Mininet-WiFi

Emulator for Software-Defined Wireless Networks

https://github.com/intrig-unicamp/mininet-wifi

IFBA

Compreendendo o Funcionamento das Redes Sem Fio com o Mininet-WiFI

Jequé, BA

2017

INFORMATION & NETWORKING TECHNOLOGIES RESEARCH & INNOVATION GROUP (INTRIG)
DEPARTMENT OF COMPUTER ENGINEERING AND INDUSTRIAL AUTOMATION (DCA)
SCHOOL OF ELECTRICAL AND COMPUTER ENGINEERING (FEEC)
UNIVERSITY OF CAMPINAS(UNICAMP) - BRAZIL

GITHUB.COM/INTRIG-UNICAMP/MININET-WIFI

Mininet-WiFi

Emulator for Software-Defined Wireless Networks

https://github.com/intrig-unicamp/mininet-wifi

Contents

1. Background

A emulação de redes tem sido bastante utilizada na avaliação de desempenho, em testes e depuração de protocolos e também em diversos assuntos relacionados a pesquisas em redes de computadores. Dentro desse universo encontra-se a emulação de redes sem fio. Devido a diversos fatores, como *modulação*, *interferência eletromagnética*, *mobilidade*, a emulação de redes sem fio não é uma tarefa fácil, mas quando implementada em um ambiente controlado pode possibilitar projetos e desenvolvimentos de novos protocolos e aplicações para redes WiFi com alta fidelidade.

Visando proporcionar esse ambiente controlado e capaz de prover alta fidelidade, este laboratório basea-se no Mininet-WiFi, uma extensão do emulador Mininet [1] com suporte a WiFi. Com ele é possível a virtualização de estações e pontos de acesso, e também utilizar dos nós já presentes do Mininet, como computadores, comutadores e controladores OpenFlow. Além disso, o Mininet-WiFi possibilita a filtragem de pacotes utilizando o protocolo OpenFlow [4], importante solução para as redes definidas por software (SDN) [3] [5].

Mininet-WiFi

O Mininet-WiFi está sendo desenvolvido com base no código do Mininet [1] 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 optar por ambientes WiFi, onde ainda assim é possível inserir computadores, comutadores e controladores, além de estações e pontos de acesso em uma mesma topologia.

Arquitetura

Todo o processo de virtualização do Mininet-WiFi funciona de forma similar ao Mininet, tendo como base processos que são executados em *network namespaces* e placas de rede virtuais. Em nossa implementação, as interfaces sem fio virtualizam o hardware WiFi, que a depender do modo de funcionamento (estação ou ponto de acesso), pode funcionar nos modos *managed* e *master*, re-

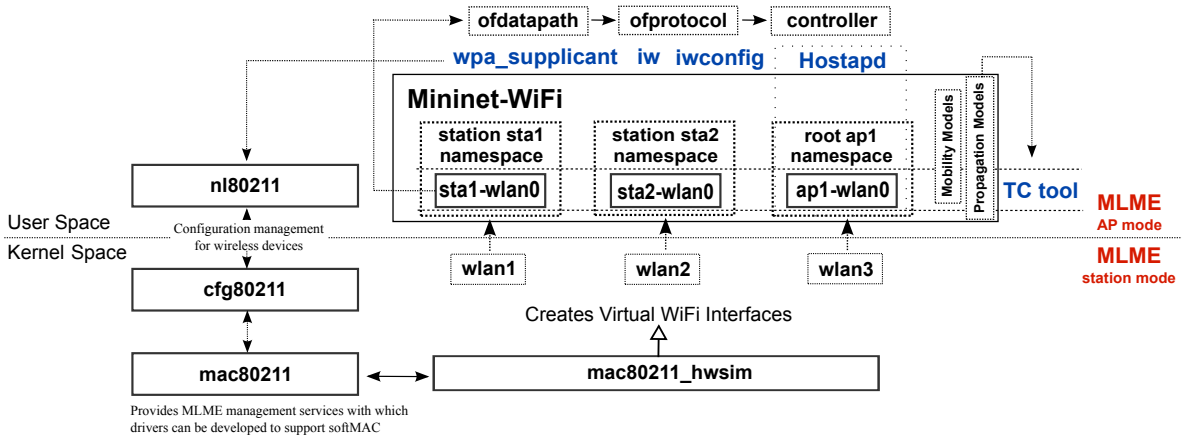


Figure 1.1: Principais componentes do Mininet-WiFi.

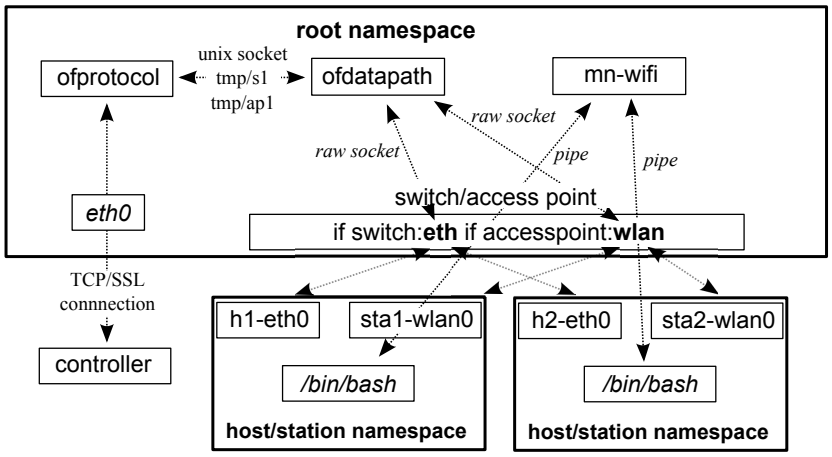


Figure 1.2: Arquitetura de sistema do Mininet-WiFi com dois sistemas finais.

spectivamente. As **estações** se comunicam com os pontos de acesso através de um processo chamado de autenticação e associação. Em nossa implementação cada estação possui uma interface sem fio (por padrão, podendo serem adicionadas mais). Uma vez associada a um ponto de acesso, as estações podem se comunicar com os tradicionais *hosts* do Mininet (tratamos como computadores na escrita deste trabalho), caso estes estiverem também conectados ao ponto de acesso (que é um switch virtual com capacidades WiFi) através de um meio cabeado. Já os pontos de acesso são responsáveis pela gestão das estações que estão associadas a ele. Os pontos de acesso são virtualizados através do daemon *hostapd*[1] que basicamente utilizam interfaces WiFi virtuais para prover capacidades de um ponto de acesso. Detalhes sobre o ambiente de execução do Mininet-WiFi são tratatadas na subseção a seguir.

[1]Hostapd (**H**ost **A**ccess **P**oint **D**aemon) é um software a nível de usuário capaz de tonar uma interface de rede sem fio em pontos de acesso e servidores de autenticação.

WorkFlow

O processo de funcionamento do Mininet-WiFi ocorre da seguinte forma: durante a inicialização de uma topologia, primeiramente ele verifica se a topologia é uma topologia WiFi. Se não for, ele irá funcionar exatamente com os mesmos recursos que o Mininet oferece. Por outro lado, se for uma topologia WiFi, o Mininet-WiFi deverá carregar um módulo chamado mac80211_hwsim com o número de interfaces sem fio virtuais definidas pelo usuário (padrão é 3). Com o módulo carregado, o daemon *hostapd* também é executado para prover os serviços de ponto de acesso.

Antes de se comunicarem com o SoftMac, as estações e pontos de acesso se comunicam com uma API chamada de cfg80211, que por sua vez se comunica com uma framework chamada mac80211. É esta framework que interage com o SoftMac através de um socket, o nl80211. A maioria dos dispositivos wireless para Linux utilizam o SoftMac e suportam a maioria dos recursos oferecidos pelas interfaces wireless [2].

Para iniciar uma topologia com duas estações e um ponto de acesso, basta utilizar o comando *mn* `--wifi --ssid=new-ssid` na CLI do Mininet-WiFi. Como esse comando as duas estações são automaticamente associadas ao ponto de acesso através do SSID *new-ssid* e um controlador OpenFlow também é iniciado e responsável pelo controle de fluxos do ponto de acesso.

Além da utilização de comandos, também é possível construir topologias através de alguns arquivos de exemplo que estão dentro da pasta *examples* no código do Mininet-WiFi. Acreditamos que os exemplos são importantes e serão facilitadores para os usuários que desejam criar novas topologias. Além da CLI, também é possível criar topologias para o Mininet-WiFi utilizando o *Visual Network Description*, disponível em `http://ramonfontes.com/vnd`. (Alguns vídeos de demonstração estão disponíveis nesta página e na *wiki* do Mininet-WiFi: `https://github.com/intrig-unicamp/mininet-wifi/wiki/Demos`).

Interagindo com o ambiente de emulação

O Mininet-WiFi mantêm a mesma estrutura de interação do Mininet, por exemplo: para realizar testes de conectividade entre dois nós de rede, comandos como `sta1 ping sta2` ou `sta1 iperf -s & sta2 iperf -c 10.0.0.1` podem ser utilizados, respectivamente, para testes de conectividade ou para mensurar largura de banda.

Adicionalmente, outros comandos podem ser utilizados para melhor experiência com o ambiente WiFi, como: `sta1 iw dev sta1-wlan0 scan` para que uma estação *sta1* busque as redes que estão ao seu alcance ou `sta1 iw dev sta1-wlan0 connect my-ssid` para possibilitar a associação de uma estação *sta1* a um ponto de acesso com SSID chamado *my-ssid*. Todos os novos comandos disponíveis no Mininet-WiFi são provenientes do ***iw*** que substitui o já conhecido *iwconfig* em sistemas Linux.

A Figura 1.3 traz informações da captura de pacote realizada pelo *wireshark* no instante em que uma estação se associava a um ponto de acesso. A imagem traz informações como o SSID, taxas de conexão suportadas e capacidades HT (high throughput).

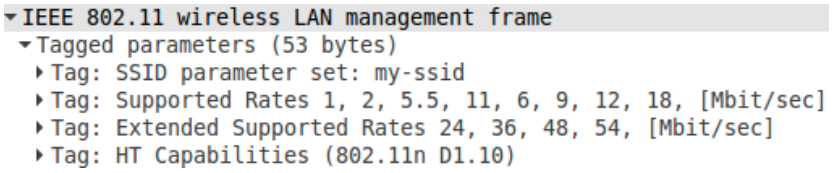


Figure 1.3: Captura de Pacotes.

Further Info & Resources

- Repositorio Github e Wiki: `https://github.com/intrig-unicamp/mininet-wifi`
- Manual do Usuário: disponível na página do código fonte.
- Demonstrações e Videos: `https://github.com/intrig-unicamp/mininet-wifi/wiki/Demos`

Q 1.1: Em sua opinião, qual a principal publicação científica relacionada ao Mininet-WiFi? Por quê? Você pode utilizar o google scholar para identificar os artigos que citam o Mininet-WiFi.

2. Primeiros Passos

Neste capítulo você vai aprender conceitos básicos de como a emulação de redes sem fio funciona no Mininet-WiFi. Para responder às perguntas contidas neste tutorial, considere clonar o Mininet-WiFi e instalá-lo, de acordo com as instruções disponíveis em `https://github.com/intrig-unicamp/mininet-wifi`. Alternativamente você pode usar a máquina virtual pre-configurada disponiblizada. O manual do usuário, também disponível no mesmo endereço pode te ajudar a responder algumas perguntas.

Utilizando o Mininet-WiFi:

O comando abaixo permite criar uma rede simples, com 2 estações e um ponto de acesso:

```
sudo mn --wifi
```

Experimente utilizar o comando **nodes** para identificar os nós que fazem parte da rede.

```
mininet-wifi>nodes
```

Agora experimente os comandos abaixo.

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

Agora, desconecte sta1 e confirme a desconexão:

```
mininet-wifi>sta1 iw dev sta1-wlan0 disconnect
mininet-wifi>sta1 iwconfig
mininet-wifi>sta1 ping sta2
```

Agora, conecte sta1 novamente:

```
mininet-wifi>sta1 iw dev sta1-wlan0 connect my-ssid
mininet-wifi>sta1 iwconfig
mininet-wifi>sta1 ping sta2
```

Q 2.1: Qual é o atraso observado entre sta1 e sta2? Houve perda de pacotes no canal? Justifique suas respostas de forma objetiva.

Você pode, teoricamente, utilizar na CLI do Mininet-WiFi qualquer comando disponível no Linux, incluindo utilitários como iw, iwconfig, etc. Porém, como trata-se de um ambiente emulado e nele podem estar múltiplos nós, você precisa identificar quem é o nó que será responsável por um dado comando (por exemplo, `sta1 iwconfig`).

Q 2.2: Use a ferramenta *iperf* para avaliar a banda disponível (Mbps) entre os nós sta1 e sta2.

Acessando informações dos nós

Se estiver dentro da CLI do Mininet-WiFi, vamos sair com *exit*.

```
mininet-wifi>exit
```

Agora, abra a mesma topologia acrescentando dois novos parâmetros: *position* e *wmediumd*. O parâmetro *position* vai definir posições iniciais para os nós; e o parâmetro *wmediumd* irá habilitar o wmediumd, um simulador do meio sem fio (*mais detalhes sobre o wmediumd estão disponíveis no manual do Mininet-WiFi*).

```
sudo mn --wifi --position --wmediumd
```

Outros recursos disponíveis no Mininet-WiFi podem facilitar a interação com os nós, por exemplo, o comando abaixo permite identificar diversas informações acerca do nó, como *posição*, *endereço ip*, *modo* e *frequência* de funcionamento, dentre outros.

```
mininet-wifi>py sta1.params
```

É possível também filtrar informações específicas, como abaixo, que permite visualizar a posição do nó (*importante: só será possível verificar a posição do nó, se ela estiver sido definida previamente no código*).

```
mininet-wifi>py sta1.params['position']
```

Através da CLI do Mininet-WiFi o usuário também pode definir uma nova posição, por exemplo:

```
mininet-wifi>py sta1.setPosition('10,20,0')
```

E finalmente encontrar a distância entre dois nós, como em:

```
mininet-wifi>distance sta1 sta2
```

Q 2.3: Faça um simples comentário para cada uma das informações especificadas na lista de parâmetros de sta1.

Wireshark: Análise de quadros 802.11

802.11 (Wireless LAN) e 802.3 (Ethernet LAN) são dois protocolos da familia de padróes IEEE 802 LAN/MAN que usam acesso a um canal de comunicação compartilhado e endereçamento de camada de enlace (Medium Access Control - MAC) com endereços do tipo Ethernet formados por 6 octetos (48-bits).

Q 2.4: Compare uma captura de pacotes em um enlace Ethernet com fio (802.3) e sem fio (802.11) e discuta eventuais diferenças em relação ao número de endereços MAC contidos nos quadros do tipo 802.11 e 802.3 e sua função no contexto da camada de enlace de redes locais.

(Obs. Fundamentação teórica disponível nos Slides 26 - 28 do material de apoio do Kurose, "Capítulo 6 Redes Sem-fio e Redes Móveis")

Visualizando gráficos 2D e 3D:

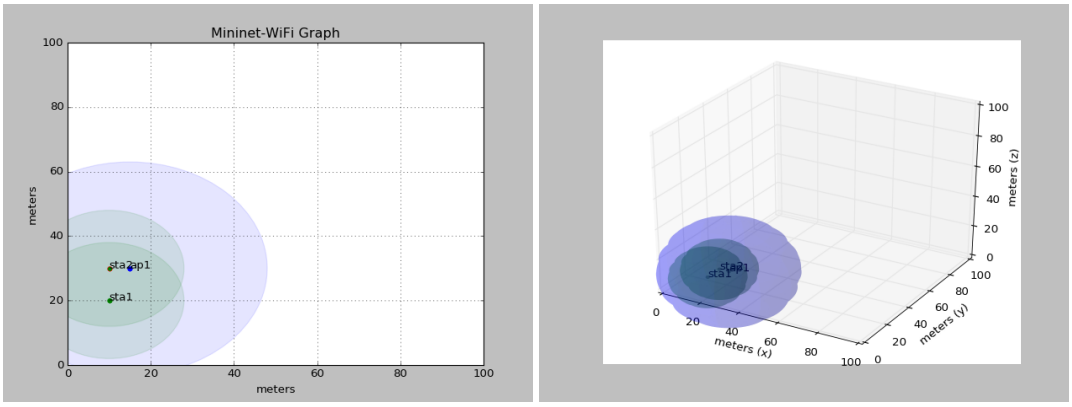
O Mininet-WiFi permite ao usuário executar topologias 2D e 3D. Em sua totalidade, os códigos disponíveis em /examples geram grafos em 2D. Caso o usuário deseje gerar gráficos em 3D, basta fazer uma simples alteração no código. Por exemplo, tomamos como exemplo o arquivo localizado em */examples/position.py*. Este arquivo contêm a seguinte linha:

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

Como pode ser visto acima, apenas os eixos x e y foram definidos e o resultado do gráfico será algo similar ao apresentado na Figura 2.1a. Logo, para gerar gráficos em 3D basta adiciona o eixo z, que resultará em algo similar ao apresentado na Figura 2.1b:

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

Os gráficos gerados pelo Mininet-WiFi são suportados pelo matplotlib, uma biblioteca disponível na linguagem de progamação Python que possibiltia a criação de gráficos. E devido a uma limitação no matplotlib, todos os eixos precisam ser definidos com valores iguais.



(a) Gráfico em 2D. (b) Gráfico em 3D.

Figure 2.1: Gráficos 2D e 3D.

3. Modelos de Propagação

Sendo o Mininet-WiFi um emulador, ele precisa utilizar de modelos matemáticos para representar com maior fidelidade possível o comportamento do meio sem fio. Este comportamento é definido através de modelos matemáticos, que podem diferenciar uns dos outros acerca dos atributos suportados. Por exemplo, um modelo pode considerar a interferência entre os sinais provenientes de diferentes dispositivos sem fio, enquanto outros não. Para responder às questões abaixo considere o arquivo **propagation_model.py** (`sudo python propagation_model.py`) disponível em `https://github.com/ramonfontes/reproducible-research/tree/master/mininet-wifi/UNICAMP-2017`. Considere também utilizar o comando iwconfig (por exemplo: `sta1 iwconfig`) no terminal do Mininet-WiFi.

Questão 3.1: Um dos nós parece estar incomunicável. Qual é ele e por qual motivo o nó está incomunicável? Ele encontra-se associado ao AP1? Tente responder de forma técnica.

Questão 3.2: O arquivo acima já vem pré-configurado com um modelo de propagação. Identifique um modelo de propagação diferente do configurado e que é suportado pelo Mininet-WiFi e então configure esse modelo no código acima. Identifique e descreva abaixo o nível de sinal percebido por sta1. Justifique o resultado obtido.

Questão 3.3: Em qual(is) tipo(s) de ambiente(s) você utilizaria o modelo escolhido acima? Por quê?

4. Relação *Distância* x *Largura de Banda/RSS*

Alguma das vantanges de se utilizar modelos de propagação passa pelo fato de ser possível calcular o nível de sinal percebido por uma estação a partir de alguns atributos, como a distância. A distância, por sua vez, impacta diretamente na transmissão de dados, uma vez que quanto menor o nível de sinal percebido, menor será a taxa de transmissão. O nível de sinal para redes WiFi normalmente se encontra entre 0 e -100 dBm. Quanto menor o valor, pior será o nível de sinal percebido.

A Figura 4.1 ilustra como exemplo uma possível relação entre Distância e Largura de Banda e RSSI a partir de um modelo de propagação aleatoriamente escolhido. Os valores para largura de banda apresentados na figura foram gerados a partir da saída da ferramenta *Iperf* que permite mensurar a largura de banda entre origem e destino.

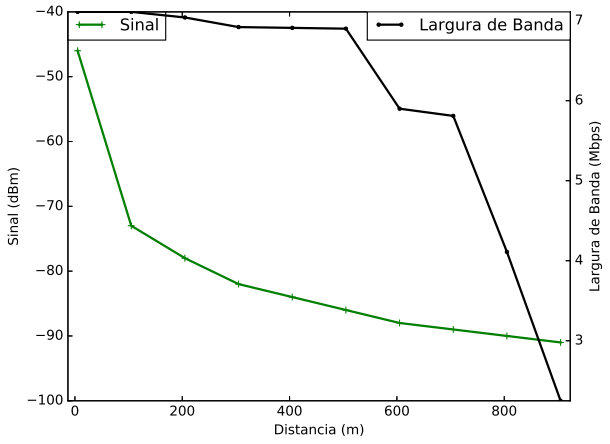


Figure 4.1: Relação Distância x Largura de Banda/RSSI

Questão 4.1: Utilizando o arquivo propagation_model.py e de forma similar ao exemplo acima, utilize o Iperf para mensurar a largura de banda entre dois nós que distam *x* metros.

Obs 1. A escolha do modelo de propagação e sua configuração é livre e você poderá apresentar os

resultados obtidos através de tabela ou se preferir, gráfico. Não esqueça de destacar, para cada distância, o nível de sinal (RSSI) e largura de banda (Mbps) observados.
Obs 2. (IMPORTANTE): O Mininet-WiFi possui um *threshold*, chamado de CCA Threshold (Clear Channel Assessment), que corta a comunicação quando o nível de sinal é inferior a -92 dBm.

Questão 4.2: O que seria o CCA Threshold?

5. Mobilidade e Handover

O Mininet-WiFi também suporta modelos de mobilidade. Além dos modelos de mobilidade, uma outra opção disponível para o usuário é criar a mobilidade configurando a posição inicial e final do nó e também o tempo que o nó levará para percorrer os 2 pontos.

Execute o arquivo **handover.py** (sudo python handover.py) disponível em `https://github.com/ramonfontes/reproducible-research/tree/master/mininet-wifi/UNICAMP-2017`. Em seguida, no terminal do Mininet-WiFi digite `sta1 ping sta2`.

Questão 5.1: A comunicação entre sta1 e sta2 se manteve ininterrupta? Por quê? O que ocorreu com sta1 que o difere de sta2?

Questão 5.2: Qual o nome do processo ocorrido com sta1? Comente como ele funciona.

6. (Opcional) Wireless Mesh x Adhoc

Até o momento, somente redes infra-estruturadas foram exploradas. Nesse tipo de rede a transferência de dados acontece sempre entre uma estação e um ponto de acesso, ou AP (Access Point). Os APs são nós responsáveis pela captura e retransmissão das mensagens enviadas pelas estações. A transferência de dados nunca ocorre diretamente entre duas estações.

Agora, vamos explorar outros 2 tipos de redes sem fio, as redes sem fio mesh e adhoc, onde não há a presença do nó central, o AP. Para responder as perguntas a seguir considere o arquivo **adhoc_mesh.py** (`sudo python adhoc_mesh.py`) disponível em:
`https://github.com/ramonfontes/reproducible-research/tree/master/mininet-wifi/UNICAMP-2017`.

Questão 6.1: O que ocorre quando sta1 tenta se comunicar com sta3? E sta4 com sta6? Comente e justifique o comportamento.

Questão 6.2: Crie rotas estáticas de forma que sta4 consiga se comunicar com sta6. Descreva abaixo os comandos utilizados para a criação das rotas.
Dica: Comandos comuns ao sistema operacional linux podem ser utilizados, como o *ip route*). Considere utilizá-los no terminal do Mininet-WiFi. Assim você terá garantias que seus comandos funcionarão como esperado através de testes que você mesmo poderá realizar após aplicar as regras de roteamento.

Agora vamos realizar uma mudança simples no código de forma que você possa gerar um cenário 3D. Para tanto, altere os seguintes trechos no código:

De: `sta1 = net.addStation('sta1', position='10,10,0')`
Para: `sta1 = net.addStation('sta1', position='10,10,50')`

De: `net.plotGraph(max_x=200, max_y=200)`

Para: `net.plotGraph(max_x=200, max_y=200, max_z=200)`

Questão 6.3: Como pode ver sta1 agora parece estar incomunicável, pois está fora do raio de alcance de sta2 e sta3. Para quais tipos de experimentos você utilizaria a visualização 3D?

Bibliography

Lantz, Bob, Brandon Heller, and Nick McKeown (2010). "A Network in a Laptop: Rapid Prototyping for Software-defined Networks". In: *Proceedings of the 9th ACM SIGCOMM Workshop on Hot Topics in Networks*. Hotnets-IX. Monterey, California: ACM, 19:1–19:6. ISBN: 978-1-4503-0409-2. DOI: 10.1145/1868447.1868466. URL: http://doi.acm.org/10.1145/1868447.1868466.

Linux Wireless: mac80211 Documentation. http://linuxwireless.org/en/developers/Documentation/mac80211. (acessado em 05/06/2015).

Mckeown, N. *How SDN will Shape Networking*. https://www.youtube.com/watch?v=c9-K5O_qYgA. (acessado em 02/07/2015).

McKeown, Nick et al. (2008). "OpenFlow: Enabling Innovation in Campus Networks". In: *SIGCOMM Comput. Commun. Rev.* 38.2, pp. 69–74. ISSN: 0146-4833. DOI: 10.1145/1355734.1355746. URL: http://doi.acm.org/10.1145/1355734.1355746.

Schenker, S. *The Future of Networking, and the Past of Protocols*. http://www.youtube.com/watch?v=YHeyuD89n1Y. (acessado em 02/07/2015).