



# EA080 - O — Compreendendo o Funcionamento das Redes sem Fio com o Mininet-WiFi

**Professor:** Christian Esteve Rothenberg

Leonardo Rodrigues Marques RA: 178610

---

## 1 Introdução

Nesse segundo laboratório de redes de computadores, fomos introduzidos ao ambiente de simulação Mininet-WiFi. Esse ambiente consiste em um conjunto de estações conectadas a um ponto de acesso. Usando dessa topologia de rede, podemos testar o desempenho da rede sem fio em diferentes condições, seja questões de interferência, ou distância e até de tecnologia implementada. Para isso, valemos de novos comandos para extrair novas informações dos parâmetros das redes. De mão disso, conseguimos aprender mais sobre o funcionamento das redes WLAN e os problemas envolvidos.

## 2 Metodologia

A metodologia usada para desenvolver esse trabalho consistiu em alguns fundamentos de apoio. Em primeiro lugar, o manual disponível no GitHub do projeto forneceu conceitos essenciais para entender o funcionamento de alguns componentes e modelos do Mininet-Wifi. Fundamentado, a próxima etapa consistia em executar o o projeto e aplicar novos comandos a fim de interagir com o sistema e extrair as propriedades e parâmetros da rede e dos dispositivos. Finalmente, co-relacionando esses dados, podemos simular condições reais com gráficos, tabelas e aprofundar nas dificuldades dos problemas.

## 3 Resultado, Discussões e Conclusões

### 3.1 Questão 1

Na tentativa de buscar a publicação mais relevante sobre o Mininet-WiFi, eu optei por procurar no [GitHub](#) do projeto. Na aba Wiki, eu achei algumas informações sobre Publicações relacionadas ao projeto. Ao selecionar a opção, eu encontrei um paper [Mininet-WiFi: Emulating Software-Defined Wireless Networks](#), explicando o funcionamento básico do Mininet-Wifi e alguns exemplos de casos de estudo úteis.

## 3.2 Questão 2

### 3.2.1

1. Após inicializar o Mininet-WiFi, testei a conexão entre sta1 e sta2 usando o comando ping com 20 iterações. Obtive 0% de perda de pacotes e um tempo médio de **0.427ms**.

```
mininet-wifi> sta1 ping -c20 sta2
ping: -: None ou serviço desconhecido
mininet-wifi> sta1 ping -c20 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=0.609 ms
64 bytes from 10.0.0.2: icmp_seq=2 ttl=64 time=0.544 ms
64 bytes from 10.0.0.2: icmp_seq=3 ttl=64 time=0.451 ms
64 bytes from 10.0.0.2: icmp_seq=4 ttl=64 time=0.423 ms
64 bytes from 10.0.0.2: icmp_seq=5 ttl=64 time=0.762 ms
64 bytes from 10.0.0.2: icmp_seq=6 ttl=64 time=0.274 ms
64 bytes from 10.0.0.2: icmp_seq=7 ttl=64 time=0.337 ms
64 bytes from 10.0.0.2: icmp_seq=8 ttl=64 time=0.459 ms
64 bytes from 10.0.0.2: icmp_seq=9 ttl=64 time=0.573 ms
64 bytes from 10.0.0.2: icmp_seq=10 ttl=64 time=0.139 ms
64 bytes from 10.0.0.2: icmp_seq=11 ttl=64 time=0.253 ms
64 bytes from 10.0.0.2: icmp_seq=12 ttl=64 time=0.395 ms
64 bytes from 10.0.0.2: icmp_seq=13 ttl=64 time=0.481 ms
64 bytes from 10.0.0.2: icmp_seq=14 ttl=64 time=0.488 ms
64 bytes from 10.0.0.2: icmp_seq=15 ttl=64 time=0.482 ms
64 bytes from 10.0.0.2: icmp_seq=16 ttl=64 time=0.450 ms
64 bytes from 10.0.0.2: icmp_seq=17 ttl=64 time=0.136 ms
64 bytes from 10.0.0.2: icmp_seq=18 ttl=64 time=0.462 ms
64 bytes from 10.0.0.2: icmp_seq=19 ttl=64 time=0.378 ms
64 bytes from 10.0.0.2: icmp_seq=20 ttl=64 time=0.453 ms

--- 10.0.0.2 ping statistics ---
20 packets transmitted, 20 received, 0% packet loss, time 19444ms
rtt min/avg/max/mdev = 0.136/0.427/0.762/0.147 ms
mininet-wifi> □
```

Figura 1: Teste de Ping entre sta1 e sta 2 com 20 iterações.

2. Após essa etapa, desconectei sta1 do ponto de acesso com o comando `sta1 iw dev sta1-wlan0 disconnect` e repeti o teste de ping com 20 iterações. Notadamente, tive 100% de perda dos pacotes.

```
mininet-wifi> sta1 iw dev sta1-wlan0 disconnect
mininet-wifi> sta1 ping -c20 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
From 10.0.0.1 icmp_seq=2 Destination Host Unreachable
From 10.0.0.1 icmp_seq=3 Destination Host Unreachable
From 10.0.0.1 icmp_seq=4 Destination Host Unreachable
From 10.0.0.1 icmp_seq=5 Destination Host Unreachable
From 10.0.0.1 icmp_seq=6 Destination Host Unreachable
From 10.0.0.1 icmp_seq=7 Destination Host Unreachable
From 10.0.0.1 icmp_seq=8 Destination Host Unreachable
From 10.0.0.1 icmp_seq=9 Destination Host Unreachable
From 10.0.0.1 icmp_seq=10 Destination Host Unreachable
From 10.0.0.1 icmp_seq=11 Destination Host Unreachable
From 10.0.0.1 icmp_seq=12 Destination Host Unreachable
From 10.0.0.1 icmp_seq=13 Destination Host Unreachable
From 10.0.0.1 icmp_seq=14 Destination Host Unreachable
From 10.0.0.1 icmp_seq=15 Destination Host Unreachable
From 10.0.0.1 icmp_seq=16 Destination Host Unreachable
From 10.0.0.1 icmp_seq=17 Destination Host Unreachable
From 10.0.0.1 icmp_seq=18 Destination Host Unreachable
From 10.0.0.1 icmp_seq=19 Destination Host Unreachable
From 10.0.0.1 icmp_seq=20 Destination Host Unreachable

--- 10.0.0.2 ping statistics ---
20 packets transmitted, 0 received, +20 errors, 100% packet loss, time 19449ms
pipe 4
mininet-wifi> □
```

Figura 2: Repetição do teste de Ping após desconexão de sta1.

3. Por fim, reconectei sta1 com o ponto de acesso com o comando `sta1 iw dev sta1-wlan0 connect my-ssid` e novamente repeti o teste de ping com 20 iterações. Obtive 5% de perda de pacotes, entretanto o tempo médio *0.431ms* manteve semelhante ao primeiro teste de ping executado.

```
mininet-wifi> sta1 iw dev sta1-wlan0 connect my-ssid
mininet-wifi> sta1 ping -c20 sta2
PING 10.0.0.2 (10.0.0.2) 56(84) bytes of data.
64 bytes from 10.0.0.2: icmp_seq=2 ttl=64 time=0.419 ms
64 bytes from 10.0.0.2: icmp_seq=3 ttl=64 time=0.324 ms
64 bytes from 10.0.0.2: icmp_seq=4 ttl=64 time=0.308 ms
64 bytes from 10.0.0.2: icmp_seq=5 ttl=64 time=0.409 ms
64 bytes from 10.0.0.2: icmp_seq=6 ttl=64 time=0.401 ms
64 bytes from 10.0.0.2: icmp_seq=7 ttl=64 time=0.258 ms
64 bytes from 10.0.0.2: icmp_seq=8 ttl=64 time=0.343 ms
64 bytes from 10.0.0.2: icmp_seq=9 ttl=64 time=0.415 ms
64 bytes from 10.0.0.2: icmp_seq=10 ttl=64 time=0.420 ms
64 bytes from 10.0.0.2: icmp_seq=11 ttl=64 time=0.359 ms
64 bytes from 10.0.0.2: icmp_seq=12 ttl=64 time=0.329 ms
64 bytes from 10.0.0.2: icmp_seq=13 ttl=64 time=0.738 ms
64 bytes from 10.0.0.2: icmp_seq=14 ttl=64 time=0.502 ms
64 bytes from 10.0.0.2: icmp_seq=15 ttl=64 time=0.434 ms
64 bytes from 10.0.0.2: icmp_seq=16 ttl=64 time=0.506 ms
64 bytes from 10.0.0.2: icmp_seq=17 ttl=64 time=0.479 ms
64 bytes from 10.0.0.2: icmp_seq=18 ttl=64 time=0.456 ms
64 bytes from 10.0.0.2: icmp_seq=19 ttl=64 time=0.477 ms
64 bytes from 10.0.0.2: icmp_seq=20 ttl=64 time=0.630 ms

--- 10.0.0.2 ping statistics ---
20 packets transmitted, 19 received, 5% packet loss, time 19436ms
rtt min/avg/max/mdev = 0.258/0.431/0.738/0.113 ms
mininet-wifi>
```

Figura 3: Repetição do teste de Ping após reconexão de sta1.

### 3.2.2

Usando o comando `iperf sta1 sta2`, pude medir uma banda disponível de 10.5 e 11.4 Mbits/sec entre sta1 e sta2.

```
mininet-wifi> iperf sta1 sta2
*** Iperf: testing TCP bandwidth between sta1 and sta2
*** Results: ['10.5 Mbits/sec', '11.4 Mbits/sec']
mininet-wifi>
```

Figura 4: Banda disponível entre sta1 e sta2.

### 3.2.3

Valendo-se do manual do Mininet-WiFi disponível no GitHub do projeto e de referências da internet, podemos definir a função ou especificidade de cada parâmetro.

1. **txpower** especifica a força do sinal que o roteador produz durante os tempos em que está transmitindo.
2. **wlan** é o nome dado para identificar um computador conectado a uma rede local sem fio.
3. **ip** representa o endereço atribuído a estação.
4. **range** define o alcance da rede sem fio provida pelo ponto de acesso.

```
mininet-wifi> py sta1.params
{'txpower': [14], 'wlan': ['sta1-wlan0'], 'ip': ['10.0.0.1/8'], 'range': [62], '
antennaGain': [5], 'apsInRange': [<OVSAP ap1: lo:127.0.0.1,ap1-wlan1:None pid=42
09> ], 'mac': ['02:00:00:00:00:00'], 'mode': ['g'], 'associatedTo': [<OVSAP ap1:
lo:127.0.0.1,ap1-wlan1:None pid=4209> ], 'antennaHeight': [1.0], 'position': [1
0.0, 2.0, 3.0], 'freq': [2.412], 'channel': ['1']}
mininet-wifi>
```

Figura 5: Lista de parâmetros de sta1.

5. **antennaGain** é uma medida de performance que combina diretividade e eficiência da antena.
6. **apsInRange** representa uma descrição simples dos pontos de acesso, cujos alcances englobam a estação.
7. **mac** é o endereço do dispositivo que especifica a unicidade no acesso a uma rede.
8. **mode** é um padrão IEEE que reúne as especificações de uma rede.
9. **associatedTo** representa uma descrição do ponto de acesso em que a estação está conectada.
10. **antennaHeight** é a medidade de altura da antena.
11. **position** são as medidas que definem a localização espacial da estação.
12. **freq** é a frequência utilizada na rede.
13. **channel** é o canal(intervalo de frequências) utilizado na rede.

### 3.2.4

Em primeiro lugar, fiz um teste na rede cabeada. Criamos a topologia da rede, abrimos o Wireshark em s1 para monitor o fluxo de pacotes, e em seguida enviamos um ping de h1 para h2. Observamos, quanto ao endereço **MAC**, que existem apenas o endereço da fonte de requisição e o endereço de destino.

```
▶ Frame 9: 98 bytes on wire (784 bits), 98 bytes captured (784 bits) on interface 0
▼ Ethernet II, Src: 00:00:00_00:00:01 (00:00:00:00:00:01), Dst: 00:00:00_00:00:02 (00:00:00:00:00:02)
  ▶ Destination: 00:00:00_00:00:02 (00:00:00:00:00:02)
  ▶ Source: 00:00:00_00:00:01 (00:00:00:00:00:01)
    Type: IPv4 (0x0800)
  ▶ Internet Protocol Version 4, Src: 10.0.0.1, Dst: 10.0.0.2
  ▶ Internet Control Message Protocol
```

Figura 6: Informações do pacote 9 na rede LAN.

Antes de darmos continuidade no teste na rede WLAN, é necessário configurar uma interface de monitoramento com o comando `sh ifconfig hwsim0 up`. Após essa etapa, reexecutamos o teste com o monitoramento em ap1. É possível notar que nesse caso que além dos endereços de fonte e destino, existe o endereço **MAC** do transmissor no quadro do protocolo.

```

▶ Frame 20: 138 bytes on wire (1104 bits), 138 bytes captured (1104 bits)
▶ Radiotap Header v0, Length 22
▶ 802.11 radio information
▼ IEEE 802.11 Data, Flags: .....T
  Type/Subtype: Data (0x0020)
  ▶ Frame Control Field: 0x0801
    .000 0000 0011 0100 = Duration: 52 microseconds
    Receiver address: 02:00:00:00:02:00 (02:00:00:00:02:00)
    Transmitter address: 02:00:00:00:01:00 (02:00:00:00:01:00)
    Destination address: 02:00:00:00:00:00 (02:00:00:00:00:00)
    Source address: 02:00:00:00:01:00 (02:00:00:00:01:00)
    BSS Id: 02:00:00:00:02:00 (02:00:00:00:02:00)
    STA address: 02:00:00:00:01:00 (02:00:00:00:01:00)
    .... = Fragment number: 0
    0000 0001 1111 .... = Sequence number: 31
  ▶ Logical-Link Control
  ▶ Internet Protocol Version 4, Src: 10.0.0.2, Dst: 10.0.0.1
  ▶ Internet Control Message Protocol

```

Figura 7: Informações do pacote 20 da rede WLAN.

### 3.3 Questão 3

#### 3.3.1

Ao observar o gráfico de localização espacial dos dispositivos, a estação sta3 aparenta estar fora do alcance do ponto de acesso AP1, consequentemente desconectado.

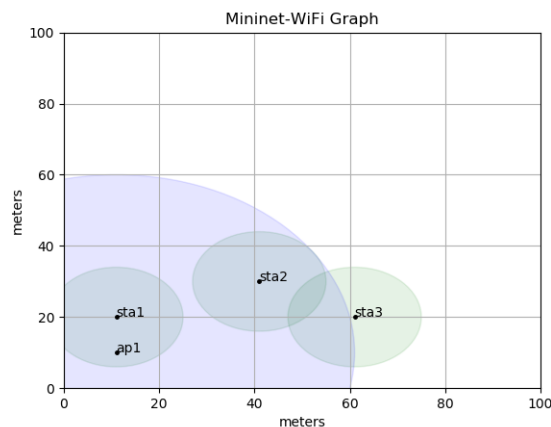


Figura 8: Gráfico de distribuição das estações e do ponto de alcance com respectivo alcance.

Para validar a hipótese, inserimos o comando `py sta3.params['associatedTo']` na linha de comando do Mininet-WiFi. O parâmetro da rede voltou vazio, indicando desconexão com o ponto de acesso.

```

mininet-wifi> py sta1.params['associatedTo']
[<OVSAP ap1: lo:127.0.0.1,ap1-wlan1:None pid=5835> ]
mininet-wifi> py sta2.params['associatedTo']
[<OVSAP ap1: lo:127.0.0.1,ap1-wlan1:None pid=5835> ]
mininet-wifi> py sta3.params['associatedTo']
['']
mininet-wifi> 

```

Figura 9: Parâmetro vazio da estação sta3, indicando desconexão.

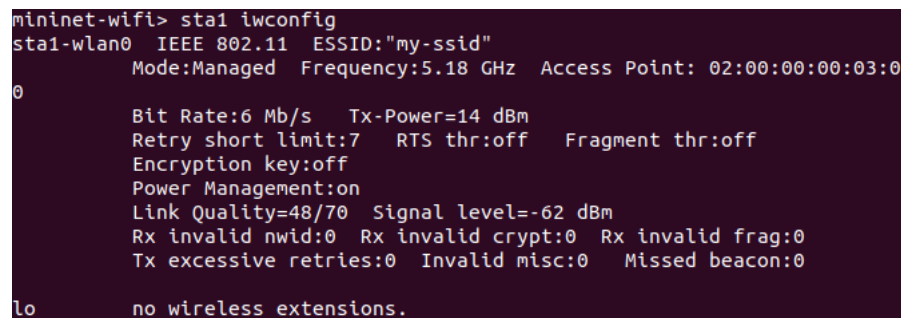
### 3.3.2

O modelo escolhido é o International Telecommunication Union (ITU) Propagation Loss Model. O código original foi modificado da seguinte forma para acomodar o seguinte modelo de propagação.

```
info("*** Creating nodes\n")
net.addStation('sta1 ', position='20,120,0')
net.addStation('sta2 ', position='40,30,0')
net.addStation('sta3 ', position='60,20,0')
net.addAccessPoint('ap1 ', ssid="my-ssid", mode="a", channel="36",
                   failMode='standalone ', position='10,10,0')

info("*** Configuring Propagation Model\n")
net.setPropagationModel(model="ITU", lF = 4, nFloors = 6, pL = 4)
net.plotGraph(max_x=100, max_y=100)
```

Valendo-se dos modelos de propagação implementados, podemos avaliar os níveis de sinal na estação sta1. Com o modelo longDistance, o sinal apresentou nível de -62dBm com qualidade de link 58/70, enquanto com o modelo ITU, o nível foi de -50dBm com qualidade de link de 60/70. Os testes foram realizados mantendo a mesma distância do ponto de acesso à estação.



```
mininet-wifi> sta1 iwconfig
sta1-wlan0 IEEE 802.11 ESSID:"my-ssid"
            Mode:Managed  Frequency:5.18 GHz  Access Point: 02:00:00:00:03:00
            Bit Rate:6 Mb/s   Tx-Power=14 dBm
            Retry short limit:7   RTS thr:off   Fragment thr:off
            Encryption key:off
            Power Management:on
            Link Quality=48/70  Signal level=-62 dBm
            Rx invalid nwid:0  Rx invalid crypt:0  Rx invalid frag:0
            Tx excessive retries:0  Invalid misc:0  Missed beacon:0

lo        no wireless extensions.
```

Figura 10: Qualidade de sinal na estação sta1 usando modelo de Perda de Propagação Log-Distance

A qualidade de sinal superior no modelo ITU se deve ao fato de que para transmitir sinais em lugares com pavimentações, ou seja, atravessar os obstáculos, a potência do sinal irradiado deve ser maior.

### 3.3.3

Esse modelo é aplicável apenas para ambientes fechados, pois seu modelo matemático leva em consideração pavimentações (paredes, pisos) e nível de penetração do sinal, características de ambientes fechados.

$$L = 20 * \log_{10} f + N \log_{10} d + P_f(n) - 28$$

```

mininet-wifi> sta1 iwconfig
lo          no wireless extensions.

sta1-wlan0  IEEE 802.11  ESSID:"my-ssid"
            Mode:Managed  Frequency:5.18 GHz  Access Point: 02:00:00:00:03:0
0
            Bit Rate:6 Mb/s   Tx-Power=14 dBm
            Retry short limit:7   RTS thr:off   Fragment thr:off
            Encryption key:off
            Power Management:on
            Link Quality=60/70  Signal level=-50 dBm
            Rx invalid nwid:0  Rx invalid crypt:0  Rx invalid frag:0
            Tx excessive retries:0  Invalid misc:0  Missed beacon:0

mininet-wifi> 

```

Figura 11: Qualidade de sinal na estação sta1 usando modelo de Perda de Propagação ITU.

Onde:

- $L$  = o caminho total de perda
- $f$  = frequência de transmissão
- $d$  = distância
- $N$  = o coeficiente de perda de potência
- $n$  = número de pavimentações entre o transmissor e o receptor
- $P_f(n)$  = o fator de perda de penetração em pavimentação.

## 3.4 Questão 4

### 3.4.1

Utilizando o modelo de propagação Log-Distance e os comandos `iperf` e `iwconfig`, mesuramos os seguintes dados.

Distância	Largura de Banda	Nível de Sinal
10	15,7	-74dBm
20	11,6	-81dBm
30	7,26	-86dBm
40	4,18	-90dBm

### 3.4.2

CCA é um mecanismo presente nas redes WLAN usado pela camada MAC para determinar se o canal está limpo para transmissão de dados e para determinar quando há dados recebidos.

### 3.5 Questão 5

#### 3.5.1

A conexão entre sta1 e sta2 variou diferentemente de acordo com o deslocamento de sta1. Na primeira iteração, o ping foi alto devido ao reconhecimento dos dispositivos conectados na rede. Posteriormente o ping se manteve constante. Em um momento, sta1 se desconecta do ponto de acesso 1, o que causou o Destination Host Unreachable. Ao reconectar com o ponto de acesso 2, temos um tempo para reconhecimento da nova conexão pelos dispositivos. Após esse tempo, o tempo de ping se estabelece em um valor praticamente constante.

```
mininet-wifi> sta1 ping sta2
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=1.78 ms
64 bytes from 10.0.0.3: icmp_seq=2 ttl=64 time=0.898 ms
64 bytes from 10.0.0.3: icmp_seq=3 ttl=64 time=0.828 ms
From 10.0.0.2 icmp_seq=4 Destination Host Unreachable
From 10.0.0.2 icmp_seq=5 Destination Host Unreachable
From 10.0.0.2 icmp_seq=6 Destination Host Unreachable
64 bytes from 10.0.0.3: icmp_seq=7 ttl=64 time=44.9 ms
64 bytes from 10.0.0.3: icmp_seq=8 ttl=64 time=1.52 ms
64 bytes from 10.0.0.3: icmp_seq=9 ttl=64 time=1.37 ms
64 bytes from 10.0.0.3: icmp_seq=10 ttl=64 time=17.1 ms
64 bytes from 10.0.0.3: icmp_seq=11 ttl=64 time=1.21 ms
64 bytes from 10.0.0.3: icmp_seq=12 ttl=64 time=1.26 ms
64 bytes from 10.0.0.3: icmp_seq=13 ttl=64 time=1.23 ms
64 bytes from 10.0.0.3: icmp_seq=14 ttl=64 time=1.29 ms
64 bytes from 10.0.0.3: icmp_seq=15 ttl=64 time=1.30 ms
64 bytes from 10.0.0.3: icmp_seq=16 ttl=64 time=1.33 ms
64 bytes from 10.0.0.3: icmp_seq=17 ttl=64 time=1.29 ms
64 bytes from 10.0.0.3: icmp_seq=18 ttl=64 time=1.30 ms
64 bytes from 10.0.0.3: icmp_seq=19 ttl=64 time=1.30 ms
64 bytes from 10.0.0.3: icmp_seq=20 ttl=64 time=1.30 ms
64 bytes from 10.0.0.3: icmp_seq=21 ttl=64 time=1.27 ms
64 bytes from 10.0.0.3: icmp_seq=22 ttl=64 time=1.29 ms
64 bytes from 10.0.0.3: icmp_seq=23 ttl=64 time=1.29 ms
64 bytes from 10.0.0.3: icmp_seq=24 ttl=64 time=1.26 ms
64 bytes from 10.0.0.3: icmp_seq=25 ttl=64 time=1.28 ms
64 bytes from 10.0.0.3: icmp_seq=26 ttl=64 time=1.30 ms
64 bytes from 10.0.0.3: icmp_seq=27 ttl=64 time=1.27 ms
```

Figura 12: Ping durante deslocamento de sta1.

#### 3.5.2

O nome do processo é Handoff. Ele acontece quando o sinal fica abaixo de -92dBm, o que promove a desconexão da estação com o ponto de acesso. Essa desconexão gerou o tempo de ping maior.