

Ramon Fontes  
Christian Rothenberg

# Emulando Redes sem Fio com Mininet-WiFi



Ramon dos Reis Fontes  
Christian Rodolfo Esteve Rothenberg

# Emulando Redes sem Fio com Mininet-WiFi

1<sup>a</sup> edição

Campinas  
Christian Rodolfo Esteve Rothenberg  
2019

## **Créditos**

### **Autores**

Ramon dos Reis Fontes  
Christian Rodolfo Esteve Rothenberg

### **Revisores**

Michel Daoud Yacoub  
Wilson Borba da Rocha Neto  
Hedertone Vieira Almeida

### **Template**

Mathias Legrand ([legrand.mathias@gmail.com](mailto:legrand.mathias@gmail.com))  
modificado por Vel ([vel@latextemplates.com](mailto:vel@latextemplates.com))  
sob licença CC BY-NC-SA 3.0:  
<http://creativecommons.org/licenses/by-nc-sa/3.0/>

## Sobre os Autores

**Ramon dos Reis Fontes** passou a ser evangelista de código-fonte aberto desde 2013. Ele é doutor em Engenharia Elétrica na área de Engenharia da Computação pela Universidade Estadual de Campinas (UNICAMP); Mestre em Sistemas e Computação pela Universidade Salvador (UNIFACS); e Bacharel em Sistemas de Informação pela Faculdade de Tecnologia e Ciências (FTC). Seus interesses de pesquisa incluem *Software-Defined Networking* (SDN), redes móveis, sistemas distribuídos, *Cloud* e *Fog computing*, *Network Functions Virtualizations* (NFV) e segurança da informação. Ramon publicou vários artigos para conferências e revistas, e tem contribuído continuamente com o desenvolvimento de software livre e de código aberto através de sua conta no Github (<https://github.com/ramonfontes>). Uma variedade de códigos e instruções de como reproduzir pesquisas em que atuou podem ser encontrados em <https://github.com/ramonfontes/reproducible-research/>.

*Gostaria de agradecer aos leitores pelo interesse no livro. Por favor, não hesite em nos contactar se você precisar de alguma ajuda em qualquer assunto abordado neste livro. Gostaria de agradecer aos meus amigos, colegas e professores que me ajudaram de várias maneiras. Gostaria de agradecer também ao Prof. Dr. Christian Rothenberg, orientador, conselheiro e co-autor deste livro, por partilhar a sua sabedoria e encorajamento durante toda a minha trajetória na Faculdade de Engenharia Elétrica e Computação, na Unicamp.*

*Os meus agradecimentos especiais vão para a minha amada esposa Suian. Este livro não seria uma realidade sem seu apoio contínuo desde o momento em que comecei o doutorado. Também gostaria de agradecer aos meus pais, Hélio e Conceição, pelos incentivos e apoio.*

*À minha filha, Pietra, dedico este livro.*

**Christian Rodolfo Esteve Rothenberg** é Professor Doutor no Departamento de Engenharia de Computação e Automação Industrial (DCA) da Facul-

---

dade de Engenharia Elétrica e Computação (FEEC) pela Universidade Estadual de Campinas (UNICAMP) desde 2013. Possui graduação em *Ingiero Superior de Telecomunicación* pela *Universidad Politécnica de Madrid* (2004), mestrado em *Dipl. Ing. Elektrotechnik* pela *Darmstadt University of Technology* (2006) e doutorado em Engenharia Elétrica e Computação pela UNICAMP (2010). De 2010 a 2013 trabalhou como pesquisador sênior da Fundação CPqD, atuando em projetos de P&D na área de plataformas IP. Atualmente é Pesquisador principal do grupo INTRIG (*Information & Networking Technologies Research & Innovation Group* – <https://intrig.dca.unicamp.br/>), bolsista de Produtividade em Pesquisa do CNPq Nível 2 (2017-2020), bolsista de Desenvolvimento Tecnológico e Extensão Inovadora do CNPq Nível 2 (2014-2016). Seus interesses de pesquisa incluem arquiteturas de redes de computadores, virtualização, computação em nuvem, SDN, NFV, entre outros. Possui mais de 120 publicações em revistas e conferências, acumulando mais de 5000 citações (h-index: 24, i10-index: 36 – <https://scholar.google.com.br/citations?user=8PxuHPkAAAAJ&hl=en>) e 2 patentes internacionais.

*A concretização de um livro é uma ótima oportunidade para refletir e expressar gratidão. Começando pelos nossos ancestrais, no meu caso, meus pais José Luis e Ana. Na vida acadêmica, agradeço todos os professores que me influenciaram. Destaco o Professor Maurício, orientador, pai acadêmico, e amigo, figura chave desde que pousei no chão deste amado Brasil. Sou grato a todas as oportunidades que recebi desse país, suas pessoas e instituições. O CPqD, a FEEC, os organismos financiadores, a Ericsson, entre outros, diretamente responsáveis pelo meu caminho. Aos alunos, da graduação até a pós, aos colegas de profissão e aos amigos do dia a dia. Ao nosso grupo INTRIG, e lógico, ao Prof. Dr. Ramon Fontes, primeiro doutor made in INTRIG, exemplo de orientado e de pessoa, pai de família e desse livro que tenho certeza vai contribuir para a formação de mais profissionais. Por último, o vetor mais importante na vida, a família, minha esposa Marcela e os meus filhos Gabriel e Marina, minhas fontes de energia e felicidade. Obrigado!*

## **Sobre os Revisores**

**Michel Daoud Yacoub** é Professor Titular da Faculdade de Engenharia Elétrica e de Computação da Unicamp, Pesquisador 1A CNPq, autor de patentes, livros e artigos na área de Comunicações Sem Fio

**Wilson Borba da Rocha Neto** é Bacharel em Engenharia Elétrica pela Universidade Federal do Piauí (UFPI) e estudante de Mestrado em Engenharia Elétrica na Universidade Estadual de Campinas (UNICAMP). Apaixonado desde a adolescência por microeletrônica e inteligência computacional, teve a oportunidade de conhecer Ramon e Christian quando ingressou no programa de mestrado, sendo imediatamente doutrinado pela religião *open source*. Atualmente desenvolve projetos relacionados a Sistemas de Transporte Inteligentes e Redes Definidas por Software.

**Hedertone Vieira Almeida** é Professor no Instituto Federal Baiano (IFBAIANO) e entusiasta de sistemas Linux desde a época da faculdade. Conheceu Ramon ainda na faculdade, em 2005, quando iniciou o Curso de Bacharelado em Sistemas de Informação. Nessa época já atuava com Servidores Linux, na Universidade Estadual do Sudoeste da Bahia. No ano de 2010, já graduado, iniciou carreira de professor no Serviço Nacional de Aprendizagem Industrial (SENAI), lecionando disciplinas relacionadas à Redes de Computadores, Servidores Linux e Equipamentos Cisco. Ingressou na Rede Federal de Ensino em 2015.

## Agradecimentos

O sucesso deste projeto e a escrita deste livro só foram possíveis graças ao apoio, colaboração e confiança de muitas pessoas e instituições que ajudaram a torná-los realidade. Por isso, fazemos questão de registrar aqui nossos agradecimentos.

Aos revisores Michel, Wilson e Hedertone, ficamos gratos pelas revisões e colaborações que ajudaram a aprimorar a qualidade do conteúdo teórico e tutoriais práticos abordados ao longo deste livro.

Agradecemos à Katia Obraczka, professora da Universidade da Califórnia, em Santa Cruz, Califórnia, pelas valorosas opiniões e sugestões sobre as etapas de desenvolvimento do Mininet-WiFi. Agradecemos também ao *Institut National de Recherche en Informatique et en Automatique* (INRIA), em especial aos doutores Thierry Turletti e Walid Dabbous, por nos receber no INRIA durante 6 (seis) meses e por contribuir em diversos aspectos relacionados ao desenvolvimento do Mininet-WiFi. Pelo suporte financeiro, agradecemos à Fundação de Amparo à Pesquisa do Estado de São Paulo (FAPESP), processo 2014/18482-4 e ao Conselho Nacional de Pesquisas (CNPq), processo 310930/2016-2. O grupo de pesquisa INTRIG agradece à Ericsson os financiamentos à pesquisa recebidos, sem os quais o grupo não teria atingido seus resultados, muitos deles alavancados por e contribuindo para o Mininet-WiFi.

Agradecemos também a alguns usuários, pesquisadores e/ou desenvolvedores que contribuíram para que este livro se tornasse uma realidade. São eles: o professor doutor Chih-Heng Ke, da National Quemoy University, Taiwan, por suas dicas e experiência compartilhada acerca do funcionamento das redes sem fio; Brian Linkletter, por escrever um tutorial, divulgar e ajudar a disseminar o Mininet-WiFi; Patrick Große, pelo desenvolvimento de uma extensão do *Wmediumd* para o Mininet-WiFi; e, claro, à Comunidade do Mininet-WiFi, por todas as discussões que resultam no desenvolvimento de um emulador cada vez mais estável e completo em termos de recursos suportados.

*Emulando Redes sem Fio com Mininet-WiFi*  
ISBN: 978-65-900571-0-5 (Impresso)

#### Aviso Legal

##### **Todos os direitos reservados.**

Nenhuma parte desta obra pode ser copiada ou reproduzida sob quaisquer forma ou meios, seja eletronicamente, de fotocópia, gravação, etc., sem a expressa e devida autorização dos autores e detentores dos direitos autorais.

Toda e qualquer infração dos direitos acima expostos, caracterizam crime, e está sujeito às sanções penais cabíveis, conforme as leis Nº 9.610/12.853 vigentes no país.



## Sumário

<b>Lista de Figuras</b>	iii
<b>Lista de Tabelas</b>	v
<b>Lista de Acrônimos</b>	viii

I

## Introdução

<b>1 Fundamentação teórica .....</b>	3
<b>1.1 Introdução às comunicações sem fio</b>	3
<b>1.2 WiFi: Redes locais sem fio baseadas no IEEE 802.11</b>	6
<b>1.3 Redes sem fio definidas por software</b>	11
<b>1.4 Mininet-WiFi</b>	14
1.4.1 Arquitetura .....	16
1.4.2 Funcionamento .....	17

**II****Nível: iniciante**

<b>2</b>	<b>Iniciante .....</b>	<b>23</b>
2.1	<b>Primeiros passos com Mininet-WiFi</b>	<b>25</b>
2.2	<b>Customizando topologias</b>	<b>32</b>
2.3	<b>Acessando informações dos nós</b>	<b>34</b>
2.4	<b>OVSAP versus UserAP</b>	<b>36</b>
2.5	<b>Utilizando interfaces gráficas</b>	<b>40</b>
2.5.1	Visual Network Descriptor .....	40
2.5.2	MiniEdit .....	41
2.5.3	Visualizando gráficos 2D e 3D .....	42
2.6	<b>Emulação do meio sem fio</b>	<b>44</b>
2.6.1	TC ( <i>Traffic Control</i> ) .....	44
2.6.2	Wmediumd .....	46
2.6.3	TC versus Wmediumd na prática .....	46
2.7	<b>Modelos de propagação</b>	<b>49</b>
2.7.1	Provendo mais realismo .....	51
2.8	<b>Relação distância versus sinal recebido</b>	<b>53</b>
2.9	<b>Modificando o <i>bitrate</i></b>	<b>54</b>
2.10	<b>Relação distância versus largura de banda</b>	<b>57</b>
2.11	<b>Modelos de mobilidade</b>	<b>59</b>

**III****Nível: intermediário**

<b>3</b>	<b>Intermediário .....</b>	<b>65</b>
3.1	<b>Manipulando interfaces</b>	<b>66</b>
3.1.1	Definindo múltiplas interfaces .....	66
3.1.2	Interfaces <i>binding</i> .....	68
3.1.3	Interfaces <i>bonding</i> .....	69

<b>3.2</b>	<b>Análise de tráfego</b>	<b>75</b>
3.2.1	Capturando pacotes .....	75
3.2.2	Capturando beacons .....	78
<b>3.3</b>	<b>Análise do espectro</b>	<b>80</b>
<b>3.4</b>	<b>Métodos de escaneamento</b>	<b>83</b>
3.4.1	Escaneamento ativo .....	83
3.4.2	Escaneamento passivo .....	83
<b>3.5</b>	<b>Wireless mesh e adhoc</b>	<b>87</b>
<b>3.6</b>	<b>Protocolo OpenFlow</b>	<b>91</b>
3.6.1	Capturando mensagens OpenFlow .....	92
3.6.2	Criando fluxos .....	95
3.6.3	OpenFlow no contexto sem fio .....	98
3.6.4	Iniciando um controlador remoto .....	101
3.6.5	OpenFlow e handover .....	103
<b>3.7</b>	<b>Aplicações</b>	<b>107</b>
3.7.1	Servidor WEB .....	107
3.7.2	Servidor DHCP .....	109
3.7.3	Lidando com loops .....	113
3.7.4	Virtual LAN (VLAN) .....	116
3.7.5	Roteamento .....	120
3.7.6	Firewall .....	124
3.7.7	Qualidade de Serviço (QoS) .....	128
3.7.8	MultiPath TCP (MP-TCP) .....	131

IV

## Nível: avançado

<b>4</b>	<b>Avançado</b> .....	<b>137</b>
<b>4.1</b>	<b>Manipulando módulos do kernel</b>	<b>137</b>
<b>4.2</b>	<b>Monitoramento de tráfego com o sFlow-RT</b>	<b>141</b>
<b>4.3</b>	<b>Reprodução de comportamentos</b>	<b>144</b>
4.3.1	Atributos de rede .....	144
4.3.2	Mobilidade .....	145

<b>4.4 Aplicações</b>	<b>146</b>
4.4.1 Contêineres .....	146
4.4.2 Interação entre ambiente virtual e real .....	148
4.4.3 Decodificando pacotes .....	158
4.4.4 Controle na associação .....	165
4.4.5 Encaminhamento por SSID .....	167
4.4.6 Segurança .....	169
4.4.7 6LoWPAN / IoT .....	186
4.4.8 Redes veiculares .....	191

## FAQ

<b>FAQ (Perguntas frequentes)</b> .....	<b>199</b>
---	------------

## Referências

<b>Referências</b> .....	<b>205</b>
--------------------------	------------

---

## Prefácio

Estamos testemunhando uma formidável revolução na área de Redes de Computadores. Avanços em comunicações sem fio, com a iminente implantação de redes 5G em todo o mundo, a possibilidade de virtualizar infraestruturas de rede (ex: *Network Slicing* e *Network Function Virtualization*) e de realizar a programação de seu comportamento (*Software-Defined Networking*) constituem exemplos concretos dessa nova era. Tais avanços habilitam o projeto, o desenvolvimento e a implantação de mecanismos inovadores visando, por exemplo, a maior resiliência, desempenho, eficiência energética e segurança do ecossistema de redes e serviços.

Um elemento-chave à exploração dessas novas oportunidades é a existência de ferramentas que permitam, em um estágio inicial, a prototipação e o teste dessas novas ideias, sem as amarras e as complexidades associadas ao emprego de infraestruturas reais. É exatamente esse o espaço ocupado pelo Mininet-WiFi, ambiente que permite, a partir de um computador pessoal, criar, explorar e experimentar redes sem fio definidas por software.

Como professor de disciplinas da área de Redes na Graduação e na Pós-Graduação do INF-UFRGS, costumo enunciar um trabalho extraclasse solicitando aos alunos o desenvolvimento de novos mecanismos sobre infraestruturas de rede definidas por software. Foi uma grata surpresa quando, em 2016, conheci o Mininet-WiFi. O ambiente expandiu, e muito, as possibilidades de trabalho, viabilizando o projeto de toda uma nova gama de propostas, tais como algoritmos de roteamento para escoamento eficiente de vídeo em redes móveis, estratégias de *handoff* cientes de flutuações momentâneas de tráfego e mecanismos de balanceamento de carga visando à sustentabilidade energética de dispositivos portáteis. Essa natureza de trabalho era muito difícil, se não impossível, de se concretizar com o ambiente Mininet clássico, em função de estar fora de seu escopo a disponibilização de primitivas para lidar especificamente com comunicação sem fio.

Este livro encerra de maneira primorosa um ciclo de desenvolvimento, inovação e transferência de novos conhecimentos para a Sociedade. Ao documentar e explicar muito didaticamente como utilizar o Mininet-WiFi, por meio de

---

roteiros, códigos-exemplo, ilustrações e outros recursos, o livro impulsionará muitas novas experiências como as recém mencionadas. Não resta dúvida de que perdurará como uma referência na academia, na indústria e nos setores governamentais brasileiros. Parabenizo os autores pela louvável iniciativa em prol do avanço dessa área tão instigante e fundamental. Não há tempo a perder: é chegada a hora de arregaçarmos as mangas e iniciarmos nossa imersão no Mininet-WiFi. Boa leitura e bom proveito!

Prof. Luciano Paschoal Gaspary  
Instituto de Informática - UFRGS

---

## **Organização dos Capítulos**

A organização deste livro se dá da seguinte forma:

**Capítulo I**, que introduz fundamentos teóricos das redes sem fio, redes sem fio definidas por software e também do Mininet-WiFi. Este capítulo revisa em alto nível conceitos relevantes para os objetivos de aprendizado deste livro. Para um maior aprofundamento nos diferentes tópicos o leitor será apontado para referencias bibliográficas relevantes na área;

**Capítulo II**, identificado como nível iniciante, que é dedicado exclusivamente a detalhes de funcionamento do Mininet-WiFi, onde os principais aspectos funcionais são apresentados. Se você já conhece o Mininet-WiFi, poderá concentrar-se apenas nos capítulos III e IV. Não é necessário conhecer previamente o Mininet para poder trabalhar com o Mininet-WiFi, mas caso você já o conheça, certamente terá uma ambientação mais suave em relação a forma de funcionamento do Mininet-WiFi. Os tutoriais neste capítulo podem ser usados no nível de graduação, como atividades complementares em disciplinas teóricas (ex: EA074 na FEEC/UNICAMP) assim como em roteiros práticos de disciplinas de laboratório (ex: EA080 na FEEC/UNICAMP);

**Capítulo III**, que é identificado como nível intermediário, abordará tutoriais relacionados às redes sem fio, redes sem fio definidas por software e também alguns conceitos relacionados à redes de computadores. Este capítulo também inclui a utilização de algumas aplicações de rede, como *tcpdump*, *Wireshark*, etc. Além de atender objetivos pedagógicos de disciplinas mais avançadas na área de redes de computadores como por exemplo laboratórios, os tutoriais neste capítulo são também adequados para disciplinas no nível de pós-graduação (ex: IA369, IA376 na FEEC/UNICAMP) e cursos de especialização (ex: INF-556 no IC/UNICAMP), uma vez que permitem pesquisas experimentais em cenários mais complexos, incluindo programabilidade de redes definidas por software com o protocolo OpenFlow;

Por fim, no **Capítulo IV**, identificado como nível avançado, será possível encontrar tutoriais relacionados à manipulação do núcleo do sistema operacional, contêineres, segurança, IoT, redes veiculares etc., com valiosas informações

---

relacionadas à adaptação do protocolo OpenFlow para as redes sem fio. Este capítulo é rotulado como avançado, pois requer conhecimentos mais aprofundados e o uso de aplicações de terceiros. Portanto, os tutoriais neste capítulo são mais adequados em cursos de especialização e pós-graduação, não só em disciplinas mas também como capacitação técnica de alunos de mestrado e doutorado para o desenvolvimento de pesquisas experimentais visando avanços no estado da arte. Mas nada impede o leitor curioso de reproduzir esses tutoriais, os quais contam com a mesma explicação passo a passo e o suporte fornecido pelos códigos dos capítulos anteriores.

---

## Convenções utilizadas neste livro

Para facilitar a leitura e entendimento deste livro, as seguintes convenções foram adotadas:

*ítально*: indica palavras em língua estrangeira ou nome de aplicação/ferramenta.

*<arquivo>*: indica a presença de arquivo ou script.

Os símbolos abaixo representam:



Informação complementar ao conteúdo exposto anteriormente.



Alerta ou observação relevante.



Pergunta referente ao assunto que está sendo explorado.



Referências bibliográficas e outras fontes complementares.



Vídeos demonstrativos.



**Pré-requisito(s)** Para cada experimento existente haverá uma indicação dos pré-requisitos necessários para a sua realização. Para facilitar, indicamos o pré-requisito de nome “apenas script(s)”, aqueles pré-requisitos que demandam apenas da utilização de scripts que já estão previamente disponíveis para o uso. Como todos os scripts foram codificados pensando no Mininet-WiFi, ignoramos o Mininet-WiFi como pré-requisito a título de informação na aba pré-requisitos. O mesmo vale para todos os pacotes que são instalados durante o processo de instalação do Mininet-WiFi, como o OVS, o protocolo OpenFlow, etc.

---

## **Outras convenções**

Como existe uma tendência na substituição de ferramentas provenientes do pacote *net-tools* pelas ferramentas do pacote *iproute2* em sistemas operacionais Linux, ferramentas como *iw* e *ip* são preferidas para a execução dos tutoriais. Por outro lado, aquelas provenientes do pacote *net-tools* também podem ser utilizadas.

Finalmente, devido a questões relacionadas à atualizações de código, todos os scripts estão disponíveis em um repositório no Github (<https://github.com/ramonfontes/mn-wifi-book-pt>).

---

## **Precauções**

É aconselhável que todos os tutoriais disponíveis neste livro sejam reproduzidos a partir da versão mais recente do Mininet-WiFi disponível no Github. Caso você, leitor, encontrar alguma inconsistência nos tutoriais, poderá contar com os autores deste livro em qualquer momento para eventuais esclarecimentos.

Embora este livro traga experiências práticas a todo instante, é recomendável que cada comando ou configuração seja avaliado previamente para que todo o processo seja compreendido.

**Não avance etapas sem compreender claramente o que está sendo feito!**

## **Devo utilizar o Linux. Mas por quê?**

Porque a base do Mininet-WiFi, o Mininet, foi desenvolvido para sistemas Linux. O desenvolvimento do Mininet-WiFi manteve a mesma estrutura de funcionamento do Mininet. O sistema operacional Ubuntu deve ser o preferido, especialmente as versões LTS (*Long Term Support*), uma vez que esta é a distribuição mais estável deste sistema operacional.

## **Por que código-fonte aberto?**

Iremos responder a essa pergunta com uma simples resposta: porque na maioria das vezes temos (eu, você e qualquer um) a liberdade de fazer com a ferramenta/aplicação o que bem entendermos. Seja porque precisamos realizar um trabalho ou pesquisa acadêmica, porque queremos conhecer mais sobre o Mininet-WiFi, ou mesmo porque precisamos modificá-lo de acordo com as nossas necessidades. Porém, essa parece ser uma resposta pronta, que muitas vezes encontramos por aí quando nos perguntamos sobre as vantagens de se abrir o código-fonte de um determinado programa.

Argumentando de forma cronológica, poderíamos afirmar que sem o Mininet-WiFi, eu, Ramon, não teria obtido título de doutor; muitos pesquisadores não teriam realizado suas pesquisas; o Mininet, emulador ao qual o Mininet-WiFi foi estendido muito provavelmente também não existiria, e por aí vai. O que queremos dizer é que sem a filosofia de código-fonte aberto, não teríamos acesso ao Mininet e não teríamos desenvolvido o Mininet-WiFi. Assim como

---

o Mininet não teria sido desenvolvido sem que sua espinha dorsal também não estivesse disponível para uso. Ah! Muito provavelmente você não estaria lendo esse livro agora e muitos menos estaria interessado nos assuntos que abordamos nele.

Você consegue imaginar a quantidade de pesquisas que seriam prejudicadas levando em consideração apenas o ramo de pesquisa que abordamos neste livro? Talvez terá uma melhor noção ao concluir todos os tutoriais propostos ao longo deste livro.

Existe toda uma cadeia que seria seriamente impactada caso os códigos não estivessem livres para uso. Somente o principal trabalho [14] acerca do Mininet já teve cerca de 1500 citações diretas, fora aquelas indiretas, utilizadas em blogs e mesmo as que são veiculadas em meios de comunicação.

### **Experiências: Impacto, Reproducibilidade e Qualidade**

Se você está em dúvidas se vale a pena desenvolver pesquisas e expor seu código ao público, mesmo ainda estando em fase inicial de desenvolvimento, aqui vão alguns relatos de experiências que obtivemos ao longo do desenvolvimento do Mininet-WiFi.

**Impacto.** Tornar códigos, dados, documentação e vídeos de demonstração, públicos, certamente contribuiu e ainda contribui bastante para o aumento da visibilidade do Mininet-WiFi. Acerca da comunidade de usuários, embora não tenha sido feita uma avaliação sistemática e qualitativa da comunidade do Mininet-WiFi, são várias as contribuições que os usuários têm feito. Seja por meio de discussões ou mesmo por sugestões de melhorias em código e novas implementações.

**Reproducibilidade.** Tornar dados públicos e reproduutíveis nem sempre é uma tarefa simples. Aliás, escrever um livro ao qual usuários possam reproduzir todos os tutoriais sem eventuais contratemplos não é fácil. Muito provavelmente haverá um ou outro tutorial que não fluirá conforme esperado. Isso ocorre devido a diversos fatores, como diferenças nas versões de ferramentas, questões que podem estar relacionadas ao sistema operacional,

---

hardware, etc. Ao longo do desenvolvimento do Mininet-WiFi, tivemos muitas dificuldades em reproduzir experimentos de outros pesquisadores, pois muitas vezes não haviam informações suficientes para tal. Por isso, optamos por, além de tornar os experimentos desenvolvidos por nós públicos, também descrever como eles podem ser reproduzidos. Tornar pesquisas reproduutíveis, em geral, agrupa credibilidade ao trabalho e aos resultados obtidos.

**Qualidade.** Em um sentido mais amplo da qualidade da pesquisa, todas as experiências obtidas ao longo do desenvolvimento do Mininet-WiFi nos permitiu aprender e aprimorar nossas pesquisas. Com relação ao tema reproducibilidade, embora seja necessário trabalho extra, um trabalho reproduutível traz uma série de outras vantagens, tais como: (i) o código aberto aumentam as chances de reproduzibilidade direta e indireta e, consequentemente, (ii) maior impacto, já que as chances dos pesquisadores utilizarem as soluções propostas aumentam; (iii) melhora os hábitos, onde uma atenção especial é dada à qualidade do código; (iv) prima pelos fluxos de trabalho científicos, uma vez que estamos mais cuidadosamente preocupados em fornecer resultados confiáveis, visando que qualquer pessoa seja capaz de produzir resultados similares ao obtidos por nós.

Assim, se você tiver oportunidade e tornar seu código público não for um problema de licença, patente ou direitos autorais, experimente fazê-lo!



## **Lista de Figuras**

1.1	Relação do nível de sinal (RSSI) . . . . .	4
1.2	Fenômenos de desvanecimento. . . . .	5
1.3	Padrões IEEE 802.11. . . . .	7
1.4	Canais no IEEE 802.11b . . . . .	7
1.5	Modo infraestruturado. . . . .	9
1.6	Modo <i>adhoc</i> . . . . .	9
1.7	O quadro 802.11. . . . .	10
1.8	Arquitetura de alto nível e genérica para SDWN . . . . .	12
1.9	Plataformas experimentais para redes sem fio . . . . .	15
1.10	Arquitetura do Mininet-WiFi . . . . .	16
1.11	Principais componentes do Mininet-WiFi . . . . .	17
2.1	Topologia simples. . . . .	25
2.2	Executando o xterm. . . . .	30
2.3	Topologia <i>single</i> . . . . .	33

2.4	Topologia linear . . . . .	34
2.5	Visual Network Descriptor. . . . .	40
2.6	Miniedit. . . . .	42
2.7	Gráfico em 2D. . . . .	44
2.8	Gráfico em 3D. . . . .	44
2.9	Mobilidade em ação. . . . .	60
3.1	Interface <i>bonding</i> . . . . .	71
3.2	Pacotes capturados . . . . .	76
3.3	Beacons capturados . . . . .	78
3.4	Tela de captura do <i>linssid</i> com dois pontos acesso. . . . .	81
3.5	Topologia com sobreposição de sinais. . . . .	82
3.6	Tela de captura do <i>linssid</i> com três pontos de acesso . . . . .	82
3.7	Topologia - <i>adhoc</i> e <i>mesh</i> . . . . .	87
3.8	Captura de mensagens OpenFlow. . . . .	93
3.9	Comunicação entre switch e controlador. . . . .	94
3.10	Topologia para o processo de handover. . . . .	103
3.11	Topologia - Servidor DHCP. . . . .	110
3.12	Interface <i>sta1-wlan0</i> . . . . .	112
3.13	Servidor DHCP em funcionamento. . . . .	113
3.14	Topologia - Loop de comutação. . . . .	114
3.15	Topologia - VLANs. . . . .	117
3.16	Identificação da VLAN ID . . . . .	119
3.17	Topologia - Roteamento estático. . . . .	121
3.18	Topologia - Roteamento dinâmico. . . . .	124
3.19	Topologia - QoS. . . . .	129
3.20	Kernel do MP-TCP . . . . .	131
4.1	sFlow-RT em execução. . . . .	143
4.2	Topologia - Internet. . . . .	152

## **LISTA DE FIGURAS**

---

iii

4.3	Placa de rede WiFi conectada ao laptop.	155
4.4	Topologia - Interação com nós físicos.	156
4.5	Pacotes enviados	164
4.6	Encaminhamento por SSID.	168
4.7	Ataque de ARP <i>spoofing</i> .	170
4.8	O handshake de 4 vias do protocolo WPA2.	175
4.9	Snort em execução	179
4.10	Captura de mensagens do protocolo Radius.	182
4.11	Topologia - Radius.	184
4.12	Nós conectados via 6LoWPAN.	186
4.13	Pacotes 6LoWPAN.	188
4.14	Simulador de mobilidade urbana - SUMO.	192





## **Lista de Tabelas**

1.1	Comparação dos padrões IEEE 802.11 .....	8
1.2	Comparação entre os padrões 802.11n e 802.11ac .....	8





## **Lista de Acrônimos**

6LoWPAN	IPv6 over Low power Wireless Personal Area Networks
AP	Access Point
BER	Bit Error Rate
BOFUSS	Basic OpenFlow User-space Software Switch
BSS	Basic Service Set
CAPWAP	Control and Provisioning of Wireless Access Points
CD	Collision Detection
CSMA/CA	Carrier Sense Multiple Access with Collision Avoidance
Hostapd	Host Access Point Daemon
HTTPS	Hyper Text Transfer Protocol Secure
IBSS	Independent Basic Service Set
IDS	Intrusion Detection System
IFB	Intermediate Functional Block

IoT	Internet of Things
LTE	Long-Term Evolution
LWAPP	Lightweight Access Point Protocol
MCS	Modulation and Coding Scheme
MIMO	Multiple Input, Multiple Output
MLME	Media Access Control Sublayer Management Entity
MPTCP	MultiPath TCP
MQTT	Message Queuing Telemetry Transport
NFV	Network Function Virtualization
OF	OpenFlow
ONF	Open Networking Foundation
OVSAP	OpenvSwitch Access Point
RSSI	Received Signal Strength Indicator
RTC	Request To Receive
RTS	Request To Send
SDN	Software Defined Networking
SDWN	Software Defined Wireless Networking
SNR	Signal to Noise Ratio
SSID	Service Set Identifier
STA	Station
SUMO	Simulation of Urban MObility
TC	Traffic Control
UserAP	User level Access Point
VANET	Vehicular Ad hoc NETwork
WLAN	Wireless Local Area Network

# Introdução



<b>1</b>	<b>Fundamentação teórica .....</b>	<b>3</b>
1.1	Introdução às comunicações sem fio	
1.2	WiFi: Redes locais sem fio baseadas no IEEE 802.11	
1.3	Redes sem fio definidas por software	
1.4	Mininet-WiFi	





## 1. Fundamentação teórica

### 1.1 Introdução às comunicações sem fio

As Comunicações Sem Fio continuam sendo um dos campos mais vibrantes da área de Telecomunicações. Embora tendo surgido no final do século XIX e primórdios do século XX, e com as atividades na área se intensificando nas décadas de 1970 a 1990, as pesquisas ainda parecem não ter fim. Isto ocorre devido à confluência de vários fatores, o mais importante deles sendo o aumento explosivo na demanda por conectividade. Impulsionado, inicialmente, pela conhecida telefonia celular e, mais recentemente, por aplicativos de dados, novas formas de comunicação e compartilhamento de acesso a conteúdos, as comunicações sem fio não param de surpreender. Elas têm se tornado fundamentais para prover conectividade de inúmeros objetos do dia-a-dia, uma tendência incorporada pela agora bem conhecida Internet das Coisas, ou, do inglês, *Internet of Things* (IoT).

Uma comunicação sem fio convencional processa-se através de uma rede contendo diversos elementos, os mais básicos listados a seguir: (i) os terminais - hospedeiros sem fio -, como *notebooks*, *smartphones*, representando a

interface entre o usuário e a rede; (ii) enlaces de rádio, conectando os terminais a um agente que provê cobertura do serviço; (iii) as estações rádio-base, representando os agentes de cobertura; (iv) as centrais de comutação e controle, concentrando as estações rádio-base e conectando-as aos demais serviços de comunicação.

São numerosas as tecnologias provendo serviços sem fio. Citam-se, por exemplo, Bluetooth, LTE, Zigbee, WiFi, dentre outras. As comunicações sem fio possuem características únicas que as tornam distintas das demais tecnologias. Uma delas, e certamente a mais predominante, é a propagação de ondas de rádio. Um sinal propagando-se de um ponto a outro sofre três tipos de fenômenos, a saber: atenuação, desvanecimento de longo prazo e desvanecimento de curto prazo. A atenuação diz respeito à perda de percurso quando o receptor se afasta da fonte. O desvanecimento de longo prazo refere-se à flutuação lenta (da ordem de dezenas de comprimentos de onda) da intensidade do sinal devido a obstruções presentes no percurso, como prédios, árvores, etc. O desvanecimento de curto prazo diz respeito à flutuação rápida (da ordem de frações de comprimento de onda) da intensidade do sinal devido ao fenômeno

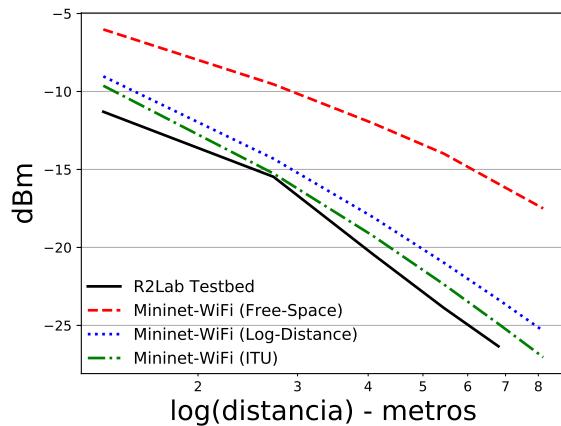


Figura 1.1: Relação do nível de sinal (RSSI) em dBm na comunicação sem fio entre dois nós 802.11 aumenta com a distância física. Fonte e mais informações: [18]

de múltiplos caminhos percorridos. Há ainda a questão da interferência por serviços utilizando a mesma frequência ou mesmo frequências próximas.

Devido à demanda cada vez mais crescente pelas comunicações sem fio, novas tecnologias surgem de forma que os sistemas possam satisfazer a referida demanda. De qualquer forma, um projeto de qualquer sistema, e mais especificamente de sistemas de comunicações sem fio, requer o conhecimento profundo dos fenômenos envolvidos.

A Figura 1.1 exemplifica o fenômeno de perda de percurso, a exemplo da atenuação do nível do sinal (em inglês, *Received Signal Strength Indicator* (RSSI)), em dBm, em função da distância física entre uma estação base e um hospedeiro sem fio conforme estimado por diferentes modelos de propagação da literatura (*Free-Space*, *Log-Distance*, *ITU*) disponíveis no emulador Mininet-WiFi comparados com medições obtidas em um ambiente laboratorial, o *testbed R2Lab*<sup>1</sup>. A Figura 1.2, por sua vez, ilustra os fenômenos de desvanecimento de longo prazo e de curto prazo.

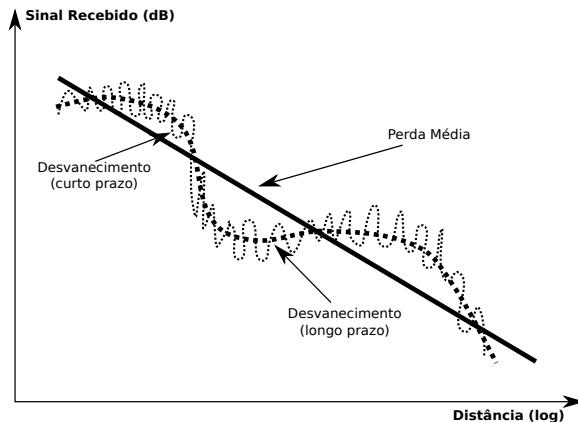


Figura 1.2: Fenômenos de desvanecimento.

<sup>1</sup><https://r2lab.inria.fr>



- H. Waldman e M. D. Yacoub, *Telecomunicações: Princípios e Tendências*, Editora Érica, 1997. ISBN: 978-8571944374
- J. C. de Oliveira Medeiros. *Princípios de Telecomunicações. Teoria e Prática*. Editora Érica, 2015. ISBN: 978-8536516288
- M. D. Yacoub, *Foundations Of Mobile Radio Engineering*. CRC Press, 1993. ISBN: 978-0849386770
- M. D. Yacoub, *Wireless Technology: Protocols, Standards, and Techniques*, CRC Press, 2001. ISBN: 978-0849309694
- C. J. Weisman, *The Essential Guide to RF and Wireless*. Prentice Hall. 2002. ISBN: 978-0130354655
- T. Rappaport, *Wireless Communications: Principles and Practice*, Pearson Education India, 2010. ISBN: 978-0130422323
- A. K. Jagannatham, *Principles of Modern Wireless Communication Systems Theory and Practice*. McGraw Hill Education, 2017. ISBN: 978-1259029578

## 1.2 WiFi: Redes locais sem fio baseadas no IEEE 802.11

Estabelecido pelo *Institute of Electrical and Electronics Engineers* (IEEE), o IEEE 802.11 é o padrão de comunicações para redes sem fio mais aceito mundialmente. A tecnologia WiFi, como é mais comumente conhecida, é a tecnologia para *Wireless Local Area Network* (WLAN) baseada no IEEE 802.11, sendo um *trademark* da WiFi Alliance. Os motivos para a larga aceitação desse padrão são diversos, destacando-se, principalmente, a relação custo-desempenho.

Conforme ilustrado na Figura 1.3, existem diversos padrões 802.11, dentre eles os mais antigos 802.11b, 802.11a e 802.11g, e também outros que podem ser considerados como mais recentes, a exemplo do 802.11n, 802.11ac, 802.11p, etc. Em geral, os padrões definidos para 802.11 operam em 2 (duas) frequências principais: 2.4 GHz ou 5 GHz, onde são alocados um número de canais com uma largura específica. No exemplo da Figura 1.4, pode-se observar como o padrão 802.11b define 13 canais na banda de 2.4 GHz a 2.4835 Ghz, alocando 22 MHz para cada canal com um espaçamento de 5 MHz entre eles. Com esta disposição, somente os canais 1, 6 e 11 conseguem operar sem sobreposição de faixas.

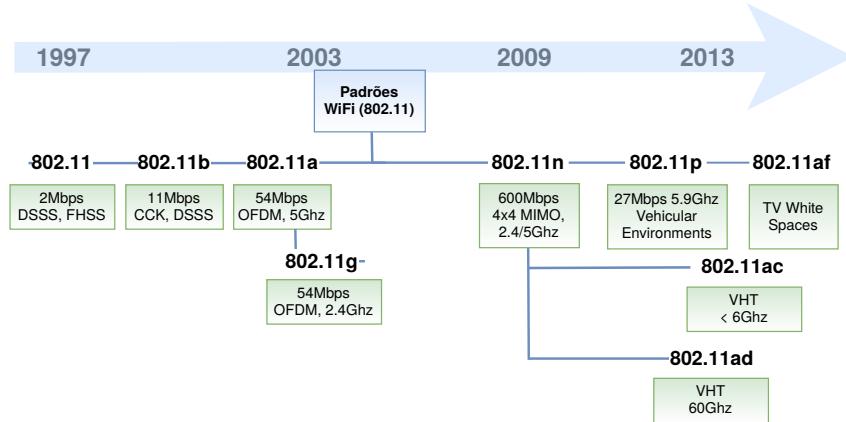


Figura 1.3: Padrões IEEE 802.11.

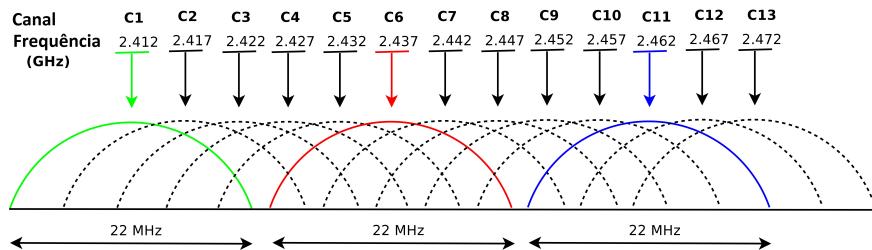


Figura 1.4: Disposição dos canais no IEEE 802.11b. Fonte: Adaptado de [3] (CC BY 2.0)

Conhecidos o esquema de modulação, o tipo de codificação, e uma relação sinal-ruído (*Signal to Noise Ratio* - SNR), é possível determinar-se a taxa de erro de bits (*Bit Error Rate* - BER), que é um requisito a ser satisfeito no projeto do sistema. Sabe-se que um aumento de potência no transmissor resulta em um maior SNR e consequente diminuição do BER. Obviamente, não se pode aumentar a potência indefinidamente, por questões de interferência e por limitação desta no próprio transmissor.

A tabela 1.1 compara diferentes padrões 802.11 em termos de frequência de operação, largura de banda do canal e estimativas do rádio de cobertura em ambientes internos e externos. Já a tabela 1.2 foca na comparação entre o 802.11n e 802.11ac, dois dos padrões mais recentes que incorporaram os novos

Tabela 1.1: Comparação técnica entre diversos padrões IEEE 802.11.

Protocolo	Freq. (GHz)	Largura de Banda (MHz)	Cobertura Aproximada Interna	Cobertura Aproximada Externa
802.11	2.4	20	20 m / 66 ft	100 m / 330 ft
802.11a	3.7 / 5	20	35 m / 115 ft	120 m / 390 ft
802.11b	2.4	20	35 m / 115 ft	140 m / 460 ft
802.11g	2.4	20	38 m / 125 ft	140 m / 460 ft
802.11n	2.4/5	20 - 40	70 m / 230 ft	250 m / 820 ft
802.11ac	5	20/40/80/160	35 m / 115 ft	n/d
802.11ad	60	2,160	60 m / 200 ft	100 m / 300 ft
802.11ay	60	8000	60 m / 200 ft	1000 m / 3000 ft

Tabela 1.2: Comparação técnica entre os padrões 802.11n e 802.11ac.

	IEEE 802.11n	IEEE 802.11ac
Frequência	2.4 GHz & 5 GHz	5 GHz
MIMO	Único Usuário (SU)	Multi Usuário (MU)
Fluxos Espaciais	4	8
Taxa PHY	600 Mbps	6.9 Gbps
Largura de Canal	20 or 40 MHz	20, 40, 80, 80-80, 160 MHz
Modulação	64 QAM	256 QAM
Vazão MAC*	390 Mbps	4.49 Gbps

\*Assumindo 65% de eficiência MAC e a maior taxa de MCS  
(*Modulation and Coding Scheme*)

avanços das comunicações sem fio, tais como fluxos espaciais baseados na tecnologia MIMO (*Multiple Input, Multiple Output*).

A arquitetura de uma rede 802.11 é composta principalmente por um ponto de acesso e estações sem fio (clientes). Nesse caso, a arquitetura é definida como um conjunto básico de serviço, *Basic Service Set (BSS)*, ou modo infraestruturado. Em contrapartida, as redes 802.11 compostas por apenas estações sem fio (clientes) são denominadas como um conjunto básico de serviço independentes (em inglês, *Independent Basic Service Set - IBSS*) ou modo *adhoc*. A representação gráfica destas duas arquiteturas (ou modos de operação) está ilustrada nas Figuras 1.5 e 1.6.

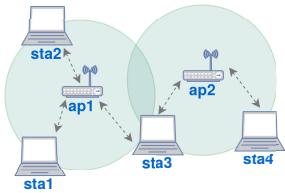
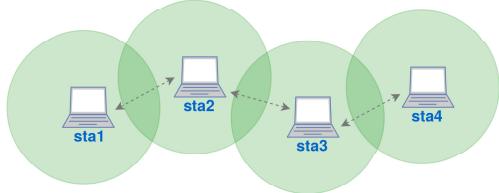


Figura 1.5: Modo infraestruturado.

Figura 1.6: Modo *adhoc*.

Assim como acontece com dispositivos Ethernet, cada dispositivo sem fio 802.11 possui um endereço MAC de 6 bytes armazenado na placa de interface de rede. É através da interface de rede sem fio que as estações podem se associar a um ponto de acesso ou mesmo outras estações clientes, antes de receber ou enviar quadros 802.11.

Uma vez que nas redes sem fios não existem meios físicos que permitam evitar colisões, estas ocorrem mesmo com as tecnologias sem fio mais avançadas. Enquanto nos padrões para Ethernet cabeados da família IEEE 802 é possível a detecção de colisão (*Collision Detection* - CD), nas redes sem fio não existem meios para detectar uma colisão. A estratégia adotada pelos padrões 802.11 para lidar com o controle de acesso ao meio sem fio é conhecido como acesso múltiplo com verificação de portadora com prevenção de colisão (*Carrier Sense Multiple Access with Collision Avoidance* - CSMA/CA).

No método CSMA/CA, cada estação avisa sobre a intenção de transmissão e o tempo associado para prevenir colisões (*Collision Avoidance* - CA). Os equipamentos com interfaces sem fio escutam o meio para verificar a presença de sinais (nível de sinal na frequência portadora) e esperam até que o meio fique livre antes de transmitir. Esses mecanismos são conhecidos como “solicitar para enviar” (*Request to Send* - RTS) e “livre para enviar” (*Clear to Send* - CTS).

Apesar das semelhanças entre quadros Ethernet e quadros 802.11, existem diversos campos que são específicos para enlaces sem fio. Os campos do quadro 802.11 podem ser visualizados na Figura 1.7. Os números acima de cada campo no quadro representam os comprimentos dos campos em *bytes*,

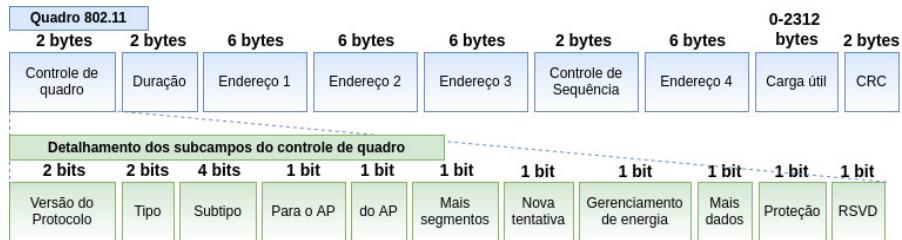


Figura 1.7: O quadro 802.11.

enquanto que os números acima de cada um dos subcampos no campo de *controle de quadro* representam os comprimentos dos subcampos em *bits*.

Embora não entremos em detalhes sobre a função de cada um dos campos e subcampos pertencentes ao quadro 802.11, é recomendável o conhecimento sobre eles mesmo que superficialmente. Esses campos podem ser úteis para exploração mais aprofundada de alguns dos tutoriais que serão apresentados ao longo deste livro.

### O futuro do WiFi

Embora sejam muito importantes, ainda existem barreiras estruturais que impedem a inovação em redes sem fio, como o próprio WiFi. Além disso, as grandes infraestruturas sem fio não estão completamente acessíveis, pois existem restrições no seu uso ou exigem autenticação. A questão não é abrir completa e gratuitamente o acesso às redes sem fio, mas permitir que usuários se conectem a várias redes (preservando requisitos de segurança e de qualidade), abrindo enorme capacidade de cobertura, além de possibilitar a inovação contínua, conforme proposto em [20].

Não obstante, já existem diversas pesquisas acerca das redes veiculares e também da Internet das Coisas que consideram as redes WiFi em suas propostas. Muitas delas, claro, representam sugestões de melhorias que podem significar avanços para 802.11 em um futuro muito breve. Não é a toa que já se fala em 802.11ax, evolução do 802.11ac, que promete conectar mais dispositivos com taxa de transmissão superior ao seu antecessor.

Dentre as propostas de melhorias e avanços em redes sem fio e principalmente o WiFi, está o conceito das redes sem fio definidas por software que também promete avanços significativos na construção de uma nova ideia de conectividade. Sendo assim, aliado a esse conceito de redes sem fio definidas por software, este livro irá apresentar uma série de tutoriais que irão explorar diversos casos de uso através do Mininet-WiFi. O Mininet-WiFi é o emulador para redes sem fio que iremos utilizar bastante ao longo deste livro. Ele foi desenvolvido com o intuito de proporcionar um ambiente capaz de servir como suporte em pesquisas para as redes sem fio e redes sem fio definidas por software, possibilitando inovações para as mais diversas tecnologias sem fio.



- Matthew S. Gast. *802.11 Wireless Networks: The Definitive Guide*. O'Reilly Media, 2005. ISBN-13: 978-0596100520
- Matthew S. Gast. *802.11ac: A Survival Guide: Wi-Fi at Gigabit and Beyond*. O'Reilly Media (Edição: 2), 2013. ISBN-13: 978-1449343149
- Jim Geier, *Designing and Deploying 802.11 Wireless Networks: A Practical Guide to Implementing 802.11n and 802.11ac Wireless Networks For Enterprise-Based Applications*. Cisco Press, 2015. ISBN-13: 978-1587144301
- IEEE 802.11 Wireless Local Area Networks. The Working Group for WLAN Standards. Disponível em: <http://www.ieee802.org/11/>

### 1.3 Redes sem fio definidas por software

Redes Sem Fio Definidas por Software (do inglês, *Software-Defined Wireless Networking* - SDWN) [5, 11] consiste de uma abordagem que permite o controle centralizado da rede através de aplicações que não precisam estar localizadas necessariamente em pontos de acesso. Assim, regras definidas através de aplicações, mais comumente conhecidas como controladores são quem ditam o comportamento da rede. Os princípios de SDWN são muito similares àqueles definidos para Redes Definidas por Software (*Software-Defined Networking* - SDN) [12], que consiste na separação do plano de controle (ou sistema operacional da rede) do plano de dados (ou plano de encaminhamento).

A abordagem definida por software permite aos administradores de rede especificar o comportamento da rede de forma lógica e centralizada, através de

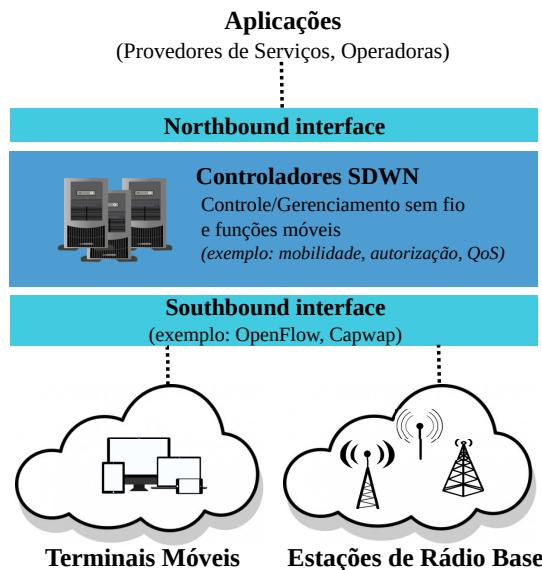


Figura 1.8: Arquitetura de alto nível e genérica para SDWN. Fonte: Adaptado de [18]

aplicações fornecidas pelas plataformas de controle que implementam o que chamamos de *southbound interfaces* para dispositivos de rede, tais como *switches*. Nesse contexto, o protocolo OpenFlow [15] aparece como a *southbound interface* mais popular. Porém, existem outras, a exemplo do CAPWAP [22], FORCES [7], NETCONF [8], etc.

Devido ao interesse das operadoras de telefonia móvel [2, 19], SDWN tem se tornado um ramo das redes definidas por software de bastante interesse da comunidade científica, impulsionado principalmente pelo aumento do interesse das operadoras de telefonia móvel pelas funções de redes virtualizadas (ou *Network Function Virtualization - NFV*) [10].

A separação entre o plano de controle o plano de dados não é uma novidade no contexto das redes sem fio. O IETF havia padronizado tanto o LWAPP (*Lightweight Access Point Protocol*), quanto o CAPWAP (*Control And Provisioning of Wireless Access Points*) muitos anos atrás por meio das RFCs RFC5412 [4] e RFC4564 [22], respectivamente. Mesmo antes das redes

definidas por software e o próprio protocolo OpenFlow.

Muitas empresas utilizam sistemas de gerenciamento de redes sem fio através de protocolos como o LWAPP e CAPWAP. O LWAPP define o controle de mensagem para configuração, autenticação e algumas outras operações, enquanto que o CAPWAP é baseado no LWAPP, e habilita um controlador para gerenciar diferentes pontos de acesso.

Estudos e pesquisas acerca das redes sem fio definidas por software cresceram bastante nos últimos anos. Vale apena consultar [11] para uma pesquisa mais abrangente, e também algumas propostas, tais como: OpenRoads [23], Odin [21], OpenRF [13], Ethanol [17], entre outras. Arquiteturas como o CloudMac [6] e Chandelle [16] inclusive, utilizam CAPWAP em suas propostas. CloudMac descreve protocolos de gerenciamento de redes sem fio, a exemplo do CAPWAP, como um protocolo difícil de implementar novas funcionalidades, uma vez que controladores de pontos de acesso com o CAPWAP são em sua maioria sistemas proprietários. Chandelle, por sua vez, propõe uma migração entre pontos de acesso suave e rápida com SDN/OpenFlow, mas sofre com desafios de integração com *switches* tradicionais e CAPWAP.



Importante destacar que há uma implementação de código aberto do protocolo CAPWAP de acordo com a RFC 4515 e a RFC 4516, chamada OpenCAPWAP [1], que teve o desenvolvimento iniciado em 2015 (<https://github.com/vollero/openCAPWAP>).

Os benefícios identificados da integração das redes sem fio com OpenFlow são comumente relacionados ao gerenciamento e monitoramento centralizados, políticas unificadas, maior capacidade de programação e controle mais refinado das funções das redes em fio.

Levando em conta esses benefícios e as limitações associadas ao CAPWAP, que provavelmente até então apresenta ser uma solução mais robusta, mas de código-fonte fechado, algumas perguntas são inevitáveis: “*O CAPWAP está no escopo de SDWN?*”, “*Como melhorar a especificação do OpenFlow para suportar o gerenciamento centralizado de redes sem fio? Ou mesmo, seria possível estendê-lo para as redes sem fio?*”, “*Novas abordagens são necessárias para integrar o CAPWAP com o OpenFlow?*”.

rias?” ou “*Quanto poderia ser aproveitado das infra-estruturas atualmente existentes?*”.



- L. E. Li, Z. M. Mao and J. Rexford, *Toward Software-Defined Cellular Networks*. European Workshop on Software Defined Networking (EWSDN), 2012.
- A. Gudipati et al., *SoftRAN: software defined radio access network*. Proceedings of Hot topics in software defined networking (HotSDN). 2013.
- C. J. Bernardos et al., *An architecture for software defined wireless networking*. IEEE Wireless Communications. 2014.
- T. Chen et al., *Software defined mobile networks: concept, survey, and research directions*, IEEE Communications Magazine. 2015.
- Mao Yang et al., *Software-Defined and Virtualized Future Mobile and Wireless Networks: A Survey*. Mob. Netw. Appl. 2015.
- I. T. Haque and N. Abu-Ghazaleh, *Wireless Software Defined Networking: A Survey and Taxonomy*, in IEEE Communications Surveys & Tutorials. 2016.
- A. Abdelaziz et al. *On Software-Defined Wireless Network (SDWN) Network Virtualization: Challenges and Open Issues*. Computer Journal. 2017.
- Linux Foundation’s Open Networking Foundation (ONF) SDN Wireless Transport. Disponível em: <https://www.opennetworking.org/tag/wireless-transport/>

## 1.4 Mininet-WiFi

A emulação de redes tem sido bastante utilizada na avaliação de desempenho, testes e depuração de protocolos e também em diversos assuntos relacionados a pesquisas em arquiteturas de redes de computadores.

Um pesquisador tem tipicamente várias possibilidades para avaliar e validar pesquisas, protocolos de rede, bem como realizar análises, entre outros. Simuladores, emuladores e *testbeds* são os principais métodos de avaliação que auxiliam os pesquisadores em suas tarefas. Em relação à aplicação real, todos esses métodos de avaliação são muito diferentes em seu grau de abstração. Algumas das plataformas experimentais que podem ser utilizadas para a experimentação em redes sem fio são apresentadas na Figura 1.9. Dentro desse universo de experimentação a emulação de redes sem fio, que possui características peculiares, principalmente se comparada com emuladores para

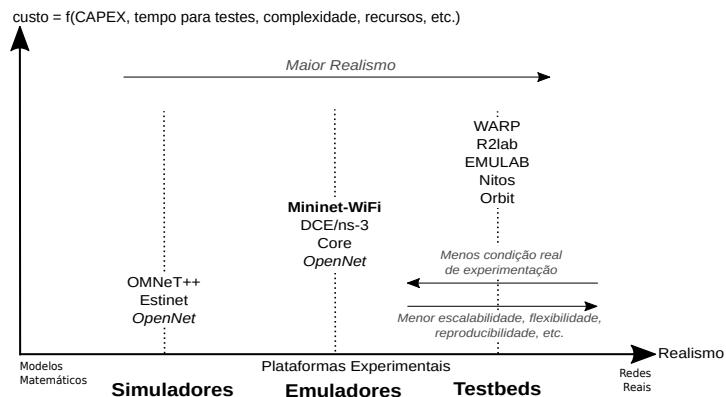


Figura 1.9: Plataformas experimentais para redes sem fio. Fonte: Adaptado de [9].

redes cabeadas, tem que implementar mobilidade dos nós, propagação do sinal, entre outros para permitir experimentos com ambientes que incluem interferência, atenuação de sinal, etc. Não iremos entrar em detalhes sobre as diferenças entre as plataformas experimentais, mas podemos destacar duas importantes características do Mininet-WiFi: (i) possibilita utilizar ferramentas de terceiros sem modificações no código-fonte destas ferramentas; e (ii) utiliza a pilha de protocolos de rede reais.

O Mininet-WiFi é um emulador para redes sem fio estendido a partir do Mininet, emulador bastante conhecido por pesquisadores que já atuam na área das redes definidas por software. O Mininet-WiFi possui suporte nativo a WiFi, mas outras tecnologias de redes sem fio também podem ser simuladas em experimentos. Com ele é possível a virtualização de estações e pontos de acesso e também utilizar os nós já presentes do Mininet, como *hosts*, *switches* e controladores OpenFlow. Consequentemente, o Mininet-WiFi também possibilita o processamento de pacotes utilizando o protocolo OpenFlow, importante solução para SDN.

O Mininet-WiFi está sendo desenvolvido com base no código do Mininet e do WiFi *driver* mais utilizado em sistemas Linux, o *SoftMac*. Com o Mininet-WiFi o usuário pode optar por utilizar os mesmos recursos do Mininet de forma independente ou utilizar as extensões implementadas para o

Mininet-WiFi.



*SoftMAC* é um termo usado para descrever um tipo de interface de rede sem fio em que se espera que o *MAC Layer Management Entity* (MLME), por exemplo, seja gerenciado em software. O mac80211 é uma API de driver para o *SoftMAC*.

#### 1.4.1 Arquitetura

Todo o processo de virtualização do Mininet-WiFi funciona de forma similar ao Mininet, tendo como base processos que são executados em Linux *Network Namespaces* e placas de rede virtuais (ver Figura 1.10). Linux *Network Namespaces*, de forma lógica, representam uma cópia da pilha de rede do sistema operacional Linux, que inclui suas próprias rotas, regras de firewall e dispositivos de rede. Eles atuam como se fossem verdadeiros computadores com as mesmas propriedades de rede que um computador físico pode ter.

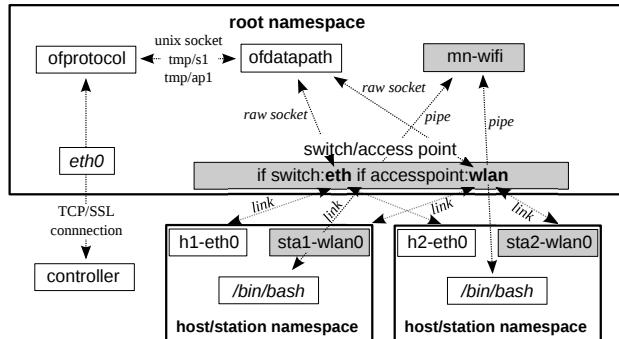


Figura 1.10: Arquitetura do Mininet-WiFi. Fonte: [9]

O modo de funcionamento das interfaces sem fio depende basicamente da função ao qual ela exerce, como no caso de estações e pontos de acesso, onde as interfaces operam nos modos *managed* ou *master*, respectivamente. Assim como em um ambiente real, as estações se comunicam com os pontos de acesso através de um processo chamado de autenticação e associação. Por padrão, cada estação possui apenas uma interface sem fio, podendo ser adicionadas

mais caso houver necessidade. Uma vez associada a um ponto de acesso, as estações podem se comunicar com os tradicionais *hosts* do Mininet, caso estes estiverem também conectados ao ponto de acesso através de um meio cabeadão. Pontos de acesso, por outro lado, são responsáveis pela gestão das estações que estão associadas a ele.

Por conceito, pontos de acesso são os mesmos *switches* do Mininet, mas equipados com placas de rede WiFi operando em modo *master*. Os pontos de acesso são virtualizados através do *daemon hostapd*<sup>2</sup>, que basicamente utilizam interfaces WiFi virtuais para prover capacidades de um ponto de acesso. Detalhes sobre o ambiente de execução do Mininet-WiFi são comentados a seguir.

#### 1.4.2 Funcionamento

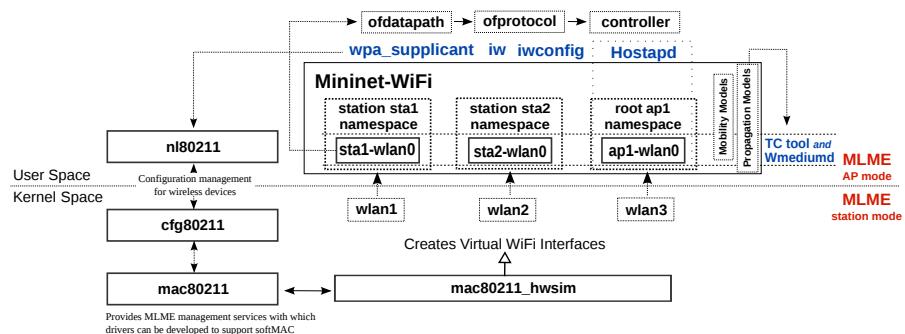


Figura 1.11: Principais componentes do Mininet-WiFi. Fonte: [9]

Os componentes que fazem parte da arquitetura do Mininet-WiFi estão apresentadas na Figura 1.11. O processo de comunicação entre eles ocorre da seguinte forma: durante sua inicialização, o módulo chamado *mac80211\_hwsim*, responsável pela virtualização das placas de rede WiFi, é carregado com o número de interfaces sem fio virtuais necessárias para todos os nós previamente definidos pelo usuário. Localizado no espaço do kernel do sistema operacional Linux, todos os recursos suportados pelo *mac80211\_hwsim* são provenientes

<sup>2</sup>Hostapd (**H**ost **A**ccess **P**oint **D**aemon) é um software a nível de usuário capaz de tornar uma interface de rede sem fio em pontos de acesso e servidores de autenticação.

do `mac80211`, *framework* que desenvolvedores de *drivers* utilizam para escrever *drivers* para dispositivos sem fio baseados no *SoftMAC*.

Ainda no espaço de kernel do sistema operacional está o `cfg80211`. O `cfg80211` é uma API de configuração da pilha 802.11 do sistema operacional Linux. A configuração é feita através do `netlink nl80211`, que também faz a função de interação entre os espaços de kernel e de usuário.

Já no espaço de usuário estão as aplicações que podem ser utilizadas no ambiente de execução do Mininet-WiFi. Dentre eles está o *daemon hostapd*, que é utilizado para prover os serviços de ponto de acesso; o `TC` e `Wmediumd` que serão apresentados logo em breve, `iw`, `iwconfig` e `wpa_supplicant`. Este último é utilizado, dentre outras coisas, para autenticação WPA/WPA2.

### **Interagindo com o ambiente de emulação**

O Mininet-WiFi também mantém a mesma estrutura de interação do Mininet. Por exemplo, comandos como os demonstrados abaixo podem ser utilizados, respectivamente, para testes de conectividade ou para mensurar largura de banda entre dois nós. Se você, leitor, já conhece o Mininet, certamente isso não é uma novidade.

```
mininet-wifi> sta1 ping sta2  
mininet-wifi> iperf sta1 sta2
```

Adicionalmente, outros comandos disponíveis apenas para o Mininet-WiFi podem ser utilizados para melhor experiência com o ambiente WiFi, tais como:

```
mininet-wifi> sta1 iw dev sta1-wlan0 scan  
mininet-wifi> sta1 iw dev sta1-wlan0 connect ssid-ap1
```

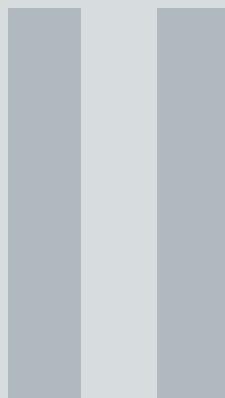
Esses comandos permitem escanear redes WiFi e conectar-se a uma delas, respectivamente. Ferramentas como o `iw`, comando utilizado acima, são suportados nativamente pela maioria dos sistemas operacionais Linux e não foram implementados ou modificados para funcionarem no Mininet-WiFi. Em geral, é possível executar no Mininet-WiFi qualquer comando e/ou aplicação que rode em sistemas operacionais Linux, como o Ubuntu.



- Ramon dos Reis Fontes, Samira Afzal, Samuel Brito, Mateus Santos, Christian Esteve Rothenberg. *Mininet-WiFi: Emulating Software-Defined Wireless Networks*. In 2nd International Workshop on Management of SDN and NFV Systems 2015. Barcelona, Spain, Nov. 2015. [9]
- Ramon dos Reis Fontes, *Mininet-WiFi: Emulation Platform for Software-Defined Wireless Networks*. Tese de Doutorado em Engenharia Elétrica, FEEC/UNICAMP, Jun. 2018. Disponível em: [http://repositorio.unicamp.br/jspui/bitstream/REPOSIP/332708/1/Fontes\\_RamonDosReis\\_D.pdf](http://repositorio.unicamp.br/jspui/bitstream/REPOSIP/332708/1/Fontes_RamonDosReis_D.pdf)



# Nível: iniciante



<b>2</b>	<b>Iniciante .....</b>	<b>23</b>
2.1	Primeiros passos com Mininet-WiFi	
2.2	Customizando topologias	
2.3	Acessando informações dos nós	
2.4	OVSAP <i>versus</i> UserAP	
2.5	Utilizando interfaces gráficas	
2.6	Emulação do meio sem fio	
2.7	Modelos de propagação	
2.8	Relação distância <i>versus</i> sinal recebido	
2.9	Modificando o <i>bitrate</i>	
2.10	Relação distância <i>versus</i> largura de banda	
2.11	Modelos de mobilidade	





## 2. Iniciante

Neste capítulo apresentamos o Mininet-WiFi e todos os recursos suportados por este emulador, além de informações importantes que irão possibilitar um bom entendimento dos tutoriais que serão explorados ao longo deste livro.

O código-fonte do Mininet-WiFi está publicamente disponível em um repositório no Github, uma plataforma que permite o depósito de códigos-fonte com controle de versão. Isso significa que o ciclo de vida de um projeto pode ser facilmente analisado através do Github, que mantém o histórico de modificações de arquivos desde a sua origem.

Para obter o código-fonte do Mininet-WiFi e instalá-lo, será necessário realizar um processo chamado de clone, onde uma espécie de *download* do código-fonte é realizado. O clone é realizado através do git, o sistema de controle de versões utilizado pelo Github.

Após essa breve introdução do git e Github, considere realizar o clone do código-fonte do Mininet-WiFi através do comando a seguir.

```
~$ git clone https://github.com/intrig-unicamp/mininet-wifi
```



O Mininet-WiFi tem sido testado exaustivamente em diferentes versões do sistema operacional Ubuntu, portanto, recomendamos a utilização deste sistema operacional.

O texto *intrig-unicamp* diz respeito ao perfil ou organização onde o repositório está localizado no Github. O texto *mininet-wifi*, por sua vez, é o nome do repositório onde o código-fonte está depositado.



Se você não tiver o git poderá instalá-lo através do comando `sudo apt install git`.

Uma vez concluído o clone, um diretório com o nome *<mininet-wifi>* deve ter sido criado. Considerando que o clone foi realizado a partir do diretório raiz do usuário do sistema, o código-fonte do Mininet-WiFi deverá estar localizado em *</home/seu\_usuario/mininet-wifi>*, ou simplesmente *<~/mininet-wifi>*.

Agora, é preciso instalar o Mininet-WiFi. Para tanto, será necessário acessar o diretório criado e executar o comando `sudo util/install.sh`, conforme abaixo.

```
~$ cd mininet-wifi  
~/mininet-wifi$ sudo util/install.sh -Wlnfv6
```



Maiores informações sobre os parâmetros *Wlnfv6* podem ser obtidas na página do código-fonte do Mininet-WiFi no Github.

De forma alternativa, também é possível utilizar a máquina virtual pré-configurada disponível na página do código-fonte. Para garantir que a máquina virtual possua a versão mais recente do Mininet-WiFi, é necessário utilizar os 2 (dois) comandos a seguir.

```
~/mininet-wifi$ git pull  
~/mininet-wifi$ sudo make install
```



Capturando o código através do `git clone` você terá garantias que o código-fonte obtido sempre terá as últimas atualizações implementadas para o Mininet-WiFi.

Mesmo que você já possua o Mininet-WiFi instalado e mesmo que já possua a máquina virtual pré-configurada, o comando `git pull` poderá ser emitido a partir do diretório do Mininet-WiFi a qualquer tempo. Este comando irá sincronizar o código que está em seu computador com o código-fonte disponível no repositório online do Mininet-WiFi. Assim, você terá sempre a versão mais recente do Mininet-WiFi.

## 2.1 Primeiros passos com Mininet-WiFi

A partir de agora vamos começar a compreender como utilizar o Mininet-WiFi.

Primeiro, é bom estarmos atentos a 3 (três) comandos: o `sudo mn --version`, que imprime a versão instalada do Mininet-WiFi; o `sudo mn --help`, que imprime um menu de ajuda; e o `sudo mn -c`, que é responsável por limpar execuções mal feitas do Mininet-WiFi. Lembre-se deste último comando, pois ele certamente será muito útil em algum momento.

O Mininet-WiFi pode ser iniciado através de um comando bastante simples, o `sudo mn --wifi`. Além de abrir a interface de linha de comando (*Command Line Interface - CLI*) do Mininet-WiFi, esse comando irá criar uma topologia que consiste em 2 (duas) estações conectadas a 1 (um) ponto de acesso através de um meio sem fio. Além disso, também há a presença de 1 (um) controlador SDN que está conectado ao ponto de acesso, conforme ilustrado na Figura 2.1.

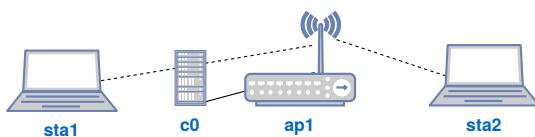


Figura 2.1: Topologia simples.

```
~/mininet-wifi$ sudo mn --wifi  
*** Creating network
```

```
*** Adding controller
*** Adding stations:
sta1 sta2
*** Adding access points:
ap1
*** Configuring wifi nodes...
*** Adding link(s):
(sta1, ap1) (sta2, ap1)
*** Configuring nodes
*** Starting controller(s)
c0
*** Starting switches and/or access points
ap1 ...
*** Starting CLI:
mininet-wifi>
```



Se você já conhece o Mininet, provavelmente já utilizou o comando `sudo mn`, comando este que também cria uma topologia simples, mas com 2 (dois) *hosts*, 1 (um) *switch* e 1 (um) controlador, conectados por um meio cabeados.



Caso observar um erro similar ao demonstrado abaixo, significa que existe um controlador já utilizando a porta 6653, porta padrão utilizada pelos controladores OpenFlow mais recentes. Este pequeno problema pode ser solucionado através do comando `sudo fuser -k 6653/tcp`, que irá encerrar o processo que estiver utilizando a porta 6653. Caso o controlador estiver sendo executado na porta 6633, o mesmo deverá ser feito com esta porta.

```
Exception: Please shut down the controller which is running on port 6653:
Active Internet connections (servers and established)
tcp  0  0 0.0.0.0:6653  0.0.0.0:*  LISTEN  2449/ovs-testcontro
tcp  0  0 127.0.0.1:55118  127.0.0.1:6653 TIME_WAIT  -
```

Para identificar a CLI do Mininet-WiFi basta observar o texto abaixo:

```
mininet-wifi>
```

Dentro da CLI é possível utilizar basicamente qualquer comando de rede ou aplicações. Além disso, também é possível listar e utilizar alguns comandos que foram implementados exclusivamente para o Mininet-WiFi. O comando

`help` irá auxiliar na relação desses comandos desenvolvidos para o Mininet-WiFi, conforme abaixo.

```
mininet-wifi> help
Documented commands (type help <topic>):
=====
EOF      exit    iperf    nodes      pingpair    py      start    x
distance  gterm   iperfudp  noecho    pingpairfull quit   stop     xterm
dpctl    help    links    pingall   ports      sh      switch
dump     intfs   net      pingallfull px      source   time
```

Boa parte desses comandos já estavam presentes no Mininet e foram mantidos para o Mininet-WiFi. No Mininet-WiFi, até então, foram adicionados apenas três novos comandos: `distance`, `start` e `stop`. O `distance` permite verificar a distância entre dois nós, enquanto que o `start` e o `stop` permitem parar e dar sequência a um experimento que implementa mobilidade dos nós.



Este livro mostrará na prática os comandos implementados para o Mininet-WiFi, além de alguns outros já implementados para o Mininet.

Experimente utilizar agora o comando `nodes` para identificar os nós que fazem parte da topologia. Observe que os nós apresentados pelo comando `nodes` são os mesmos nós apresentados anteriormente na Figura 2.1.

**Nota:** O nó C0 será abordado mais adiante.

```
mininet-wifi> nodes
available nodes are:
ap1 c0 sta1 sta2
```

Como já foi dito anteriormente, o comando `sudo mn --wifi` cria uma topologia com estações conectadas através de um meio sem fio a um ponto de acesso. Isso pode ser facilmente verificado através de ferramentas de redes sem fio.

Embora o comando `sudo mn --wifi` crie uma rede WiFi com SSID chamado de “my-ssid”, operando no canal 1 (frequência 2412MHz), esses valores também podem ser customizados. Por exemplo, vamos sair da CLI do Mininet-WiFi com o comando `exit` e então configurar um novo SSID e um novo canal, conforme abaixo:

```
mininet-wifi> exit  
~/mininet-wifi$ sudo mn --wifi --ssid=new-ssid --channel=10
```

Então, experimente o comando abaixo.

```
mininet-wifi> sta1 iw dev sta1-wlan0 info  
Interface sta1-wlan0  
    ifindex 33  
    wdev 0x1000000001  
    addr 02:00:00:00:00:00  
    ssid new-ssid  
    type managed  
    wiphy 16  
    channel 10 (2457 MHz), width: 20 MHz (no HT), center1: 2457 MHz  
    txpower 14.00 dBm
```

Se você é um iniciante em redes sem fio, especialmente em sistemas operacionais Linux, talvez não tenha observado, mas acabou de utilizar um programa muito comum em ambientes de redes sem fio, o *iw*. O *iw* é um utilitário para redes sem fio que vem gradativamente substituindo o *iwconfig*. Vamos utilizá-lo bastante ao longo deste livro.



O *iwconfig* certamente já está instalado em seu sistema e você também pode utilizá-lo. Por exemplo, *sta1 iwconfig* produzirá resultado similar ao apresentado anteriormente pelo *iw*. Experimente rodar *iwconfig --help* para mais informações de como utilizá-lo.

Com relação ao comando que acabamos de utilizar, o parâmetro *info* traz informações acerca da associação (ou não associação) dos nós. É possível notar que *sta1* está associado a um ponto de acesso com SSID *new-ssid*, e que está operando no canal 10, exatamente como definido no comando.

De forma complementar, utilizando o parâmetro *link* ao invés do *info* é possível obter o nível de sinal percebido pelo nó, o *bitrate*, além de pacotes transmitidos e recebidos, etc.

```
mininet-wifi> sta1 iw dev sta1-wlan0 link  
Connected to 02:00:00:00:02:00 (on sta1-wlan0)  
SSID: new-ssid
```

```

freq: 2457
RX: 1241 bytes (22 packets)
TX: 93 bytes (2 packets)
signal: -36 dBm
tx bitrate: 1.0 MBit/s

bss flags:      short-slot-time
dtim period:    2rendering this PDF.

beacon int:     100

```

Agora, vamos utilizar o comando *ping* para verificar a conectividade entre `sta1` e `sta2`:

```

mininet-wifi> sta1 ping -c1 sta2
PING 10.0.0.2 (10.0.0.2) 56(84) bytes of data.
64 bytes from 10.0.0.2: icmp_seq=1 ttl=64 time=0.380 ms

--- 10.0.0.2 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.380/0.380/0.380/0.000 ms

```

O comando nos mostra que há comunicação entre os dois nós em questão, pois há um tempo de resposta de `sta2` em milissegundos (*ms*). É importante destacar que como o Mininet-WiFi é uma plataforma de emulação capaz de emular diversos nós, é necessário definir na CLI a origem ou o nó que será responsável, na prática, por emitir um dado comando.



O comando `-c1` utilizado junto ao *ping*, significa que apenas um pacote ICMP será enviado. Do contrário, `sta1` irá enviar infinitos pacotes ICMP.

Sendo assim, como o comando *ping* necessita de um nó destino e este pode ser tanto um nome quanto um endereço IP, o destino `sta2` também pode ser substituído pelo seu endereço IP. Conforme pode ser visto a seguir, o endereço IP que identifica `sta2` é o `10.0.0.2/8`.

```

mininet-wifi> sta2 ip addr
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group
    <-- default qlen 1000
        link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
        inet 127.0.0.1/8 scope host lo

```

```

        valid_lft forever preferred_lft forever
inet6 ::1/128 scope host
    valid_lft forever preferred_lft forever
34: sta2-wlan0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc htb state
    UP group default qlen 1000
    link/ether 02:00:00:00:01:00 brd ff:ff:ff:ff:ff:ff
    inet 10.0.0.2/8 scope global sta2-wlan0
        valid_lft forever preferred_lft forever
    inet6 fe80::ff:fe00:100/64 scope link
        valid_lft forever preferred_lft forever

```

De forma alternativa, também é possível abrir diferentes terminais para cada nó e emitir comandos como se eles estivessem sendo emitidos diretamente em um computador, exatamente como ocorre no mundo real (ver Figura 2.2). Por exemplo, o comando abaixo abrirá dois terminais, um para `sta1` e outro para `sta2`, e havendo um terminal para cada nó, não será mais necessário informar quem é a origem, conforme a explicação do parágrafo anterior.

```
mininet-wifi> xterm sta1 sta2
```



O *xterm* pode não funcionar como esperado caso não exista ambiente gráfico habilitado em seu sistema operacional.

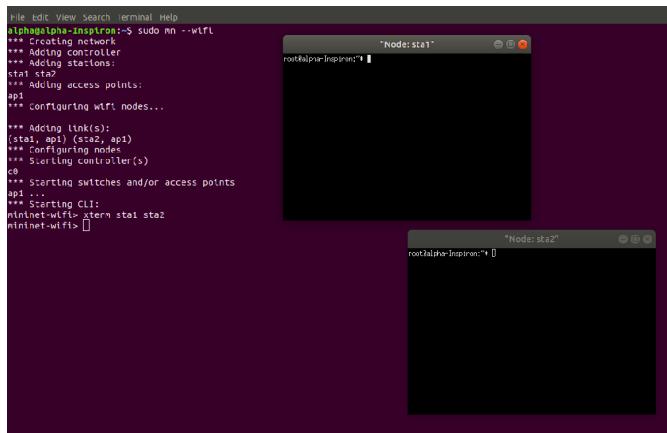


Figura 2.2: Executando o xterm.

A partir de agora, vamos realizar algumas ações corriqueiras e exclusivas para o meio sem fio. Para começar, vamos desconectar sta1 de ap1 e confirmar a desassociação através dos comandos abaixo:

```
mininet-wifi> sta1 iw dev sta1-wlan0 disconnect  
mininet-wifi> sta1 iw dev sta1-wlan0 link  
Not connected.
```

Então, vamos tentar um novo *ping* entre sta1 e sta2.

```
mininet-wifi> sta1 ping -c1 sta2  
PING 10.0.0.2 (10.0.0.2) 56(84) bytes of data.  
From 10.0.0.1 icmp_seq=1 Destination Host Unreachable  
  
--- 10.0.0.2 ping statistics ---  
1 packets transmitted, 0 received, +1 errors, 100% packet loss, time 0ms
```

Como é possível perceber, a estação sta1 não está mais associada ao ponto de acesso ap1, portanto, seria logicamente impossível realizar qualquer tipo de comunicação com sta2.

Agora, vamos conectar sta1 novamente ao ponto de acesso ap1 e confirmar a associação:

```
mininet-wifi> sta1 iw dev sta1-wlan0 connect new-ssid  
mininet-wifi> sta1 iw dev sta1-wlan0 link  
Connected to 02:00:00:00:02:00 (on sta1-wlan0)  
    SSID: new-ssid  
    freq: 2457  
    RX: 370 bytes (9 packets)  
    TX: 202 bytes (3 packets)  
    signal: -36 dBm  
    tx bitrate: 6.0 MBit/s  
  
    bss flags:      short-slot-time  
    dtim period:    2  
    beacon int:     100
```

E, então, tentamos um novo *ping* entre sta1 e sta2. O *ping* deverá ocorrer com sucesso, conforme demonstrado a seguir.

```
mininet-wifi> sta1 ping -c1 sta2  
PING 10.0.0.2 (10.0.0.2) 56(84) bytes of data.
```

---

```
64 bytes from 10.0.0.2: icmp_seq=1 ttl=64 time=1011 ms
--- 10.0.0.2 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 1011.206/1011.206/1011.206/0.000 ms
```

Uma outra operação bastante útil para redes WiFi é a realização do escaneamento (ou *scan*), que permite verificar quais pontos de acesso uma determinada estação consegue enxergar. Por exemplo, vamos supor que o SSID do ponto de acesso ap1 é desconhecido. Neste caso, o comando abaixo poderá ser utilizado e assim será possível visualizar o SSID do ponto de acesso em questão, além de outras informações.

```
mininet-wifi> sta1 iw dev sta1-wlan0 scan
BSS 02:00:00:00:02:00(on sta1-wlan0) -- associated
    TSF: 1534710096681871 usec (17762d, 20:21:36)
    freq: 2457
    beacon interval: 100 TUs
    capability: ESS ShortSlotTime (0x0401)
    signal: -36.00 dBm
    last seen: 0 ms ago
    Information elements from Probe Response frame:
    SSID: new-ssid
    Supported rates: 1.0* 2.0* 5.5* 11.0* 6.0 9.0 12.0 18.0
    DS Parameter set: channel 1
    ERP: Barker_Preamble_Mode
    Extended supported rates: 24.0 36.0 48.0 54.0
    Extended capabilities:
        * Extended Channel Switching
        * Operating Mode Notification
```

## 2.2 Customizando topologias

Assim como acontece com o Mininet, o Mininet-WiFi permite que diferentes topologias sejam criadas. Elas podem ser criadas através de simples comandos ou mesmo através de scripts escritos na linguagem de programação *Python*.

As topologias que podem ser criadas através de comandos são a *single* e *linear*. Para gerarmos essas 2 (duas) topologias será necessário encerrar o Mininet-WiFi, caso ele ainda estiver em execução.

```
mininet-wifi> exit
```

Então, vamos começar pela topologia *single*, topologia esta que consiste em 1 (um) ponto de acesso ap1 e N estações associadas a ele.

Por exemplo, o comando abaixo cria 4 (quatro) estações, 1 (um) ponto de acesso e 1 (um) controlador SDN, conforme ilustrado na Figura 2.3.

```
~/mininet-wifi$ sudo mn --wifi --topo single,4
```

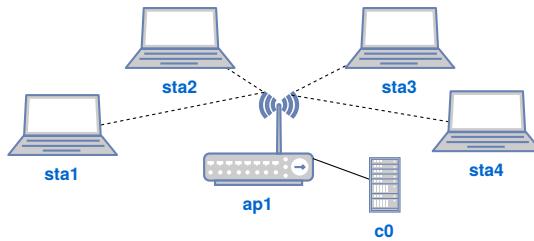


Figura 2.3: Topologia *single*.

Nesse momento, é possível testar a conectividade entre todos os nós através do comando pingall, conforme abaixo. A saída do comando foi filtrada de forma a apresentar apenas a saída partindo de sta1.

```
mininet-wifi> pingall
*** Ping: testing ping reachability
sta1 -> *** sta1 : ('ping -c1 10.0.0.2',)
PING 10.0.0.2 (10.0.0.2) 56(84) bytes of data.
64 bytes from 10.0.0.2: icmp_seq=1 ttl=64 time=0.170 ms

--- 10.0.0.2 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.170/0.170/0.170/0.000 ms
sta2 *** sta1 : ('ping -c1 10.0.0.3',)
PING 10.0.0.3 (10.0.0.3) 56(84) bytes of data.
64 bytes from 10.0.0.3: icmp_seq=1 ttl=64 time=0.121 ms

--- 10.0.0.3 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.121/0.121/0.121/0.000 ms
sta3 *** sta1 : ('ping -c1 10.0.0.4',)
PING 10.0.0.4 (10.0.0.4) 56(84) bytes of data.
64 bytes from 10.0.0.4: icmp_seq=1 ttl=64 time=0.129 ms

--- 10.0.0.4 ping statistics ---
```

```
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.129/0.129/0.129/0.000 ms
```

A outra topologia que pode ser criada através de comandos é a *linear*, que consiste em criar N pontos de acesso e N estações. Nesta topologia, cada estação é associada a um ponto de acesso e todos os pontos de acesso são conectados de forma linear.

Por exemplo, o comando abaixo cria 4 (quatro) pontos de acesso, 4 (quatro) estações e 1 (um) controlador SDN, conforme ilustrado na Figura 2.4.

```
~/mininet-wifi$ sudo mn --wifi --topo linear,4
```

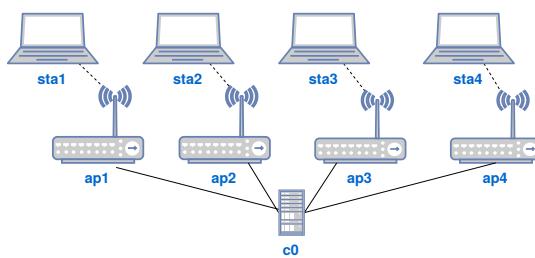


Figura 2.4: Topologia linear.

A customização de topologias, em contrapartida, requer que elas sejam criadas através de scripts que contenham todas as informações acerca da topologia e também a configuração dos nós. Dentro do diretório `</mininet-wifi/examples>` existe uma grande variedade de scripts que podem ser utilizados como referência para a criação de topologias customizadas.

Sempre é recomendado que os usuários analisem os scripts de maior interesse para a criação dos seus próprios. Ao longo deste livro iremos utilizar diversos scripts e a utilização deles certamente vai auxiliar na compreensão de como eles podem ser customizados.

### 2.3 Acessando informações dos nós

Agora, vamos aprender como obter algumas informações dos nós que compõem uma topologia. Para tanto, vamos criar a topologia mais simples e

acrescentar dois novos parâmetros: *position* e *wmediummd*. O parâmetro *position* vai definir posições iniciais para os nós; enquanto o parâmetro *wmediummd* irá habilitar o *wmediummd*, um simulador do meio sem fio que será apresentado em 2.6.2.

```
~/mininet-wifi$ sudo mn --wifi --link=wmediummd --position
```

Em seguida, experimente utilizar o comando *distance*, conforme abaixo:

```
mininet-wifi> distance sta1 sta2
The distance between sta1 and sta2 is 100.00 meters
```

Agora, verifique as posições de *sta1* e *sta2*. Observe que os eixos x, y e z estão separados por vírgula.

```
mininet-wifi> py sta1.position
[1.0, 0.0, 0.0]

mininet-wifi> py sta2.position
[101.0, 0.0, 0.0]
```

Como é possível perceber, posições iniciais foram definidas e o comando *distance* pôde servir de suporte para verificar a distância entre dois nós.

Neste momento uma pergunta certamente pode surgir: e se for necessário definir uma posição específica para um nó? Neste caso, existem duas possibilidades: através da CLI do Mininet-WiFi ou através de scripts. O exemplo abaixo traz o método *setPosition()*, que pode ser utilizado tanto na CLI quanto através de scripts.

```
mininet-wifi> py sta1.setPosition('10,0,0')
```

Observe que quando um método implementado no código-fonte do Mininet-WiFi é evocado pela CLI, o prefixo *py* deve ser sempre utilizado. Além do *setPosition()*, outros métodos serão apresentados ao longo deste livro.

Agora, vamos conferir a posição recém-definida:

```
mininet-wifi> py sta1.position
[10.0, 0.0, 0.0]
```

Neste caso, a posição está definida da seguinte forma: x=10, y=0 e z=0.

Várias outras informações sobre um determinado nó podem ser obtidas através de *node.wintfs*, conforme a seguir:

```
mininet-wifi> py sta1.wintfs
{0: <managed sta1-wlan0>}
mininet-wifi> py sta1.wintfs[0].txpower
14
mininet-wifi> py sta1.wintfs[0].ip
10.0.0.1/8
mininet-wifi> py sta1.wintfs[0].range
62
mininet-wifi> py sta1.wintfs[0].antennaGain
5
mininet-wifi> py sta1.wintfs[0].freq
2.412
mininet-wifi> py sta1.wintfs[0].mode
g
```

A forma *wintfs[0]* significa que a informação a ser obtida é da primeira interface sem fio. Caso o nó possuir múltiplas interfaces sem fio, o número zero poderá ser substituído pelo ID de uma outra interface. Outros atributos disponíveis podem ser conhecidos através do site oficial do Mininet-WiFi<sup>1</sup>.

## 2.4 OVSAp versus UserAP

O Mininet-WiFi suporta dois tipos de pontos de acesso que diferem basicamente no local onde eles são executados. O *OVSAp* ou *OVSKernelAP* é executado no espaço do kernel do sistema operacional, enquanto que o *UserAP* é executado no espaço do usuário. Além disso, algumas peculiaridades podem fazer você escolher entre um deles, como recursos suportados e desempenho.

Por exemplo, alguns recursos podem ser suportados por um e não por outro. Até pouco tempo atrás o *OVSAp* não suportava *meter tables*, tipo de tabela do protocolo OpenFlow responsável por questões relacionadas a QoS (*Quality of Service*), que foi incluído na versão 1.3 deste protocolo. Por outro lado, o *UserAP* já o suportava.

---

<sup>1</sup><http://mininet-wifi.github.io/>

Uma outra questão importante é a possibilidade de executar *switches* ou pontos de acesso em *network namespaces* particulares. Neste caso, o *OVS* ainda não suporta esse recurso nativamente, mas o *UserAP* sim. O que isso quer dizer? Experimente utilizar o comando a seguir.

```
~/mininet-wifi$ sudo mn --wifi
```

E, então, visualize as interfaces do ponto de acesso ap1.

```
mininet-wifi> ap1 ip link
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN mode
    ↳ DEFAULT group default qlen 1000
        link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
2: enp2s0: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc fq_codel
    ↳ state DOWN mode DEFAULT group default qlen 1000
        link/ether 84:7b:eb:fc:63:1a brd ff:ff:ff:ff:ff:ff
3: wlpis0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state
    ↳ UP mode DORMANT group default qlen 1000
        link/ether f8:da:0c:95:12:d3 brd ff:ff:ff:ff:ff:ff
4: docker0: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc noqueue
    ↳ state DOWN mode DEFAULT group default
        link/ether 02:42:04:ed:bc:24 brd ff:ff:ff:ff:ff:ff
5: br-7e51375c6c71: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc
    ↳ noqueue state DOWN mode DEFAULT group default
        link/ether 02:42:6f:43:07:ee brd ff:ff:ff:ff:ff:ff
6: hwsim0: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN mode
    ↳ DEFAULT group default qlen 1000
        link/ieee802.11/radiotap 12:00:00:00:00:00 brd ff:ff:ff:ff:ff:ff
9: ap1-wlan1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc tbf master
    ↳ ovs-system state UP mode DEFAULT group default qlen 1000
        link/ether 02:00:00:00:02:00 brd ff:ff:ff:ff:ff:ff
10: ovs-system: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN mode
    ↳ DEFAULT group default qlen 1000
        link/ether ee:99:70:bb:39:89 brd ff:ff:ff:ff:ff:ff
11: ap1: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN mode DEFAULT
    ↳ group default qlen 1000
        link/ether 0a:94:5d:2c:8b:40 brd ff:ff:ff:ff:ff:ff
```

Observe que foi possível obter como resposta um grande número de interfaces de rede, incluindo aquelas que na prática não fazem parte do ponto de acesso ap1, como interfaces sem fio e com fio do computador ao qual o Mininet-WiFi está em execução. Esse é o comportamento observado quando o *OVS* está sendo utilizado, que é o tipo de *switch* ou ponto de acesso definido como padrão para o Mininet-WiFi.

Agora, vamos verificar como se comporta o *UserAP*. Para tanto, execute o comando abaixo.

```
~/mininet-wifi$ sudo mn --wifi --ap user --innamespace
```



O parâmetro *--innamespace* não foi utilizado com o OVS, pois o OVS ainda não suporta este comando. É ele que é responsável por fazer com que o nó seja executado em seu próprio *network namespace*, em vez do *network namespace* raiz.

Então, verifique as interfaces do ponto de acesso ap1.

```
mininet-wifi> ap1 ip link
1: lo: <LOOPBACK> mtu 65536 qdisc noop state DOWN mode DEFAULT group
    ↳ default qlen 1000
        link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
2: ap1-eth0@if46: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue
    ↳ state UP mode DEFAULT group default qlen 1000
        link/ether de:93:af:2c:68:a0 brd ff:ff:ff:ff:ff:ff link-netnsid 0
45: ap1-wlan1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc tbf state
    ↳ UP mode DEFAULT group default qlen 1000
        link/ether 02:00:00:00:02:00 brd ff:ff:ff:ff:ff:ff
```

Como é possível perceber o número de interfaces de rede caiu consideravelmente. Já era esperada a presença da interface de *loopback* e também da interface ap1-wlan1, que é a interface sem fio do ponto de acesso ap1. A única interface que poderia ser considerada como não esperada, seria a interface ap1-eth0, que é a interface utilizada para conexão com o controlador SDN.

Uma outra questão relevante entre o *OVSAP* e o *UserAP* é a performance. A performance do *UserAP* tem piorado significamente nas versões mais recentes do kernel do sistema operacional Linux. O motivo? Confesso não ter uma resposta para essa pergunta. Porém, convido-o a verificar essa questão na prática.

O comando a seguir irá executar o Mininet com o OVS e testar a largura de banda entre dois nós h1 e h2.

```
~/mininet-wifi$ sudo mn --test iperf  
*** Iperf: testing TCP bandwidth between h1 and h2  
*** Results: ['40.1 Gbits/sec', '40.0 Gbits/sec']
```

Já o comando abaixo executa o Mininet com o *UserSwitch* e testa a largura de banda entre os mesmos nós h1 e h2.

```
~/mininet-wifi$ sudo mn --switch=user --test iperf  
*** Iperf: testing TCP bandwidth between h1 and h2  
*** Results: ['171 Mbits/sec', '172 Mbits/sec']
```



Caso não for possível executar o comando anterior, será necessário iniciar um controlador SDN em um outro terminal e substituir o parâmetro `--test iperf` por `--controller=remote`. Então, após iniciar o controlador, execute o *iperf* dentro da CLI do Mininet-WiFi da seguinte forma: `iperf h1 h2`.

Você pode estar se perguntando, por que *UserSwitch*? O *UserAP* no Mininet-WiFi foi estendido do *UserSwitch* do Mininet. Então, na prática, eles são o mesmo *switch* ou ponto de acesso. Mas voltando ao resultado, percebeu a diferença entre eles? O *UserSwitch* tem desempenho bastante inferior se comparado ao *OVS*.

Ainda com relação ao *UserAP*, uma implementação de ponto de acesso deste tipo é o *Basic OpenFlow Software Switch (BOFUSS)*, sucessor do *ofsoftswitch13*<sup>2</sup>. Ele vem sendo utilizado em várias pesquisas já faz um bom tempo e certamente você pode querer utilizá-lo em algum momento. O *BOFUSS* promete eliminar boa parte dos problemas relacionados à performance.



Para instalar o *BOFUSS* basta executar o comando `sudo util/install.sh -3f` a partir do diretório raiz do Mininet-WiFi.

<sup>2</sup><https://github.com/CPqD/ofsoftswitch13>

## 2.5 Utilizando interfaces gráficas

Para aqueles que não conhecem a linguagem de programação *Python* ou seja um iniciante no Mininet-WiFi, atualmente existem 2 (duas) opções para a criação de scripts em *Python* com o suporte de interfaces gráficas: através do *Visual Network Descriptor (VND)* e do *MiniEdit*.

### 2.5.1 Visual Network Descriptor



**Pré-requisito(s):** servidor web, php, *flash player*, *visual network descriptor*

O *Visual Network Descriptor*, ou simplesmente VND, é uma ferramenta fruto de um trabalho de mestrado que é capaz de gerar scripts em *Python* para o Mininet-WiFi através de um navegador WEB. Escrito predominantemente na linguagem de programação *Flex*, o VND também inclui algumas instruções em PHP e XML.

Utilizar o VND é relativamente simples. Primeiro é necessário fazer clone do código-fonte disponível em <https://github.com/ramonfontes/vnd> e seguir os passos de instalação disponíveis na página do código-fonte. Em geral, é necessário ter um servidor Web, PHP e o *Flash Player* instalados. Em seguida, basta acessá-lo através de um navegador Web preferido. Se tudo der certo, uma tela similar a apresentada na Figura 2.5 deverá aparecer.

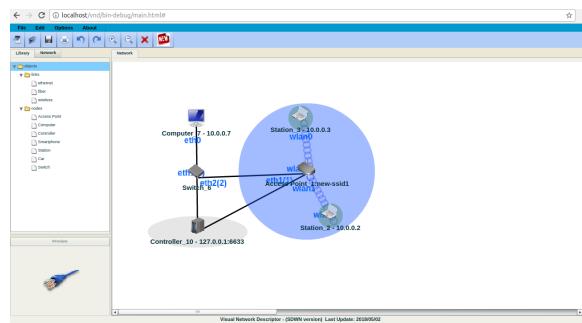
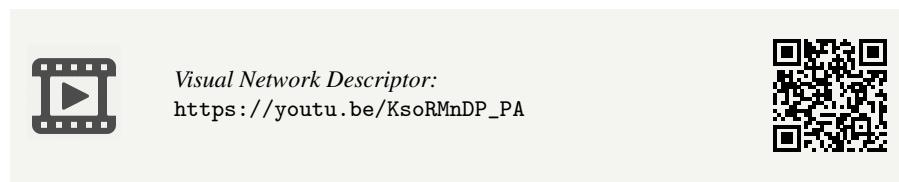


Figura 2.5: Visual Network Descriptor.

Com o VND aberto já é possível utilizar o cursor do mouse para selecionar os nós que se deseja incluir na topologia e suas respectivas conexões. Também é possível realizar configurações para os nós e salvar a topologia para uso posterior. Para gerar scripts para o Mininet-WiFi, basta seguir o menu *File->Export->Export to Mininet-WiFi*. Um arquivo padrão com extensão .sh será criado, porém, ele consiste de instruções em *Python* e poderá ser executado como se fosse um arquivo em *Python*.

Por exemplo, se houver um script com nome <*minhatopologia.sh*>, ele poderá ser executado conforme abaixo:

```
~/mininet-wifi$ sudo python minhatopologia.sh
```



### 2.5.2 MiniEdit



**Pré-requisito(s):** apenas script(s)

Uma outra alternativa para a criação de topologias com o suporte de gráficos é o *MiniEdit*. Escrito na linguagem de programação *Python*, o *MiniEdit* foi inicialmente desenvolvido para o Mininet e tem sido constantemente aprimorado para o Mininet-WiFi. O objetivo é fazer com que todos os recursos suportados pelo Mininet-WiFi estejam disponíveis no *MiniEdit*.

O *MiniEdit* tem uma interface de usuário bastante simples que apresenta uma tela com uma linha de ícones de ferramentas no lado esquerdo da janela e uma barra de menu na parte superior. Ele já vem incluso no código do Mininet-WiFi.

Para utilizá-lo basta executar o arquivo <*miniedit.py*> disponível no diretório <*examples*>.

```
~/mininet-wifi$ sudo python examples/miniedit.py
```

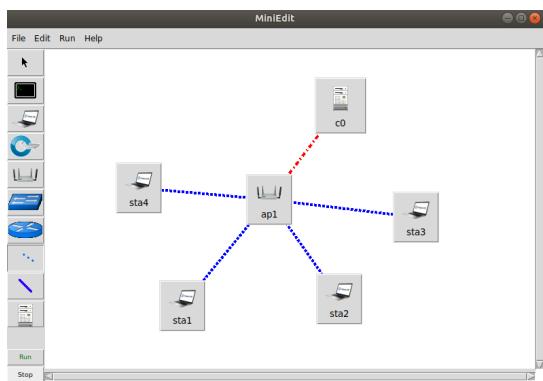


Figura 2.6: Miniedit.

Após executá-lo, uma tela similar a apresentada na Figura 2.6 deverá aparecer. Nela é possível adicionar os nós suportados pelo Mininet-WiFi e suas respectivas configurações, além de seus links, claro.

Na versão atual do *MiniEdit*, diferentes tipos de cenários já são suportados, como a criação de redes *adhoc*, *mesh*, *WiFi-Direct*, utilização do protocolo Radius, autenticação WPA, dentre outros.

Entre o *MiniEdit* e o VND, qual seria então a melhor alternativa? Por já estar incluso no Mininet-WiFi, certamente o *MiniEdit* tende a ser o preferido. Além disso, há a tendência de que o VND seja descontinuado aos poucos.



*MiniEdit and Mininet-WiFi:*  
<https://youtu.be/j4JS4xxCrCA>



### 2.5.3 Visualizando gráficos 2D e 3D

A visualização de topologias através de gráficos é um outro recurso que pode ser utilizado no Mininet-WiFi. É possível gerar tanto gráficos 2D quanto 3D. Existem situações em que gráficos 3D são muito úteis, como em pesquisas

que envolvem drones e satélites, uma vez que a representação de diferentes níveis de altitudes pode ser necessária.

Assim, dada a sua importância, vamos compreender então como é possível gerar gráficos 2D e 3D no Mininet-WiFi. Inicialmente vamos compreender como gerar os dois tipos de gráficos através do terminal.

O comando abaixo irá gerar uma topologia em 2D.

```
~/mininet-wifi$ sudo mn --wifi --plot --position
```

Enquanto que comando a seguir irá gerar uma topologia em 3D.

```
~/mininet-wifi$ sudo mn --wifi --plot3d --position
```

Em sua totalidade, os scripts disponíveis no diretório *</examples>*, diretório do Mininet-WiFi onde é possível encontrar uma ampla variedade de scripts prontos para serem executados, geram gráficos em 2D. Caso o usuário opte por gerar gráficos em 3D, basta fazer uma simples alteração no código.

Por exemplo, tomemos como exemplo o arquivo *<position.py>*, disponível no diretório *</examples>*. Este arquivo possui o seguinte conteúdo:

```
net.plotGraph(max_x=100, max_y=100)
```

Como pode ser visto, apenas os eixos *x* e *y* foram definidos e o resultado do gráfico será algo similar ao apresentado na Figura 2.7. Logo, para gerar gráficos em 3D basta adicionar o eixo *z*, que resultará em algo similar ao apresentado na Figura 2.8.

```
net.plotGraph(max_x=100, max_y=100, max_z=100)
```

De forma opcional, também podem ser definidos valores mínimos dos eixos *x*, *y* e *z*, caso valores diferentes de zero forem necessários.

```
net.plotGraph(min_x=10, min_y=10, min_z=10, max_x=100, max_y=100,  
              max_z=100)
```

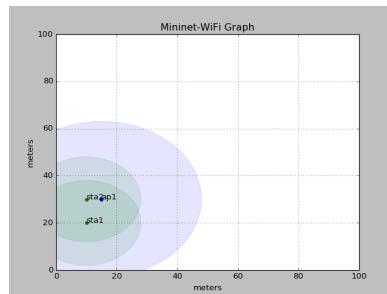


Figura 2.7: Gráfico em 2D.

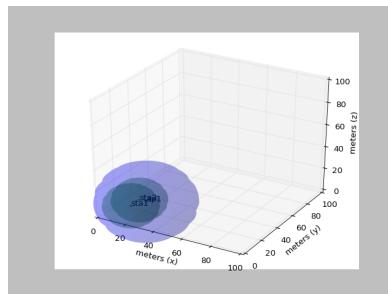


Figura 2.8: Gráfico em 3D.



Os gráficos gerados pelo Mininet-WiFi são suportados pelo *matplotlib*, uma biblioteca de visualização de dados disponível para a linguagem de programação *python*.



*Building 3D Graphic:*  
<https://youtu.be/lMkIVOYBTss>



## 2.6 Emulação do meio sem fio

A emulação do meio sem fio no Mininet-WiFi pode ser feito de duas formas: através do TC<sup>3</sup> ou Wmediumd<sup>4</sup>. Vamos então compreender como utilizá-los e quais as diferenças entre eles.

### 2.6.1 TC (*Traffic Control*)

Se o Mininet-WiFi estiver em execução, encerre-o. Então, inicie-o novamente através do comando a seguir.

```
~/mininet-wifi$ sudo mn --wifi --position
```

Agora, visualize as informações sobre o TC dentro da CLI do Mininet-WiFi:

<sup>3</sup>[https://en.wikipedia.org/wiki/Tc\\_\(Linux\)](https://en.wikipedia.org/wiki/Tc_(Linux))

<sup>4</sup><https://github.com/ramonfontes/wmediumd/>

```
mininet-wifi> ap1 tc qdisc
qdisc noqueue 0: dev lo root refcnt 2
qdisc pfifo_fast 0: dev enp2s0 root refcnt 2 bands 3
priomap 1 2 2 2 1 2 0 0 1 1 1 1 1 1 1 1
qdisc noqueue 0: dev wlp1s0 root refcnt 2
qdisc noqueue 0: dev docker0 root refcnt 2
qdisc tbf 2: dev ap1-wlan1 root refcnt 5 rate 54Mbit burst 14998b lat 1.0ms
qdisc pfifo 10: dev ap1-wlan1 parent 2:1 limit 1000p
```

Destacamos então a saída que nos interessa:

```
qdisc tbf 2: dev ap1-wlan1 root refcnt 5
rate 54Mbit burst 14998b lat 1.0ms
```

Em linhas gerais, esta saída instrui ao ponto de acesso ap1 a limitar a largura de banda total em 54 Mbits/s, que representa o valor máximo que o padrão IEEE 802.11g pode nominalmente suportar. Diante desta informação fica mais fácil compreender como o TC funciona.

Considerando que valores do TC são aplicados em estações e considerando também os modelos de propagação suportados pelo Mininet-WiFi, para uma distância  $d$  entre ponto de acesso e estação, há sempre um valor de sinal recebido, que pode variar de acordo com o modelo de propagação escolhido. Então, à medida que o nó se encontra em uma nova posição, um novo valor para  $d$  é calculado e a partir daí obtém-se o sinal recebido e o valor de largura de banda que será aplicado pelo TC na interface sem fio do nó.

Na prática, o valor aplicado pelo TC no ap1 não muda, o que muda são os valores aplicados nas interfaces das estações. Sendo assim, em um ambiente infraestruturado sempre teremos dois pontos de referência para o cálculo de  $d$ : a estação e o ponto de acesso. Duas estações podem estar associadas ao mesmo ponto de acesso e diferentes valores de largura de banda podem ser atribuídos para suas interfaces. Um valor para sta1 e outro para sta2.

Agora, imaginemos uma rede sem fio *adhoc* com três estações. Sendo uma rede sem fio *adhoc*, podemos dizer que essa rede consiste de um ambiente não-infraestruturado, ou seja, que não tem a presença do concentrador principal, o ponto de acesso. Neste tipo de rede as três estações podem associar-se entre si, onde, por exemplo, sta1 pode manter associação com sta2 e sta3. Porém,

eles só possuem uma interface sem fio, e, nesse caso, é particularmente difícil fazer o controle pelo TC.

Qual seria o ponto de referência para o cálculo do  $d$  para `sta1?` `sta2` ou `sta3`? Diferentemente de uma rede infraestruturada, onde temos dois pontos de referência para o cálculo do  $d$ , a estação e o ponto de acesso, em uma rede não-infraestruturada isso não ocorre. Sendo assim, cenários como *adhoc*, *mesh*, dentre outros, exigem a utilização do *Wmediumd*, que implementa um simulador de meio sem fio ideal para estes tipos de redes.

### 2.6.2 Wmediumd

O módulo responsável pela virtualização das placas de rede WiFi no Mininet-WiFi, `mac80211_hwsim`, usa o mesmo meio virtual para todos os nós sem fio. Isso significa que todos os nós estão internamente ao alcance um do outro e podem ser descobertos através de um escaneamento, como já fizemos com o *iw* anteriormente. Se as interfaces sem fio precisam ser isoladas umas das outras, a utilização do *Wmediumd* é recomendada.

O *Wmediumd* já vinha sendo desenvolvido desde 2011<sup>5</sup>, mas somente em 2017 ele foi integrado ao Mininet-WiFi, graças a Patrick Große<sup>6</sup>, desenvolvedor responsável por criar a primeira extensão do *Wmediumd* para o Mininet-WiFi.

Diferente do TC que limita a largura de banda na interface, o *Wmediumd* utiliza de uma tabela de sinais<sup>7</sup> e gerencia o isolamento das interfaces em tempo real à medida que dados trafegam na rede.

### 2.6.3 TC versus Wmediumd na prática

Inicie o Mininet-WiFi com o comando abaixo.

```
~/mininet-wifi$ sudo mn --wifi --topo single,3 --position --plot
```

Esse comando irá criar 3 (três) estações que irão estar associadas ao ponto de acesso `ap1`. O parâmetro `--plot` abrirá um gráfico da topologia. Mais detalhes

---

<sup>5</sup>[https://github.com/jlopedex/mac80211\\_hwsim](https://github.com/jlopedex/mac80211_hwsim)

<sup>6</sup><https://github.com/patgrosse>

<sup>7</sup>[https://github.com/ramonfontes/wmediumd/blob/mininet-wifi/tests/signal\\_table\\_ieee80211ax](https://github.com/ramonfontes/wmediumd/blob/mininet-wifi/tests/signal_table_ieee80211ax)

sobre esse parâmetro serão vistos mais adiante.

Agora, verifique o nível de sinal recebido por `sta1` em relação ao ponto de acesso `ap1` através do comando `scan`.

```
mininet-wifi> sta1 iw dev sta1-wlan0 scan
BSS 02:00:00:00:03:00(on sta1-wlan0) -- associated
    TSF: 1536705774286475 usec (17785d, 22:42:54)
    freq: 2412
    beacon interval: 100 TUs
    capability: ESS ShortSlotTime (0x0401)
    signal: -36.00 dBm
    last seen: 0 ms ago
    Information elements from Probe Response frame:
    SSID: my-ssid
    Supported rates: 1.0* 2.0* 5.5* 11.0* 6.0 9.0 12.0 18.0
    DS Parameter set: channel 1
    ERP: Barker_Preamble_Mode
    Extended supported rates: 24.0 36.0 48.0 54.0
    Extended capabilities:
        * Extended Channel Switching
        * Operating Mode Notification
```

Verifique também na opção `node.wintfs`:

```
mininet-wifi> py sta1.wintfs[0].rss
[-66.0]
```

Como é possível notar, existe uma diferença no sinal recebido via `iw scan` e via `node.wintfs`. Pelo `iw`, o sinal obtido foi de -36 dBm, enquanto que via `node.wintfs` -66 dBm. Enquanto o TC estiver sendo utilizado, não será possível obter nenhuma informação atualizada acerca do nível de sinal ou qualquer outra informação que dela dependa através de comandos de rede, como o `iw`. Talvez uma das poucas informações úteis é a de verificar se de fato a estação `sta1` está associada ao ponto de acesso `ap1`.

O `iw link` também pode ser utilizado para tal, conforme abaixo:

```
mininet-wifi> sta1 iw dev sta1-wlan0 link
Connected to 02:00:00:00:03:00 (on sta1-wlan0)
SSID: my-ssid
freq: 2412
RX: 12486 bytes (236 packets)
TX: 805 bytes (9 packets)
```

```

signal: -36 dBm
tx bitrate: 18.0 MBit/s

bss flags:      short-slot-time
dtim period:    2
beacon int:     100

```

Agora, vamos mover `sta1` e testar novamente o nível de sinal através de um novo escaneamento.

```

mininet-wifi> py sta1.setPosition('250,250,0')
mininet-wifi> sta1 iw dev sta1-wlan0 scan
BSS 02:00:00:00:03:00(on sta1-wlan0)
    TSF: 1536706071142532 usec (17785d, 22:47:51)
    freq: 2412
    beacon interval: 100 TUs
    capability: ESS ShortSlotTime (0x0401)
    signal: -36.00 dBm
    last seen: 0 ms ago
    Information elements from Probe Response frame:
    SSID: my-ssid
    Supported rates: 1.0* 2.0* 5.5* 11.0* 6.0 9.0 12.0 18.0
    DS Parameter set: channel 1
    ERP: Barker_Preamble_Mode
    Extended supported rates: 24.0 36.0 48.0 54.0
    Extended capabilities:
        * Extended Channel Switching
        * Operating Mode Notification

```

Surpreendentemente o sinal se manteve o mesmo do anterior, mesmo alterando a posição de `sta1`. Na verdade, `ap1` não deveria nem aparecer no escaneamento, uma vez que `sta1` não está mais sob o raio de alcance do ponto de acesso `ap1`. Isso mostra que realmente o TC não possui nenhuma noção do meio sem fio, pois mesmo a estação não estando mais sob o raio de alcance do ponto de acesso `ap1`, ele, o nó `sta1`, ainda é capaz de enxergá-lo.

Agora, vamos repetir o que fizemos anteriormente, mas utilizando o *Wmediumd*.

```

~/mininet-wifi$ sudo mn --wifi --topo single,3 --link=wmediumd --position
↪ --plot

```

Então, realizamos o escaneamento a partir de `sta1`, alteramos sua posição e mais uma vez repetimos o escaneamento.

```
mininet-wifi> sta1 iw dev sta1-wlan0 scan
BSS 02:00:00:00:03:00(on sta1-wlan0) -- associated
    TSF: 1536709235310507 usec (17785d, 23:40:35)
    freq: 2412
    beacon interval: 100 TUs
    capability: ESS ShortSlotTime (0x0401)
    signal: -67.00 dBm
    last seen: 0 ms ago
    Information elements from Probe Response frame:
    SSID: my-ssid
    Supported rates: 1.0* 2.0* 5.5* 11.0* 6.0 9.0 12.0 18.0
    DS Parameter set: channel 1
    ERP: Barker_Preamble_Mode
    Extended supported rates: 24.0 36.0 48.0 54.0
    Extended capabilities:
        * Extended Channel Switching
        * Operating Mode Notification
mininet-wifi> py sta1.setPosition('250,250,0')
mininet-wifi> sta1 iw dev sta1-wlan0 scan
```

Como é possível perceber, inicialmente o sinal percebido por `sta1` foi de -67 dBm. Porém, quando ele saiu do raio de alcance do ponto de acesso `ap1`, houve uma mudança esperada no resultado. Além de retornar um valor de sinal esperado no primeiro momento, no segundo não foi possível visualizar o ponto de acesso `ap1`, uma vez que o sinal deste não fora capaz de chegar até `sta1`.



Com o `Wmediumd` não é recomendável a utilização do `node.wintfs`, uma vez que algumas implementações do `Wmediumd` para o cálculo do sinal não são transferidas para o Mininet-WiFi. Nesse caso, é recomendável a utilização de aplicações como `iw` ou `iwconfig`.

## 2.7 Modelos de propagação

Modelos de propagação são modelos matemáticos que normalmente são utilizados por simuladores e emuladores de redes sem fio para tentar imitar o comportamento do meio sem fio. Na literatura existem diversos modelos de propagação que foram propostos de forma a suportar diferentes características do meio sem fio, como diferentes tipos de ambientes (interno e externo), atenuação do sinal, interferência, etc.

O Mininet-WiFi atualmente suporta os seguintes modelos de propagação: *Friis Propagation Loss Model*, *Log-Distance Propagation Loss Model* (padrão do Mininet-WiFi), *Log-Normal Shadowing Propagation Loss Model*, *International Telecommunication Union (ITU) Propagation Loss Model* e *Two-Ray Ground Propagation Loss Model*.

A escolha correta do modelo de propagação faz muita diferença. Por exemplo, uma das variáveis utilizadas nos modelos de propagação é o expoente (ou *exponent*). O expoente é quem vai instruir o modelo de propagação acerca do ambiente de testes, ou seja, se é um ambiente interno ou externo, ou mesmo se é um ambiente livre de interferências ou não.

Especificar um modelo de propagação é uma tarefa simples. Os vários scripts de exemplo do Mininet-WiFi certamente servirão de apoio nessa tarefa, em especial o script `<propagationModel.py>`. Nele é possível encontrar a função responsável por definir o modelo de propagação e seus parâmetros.

Para demonstrar como o modelo de propagação pode afetar a configuração dos nós que compõem a rede, vamos executar o script a seguir.

```
~/mininet-wifi$ sudo python examples/propagationModel.py
```

Com o modelo de propagação pré-definido, podemos observar que o nível de sinal percebido por `sta1` girou em torno de -79 dBm.

```
mininet-wifi> py sta1.wintfs[0].rss
[-79.0]
```

Por outro lado, após configurar o modelo de propagação *free space*, o nível de sinal aumentou para algo em torno de -47 dBm. Se você não encontrou os valores -79 dBm e -47 dBm, não se preocupe. O que importa é o valor obtido após configurar o modelo de propagação. Este deverá ser superior ao anteriormente observado.

O modelo de propagação pode ser modificado da seguinte forma:

de:

```
net.setPropagationModel(model="logDistance", exp=4)
```

para:

```
net.setPropagationModel(model="friis")
```

E, então, checa-se o RSSI após executar o script modificado.

```
mininet-wifi> py sta1.wintfs[0].rss
[-47.0]
```

Outra mudança relevante que é possível notar está relacionada ao raio de alcance do ponto de acesso. Certamente o novo raio de alcance do ponto de acesso ap1 agora é bem maior que aquele observado anteriormente.



O arquivo `<propagationModel.py>` não considera o `Wmediumd` em sua configuração, por isso, caso for necessário obter o nível de sinal, ele sempre deverá ser obtido através da utilização do comando `node.wintfs`

O novo valor no raio de alcance evidencia a importância na escolha do modelo de propagação correto para o cenário em que se precisa trabalhar. O novo nível de sinal, que é superior ao anterior, também evidencia o comportamento do modelo de propagação *free space*, uma vez que o modelo de propagação *free space* não considera nenhum tipo de interferência ou qualquer barreira que possa fazer com que o sinal seja atenuado.

É importante destacar que além do expoente discutido anteriormente, para cada modelo existem alguns outros parâmetros que podem ser únicos ou não. Você poderá obter mais informações sobre os modelos suportados e seus parâmetros através do manual do Mininet-WiFi, disponível na página do seu código-fonte.

### 2.7.1 Provendo mais realismo

Alguns modelos de propagação não possuem variação do sinal no decorrer do tempo. Isso significa que se verificarmos o nível de sinal de um determinado nó, o nível de sinal percebido sempre será o mesmo. Porém, como sabemos, o

meio sem fio não é constante e muitos fatores podem afetar o nível de sinal percebido.

Portanto, em casos onde a variação no nível de sinal é importante, até pra representar com maior fidelidade o que ocorre no mundo real, faz-se necessário definir um coeficiente de atenuação, ou *fading\_coefficient*, que em termos práticos produz atenuação do sinal no decorrer do tempo.

Para verificar o efeito do *fading* na prática, vamos executar o código abaixo.

```
~/mininet-wifi$ sudo python examples/wmediumd_interference.py
```

Em seguida, já é possível verificar a variação do sinal percebido por uma dada estação através do *iw*, conforme abaixo. Perceba que como o script *wmediumd\_interference.py* utiliza o *Wmediumd*, o nível de sinal pode ser obtido através do *iw* ou mesmo *iwconfig*.

```
mininet-wifi> sta1 iw dev sta1-wlan0 link
Connected to 02:00:00:00:03:00 (on sta1-wlan0)
SSID: new-ssid
freq: 5180
RX: 6901 bytes (124 packets)
TX: 712 bytes (8 packets)
signal: -65 dBm
tx bitrate: 12.0 MBit/s

bss flags:      short-slot-time
dtim period:    2
beacon int:     100

mininet-wifi> sta1 iw dev sta1-wlan0 link
Connected to 02:00:00:00:03:00 (on sta1-wlan0)
SSID: new-ssid
freq: 5180
RX: 8827 bytes (165 packets)
TX: 800 bytes (9 packets)
signal: -62 dBm
tx bitrate: 12.0 MBit/s

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

Como é possível notar, o nível de sinal percebido por **sta1** era inicialmente -65 dBm e passou para -62 dBm em um outro momento. É esperado que essa

variação ocorra sempre que for verificado o nível de sinal recebido. Essa é uma variação que ocorre de forma aleatória, mas respeitando o intervalo definido no parâmetro *fadding*.



Tente alterar o valor do *fadding* e verifique o resultado. Quanto maior o valor do *fadding*, maior será a variação do sinal.



Todos os modelos de propagação suportados pelo Mininet-WiFi podem ser encontrados no arquivo `<mn_wifi/propagationModels.py>`. Caso você queira implementar novos modelos de propagação, é neste arquivo que você deve incluir os novos modelos.

## 2.8 Relação distância *versus* sinal recebido

Além da largura de banda, a variação da distância impactará também no nível de sinal recebido dos nós. Obviamente, quanto mais distantes origem e destino pior deverá ser o sinal percebido. Isso se dá devido a atenuação do sinal.

Já vimos na seção 2.1 como é possível visualizar o nível de sinal percebido por um nó. Logo, vamos utilizar do mesmo comando para observar o nível de sinal percebido para diferentes posições. Para tanto, iremos utilizar como referência o script `<wmediumd_interference.py>`.

```
~/mininet-wifi$ sudo python examples/wmediumd_interference.py
```

Em seguida, verificamos o nível de sinal recebido.

```
mininet-wifi> sta1 iw dev sta1-wlan0 link
Connected to 02:00:00:00:03:00 (on sta1-wlan0)
    SSID: new-ssid
    freq: 5180
    RX: 9142 bytes (184 packets)
    TX: 88 bytes (2 packets)
    signal: -64 dBm
    tx bitrate: 6.0 MBit/s

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

Agora, vamos utilizar o comando `distance` para observar a distância entre `sta1` e `ap1`.

```
mininet-wifi> distance sta1 ap1
The distance between sta1 and ap1 is 11.18 meters
```

Como é possível observar, a distância entre eles é de pouco mais de 11 metros e o nível de sinal percebido por `sta1` foi de -64 dBm. Então, vamos alterar a posição de `sta1` de forma a aumentar a distância em relação ao ponto de acesso `ap1` e verificar novamente o nível de sinal recebido por `sta1`.

```
mininet-wifi> py sta1.setPosition('40,40,0')
mininet-wifi> distance sta1 ap1
The distance between sta1 and ap1 is 26.93 meters
mininet-wifi> sta1 iw dev sta1-wlan0 link
Connected to 02:00:00:00:03:00 (on sta1-wlan0)
    SSID: new-ssid
    freq: 5180
    RX: 176746 bytes (4379 packets)
    TX: 1668 bytes (19 packets)
    signal: -79 dBm
    tx bitrate: 18.0 MBit/s

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

Como podemos observar, após alterar a posição a distância aumentou e consequentemente o nível de sinal caiu de -64 dBm para -79 dBm. Essa pode ser uma conclusão simples e óbvia, porém, os passos que acabamos de executar podem servir de suporte no processo de ensino e aprendizagem.



- Ramon dos Reis Fontes, Mohamed Mahfoudi, Walid Dabbous, Thierry Turletti, Christian Esteve Rothenberg. *How far can we go? Towards Realistic Software-Defined Wireless Networking Experiments*. In The Computer Journal (Special Issue on Software Defined Wireless Networks), 2017.

## 2.9 Modificando o *bitrate*

*Bitrate* diz respeito à taxa de transmissão de dados suportada para um dado momento. Dispositivos WiFi são capazes de ajustar seu esquema de modulação e codificação (*Modulation and Coding Scheme* - MCS) de acordo com

o nível de sinal recebido. Na prática, quanto mais complexo o esquema de modulação, mais bits podem ser transmitidos, em contrapartida eles se tornam mais sensíveis a interferência e ruído, e consequentemente, requerem um canal mais limpo.

A ferramenta que iremos utilizar para a modificação das taxas de bits é o *iw*. Para tanto, considere utilizar novamente como referência o arquivo <*wmediumd\_interference.py*>

```
~/mininet-wifi$ sudo python examples/wmediumd_interference.py
```

Em seguida, vamos fazer alguns testes simples e observar a diferença na largura de banda obtida para diferentes valores de *bitrate*.

Primeiro, execute o *iperf* sem alterar os valores de *bitrate*.

```
mininet-wifi> iperf sta1 sta2
*** Iperf: testing TCP bandwidth between sta1 and sta2
*** Results: ['14.3 Mbits/sec', '14.4 Mbits/sec']
```

E, então, observe o *bitrate* atual. Como é possível ver logo abaixo, o valor do *bitrate* foi de 54 Mbits/s.

```
mininet-wifi> sta1 iw dev sta1-wlan0 link
Connected to 02:00:00:00:03:00 (on sta1-wlan0)
    SSID: new-ssid
    freq: 5180
    RX: 581186 bytes (7475 packets)
    TX: 19278284 bytes (12610 packets)
    signal: -64 dBm
    tx bitrate: 54.0 MBit/s

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

Agora, altere o *bitrate* e faça um novo teste de largura de banda.

```
mininet-wifi> sta1 iw dev sta1-wlan0 set bitrates legacy-5 6 9
mininet-wifi> iperf sta1 sta2
*** Iperf: testing TCP bandwidth between sta1 and sta2
*** Results: ['5.87 Mbits/sec', '5.93 Mbits/sec']
mininet-wifi> sta1 iw dev sta1-wlan0 link
```

```

Connected to 02:00:00:00:03:00 (on sta1-wlan0)
    SSID: new-ssid
    freq: 5180
    RX: 840551 bytes (12506 packets)
    TX: 23301226 bytes (15251 packets)
    signal: -64 dBm
    tx bitrate: 9.0 MBit/s

    bss flags:      short-slot-time
    dtim period:    2
    beacon int:     100

```

Observe que o *bitrate* ficou limitado a 9 Mbits/s e a medição através do *iperf* caiu para menos de 6 Mbits/s.

Finalmente, façamos mais uma modificação no *bitrate* e mensuremos a largura de banda mais uma vez.

```

mininet-wifi> sta1 iw dev sta1-wlan0 set bitrates legacy-5 6
mininet-wifi> iperf sta1 sta2
*** Iperf: testing TCP bandwidth between sta1 and sta2
*** Results: ['4.37 Mbits/sec', '4.44 Mbits/sec']
mininet-wifi> sta1 iw dev sta1-wlan0 link
Connected to 02:00:00:00:03:00 (on sta1-wlan0)
    SSID: new-ssid
    freq: 5180
    RX: 1044693 bytes (16503 packets)
    TX: 26353234 bytes (17256 packets)
    signal: -64 dBm
    tx bitrate: 6.0 MBit/s

    bss flags:      short-slot-time
    dtim period:    2
    beacon int:     100

```

Mais uma vez a largura de banda disponível caiu e o *bitrate* ficou limitado a 6 Mbits/s, conforme foi definido no comando.

Um outro teste interessante é verificar a diferença na taxa de transferência suportada por diferentes padrões de funcionamento WiFi. Por exemplo, como o script está configurado para operar no padrão IEEE 802.11a e este suporta até 54 Mbits/s, foi possível obter os 14Mbits/s no teste anterior. Por outro lado, um outro padrão, o IEEE 802.11b, suportaria até 11 Mbits/s.

Para um teste simples, altere o modo de funcionamento do script de *mode='a'* para *mode='b'*, além de alterar o canal de 36 para 1. Então, execute o *iperf* mais uma vez e observe o resultado.



Por falta de conhecimento, muitos usuários acabam cometendo equívocos quando configuram o canal em um ponto de acesso. O que acontece é que, por exemplo, o canal 1 (um) não funciona a 5 GHz, frequência utilizada para o padrão IEEE 802.11a. Então, não é possível utilizar faixas de canais que não funcionam para determinados padrões. A página do *hostapd*<sup>8</sup> pode servir como um bom ponto de referência para identificar os canais corretos para determinados padrões de operação.

```
mininet-wifi> iperf sta1 sta2
*** Iperf: testing TCP bandwidth between sta1 and sta2
*** Results: ['4.50 Mbits/sec', '4.57 Mbits/sec']
```

Como é possível perceber, a largura de banda mensurada foi de 4.5 Mbits/s, que está limitado aos 11 Mbits/s, exatamente conforme definido para o padrão IEEE 802.11b.

## 2.10 Relação distância *versus* largura de banda

A largura de banda (ou *bandwidth*) refere-se à capacidade de transmissão de dados de um ponto de rede para outro em um intervalo de tempo, determinando a velocidade em que os dados percorrem um link. Em redes sem fio, a largura de banda, é, teoricamente, bastante impactada pela distância entre 2 (dois) nós.

Dentre as ferramentas para mensuração da largura de banda, certamente o *iperf* é aquela que mais se destaca, sendo a preferida na grande maioria dos casos. A medição de largura de banda com o *iperf* é relativamente simples, pois basta que dois nós o executem, sendo um deles operando em modo cliente e outro em modo servidor.

<sup>8</sup><https://w1.fi/cgit/hostap/plain/hostapd/hostapd.conf>

Para utilizar o *iperf* e verificar a relação entre distância e largura de banda, vamos tomar como referência o arquivo *<position.py>*.

```
~/mininet-wifi$ sudo python examples/position.py
```

Após executá-lo, será possível observar uma topologia com duas estações e um ponto de acesso.

Considerando que o arquivo de script foi executado com sucesso, vamos medir a largura de banda entre *sta1* e *sta2* conforme a disposição inicial deles.

```
mininet-wifi> iperf sta1 sta2
*** Iperf: testing TCP bandwidth between sta1 and sta2
*** Results: ['8.42 Mbits/sec', '9.04 Mbits/sec']
```

Guarde o resultado observado nessa rodada de teste. *O resultado pode sofrer pequenas variações.*

Agora, conforme abaixo, vamos alterar a posição de *sta1* e *sta2* de forma que elas fiquem mais distantes do ponto de acesso *ap1*. Em seguida, vamos novamente mensurar a largura de banda entre *sta1* e *sta2*.

```
mininet-wifi> py sta1.setPosition('40,90,0')
mininet-wifi> py sta2.setPosition('60,10,0')
mininet-wifi> iperf sta1 sta2
*** Iperf: testing TCP bandwidth between sta1 and sta2
*** Results: ['6.98 Mbits/sec', '7.14 Mbits/sec']
```

Comparando este novo resultado com o anterior, fica claro que quanto mais distantes as estações estiverem do ponto de acesso menor tende a ser a largura de banda.



No Mininet-WiFi, o comando *iperf sta1 sta2* automaticamente define *sta1* como servidor e *sta2* como cliente. Mais a frente veremos exemplos da forma mais comum de utilização do *iperf*, que é compatível com o ambiente real, ou seja, com computadores físicos.

**Publicações que já utilizaram o Mininet-WiFi para pesquisas em desempenho:**

- Gilani S.M.M., Heang H.M., Hong T., Zhao G., Xu C. *OpenFlow-Based Load Balancing in WLAN: Throughput Analysis*. Communications, Signal Processing, and Systems (CSPS), 2016.
- Krishna Vijay Singh, Sakshi Gupta, Saurabh Verma, Mayank Pandey. *Improving performance of TCP for wireless network using SDN*. Proceedings of ICDCN, 2019.

## 2.11 Modelos de mobilidade

Modelos de mobilidade também são bastante importantes, pois são eles que tentam imitar a mobilidade das pessoas, veículos ou qualquer outro objeto que tenha a mobilidade como característica. Existem diversas pesquisas científicas, inclusive, que tentam identificar o padrão de mobilidade de pessoas em casos de desastres naturais, como grandes tempestades, alagamentos, etc.

Assim como para o modelo de propagação, existem diversos modelos de mobilidade já conceituados pela comunidade científica, sendo alguns deles já suportados pelo Mininet-WiFi, como o *Random Direction*, *Random Walk*, *Gauss Markov*, entre outros. A mobilidade pode ser observada somente através da CLI ou através de um gráfico, conforme ilustrado na Figura 2.9.

Assim, devido a sua importância, vamos verificar, então, como os modelos de mobilidade podem ser configurados no Mininet-WiFi. Para tanto, vamos tomar como exemplo o script `<mobilityModel.py>`, que já vem configurado com o modelo *Random Direction*. É importante destacar que para cada modelo de mobilidade podem existir parâmetros únicos, como limites mínimos e máximos de velocidade, áreas onde os nós poderão se movimentar, etc. Você encontrará todas as informações que precisa acerca das configurações dos modelos de mobilidade no manual do Mininet-WiFi.

Uma configuração que vale a pena ser destacada é a semente (ou *seed*). A semente faz com que a mobilidade sofra uma importante modificação. Por exemplo, se a semente de número 1 (um) faz com que os nós se movimentem

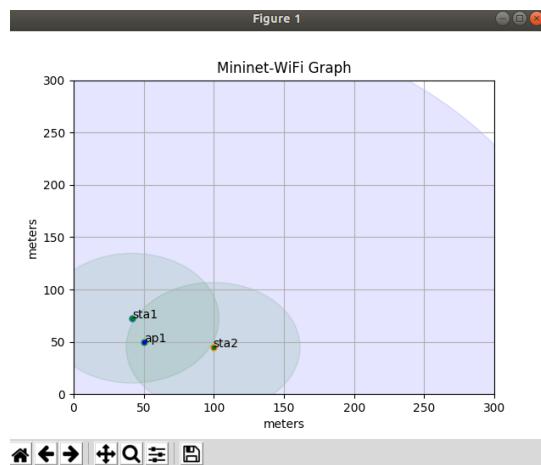


Figura 2.9: Mobilidade em ação.

a partir de um determinado valor para x e y, a semente de número 2 (dois) irá alterar os valores iniciais de x e y. Ela não será capaz de alterar o comportamento da mobilidade, mas poderá alterar as posições iniciais da mobilidade. Esse é um recurso importante, pois nem sempre a disposição inicial dos nós pode atender a requisitos definidos para um determinado experimento.

Agora que já conhecemos na teoria um pouco sobre modelos de mobilidade, vamos executar o script abaixo e observar como os nós se comportam quando configurados com o modelo de mobilidade *Random Direction*.

```
~/mininet-wifi$ sudo python examples/mobilityModel.py
```



Observe o comportamento na mobilidade dos nós e experimente alterar o modelo de mobilidade em um outro momento para efeito de comparação.

Em seguida, vamos testar 2 (dois) dos 3 (três) comandos implementados para o Mininet-WiFi que foram apresentados na seção 2.1. O `stop` e o `start`.

Vamos utilizar primeiro o comando `stop`.

```
mininet-wifi> stop
```

Se tudo ocorrer como esperado, o comando `stop` irá congelar a mobilidade, fazendo com que os nós parem de se movimentar. Esse recurso pode ser bastante útil em casos onde o usuário deseja observar informações como nível de sinal ou mesmo largura de banda disponível dada uma disposição dos nós.

Agora, podemos emitir o comando `start` para fazer com que o processo de mobilidade tenha continuidade.

```
mininet-wifi> start
```



Todos os modelos de mobilidade suportados pelo Mininet-WiFi podem ser encontrados no diretório `<mn_wifi/mobility.py>`. Caso você queira implementar novos modelos de mobilidade, é neste arquivo que você deve incluir os novos modelos.

#### Publicações que já utilizaram o Mininet-WiFi para pesquisas em mobilidade:



- K. V. K. Singh, M. Pandey. *Software-defined mobility in IP based Wi-Fi networks: Design proposal and future directions*. IEEE ANTS, 2016
- D. Tu, Z. Zhao and H. Zhang. *ISD-WiFi: An intelligent SDN based solution for enterprise WLANs*. WCSP, 2016,
- F. S. Dantas Silva, A. Neto, D. Maciel, J. Castillo-Lema, F. Silva, P. Frovi, E. Cerqueira, *An innovative software-defined WiNeMO architecture for advanced QoS-guaranteed mobile service transport*. Computer Networks, 2016.
- C. H. F. dos Santos, M. P. S. de Lima, F. S. Dantas Silva, A. Neto. *Performance Evaluation of Multiple Attribute Mobility Decision Models: A QoE-efficiency Perspective*. WiMob, 2017.
- Y. Bi, G. Han, C. Lin, Q. Deng, L. Guo and F. Li. *Mobility Support for Fog Computing: An SDN Approach*. IEEE Communications Magazine, 2018.
- A. Kaul, L. Xue, K. Obraczka, M. Santos, T. Turletti. *WiMbitHandover and Load Balancing for Distributed Network Control: Applications in ITS Message Dissemination*. ICCCN, 2018.
- Z. Han, T. Lei, Z. Lu, X. Wen, W. Zheng, L. Guo. *Artificial Intelligence Based Handoff Management for Dense WLANs: A Deep Reinforcement Learning Approach*. IEEE Access, 2019.