00:16 actions=drop
 cookie=0x0, duration=13.678s, table=0, n_packets=2, n_bytes=140, idle_age=5,
n_port=3,dl_src=00:00:00:00:00:03,dl_dst=33:33:00:00:00:02 actions=drop
 cookie=0x0, duration=13.660s, table=0, n_packets=2, n_bytes=140, idle_age=5,
n_port=4,dl_src=00:00:00:00:00:04,dl_dst=33:33:00:00:00:02 actions=drop
 cookie=0x0, duration=13.387s, table=0, n_packets=1, n_bytes=90, idle_age=12,
n_port=2,dl_src=00:00:00:00:00:02,dl_dst=33:33:00:00:00:16 actions=drop
 cookie=0x0, duration=13.382s, table=0, n_packets=2, n_bytes=140, idle_age=5,
n_port=2,dl_src=00:00:00:00:00:02,dl_dst=33:33:00:00:00:02 actions=drop
 cookie=0x0, duration=7.808s, table=0, n_packets=0, n_bytes=0, idle_age=7, in_
ort=1,dl_src=00:00:00:00:00:01,dl_dst=00:00:00:00:00:02 actions=output:2
 cookie=0x0, duration=7.787s, table=0, n_packets=0, n_bytes=0, idle_age=7, in_
ort=2,dl_src=00:00:00:00:00:02,dl_dst=00:00:00:00:00:01 actions=output:1
 cookie=0x0, duration=7.767s, table=0, n_packets=0, n_bytes=0, idle_age=7, in_
ort=1,dl_src=00:00:00:00:00:01,dl_dst=00:00:00:00:00:03 actions=output:3
 cookie=0x0, duration=7.747s, table=0, n_packets=0, n_bytes=0, idle_age=7, in_
ort=3,dl_src=00:00:00:00:00:03,dl_dst=00:00:00:00:00:01 actions=output:1
 cookie=0x0, duration=7.733s, table=0, n_packets=0, n_bytes=0, idle_age=7, in_
ort=3,dl_src=00:00:00:00:00:03,dl_dst=00:00:00:00:00:02 actions=output:2
 cookie=0x0, duration=5.978s, table=0, n_packets=0, n_bytes=0, idle_age=5, in_
ort=2,dl_src=00:00:00:00:00:02,dl_dst=00:00:00:00:00:03 actions=output:3
```

Figura 15.c.2: Aplicação da regra de fluxo entre o host 01, host 02 e host 03

```
mininet> pingallfull
*** Ping: testing ping reachability
h1 -> h2 h3 X
h2 -> h1 h3 X
h3 -> h1 h2 X
h4 -> X X X
*** Results:
 h1->h2: 1/1, rtt min/avg/max/mdev 0.508/0.508/0.508/0.000 ms
 h1->h3: 1/1, rtt min/avg/max/mdev 0.370/0.370/0.370/0.000 ms
 h1->h4: 1/0, rtt min/avg/max/mdev 0.000/0.000/0.000/0.000 ms
 h2->h1: 1/1, rtt min/avg/max/mdev 0.067/0.067/0.067/0.000 ms
 h2->h3: 1/1, rtt min/avg/max/mdev 0.600/0.600/0.600/0.000 ms
 h2->h4: 1/0, rtt min/avg/max/mdev 0.000/0.000/0.000/0.000 ms
 h3->h1: 1/1, rtt min/avg/max/mdev 0.309/0.309/0.309/0.000 ms
 h3->h2: 1/1, rtt min/avg/max/mdev 0.092/0.092/0.092/0.000 ms
 h3->h4: 1/0, rtt min/avg/max/mdev 0.000/0.000/0.000/0.000 ms
 h4->h1: 1/0, rtt min/avg/max/mdev 0.000/0.000/0.000/0.000 ms
 h4->h2: 1/0, rtt min/avg/max/mdev 0.000/0.000/0.000/0.000 ms
 h4->h3: 1/0, rtt min/avg/max/mdev 0.000/0.000/0.000/0.000 ms
```

Nas Figura 15.d.1 e Figura 15.d.2 realizou-se a autenticação do host 04 desta forma todos os hosts estão conectados. Na Figura 15.d.1 tabela de fluxos atualizada, ou seja, com o fluxo entre os hosts autorizados e na Figura 15.d.2 temos a confirmação do fluxo apenas entre os hosts autenticados. Utiliza-se o mesmo procedimento descrito no capítulo 3.2, para realizar a autenticação do host 4.

Na Figura 16 observa-se que ao desconectar a rede, ou seja, ao finalizar a rede através do MININET também o controlador é finalizado. A Figura 16 apresenta a utilização do comando *exit* que faz o MININET parar todas as execuções de rede. O tempo demonstrado na finalização do MININET é referente ao tempo de duração dos experimentos na plataforma virtual.

Figura 15.d.1: Tabela de Fluxo

```
mininet> sh ovs-ofctl dump-flows s1
NXST_FLOW reply (xid=0x4):
 cookie=0x0, duration=975.246s, table=0, n_packets=2, n_bytes=180, idle_age=97
3, in_port=4,dl_src=00:00:00:00:00:04,dl_dst=33:33:00:00:00:16 actions=drop
 cookie=0x0, duration=975.176s, table=0, n_packets=0, n_bytes=0, idle_age=975,
 in_port=3,dl_src=00:00:00:00:00:03,dl_dst=33:33:ff:00:00:03 actions=drop
 cookie=0x0, duration=975.106s, table=0, n_packets=0, n_bytes=0, idle_age=975,
 in_port=4,dl_src=00:00:00:00:00:04,dl_dst=33:33:ff:00:00:04 actions=drop
 cookie=0x0, duration=974.884s, table=0, n_packets=2, n_bytes=180, idle_age=97
3, in_port=1,dl_src=00:00:00:00:00:01,dl_dst=33:33:00:00:00:16 actions=drop
 cookie=0x0, duration=974.849s, table=0, n_packets=0, n_bytes=0, idle_age=974,
 in_port=2,dl_src=00:00:00:00:00:02,dl_dst=33:33:ff:00:00:02 actions=drop
 cookie=0x0, duration=974.520s, table=0, n_packets=2, n_bytes=140, idle_age=96
6, in_port=1,dl_src=00:00:00:00:00:01,dl_dst=33:33:00:00:00:02 actions=drop
 cookie=0x0, duration=974.141s, table=0, n_packets=1, n_bytes=90, idle_age=973
, in_port=3,dl_src=00:00:00:00:00:03,dl_dst=33:33:00:00:00:16 actions=drop
 cookie=0x0, duration=974.136s, table=0, n_packets=2, n_bytes=140, idle_age=96
6, in_port=3,dl_src=00:00:00:00:00:03,dl_dst=33:33:00:00:00:02 actions=drop
 cookie=0x0, duration=974.118s, table=0, n_packets=2, n_bytes=140, idle_age=96
6, in_port=4,dl_src=00:00:00:00:00:04,dl_dst=33:33:00:00:00:02 actions=drop
 cookie=0x0, duration=973.845s, table=0, n_packets=1, n_bytes=90, idle_age=973
, in_port=2,dl_src=00:00:00:00:00:02,dl_dst=33:33:00:00:00:16 actions=drop
 cookie=0x0, duration=973.840s, table=0, n_packets=2, n_bytes=140, idle_age=96
5, in_port=2,dl_src=00:00:00:00:00:02,dl_dst=33:33:00:00:00:02 actions=drop
 cookie=0x0, duration=968.266s, table=0, n_packets=0, n_bytes=0, idle_age=968,
 in_port=1,dl_src=00:00:00:00:00:01,dl_dst=00:00:00:00:00:02 actions=output:2
 cookie=0x0, duration=968.245s, table=0, n_packets=0, n_bytes=0, idle_age=968,
 in_port=2,dl_src=00:00:00:00:00:02,dl_dst=00:00:00:00:00:01 actions=output:1
 cookie=0x0, duration=968.225s, table=0, n_packets=0, n_bytes=0, idle_age=968,
 in_port=1,dl_src=00:00:00:00:00:01,dl_dst=00:00:00:00:00:03 actions=output:3
 cookie=0x0, duration=968.205s, table=0, n_packets=0, n_bytes=0, idle_age=968,
 in_port=3,dl_src=00:00:00:00:00:03,dl_dst=00:00:00:00:00:01 actions=output:1
 cookie=0x0, duration=968.191s, table=0, n_packets=0, n_bytes=0, idle_age=968,
 in_port=3,dl_src=00:00:00:00:00:03,dl_dst=00:00:00:00:00:02 actions=output:2
 cookie=0x0, duration=966.436s, table=0, n_packets=0, n_bytes=0, idle_age=966,
 in_port=2,dl_src=00:00:00:00:00:02,dl_dst=00:00:00:00:00:03 actions=output:3
 cookie=0x0, duration=10.183s, table=0, n_packets=0, n_bytes=0, idle_age=10, i
n_port=1,dl_src=00:00:00:00:00:01,dl_dst=00:00:00:00:00:04 actions=output:4
 cookie=0x0, duration=10.167s, table=0, n_packets=0, n_bytes=0, idle_age=10, i
n_port=4,dl_src=00:00:00:00:00:04,dl_dst=00:00:00:00:00:01 actions=output:1
 cookie=0x0, duration=10.157s, table=0, n_packets=0, n_bytes=0, idle_age=10, i
n_port=4,dl_src=00:00:00:00:00:04,dl_dst=00:00:00:00:00:02 actions=output:2
 cookie=0x0, duration=10.135s, table=0, n_packets=0, n_bytes=0, idle_age=10, i
n_port=2,dl_src=00:00:00:00:00:02,dl_dst=00:00:00:00:00:04 actions=output:4
 cookie=0x0, duration=10.117s, table=0, n_packets=0, n_bytes=0, idle_age=10, i
n_port=3,dl_src=00:00:00:00:00:03,dl_dst=00:00:00:00:00:04 actions=output:4
 cookie=0x0, duration=7.089s, table=0, n_packets=0, n_bytes=0, idle_age=7, in_
port=4,dl_src=00:00:00:00:00:04,dl_dst=00:00:00:00:00:03 actions=output:3
```

Figura 15.d.2: Aplicação da regra de fluxo entre todos os hosts

```
mininet> pingallfull
*** Ping: testing ping reachability
h1 -> h2 h3 h4
h2 -> h1 h3 h4
h3 -> h1 h2 h4
h4 -> h1 h2 h3
*** Results:
 h1->h2: 1/1, rtt min/avg/max/mdev 0.325/0.325/0.325/0.000 ms
 h1->h3: 1/1, rtt min/avg/max/mdev 0.312/0.312/0.312/0.000 ms
 h1->h4: 1/1, rtt min/avg/max/mdev 0.249/0.249/0.249/0.000 ms
 h2->h1: 1/1, rtt min/avg/max/mdev 0.059/0.059/0.059/0.000 ms
 h2->h3: 1/1, rtt min/avg/max/mdev 0.310/0.310/0.310/0.000 ms
 h2->h4: 1/1, rtt min/avg/max/mdev 0.316/0.316/0.316/0.000 ms
 h3->h1: 1/1, rtt min/avg/max/mdev 0.065/0.065/0.065/0.000 ms
 h3->h2: 1/1, rtt min/avg/max/mdev 0.117/0.117/0.117/0.000 ms
 h3->h4: 1/1, rtt min/avg/max/mdev 1.096/1.096/1.096/0.000 ms
 h4->h1: 1/1, rtt min/avg/max/mdev 0.095/0.095/0.095/0.000 ms
 h4->h2: 1/1, rtt min/avg/max/mdev 0.061/0.061/0.061/0.000 ms
 h4->h3: 1/1, rtt min/avg/max/mdev 0.195/0.195/0.195/0.000 ms
```

Figura 16: Finalizando o MININET

```
mininet> exit
*** Stopping 1 controllers
c0
*** Stopping 4 links
....
*** Stopping 1 switches
s1
*** Stopping 4 hosts
h1 h2 h3 h4
*** Done
completed in 878.192 seconds
```

4.3. Gerenciamento de Endereços de Entrada e Saída Utilizando Duas Tabelas

Nesta etapa utilizou-se o seguinte código apresentado abaixo:

$ sudo ovs-vsctl set bridge s1 protocols=OpenFlow10

$ sudo ./pox.py log.level --DEBUG forwarding. l2_nx

Nas Figura 17.a e Figura 17.b é possível observar a utilização e aplicação da regra de gerenciamento de entrada e saída de dados. Na Figura 17.a temos a utilização da função Gerenciamento por Tabela responsável por inicializar o comando para a geração da topologia no terminal de controle. A Figura 17.b demonstra a inicialização da regra no controlador, lembrando que deve-se criar a topologia antes de aplicar a regra ao controlador. Percebe-se que foram criadas duas novas regras, uma de fluxos de entrada e outra de fluxos de saída, geradas pelo controlador. Realizou-se a autenticação dos hosts, utilizando os procedimentos descritos no capitulo 3.2, para realizar os experimentos.

Na Tabela 1 têm-se os resultados obtidos nos experimentos de utilização da função *pingall* para testar a conexão da rede, utilizou-se de 10 experimentos aplicando a regra para avaliar o tempo de resposta do envio e recebimento de pacotes e 10 experimentos sem a aplicação para realizar a comparação. Vê-se que para realizar essa tarefa de enviar 12 pacotes levou-se apenas 0.000013 segundos em média. Em contrapartida percebe-se que esta tarefa é normalmente realizada em 0.000019 segundos em média para enviar a mesma quantidade de pacotes em uma rede sem utilizar a métrica aplicada. Nesse caso em específico nota-se uma melhora de 68,4% no tempo de envio e recebimento dos pacotes analisados.

Figura 17.a: Utilização da função Gerenciamento por Tabela.

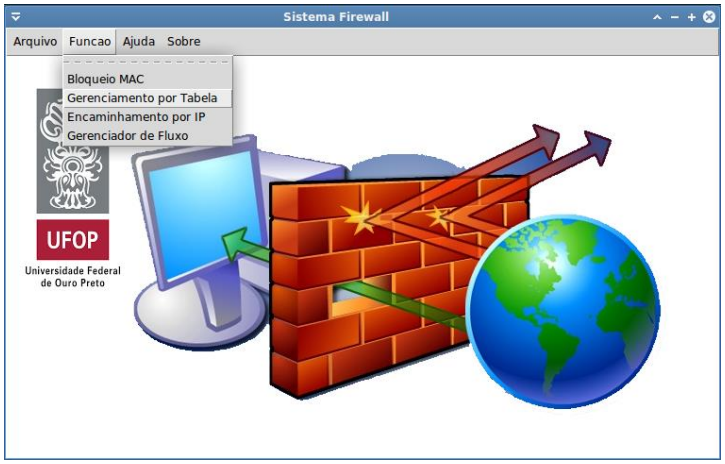


Figura 17.b: Aplicação da regra de gerenciamento por tabela

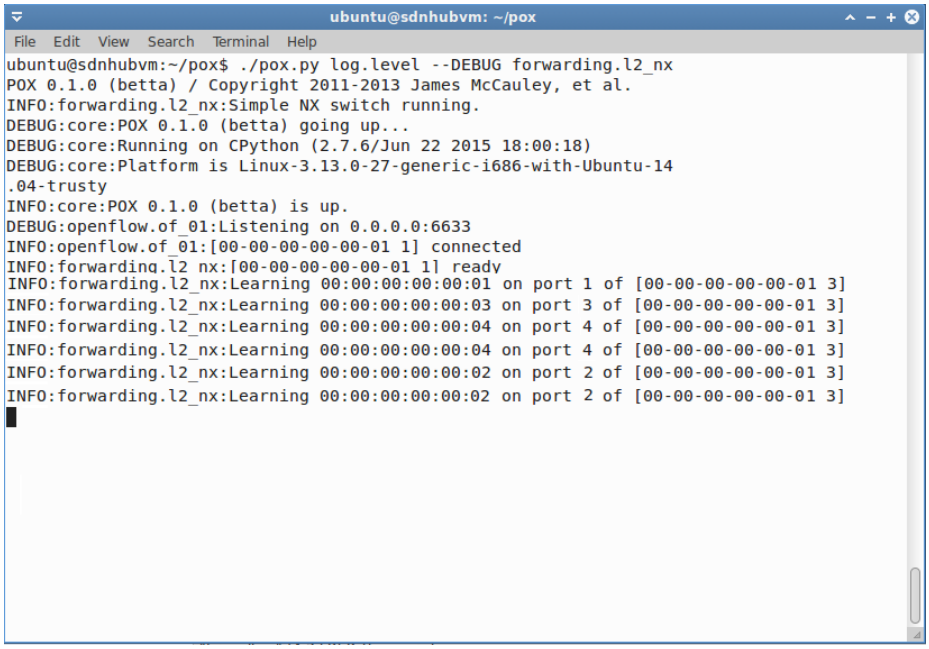


Figura 17.c: Tempo de execução para o envio de pacotes

Índice do experimento	Regra Aplicada	Regra não aplicada
1	0.000013	0.000019
2	0.000014	0.000020
3	0.000012	0.000021
4	0.000013	0.000018
5	0.000013	0.000018
6	0.000015	0.000017
7	0.000014	0.000018
8	0.000013	0.000020
9	0.000012	0.000019
10	0.000011	0.000020
Total do tempo do experimento	0.000130	0.000190
Média	0.000013	0.000019
Desvio Padrão	0,000001	0,000001

4.4. Encaminhar de pacotes com base em endereços IP

Nesta etapa utilizou-se o seguinte código para realizar esta operação.

$ ./pox.py log.level --DEBUG forwarding. topo_proactive

Com a aplicação do código acima, obteve-se os seguintes resultados nas Figura 18.a, Figura 18.b, Figura 18.c e Figura 18.d, devido à utilização da aplicação da regra para realizar o gerenciamento de fluxo. Na Figura 18.a temos a utilização da função Encaminhamento por IP responsável por inicializar o comando para a geração da topologia no terminal de controle. Realizou-se a autenticação dos hosts, utilizando os procedimentos descritos no capitulo 3.2, para realizar os experimentos.

Na Figura 18.b tem-se a inicialização da regra no controlador, lembrando-se que deve criar a topologia antes de aplicar a regra ao controlador. Em seguida, vemos que na Figura 18.c utilizou-se a função *iperfudp* e na Figura 18.d é exibido o tráfego udp que estava ativo.

Figura 18.a: Utilização da função Encaminhamento por IP.

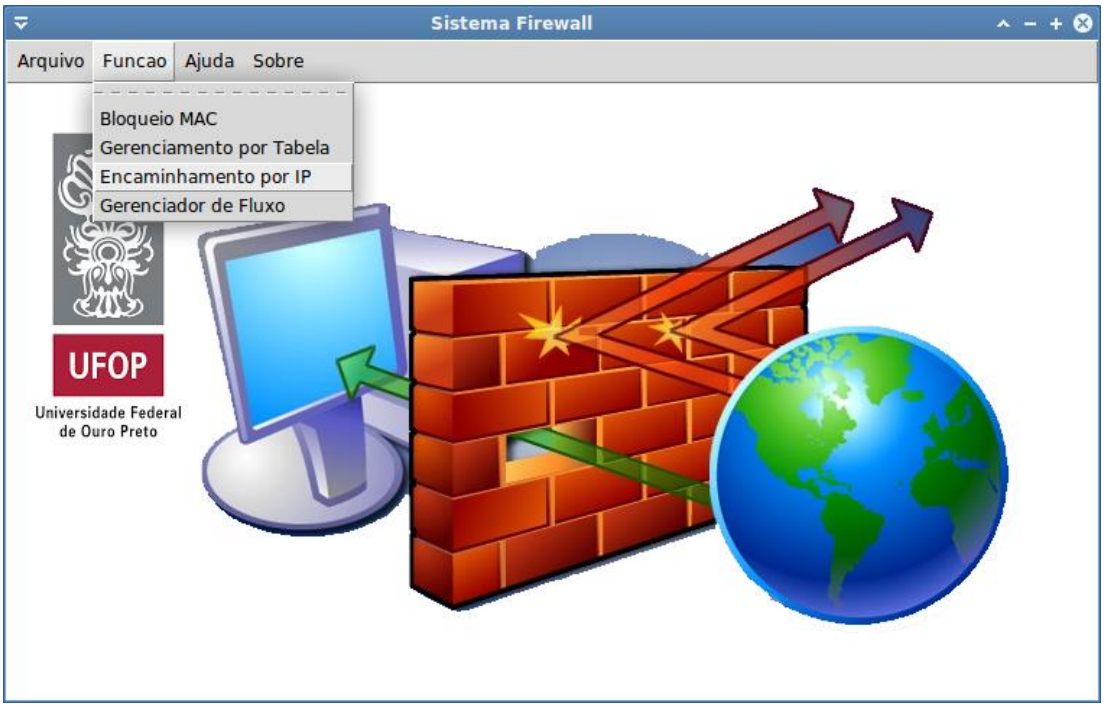


Figura 18.b: Aplicando a regra de gerenciamento de fluxo no controlador

```
ubuntu@sdnhubvm:~/pox$ ./pox.py log.level --DEBUG forwarding.topo_proactive
POX 0.5.0 (eel) / Copyright 2011-2014 James McCauley, et al.
DEBUG:core:POX 0.5.0 (eel) going up...
DEBUG:core:Running on CPython (2.7.6/Jun 22 2015 18:00:18)
DEBUG:core:Platform is Linux-3.13.0-27-generic-i686-with-Ubuntu-14.04-trusty
DEBUG:core:topo_addressing still waiting for: openflow_discovery
WARNING:core:Still waiting on 1 component(s)
INFO:core:POX 0.5.0 (eel) is up.
DEBUG:openflow.of_01:Listening on 0.0.0.0:6683
INFO:openflow.of_01:[00-00-00-00-00-01 1] connected
```

Figura 18.c: Aplicando a regra de gerenciamento de fluxo usando o MININET

```
*** Adding hosts:
h1 h2 h3 h4
*** Adding switches:
s1
*** Adding links:
(h1, s1) (h2, s1) (h3, s1) (h4, s1)
*** Configuring hosts
h1 h2 h3 h4
*** Starting controller
c0
*** Starting 1 switches
s1 ...
*** Starting CLI:
mininet> xterm h1 h2 h3 h4
mininet> iperfudp
*** Iperf: testing UDP bandwidth between h1 and h4
could not parse iperf output: ------------------------------------------------
-----------
Server listening on UDP port 5001
Receiving 1470 byte datagrams
UDP buffer size:  160 KByte (default)
------------------------------------------------------------
*** Results: ['10M', '', '9.98 Mbits/sec']
```

Figura 18.d: Execução do gerenciamento de fluxo no controlador

```
ubuntu@sdnhubvm:~/pox$ ./pox.py log.level --DEBUG forwarding.topo_proactive
POX 0.5.0 (eel) / Copyright 2011-2014 James McCauley, et al.
DEBUG:core:POX 0.5.0 (eel) going up...
DEBUG:core:Running on CPython (2.7.6/Jun 22 2015 18:00:18)
DEBUG:core:Platform is Linux-3.13.0-27-generic-i686-with-Ubuntu-14.04-trusty
DEBUG:core:topo_addressing still waiting for: openflow_discovery
WARNING:core:Still waiting on 1 component(s)
INFO:core:POX 0.5.0 (eel) is up.
DEBUG:openflow.of_01:Listening on 0.0.0.0:6683
INFO:openflow.of_01:[00-00-00-00-00-01 1] connected
INFO:packet:(udp parse) warning UDP packet data shorter than UDP len: 94 < 1478
INFO:packet:(udp parse) warning UDP packet data shorter than UDP len: 94 < 1478
INFO:packet:(udp parse) warning UDP packet data shorter than UDP len: 94 < 1478
INFO:packet:(udp parse) warning UDP packet data shorter than UDP len: 94 < 1478
INFO:packet:(udp parse) warning UDP packet data shorter than UDP len: 94 < 1478
INFO:packet:(udp parse) warning UDP packet data shorter than UDP len: 94 < 1478
INFO:packet:(udp parse) warning UDP packet data shorter than UDP len: 94 < 1478
INFO:packet:(udp parse) warning UDP packet data shorter than UDP len: 94 < 1478
INFO:packet:(udp parse) warning UDP packet data shorter than UDP len: 94 < 1478
INFO:packet:(udp parse) warning UDP packet data shorter than UDP len: 94 < 1478
INFO:packet:(udp parse) warning UDP packet data shorter than UDP len: 94 < 1478
```

Após a análise dos dados descritos acima, aplicou-se a função *exit* no terminal do MININET para desligar toda a topologia da rede. E como ilustra as Figura 19 todo fluxo inserido foi excluído do sistema. O tempo demonstrado na finalização do MININET é referente ao tempo de duração dos experimentos na plataforma virtual.

Figura 19: Exclusão das regras de gerenciamento de fluxo

```
INFO:packet:(udp parse) warning UDP packet data shorter than UDP len: 94 < 1478
INFO:packet:(udp parse) warning UDP packet data shorter than UDP len: 94 < 1478
INFO:packet:(udp parse) warning UDP packet data shorter than UDP len: 94 < 1478
INFO:packet:(udp parse) warning UDP packet data shorter than UDP len: 94 < 1478
INFO:packet:(udp parse) warning UDP packet data shorter than UDP len: 94 < 1478
INFO:packet:(udp parse) warning UDP packet data shorter than UDP len: 94 < 1478
INFO:packet:(udp parse) warning UDP packet data shorter than UDP len: 94 < 1478
INFO:packet:(udp parse) warning UDP packet data shorter than UDP len: 94 < 1478
INFO:packet:(udp parse) warning UDP packet data shorter than UDP len: 94 < 1478
INFO:packet:(udp parse) warning UDP packet data shorter than UDP len: 94 < 1478
INFO:packet:(udp parse) warning UDP packet data shorter than UDP len: 94 < 1478
INFO:openflow.of_01:[00-00-00-00-00-02 1] disconnected
INFO:core:Down.
```

Analisando os resultados acima apresentados, percebe-se que com a aplicação de regras simples têm-se um gerenciamento eficaz do fluxo de dados. Além disto, outras funções ao firewall podem ser aplicadas de acordo com a necessidade das situações, uma vez que a inserção de informações no controlador é um processo simples de ser realizado. As vantagens da utilização um controlador é a acessibilidade dos resultados para a sua utilização e manipulação em outras formas de aplicação, garantindo um gerenciamento que atenda a demanda de problemas cotidianos.

4.5. Gerenciamento de Fluxo

Nesta etapa aplicou-se o seguinte comando:

$ cd ./pox.py log.level --DEBUG forwarding.l3_learning

Como ilustrado nas Figura 20.a, Figura 20.b, Figura 20.c e Figura 20.d vê-se a aplicação da regra de encaminhamento de pacotes com base em endereços IP. Na Figura 20.a temos a utilização da função Gerenciamento de Fluxo responsável por inicializar o comando para a geração da topologia no terminal de controle. A Figura 20.b mostra a inicialização da regra no controlador, lembrando que deve-se criar a topologia antes de aplicar a regra ao controlador. Realizou-se a autenticação dos hosts, utilizando os procedimentos descritos no capitulo 3.2, para realizar os experimentos.

Já a Figura 20.c tem-se a solicitação no MININET para a função *ping* e na Figura 20.d visualiza-se o encaminhamento de dados no controlador. Além disto, pode-se visualizar na Figura 20.d o acompanhamento dos pacotes por IP, visualizar as inserções das regras na tabela de gerenciamento *OpenFlow* e as conexões das portas.

Utilizou-se a topologia apresentada na Figura 20.c apenas para demonstrar a inserção das regras de fluxo apresentadas no controlador, Figura

20.d, e sendo assim apresentar diferentes formas de controlar o fluxo de dados que trafegam na rede.

Figura 20.a: Utilização da função Gerenciador de Fluxo.

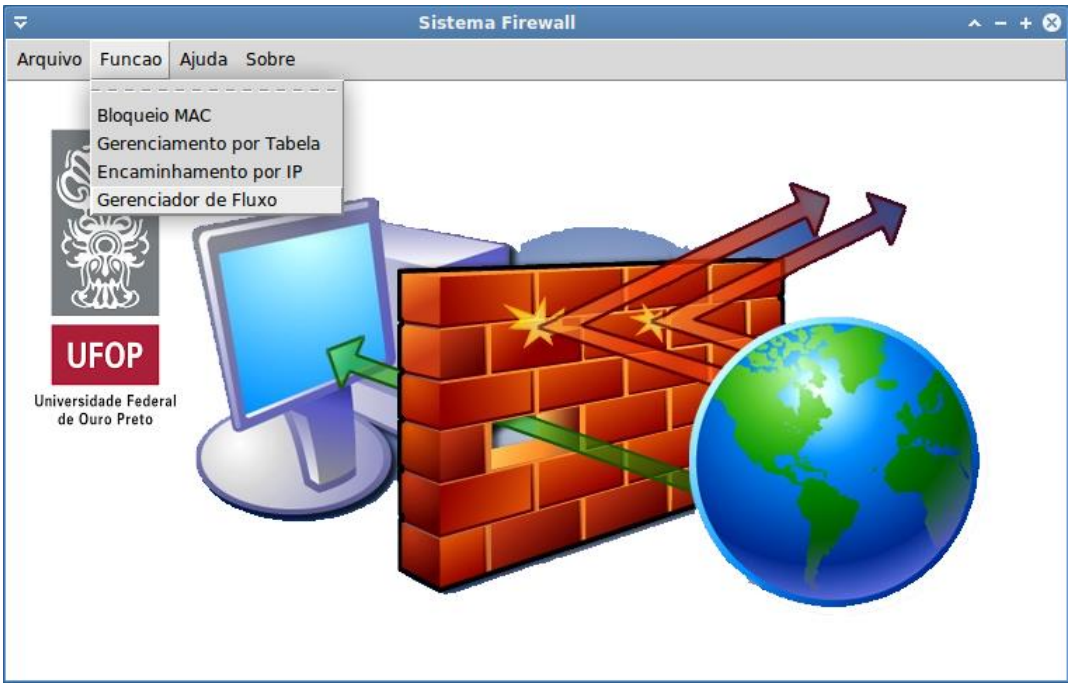


Figura 20.b: Aplicação da regra de encaminhamento de pacotes no controlador

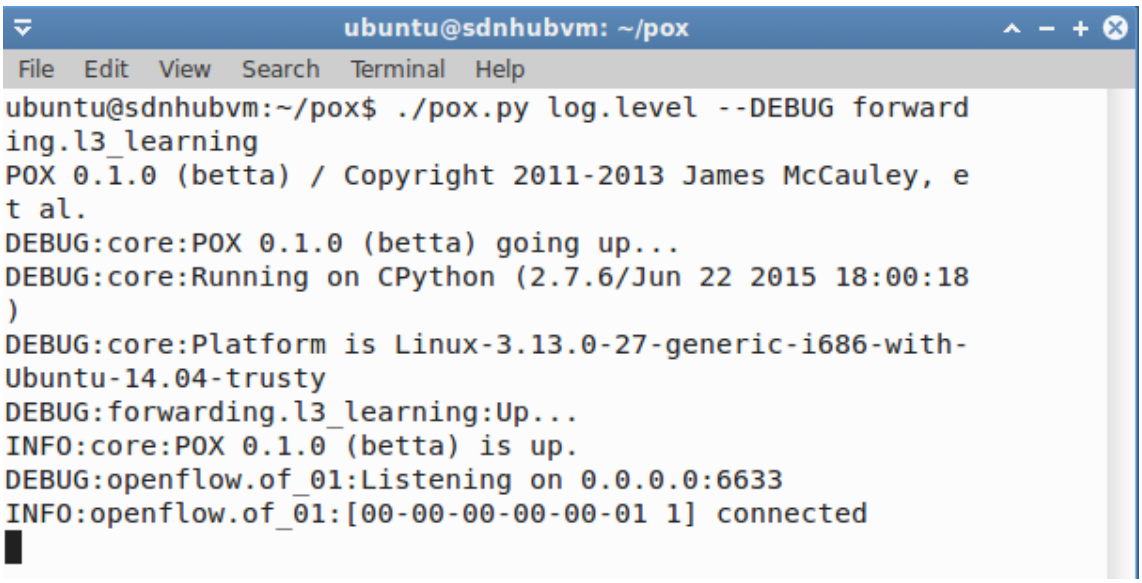


Figura 20.c: Aplicação da regra de encaminhamento de pacotes usando o MININET

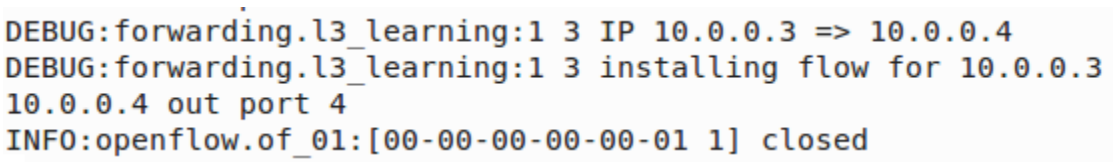
```
mininet> pingall
*** Ping: testing ping reachability
h1 -> X X X
h2 -> h1 X X
h3 -> h1 h2 X
h4 -> h1 h2 h3
*** Results: 50% dropped (6/12 received)
mininet>
```

Figura 20.d: Execução do encaminhamento de pacotes no controlador

```
DEBUG:forwarding.l3_learning:1 1 IP 10.0.0.1 => 10.0.0.2
DEBUG:forwarding.l3_learning:1 1 learned 10.0.0.1
DEBUG:forwarding.l3_learning:1 1 IP 10.0.0.1 => 10.0.0.3
DEBUG:forwarding.l3_learning:1 1 IP 10.0.0.1 => 10.0.0.4
DEBUG:forwarding.l3_learning:1 2 IP 10.0.0.2 => 10.0.0.1
DEBUG:forwarding.l3_learning:1 2 learned 10.0.0.2
DEBUG:forwarding.l3_learning:1 2 installing flow for 10.0.0.2
10.0.0.1 out port 1
DEBUG:forwarding.l3_learning:1 1 IP 10.0.0.1 => 10.0.0.2
DEBUG:forwarding.l3_learning:1 1 installing flow for 10.0.0.1
10.0.0.2 out port 2
DEBUG:forwarding.l3_learning:1 2 IP 10.0.0.2 => 10.0.0.3
DEBUG:forwarding.l3_learning:1 2 IP 10.0.0.2 => 10.0.0.4
DEBUG:forwarding.l3_learning:1 3 IP 10.0.0.3 => 10.0.0.1
DEBUG:forwarding.l3_learning:1 3 learned 10.0.0.3
DEBUG:forwarding.l3_learning:1 3 installing flow for 10.0.0.3
10.0.0.1 out port 1
DEBUG:forwarding.l3_learning:1 1 IP 10.0.0.1 => 10.0.0.3
DEBUG:forwarding.l3_learning:1 1 installing flow for 10.0.0.1
10.0.0.3 out port 3
DEBUG:forwarding.l3_learning:1 3 IP 10.0.0.3 => 10.0.0.2
DEBUG:forwarding.l3_learning:1 3 installing flow for 10.0.0.3
10.0.0.2 out port 2
DEBUG:forwarding.l3_learning:1 2 IP 10.0.0.2 => 10.0.0.3
DEBUG:forwarding.l3_learning:1 2 installing flow for 10.0.0.2
10.0.0.3 out port 3
DEBUG:forwarding.l3_learning:1 3 IP 10.0.0.3 => 10.0.0.4
DEBUG:forwarding.l3_learning:1 4 IP 10.0.0.4 => 10.0.0.1
DEBUG:forwarding.l3_learning:1 4 learned 10.0.0.4
DEBUG:forwarding.l3_learning:1 4 installing flow for 10.0.0.4
10.0.0.1 out port 1
DEBUG:forwarding.l3_learning:1 1 IP 10.0.0.1 => 10.0.0.4
DEBUG:forwarding.l3_learning:1 1 installing flow for 10.0.0.1
10.0.0.4 out port 4
DEBUG:forwarding.l3_learning:1 4 IP 10.0.0.4 => 10.0.0.2
DEBUG:forwarding.l3_learning:1 4 installing flow for 10.0.0.4
10.0.0.2 out port 2
DEBUG:forwarding.l3_learning:1 2 IP 10.0.0.2 => 10.0.0.4
DEBUG:forwarding.l3_learning:1 2 installing flow for 10.0.0.2
10.0.0.4 out port 4
DEBUG:forwarding.l3_learning:1 4 IP 10.0.0.4 => 10.0.0.3
DEBUG:forwarding.l3_learning:1 4 installing flow for 10.0.0.4
10.0.0.3 out port 3
DEBUG:forwarding.l3_learning:1 3 IP 10.0.0.3 => 10.0.0.4
DEBUG:forwarding.l3_learning:1 3 installing flow for 10.0.0.3
10.0.0.4 out port 4
```

Aplicando a função *exit* no terminal do MININET para desligar toda a topologia da rede, é possível observar na Figura 21 que todo o fluxo que foi inserido e excluído do sistema.

Figura 21: Exclusão das Regras Encaminhamento de Dados.



5. CONCLUSÃO

A proposta deste estudo foi desenvolver um sistema firewall para a rede Ethernet do campus do ICEA/UFOP utilizando o protocolo *OpenFlow*. O propósito inicial era desenvolver um protótipo do sistema firewall utilizando a plataforma MININET. Através de todos os dados acima descritos percebe-se outras possíveis aplicações para outras regras no sistema firewall, fazendo com que este esteja sempre adaptado as novas situações que vão surgir com o passar do tempo, sempre possibilitando a melhoria, expansão e eficiência da rede.

Fica nítido que o sistema firewall, utilizando o protocolo *OpenFlow*, é capaz de fornecer uma proteção muito mais ampla do que a que o sistema possui atualmente. Os benefícios obtidos com a implementação do protótipo na rede física podem gerar contribuições em outras áreas, além do gerenciamento do fluxo de dados, como por exemplo, o gerenciamento dos recursos da rede, através de um processamento utilizando apenas o controlador e não os equipamentos de encaminhamento de dados, isto permitirá um fluxo de informações mais eficaz em horários de instabilidade de rede. Além disto, a rede pode ser gerenciada remotamente tendo acesso apenas às configurações do firewall, entre outras funcionalidades que podem ser realizadas utilizando um firewall com o protocolo OpenFlow.

A realização deste estudo apresenta uma lacuna nos referenciais metodológicos, no que diz a aplicação e desenvolvimento das redes definidas por software, pois o estudo em questão não é uma tecnologia consolidada na área de redes de computadores. Desta forma, vê-se a necessidade de melhores referenciais teóricos para que novas pesquisas sejam mais embasadas. Outro ponto de dificuldade foi encontrar manuais referenciais da plataforma utilizada (MININET), pois existem poucas bases de dados com informações sobre as funções que podem ser realizadas pela plataforma. Nota-se que o MNINET contempla uma gama muito grande de utilidades, porém as

aplicações são voltadas para funções muito específicas e não possui manuais gerais para a utilização do sistema.

Como o desenvolvimento de redes definidas por software ainda é uma nova sub área de redes de computadores existem ainda muitas questões que precisam ser exploradas. Os resultados apresentados neste estudo podem ser aplicados na rede física do ICEA e até mesmo expandir tal estudo para a rede geral de toda a UFOP, realizando a implantação da autenticação do servidor RADIUS ligado a um controlador SDN para a rede Ethernet.

REFERÊNCIAS

BHATIA J.. Mininet: Na emulator for phototyping large network topologies on a singles machine. INDIA: **Open Source Network Simulations**, 2015. Disponível em: <http://opensourceforu.com/2015/10/mininet-an-emulator-for-prototyping-large-network-topologies-on-a-single-machine/>. Acessado em: 30 nov. 2017.

CASADO, M.; KOPONEN, T., RAMANATHAN, R.; SHENKER, S. Virtualizing the network forwarding plane. In Proceedings of the Workshop on Programmable Routers for Extensible Services of Tomorrow. **ACM Presto**, v. 8, n. 6, pag.1-8, 2010.

COMER, D. E. **Redes de Computadores e Internet**. Porto Alegre. Bookman Companhia Editora Ltda, 2016.

CORDEIRO, C. T. **Desenvolvimento de um Sistema Web de Apoio ao Gerenciamento de Dispositivos da Rede do ICEA**. 2015. Monografia – Departamento de Sistemas de Informação, Universidade Federal de Ouro Preto, Ouro Preto, 2015.

DAVIE, B. S.; FARREL, A. **MPLS: Next Steps**. San Francisco. Morgan Kaufmann, 2008.

DIAS, D. **Guia Básico para Configuração de Switches Cisco**, 2014. RD PRESS. Disponível em: <http://www.comutadores.com.br/wp-content/uploads/2014/11/Guia C3%A1sico-para-configura%C3%A7%C3%A3o-de-Switches-Catalyst-Ciscoamostra .pdf.> Acesso em: 30 nov. 2017.

DROMS, R. **Dynamic Host Configuration Protocol**. Bucknell University, 1997.

GIT HUB. Mininet/openflow-tutorial. USA: **Git Hub**, 2013. Disponível em: <https://github.com/mininet/openflow-tutorial>. Acessado em: 15 nov. 2017.

GIT HUB. Noxrepo/openflow. USA: **Git Hub**, 2011. Disponível em: <https://github.com/noxrepo/openflow>. Acessado em: 20 nov. 2017.

GIT HUB. Noxrepo/pox. USA: **Git Hub**, 2013 Disponível em: < https://github.com/noxrepo/pox>. Acessado em: 20 nov. 2017.

GIT HUB. Pratiklotia/ SDN- Firewall. USA: **Git Hub**, 2017. Disponível em: < https://github.com/pratiklotia/SDN-Firewall>. Acessado em: 20 nov. 2017.

GUEDES, D *et. al*.. **Redes definidas por Software: uma abordagem sistêmica para o desenvolvimento das pesquisas em Redes de Computadores**. Ouro Preto, 2012.

HASSEL, Jonathan**. RADIUS**. O'Reilly Media, 2002.

KEMPF, J *et. al*.. Openflow mpls and the open source label switched router. In **Proceedings of the 23rd International Teletraffic Congress**, ITCP, 2011.

KICKSTART SDN. Add custom scripts. USA: **Kickstart**, 2015. Disponível em:<http://kickstartsdn.com/author/abdan/page/2/>. Acesso em: 15 Mar. 2017.

KUROSE, J. F.; ROSS, K. W. **Redes de Computadores e a Internet: uma abordagem top-down**. São Paulo: Addison Wesley, 2010.

LINKLETTER, B.. Open-Source Routing and Network Simulation. CANADA: **Open-Source Network Simulations**, 2017. Disponível em: <http://www.brianlinkletter.com/open-source-network-simulators/>. Acessado em: 30 nov. 2017.

MCKEOWN, N et al. Openflow: enabling innovation in campus networks. **ACM SIGCOMM Comput. Commun**. Rev., v. 38, p. 69–74, 2008.

MEHDI, S. A.; KHALID, J.; KHAYAM, S. A. Evisiting traffic anomaly detection using software defined networking. **In Proceedings of the 14th International Symposium on Recent Advances in Intrusion Detection (RAID),** p. 161–180, 2011.

NADEAU, T. D.; GRAY, W. K. **SDN: Software Defined Networks**. O'Reilly Media, 2013.

ORACLE. MYSQL Reference Manual.USA/CANADA: **Oracle Corporation and/or its affiliates**, 2017. Disponível em: < https://dev.mysql.com/doc/>. Acesso em: 2 Ago. 2017.

PATEL, P.. Implementing software-defined network (SDN) based firewall. INDIA: **Open Source Network Simulations**, 2016. Disponível em: < http://opensourceforu.com/2016/07/implementing-a-software-defined-network-sdn-based-firewall/>. Acessado em: 30 nov. 2017.

PEREIRA, Marcelo Veiga. **Implementando Segurança no Nível de Acesso Utilizando Servidor Radius**. 2011. Monografia – Departamento Acadêmico de Eletrônica. Universidade Tecnológica Federal do Paraná, Curitiba, 2011.

PETERSON, L; ROSCOE, T. The design principles of planetlab. **SIGOPS Oper. Syst. Rev**., v. 40, n. 1, p.11–16, 2006.

PLUMMER, D. C. RFC 826, **An Ethernet Address Resolution Protocol – or – Converting Network Protocol Addresses to 48 bit Ethernet Address for Transmission on Ethernet Hardware.** Internet Engineering Task Force, Network Working Group, 1982.

PYTHON SOFTWARE FOUNDATION. Built-in Functions. USA: **Python Software Foundation,** 2017. Disponível em: <https://docs.python.org/2/library/functions.html>. Acessado em: 2 dez. 2017.

PYTHON SOFTWARE FOUNDATION. The Python tutorial. USA: **Python Software Foundation,** 2017. Disponível em: <https://docs.python.org/3/tutorial/>. Acessado em: 15 nov. 2017.

RIGNEY, C. RADIUS Accouting. USA: **Tools Ietf**, 2000. Disponível em: <https://tools.ietf.org/html/rfc2866>. Acesso em: 2 Ago. 2017

ROTHENBERG, C. E.; NASCIMENTO, M. R.; SALVADOR, M. R; MAGALHÃES, M. F. OpenFlow e redes definidas por software: um novo paradigma de controle e inovação em redes de pacotes. **Cad. CPqD Tecnologia**. Campinas, v. 7, n. 1, p. 65 – 76, 2011.

SDN HUB. All-in-one SDN App Development Starter VM. USA: **SDN Hub**, 2014. Disponível em: <http://sdnhub.org/tutorials/sdn-tutorial-vm/> Acesso em: 15 Mar. 2017.

SDN HUB. POX Controller Tutorial. USA: **SDN Hub**, 2014. Disponível em: <http://sdnhub.org/tutorials/pox/>. Acesso em: 15 Mar. 2017.

SDN HUB. Useful mininet setups. USA: **SDN Hub**, 2014. Disponível em: <http://sdnhub.org/resources/useful-mininet-setups/> Acesso em: 15 Mar. 2017.

SILVA, L. C.. POXDoc. USA: **Git Hub,** 2017. Disponível em: <https://github.com/Lafaiet/POXDoc/blob/master/PT_BR/Main.md> Acesso em: 15 Nov. 2017.

SOFKA, V.. A pesquisa no museu e sobre o museu. In: JELÍNEK, Jan; SLANÁ, Věra (Org.) Possibilities and Limits of Scientific Research typical for the museums /Possibilités et limites de la recherche scientifique pour les musées. ICOM-International Committee for Museology / Comité International pour la Muséologie. **Revista Eletrônica do Programa de Pós-Graduação em Museologia e Patrimônio**, v. 2, n. 1, 2009.

STANFORD. The OPENFLOW Specification. USA: Califórnia, **Stanford University**, 2010. Disponível em: <http://www.openflowswitch.org/documents/openflow-wp-latest.pdf>. Acesso em: 5 Jan. 2017.

STANFORD. The OPENFLOW POX. USA: Califórnia, **Stanford University**, 2010.. Disponível em: <https://openflow.stanford.edu/display/ONL/POX+Wiki/>. Acesso em: 15 Mar. 2017.

TANENBAUM, Andrew S. **Redes de Computadores**. Rio de Janeiro: Elsevier, 2003.

TENNENHOUSE, D. L.; WETHERALL, D. J. Towards an active network architecture. **SIGCOMM Comput. Commun. Rev.,** v. 37, n. 5, p. 81–94, 2007

TORRES, Gabriel, **Redes de Computadores Versão Revisada e Atualizada**. Novaterra Editora e Distribuidora Ltda. Rio de Janeiro: Novaterra Editora e Distribuidora Ltda, 2013.

VICENTINO, Cláudio. **História Geral**. São Paulo: Scipione, 1991.

ZWICKY, E. D.; COOPER, S., CHAPMAN, D. B. **Building Internet Firewalls**. O'Reilly Media, 2009.

TERMO DE RESPONSABILIDADE

Eu, Raquel Marina da Conceição declaro que o texto do trabalho de conclusão de curso intitulado "***Sistema firewall para o ambiente acadêmico do instituto de ciências exatas e aplicadas – ICEA utilizando o protocolo openflow***" é de minha inteira responsabilidade e que não há utilização de texto, material fotográfico, código fonte de programa ou qualquer outro material pertencente a terceiros sem as devidas referências ou consentimento dos respectivos autores.

João Monlevade, 13 de Julho de 2018.

Raquel Marina da Conceição

Certifico que o aluna **Raquel Marina da Conceição,** autora do trabalho de conclusão de curso intitulado "**SISTEMA FIREWALL PARA O AMBIENTE ACADÊMICO DO INSTITUTO DE CIÊNCIAS EXATAS E APLICADAS – ICEA UTILIZANDO O PROTOCOLO OPENFLOW**", efetuou as correções sugeridas pela banca examinadora e que estou de acordo com a versão final do trabalho.

Erik de Britto e Silva
Orientador

João Monlevade, 21 de julho de 2018.