EA080 - Laboratório de redes de computadores
Relatório 2 - Mininet WiFi

Gabriela Surita
RA: 139095

Introdução

Redes de Internet sem fio foram propostas durante a década de 90, mas se popularizaram na segunda metade da década de 2000, com a popularização dos dispositivos móveis, como Notebooks e Smartphones, elas já configuram a maioria dos acessos à internet obtidos por usuários finais.

Este laboratório propõe analisar o funcionamento da camada de rede em redes sem fio, em especial as redes WiFi (802.11) através da ferramenta de emulação Mininet WiFi [1]. A ferramenta Mininet WiFi incluí modelos de propagação de alcance e propagação redes sem fio modelados sobre a ferramenta Mininet.

Dentre os experimentos trabalhados, procurou-se (1) adquirir familiaridade com as ferramentas exclusivas para análise de redes sem fio e estudar (2) o efeito de diferentes modelos de propagação (para diferentes tipos de ambientes) no alcance da transmissão, (3) o efeito da relação sinal ruído na banda da conexão e (4) o comportamento de dispositivos durante a troca de redes (crossover).

Implementação

O experimento foi conduzido usando uma máquina virtual fornecida usando Ubuntu 16.04 e com o Mininet WiFi já instalado. Verificamos a versão do Mininet como sendo a 2.3.0d1.

```
wifi@wifi-VirtualBox:~$ mn --version
2.3.0d1
```

1. Primeiros passos com o Mininet WiFi

Em seguida, prosseguimos com a inicialização do Mininet WiFi.

```
wifi@wifi-VirtualBox:~$ 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>
```

Um teste simples para a rede é uma chamada de ping, ou seja o envio de requisições ICMP entre as duas estações (sta1 e sta2).

```
mininet-wifi> sta1 ping 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=0.116 ms
64 bytes from 10.0.0.2: icmp_seq=2 ttl=64 time=0.076 ms
64 bytes from 10.0.0.2: icmp_seq=3 ttl=64 time=0.075 ms

(...)

64 bytes from 10.0.0.2: icmp_seq=33 ttl=64 time=0.109 ms

--- 10.0.0.2 ping statistics ---
33 packets transmitted, 33 received, 0% packet loss, time 32066ms
rtt min/avg/max/mdev = 0.053/0.101/0.344/0.055 ms
```

Podemos observar que não houve perda de pacotes, e a latência entre as duas estações é baixa (da ordem de 0.1ms).

Podemos testar a banda disponível na rede usando a ferramenta iperf. Iniciando um servidor na estação sta1 e um cliente na estação sta2, observamos que a banda é da ordem de 7 Mbits/sec.

```
mininet-wifi> iperf sta1 sta2
*** Iperf: testing TCP bandwidth between sta1 and sta2
.*** Results: ['6.87 Mbits/sec', '6.91 Mbits/sec']
```

A ferramenta Mininet WiFi trabalha a partir de emulação da camada de rede, desta maneira, alguns parâmetros de configuração devem ser configurados para representar os dispositivos e a topologia física. Podemos observar a configuração default da primeira estação, por exemplo.

```
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=26544>
      ],
      'mac': ['02:00:00:00:00:00'],
      'frequency': [2.412],
      'mode': ['g'],
      'associatedTo': [
            <OVSAP ap1: lo:127.0.0.1,ap1-wlan1:None pid=26544>
      ],
      'antennaHeight': [1.0],
      'position': [1.0, 0.0, 0.0],
      'channel': ['1'],
      'rssi': [-60]
}
```

Os parâmetros configurados são:

- **txpower**: Potência de transmissão do emissor da interface (em dBmW - decibel miliwatt)
- **wlan**: Nome da interface de rede.
- **ip**: Endereço de IP (v4) da interface na rede (subrede de 8 bits).
- **range**: Alcance da rede.
- **gain**: Relação de potência de entrada/saída ou ganho da antena (em dB).
- **apsInRange**: Access Points, ou pontos de acesso no alcance da interface.
- **mode**: Modo de operação do protocolo 802.11, ou seja, a versão do protocolo Wi-Fi usada.
- **associatedTo**: interfaces de rede associadas a esta no host.
- **antenaHeight**: Altura da antena.
- **position**: Posição da antena.
- **channel**: canal de operação da rede Wifi conectada. Varia entre países, mas em geral é um número entre 1-13.
- **rssi**: Indicador da potência do sinal. Varia entre -100 e 0 e quanto maior o Rssi, maior a potência recebida.

Usando a ferramenta Wireshark, e comparando os quadros 802.11 (WiFi) e 802.3 (Ethernet), vemos que o quadro Ethernet tem contém apenas dois endereços MAC, sendo eles o da interface de origem (Source) e o de destino (Destination). Já os quadros 802.11/g, tem, além dos campos já citados, outros endereços referentes ao ponto de acesso, como os

endereços de transmissor e receptor (Transmitter, Receiver), bem como a estação de acesso (STA).

Os campos adicionais do quadro 802.11 se justificam pois um acesso WiFi geralmente não se dá de maneira direta entre hosts, mas através de um ponto de acesso, desta maneira é necessário que o host enviando o pacote especifique qual ponto de acesso está utilizando, para evitar que outros pontos de acesso capturem o pacote.

Finalmente o Mininet WiFi também possuí uma maneira bastante direta de visualizar uma topologia de rede usando a biblioteca Matplotlib. Podemos visualizar a topologia estudada nesta introdução.

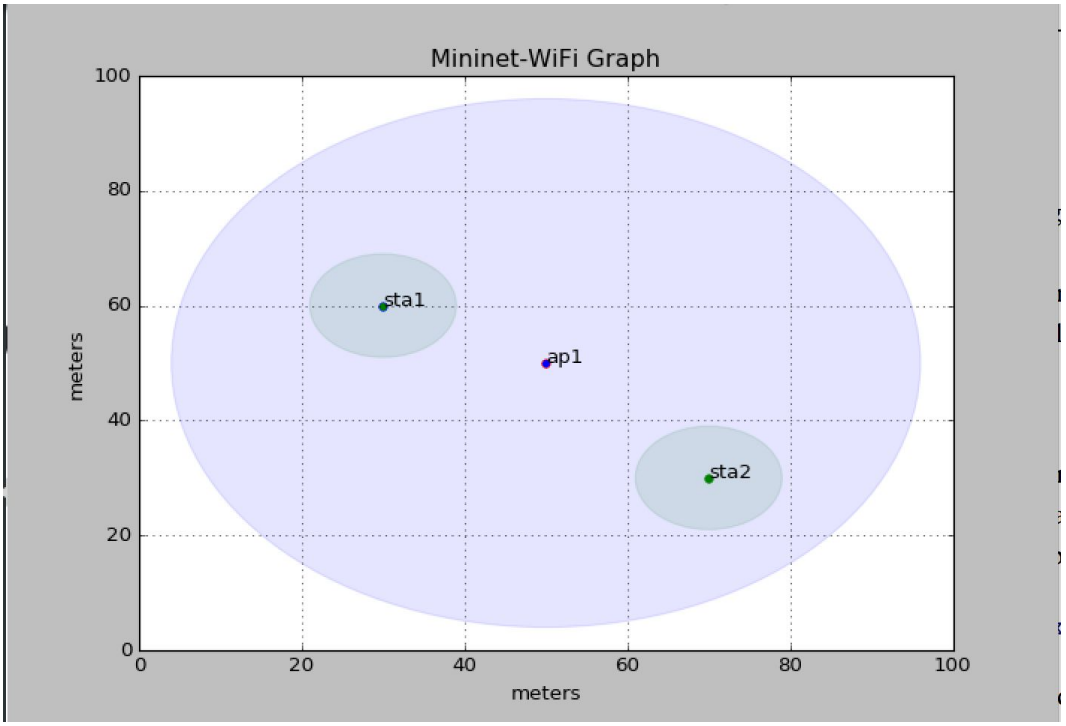

Figura 1: topologia usada na seção de primeiros passos.

2. Modelos de Propagação

Nesta seção, vamos analisar alguns modelos de propagação usados para emular redes físicas implementados no Mininet WiFi. Para isso usaremos o exemplo abaixo, que está disponível no Github [2].

```
#!/usr/bin/python

'This example creates a simple network topology with 1 AP and 3 stations'

from mininet.log import setLogLevel, info
from mn_wifi.net import Mininet_wifi
```

```
from mn_wifi.cli import CLI_wifi
from mn_wifi.link import wmediumd
from mn_wifi.wmediumdConnector import interference


def topology():
    "Create a network."
    net = Mininet_wifi(link=wmediumd, wmediumd_mode=interference)

    info("*** Creating nodes\n")
    net.addStation('sta1', position='10,20,0')
    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.propagationModel(model="logDistance", exp=4)

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

    info("*** Configuring wifi nodes\n")
    net.configureWifiNodes()

    info("*** Starting network\n")
    net.build()

    info("*** Running CLI\n")
    CLI_wifi(net)

    info("*** Stopping network\n")
    net.stop()

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

Através do comando plot, podemos observar a topologia da rede. Vemos claramente que as estações sta1, e sta2 estão dentro da área de cobertura de ap1, enquanto a sta3 está fora da área de cobertura do ponto de acesso.

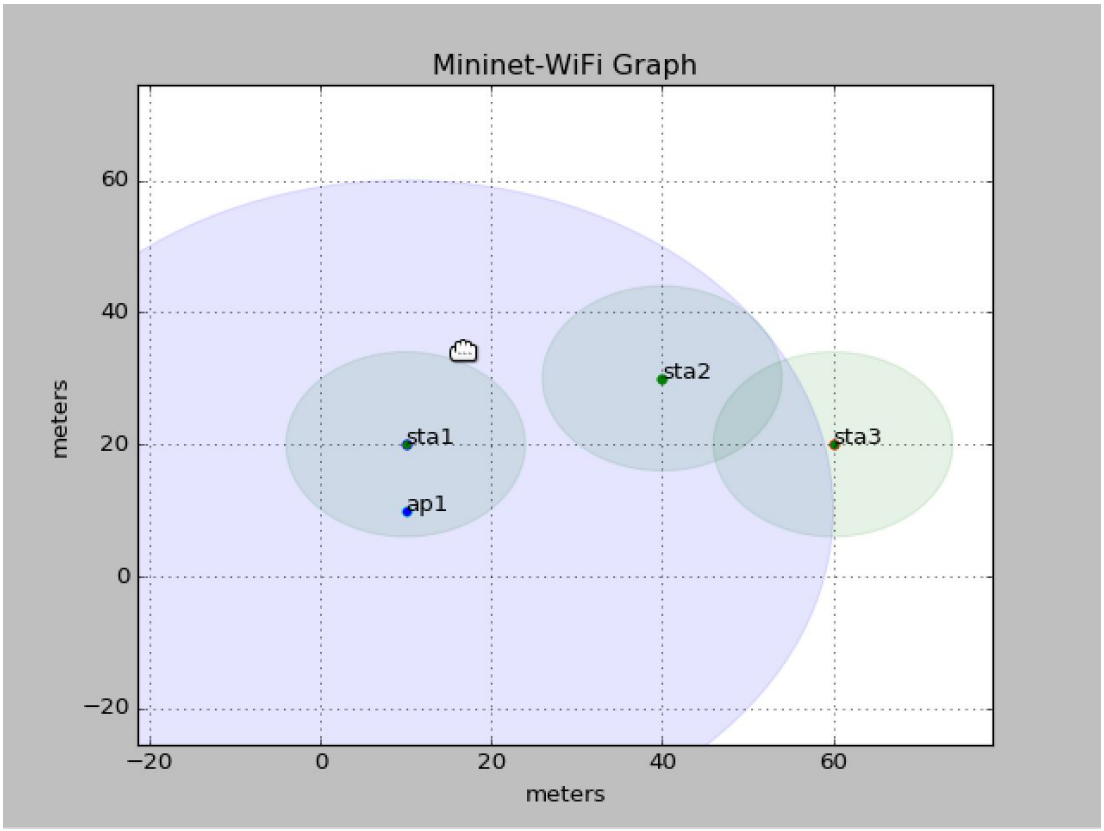

Figura 2: Topologia usada na seção 2 usando Log-Distance Propagation Model com fator de decaimento 4.

podemos obter as consequências do desenho da topologia observando as configurações de rede das estações usando o comando iwconfig. Como, esperado nível de sinal percebido por sta1 é o maior, com -62 dBm, sta2 tem -85 dBm e sta3 não está conectado ao ponto de acesso.

```
mininet-wifi> sta1 iwconfig
lo    no wireless extensions.

sta1-wlan0  IEEE 802.11abgn  ESSID:"my-ssid"
      Mode:Managed  Frequency:5.18 GHz  Access Point: 02:00:00:00:03:00
      Bit Rate:24 Mb/s   Tx-Power=14 dBm
      Retry short limit:7   RTS thr:off   Fragment thr:off
      Encryption key:off
      Power Management:off
      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:15   Missed beacon:0

mininet-wifi> sta2 iwconfig
lo    no wireless extensions.

sta2-wlan0  IEEE 802.11abgn  ESSID:"my-ssid"
      Mode:Managed  Frequency:5.18 GHz  Access Point: 02:00:00:00:03:00
```

```
      Bit Rate:24 Mb/s   Tx-Power=14 dBm
      Retry short limit:7   RTS thr:off   Fragment thr:off
      Encryption key:off
      Power Management:off
      Link Quality=25/70  Signal level=-85 dBm
      Rx invalid nwid:0  Rx invalid crypt:0  Rx invalid frag:0
      Tx excessive retries:0  Invalid misc:8   Missed beacon:0

mininet-wifi> sta3 iwconfig
lo    no wireless extensions.

sta3-wlan0  IEEE 802.11abgn  ESSID:off/any
      Mode:Managed  Access Point: Not-Associated   Tx-Power=20 dBm
      Retry short limit:7   RTS thr:off   Fragment thr:off
      Encryption key:off
      Power Management:off
```

O modelo utilizado neste primeiro exemplo foi o “Log-distance path loss model”, que é um modelo empírico para perdas por dissipação em ambientes fechados. Para efeito de comparação, estudamos outros modelos de propagação e dissipação implementados no Mininet WiFi. Segundo a documentação, os modelos de propagação implementados são:

- Modelos de propagação indoor (ambientes fechados):
    - Free-Space Propagation Model
    - Log-Distance Propagation Model
    - International Telecommunication Union (ITU) Propagation Model

- Modelos de propagação outdoor(espaços livres):
    - Two-Ray-Ground Propagation Model

No entanto, na base de código do Mininet WiFi [3], foi possível encontrar alguns outros modelos de propagação que não estavam documentados. Alguns deles são:

- Modelos de propagação indoor (ambientes fechados):
    - Log-distance path loss model with normal shadowing

- Modelos de propagação outdoor(espaços livres):
    - Friis Free Space Propagation Model

Como verificamos, o modelo testado no exemplo está na classe dos modelos de propagação em ambiente fechado. Foi escolhido o modelo “Friis Propagation Loss Model” para verificar as consequências da alteração na forma de cálculo da dissipação do sinal para um modelo em espaço aberto. O modelo de Friis é útil neste caso pois considera sinais propagados por antenas tridimensionais em espaço aberto e usando antenas de mesma polarização. O exemplo foi alterado usando a seguinte linha.

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

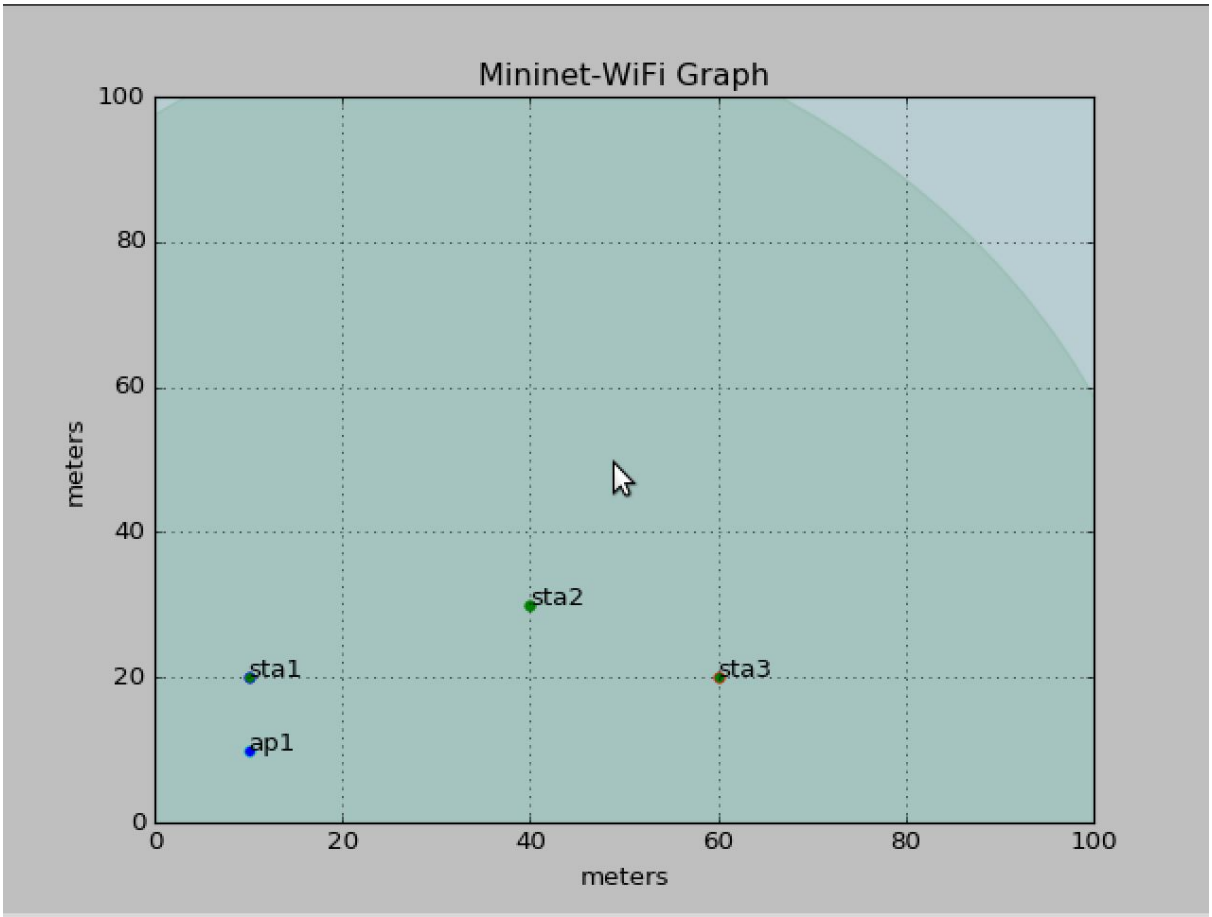

Figura 2: Topologia usada na seção 2 usando Friis Free Space Propagation Model.

A latência dos terminais também são consideravelmente maiores. Vemos que, em ambientes livres de obstáculos, o alcance de um ponto de acesso é muito maior que em ambientes fechados, o que é esperado.

```
mininet-wifi> sta1 iwconfig
lo    no wireless extensions.

sta1-wlan0  IEEE 802.11abgn  ESSID:"my-ssid"
      Mode:Managed  Frequency:5.18 GHz  Access Point: 02:00:00:00:03:00
      Bit Rate:24 Mb/s   Tx-Power=14 dBm
      Retry short limit:7   RTS thr:off   Fragment thr:off
      Encryption key:off
      Power Management:off
      Link Quality=68/70  Signal level=-42 dBm
      Rx invalid nwid:0  Rx invalid crypt:0  Rx invalid frag:0
      Tx excessive retries:0  Invalid misc:7   Missed beacon:0

mininet-wifi> sta2 iwconfig
lo    no wireless extensions.

sta2-wlan0  IEEE 802.11abgn  ESSID:"my-ssid"
```

```
      Mode:Managed  Frequency:5.18 GHz  Access Point: 02:00:00:00:03:00
      Bit Rate:24 Mb/s   Tx-Power=14 dBm
      Retry short limit:7   RTS thr:off   Fragment thr:off
      Encryption key:off
      Power Management:off
      Link Quality=57/70  Signal level=-53 dBm
      Rx invalid nwid:0  Rx invalid crypt:0  Rx invalid frag:0
      Tx excessive retries:0  Invalid misc:8   Missed beacon:0

mininet-wifi> sta3 iwconfig
lo    no wireless extensions.

sta3-wlan0  IEEE 802.11abgn  ESSID:"my-ssid"
      Mode:Managed  Frequency:5.18 GHz  Access Point: 02:00:00:00:03:00
      Bit Rate:24 Mb/s   Tx-Power=14 dBm
      Retry short limit:7   RTS thr:off   Fragment thr:off
      Encryption key:off
      Power Management:off
      Link Quality=54/70  Signal level=-56 dBm
      Rx invalid nwid:0  Rx invalid crypt:0  Rx invalid frag:0
      Tx excessive retries:0  Invalid misc:8   Missed beacon:0
```

3. Relação Distância - Largura de banda

Ainda sobre dissipação de sinais, é natural assumir que quanto mais baixa a relação de sinal ruído, menor é a banda útil da conexão, podemos usar uma topologia arbitrária para verificar o efeito da distância na largura de banda.

```
info("*** Creating nodes\n")
net.addStation('sta1', position='10,10.1,0')
net.addStation('sta2', position='10,20,0')
net.addStation('sta3', position='10,30,0')
net.addStation('sta4', position='10,40,0')
net.addStation('sta5', position='10,50,0')
net.addAccessPoint('ap1', ssid="my-ssid", mode="a", channel="36",
                   failMode='standalone', position='10,10,0')
```

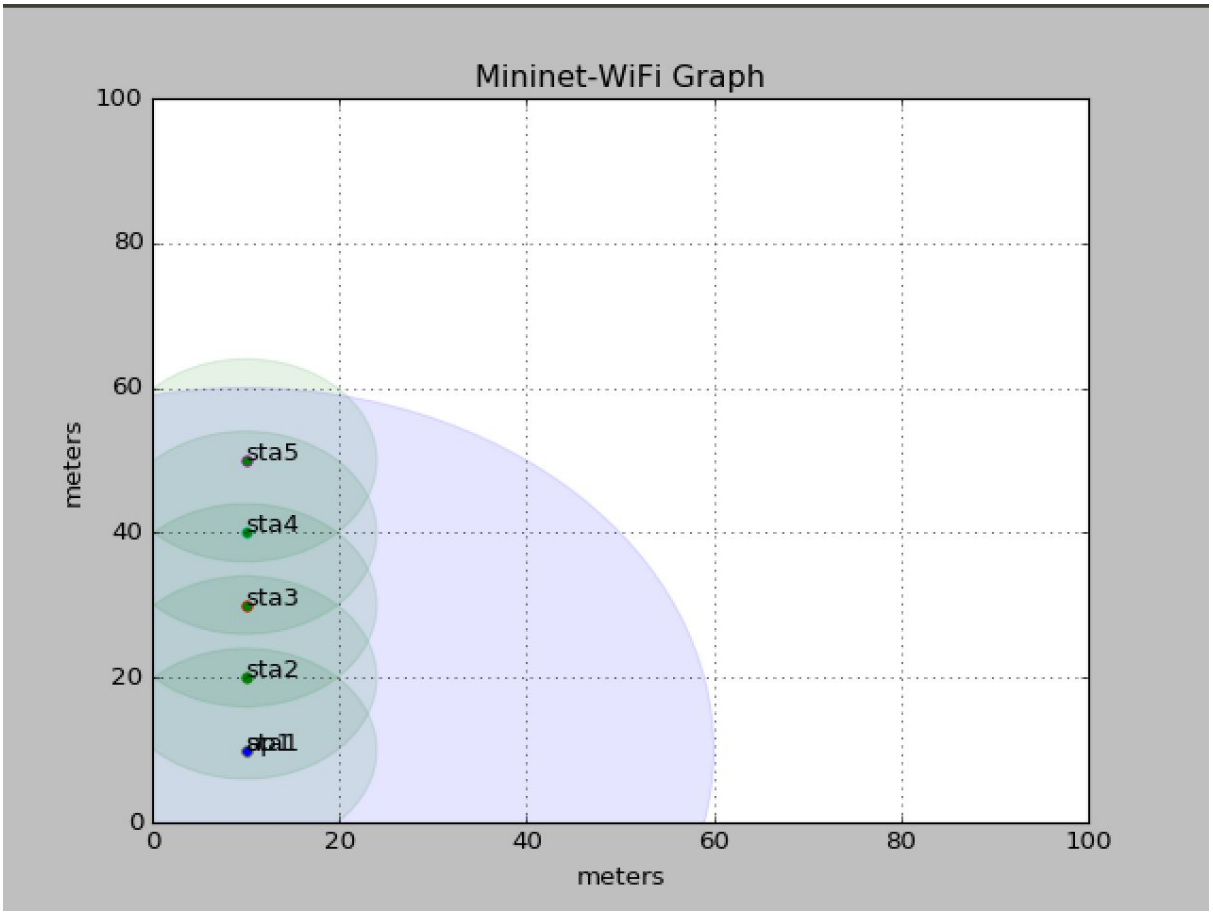

Figura 3: Topologia da rede usada na seção 3.

Podemos usar a ferramenta iperf para medir a banda. Como esperado, os nós próximos tem uma banda consideravelmente maior que os distantes. Podemos verificar também que a relação de banda é sobre-linear, ainda que não tenhamos pontos suficientes para fazer alguma inferência da relação.

```
mininet-wifi> iperf sta1 sta2
*** Iperf: testing TCP bandwidth between sta1 and sta2
*** Results: ['16.6 Mbits/sec', '16.7 Mbits/sec']
mininet-wifi> iperf sta1 sta3
*** Iperf: testing TCP bandwidth between sta1 and sta3
*** Results: ['16.3 Mbits/sec', '16.4 Mbits/sec']
mininet-wifi> iperf sta1 sta4
*** Iperf: testing TCP bandwidth between sta1 and sta4
*** Results: ['9.32 Mbits/sec', '9.36 Mbits/sec']
mininet-wifi> iperf sta1 sta5
*** Iperf: testing TCP bandwidth between sta1 and sta5
*** Results: ['4.26 Mbits/sec', '4.36 Mbits/sec']
```

4. Handover

Nesta última seção, verificamos qual o comportamento durante uma transição de uma estação entre dois pontos de acesso.

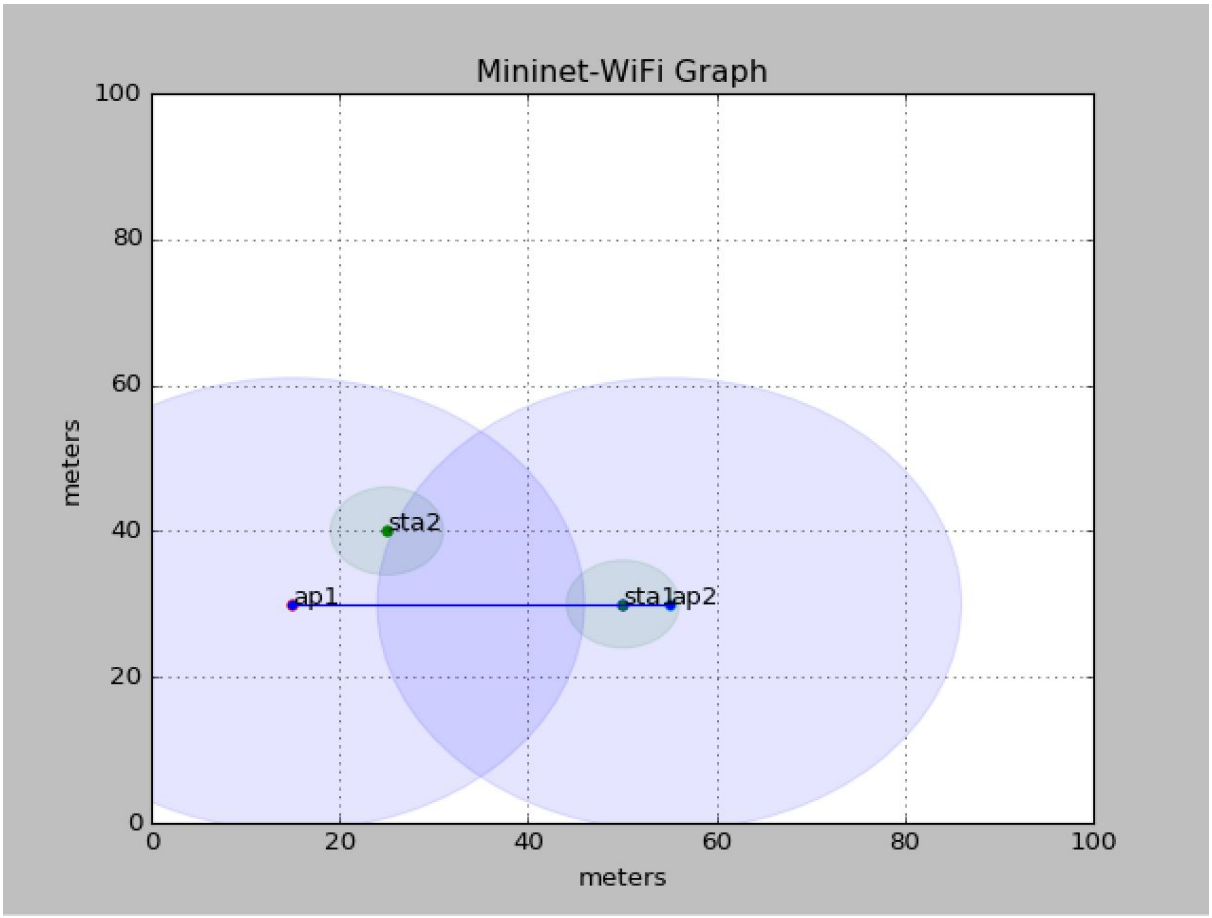

Figura 4: Topologia usada na seção 4. Neste exemplo a estação sta1 percorre o caminho delineado em azul.

Um comando ping foi executado durante o deslocamento de sta1. Podemos verificar que a latência da chamada cresce gradativamente até que deixam de ser recebidos pacotes. Houve perda de pacotes depois da troca de ponto de acesso. Podemos verificar também que sta1 está conectado ao ponto de acesso 2.

Pode parecer contra-intuitivo, a princípio, que a conexão não fosse restabelecida após a troca de ponto de acesso. No entanto, no exemplo, não há conexão entre os dois pontos de acesso. Vemos ainda, que o ip no qual o ping está sendo realizado pertence a subrede de ap1, que não pertence mais à estação 2 após a transição.

```
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=2.85 ms
64 bytes from 10.0.0.3: icmp_seq=2 ttl=64 time=1.66 ms
```

```
64 bytes from 10.0.0.3: icmp_seq=3 ttl=64 time=1.75 ms
64 bytes from 10.0.0.3: icmp_seq=4 ttl=64 time=1.88 ms
64 bytes from 10.0.0.3: icmp_seq=5 ttl=64 time=2.33 ms
64 bytes from 10.0.0.3: icmp_seq=6 ttl=64 time=1.94 ms
64 bytes from 10.0.0.3: icmp_seq=7 ttl=64 time=2.20 ms
64 bytes from 10.0.0.3: icmp_seq=8 ttl=64 time=2.12 ms
64 bytes from 10.0.0.3: icmp_seq=9 ttl=64 time=2.23 ms
64 bytes from 10.0.0.3: icmp_seq=10 ttl=64 time=2.24 ms
64 bytes from 10.0.0.3: icmp_seq=11 ttl=64 time=2.35 ms
64 bytes from 10.0.0.3: icmp_seq=12 ttl=64 time=2.68 ms
64 bytes from 10.0.0.3: icmp_seq=13 ttl=64 time=2.57 ms
64 bytes from 10.0.0.3: icmp_seq=14 ttl=64 time=2.95 ms
^C
--- 10.0.0.3 ping statistics ---
32 packets transmitted, 14 received, 56% packet loss, time 31042ms
rtt min/avg/max/mdev = 1.662/2.271/2.953/0.379 ms
mininet-wifi> sta1 ping sta2


mininet-wifi> sta1 iwconfig
lo    no wireless extensions.

sta1-wlan0  IEEE 802.11abgn  ESSID:"ssid-ap2"
      Mode:Managed  Frequency:2.437 GHz  Access Point: 02:00:00:00:03:00
      Bit Rate:24 Mb/s   Tx-Power=14 dBm
      Retry short limit:7   RTS thr:off   Fragment thr:off
      Encryption key:off
      Power Management:off
      Link Quality=70/70  Signal level=-36 dBm
      Rx invalid nwid:0  Rx invalid crypt:0  Rx invalid frag:0
      Tx excessive retries:0  Invalid misc:0   Missed beacon:0
```

Anexos

Anexo 1: Traceback do erro obtido ao tentar iniciar o programa com o modelo de propagação “logNormalShadowing”.

```
net.propagationModel(model="logNormalShadowing", exp=4)

wifi@wifi-VirtualBox:~$ sudo python propagation_model.py
*** Creating nodes
*** Configuring Propagation Model
*** Configuring wifi nodes
Traceback (most recent call last):
  File "propagation_model.py", line 43, in <module>
      topology()
  File "propagation_model.py", line 30, in topology
      net.configureWifiNodes()
```

```
  File
"/usr/local/lib/python2.7/dist-packages/mininet_wifi-2.3.0d1-py2.7.egg/mn_w
ifi/net.py", line 2017, in configureWifiNodes
      node.params['range'][wlan] = node.getRange(intf=intf)
  File
"/usr/local/lib/python2.7/dist-packages/mininet_wifi-2.3.0d1-py2.7.egg/mn_w
ifi/node.py", line 305, in getRange
      value = GetSignalRange(self, wlan, interference_enabled)
  File
"/usr/local/lib/python2.7/dist-packages/mininet_wifi-2.3.0d1-py2.7.egg/mn_w
ifi/propagationModels.py", line 221, in __init__
      interference=enable_interference)
  File
"/usr/local/lib/python2.7/dist-packages/mininet_wifi-2.3.0d1-py2.7.egg/mn_w
ifi/propagationModels.py", line 280, in logNormalShadowing
      from mininet.wifi.wmediumdConnector import WmediumdGRandom, \
ImportError: No module named wifi.wmediumdConnector
```

Referências

[1] R. R. Fontes, S. Afzal, S. H. B. Brito, M. A. S. Santos and C. E. Rothenberg, "Mininet-WiFi: Emulating software-defined wireless networks," *2015 11th International Conference on Network and Service Management (CNSM)*, Barcelona, 2015, pp. 384-389. doi: 10.1109/CNSM.2015.7367387

[2] Mininet WiFi, Exemplo de código para estudo de modelos de propagação. https://github.com/ramonfontes/reproducible-research/blob/master/mininet-wifi/UNICAMP-2017/propagation_model.py

[3] Mininet WiFi, Implementação dos modelos de propagação. https://github.com/intrig-unicamp/mininet-wifi/blob/master/mn_wifi/propagationModels.py