

Leandro Aurélio da Silva

REDES SEM FIO DEFINIDAS POR SOFTWARE: ESTUDO DO MININET WI-FI

Santo André

2017

Leandro Aurélio da Silva

REDES SEM FIO DEFINIDAS POR SOFTWARE: ESTUDO DO MININET WI-FI

Trabalho desenvolvido durante as disciplinas de Trabalho de Graduação I, II e III, apresentado no curso de graduação em Engenharia de Informação como requisito para obtenção do Título de Engenheiro de Informação pela Universidade Federal do ABC.

Universidade Federal do ABC – UFABC

Centro de Engenharia, Modelagem e Ciências Sociais Aplicadas – CECS

Trabalho de Conclusão de Curso

Orientador: Prof. Dr. João Henrique Kleinschmidt

Santo André

2017

Leandro Aurélio da Silva

REDES SEM FIO DEFINIDAS POR SOFTWARE: ESTUDO DO MININET WI-FI

Trabalho desenvolvido durante as disciplinas de Trabalho de Graduação I, II e III, apresentado no curso de graduação em Engenharia de Informação como requisito para obtenção do Título de Engenheiro de Informação pela Universidade Federal do ABC.

Trabalho aprovado. Santo André, 2017

Prof. Dr. João Henrique Kleinschmidt
Orientador

Prof. Dr. Murilo Bellezoni Loiola
Avaliador

Prof. Dr. Luiz Henrique Bonani
Avaliador

Santo André
2017

Dedico este trabalho à minha família, que muitas vezes foi necessário eu me abster de presença física para poder me dedicar a algo no qual sempre tive apoio e suporte destes.

Agradecimentos

Agradeço a meus pais, toda minha família, minha namorada e meus amigos, pelo amor e apoio incondicional.

Ao meu orientador João Henrique Kleinshmidt pela sugestão do tema e todas as possibilidades que o assunto pode proporcionar, além do incondicional apoio, supervisão e incentivo para que fosse realizado um bom trabalho.

Ao Ramon Fontes, criador do Mininet Wifi, que sempre que foi preciso me deu sugestões valiosas para que fosse possível a continuidade do trabalho.

A Universidade Federal do ABC, por me proporcionar um ambiente de constante obtenção de conhecimento, além de uma grande infraestrutura capaz de suprir minhas necessidades técnicas e operacionais.

A todos os professores que independente do momento estavam sempre dispostos a ceder um tempo para compartilhar de seus conhecimentos.

*“Só se pode alcançar um grande êxito
quando nos mantemos fiéis a nós mesmos. ”
(Friedrich Nietzsche)*

Resumo

Com a dinâmica e o substancial aumento no tráfego de dados bem como sua mobilidade, pode-se notar que a tecnologia da informação está em franco desenvolvimento, porém nota-se que a infraestrutura para o tráfego destes dados, tanto em tecnologias cabeadas, bem como tecnologias sem fio, não conseguem se desenvolver com a mesma velocidade. Isso se dá principalmente pelo modo como a Internet foi estruturada, fazendo com que não fosse possível realizar grandes mudanças em seu núcleo. Com o surgimento do protocolo OpenFlow e o conceito de Redes SDN (*Software Defined Network*), foi possível implementar soluções em paralelo com tráfego atual sem afetar diretamente o que existe na infraestrutura, e com a finalidade de atender diferentes demandas emergentes. Com o uso deste novo modelo, foi desenvolvido também o conceito de SDN para redes sem fio (SDN-WIFI), cujo objetivo principal é utilizar as mesmas aplicações de redes SDN para seu uso em enlaces sem fio e mobilidade. Neste trabalho será apresentado o conceito de Redes definidas por software, bem como os conceitos de SDN-WIFI e alternativas para o tráfego de dados sem fio através do uso deste modelo. Será avaliado o Mininet-WiFi para na realização das simulações de ambientes.

Palavras-chaves: OpenFlow, SDN, SDN-WIFI, POX, Redes definidas por Software.

Lista de ilustrações

Figura 1 – Modelo com plano de controle e plano de dados (VELRAJAN, 2012) .	24
Figura 2 – Projeto inicial de proposta de rede OpenFlow (MCKEOWN et al., 2008)	26
Figura 3 – Estrutura geral do funcionamento de uma rede SDN Fonte: Anderson Coelho Weller - Unicamp	27
Figura 4 – Funcionamento do controlador Floodlight. Fonte www.floodlight.com .	29
Figura 5 – Estrutura do controlador OpenDaylight. Fonte www.opendaylight.org .	30
Figura 6 – Modo interativo de gerenciamento de rede com Mininet	30
Figura 7 – Tela inicial do Wireshark	36
Figura 8 – Tráfego monitorado pela interface hwsim0	37
Figura 9 – Topologia Linear criada para Mininet-WiFi	38
Figura 10 – Saídas geradas pelo comando iw	39
Figura 11 – Troca de conexões entre ap1 e ap2 para st1	40
Figura 12 – Saída comando ping entre Sta1 e Sta3	41
Figura 13 – Xterm para sta3	42
Figura 14 – Fluxo do protocolo OpenFlow através com o Wireshark	42
Figura 15 – Gráfico com a topologia do script position-test.py	46
Figura 16 – Fluxo de saída do comando iperf realizado por h1 em sta3	48
Figura 17 – Status de stations conectadas e desconectadas	49
Figura 18 – Ambiente criado para modelo de propagação	50
Figura 19 – Handover ocorrido com o modelo Friis de propagação	51
Figura 20 – Iniciando o mininet com o uso do controlador NOX	61
Figura 21 – Iniciando o controlador NOX	63

Lista de abreviaturas e siglas

API	Application Programming Interface (Interface de Programação de Aplicativos)
ARP	Address Resolution Protocol (Protocolo de Resolução de Endereços)
CLI	Command Line Interface (Interface de Linha de Comando)
FIB	Fowarding Information Base (Base de Encaminhamento de Informação)
GPL	General Public License (Licença Pública Geral)
ICMP	Internet Control Message Protocol (Protocolo de Mensagem de Controle da Internet)
IoT	Internet of Things (Internet das Coisas)
IP	Internet Protocol (Protocolo de Internet)
MAC	Media Access Control (Endereço Físico associado a Placa de Rede)
NIB	Networking Information Base (Base de Informações de Rede)
RSSI	Received Signal Strength Indicator (Indicador de intensidade do sinal recebido)
SDN	Software Defined Network (Rede Definida por Software)
SNMP	Simple Network Management Protocol(Protocolo Simples de Gerenciamento de Redes)
TI	Tecnologia da Informação
VM	Virtual Machine (Máquina Virtual)
WiFi	Wireless Fidelity (Rede sem Fio)

Sumário

1	Introdução	19
1.1	Objetivos	21
1.1.1	Objetivo Geral	21
1.1.2	Objetivos Específicos	21
1.2	Metodologia	21
2	Redes SDN e OpenFlow	23
2.1	Redes SDN	23
2.2	OpenFlow	25
2.3	Controladores	26
2.4	Mininet	30
3	SDN-WIFI	33
3.1	Proposta	33
3.2	Mininet-WiFi	33
3.2.1	Estrutura	33
3.3	Testes de Simulação de Mininet-WiFi	34
3.3.1	Instalação do Mininet-WiFi	34
3.3.2	Ambientes de Mininet-WiFi propostos	35
3.3.2.1	Um ponto de acesso	35
3.3.2.2	Múltiplos pontos de acesso	37
4	Testes de Simulação com o uso de Python	45
4.1	Mininet-WiFi: Python API e scripts	45
5	Conclusão	53
	Referências	55
	Apêndices	57
	APÊNDICE A – Instalação do computador virtual Mininet	59
	APÊNDICE B – Instalação do Controlador NOX	61

Anexos	65
ANEXO A – Script utilizado para ambiente Python: Modelo de Mobilidade	67
ANEXO B – Script Python: Modelo de Propagação	71
Índice	75

1 Introdução

As redes de computadores são elementos essenciais para o desenvolvimento de comunicações e a internet, entre os anos 90 e os anos 2000 teve um grande crescimento, tornando-se uma ferramenta essencial para as comunicações atuais (GANTZ; REINSEL, 2012). Concomitantemente com o crescimento da conhecida “internet das coisas” (Internet of things - IoT), diversos dispositivos hoje estão funcionando com os recursos da internet, além da crescente “computação em nuvem” ¹ (KOPETZ, 2011).

Atualmente sua estrutura está dentro de seus limites, sendo necessário implementar cada vez mais improvisações para suprir tal demanda, pois pelo fato de não se ter ocorrido planejamento para este crescimento de usuários e dispositivos com acesso a rede, os equipamentos que foram desenvolvidos tornaram-se limitados, impossibilitando a manipulação de seus softwares para atender diferentes ambientes.

Pode-se dizer que o modelo atual ficou “ossificado”, não permitindo que fossem realizadas atualizações necessárias para suprir uma demanda em crescimento exponencial (BAI; HELMY, 2004). Pelo fato de não terem sido realizadas atualizações consistentes no núcleo da rede e o modelo atual não permitir que sejam realizadas análises e propostas de melhoria, é de grande importância que sejam desenvolvidos equipamentos mais próximos da realidade atual, capazes de prover soluções que possam suprir esta demanda vigente, assim como planejar com melhor estruturação o seu crescimento. Com isso, entende-se que haja a necessidade de projetos para auxiliar no gerenciamento de tráfego e na confiabilidade da rede.

Estudos para propostas de rede dinâmicas não são novos. Na década de 90 já era estudado o conceito de redes ativas (FEAMSTER; REXFORD; ZEGURA, 2014), na qual foi possível repensar o projeto de infraestrutura de redes e seus protocolos e, com isso, começando a se consolidarem grupos de pesquisa dedicados a estudar modelos diferenciados de redes de computadores com maior centralização, capazes de suprir uma demanda que, na realidade da época não se imaginava possível. As melhores propostas partiram de SANE (CASADO et al., 2006), ETHANE (CASADO et al., 2007), que desenvolveram controles centralizados de regras de firewall em uma rede corporativa.

Porém foi somente a partir do ano de 2008 que o tema começou a se consolidar no ramo técnico-científico com a divulgação do paper de McKeown (MCKEOWN et al., 2008), no qual apresentava as vantagens da implementação de uma rede definida por software e a proposta do protocolo OpenFlow. Pelo fato do OpenFlow ser um protocolo aberto, gera

¹ computação em nuvem (em inglês, cloud computing) refere-se à utilização da memória e da capacidade de armazenamento e cálculo de computadores e servidores compartilhados e interligados por meio da Internet, seguindo o princípio da computação em grade.

economia para a fabricação de equipamentos. E o grupo desenvolveu soluções baseadas na estrutura de redes definidas por software, apresentando o modelo de uma possível rede em um Campus, mostrando que com essa nova estrutura seria possível definir novas formas de administração, novos protocolos assim como um novo modelo de gerenciamento de rede e balanceamento de carga, sem grandes prejuízos para estrutura existente e maior economia na aquisição de equipamentos.

Empresas e universidades começaram a se aprofundar sobre o protocolo OpenFlow e como consequência, a indústria começou a desenvolver produtos pensados para este tipo de rede e protocolo, desenvolvendo equipamentos de rede que utilizam o modelo OpenFlow em sua linha de equipamentos de comutação para implantação em backbones (MCKEOWN et al., 2008).

Com esse desenvolvimento, foi se consolidando o conceito de redes definidas por software (conhecido como SDN), bem como seus desafios para implementação na estrutura atual (SEZER et al., 2013). Porém ao vislumbrar a possibilidade de economia e bem como o desacoplamento do plano de controle do plano de dados, surgiram novas soluções para as redes, e por fim, ocorrendo a possibilidade de modificar a estrutura atual das redes.

Para que todos esses conceitos fossem implementados sem que gerasse riscos, foi também criado uma "engine" capaz de emular ambientes de diferentes complexidades e com isso realizar diferentes testes, bem como aprimorar os conhecimentos antes de qualquer migração de equipamentos para o uso de redes OpenFlow. Este modelo de ambiente simulado se chama Mininet, trata-se de um computador virtual, capaz de gerar diferentes ambientes de rede que funcionam à partir do conceito de redes SDN, bem como sob o protocolo OpenFlow, podendo assim simular Switches camada 2 que funcionam com o protocolo OpenFlow, roteadores e controladores de rede (LANTZ; HELLER, 2013a). Porém o ambiente Mininet carecia de ambientes com redes sem fio na qual diversos dispositivos estão conectados em redes Wi-Fi e podem se mover na rede. Considerando nessa dinâmica, bem como no uso de SDN, foi desenvolvido também um ambiente sem fio, conhecido como Mininet-Wifi, que trata de gerar a possibilidade de arquitetar ambientes de rede sem fio e realizar testes usando mobilidade (FONTES; ROTHENBERG, 2015).

Este trabalho irá apresentar os conceitos nos quais se baseiam estes ambientes e também apresentará seus usos e discussões sobre Redes Definidas por Software, OpenFlow, Mininet e Mininet-WiFi

1.1 Objetivos

1.1.1 Objetivo Geral

O objetivo do presente trabalho é apresentar uma alternativa ao sistema de controle centralizado de redes, com o uso de ferramentas que têm sido desenvolvidas no meio acadêmico e implementadas em escala industrial para redes de computadores, como ferramentas disponíveis para redes definidas por software ou SDN. Serão estudadas no presente trabalho o uso desta solução, bem como sua viabilidade no uso de redes SDN sem fio.

1.1.2 Objetivos Específicos

- Estudar redes SDN e OpenFlow , bem como sua utilidade no contexto atual.
- Apresentar como alternativa o uso de soluções SDN para problemas de redes
- Obter maior conhecimento de administração de dispositivos de rede sem fio, bem como suas possibilidades para resolução de problemas, realizando simulações através do uso da ferramenta Mininet-WiFi.

1.2 Metodologia

Este trabalho será realizado através da seguinte metodologia:

- Revisão bibliográfica de autores pertinentes para elaboração do tema.
- Testes de simulação de ambiente baseado na infraestrutura funcional através do uso de Mininet-WiFi.
- Análise de diferentes modelos de Rede simuladas com Mininet-WiFi.

2 Redes SDN e OpenFlow

2.1 Redes SDN

A arquitetura SDN trabalha com um conceito diferente dos modelos atuais. Basicamente a SDN permite que a rede seja separada em duas partes, o plano de controle e o plano de dados, com isso é possível que a rede seja diretamente programável com sua estrutura separada entre aplicações e serviços (LI; LIAO, 2013). Com isso, é possível trabalhar do mesmo modo que a equipe de TI atua com servidores virtuais, estes servidores podem ser criados em minutos, movidos de um host a outro de forma dinâmica, sem que o hardware físico gere algum impacto inesperado. As Redes Definidas por Software estão possibilitando que as empresas acelerem a implantação e a distribuição de aplicativos, reduzindo custos de TI por meio da automação de fluxos de trabalho compatíveis com suas políticas. A tecnologia SDN funciona em conjunto com as arquiteturas de nuvem, disponibilizando distribuição e mobilidade de serviços de forma automatizada, sob demanda e em grande escala. As SDN aprimoram os benefícios da virtualização do data center, aumentando a flexibilidade e a utilização de recursos e reduzindo custos e sobrecargas de infraestrutura (VELRAJAN, 2012) (LI; LIAO, 2013).

As SDN alcançam esses objetivos empresariais com a convergência do gerenciamento de serviços de rede e de aplicativos em plataformas de orquestração centralizadas e extensíveis que podem automatizar o provisionamento e a configuração de toda a rede. Com as políticas de TI comuns centralizadas, reúnem-se grupos e fluxos de trabalho de TI heterogêneos.

Atualmente o administrador da rede trabalha com configuração de equipamentos através de CLI (*Command Line Interface* - Interface de Linha de Comando), criando linhas de configuração para cada equipamento utilizado ou SNMP (*Simple Network Management Protocol* - Protocolo Simples de Gerenciamento de Redes) que, mesmo atuando de modo mais simples, utiliza a mesma metodologia, não havendo assim uma centralização do plano de controle.

Com o uso de SDN é possível criar estruturas de redes com a mesma agilidade nas quais se criam servidores virtuais, utilizando uma controladora SDN centralizada e aplicações SDN para que seja possível montar melhores tabelas de fluxo, substituindo assim as FIB's (*Forwarding Information Base*) dos equipamentos de rede. Em ambientes de camada 2, as tabelas de fluxo podem substituir os endereços MAC dos computadores. Com essas mudanças será possível que os desenvolvedores de soluções para rede possam criar aplicações que facilitem a segurança, assim como melhora QoS ou outros serviços,

pois é possível estabelecer dinamicamente todo o tráfego para um recurso específico de uma aplicação da rede. Enquanto o plano de controle é movido para a controladora SDN esta gerencia protocolos de padrão aberto como o OpenFlow e os equipamentos de rede poderão manter o plano de dados localmente (HEWLETT-PACKARD, 2014) (ONS, 2012). A Figura 1 apresenta a estrutura básica de uma rede SDN.

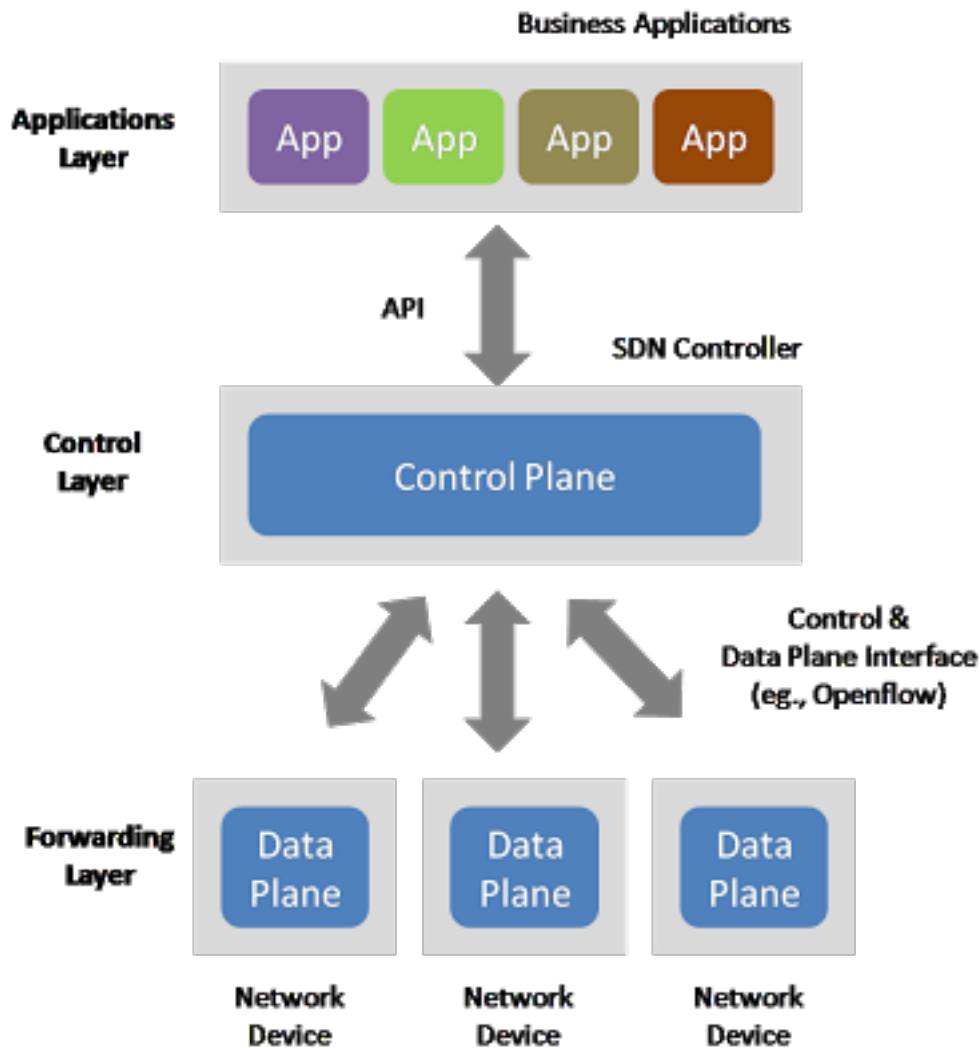


Figura 1: Modelo com plano de controle e plano de dados (VELRAJAN, 2012)

Redes do tipo SDN, são sistemas que tem como objetivo principal desacoplar o hardware da rede de seu sistema proprietário de configuração, ou seja, ao obter um equipamento de rede, seria possível gerenciá-lo de modo dinâmico de acordo as necessidades da rede. Dentro deste segmento o uso da ferramenta OpenFlow chega como uma boa alternativa, pois por se tratar de um sistema de código aberto é possível criar modelos mais dinâmicos e de acordo com as necessidades de cada rede.

2.2 OpenFlow

O protocolo OpenFlow é um protocolo padronizado para interagir com os comportamentos de encaminhamento feitos por equipamentos de rede de diferentes fornecedores e empresas. Isto nos permite controlar o comportamento destes dispositivos em toda rede de forma dinâmica. OpenFlow é um protocolo chave em muitas soluções SDN . Com ele o administrador da rede tem a possibilidade de personalizar as redes de acordo com as necessidades solicitadas, deixando o dispositivo mais robusto, pois é possível excluir serviços não utilizados no tráfego de pacotes, criando assim redes virtuais individuais para cada processo (PFAFF et al., 2011).

Normalmente em roteadores ou switches, o encaminhamento de pacotes e as decisões de alto nível do roteador são realizadas no mesmo dispositivo. Um OpenFlow Switch poderia, por exemplo, separar essas duas funções. A transferência de pacote de dados continuaria funcionando à partir do switch, enquanto as decisões de alto nível do roteador poderiam ser encaminhadas para um controlador separado, como um servidor padrão. A comunicação entre ambos se daria através do protocolo OpenFlow que, por mensagens definidas, acompanharia a transmissão de pacotes, seus encaminhamentos, bem como faria a obtenção do status do serviço.

O encaminhamento de pacotes de um OpenFlow Switch apresenta uma abstração mais clara da tabela de fluxo de dados. Cada entrada contém um conjunto de campos do pacote e uma ação a ser realizada. Quando um OpenFlow Switch recebe um pacote, o controlador realiza a decisão sobre o que fazer com esse pacote, como por exemplo excluir, ou encaminhá-lo para uma tabela que encaminhará o pacote diretamente ao destino.

Mesmo que esta alternativa para aumentar a dinâmica da rede já seja possível com outras soluções como, por exemplo, em equipamentos Cisco, com o protocolo OpenFlow existe a possibilidade de editar entradas e saídas e criar comandos personalizados, diferente do que é possível fazer com roteadores ou switches de software fechado (MCKEOWN et al., 2008). O que se pontua principalmente como vantagem da implementação de protocolo OpenFlow é que por ser um protocolo de software aberto, é possível que em um aumento de tráfego da rede, ao invés de mudar todo o hardware disponível em rede, pode-se apenas rever a programação, deixando-a mais leve e dinâmica, de acordo com a nova necessidade do local, gerando economia sem mudar a estrutura atual (Figura 2).

A comunidade que defende o uso de protocolos abertos como o OpenFlow teve grande preocupação que seu uso fosse possível para implementação em qualquer equipamento de diferentes empresas, e por isso deixou disponível em seu site a API com todas as configurações necessárias para que os fabricantes possam criar dispositivos que suportem este protocolo (PFAFF et al., 2011).

Com essa solução é possível gerenciar, através do protocolo OpenFlow, equipamentos

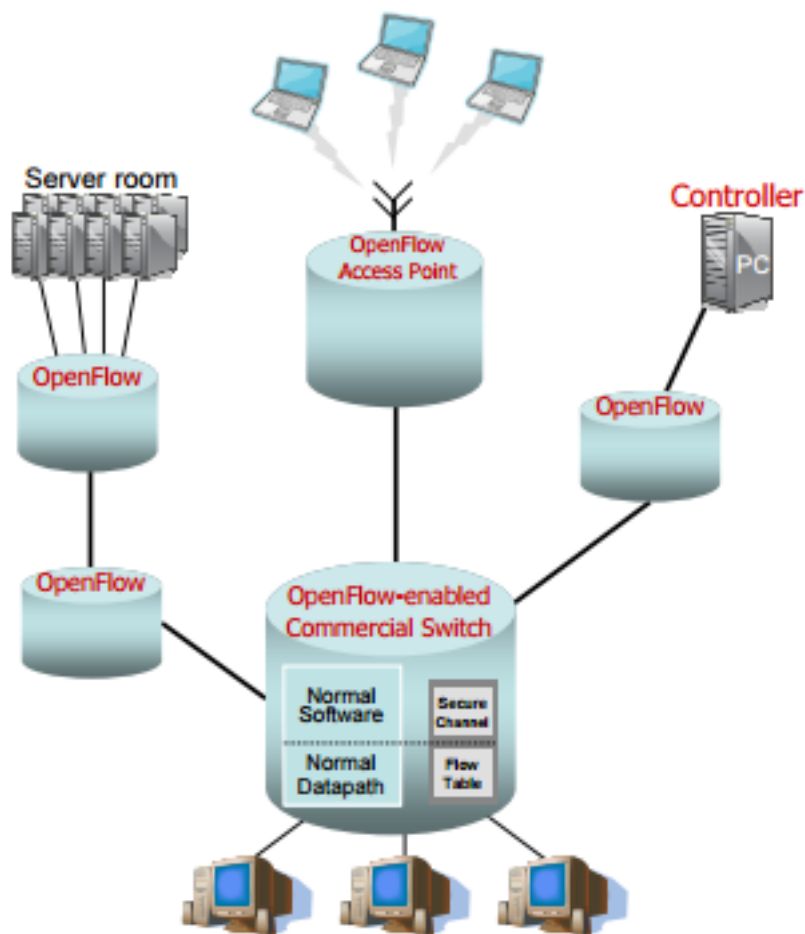


Figura 2: Projeto inicial de proposta de rede OpenFlow ([MCKEOWN et al., 2008](#))

de diferentes empresas sem precisar que cada switch seja configurado em interfaces diferentes. Se todos os dispositivos estiverem com soluções OpenFlow, o gerenciamento ficará centralizado, independente da marca do equipamento utilizado na rede, auxiliando empresas a cada vez usar dispositivos que melhorem na solução da infraestrutura, sem precisar se ater a marcas e modelos específicos.

OpenFlow permite que o profissional de redes possa facilmente prover roteamentos melhores e protocolos de comunicação diferenciados nas redes, que podem ser usados na mobilidade de máquinas virtuais, em redes de alta segurança e na próxima geração de endereçamento IP.

2.3 Controladores

Controladores SDN são como o cérebro de uma rede baseada em OpenFlow. Com esta estrutura, é possível ter um ambiente de programação, no qual o desenvolvedor pode ter acesso a todos eventos que estão ocorrendo em uma rede por sua interface padrão

OpenFlow e, com isso, é possível desenvolver comandos para auxiliar no controle da infraestrutura de chaveamento, implementar de modo mais simples políticas de segurança baseadas em níveis de abstração maiores do que endereços IP, cobrindo, por exemplo, áreas específicas da rede. Além disso com o uso de redes SDN, existe a possibilidade da implementação de controladores de rede capazes de monitorar a rede em tempo real, ou disponibilizar estruturas mais personalizadas, como, por exemplo, apresentar na tela um datacenter e mostrar que todos estes computadores estão centralizados em um único switch virtual (GUEDES et al., 2012). A Figura 3, apresenta o funcionamento de um controlador, realizando a divisão entre o plano de controle e o plano de dados.

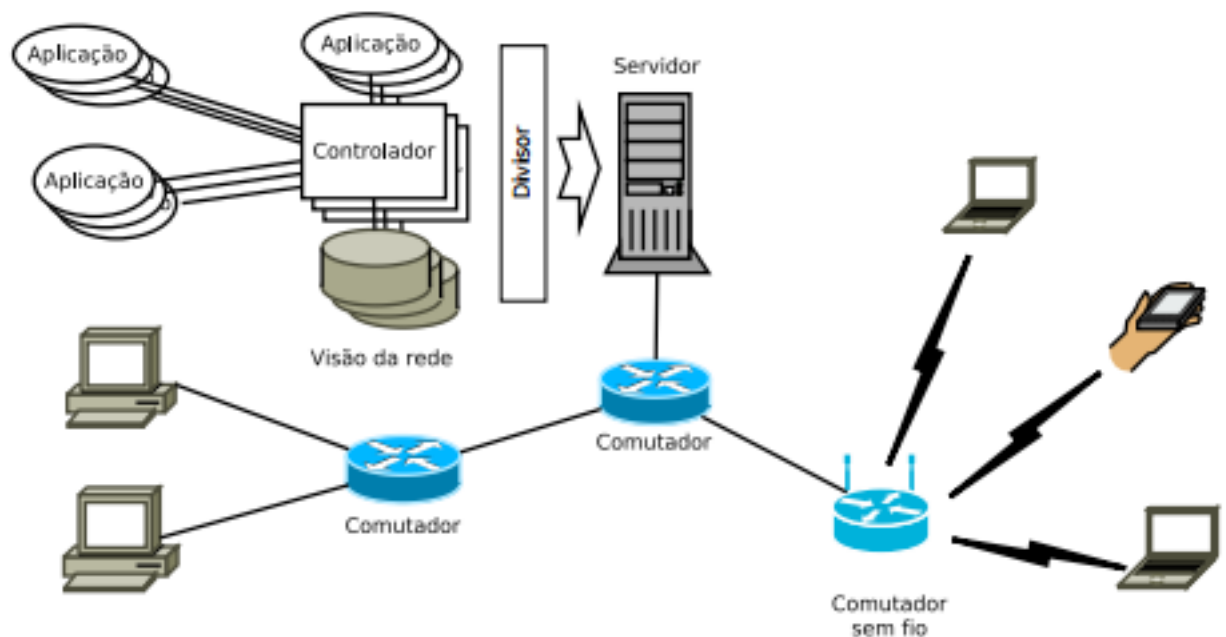


Figura 3: Estrutura geral do funcionamento de uma rede SDN Fonte: Anderson Coelho Weller - Unicamp

O primeiro controlador desenvolvido foi o NOX (C++) e depois foi doado para a comunidade. Além desse foram desenvolvidos diversos controladores, tais como POX e ONIX. Mais informações sobre o controlador NOX podem ser encontradas no Apêndice B.:

- ONIX - foi criado para gerenciar grandes redes de forma confiável (algoritmos de comparação em diferentes níveis) e, com isso, poder garantir maior confiabilidade no sistema provendo abstrações para particionar e distribuir o estado da rede em diversos controladores distribuídos, criando escalabilidade. Ou seja, a NIB (*Network information Base*) pode ser dividida entre controladores, garantindo com esse modelo, tolerância a falhas, pois através do modelo escalável, é possível criar níveis de administração, e com isso prover maior confiabilidade. A NIB replicada

no modelo mestre-escravo, surge quando um controlador é centralizado na rede, e outros controladores respondem ao controlador principal. Utiliza o modelo de NIB na qual a estrutura de dados que apresenta a visão global da rede é mostrada através de grafos. Suas aplicações podem ler e escrever a NIB e, automaticamente, pode atualizar switches e roteadores ([KOPONEN et al., 2010](#)).

- POX (Phyton) - Distribuído sob licença Apache, o controlador POX vem sendo desenvolvido como um sucessor do NOX, principalmente no que se trata de iniciativas de ensino e pesquisa e, por isso, o foco deste controlador não é o alto desempenho. Por este motivo, este controlador tem uma interface baseada em Phyton que auxilia no entendimento dos processos, além de oferecer melhor desempenho se comparado ao NOX. Oficialmente foi desenvolvido para ser um controlador de rede, porém hoje pode atuar como switch OpenFlow e pode ser útil para desenhar modelos de rede em geral ([KHONDOKER et al., 2014](#)) ([MCCAULEY; SCOTT; GUEDES, 2004](#)).
- Ryu (Phyton) - Criado pela NTT, Ryu é um controlador baseado em componentes. Neste controlador há um grupo pré definido de componentes que podem ser modificados, estendidos ou até personalizados. Para se criar um novo componente pode-se usar qualquer linguagem de programação. Possui integração OpenStack ([KHONDOKER et al., 2014](#)) ([RYU, 2015](#)).
- Floodlight (Java) - Desenvolvido à partir do Beacon, o controlador Floodlight consiste em um grupo de módulos, no qual cada módulo disponibiliza um tipo de serviço e todos os módulos se comunicam entre si através de um controlador lógico por um API em Java ou em REST. Possui fácil configuração para se construir estruturas de dependências mínimas. Construído sob licença Apache, tem o apoio da Big Switch Networks, que auxilia no desenvolvimento de produtos mais consistentes e de uso nos equipamentos da empresa, e consiste em módulos que exportam os serviços. Também permite realizar integração com redes não- OpenFlow e compatível com a ferramenta de simulação Mininet. Sua estrutura possui integração OpenStack ([KHONDOKER et al., 2014](#)) ([CARDOSO et al., 2011](#)). A Figura 4 apresenta a estrutura do controlador Floodlight.

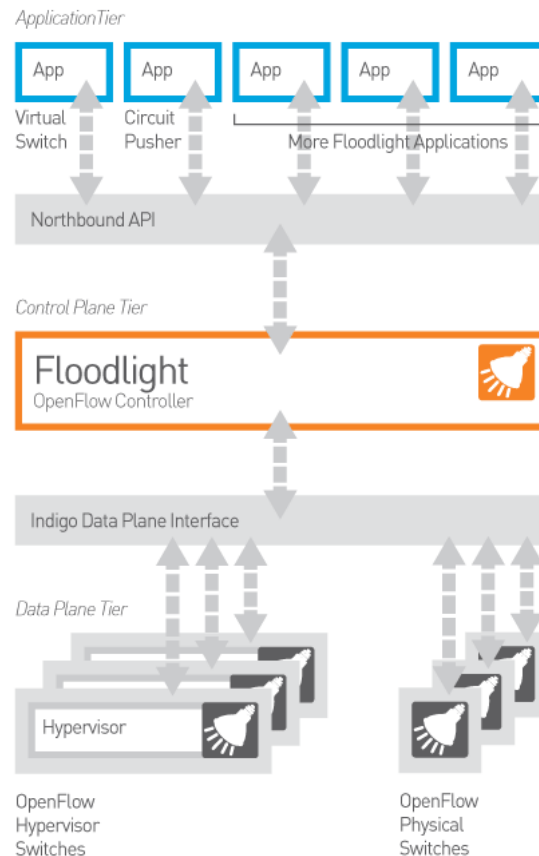


Figura 4: Funcionamento do controlador Floodlight. Fonte www.floodlight.com

- OpenDaylight (Java) - Trata-se de um projeto de software aberto baseado nos modelos utilizados em sistemas operacionais Linux. O maior objetivo do projeto é criar uma estrutura sólida que atenda o maior grupo de necessidades requeridas para componentes na arquitetura SDN, para se conseguir aceitação entre fornecedores e clientes, e para aumentar o crescimento da comunidade que contribui com o código assim como seu uso para o comércio de produtos e serviços. O OpenDaylight hoje é de alta disponibilidade, modular, extensível, escalável e multiprotocolar. Possui integração OpenStack (KHONDOKER et al., 2014) (LINUX-FOUNDATION, 2013). A Figura 5 apresenta a estrutura de funcionamento do controlador OpenDaylight.

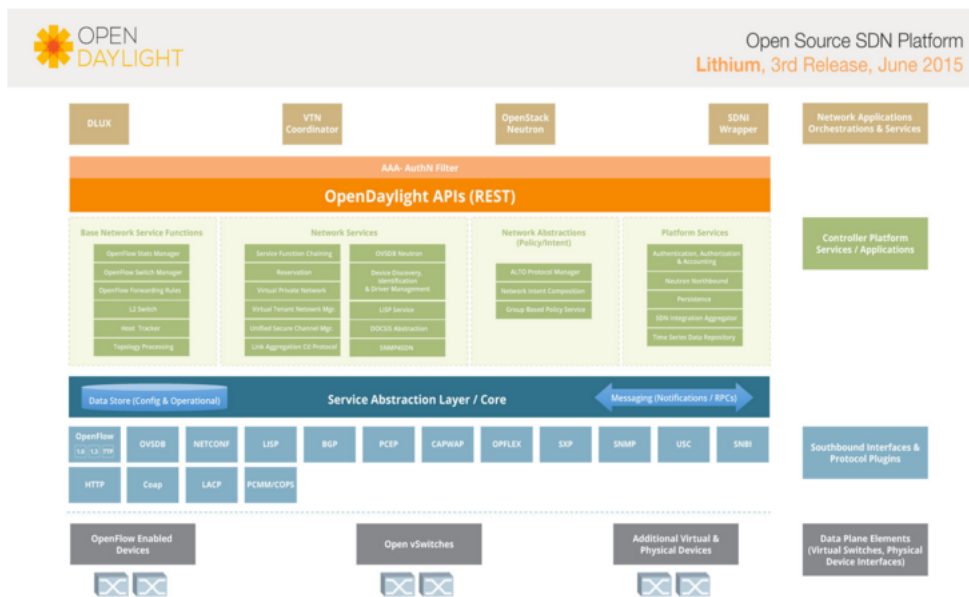


Figura 5: Estrutura do controlador OpenDaylight. Fonte www.opendaylight.org

2.4 Mininet

Para que fosse possível aprender a utilizar as ferramentas OpenFlow e com isso poder realizar simulações sem influenciar diretamente na estrutura da rede local foi desenvolvido um computador virtual comumente conhecido como Mininet (LANTZ; HELLER, 2013b). Trata-se de uma máquina virtual que funciona geralmente em Linux, e que contém os mesmos códigos aplicados nos switches e roteadores OpenFlow. Pelo fato de ser possível interagir facilmente com a rede utilizando Mininet este modelo tornou-se fundamental para implantação, ensino e pesquisa de redes dinâmicas. A Figura 6 apresenta algumas das telas de saída no funcionamento do ambiente Mininet.

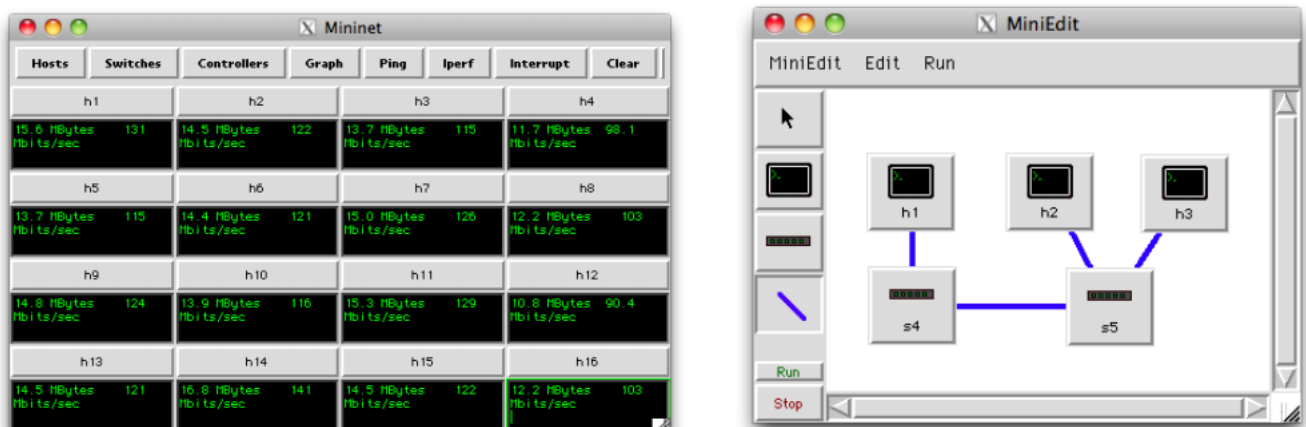


Figura 6: Modo interativo de gerenciamento de rede com Mininet

Por se tratar de uma ferramenta colaborativa, o Mininet é um produto que está em constantes atualizações e, com a implementação de novas funcionalidades. Na Internet é possível encontrar diversos tutoriais para uso e exemplos de simulações ([LANTZ; HELLER, 2013b](#)) ([LANTZ; HELLER, 2013a](#)). Mais sobre a instalação do Mininet pode ser visto no Apêndice [A](#).

3 SDN-WIFI

3.1 Proposta

Com o crescimento e consolidação do uso de redes SDN, principalmente para a emulação de redes e análise de desempenho, foi possível à partir desse conceito aprimorar seu uso para diferentes tipos de rede e melhores propostas de solução para depuração de protocolos. Assim foi possível desenvolver serviços como o SDN-WIFI, que baseado nos mesmos moldes de redes SDN também é possível realizar emulação de redes WiFi utilizando os mesmos conceitos de redes SDN com protocolos OpenFlow dentro da estrutura de Mininet, proposto inicialmente por Ramon Fontes da FEE-Unicamp ([FONTES; ROTHENBERG, 2015](#)).

Com este tipo de rede é possível realizar os mesmos procedimentos de controles para ponto de acesso no mesmos moldes de SDN (separação entre plano de controle e plano de dados) e com isso executar os procedimentos aplicados em OpenFlow .

As redes definidas por software voltadas para as redes sem fio (*Software Defined Wireless Network*) tornaram-se assim um ramo de pesquisa de grande importância, pois complementa a utilização de serviços baseados em redes SDN para redes sem fio ([YANG et al., 2015](#)). Mesmo com o OpenFlow não sendo capaz de resolver todos os problemas baseados em redes sem fio, é possível que através da utilização do Mininet possam ser realizados em ambientes controlados para que seja possível prover soluções pertinentes.

3.2 Mininet-WiFi

Com o Mininet-WiFi é possível simular todo ambiente existente no Mininet, com a vantagem de se acrescentar a este modelo pontos de acesso e hosts sem fio, e com isso efetuar diferentes simulações para outros modelos de ambiente proposto.

3.2.1 Estrutura

Todo modelo Mininet-WiFi é baseado no mesmo sistema de Mininet comum, gerando endereços MAC para as máquinas virtuais, assim como seus endereçamentos IP funcionam através de namespaces. As estações se comunicam através de um processo de autenticação, se comunicando com as estações de trabalho Mininet tanto cabeadas quanto sem fio. O funcionamento do Mininet-WiFi acontece de maneira semelhante ao funcionamento do Mininet padrão, mas caso a estrutura seja inicializada com o modo WiFi ativado, ele irá inicializar partindo de uma estrutura que compõe 3 interfaces sem

fio (inicialização padrão). Para que seja possível inicializar o Mininet no modo WiFi, deve-se ao digitar o seguinte comando `sudo mn --wifi` na CLI do Mininet-WiFi como apresentado abaixo:

```
$ sudo mn --wifi
[sudo] password for wifi:
*** Creating network
*** Adding controller
*** Adding hosts and stations:
sta1 sta2
*** Adding switches and access point(s):
ap1
*** Adding links and associating station(s):
(sta1, ap1) (sta2, ap1)
*** Starting controller(s)
c0
*** Starting switches and access points
ap1 ...
*** Starting CLI \index{CLI}:
mininet-wifi>
```

Além da possibilidade de utilização de comandos, é possível também construir topologias que estão dentro da pasta *examples*. Pode-se reproduzir exemplos do tipo *ad-hoc*, para simulação de redes tipo *ad-hoc*, *2AccessPoints* que cria uma topologia com duas estações e a comunicação entre as estações associadas a diferentes pontos de acesso, entre outros.

3.3 Testes de Simulação de Mininet-WiFi

Nesta etapa serão apresentados os testes realizados de funcionamento da estrutura criada em mininet-WiFi bem como os resultados obtidos.

3.3.1 Instalação do Mininet-WiFi

Para que seja realizada a instalação do Mininet-WiFi é recomendável a instalação de um computador virtual Linux Ubuntu (para realização deste trabalho foi utilizada a versão 14.04). Para seguir os mesmos padrões do desenvolvedor, o computador foi nomeado como *wifi*. Para seu funcionamento, a máquina hospedeira na qual o computador virtual foi instalado precisa ter um dispositivo de rede WiFi, pois o Mininet-WiFi emula os computadores com acesso a rede sem fio, baseado na estrutura original do computador

que realiza o controle. Após a instalação do sistema operacional, será necessário realizar os downloads dos pacotes disponíveis para o funcionamento do Mininet WiFi ([FONTES; ROTHENBERG, 2015](#)), realizando a instalação completa, ou somente os pacotes necessários para a simulação desejada ([LINKLETTER, 2016](#)).

É importante citar que para as simulações realizadas em Mininet-WiFi todo acesso realizado em computador virtual é feito remotamente para que a máquina real utilize a CLI dos computadores virtuais a fim de uma melhor centralização e realização dos testes propostos.

3.3.2 Ambientes de Mininet-WiFi propostos

Nesta etapa para testar o funcionamento do Mininet WiFi, iremos realizar os seguintes testes ([LINKLETTER, 2016](#)):

- Um ponto de acesso - será apresentado o modo de funcionamento mais simples do Mininet-WiFi, apresentando os métodos para capturar tráfego wireless em uma rede Mininet-WiFi, bem como discussões pertinentes sobre OpenFlow e redes sem fio.
- Múltiplos pontos de acesso - será apresentado o método de criação de uma rede de topologia mais complexa.
- Python e arquivos de scripts - serão apresentados os meios de como se criar topologias mais complexas usando Mininet-WiFi
- Mobilidade - Mostra como criar um cenário de rede móvel, cujas estações podem se mover através do espaço para dentro ou fora do alcance dos pontos de acesso. Ele também aborda as funções disponíveis que podem ser utilizadas para implementar diferentes modelos de mobilidade usando a API Mininet-WiFi Python.

3.3.2.1 Um ponto de acesso

Este é o modelo mais simples da topologia padrão de Mininet-WiFi, que consiste em um ponto de acesso sem fio e duas estações remotas. O ponto de acesso consiste em um switch conectado a um controlador e as estações são os hosts. Com este modelo, será possível apresentar a captura do controle de tráfego e demonstrar o funcionamento do OpenFlow, para visualizar o tráfego de rede sem fio na interface *Wlan*.

Capturar o controle de tráfego realizado pelo Mininet-WiFi

- Iniciar o Mininet-WiFi para um ambiente padrão através do seguinte comando: `sudo mn --wifi` na CLI do Mininet-WiFi; o resultado obtido será a compilação do ambiente mais simples do Mininet WiFi, composto por um ponto de acesso e duas estações remotas.
- Habilitar a interface `hwsim0` que é um software criado pelo Mininet-WiFi que copia todo tráfego de todas as interfaces wireless. Isso facilita a visualização de todo tráfego sem fio. Para habilitar o serviço utilize o seguinte comando: `mininet-wifi> sh ifconfig hwsim0 up`
- Inicializar o wireshark em uma nova aba, conectando-se remotamente ao Mininet-WiFi e executando o seguinte comando: `wifi@wifi: $ sudo wireshark &`. Selecione o tráfego para a interface habilitada conforme a Figura 7

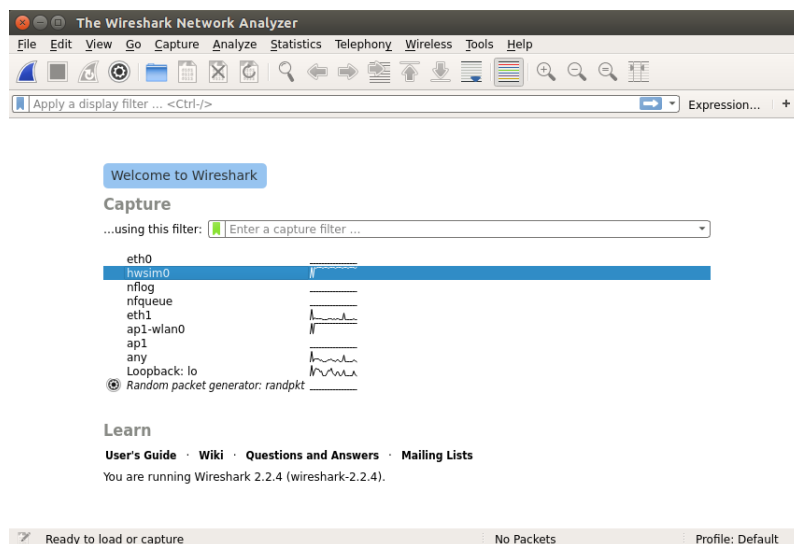


Figura 7: Tela inicial do Wireshark

É possível verificar o controle de tráfego executando o comando *ping*. Pode ser feito da seguinte forma, utilizando a CLI do computador virtual: `mininet-wifi> sta1 ping sta2`

Assim, pode-se visualizar a resposta apresentada pela requisição ARP (trata-se de uma mensagem de broadcast que ao ser enviada responde qual é o endereço físico associado ao endereço IP) (TAROUCO, 1998) para broadcast buscando o host de destino, e o funcionamento do protocolo ICMP (*Internet Control Message Protocol* - Protocolo que permite o transporte de mensagens de controle e mensagens de teste entre equipamentos da internet) (FALL; STEVENS, 2011) mostrados na Figura 8.

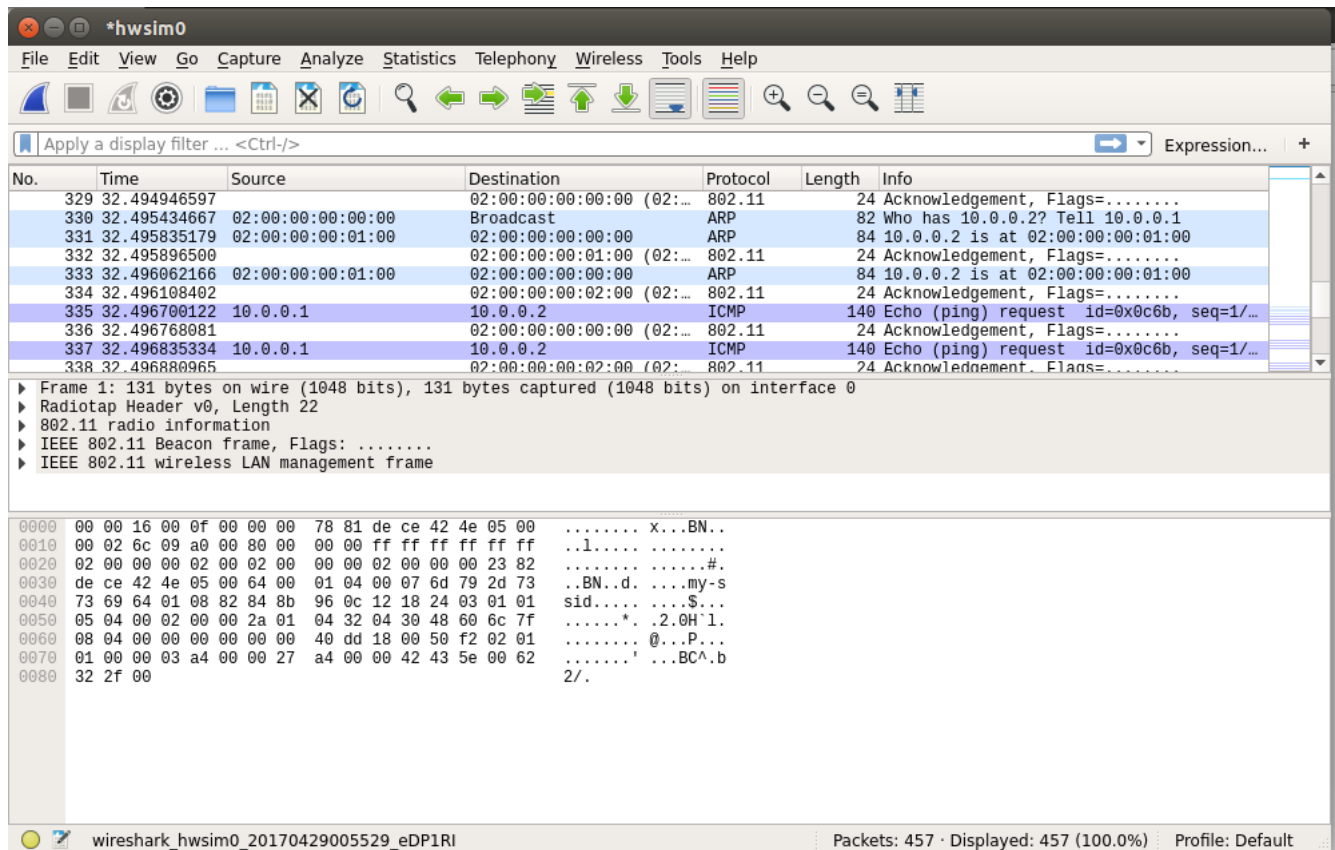


Figura 8: Tráfego monitorado pela interface hwsim0

Para finalizar esta simulação feche o mininet e remova todas as configurações realizadas através do comando:

```
mininet-wifi> exit

wifi:$ sudo mn -c
```

3.3.2.2 Múltiplos pontos de acesso

Nesta etapa será criada uma topologia linear, na qual uma estação está ligada a cada ponto de acesso.

- Iniciar o Mininet para uma topologia linear com três pontos de acesso através do seguinte comando:

```
wifi:$ sudo mn -wifi -topo linear,3
```

A Figura 9 representa a topologia linear a ser representada nesta seção.

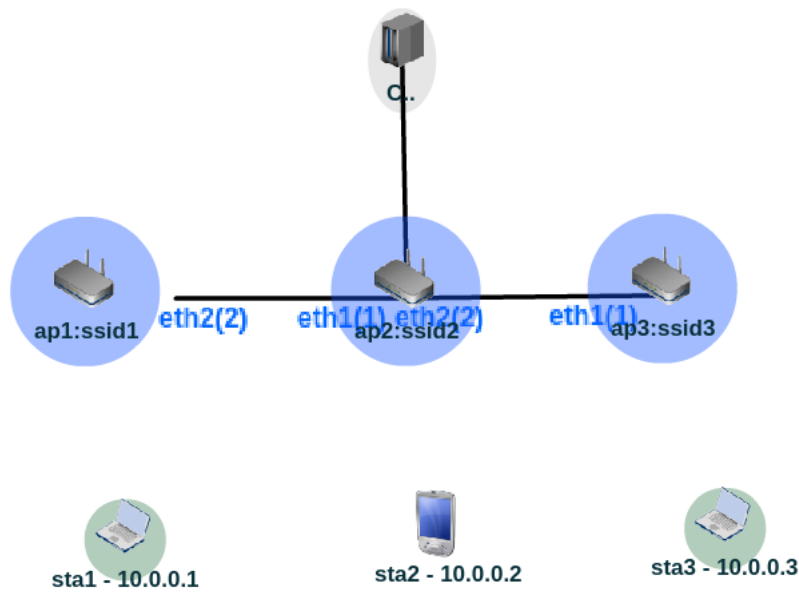


Figura 9: Topologia Linear criada para Mininet-WiFi

Através do comando `net`, é possível visualizar todas as conexões que cada estação de trabalho (`stax`) e todos os pontos de acesso (`apx`) estão realizando durante o processo (onde “x” é representado pelo número do ponto de acesso ou da estação móvel).

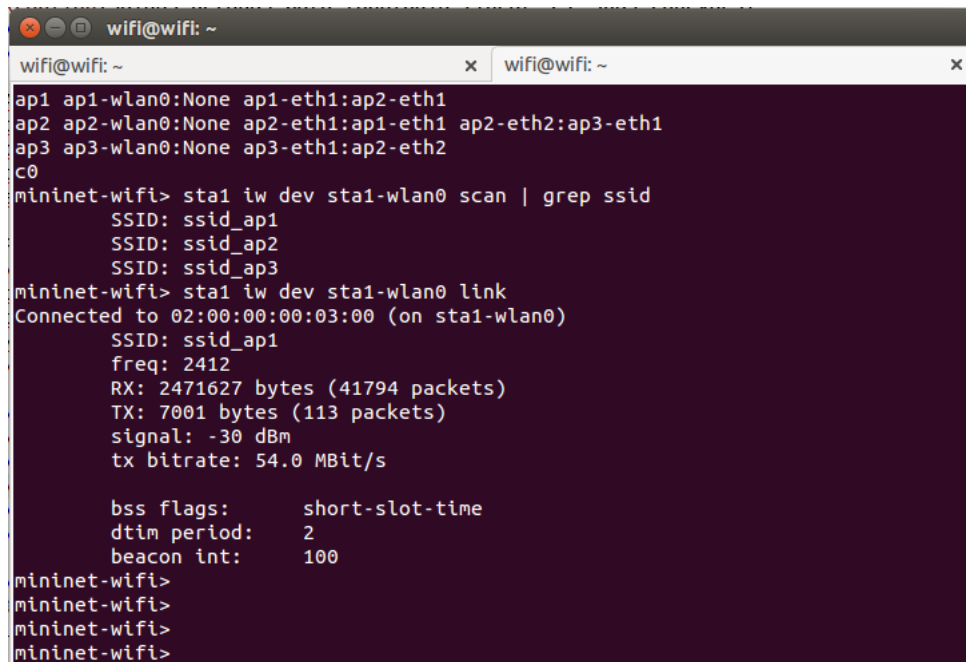
Como saída dos comandos executados, é possível notar que `ap1`, `ap2` e `ap3` estão todos conectados ao `wlan0` pelo modelo linear, porém não há nenhuma informação sobre quais *access points* eles estão conectados. Por isso, é preciso executar o comando `iw` em cada estação para observar a que ponto de acesso cada um está associado.

Para verificar quais os pontos de acesso são “visíveis” para cada estação, pode-se utilizar o comando `mininet-wifi> sta1 iw dev sta1-wlan0 scan | grep ssid`. Com isso, é possível verificar qual ponto de acesso está visível para cada estação. Através do uso deste mesmo comando, pode-se também visualizar a quais conexões cada estação está alocada. Por exemplo, para ver o ponto de acesso ao qual estação de `sta1` está ligado, use o seguinte comando: `mininet-wifi> sta1 iw dev sta1-wlan0 link`. Os resultados obtidos são apresentados na Figura 10.

Para este cenário, foi apresentado um ganho de recepção de `-30dBm`, porém para detectar se esse valor poderia simular um ambiente real, foi considerado uma antena transmissora WiFi de 2.4 GHz que utiliza o protocolo 802.11b.

Cenário Móvel Simples Continuando no modelo de topologia linear, cada estação está ligada a um ponto de acesso diferente, como já apresentado na Figura 9. É possível, com o uso do comando `iw` alterar o ponto de acesso a que cada estação está conectada.

É importante salientar que comandos como `iw` são usados em cenários estáticos,



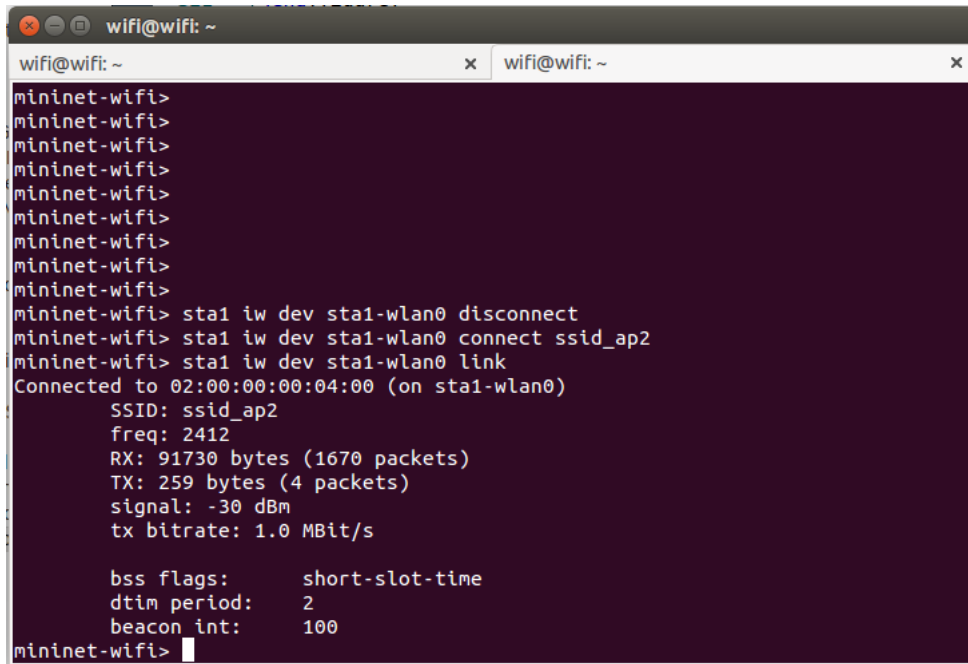
```
wifi@wifi: ~
wifi@wifi: ~
ap1 ap1-wlan0:None ap1-eth1:ap2-eth1
ap2 ap2-wlan0:None ap2-eth1:ap1-eth1 ap2-eth2:ap3-eth1
ap3 ap3-wlan0:None ap3-eth1:ap2-eth2
c0
mininet-wifi> sta1 iw dev sta1-wlan0 scan | grep ssid
        SSID: ssid_ap1
        SSID: ssid_ap2
        SSID: ssid_ap3
mininet-wifi> sta1 iw dev sta1-wlan0 link
Connected to 02:00:00:00:03:00 (on sta1-wlan0)
        SSID: ssid_ap1
        freq: 2412
        RX: 2471627 bytes (41794 packets)
        TX: 7001 bytes (113 packets)
        signal: -30 dBm
        tx bitrate: 54.0 MBit/s

        bss flags:      short-slot-time
        dtim period:    2
        beacon int:     100
mininet-wifi>
mininet-wifi>
mininet-wifi>
mininet-wifi>
```

Figura 10: Saídas geradas pelo comando iw

porém não deve ser usado quando Mininet-WiFi atribui automaticamente associações em cenários móveis mais realistas, pois normalmente estações móveis irão se desconectar e se reconectar automaticamente em diferentes pontos de acesso.

Caso `sta1` esteja conectado em `ap1`, e queremos conectá-lo em `ap2`. É possível alternar manualmente através dos seguintes comandos: `mininet-wifi> sta1 iw dev sta1-wlan0 disconnect`, para remover `sta1` de `ap1` e “`mininet-wifi> sta1 iw dev sta1-wlan0 connect ssid_ap2`”, para conectar `sta1` ao `ap2`. É possível verificar a mudança efetuada através do comando `iw link` no Mininet-WiFi. O resultado obtido pode ser visto na Figura 11.



```
wifi@wifi: ~
wifi@wifi: ~
mininet-wifi>
mininet-wifi>
mininet-wifi>
mininet-wifi>
mininet-wifi>
mininet-wifi>
mininet-wifi>
mininet-wifi>
mininet-wifi> sta1 iw dev sta1-wlan0 disconnect
mininet-wifi> sta1 iw dev sta1-wlan0 connect ssid_ap2
mininet-wifi> sta1 iw dev sta1-wlan0 link
Connected to 02:00:00:00:04:00 (on sta1-wlan0)
    SSID: ssid_ap2
    freq: 2412
    RX: 91730 bytes (1670 packets)
    TX: 259 bytes (4 packets)
    signal: -30 dBm
    tx bitrate: 1.0 MBit/s

    bss flags:      short-slot-time
    dtim period:    2
    beacon int:     100
mininet-wifi>
```

Figura 11: Troca de conexões entre ap1 e ap2 para st1

OpenFlow em um cenário Móvel Pode-se ver como o controlador de referência Mininet lida com esse cenário de mobilidade simples. Será necessário obter algum tráfego entre sta1 e sta3 de uma maneira que seja possível acessar a linha de comando Mininet-WiFi. É possível fazer isso executando o comando *ping* em uma janela xterm (CLI da estação) no sta3.

Analisando inicialmente o tráfego na interface apresentadas pelo Wireshark, é possível perceber que ao realizar a execução do comando *ping*, inicia-se um aumento do tráfego nas placas referentes a sta1 e sta3, enquanto em sta2 não ocorre tráfego, pois trata-se de um comando direcional que envia uma mensagem ICMP para o dispositivo referenciado (Figura 12).

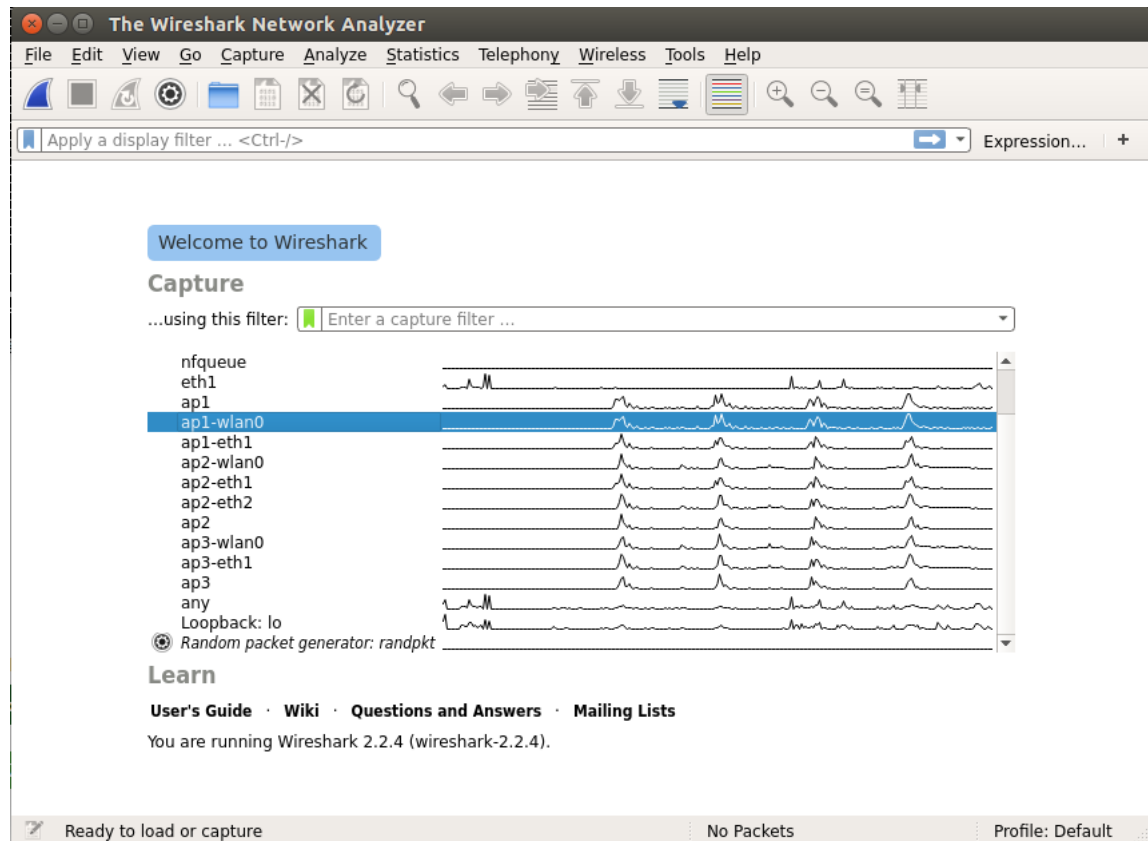


Figura 12: Saída comando ping entre Sta1 e Sta3

Através do comando `dump`, é possível ver que o controlador está endereçado a placa de rede *loopback* da máquina. Por isso, o tráfego será direcionado para esta interface.

Com o uso do comando `mininet-wifi> xterm sta3`, podemos abrir um terminal para a *sta3* e, com isso, realizar testes de conectividade, executando o comando *ping*. Pode-se notar que há um atraso entre a primeira requisição do comando em relação as seguintes. Isso ocorre pelo mesmo procedimento que foi apresentado na análise da Figura 8, pois inicialmente ao realizar o comando *ping*, o computador busca na rede a interface associada ao endereço de rede no qual se deseja obter a resposta do comando *ping*. O resultado é apresentado na Figura 13.

```

UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
RX packets:3420 errors:0 dropped:0 overruns:0 frame:0
TX packets:8 errors:0 dropped:0 overruns:0 carrier:0
collisions:0 txqueuelen:1000
RX bytes:644251 (644.2 KB) TX bytes:808 (808.0 B)

root@wifi:~# ping 10.0.0.1
PING 10.0.0.1 (10.0.0.1) 56(84) bytes of data:
64 bytes from 10.0.0.1: icmp_seq=1 ttl=64 time=73.8 ms
64 bytes from 10.0.0.1: icmp_seq=2 ttl=64 time=12.4 ms
64 bytes from 10.0.0.1: icmp_seq=3 ttl=64 time=1.54 ms
64 bytes from 10.0.0.1: icmp_seq=4 ttl=64 time=1.63 ms
64 bytes from 10.0.0.1: icmp_seq=5 ttl=64 time=0.818 ms
64 bytes from 10.0.0.1: icmp_seq=6 ttl=64 time=0.794 ms
64 bytes from 10.0.0.1: icmp_seq=7 ttl=64 time=7.15 ms
64 bytes from 10.0.0.1: icmp_seq=8 ttl=64 time=0.844 ms
64 bytes from 10.0.0.1: icmp_seq=9 ttl=64 time=1.32 ms
64 bytes from 10.0.0.1: icmp_seq=10 ttl=64 time=1.48 ms
64 bytes from 10.0.0.1: icmp_seq=11 ttl=64 time=2.33 ms
64 bytes from 10.0.0.1: icmp_seq=12 ttl=64 time=1.00 ms
64 bytes from 10.0.0.1: icmp_seq=13 ttl=64 time=1.48 ms
64 bytes from 10.0.0.1: icmp_seq=14 ttl=64 time=2.18 ms
64 bytes from 10.0.0.1: icmp_seq=15 ttl=64 time=2.11 ms

```

Figura 13: Xterm para sta3

No Wireshark é possível utilizar filtros para facilitar a visualização do ambiente. Em ambiente virtualizado, o Mininet utiliza como controlador local o próprio endereço da máquina, ou seja, a placa de rede *loopback*. Com o uso do filtro para visualização do protocolo OpenFlow, para a interface de rede *loopback*. É possível verificar as entradas e saídas do protocolo realizadas pelo protocolo OpenFlow, ou seja, enquanto no plano de dados ocorre o envio e resposta do comando *ping*, no plano de controle, através do protocolo OpenFlow, todos os pacotes passam pelo protocolo e realizam o encaminhamento. A Figura 14 apresenta o funcionamento do protocolo OpenFlow, atuando no plano de controle, gerenciando o tráfego da solicitação de *ping* realizada no plano de dados.

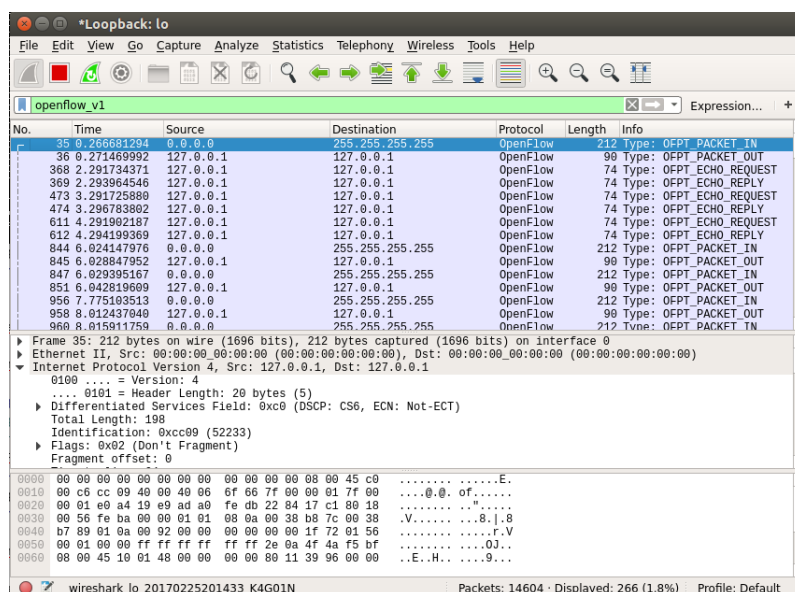


Figura 14: Fluxo do protocolo OpenFlow através com o Wireshark

Pelo comando `dpctl dump-flows` pode-se confirmar que todo o fluxo ocorre entre ap2 e ap3, sendo possível confirmar que o ap1 foi desconectado.

Caso conecte novamente a estação *sta1*, por exemplo, para o ponto de acesso *1* novamente e queira checar se há fluxo ocorrendo no *ap1* não será possível, pois o Mininet-WiFi não atualiza a cache para visualizar os fluxos de novas conexões. Para resolver este problema, pode-se deletar todos os fluxos anteriores com o uso do comando `dpctl del-flows` e utilizar o comando de visualização de fluxos novamente.

Com o uso das ferramentas citadas anteriormente foi possível mostrar como o controlador de referência Mininet trabalha em Mininet-WiFi. O controlador de referência Mininet não tem a capacidade de detectar quando uma estação move a partir de um ponto de acesso para outro. Quando isso ocorre, é necessário excluir os fluxos existentes para que novos fluxos possam ser criados. Uma possível solução para esta situação seria o uso de um controlador mais avançado, tal como OpenDaylight, para permitir a mobilidade da estação.

4 Testes de Simulação com o uso de Python

4.1 Mininet-WiFi: Python API e scripts

Teste com modelo de mobilidade Para que fosse possível demonstrar como o Mininet-WiFi cria cenários com estações móveis que exista handover (desconexão de um ponto de acesso e reconexão em outro ponto) entre os pontos de acesso, pode-se criar um script que monta um ambiente com estações móveis, e com isso, criar deslocamentos que permita a passagem por diferentes pontos de acesso.

Neste exemplo serão criados dois pontos de acesso (ap1, e ap2), em posições fixas, com distância de 50 m entre eles, três estações móveis, se movimentando de modo aleatório e também será criado um host *h1*, para funcionar como uma estação de trabalho fixa, para assim, realizar testes com as estações móveis. Estes ambientes foram criados com a simulação de um controlador genérico *c0*. Em ambientes de rede SDN no qual funciona o protocolo OpenFlow, os controladores tem função vital na rede, definindo os melhores caminhos para o fluxo de dados, o balanceamento do tráfego. Além de decidir em qual ponto de acesso cada estação irá se conectar, criando redes virtuais para cada grupo de trabalho específico, e com isso, gerar menos conflito no fluxo do tráfego entre diferentes pontos. O script disponibilizado está disponível no Anexo [A](#)

Neste script foram criados dois pontos de acesso fixos e três estações de trabalho que se deslocam de modo aleatório, seguindo os padrões de deslocamento do modelo de mobilidade “*Random Walk*”, foram realizados testes de conectividade e discutidos seus funcionamentos durante a realização dos testes. A janela inicial que mostra o funcionamento do script é apresentada na Figura [15](#).

Comandos de potência de sinal e distância O comando *distance* mostra a distância entre os nós. Utilizando este comando, pode-se obter saídas como a apresentada abaixo, que mostra a distância entre uma estação e um ponto de acesso:

```
mininet-wifi> distance sta1 ap1
The distance between sta1 and ap1 is 53.09 meters
```

```
mininet-wifi> distance sta1 ap2
The distance between sta1 and ap2 is 86.65 meters
```

```
mininet-wifi> distance sta1 sta2
The distance between sta1 and sta2 is 67.13 meters
```

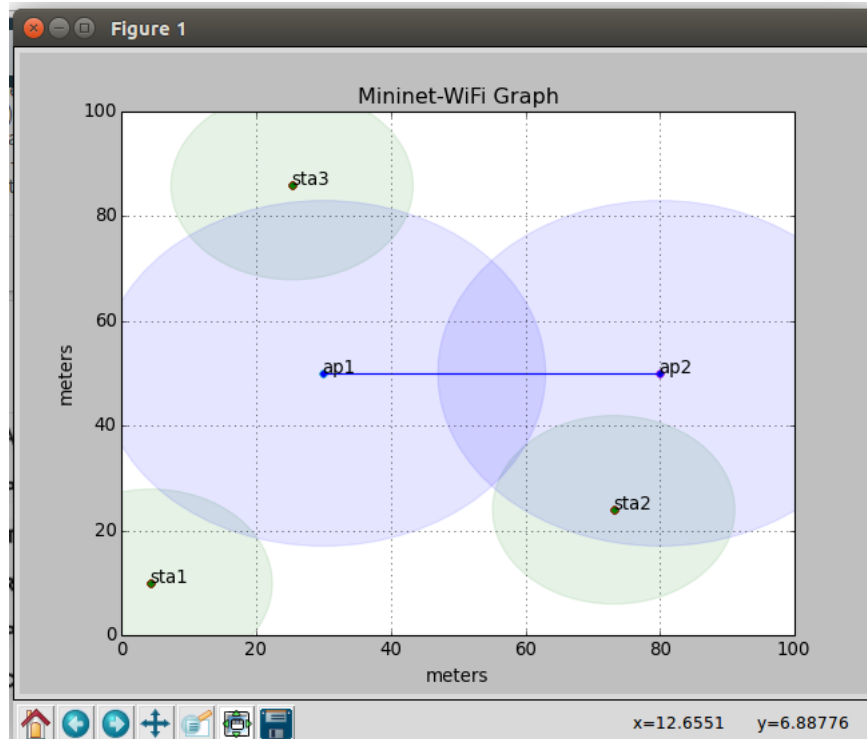


Figura 15: Gráfico com a topologia do script position-test.py

Também é possível obter maiores informações sobre potência do sinal e qual *access point* o nó está conectado, através do comando *info*.

```
mininet-wifi> info ap2
Tx-Power: 14 dBm
SSID: ssid-ap2
Number of Associated Stations: 1
```

```
mininet-wifi> info sta2
-----
Interface: sta2-wlan0
Associated To: ap2
Frequency: 2.417 GHz
Signal level: -44.73 dbm
Tx-Power: 14 dBm
```

O comando *info* apresenta a potência do sinal que foi recebido pela estação, o resultado calculado no Mininet-WiFi é -44,73 dBm. Este valor mudará caso a estação seja

reposicionada.

É possível também obter informações sobre o status da rede e executar comandos utilizando diretamente o python, através do comando *py*:

```
mininet-wifi> py ap1.associatedStations
[<Host sta3: sta3-wlan0:10.0.0.4 pid=11290> ]
```

Execução do comando nos nós Ao executar um cenário, os usuários podem fazer alterações de configuração em nós para implementar algumas funcionalidades adicionais. Isso pode ser feito a partir da linha de comando Mininet-WiFi, enviando comandos para *shell* de comando do nó. Ao iniciar o comando com o nome do nó seguido por um espaço, digita-se o comando a ser executado nesse nó.

Por exemplo, para ver as informações sobre a interface WLAN em uma estação chamada *sta1*, pode ser executado o comando:

```
Mininet-wi-fi> sta1 iw dev sta1-wlan0
```

Outra maneira de executar comandos em nós é abrir uma janela *Xterm* nesse nó e entrar comandos na janela de *Xterm*. Por exemplo, para abrir uma janela *xterm* na estação *STA1*:

```
Mininet-wi-fi> xterm sta1
```

Teste com *iperf* Para analisar como o sistema responde ao tráfego, será gerado tráfego entre o host *h1* e *sta3* assim que o ambiente for inicializado. Para isso iremos utilizado o comando *iperf* deixando a máquina 3 com a porta 5001 escutando, e o host *h1* realizando testes de conexão na porta e assim gerar o tráfego requerido.

Para isso será necessário realizar as seguintes etapas:

- Inicialize o script *line.py*. Depois inicialize um servidor *iperf* em uma estação.

```
mininet-wifi> sta3 iperf --server &
```

```
-----
Server listening on TCP port 5001
TCP window size: 85.3 KByte (default)
-----
```

- Inicialize o *xterm* para o host *h1*

```
mininet-wifi> xterm h1
```

Através de h1 será inicializado o cliente iperf e será criado um stream de dados entre h1 e sta3, através do seguinte comando executado em h1:

```
iperf --client 10.0.0.4 --time 60 --interval 2
```

Pelo computador h1 que está enviando o fluxo, é possível ver como resultado, as saídas geradas do fluxo realizado entre h1 e sta3. Também é possível ver a perda de dados quando a estação se desconecta do ponto de acesso, ou realiza handover, o fluxo de conectividade pode ser visto na [Figura 16](#).

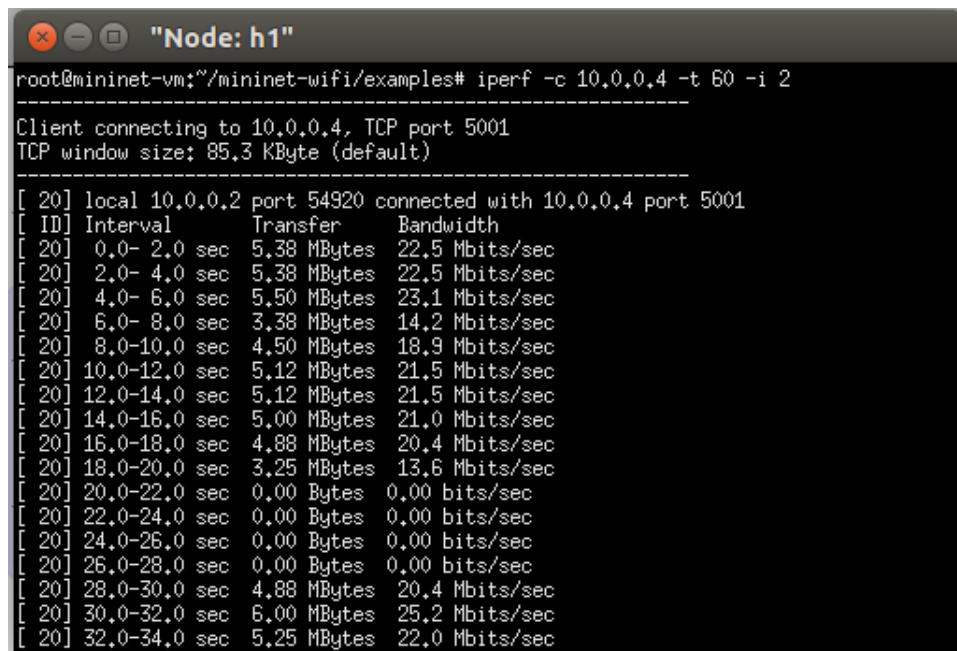


Figura 16: Fluxo de saída do comando iperf realizado por h1 em sta3

Para inicializar novamente qualquer outro script em python ou criar qualquer ambiente novo em Mininet-WiFi, é necessário limpar toda cache dos ambientes que foram utilizados anteriormente. Para que isso seja possível, é necessário realizar o comando `sudo mn -c`, que limpa todos ambientes que foram utilizados anteriormente e o computador Mininet estará pronto para ser utilizado novamente.

Quanto aos resultados, para o ambiente sem fio simulado, foi obtido um bom funcionamento do Mininet-WiFi, com pequenas considerações a serem feitas, porém que não impactam diretamente em seu funcionamento.

Considerações como alguns resultados que não condiziam com os cálculos realizados, ou quando uma estação realizava um deslocamento entre pontos de acesso, esperava-se a desconexão de uma estação em um ponto de acesso e sua reconexão em outro ponto.

Esta estação só voltava a se conectar quando ela estivesse com o seu ponto central dentro do ambiente do ponto de acesso, como ilustrado na Figura 17. Segundo a imagem, nos momentos que a imagem foi registrada, os ambientes seguiam com o status marcado, porém, outras vezes dependendo da quantidade de testes se desconectava, e mesmo assim o fluxo (respostas do comando *ping*) seguia contínuo, obtendo resposta de recebimento.

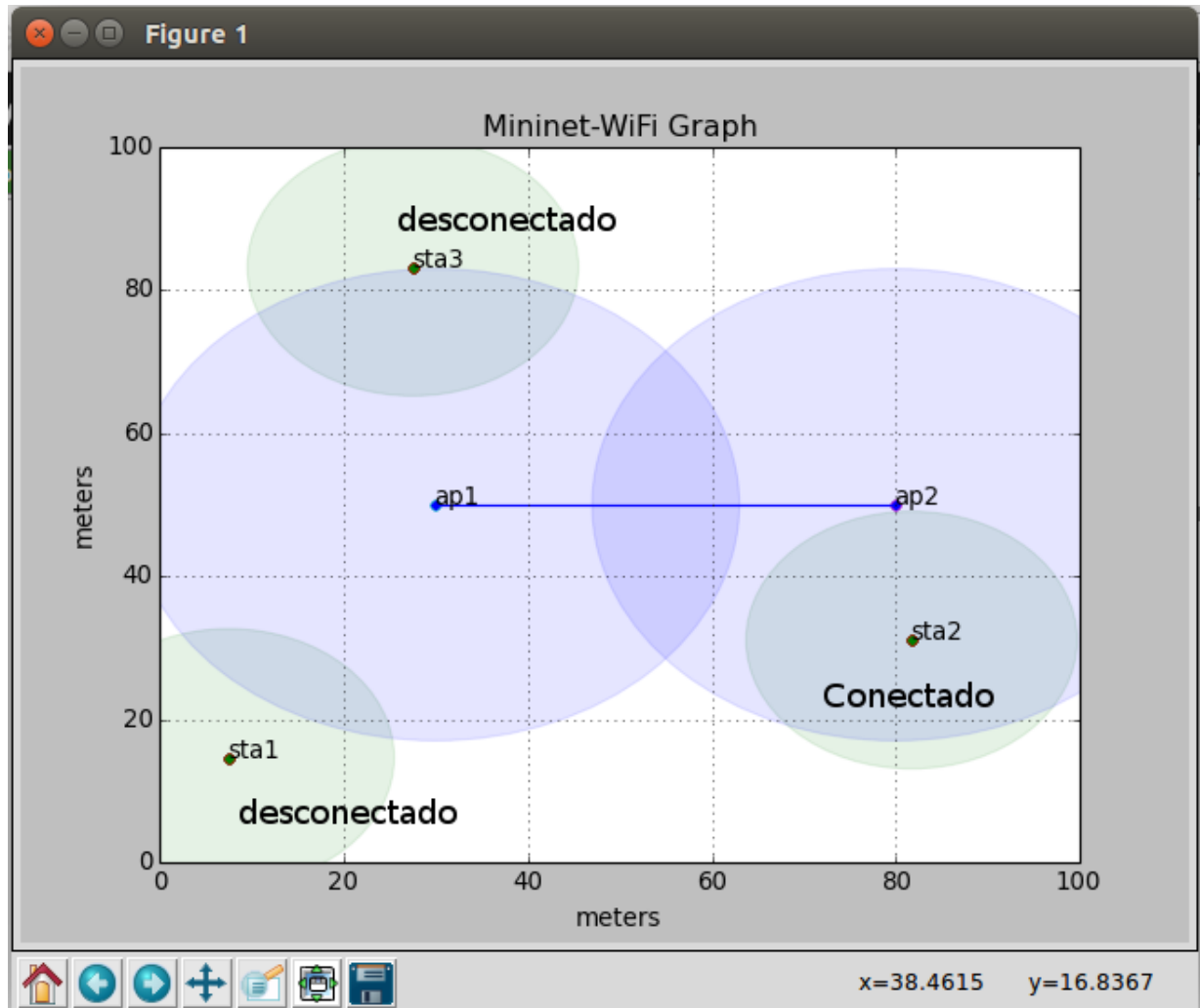


Figura 17: Status de stations conectadas e desconectadas

As configurações apresentadas não impactaram nos testes realizados, tanto que foi possível gerar escalabilidade, aumentando em scripts criados, a quantidade de pontos de acesso, o modo de deslocamento, a interação entre as estações e o ponto de acesso, bem como a implementação de host e controladora diretamente nos scripts python. A facilidade de se criar e editar esses scripts faz com que seja possível ter mais agilidade para simular diferentes tipos de ambientes e com isso testar ambientes criados de diversos modos.

Devemos também dizer que devido ao tipo de configuração feita para ambiente Mininet, é possível efetuar qualquer comando existente em Linux para qualquer computador

virtual criado em ambiente simulado, podendo assim criar um paralelo com o ambiente real e, com isso, apresentar como seria o comportamento da rede em estações de trabalho.

Testes com modelo de propagação : Foram realizados testes utilizando como base o arquivo de exemplo *wifiPropagationModel*, realizando mudanças para aumentar a complexidade do ambiente, alterando o modelo de propagação, e o tipo de mobilidade. Os testes realizados e comentários serão apresentados à seguir. O script utilizado para o modelo de propagação está disponível no Anexo B.

Foi inicializado um ambiente com duas estações móveis, se deslocando de modo aleatório através do modelo de mobilidade “GaussMarkov” (BAI; HELMY, 2004) e o modelo de propagação de “Friis”, capazes de simular um ambiente fechado, com mobilidade aleatória, se conectando entre dois pontos de acesso fixo, distanciados entre si a uma distância de 200m. O modelo foi desenvolvido em área limitada, a fim de ocorrer mais handover durante os testes. A Figura 18 ilustra o ambiente proposto para esta etapa.

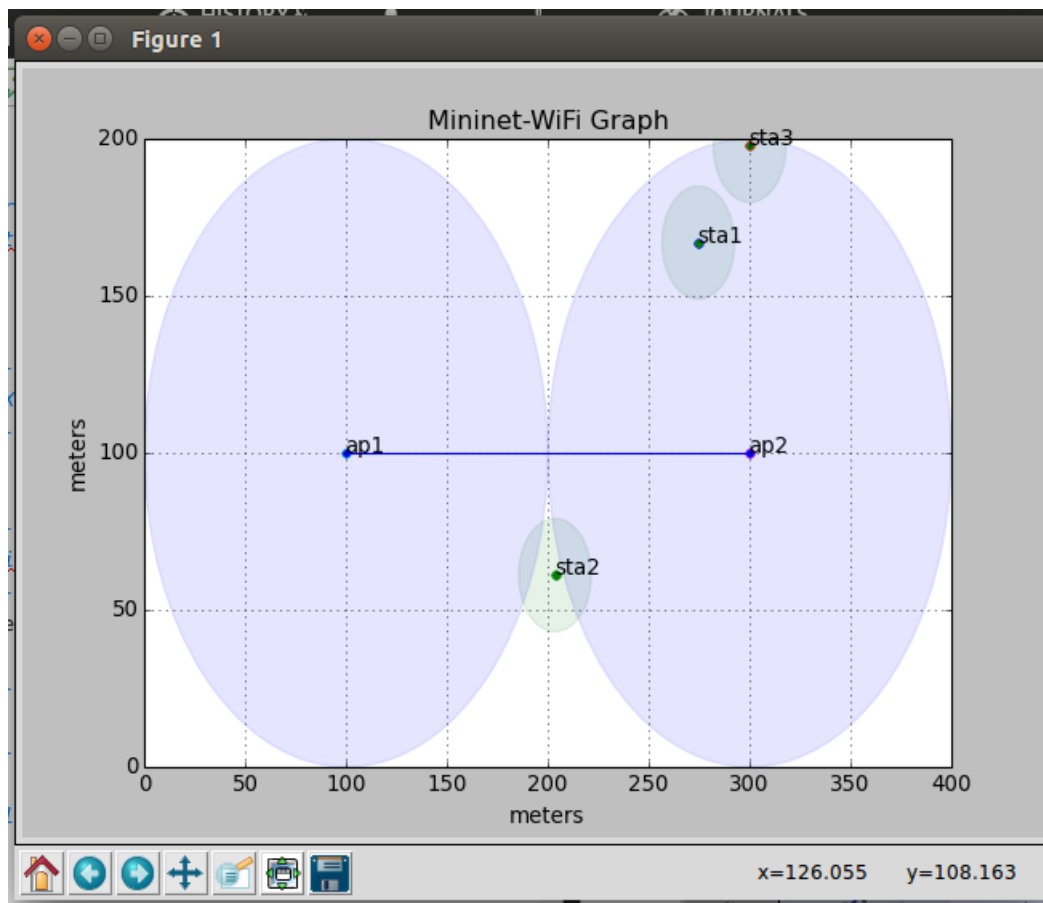


Figura 18: Ambiente criado para modelo de propagação

Com o uso do comando `iw`, através do parâmetro `scan`, é possível verificar quais redes estão acessíveis. Para realizar os testes em ambiente de propagação, foi utilizado

o Wireshark com uma placa para visualizar a saída dos fluxos de dados. A Figura 19 apresenta a perda e reconexão da estação móvel durante o processo.

```
mininet-wifi> sta1 iw dev sta1-wlan0 scan | grep ssid
SSID: new-ssid2
SSID: new-ssid1
```

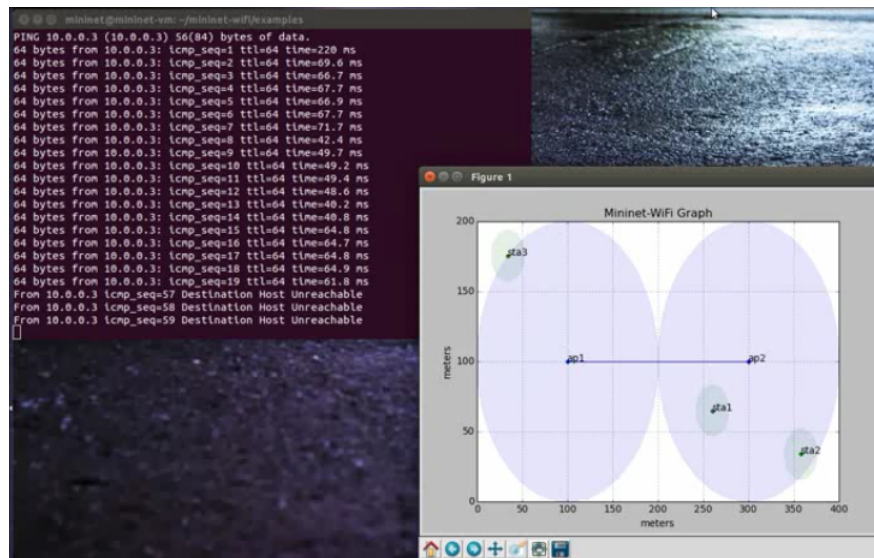


Figura 19: Handover ocorrido com o modelo Friis de propagação

Com o uso do Wireshark foi possível analisar a troca de pacotes que ocorreu durante os processos de *ping* e *iperf* na rede. Como resultados, foi visto que as estações realizam desconexões e reconexões de acordo com a sua proximidade na amplitude de alcance do ponto de acesso. Foi realizado um segundo teste alterando o modelo de propagação para um modelo propício a um ambiente externo (“*logDistancePropagationLossModel*”) e os resultados obtidos na saída foram semelhantes, não sendo possível analisar as diferenças entre os modelos de propagação. Com o uso do comando `py X.params` (onde X é uma estação ou um ponto de acesso da rede), é possível visualizar todos os parâmetros utilizados nos pontos de acesso, foram obtidos dados que estão na Tabela 1, é possível ver que o alcance (Range) em ambos os modelos de propagação são iguais, porém a intensidade do sinal recebido (RSSI) para cada estação estão com valores diferentes, isso ocorre pois o modelo de propagação obriga a configurar o script com um modelo de mobilidade, não sendo possível capturar os dados exatamente nos mesmos locais, porém, em locais próximos, é possível notar que não há grande diferença entre os valores, mostrando que os parâmetros obtidos são iguais para os dois modelos de propagação.

Tabela 1: Comparativo Friis e logLoss

	RSSI		Range	
	friis	logLoss	friss	logLoss
ap1			100	100
ap2			100	100
sta1	30.9	28.25		
sta2	27.31	27.01		

5 Conclusão

Como resultados finais, podemos dizer que os resultados sobre os conceitos discutidos a respeito de redes SDN foram visualizados nos experimentos com Mininet-WiFi, bem como o funcionamento do protocolo OpenFlow, tanto seu funcionamento teórico (comandos ping apresentando resposta), quanto prático (resultados gerados e vistos através do software Wireshark) funcionaram sem maiores problemas, podendo assim fazer um paralelo com o ambiente real e vislumbrar como seria sua aplicação em ambientes reais.

Quanto aos controladores, foram apresentados teoricamente diferentes modelos de controladores e testado o funcionamento do primeiro controlador criado para redes SDN, o controlador NOX. O controlador OpenDaylight é o mais completo dentre os controladores apresentados, sua instalação ocorreu sem maiores problemas, porém não foi possível visualizar o controlador dentro de um ambiente de Mininet-WiFi. O motivo se dá pelo fato dos controladores serem desenvolvidos baseados em equipamentos reais ou que suportem Mininet padrão.

Ao final desse trabalho, foi possível concluir que o ambiente SDN bem como o protocolo OpenFlow se tornaram ferramentas necessárias capazes de melhorar a dinâmica de ambientes de rede, bem como aumentar o uso de seus dispositivos e sua escalabilidade. Sua facilidade para criação de diferentes ambientes, faz com que a rede se comporte de maneira mais dinâmica, atendendo assim as demandas atuais de redes ativas.

Conclui-se também que o funcionamento do ambiente simulado é suficientemente útil para realizar um paralelo com ambientes reais em diferentes estruturas de rede. É possível também concluir que o ambiente de Mininet-WiFi foi de grande importância para a realização desse trabalho e, através de seu uso, foi possível realizar um paralelo com ambientes reais sem fio, bem como saber qual seu funcionamento e como seria o comportamento do protocolo OpenFlow para redes sem fio em ambientes SDN. Para projetos futuros, é possível adicionair controladores ao MiniNet Wifi e criar funções de gerenciamento capazes de funcionar normalmente com controladores, sendo assim capaz de visualizar ambientes fixos e móveis e controlar o fluxo de dados entre estes ambientes.

Referências

- BAI, F.; HELMY, A. A survey of mobility models. *Wireless Adhoc Networks. University of Southern California, USA*, v. 206, p. 147, 2004. Citado 2 vezes nas páginas 19 e 50.
- CARDOSO, R. et al. *FloodlightProject Floodlight*. 2011. Disponível em: <<http://www.projectfloodlight.org/>>. Citado na página 28.
- CASADO, M. et al. Ethane: Taking control of the enterprise. In: ACM. *ACM SIGCOMM Computer Communication Review*. [S.l.], 2007. v. 37, n. 4, p. 1–12. Citado na página 19.
- CASADO, M. et al. Sane: A protection architecture for enterprise networks. In: *Usenix Security*. [S.l.: s.n.], 2006. Citado na página 19.
- FALL, K. R.; STEVENS, W. R. *TCP/IP illustrated, volume 1: The protocols*. [S.l.]: addison-Wesley, 2011. Citado na página 36.
- FEAMSTER, N.; REXFORD, J.; ZEGURA, E. The road to sdn: an intellectual history of programmable networks. *ACM SIGCOMM Computer Communication Review*, ACM, v. 44, n. 2, p. 87–98, 2014. Citado na página 19.
- FONTES, R. dos R.; ROTHENBERG, C. E. Emulando redes wifi com o mininet-wifi. p. 1–4, 2015. Citado 3 vezes nas páginas 20, 33 e 35.
- GANTZ, J.; REINSEL, D. The digital universe in 2020: Big data, bigger digital shadows, and biggest growth in the far east. *IDC iView: IDC Analyze the future*, v. 2007, n. 2012, p. 1–16, 2012. Citado na página 19.
- GUEDES, D. et al. Redes definidas por software: uma abordagem sistêmica para o desenvolvimento de pesquisas em redes de computadores. *Minicursos do Simpósio Brasileiro de Redes de Computadores-SBRC 2012*, v. 30, n. 4, p. 160–210, 2012. Citado na página 27.
- HEWLETT-PACKARD. Creating hp software-defined networks. HP, Book 1, 2014. Disponível em: <<https://community.hpe.com/hpeb/attachments/hpeb/sdn-discussions/331/1/Creating20HP20Software-defined-Networks.pdf>>. Citado na página 24.
- KHONDOKER, R. et al. Feature-based comparison and selection of software defined networking (sdn) controllers. In: IEEE. *Computer Applications and Information Systems (WCCAIS), 2014 World Congress on*. [S.l.], 2014. p. 1–7. Citado 2 vezes nas páginas 28 e 29.
- KOPETZ, H. Internet of things. In: *Real-time systems*. [S.l.]: Springer, 2011. p. 307–323. Citado na página 19.
- KOPONEN, T. et al. Onix: A distributed control platform for large-scale production networks. In: *OSDI*. [S.l.: s.n.], 2010. v. 10, p. 1–6. Citado na página 28.
- LANTZ, B.; HELLER, B. *Mininet Documentation*. 2013. Disponível em: <<https://github.com/mininet/mininet/wiki/Documentation>>. Citado 2 vezes nas páginas 20 e 31.

- LANTZ, B.; HELLER, B. *Mininet Overview*. 2013. Disponível em: <<http://mininet.org/overview/>>. Citado 2 vezes nas páginas 30 e 31.
- LI, C.-S.; LIAO, W. Software defined networks. *IEEE Communications Magazine*, v. 51, n. 2, p. 113–113, 2013. Citado na página 23.
- LINKLETTER, B. *Mininet-WiFi: SDN emulator supports WiFi networks*. 2016. Disponível em: <<http://www.brianlinkletter.com/mininet-wifi-software-defined-network-emulator-supports-wifi-networks/>>. Citado na página 35.
- LINUX-FOUNDATION. *OpenDaylight*. 2013. Disponível em: <<https://www.opendaylight.org/>>. Citado na página 29.
- MCCAULEY, J.; SCOTT, A. W. C.; GUEDES, N. N. D. *POX The POX Controller*. 2004. Disponível em: <<http://archive.openflow.org/documents/openflow-spec-v1.1.0.pdf>>. Citado na página 28.
- MCKEOWN, N. et al. Openflow: enabling innovation in campus networks. *ACM SIGCOMM Computer Communication Review*, ACM, v. 38, n. 2, p. 69–74, 2008. Citado 5 vezes nas páginas 13, 19, 20, 25 e 26.
- ONS. *OpenFlow/Software-Defined Networking (SDN)*. 2012. Disponível em: <<http://opennetsummit.org/archives/apr12/site/why.html>>. Citado na página 24.
- PFAFF, B. et al. *OpenFlow OpenFlow Switch Specification*. 2011. Disponível em: <<http://archive.openflow.org/documents/openflow-spec-v1.1.0.pdf>>. Citado na página 25.
- RYU, S. *Framework Community, “Ryu SDN Framework,”*. 2015. Citado na página 28.
- SEZER, S. et al. Are we ready for sdn? implementation challenges for software-defined networks. *IEEE Communications Magazine*, IEEE, v. 51, n. 7, p. 36–43, 2013. Citado na página 20.
- TAROUCO, L. *Ethernet encapsulation: ARP*. 1998. Disponível em: <<http://penta2.ufrgs.br/Liane/arp.html>>. Citado na página 36.
- VELRAJAN, S. *SDN Architecture*. 2012. Disponível em: <<http://www.thetech.in/2012/12/sdn-architecture.html>>. Citado 3 vezes nas páginas 13, 23 e 24.
- YANG, M. et al. Software-defined and virtualized future mobile and wireless networks: A survey. *Mobile Networks and Applications*, Springer, v. 20, n. 1, p. 4–18, 2015. Citado na página 33.

Apêndices

APÊNDICE A – Instalação do computador virtual Mininet

A máquina virtual com mininet está disponível online através do site [Mininet](#) que contém diferentes modos de instalação e toda documentação necessária para o início do entendimento de SDN e o funcionamento do protocolo OpenFlow.

Para realizar a instalação de uma máquina virtual com Mininet, pode-se seguir as seguintes etapas:

Faça o download da imagem Mininet VM através do link:
<https://github.com/mininet/mininet/wiki/Mininet-VM-Images>

Após o download e instale um sistema de virtualização. Recomenda-se o uso de VirtualBox (gratuito, GPL) através do link <https://www.virtualbox.org/wiki/Downloads>.

Execute as Configurações da VM para efetuar login na VM e personalizá-la conforme desejado (disponível em: <http://mininet.org/vm-setup-notes/>).

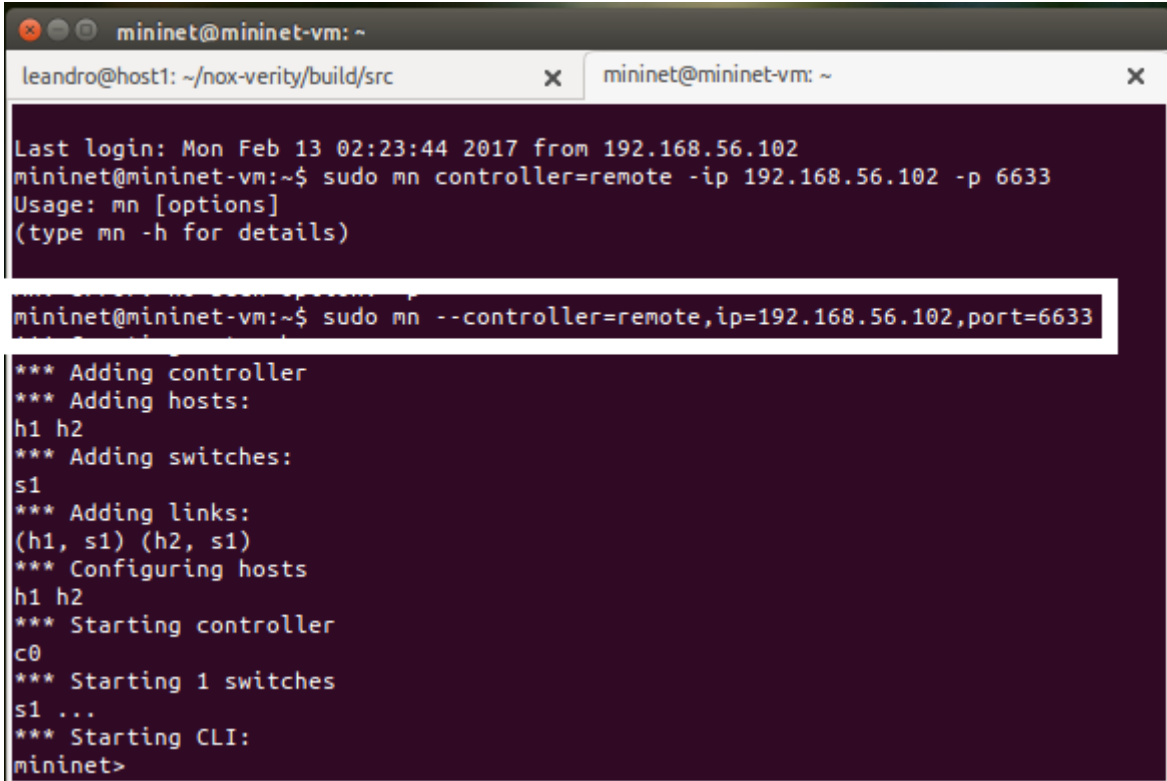
Siga o passo a passo para se familiarizar com os comandos Mininet.

Depois de concluir o passo a passo, você deve ter uma idéia clara do que é o Mininet e como usá-lo.

Caso haja interesse há disponível também um material para melhor entendimento do protocolo OpenFlow disponível em: <https://github.com/mininet/openflow-tutorial/wiki>

APÊNDICE B – Instalação do Controlador NOX

Pelo fato de existir diversos controladores, foi optado pelo uso do NOX por ser primeiro controlador criado para uso em redes SDN. Abaixo serão mostradas as etapas de sua instalação e como executar alguns comandos.



```

mininet@mininet-vm: ~
leandro@host1: ~/nox-verity/build/src x mininet@mininet-vm: ~ x

Last login: Mon Feb 13 02:23:44 2017 from 192.168.56.102
mininet@mininet-vm:~$ sudo mn controller=remote -ip 192.168.56.102 -p 6633
Usage: mn [options]
(type mn -h for details)

mininet@mininet-vm:~$ sudo mn --controller=remote,ip=192.168.56.102,port=6633

*** Adding controller
*** Adding hosts:
h1 h2
*** Adding switches:
s1
*** Adding links:
(h1, s1) (h2, s1)
*** Configuring hosts
h1 h2
*** Starting controller
c0
*** Starting 1 switches
s1 ...
*** Starting CLI:
mininet>
  
```

Figura 20: Iniciando o mininet com o uso do controlador NOX

Preparando as dependências para instalação do NOX

```

$ cd /etc/apt/sources.list.d/
$ wget http://openflowswitch.org/downloads/debian/nox.list
$ apt-get update
$ apt-get install nox-dependencies
  
```

Instalação das dependências:

```

$ apt-get install nox-dependencies
  
```

```
$ apt-get install libtbb-dev
$ apt-get install libboost-serialization-dev libboost-all-dev
$ git clone git://github.com/noxrepo/nox
```

```
$ ./boot.sh
$ cd nox
$ mkdir build
$ cd build
```

Configurar o "make && make install"

```
$ ../configure
$ make
$ make install
```

Verificar a Instalação:

```
$ cd src
$ pwd
/home/brent/nox/build/src (Path)
$ make check
$ ./nox_core -v
$ ./nox_core -h
$ ./nox_core -i ptcp:6633
./nox_core -v -i ptcp:6633 switch module
```

O uso do comando

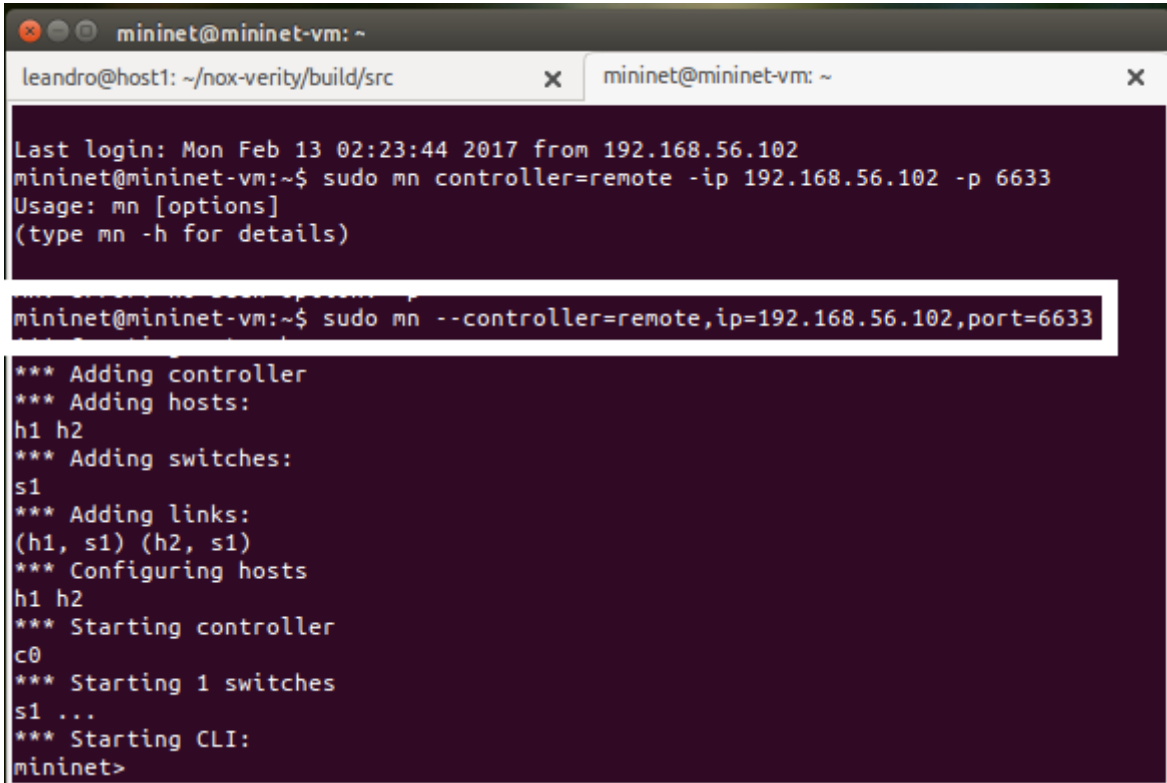
```
./nox_core -v -i ptcp:6633 switch module
```

Configura o controlador NOX para escutar através da porta 6633 e faz o OpenFlow switch se comportar como um switch normal.

Para iniciar o controlador através da mininet utilize o comando

```
sudo mn --controller=remote,ip=[Endereço IP do controlador],port=[Porta do controlador]
```

A Figura 21 apresenta a inicialização do ambiente mininet, bem como o uso de um controlador remoto para administrar o ambiente simulado.



```
mininet@mininet-vm: ~
leandro@host1: ~/nox-verity/build/src
mininet@mininet-vm: ~
Last login: Mon Feb 13 02:23:44 2017 from 192.168.56.102
mininet@mininet-vm:~$ sudo mn controller=remote -ip 192.168.56.102 -p 6633
Usage: mn [options]
(type mn -h for details)

mininet@mininet-vm:~$ sudo mn --controller=remote,ip=192.168.56.102,port=6633
*** Adding controller
*** Adding hosts:
h1 h2
*** Adding switches:
s1
*** Adding links:
(h1, s1) (h2, s1)
*** Configuring hosts
h1 h2
*** Starting controller
c0
*** Starting 1 switches
s1 ...
*** Starting CLI:
mininet>
```

Figura 21: Iniciando o controlador NOX

Anexos

ANEXO A – Script utilizado para ambiente Python: Modelo de Mobilidade

```
#!/usr/bin/python

"""
Setting the position of Nodes (only for Stations and Access
    ↪ Points) and providing mobility using mobility models.
"""

from mininet.net import Mininet
from mininet.node import Controller, OVSKernelSwitch
from mininet.link import TCLink
from mininet.cli import CLI
from mininet.log import setLogLevel

def topology():

    "Create a network."
    net = Mininet( controller=Controller, link=TCLink, switch=
        ↪ OVSKernelSwitch )

    print "*** Creating nodes "
    sta1 = net.addStation( 'sta1', mac='00:00:00:00:00:02', ip='
        ↪ 10.0.0.2/8' )
    sta2 = net.addStation( 'sta2', mac='00:00:00:00:00:03', ip='
        ↪ 10.0.0.3/8' )
    sta3 = net.addStation( 'sta3', mac='00:00:00:00:00:04', ip='
        ↪ 10.0.0.4/8' )
    h1 = net.addHost( 'h1', mac='00:00:00:00:00:05', ip='
        ↪ 10.0.0.2/8' )
    ap1 = net.addBaseStation( 'ap1', ssid= 'new-ssid1', mode= 'g
        ↪ ', channel= '1', position='30,50,0' )
    ap2 = net.addBaseStation( 'ap2', ssid= 'new-ssid2', mode= 'g
        ↪ ', channel= '2', position='80,50,0' )
```

```

c1 = net.addController( 'c1', controller=Controller )

print " *** Associating and Creating links "
net.addLink(ap1, ap2)
net.addLink(ap1, sta1)
net.addLink(ap1, sta2)
net.addLink(ap1, h1)

print " *** Starting network "
net.build()
c1.start()
ap1.start( [c1] )
ap2.start( [c1] )

"""uncomment to plot graph"""
net.plotGraph(max_x=100, max_y=100)

"""Seed"""
net.seed(20)

""" Available models: RandomWalk, TruncatedLevyWalk,
    ↪ RandomDirection, RandomWayPoint, GaussMarkov,
    ↪ ReferencePoint, TimeVariantCommunity """
net.startMobility( startTime=0, model='RandomWalk', max_x
    ↪ =100, max_y=100, min_v=0.5, max_v=0.8)

print " *** Running CLI "
CLI( net )

print " *** Stopping network "
net.stop()

if __name__ == '__main__':
    setLogLevel( 'info' )
    topology()

```

O script pode ser executado através do seguinte comando:

```
wifi:~/scripts $ sudo ./wifiMobilityModel2.py
```

ou

```
wifi:~$ sudo python wifiMobilityModel2.py
```


ANEXO B – Script Python: Modelo de Propagação

```
#!/usr/bin/python

"""
    Example: Propagation Models
"""

from mininet.net import Mininet
from mininet.node import Controller, OVSKernelAP
from mininet.link import TCLink
from mininet.cli import CLI
from mininet.log import setLogLevel

def topology():

    "Create a network."
    net = Mininet( controller=Controller, link=TCLink,
        ↪ accessPoint=OVSKernelAP )

    print "*** Creating nodes "
    sta1 = net.addStation( 'sta1', mac='00:00:00:00:00:01', ip='
        ↪ 10.0.0.1/8 ' )
    sta2 = net.addStation( 'sta2', mac='00:00:00:00:00:02', ip='
        ↪ 10.0.0.2/8 ' )
    sta3 = net.addStation( 'sta3', mac='00:00:00:00:00:03', ip='
        ↪ 10.0.0.3/8 ' )
    h1 = net.addHost( 'h1', mac='00:00:00:00:00:08', ip='
        ↪ 10.0.0.8/8 ' )
    ap1 = net.addAccessPoint( 'ap1', ssid= 'new-ssid1',
        ↪ equipmentModel='DI524', mode='b', channel='1',
        ↪ position='100,100,0' )
    ap2 = net.addAccessPoint( 'ap2', ssid= 'new-ssid2',
        ↪ equipmentModel='DI524', mode='b', channel='2',
        ↪ position='300,100,0' )
```

```

c1 = net.addController( 'c1', controller=Controller )

print "***_Configuring_wifi_nodes"
net.configureWifiNodes()

print "***_Associating_and_Creating_links"
net.addLink(ap1, ap2)
net.addLink(ap1, sta1)
net.addLink(ap1, sta2)
net.addLink(sta3, ap2, link='wired')
print "***_Starting_network"
net.build()
c1.start()
ap1.start( [c1] )
ap2.start( [c1] )

"""uncomment to plot graph"""
net.plotGraph(max_x=400, max_y=200)

"""Seed"""
net.seed(1)

***_Available_propagation_models:_friisPropagationLossModel
↪ ,_twoRayGroundPropagationLossModel,_
↪ logDistancePropagationLossModel_***
net.propagationModel( 'friisPropagationLossModel', sL=2)

***_Available_mobility_models:_RandomWalk,_
↪ TruncatedLevyWalk,_RandomDirection,_RandomWayPoint,_
↪ GaussMarkov_***
net.startMobility( startTime=0, model='RandomWayPoint', max_x
↪ =400, max_y=200, min_v=10.5, max_v=20.5)

print "***_Running_CLI"
CLI( net )

print "***_Stopping_network"
net.stop()

```

```
if __name__ == '__main__':  
    setLogLevel( 'info' )  
    topology()
```


Índice

ad-hoc, [34](#)
API, [28](#), [35](#), [45](#)
C ++, [27](#)
Cisco, [25](#)
CLI, [23](#), [34–36](#)
controlador, [25–29](#), [35](#), [40](#), [45](#), [49](#), [53](#)

FIB, [15](#), [23](#)
Floodlight, [28](#), [29](#)

handover, [45](#), [48](#), [50](#)

Internet das Coisas, [19](#)

Java, [28](#), [29](#)

Linux, [29](#), [30](#), [34](#), [49](#)

Mininet, [20](#), [28](#), [30](#), [31](#), [33](#), [34](#), [37](#), [40](#), [48](#),
[49](#), [53](#)
Mininet-WiFi, [20](#), [21](#), [33–36](#), [38–40](#), [43](#),
[45–48](#), [53](#)

NIB, [27](#), [28](#)
NOX, [27](#), [28](#), [53](#), [61](#), [63](#)

ONIX, [27](#)
OpenDaylight, [29](#), [43](#), [53](#)
OpenFlow, [20](#), [21](#), [24–28](#), [30](#), [33](#), [35](#), [40](#),
[42](#), [45](#), [53](#)

plano de controle, [20](#), [23](#), [24](#), [33](#)
plano de dados, [20](#), [23](#), [24](#), [33](#)
POX, [27](#), [28](#)
Python, [28](#), [35](#), [45](#), [47–49](#), [67](#), [71](#)

QoS, [23](#)

redes definidas por software, [20](#), [21](#), [23](#), [33](#)
roteador, [25](#), [28](#), [30](#)

Ryu, [28](#)

SDN, [20](#), [23–27](#), [29](#), [33](#), [45](#), [53](#), [61](#)
switch, [25](#), [27](#), [28](#), [30](#)

Wireshark, [36](#), [40](#), [42](#), [51](#), [53](#)