# 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

#### 3.1.1

Após executar a topologia do arquivo **topo_A3.py** e verificar a criação do nós e links, um novo terminal foi aberto e o comando `sudo ps aux | grep quagga` executado. Constatei que 12 processos relacionados ao quagga estavam sendo executados.

```
leonardo@leonardo-PC:~/Unicamp/EA080/lab-4/code-git$ ps aux | grep quagga
quagga   17763  0.0  0.0  27804  3244 ?        Ss   11:11   0:00 /usr/sbin/zebra
quagga   17765  0.0  0.0  29936  3092 ?        Ss   11:11   0:00 /usr/sbin/ospfd
quagga   17846  0.0  0.0  27804  3340 ?        Ss   11:11   0:00 /usr/sbin/zebra
quagga   17851  0.0  0.0  29936  3076 ?        Ss   11:11   0:00 /usr/sbin/ospfd
quagga   17917  0.0  0.0  27808  3428 ?        Ss   11:11   0:00 /usr/sbin/zebra
quagga   17919  0.0  0.0  29936   976 ?        Ss   11:11   0:00 /usr/sbin/ospfd
quagga   18121  0.0  0.0  27800  3320 ?        Ss   11:12   0:00 /usr/sbin/zebra
quagga   18123  0.0  0.0  29932  3168 ?        Ss   11:12   0:00 /usr/sbin/ospfd
quagga   18125  0.0  0.0  27804  3348 ?        Ss   11:12   0:00 /usr/sbin/zebra
quagga   18127  0.0  0.0  29932   972 ?        Ss   11:12   0:00 /usr/sbin/ospfd
quagga   18153  0.0  0.0  27804  3292 ?        Ss   11:12   0:00 /usr/sbin/zebra
quagga   18155  0.0  0.0  29932   972 ?        Ss   11:12   0:00 /usr/sbin/ospfd
leonardo 18536  0.0  0.0  21532  1152 pts/22   S+   11:12   0:00 grep --color=aut
leonardo@leonardo-PC:~/Unicamp/EA080/lab-4/code-git$
```

Figura 1: Processos Quagga.

### 3.1.2

Quagga é um suíte composta por um daemon principal chamada zebra e um daemon adicional responsável pelo roteamento dinâmico, no caso ospfd. Existem 12 processos relacionados ao Quagga, no qual 6 estão associados ao Zebra e 6 ao Ospfd. Cada algoritmo de roteamento exige um processo da Zebra e um processo do Ospfd, oque nos permite concluir que 6 algoritmos de roteamento estão sendo executado.

### 3.1.3

Ao executar o comando `x1 ping -c3 y1`, obtivemos perda de 100% dos pacotes. Para o comando `x1 tracepath y1`, não obtivemos uma alcance para o endereço IP de **y1**. Isso mostra que não há conectividade entre x1 e y1.

```
mininet> x1 ping -c3 y1
PING 10.0.12.1 (10.0.12.1) 56(84) bytes of data.
From 10.0.2.21 icmp_seq=1 Destination Net Unreachable
From 10.0.2.21 icmp_seq=2 Destination Net Unreachable
From 10.0.2.21 icmp_seq=3 Destination Net Unreachable

--- 10.0.12.1 ping statistics ---
3 packets transmitted, 0 received, +3 errors, 100% packet loss, time 2032ms

mininet>
```

Figura 2: Comando Ping entre x1 e y1.

```
mininet> x1 tracepath y1
 1?: [LOCALHOST]                      pmtu 1500
 1:  _gateway                                              0.175ms !N
 1:  _gateway                                              0.147ms !N
     Resume: pmtu 1500
mininet>
```

Figura 3: Comando Tracepath entre x1 e y1.

## 3.2 Questão 2

### 3.2.1

Na topologia experimental com Protocolo OSPF, existe apenas 1 sub-rede: 10.0.0.0/23.

### 3.2.2

A rota 10.0.6.0 difere das outras rotas pelos seguintes motivos: há um gateway especificado diferente de 0.0.0.0, o campo de flags mostrado está UG (up and gateway) e o campo metric está com o valor 20.

```
root@wifi-VirtualBox:~/lab4# route -n
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
10.0.2.0        0.0.0.0         255.255.254.0   U     0      0        0 r1-eth4
10.0.4.0        0.0.0.0         255.255.254.0   U     0      0        0 r1-eth1
10.0.6.0        10.0.4.22       255.255.254.0   UG    20     0        0 r1-eth1
10.0.10.0       0.0.0.0         255.255.254.0   U     0      0        0 r1-eth3
root@wifi-VirtualBox:~/lab4#
```

Figura 4: Tabela de roteamento IP do roteador 1.

### 3.2.3

Analisando a topologia da figura 1 do roteiro 4, é possível observar que o roteador 1 possui dois roteadores vizinhos (2 e 5) e um host. Os endereços das interfaces dos roteadores vizinhos são **r1-eht1: 10.0.4.21 (2); r1-eth3: 10.0.10.21 (5)**. O roteador **2** está conectado pela interface **10.0.4.22** e o roteador **5** está conectado pela interface **10.0.10.25**. Entretanto ao executarmos os comandos `sh ip ospf route` e `sh ip ospf neighbor`, obtivemos uma tabela de roteamento indicando as rotas para sub-redes aos quais os roteadores vizinhos estão inseridos e uma tabela de roteamento externo em que apenas o roteador vizinho 2 é detalhado com informações de interface. Essas informações de interface condizem com as informações da topologia da figura 1 do rotiero 4.

```
ospfd-r1# sh ip ospf route
============ OSPF network routing table ============
N    10.0.2.0/23          [10] area: 0.0.0.0
                          directly attached to r1-eth4
N    10.0.4.0/23          [10] area: 0.0.0.0
                          directly attached to r1-eth1
N    10.0.6.0/23          [20] area: 0.0.0.0
                          via 10.0.4.22, r1-eth1
N    10.0.10.0/23         [10] area: 0.0.0.0
                          directly attached to r1-eth3

============ OSPF router routing table =============

============ OSPF external routing table ===========

ospfd-r1# sh ip ospf neighbor

    Neighbor ID Pri State          Dead Time Address         Interface           RXmtL RqstL DBsmL
10.0.6.22         1 Full/DR          35.178s 10.0.4.22       r1-eth1:10.0.4.21       0     0     0
ospfd-r1#
```

Figura 5: Tabelas de Roteamento de R1.

#### 3.2.4

Ambos os comandos `route n` e `sh ip ospf route` se assemelham ao mostrarem aspectos idênticos na atribuição de endereços IP as rotas e interfaces.

#### 3.2.5

Com base na topologia da figura 4, concluímos que o roteador 2 é responsável pela rota diferente.

### 3.3 Questão 3

#### 3.3.1

Aos executarmos o mesmos comandos de r1 para r2, obtemos duas rotas diferentes: **10.0.2.0 e 10.0.10 .0**. Essas rotas informam um endereço IP de gateway e apresentam a flag UG ao invés de U. Provavelmente, são rotas externas conhecidas pelo roteador 2.

```
root@wifi-VirtualBox:~/lab4# route -n
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
10.0.2.0        10.0.4.21       255.255.254.0   UG    20     0        0 r2-eth2
10.0.4.0        0.0.0.0         255.255.254.0   U     0      0        0 r2-eth2
10.0.6.0        0.0.0.0         255.255.254.0   U     0      0        0 r2-eth1
10.0.10.0       10.0.4.21       255.255.254.0   UG    20     0        0 r2-eth2
```

Figura 6: Tabela de Roteamento de R2.

#### 3.3.2

Considerando apenas as tabelas obtidas pelos comandos, concluímos que o roteador 2 possui apenas 1 vizinho. O endereço da interface do roteador vizinho é **r2-eth:10.0.4.22** e ele está conectado pela interface **10.0.4.21**.

```
ospfd-r2# sh ip ospf route
============ OSPF network routing table ============
N    10.0.2.0/23           [20] area: 0.0.0.0
                           via 10.0.4.21, r2-eth2
N    10.0.4.0/23           [10] area: 0.0.0.0
                           directly attached to r2-eth2
N    10.0.6.0/23           [10] area: 0.0.0.0
                           directly attached to r2-eth1
N    10.0.10.0/23          [20] area: 0.0.0.0
                           via 10.0.4.21, r2-eth2

============ OSPF router routing table =============

============ OSPF external routing table ===========

ospfd-r2# sh ip ospf neighbor

    Neighbor ID Pri State           Dead Time Address         Interface            RXmtL RqstL DBsmL
10.0.4.21         1 Full/Backup       37.111s 10.0.4.21       r2-eth2:10.0.4.22        0     0     0
ospfd-r2#
```

Figura 7: Caption

#### 3.3.3

Ambos os comandos `route n` e `sh ip ospf route` se assemelham ao mostrarem aspectos idênticos na atribuição de endereços IP as rotas e interfaces.

#### 3.3.4

Com base na topologia da figura 4, concluímos que o roteador 1 é responsável pelas rotas diferentes.

#### 3.3.5

O pacote OSPF enviado é do tipo Hello. O endereço listado em Neighbor list é 10.0.6.22. Ele representa uma interface vizinha r2-eth1 em relação a interface r2-eth2 que está se comunicando com a interface r1-eth1: 10.0.4.21 através de um multicast.

```
root@wifi-VirtualBox:~/lab4# timeout 10 tcpdump -i r2-eth2 -vvlan
tcpdump: listening on r2-eth2, link-type EN10MB (Ethernet), capture size 262144
bytes
16:39:11.640402 IP (tos 0xc0, ttl 1, id 9559, offset 0, flags [none], proto OSPF
 (89), length 68)
    10.0.4.22 > 224.0.0.5: OSPFv2, Hello, length 48
        Router-ID 10.0.6.22, Backbone Area, Authentication Type: none (0)
        Options [External]
          Hello Timer 10s, Dead Timer 40s, Mask 255.255.254.0, Priority 1
          Designated Router 10.0.4.22, Backup Designated Router 10.0.4.21
          Neighbor List:
            10.0.4.21
16:39:11.640527 IP (tos 0xc0, ttl 1, id 10590, offset 0, flags [none], proto OSP
F (89), length 68)
    10.0.4.21 > 224.0.0.5: OSPFv2, Hello, length 48
        Router-ID 10.0.4.21, Backbone Area, Authentication Type: none (0)
        Options [External]
          Hello Timer 10s, Dead Timer 40s, Mask 255.255.254.0, Priority 1
          Designated Router 10.0.4.22, Backup Designated Router 10.0.4.21
          Neighbor List:
            10.0.6.22

2 packets captured
2 packets received by filter
0 packets dropped by kernel
```

Figura 8: Captura de Pacotes OSPF em r2-eth2.

### 3.4 Questão 4

#### 3.4.1

Ao executarmos os comandos para solicitar as tabelas de roteamento dos roteadores r3, r4, r5 e r6, não foi possível obter nenhuma informação por falta de configuração nos arquivos do diretório confs/.


Figura 9: Tabela de Roteamento de r3.

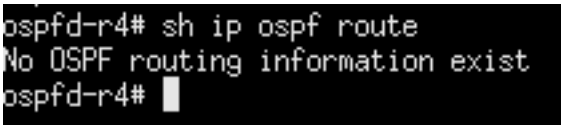


Figura 10: Tabela de Roteamento de r4.

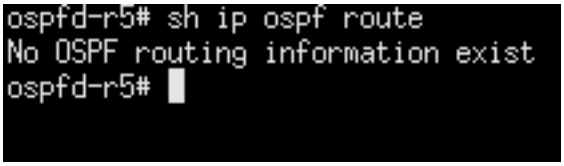


Figura 11: Tabela de Roteamento de r5.

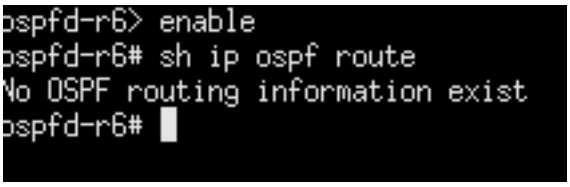


Figura 12: Tabela de Roteamento de r6.

Tomando como base o rotador r3, foi acrescentando duas linhas de configuração no arquivo confs/r1/ospfd-r3.conf: `network 10.0.8.23/23 area 0 network 10.0.6.23/23 area 0`.

```
hostname ospfd-r3
password quagga
!
router ospf
        network 10.0.8.23/23 area 0
        network 10.0.6.23/23 area 0
!
```

Figura 13: Arquivo de confiuração de r3 pós-edição.

Após edição e reexecução do mininet, é possível ver a tabela de roteamento de r3.


Figura 14: Tabela de Roteamento de r3 pós-edição.

A rota de rede para 10.0.2.0/23 possui custo 30 pois é necessário passar por **3** roteadores até chegar na sub-rede em questão, sendo que cada relação de transmissão do roteador vale 10.

## 3.5 Questão 05

Após as configurações dos roteadores r4, r5 e r6, a conexão entre x1 e y1 foi testada. O teste mostrou que a conexão entre os hosts acontece com 4 saltos entre roteadores. Posteriormente, foi adicionado peso com o comando `ospf cost 100`. Com o peso, foi observado uma diminuição no RTT entre os hosts especificados. Nesse caso, o número de saltos aumentou de 4 para 5, indicando que o caminho do p