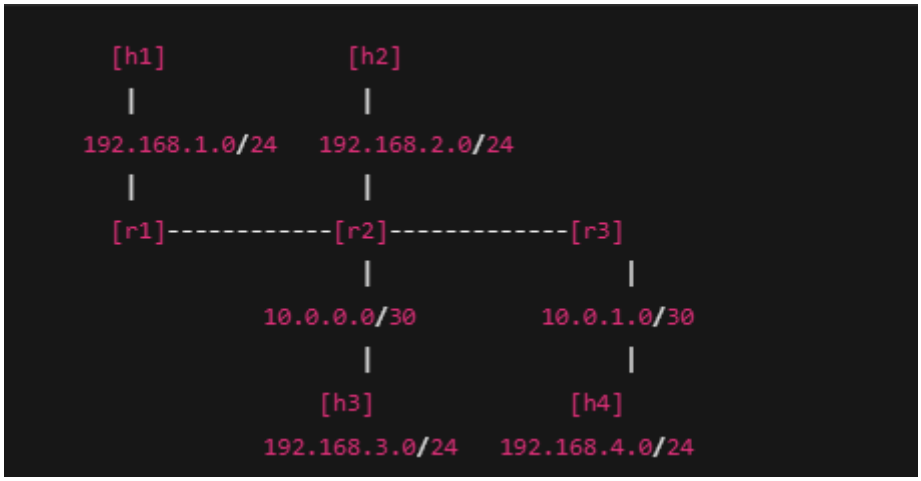
Descrição da topologia

A topologia da rede implementada no Mininet é composta por quatro hosts (h1, h2, h3 e h4) e três roteadores (r1, r2 e r3). Cada host pertence a uma sub-rede específica e os roteadores estão interconectados por links de rede distintos para garantir a comunicação entre todas as máquinas por meio de roteamento estático.

- **Host h1 (192.168.1.10/24)** está conectado ao roteador **r1 (192.168.1.1/24)**, formando a **sub-rede LAN1 (192.168.1.0/24)**.
- **Host h2 (192.168.2.10/24)** está conectado ao roteador **r2 (192.168.2.1/24)**, pertencendo à **sub-rede LAN2 (192.168.2.0/24)**.
- **Host h3 (192.168.3.10/24)** está conectado ao roteador **r3 (192.168.3.1/24)**, compondo a **sub-rede LAN3 (192.168.3.0/24)**.
- **Host h4 (192.168.4.10/24)** também está conectado ao roteador **r3 (192.168.4.1/24)** e pertence à **sub-rede LAN4 (192.168.4.0/24)**.

Os roteadores se comunicam entre si por meio de links diretos:

- **r1 e r2** estão conectados pelo link **10.0.0.0/30**, onde **r1 usa 10.0.0.1/30** e **r2 usa 10.0.0.2/30**.
- **r2 e r3** estão conectados pelo link **10.0.1.0/30**, onde **r2 usa 10.0.1.1/30** e **r3 usa 10.0.1.2/30**.



Dispositivos e ips:

Dispositivo	Interface	Endereço IP	Sub-Rede
H1	h1-eth0	192.168.1.10/24	LAN1
H2	h2-eth0	192.168.2.10/24	LAN2
H3	h3-eth0	192.168.3.10/24	LAN3
H4	h4-eth0	192.168.4.10/24	LAN4

R1	r1-eth0	192.168.1.1/24	LAN1
R1	r1-eth1	10.0.0.1/30	Link r1-r2
R2	r2-eth0	192.168.2.1/24	LAN2
R2	r2-eth1	10.0.0.2/30	Link r1-r2
R2	r2-eth2	10.0.1.1/30	Link r2-r3
R3	r3-eth0	10.0.1.2/30	Link r2-r3
R3	r3-eth1	192.168.3.1/24	LAN3
R3	r3-eth2	192.168.4.1/24	LAN4

Tabelas de Rotas

R1

Rede de Destino	Máscara	Gateway	Interface
192.168.2.0/24	255.255.255.0	10.0.0.2	r1-eth1
192.168.3.0/24	255.255.255.0	10.0.0.2	r1-eth1
192.168.4.0/24	255.255.255.0	10.0.0.2	r1-eth1

R2

Rede de Destino	Máscara	Gateway	Interface
192.168.1.0/24	255.255.255.0	10.0.0.1	R2-eth1
192.168.3.0/24	255.255.255.0	10.0.1.2	R2-eth2
192.168.4.0/24	255.255.255.0	10.0.1.2	R2-eth2

R3

Rede de Destino	Máscara	Gateway	Interface
192.168.1.0/24	255.255.255.0	10.0.1.1	R3-eth0
192.168.2.0/24	255.255.255.0	10.0.1.1	R3-eth0

R1 encaminha pacotes para qualquer outra rede através do roteador R2.

R2 funciona como um intermediário entre R1 e R3, distribuindo pacotes para as redes conectadas a ele.

R3 direciona pacotes para R2 quando precisa alcançar redes que não estão diretamente conectadas a ele.

Configurações

Foi criado um script Python personalizado utilizando o Mininet. Nele foram definidos os hosts, switches, links e roteadores, com suas respectivas interfaces.

```
# -*- coding: utf-8 -*-

from mininet.topo import Topo

from mininet.net import Mininet

from mininet.node import Node

from mininet.cli import CLI

from mininet.log import setLogLevel

class Router(Node):

def config(self, **params):

super(Router, self).config(**params)

self.cmd('sysctl -w net.ipv4.ip_forward=1') # Habilita roteamento

def terminate(self):

self.cmd('sysctl -w net.ipv4.ip_forward=0')

super(Router, self).terminate()

class CustomTopo(Topo):

def build(self):

# Hosts

h1 = self.addHost("h1", ip="192.168.1.10/24")

h2 = self.addHost("h2", ip="192.168.2.10/24")

h3 = self.addHost("h3", ip="192.168.3.10/24")

h4 = self.addHost("h4", ip="192.168.4.10/24")

# Roteadores

r1 = self.addNode("r1", cls=Router)

r2 = self.addNode("r2", cls=Router)

r3 = self.addNode("r3", cls=Router)

# Conexões hosts → roteadores

self.addLink(h1, r1, intfName2='r1-eth0')
```

```
self.addLink(h2, r2, intfName2='r2-eth0')
self.addLink(h3, r3, intfName2='r3-eth1')
self.addLink(h4, r3, intfName2='r3-eth2')
# Conexões entre roteadores
self.addLink(r1, r2,
intfName1='r1-eth1', params1={'ip': '10.0.0.1/30'},
intfName2='r2-eth1', params2={'ip': '10.0.0.2/30'})
self.addLink(r2, r3,
intfName1='r2-eth2', params1={'ip': '10.0.1.1/30'},
intfName2='r3-eth0', params2={'ip': '10.0.1.2/30'})
def run():
topo = CustomTopo()
net = Mininet(topo=topo, controller=None)
net.start()
# Acesso aos roteadores
r1 = net.get('r1')
# Acesso aos hosts
h1 = net.get('h1')
h2 = net.get('h2')
h3 = net.get('h3')
h4 = net.get('h4')
# IPs nos roteadores
r1.setIP('192.168.1.1/24', intf='r1-eth0')
r1.setIP('10.0.0.1/30', intf='r1-eth1')
r2.setIP('192.168.2.1/24', intf='r2-eth0')
r2.setIP('10.0.0.2/30', intf='r2-eth1')
r2.setIP('10.0.1.1/30', intf='r2-eth2')
r3.setIP('10.0.1.2/30', intf='r3-eth0')
r3.setIP('192.168.3.1/24', intf='r3-eth1')
```

```
r3.setIP('192.168.4.1/24', intf='r3-eth2')
# Configurações de gateway nos hosts
h1.cmd('ip route add default via 192.168.1.1')
h2.cmd('ip route add default via 192.168.2.1')
h3.cmd('ip route add default via 192.168.3.1')
h4.cmd('ip route add default via 192.168.4.1')
# Rotas no r1
r1.cmd('ip route add 192.168.2.0/24 via 10.0.0.2')
r1.cmd('ip route add 192.168.3.0/24 via 10.0.0.2')
r1.cmd('ip route add 192.168.4.0/24 via 10.0.0.2')
r1.cmd('ip route add 10.0.1.0/30 via 10.0.0.2')
# Rotas no r2
r2.cmd('ip route add 192.168.1.0/24 via 10.0.0.1')
r2.cmd('ip route add 192.168.3.0/24 via 10.0.1.2')
r2.cmd('ip route add 192.168.4.0/24 via 10.0.1.2')
# Rotas no r3
r3.cmd('ip route add 192.168.1.0/24 via 10.0.1.1')
r3.cmd('ip route add 192.168.2.0/24 via 10.0.1.1')
r3.cmd('ip route add 10.0.0.0/30 via 10.0.1.1')
CLI(net)
net.stop()
if __name__ == '__main__':
setLogLevel('info')
run()
```

Pings solicitados

- **Capturas de tela (prints) dos pings entre:**
  - **h1 ↔ h2**

```
mininet> h1 ping -c 3 h2
PING 192.168.2.10 (192.168.2.10) 56(84) bytes of data.
64 bytes from 192.168.2.10: icmp_seq=1 ttl=62 time=0.369 ms
64 bytes from 192.168.2.10: icmp_seq=2 ttl=62 time=0.185 ms
64 bytes from 192.168.2.10: icmp_seq=3 ttl=62 time=0.157 ms

--- 192.168.2.10 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 1999ms
rtt min/avg/max/mdev = 0.157/0.237/0.369/0.094 ms
mininet> h2 ping -c 3 h1
PING 192.168.1.10 (192.168.1.10) 56(84) bytes of data.
64 bytes from 192.168.1.10: icmp_seq=1 ttl=62 time=0.153 ms
64 bytes from 192.168.1.10: icmp_seq=2 ttl=62 time=0.154 ms
64 bytes from 192.168.1.10: icmp_seq=3 ttl=62 time=0.148 ms

--- 192.168.1.10 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 1998ms
rtt min/avg/max/mdev = 0.148/0.151/0.154/0.014 ms
mininet>
```

- **h1 ↔ h3**

```
mininet> h1 ping -c 3 h3
PING 192.168.3.10 (192.168.3.10) 56(84) bytes of data.
64 bytes from 192.168.3.10: icmp_seq=1 ttl=61 time=0.290 ms
64 bytes from 192.168.3.10: icmp_seq=2 ttl=61 time=0.195 ms
64 bytes from 192.168.3.10: icmp_seq=3 ttl=61 time=0.198 ms

--- 192.168.3.10 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 1999ms
rtt min/avg/max/mdev = 0.195/0.227/0.290/0.047 ms
mininet> h3 ping -c 3 h1
PING 192.168.1.10 (192.168.1.10) 56(84) bytes of data.
64 bytes from 192.168.1.10: icmp_seq=1 ttl=61 time=0.215 ms
64 bytes from 192.168.1.10: icmp_seq=2 ttl=61 time=0.348 ms
64 bytes from 192.168.1.10: icmp_seq=3 ttl=61 time=0.286 ms

--- 192.168.1.10 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2006ms
rtt min/avg/max/mdev = 0.215/0.283/0.348/0.054 ms
mininet>
```

- **h1 ↔ h4**

```
mininet> h1 ping -c 3 h4
PING 192.168.4.10 (192.168.4.10) 56(84) bytes of data.
64 bytes from 192.168.4.10: icmp_seq=1 ttl=61 time=0.302 ms
64 bytes from 192.168.4.10: icmp_seq=2 ttl=61 time=0.197 ms
64 bytes from 192.168.4.10: icmp_seq=3 ttl=61 time=0.378 ms

--- 192.168.4.10 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 1998ms
rtt min/avg/max/mdev = 0.197/0.292/0.378/0.075 ms
mininet> h4 ping -c 3 h1
PING 192.168.1.10 (192.168.1.10) 56(84) bytes of data.
64 bytes from 192.168.1.10: icmp_seq=1 ttl=61 time=0.304 ms
64 bytes from 192.168.1.10: icmp_seq=2 ttl=61 time=0.421 ms
64 bytes from 192.168.1.10: icmp_seq=3 ttl=61 time=0.196 ms

--- 192.168.1.10 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 1998ms
rtt min/avg/max/mdev = 0.196/0.307/0.421/0.091 ms
mininet>
```

- **h2 ↔ h3**

```
mininet> h2 ping -c 3 h3
PING 192.168.3.10 (192.168.3.10) 56(84) bytes of data.
64 bytes from 192.168.3.10: icmp_seq=1 ttl=62 time=0.363 ms
64 bytes from 192.168.3.10: icmp_seq=2 ttl=62 time=0.370 ms
64 bytes from 192.168.3.10: icmp_seq=3 ttl=62 time=0.166 ms

--- 192.168.3.10 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2001ms
rtt min/avg/max/mdev = 0.166/0.299/0.370/0.096 ms
mininet> h3 ping -c 3 h2
PING 192.168.2.10 (192.168.2.10) 56(84) bytes of data.
64 bytes from 192.168.2.10: icmp_seq=1 ttl=62 time=0.158 ms
64 bytes from 192.168.2.10: icmp_seq=2 ttl=62 time=0.234 ms
64 bytes from 192.168.2.10: icmp_seq=3 ttl=62 time=0.199 ms

--- 192.168.2.10 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 1999ms
rtt min/avg/max/mdev = 0.158/0.197/0.234/0.031 ms
mininet>
```

- **h2 ↔ h4**

```
mininet> h2 ping -c 3 h4
PING 192.168.4.10 (192.168.4.10) 56(84) bytes of data.
64 bytes from 192.168.4.10: icmp_seq=1 ttl=62 time=0.192 ms
64 bytes from 192.168.4.10: icmp_seq=2 ttl=62 time=0.157 ms
64 bytes from 192.168.4.10: icmp_seq=3 ttl=62 time=0.251 ms

--- 192.168.4.10 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2001ms
rtt min/avg/max/mdev = 0.157/0.200/0.251/0.038 ms
mininet> h4 ping -c 3 h2
PING 192.168.2.10 (192.168.2.10) 56(84) bytes of data.
64 bytes from 192.168.2.10: icmp_seq=1 ttl=62 time=0.182 ms
64 bytes from 192.168.2.10: icmp_seq=2 ttl=62 time=0.174 ms
64 bytes from 192.168.2.10: icmp_seq=3 ttl=62 time=0.158 ms

--- 192.168.2.10 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 1998ms
rtt min/avg/max/mdev = 0.158/0.171/0.182/0.014 ms
mininet>
```

- **h3 ↔ h4**

```
mininet> h3 ping -c 3 h4
PING 192.168.4.10 (192.168.4.10) 56(84) bytes of data.
64 bytes from 192.168.4.10: icmp_seq=1 ttl=63 time=0.215 ms
64 bytes from 192.168.4.10: icmp_seq=2 ttl=63 time=0.698 ms
64 bytes from 192.168.4.10: icmp_seq=3 ttl=63 time=0.152 ms

--- 192.168.4.10 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2006ms
rtt min/avg/max/mdev = 0.152/0.355/0.698/0.243 ms
mininet> h4 ping -c 3 h3
PING 192.168.3.10 (192.168.3.10) 56(84) bytes of data.
64 bytes from 192.168.3.10: icmp_seq=1 ttl=63 time=0.113 ms
64 bytes from 192.168.3.10: icmp_seq=2 ttl=63 time=0.113 ms
64 bytes from 192.168.3.10: icmp_seq=3 ttl=63 time=0.156 ms

--- 192.168.3.10 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 1998ms
rtt min/avg/max/mdev = 0.113/0.127/0.156/0.022 ms
mininet>
```

```
mininet> pingall
*** Ping: testing ping reachability
h1 -> h2 h3 h4 r1 r2 r3
h2 -> h1 h3 h4 r1 r2 r3
h3 -> h1 h2 h4 r1 r2 r3
h4 -> h1 h2 h3 r1 r2 r3
r1 -> h1 h2 h3 h4 r2 r3
r2 -> h1 h2 h3 h4 r1 r3
r3 -> h1 h2 h3 h4 r1 r2
*** Results: 0% dropped (42/42 received)
mininet>
```

Dificuldades encontradas:

Durante a realização da atividade, a principal dificuldade encontrada foi na comunicação entre o roteador **r3** (e os hosts conectados a ele, como o **h3**) com o **r1** e **h1**. O problema estava relacionado à ausência de rotas estáticas adequadas, que impediam o roteamento correto dos pacotes entre redes diferentes. Essa etapa exigiu várias tentativas, análises de interfaces e verificação manual das tabelas de roteamento para identificar onde estava o erro. A resolução só foi possível após adicionar corretamente as rotas no **r3**, utilizando o IP de encaminhamento via **r2**, garantindo que os pacotes alcançassem suas redes de destino. Esse foi um dos momentos mais trabalhosos e importantes da prática.

Aprendizado

Durante esta atividade, foi possível consolidar conhecimentos teóricos sobre redes de computadores, como:

- Endereçamento IP e sub-redes
- Comunicação entre dispositivos em redes diferentes

- Importância do roteamento estático para viabilizar a troca de pacotes em redes complexas

Na prática, o uso do Mininet proporcionou uma experiência próxima à configuração real de uma rede, permitindo:

- Criação de topologias personalizadas
- Configuração manual de IPs em roteadores e hosts
- Ativação do roteamento IP em dispositivos intermediários
- Adição de rotas estáticas e análise de conectividade

Além disso, foi possível entender como a **ausência de uma única rota pode comprometer a comunicação de toda a rede** e como interpretar corretamente os resultados dos testes de ping.

Sugestões de melhoria

- Adicionar visualizações gráficas da topologia para facilitar o entendimento da estrutura

- Automatizar parte da configuração com scripts de setup e verificação

- Usar ferramentas como Wireshark ou tcpdump no Mininet para observar o tráfego de pacotes e diagnosticar falhas de forma mais eficiente