



EA080 - O — Introdução ao ambiente de emulação de redes Mininet

Professor: Christian Esteve Rothenberg

Leonardo Rodrigues Marques RA: 178610

Atividade 1

a)

```
leonardo@leonardo-PC:~$ sudo mn --mac
[sudo] senha para leonardo:
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2
*** Adding switches:
s1
*** Adding links:
(h1, s1) (h2, s1)
*** Configuring hosts
h1 h2
*** Starting controller
c0
*** Starting 1 switches
s1 ...
*** Starting CLI:
mininet-wifi> 
```

Figura 1: Comando para iniciar topologia padrão.

b)

```
*** Starting CLI:
mininet-wifi> nodes
available nodes are:
c0 h1 h2 s1
mininet-wifi> intfs
h1: h1-eth0
h2: h2-eth0
s1: lo,s1-eth1,s1-eth2
c0:
mininet-wifi> 
```

Figura 2: Comando para verificação de nodos e interfaces.

c)

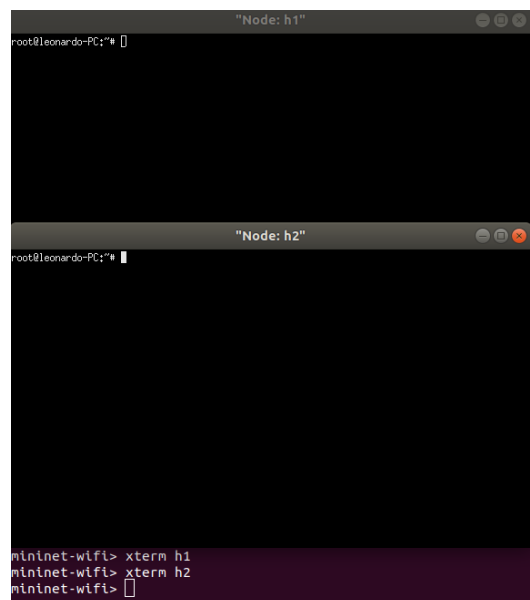


Figura 3: Terminais dos nodos h1 e h2.

d)

```
mininet-wifi> h1 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=33.0 ms

--- 10.0.0.2 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 33.013/33.013/33.013/0.000 ms
mininet-wifi> 
```

Figura 4: Comando para teste de ping entre h1 e h2.

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	10.0.0.1	10.0.0.2	ICMP	98	Echo (ping) request id=0x426e, seq=1/256, ttl=64 (reply in 2)
2	0.000017526	10.0.0.2	10.0.0.1	ICMP	98	Echo (ping) reply id=0x426e, seq=1/256, ttl=64 (request in 1)
3	5.168610877	00:00:00:00:00:02	00:00:00:00:00:01	ARP	42	Who has 10.0.0.1? Tell 10.0.0.2
4	5.169204042	00:00:00:00:00:01	00:00:00:00:00:02	ARP	42	10.0.0.1 is at 00:00:00:00:00:01
5	5.177190147	00:00:00:00:00:01	00:00:00:00:00:02	ARP	42	Who has 10.0.0.2? Tell 10.0.0.1
6	5.177242534	00:00:00:00:00:02	00:00:00:00:00:01	ARP	42	10.0.0.2 is at 00:00:00:00:00:02
7	6.448940401	fe80::b85c:6aff:fec... ff02::2		ICMPv6	70	Router Solicitation from ba:5c:6a:c1:eb:c4
8	14.739488679	fe80::b85c:6aff:fec... ff02::fb		MDNS	203	Standard query 0x0000 PTR _nfs._tcp.local, "QM" question PTR _ipp.

Figura 5: Monitoramento de interface do host h2 pelo Wireshark.

e)

e.(1-2) - h1)

Endereço IP: 10.0.0.1

Endereço MAC: 00:00:00:00:00:01

```
h1-eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
inet 10.0.0.1 netmask 255.0.0.0 broadcast 10.255.255.255
inet6 fe80::200:ff:fe00:1 prefixlen 64 scopeid 0x20<link>
ether 00:00:00:00:00:01 txqueuelen 1000 (Ethernet)
RX packets 38 bytes 4842 (4.8 KB)
RX errors 0 dropped 0 overruns 0 frame 0
TX packets 12 bytes 996 (996.0 B)
TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```

Figura 6: Endereços IP e MAC do host h1.

e.3 - h1)

```
mininet-wifi> h1 route
Tabela de Roteamento IP do Kernel
Destino      Roteador      MáscaraGen.   Opções Métrica Ref  Uso Iface
10.0.0.0     0.0.0.0       255.0.0.0    U      0      0      0 h1-eth0
mininet-wifi>
```

Figura 7: Tabela de roteamento IP do host h1.

e.4 - h1)

```
mininet-wifi> h1 arp
Endereço      TipoHW  EndereçoHW    Flags Mascara      Iface
10.0.0.2      ether   00:00:00:00:00:02 C          h1-eth0
mininet-wifi>
```

Figura 8: Tabela ARP do host h1.

e.(1-2) - h2)

Endereço IP: 10.0.0.2

Endereço MAC: 00:00:00:00:00:02

```
h2-eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 10.0.0.2 netmask 255.0.0.0 broadcast 10.255.255.255
    inet6 fe80::200:ff:fe00:2 prefixlen 64 scopeid 0x20<link>
    ether 00:00:00:00:00:02 txqueuelen 1000 (Ethernet)
    RX packets 41 bytes 5185 (5.1 KB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 13 bytes 1066 (1.0 KB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```

Figura 9: Endereços IP e MAC do host h2.

e.3 - h2)

```
mininet-wifi> h2 route
Tabela de Roteamento IP do Kernel
Destino      Roteador      MáscaraGen.   Opções Métrica Ref  Uso Iface
10.0.0.0     0.0.0.0       255.0.0.0     U      0      0      0 h2-eth0
mininet-wifi>
```

Figura 10: Tabela de roteamento IP do host h2.

e.4 - h2)

```
mininet-wifi> h2 arp
Endereço      TipoHW  EndereçoHW    Flags Mascara      Iface
10.0.0.1      ether   00:00:00:00:00:01 C          h2-eth0
mininet-wifi>
```

Figura 11: Tabela ARP do host h2.

e.(1-2) - s1)

```
s1-eth1: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet6 fe80::466:2bff:fe77:4d04 prefixlen 64 scopeid 0x20<link>
    ether 06:66:2b:77:4d:04 txqueuelen 1000 (Ethernet)
    RX packets 11 bytes 926 (926.0 B)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 34 bytes 4389 (4.3 KB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

s1-eth2: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet6 fe80::d8a2:91ff:fec2:5ff9 prefixlen 64 scopeid 0x20<link>
    ether da:a2:91:c2:5f:f9 txqueuelen 1000 (Ethernet)
    RX packets 11 bytes 926 (926.0 B)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 33 bytes 4303 (4.3 KB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```

Figura 12: Endereços IP e MAC das interfaces do switch s1.

s1-eth1: Endereço IP: fe60::466:2bff:fe77:4d04 - Endereço MAC: 06:66:2b:77:4d:04

s1-eth2: Endereço IP: fe80::d8a2:91ff:fec2:5ff9 - Endereço MAC: da:a2:91:c2:5f:f9

e.(3-4) - s1)

Switchs são ignorantes aos endereços de IP. Eles não possuem tabela ARP e nem tabelamento de rota de IP. Apenas alguns modelos gerenciáveis possuem IP, no entanto sua única funcionalidade é para alterar configurações internas.

Atividade 2

a-b-c)

```
leonardo@leonardo-PC:~/Unicamp/EA080/lab-1$ sudo python pratica-1-II.py
[sudo] senha para leonardo:
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2 h3 h4
*** Adding switches:
s1 s2 s3
*** Adding links:
(h1, s2) (h2, s2) (h3, s3) (h4, h3) (s1, s2) (s1, s3)
*** Configuring hosts
h1 h2 h3 h4
*** Starting controller
c0
*** Starting 3 switches
s1 s2 s3 ...
*** Starting CLI:
mininet> pingall
*** Ping: testing ping reachability
h1 -> h2 h3 X
h2 -> h1 h3 X
h3 -> h1 h2 X
h4 -> X X X
*** Results: 50% dropped (6/12 received)
mininet> 
```

Figura 13: Teste da rede gerada pelo arquivo pratica-1-II.py.

É possível observar na figura 13 que o host h4 não consegue conectar com os outros hosts. A figura 14 abaixo mostra a topologia da rede testada sem alteração. Ao inserir a linha `topo.addLink('h4','s3')`, garantimos que h4 tenha acessos aos outros hosts, como verificado no teste da figura 15.

d)

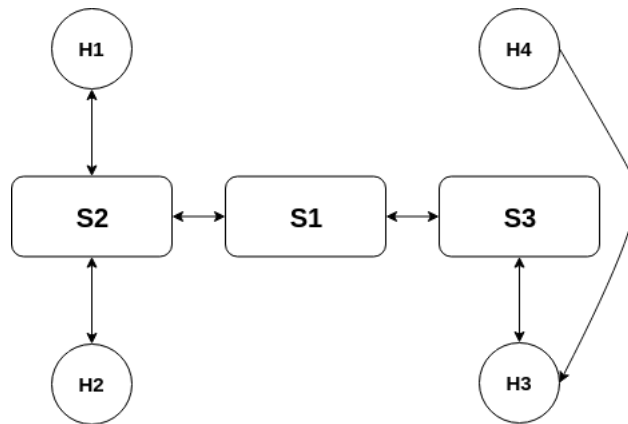


Figura 14: Topologia da rede sem alteração.

Com a adição de uma ligação entre o host h4 e o switch s3, foi obtido uma taxa de 100% de sucesso no estabelecimento das conexões.

```
leonardo@leonardo-PC:~/Unicamp/EA080/lab-1$ sudo python pratica-1-II.py
[sudo] senha para leonardo:
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2 h3 h4
*** Adding switches:
s1 s2 s3
*** Adding links:
(h1, s2) (h2, s2) (h3, s3) (h4, s3) (s1, s2) (s1, s3)
*** Configuring hosts
h1 h2 h3 h4
*** Starting controller
c0
*** Starting 3 switches
s1 s2 s3 ...
*** Starting CLI:
mininet> pingall
*** Ping: testing ping reachability
h1 -> h2 h3 h4
h2 -> h1 h3 h4
h3 -> h1 h2 h4
h4 -> h1 h2 h3
*** Results: 0% dropped (12/12 received)
mininet> 
```

Figura 15: Teste da rede gerada pelo arquivo alterado pratia-1-II.py.

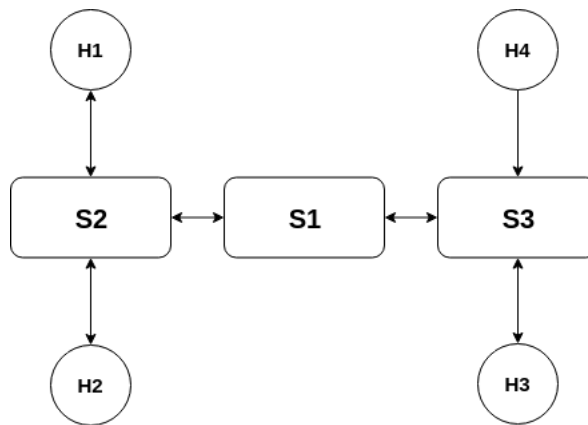


Figura 16: Topologia da rede com alteração.

Atividade 3

a-b)

```
mininet> h1 ping -c15 h2
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=17.9 ms
64 bytes from 10.0.0.2: icmp_seq=2 ttl=64 time=0.640 ms
64 bytes from 10.0.0.2: icmp_seq=3 ttl=64 time=0.081 ms
64 bytes from 10.0.0.2: icmp_seq=4 ttl=64 time=0.078 ms
64 bytes from 10.0.0.2: icmp_seq=5 ttl=64 time=0.080 ms
64 bytes from 10.0.0.2: icmp_seq=6 ttl=64 time=0.123 ms
64 bytes from 10.0.0.2: icmp_seq=7 ttl=64 time=0.099 ms
64 bytes from 10.0.0.2: icmp_seq=8 ttl=64 time=0.080 ms
64 bytes from 10.0.0.2: icmp_seq=9 ttl=64 time=0.083 ms
64 bytes from 10.0.0.2: icmp_seq=10 ttl=64 time=0.075 ms
64 bytes from 10.0.0.2: icmp_seq=11 ttl=64 time=0.101 ms
64 bytes from 10.0.0.2: icmp_seq=12 ttl=64 time=0.090 ms
64 bytes from 10.0.0.2: icmp_seq=13 ttl=64 time=0.101 ms
64 bytes from 10.0.0.2: icmp_seq=14 ttl=64 time=0.072 ms
64 bytes from 10.0.0.2: icmp_seq=15 ttl=64 time=0.080 ms

--- 10.0.0.2 ping statistics ---
15 packets transmitted, 15 received, 0% packet loss, time 14303ms
rtt min/avg/max/mdev = 0.072/1.316/17.967/4.452 ms
mininet> 
```

Figura 17: Teste de ping entre hosts h1 e h2


```
mininet> h1 ping -c15 h4
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=495 ms
64 bytes from 10.0.0.4: icmp_seq=3 ttl=64 time=200 ms
64 bytes from 10.0.0.4: icmp_seq=4 ttl=64 time=200 ms
64 bytes from 10.0.0.4: icmp_seq=7 ttl=64 time=200 ms
64 bytes from 10.0.0.4: icmp_seq=10 ttl=64 time=200 ms
64 bytes from 10.0.0.4: icmp_seq=11 ttl=64 time=200 ms
64 bytes from 10.0.0.4: icmp_seq=12 ttl=64 time=200 ms
64 bytes from 10.0.0.4: icmp_seq=13 ttl=64 time=200 ms
64 bytes from 10.0.0.4: icmp_seq=14 ttl=64 time=200 ms
64 bytes from 10.0.0.4: icmp_seq=15 ttl=64 time=200 ms

--- 10.0.0.4 ping statistics ---
15 packets transmitted, 10 received, 33% packet loss, time 14088ms
rtt min/avg/max/mdev = 200.236/230.007/495.946/88.648 ms
mininet> 
```

Figura 18: Teste de ping entre hosts h1 e h4

Comparando-se os dois testes mostrados nas figuras 17 e 18, é possível observar que a introdução de parâmetros na conexão h1 e h4 gerou uma rede bem mais lenta e instável com tempo de +/- 200ms e uma elevada taxa de perda de pacotes de 33%. Em comparação com a rede h1 e h2, o tempo foi de +/- 0.950 ms e taxa de perda de pacotes foi de 0%.

c)

```
mininet> iperf h1 h2
*** Iperf: testing TCP bandwidth between h1 and h2
*** Results: ['41.9 Gbits/sec', '42.0 Gbits/sec']
mininet> 
```

Figura 19: Largura de banda entre hosts h1 e h2

```
mininet> iperf h1 h4
*** Iperf: testing TCP bandwidth between h1 and h4
*** Results: ['330 Kbits/sec', '1.11 Mbits/sec']
mininet> 
```

Figura 20: Largura de banda entre hosts h1 e h4

Atividade 4

a-b)

```
host = ["h1", "h2", "h3", "h4", "h5", "h6", "h7", "h8"]
a_switch = ["a1", "a2", "a3", "a4"]
d_switch = ["d1", "d2"]
core = "core1"

topo=Topo()

#Create Nodes

for i in range (0, 8):
    topo.addHost(host[i])
    print(host[i])

for i in range (0, 4):
    topo.addSwitch(a_switch[i])

for i in range (0, 2):
    topo.addSwitch(d_switch[i])

topo.addSwitch(core)

#Create links between hosts and links

topo.addLink(host[0], a_switch[0], bw=10**8, delay='5ms')
topo.addLink(host[1], a_switch[0], bw=10**8, delay='5ms')
topo.addLink(host[2], a_switch[1], bw=10**8, delay='5ms')
topo.addLink(host[3], a_switch[1], bw=10**8, delay='5ms')
topo.addLink(host[4], a_switch[2], bw=10**8, delay='5ms')
topo.addLink(host[5], a_switch[2], bw=10**8, delay='5ms')
topo.addLink(host[6], a_switch[3], bw=10**8, delay='5ms')
topo.addLink(host[7], a_switch[3], bw=10**8, delay='5ms', loss=15)

#Create links between *A* switches and *D* switches

topo.addLink(a_switch[0], d_switch[0], bw=10**9, delay='3ms')
```

```

topo.addLink(a_switch[1], d_switch[0], bw=10**9, delay='3ms')
topo.addLink(a_switch[2], d_switch[1], bw=10**9, delay='3ms')
topo.addLink(a_switch[3], d_switch[1], bw=10**9, delay='3ms')

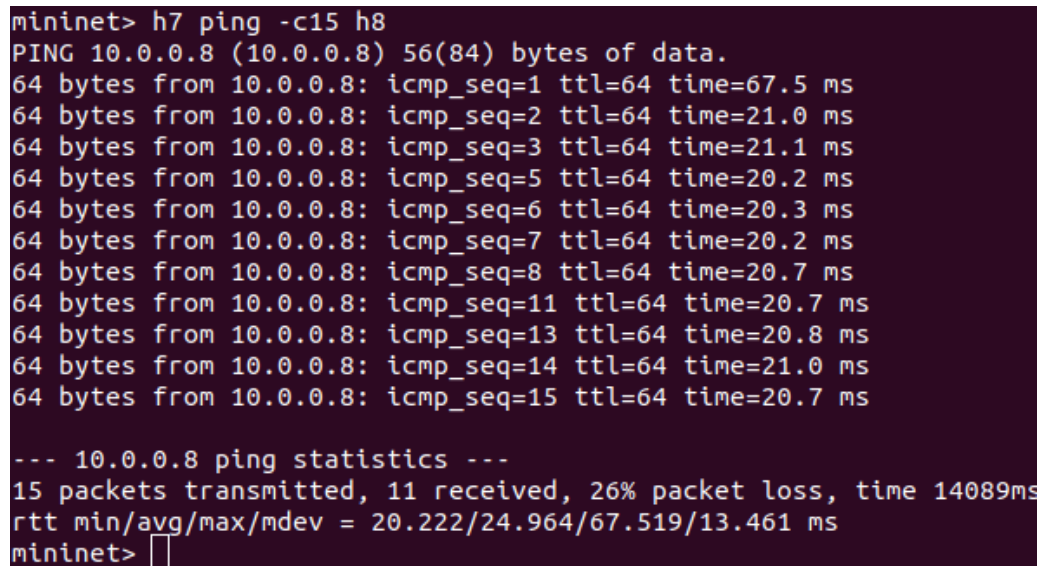
#Create links between *D* switchs and core
topo.addLink(core, d_switch[0], bw=10**10, delay='1ms')
topo.addLink(core, d_switch[1], bw=10**10, delay='1ms')

return topo

```

c)

Para medir a perda de pacotes na conexão entre o switch d4 e o host a8, eu me basearia na menor distância entre os hosts, no caso, enviaria um ping do host a7 para a8 através do switch d4. Na prática, a perda de pacotes foi de 25% usando esse método.



```

mininet> h7 ping -c15 h8
PING 10.0.0.8 (10.0.0.8) 56(84) bytes of data.
64 bytes from 10.0.0.8: icmp_seq=1 ttl=64 time=67.5 ms
64 bytes from 10.0.0.8: icmp_seq=2 ttl=64 time=21.0 ms
64 bytes from 10.0.0.8: icmp_seq=3 ttl=64 time=21.1 ms
64 bytes from 10.0.0.8: icmp_seq=5 ttl=64 time=20.2 ms
64 bytes from 10.0.0.8: icmp_seq=6 ttl=64 time=20.3 ms
64 bytes from 10.0.0.8: icmp_seq=7 ttl=64 time=20.2 ms
64 bytes from 10.0.0.8: icmp_seq=8 ttl=64 time=20.7 ms
64 bytes from 10.0.0.8: icmp_seq=11 ttl=64 time=20.7 ms
64 bytes from 10.0.0.8: icmp_seq=13 ttl=64 time=20.8 ms
64 bytes from 10.0.0.8: icmp_seq=14 ttl=64 time=21.0 ms
64 bytes from 10.0.0.8: icmp_seq=15 ttl=64 time=20.7 ms

--- 10.0.0.8 ping statistics ---
15 packets transmitted, 11 received, 26% packet loss, time 14089ms
rtt min/avg/max/mdev = 20.222/24.964/67.519/13.461 ms
mininet> 

```

Figura 21: Teste de ping entre hosts h7 e h8.

d)

- (i) Results: ['2.50 Gbits/sec', '2.51 Gbits/sec']
- (ii) Results: ['1.49 Gbits/sec', '1.50 Gbits/sec']
- (iii) Results: ['1.58 Gbits/sec', '1.58 Gbits/sec']

Atividade 5

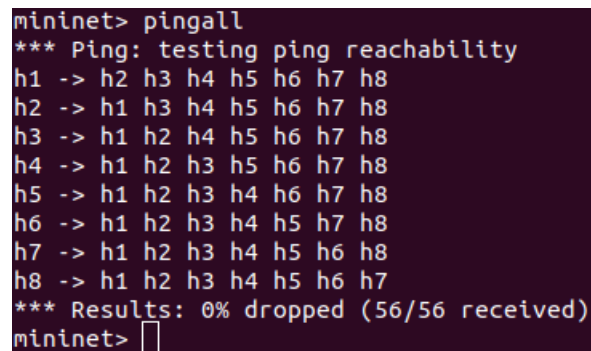
a-b)

```
#Create links between hosts and links
topo.addLink(host[0], a_switch[0], bw=10**8, delay='2ms')
topo.addLink(host[1], a_switch[0], bw=10**8, delay='2ms')
topo.addLink(host[2], a_switch[1], bw=10**8, delay='2ms')
topo.addLink(host[3], a_switch[1], bw=10**8, delay='2ms')
topo.addLink(host[4], a_switch[2], bw=10**8, delay='2ms')
topo.addLink(host[5], a_switch[2], bw=10**8, delay='2ms')
topo.addLink(host[6], a_switch[3], bw=10**8, delay='2ms')
topo.addLink(host[7], a_switch[3], bw=10**8, delay='2ms')

#Create links between *A* switchs and *D* switchs
topo.addLink(a_switch[0], d_switch[0], bw=10**9, delay='2ms')
topo.addLink(a_switch[1], d_switch[0], bw=10**9, delay='2ms')
topo.addLink(a_switch[2], d_switch[1], bw=10**9, delay='2ms')
topo.addLink(a_switch[3], d_switch[1], bw=10**9, delay='2ms')

#Create links between *D* switchs and core
topo.addLink(core, d_switch[0], bw=10**10, delay='2ms')
topo.addLink(core, d_switch[1], bw=10**10, delay='2ms')
```

i) Testar o alcance do todas as conexões.



```
mininet> pingall
*** Ping: testing ping reachability
h1 -> h2 h3 h4 h5 h6 h7 h8
h2 -> h1 h3 h4 h5 h6 h7 h8
h3 -> h1 h2 h4 h5 h6 h7 h8
h4 -> h1 h2 h3 h5 h6 h7 h8
h5 -> h1 h2 h3 h4 h6 h7 h8
h6 -> h1 h2 h3 h4 h5 h7 h8
h7 -> h1 h2 h3 h4 h5 h6 h8
h8 -> h1 h2 h3 h4 h5 h6 h7
*** Results: 0% dropped (56/56 received)
mininet> 
```

Figura 22: Teste de ping entre todos os hosts.

ii) Testar ping entre vizinhos passando por enlaces.

```
mininet> h1 ping -c5 h2
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=30.1 ms
64 bytes from 10.0.0.2: icmp_seq=2 ttl=64 time=16.6 ms
64 bytes from 10.0.0.2: icmp_seq=3 ttl=64 time=8.53 ms
64 bytes from 10.0.0.2: icmp_seq=4 ttl=64 time=8.95 ms
64 bytes from 10.0.0.2: icmp_seq=5 ttl=64 time=8.99 ms

--- 10.0.0.2 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4005ms
rtt min/avg/max/mdev = 8.539/14.658/30.129/8.312 ms
mininet> h1 ping -c5 h3
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=38.4 ms
64 bytes from 10.0.0.3: icmp_seq=2 ttl=64 time=42.3 ms
64 bytes from 10.0.0.3: icmp_seq=3 ttl=64 time=18.4 ms
64 bytes from 10.0.0.3: icmp_seq=4 ttl=64 time=16.9 ms
64 bytes from 10.0.0.3: icmp_seq=5 ttl=64 time=16.3 ms

--- 10.0.0.3 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4005ms
rtt min/avg/max/mdev = 16.335/26.511/42.309/11.416 ms
mininet> h1 ping -c5 h5
PING 10.0.0.5 (10.0.0.5) 56(84) bytes of data.
64 bytes from 10.0.0.5: icmp_seq=1 ttl=64 time=92.2 ms
64 bytes from 10.0.0.5: icmp_seq=2 ttl=64 time=68.0 ms
64 bytes from 10.0.0.5: icmp_seq=3 ttl=64 time=26.3 ms
64 bytes from 10.0.0.5: icmp_seq=4 ttl=64 time=25.4 ms
64 bytes from 10.0.0.5: icmp_seq=5 ttl=64 time=25.1 ms

--- 10.0.0.5 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4004ms
rtt min/avg/max/mdev = 25.184/47.457/92.228/27.772 ms
mininet> □
```

Figura 23: Teste de ping entre hosts h1, h2, h3, h5.

iii) Testar largura de banda entre vizinhos passando por enlaces.

```
mininet> iperf h1 h2
*** Iperf: testing TCP bandwidth between h1 and h2
*** Results: ['7.50 Gbits/sec', '7.50 Gbits/sec']
mininet> iperf h1 h3
*** Iperf: testing TCP bandwidth between h1 and h3
*** Results: ['3.83 Gbits/sec', '3.83 Gbits/sec']
mininet> iperf h1 h5
*** Iperf: testing TCP bandwidth between h1 and h5
*** Results: ['2.43 Gbits/sec', '2.43 Gbits/sec']
mininet> □
```

Figura 24: Largura de banda entre hosts h1, h2, h3 e h5

c)

É possível observar que a cada enlace que é acrescido na topologia da rede, há um aumento de aproximadamente 0.002 segundos na transmissão dos pacotes. Portanto, hosts mais distantes como h1 e h5 obterão parâmetros de desempenhos menores, comprovados nos testes de largura de banda e ping executados no item b.

Ethernet · 8		IPv4 · 7		IPv6 · 1		TCP	UDP				
Address A	Address B	Packets	Bytes	Packets A → B	Bytes A → B	Packets B → A	Bytes B → A	Rel Start	Duration		
10.0.0.1	10.0.0.2	6	588	3	294	3	294	0.000000	2.0102		
10.0.0.1	10.0.0.3	6	588	3	294	3	294	3.696395	2.0196		
10.0.0.1	10.0.0.4	6	588	3	294	3	294	7.323032	2.0207		
10.0.0.1	10.0.0.5	6	588	3	294	3	294	11.096726	2.0273		
10.0.0.1	10.0.0.6	6	588	3	294	3	294	14.979633	2.0285		
10.0.0.1	10.0.0.7	6	588	3	294	3	294	18.667393	2.0287		
10.0.0.1	10.0.0.8	6	588	3	294	3	294	22.517628	2.0294		

Figura 25: Estatísticas para transmissão de pacotes.