

Path MTU Discovery e Tunelamento IPv6 para IPv4

Nome: Luiza Kuze Gomes

Disciplina: RCO786202

Parte 1: Path MTU Discovery

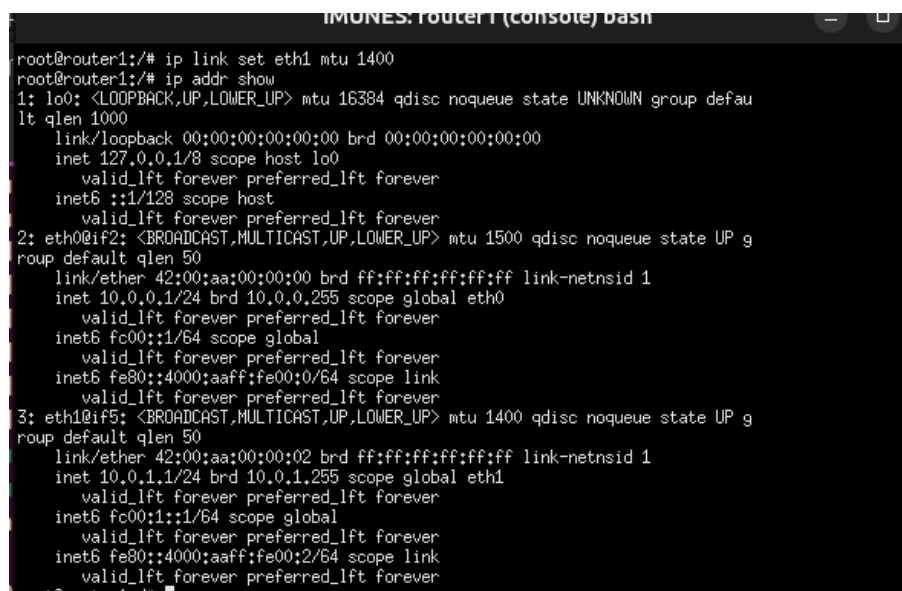
5. Altere o valor de MTU no dispositivo **router1**:

`ip link set eth1 mtu 1400`

(Próxima imagem também inclui)

6. Verifique o valor de MTU no dispositivo **router1**:

`ip addr show`.



```
IMONES:router1 (console) bash
root@router1:~# ip link set eth1 mtu 1400
root@router1:~# ip addr show
1: lo0: <LOOPBACK,UP,LOWER_UP> mtu 16384 qdisc noqueue state UNKNOWN group defau
lt qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo0
        valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
        valid_lft forever preferred_lft forever
2: eth0@if2: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP g
roup default qlen 50
    link/ether 42:00:aa:00:00:02 brd ff:ff:ff:ff:ff:ff link-netnsid 1
    inet 10.0.1.1/24 brd 10.0.1.255 scope global eth0
        valid_lft forever preferred_lft forever
    inet6 fc00::1/64 scope global
        valid_lft forever preferred_lft forever
    inet6 fe80::4000:aaff:fe00:0/64 scope link
        valid_lft forever preferred_lft forever
3: eth1@if5: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1400 qdisc noqueue state UP g
roup default qlen 50
    link/ether 42:00:aa:00:00:02 brd ff:ff:ff:ff:ff:ff link-netnsid 1
    inet 10.0.1.1/24 brd 10.0.1.255 scope global eth1
        valid_lft forever preferred_lft forever
    inet6 fc00::1/64 scope global
        valid_lft forever preferred_lft forever
    inet6 fe80::4000:aaff:fe00:2/64 scope link
        valid_lft forever preferred_lft forever
root@router1:~#
```

7. Altere o valor de MTU no dispositivo **pc2**:

`ip link set eth0 mtu 1400`

(Próxima imagem inclui)

8. Verifique o valor de MTU no dispositivo **pc2**:

`ip addr show`

```
IMUNES: pc2 (console) bash
root@pc2:/# ip link set eth0 mtu 1400
root@pc2:/# ip addr show
1: lo0: <LOOPBACK,UP,LOWER_UP> mtu 16384 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo0
        valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
        valid_lft forever preferred_lft forever
2: eth0@if4: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1400 qdisc noqueue state UP group default qlen 50
    link/ether 42:00:aa:00:00:03 brd ff:ff:ff:ff:ff:ff link-netnsid 1
    inet 10.0.1.20/24 scope global eth0
        valid_lft forever preferred_lft forever
    inet6 fc00:1::20/64 scope global
        valid_lft forever preferred_lft forever
    inet6 fe80::4000:aaff:fe00:3/64 scope link
        valid_lft forever preferred_lft forever
root@pc2:/#
```

10. Abra um terminal no **pc1** e verifique a conectividade IPv6 com o **pc2**:

`ping6 -s 1500 -M want -c 4 fc00:1::20`. Veja que o comando `ping6`, com os parâmetros apresentados, configura os pacotes enviados para conterem 1500 bytes de tamanho, por meio da opção `-s 1500`, e a interface para permitir a fragmentação de pacotes, utilizando a opção `-M want`.

```
root@pc1:/# ping6 -s 1500 -M want -c 4 fc00:1::20
PING fc00:1::20(fc00:1::20) 1500 data bytes
--- fc00:1::20 ping statistics ---
4 packets transmitted, 0 received, 100% packet loss, time 3096ms
root@pc1:/#
```

12. Procure pelo pacote **Packet Too Big**.

É o pacote número 6.

*-[eth0@pc1 (f15c2)]				
No.	Time	Source	Destination	Protocol Length Info
1	0.000000	fe80::4000:aaff:fe00:0	ff02::1	RIPng 100 Command Response, Version 1
2	7.483106	fc00::20	ff02::1:ff00:1	ICMPv6 86 Neighbor Solicitation for fc00::1 from 42:00:aa:00:00:01
3	7.483180	fc00::1	fc00::20	ICMPv6 86 Neighbor Advertisement fc00::1 (rtr, sol, ovr) is at 42:00:aa:00:00:00
4	7.483191	fc00::20	fc00::1:20	IPv6 1510 IPv6 fragment (off=0 more=y ident=0xd4afa414 nxt=58)
5	7.483203	fc00::20	fc00::1:20	ICMPv6 122 Echo (ping) request id=0x0013, seq=1, hop limit=64 (no response found!)
6	7.483256	fc00::1	fc00::20	ICMPv6 1294 Packet Too Big
7	8.486984	fc00::20	fc00::1:20	IPv6 1414 IPv6 fragment (off=0 more=y ident=0x7a6a16bd nxt=58)
8	8.487002	fc00::20	fc00::1:20	ICMPv6 218 Echo (ping) request id=0x0013, seq=2, hop limit=64 (reply in 10)
9	8.487177	fc00::1:20	fc00::20	IPv6 1414 IPv6 fragment (off=0 more=y ident=0x454accc1 nxt=58)
10	8.487178	fc00::1:20	fc00::20	ICMPv6 218 Echo (ping) reply id=0x0013, seq=2, hop limit=63 (request in 8)
11	9.510996	fc00::20	fc00::1:20	IPv6 1414 IPv6 fragment (off=0 more=y ident=0x8fbb7092 nxt=58)
12	9.511018	fc00::20	fc00::1:20	ICMPv6 218 Echo (ping) request id=0x0013, seq=3, hop limit=64 (reply in 14)
13	9.511110	fc00::1:20	fc00::20	IPv6 1414 IPv6 fragment (off=0 more=y ident=0xf9732be nxt=58)
14	9.511111	fc00::1:20	fc00::20	ICMPv6 218 Echo (ping) reply id=0x0013, seq=3, hop limit=63 (request in 12)
15	10.535118	fc00::20	fc00::1:20	IPv6 1414 IPv6 fragment (off=0 more=y ident=0x4c811781 nxt=58)
16	10.535139	fc00::20	fc00::1:20	ICMPv6 218 Echo (ping) request id=0x0013, seq=4, hop limit=64 (reply in 18)
17	10.535234	fc00::1:20	fc00::20	IPv6 1414 IPv6 fragment (off=0 more=y ident=0x55c50dd2 nxt=58)
18	10.535235	fc00::1:20	fc00::20	ICMPv6 218 Echo (ping) reply id=0x0013, seq=4, hop limit=63 (request in 16)
19	12.583064	fe80::4000:aaff:fe00:0	fc00::20	ICMPv6 86 Neighbor Solicitation for fc00::20 from 42:00:aa:00:00:00
20	12.583096	fc00::20	fe80::4000:aaff:fe00:0	ICMPv6 78 Neighbor Advertisement fc00::20 (sol)
21	14.118908	fe80::4000:aaff:fe00:1	ff02::2	ICMPv6 70 Router Solicitation from 42:00:aa:00:00:01

1. Veja se os dados contidos nos pacotes conferem com a teoria.

Wireshark - Packet 6 -	
▶ Frame 6: 1294 bytes on wire (10352 bits), 1294 bytes captured (10352 bits) on interface -, id 0 ▶ Ethernet II, Src: 42:00:aa:00:00:00 (42:00:aa:00:00:00), Dst: 42:00:aa:00:00:01 (42:00:aa:00:00:01) ▶ Internet Protocol Version 6, Src: fc00::1, Dst: fc00::20 ▼ Internet Control Message Protocol v6	
Type: Packet Too Big (2) Code: 0 Checksum: 0x99be [correct] [Checksum Status: Good] MTU: 1400	
▼ Internet Protocol Version 6, Src: fc00::20, Dst: fc00::1:20 0110 = Version: 6 ▶ 0000 0000 = Traffic Class: 0x00 (DSCP: CS0, ECN: Not-ECT) 1011 1100 0100 0001 1100 = Flow Label: 0xbc41c Payload Length: 1508 Next Header: ICMPv6 (58) Hop Limit: 64 Source Address: fc00::20 Destination Address: fc00::1:20	
▼ Internet Control Message Protocol v6 Type: Echo (ping) request (128) Code: 0 Checksum: 0x7c33 [unverified] [in ICMP error packet] [Checksum Status: Unverified] Identifier: 0x0013 Sequence: 1 Timestamp from Echo data: Jan 15, 2025 21:17:39.077258000 -03 [Timestamp from Echo data (relative): 0.000211000 seconds]	
▼ Data (1168 bytes) Data [truncated]: 101112131415161718191a1b1c1d1e1f202122232425262728292a2b2c2d2e2f303132333435363738393a3b3c3d [Length: 1168]	

Sim, o pacote foi quebrado pois o MTU é maior do que a capacidade definida.

2. Qual é o tipo (type) e código (code) do pacote? Está de acordo com a teoria? ([1])

Tipo (Type): 2. O tipo 2 no ICMPv6 representa um pacote "Packet Too Big", conforme a especificação do protocolo IPv6.

Código (Code): 0. O código 0 indica que não há subtipos ou informações adicionais associadas ao erro.

Ambos estão **de acordo com a teoria** descrita no protocolo ICMPv6.

3. Procure o campo **MTU: 1400**, qual o significado desta informação?

O **MTU 1400** indica que, neste caso, o remetente deve limitar os pacotes subsequentes a **1400 bytes ou menos** para evitar novos erros de "Packet Too Big". Para a topologia apresentada, o menor valor de MTU ao longo do caminho foi descoberto corretamente a partir do primeiro pacote. Este foi descartado ao chegar em uma rede cujo limite do MTU era de 1400 bytes. Na sequência, os pacotes foram enviados fragmentados de acordo com o valor de MTU descoberto e passaram a transitar corretamente pela rede.

13. Procure pelos pacotes **Echo (ping) request**.

No.	Time	Source	Destination	Protocol	Length	Info
2	7.483106	fc00::20	ff02::1:ff00:1	ICMPv6	86	Neighbor Solicitation for fc00::1 from 42:00:aa:00:00:01
3	7.483180	fc00::1	fc00::20	ICMPv6	86	Neighbor Advertisement fc00::1 (rtr, sol, ovr) is at 42:00:aa:00:00:00
5	7.483203	fc00::20	fc00::1::20	ICMPv6	122	Echo (ping) request id=0x0013, seq=1, hop limit=64 (no response found!)
6	7.483256	fc00::1	fc00::20	ICMPv6	1294	Packet Too Big
8	8.487002	fc00::20	fc00::1::20	ICMPv6	218	Echo (ping) request id=0x0013, seq=2, hop limit=64 (reply in 10)
10	8.487178	fc00::1::20	fc00::20	ICMPv6	218	Echo (ping) reply id=0x0013, seq=2, hop limit=63 (request in 8)
12	9.511018	fc00::20	fc00::1::20	ICMPv6	218	Echo (ping) request id=0x0013, seq=3, hop limit=64 (reply in 14)
14	9.511111	fc00::1::20	fc00::20	ICMPv6	218	Echo (ping) reply id=0x0013, seq=3, hop limit=63 (request in 12)
16	10.535139	fc00::20	fc00::1::20	ICMPv6	218	Echo (ping) request id=0x0013, seq=4, hop limit=64 (reply in 18)
18	10.535235	fc00::1::20	fc00::20	ICMPv6	218	Echo (ping) reply id=0x0013, seq=4, hop limit=63 (request in 16)
19	12.583064	fe80::4000:aaff:fe00:0	fc00::20	ICMPv6	86	Neighbor Solicitation for fc00::20 from 42:00:aa:00:00:00
20	12.583096	fc00::20	fe80::4000:aaff:fe00:0	ICMPv6	78	Neighbor Advertisement fc00::20 (sol)
21	14.118998	fe80::4000:aaff:fe00:1	ff02::2	ICMPv6	70	Router Solicitation from 42:00:aa:00:00:01

1. Analise os mesmos na camada 3 (IPv6):

i. Quantos fragmentos os mesmos possuem?

Dois fragmentos diferentes: Fragmento 1 de 1510 bytes e os restantes de 1414 bytes

ii. Qual o tamanho de cada fragmento?

1510 e 1414.

2. Onde (máquina) foi realizada a fragmentação?

A fragmentação foi realizada na **máquina de origem (fc00::20)**

i. Como você chegou a esta conclusão?

Pela captura do Wireshark nos pacotes de fragmentação. A responsabilidade de fragmentar pacotes em IPv6 é da máquina de origem fc00::20.

3. Procure pelos pacotes **Echo (ping) reply**.

Estão sublinhados abaixo.

icmpv6					
No.	Time	Source	Destination	Protocol	Length Info
2	7.483106	fc00::20	ff02::1:ff00:1	ICMPv6	86 Neighbor Solicitation for fc00::1 from 42:00:aa:00:00:01
3	7.483180	fc00::1	fc00::20	ICMPv6	86 Neighbor Advertisement fc00::1 (rtr, sol, ovr) is at 42:00:aa:00:00:00
5	7.483203	fc00::20	fc00::1:20	ICMPv6	122 Echo (ping) request id=0x0013, seq=1, hop limit=64 (no response found!)
6	7.483256	fc00::1	fc00::20	ICMPv6	1294 Packet Too Big
8	8.487002	fc00::20	fc00::1:20	ICMPv6	218 Echo (ping) request id=0x0013, seq=2, hop limit=64 (reply in 10)
10	8.487178	fc00::1:20	fc00::20	ICMPv6	218 Echo (ping) reply id=0x0013, seq=2, hop limit=63 (request in 8)
12	9.511018	fc00::20	fc00::1:20	ICMPv6	218 Echo (ping) request id=0x0013, seq=3, hop limit=64 (reply in 14)
14	9.511111	fc00::1:20	fc00::20	ICMPv6	218 Echo (ping) reply id=0x0013, seq=3, hop limit=63 (request in 12)
16	10.535139	fc00::20	fc00::1:20	ICMPv6	218 Echo (ping) request id=0x0013, seq=4, hop limit=64 (reply in 18)
18	10.535235	fc00::1:20	fc00::20	ICMPv6	218 Echo (ping) reply id=0x0013, seq=4, hop limit=63 (request in 16)
19	12.583064	fe80::4000:aaff:fe00:0	fc00::20	ICMPv6	86 Neighbor Solicitation for fc00::20 from 42:00:aa:00:00:00
20	12.583096	fc00::20	fe80::4000:aaff:fe00:0	ICMPv6	78 Neighbor Advertisement fc00::20 (sol)
21	14.118908	fe80::4000:aaff:fe00:1	ff02::2	ICMPv6	70 Router Solicitation from 42:00:aa:00:00:01

i. Eles também estão fragmentados? Prove.

Sim. Existe o campo “Fragment Header for IPv6” nos pacotes reply também juntamente com seu detalhamento.

```

Wireshark - Packet 14 - packet1000bigscapyng
+-----+
| Frame 14: 218 bytes on wire (1744 bits), 218 bytes captured (1744 bits) on interface -, id 0 |
+-----+
| Ethernet II, Src: 42:00:aa:00:00:00 (42:00:aa:00:00:00), Dst: 42:00:aa:00:00:01 (42:00:aa:00:00:01) |
+-----+
| Internet Protocol Version 6, Src: fc00::1:20, Dst: fc00::20 |
+-----+
| 0110 .... = Version: 6 |
+-----+
|  .... 0000 0000 .... = Traffic Class: 0x00 (DSCP: CS0, ECN: Not-ECT) |
|  .... 0000 00.. .... = Differentiated Services Codepoint: Default (0) |
|  .... ..00 .... = Explicit Congestion Notification: Not ECN-Capable Transport (0) |
|  .... 1010 0101 1000 1001 0011 = Flow Label: 0xa5893 |
+-----+
| Payload Length: 164 |
| Next Header: Fragment Header for IPv6 (44) |
| Hop Limit: 63 |
| Source Address: fc00::1:20 |
| Destination Address: fc00::20 |
+-----+
| Fragment Header for IPv6 |
| Next header: ICMPv6 (58) |
| Reserved octet: 0x00 |
| 0000 0101 0100 1... = Offset: 169 (1352 bytes) |
| .... ..00. = Reserved bits: 0 |
| .... ..0 = More Fragments: No |
| Identification: 0x1f9732be |
+-----+
| [2 IPv6 Fragments (1508 bytes): #13(1352), #14(156)] |
+-----+
| Internet Control Message Protocol v6 |
| Type: Echo (ping) reply (129) |
| Code: 0 |
| Checksum: 0x76c4 [correct] |
| [Checksum Status: Good] |
| Identifier: 0x0013 |
| Sequence: 3 |
| [Response To: 12] |
| [Response Time: 0.093 ms] |
| Timestamp from Echo data: Jan 15, 2025 21:17:41.105164000 -03 |
| [Timestamp from Echo data (relative): 0.000160000 seconds] |
+-----+
| Data (1484 bytes) |
+-----+

```

ii. Qual o sentido dessa fragmentação?

Melhor desempenho da rede e garantir a entrega de pacotes maiores que o MTU.

PARTE 2: Tunelamento IPv6 para IPv4

6. Teste a conectividade da rede. No terminal do **pc1** execute:

ping 10.0.1.20

ping6 fc00::1:20

```

root@pc1:~# ping 10.0.1.20
PING 10.0.1.20 (10.0.1.20) 56(84) bytes of data.
64 bytes from 10.0.1.20: icmp_seq=1 ttl=63 time=0.093 ms
64 bytes from 10.0.1.20: icmp_seq=2 ttl=63 time=0.065 ms
64 bytes from 10.0.1.20: icmp_seq=3 ttl=63 time=0.080 ms
64 bytes from 10.0.1.20: icmp_seq=4 ttl=63 time=0.103 ms
64 bytes from 10.0.1.20: icmp_seq=5 ttl=63 time=0.046 ms
^C
--- 10.0.1.20 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4076ms
rtt min/avg/max/mdev = 0.046/0.077/0.103/0.020 ms
root@pc1:~# ping6 fc00::1:20
PING fc00::1:20(fc00::1:20) 56 data bytes
From fc00::1:20 icmp_seq=1 Destination unreachable: Address unreachable
From fc00::1:20 icmp_seq=2 Destination unreachable: Address unreachable
From fc00::1:20 icmp_seq=3 Destination unreachable: Address unreachable
^C
--- fc00::1:20 ping statistics ---
5 packets transmitted, 0 received, +3 errors, 100% packet loss, time 4126ms

```

7. Ambos os pings tiveram sucesso? Sim ou não e por quê?

Não. Somente o IPv4 está configurado para encaminhar pacotes corretamente.

8. Vamos configurar o túnel IPv6 para IPv4. No terminal do **pc1** execute

ip tunnel add toPC2 mode sit ttl 64 remote 10.0.1.20 local 10.0.0.20

ip link set dev toPC2 up

ip -6 route add fc00::1:20 dev toPC2

ip addr show

ip -6 route show

1. Descreva e interprete a funcionalidade de cada um dos comandos acima.

```

root@pc1:~# ip tunnel add toPC2 mode sit ttl 64 remote 10.0.1.20 local 10.0.0.20
root@pc1:~# ip link set dev toPC2 up
root@pc1:~# ip -6 route add fc00::1:20 dev toPC2
root@pc1:~# ip addr show
1: lo0: <LOOPBACK,UP,LOWER_UP> mtu 16384 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo0
        valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
        valid_lft forever preferred_lft forever
2: sit0@NONE: <NOARP> mtu 1480 qdisc noop state DOWN group default qlen 1000
    link/sit 0.0.0.0 brd 0.0.0.0
3: eth0@if4: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group default qlen 50
    link/ether 42:00:aa:00:00:01 brd ff:ff:ff:ff:ff:ff link-netnsid 1
    inet 10.0.0.20/24 scope global eth0
        valid_lft forever preferred_lft forever
    inet6 fc00::20/64 scope global
        valid_lft forever preferred_lft forever
    inet6 fe80::4000:aaff:fe00:1/64 scope link
        valid_lft forever preferred_lft forever
4: toPC2@NONE: <POINTOPOINT,NOARP,UP,LOWER_UP> mtu 1480 qdisc noqueue state UNKNOWN group default qlen 1000
    link/sit 10.0.0.20 peer 10.0.1.20
    inet6 fe80::a00:14/64 scope link
        valid_lft forever preferred_lft forever
root@pc1:~# ip -6 route show
::1 dev lo0 proto kernel metric 256 pref medium
fc00::/64 dev eth0 proto kernel metric 256 pref medium
fc00::1:20 dev toPC2 metric 1024 pref medium
fe80::/64 dev eth0 proto kernel metric 256 pref medium
fe80::/64 dev toPC2 proto kernel metric 256 pref medium
default via fc00::1 dev eth0 metric 1024 pref medium
root@pc1:~#

```

1. **ip tunnel add toPC2 mode sit ttl 64 remote 10.0.1.20 local 10.0.0.20**

Este comando cria uma interface de túnel chamada toPC2 utilizando o modo **SIT (Simple Internet Transition)**, que encapsula pacotes IPv6 dentro de pacotes IPv4.

Parâmetros:

- **ttl 64**: Define o valor padrão do campo Time-To-Live (TTL) dos pacotes encapsulados.
- **remote 10.0.1.20**: Define o endereço IPv4 do destino (pc2 neste caso).
- **local 10.0.0.20**: Define o endereço IPv4 do ponto de origem (pc1 neste caso).

Esse comando configura o túnel, permitindo que pacotes IPv6 sejam encapsulados em IPv4, para serem enviados do pc1 para o pc2.

2. **ip link set dev toPC2 up**

Este comando ativa a interface de rede chamada toPC2.

3. **ip -6 route add fc00:1::20 dev toPC2**

Adiciona uma rota IPv6 para o destino fc00:1::20 através da interface de túnel toPC2.

4. **ip addr show**

Exibe todas as interfaces de rede ativas e suas configurações no sistema, incluindo endereços IPv4, IPv6, e estado (ativo/inativo). Esse comando é usado para verificar se a interface de túnel toPC2 foi configurada corretamente e está ativa.

5. **ip -6 route show**

Lista todas as rotas IPv6 configuradas no sistema. Este comando é usado para confirmar que a rota IPv6 para o destino fc00:1::20 foi adicionada corretamente, associada à interface toPC2.

9. Vamos configurar o túnel IPv6 para IPv4. No terminal do **pc2** execute

```
ip tunnel add toPC1 mode sit ttl 64 remote 10.0.0.20 local 10.0.1.20
```

```
ip link set dev toPC1 up
```

```
ip -6 route add fc00::20 dev toPC1
```

```
ip addr show
```

```
ip -6 route show
```

```

root@pc2:/# ip tunnel add toPC1 mode sit ttl 64 remote 10.0.0.20 local 10.0.1.20
root@pc2:/# ip link set dev toPC1 up
root@pc2:/# ip -6 route add fc00::20 dev toPC1
root@pc2:/# ip addr show
1: lo0: <LOOPBACK,UP,LOWER_UP> mtu 16384 qdisc noqueue state UNKNOWN group defau
lt qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo0
        valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
        valid_lft forever preferred_lft forever
2: sit0@NONE: <NOARP> mtu 1480 qdisc noop state DOWN group default qlen 1000
    link/sit 0.0.0.0 brd 0.0.0.0
3: eth0@if5: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP g
roup default qlen 50
    link/ether 42:00:aa:00:00:03 brd ff:ff:ff:ff:ff:ff link-netnsid 1
    inet 10.0.1.20/24 scope global eth0
        valid_lft forever preferred_lft forever
    inet6 fc00:1::20/64 scope global
        valid_lft forever preferred_lft forever
    inet6 fe80::4000:aaff:fe00:3/64 scope link
        valid_lft forever preferred_lft forever
4: toPC1@NONE: <POINTOPOINT,NOARP,UP,LOWER_UP> mtu 1480 qdisc noqueue state UNKN
OWN group default qlen 1000
    link/sit 10.0.1.20 peer 10.0.0.20
    inet6 fe80::a00:114/64 scope link
        valid_lft forever preferred_lft forever
root@pc2:/# ip -6 route show
::1 dev lo0 proto kernel metric 256 pref medium
fc00::20 dev toPC1 metric 1024 pref medium
fc00:1::/64 dev eth0 proto kernel metric 256 pref medium
fe80::/64 dev eth0 proto kernel metric 256 pref medium
fe80::/64 dev toPC1 proto kernel metric 256 pref medium
default via fc00:1::1 dev eth0 metric 1024 pref medium
root@pc2:/# █

```

11. Abra um terminal no **pc1** e verifique a conectividade IPv6 com o **pc2**:


```
IMUNES: pc1 (console) bash
root@pc1:/# ping6 -c3 fc00:1::20
PING fc00:1::20(fc00:1::20) 56 data bytes
64 bytes from fc00:1::20: icmp_seq=1 ttl=64 time=0.326 ms
64 bytes from fc00:1::20: icmp_seq=2 ttl=64 time=0.071 ms
64 bytes from fc00:1::20: icmp_seq=3 ttl=64 time=0.146 ms

--- fc00:1::20 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2036ms
rtt min/avg/max/ndev = 0.071/0.181/0.326/0.107 ms
root@pc1:/#
```

```
*- [eth0@router1 (i15c3)]
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
icmpv6
No. Time Source Destination Protocol Length Info
8 8.106673 fc00::20 fc00:1::20 ICMPv6 138 Echo (ping) request id=0x0016, seq=1, hop limit=64 (reply in 9)
9 8.106781 fc00:1::20 fc00::20 ICMPv6 138 Echo (ping) reply id=0x0016, seq=1, hop limit=64 (request in 8)
10 9.118556 fc00::20 fc00:1::20 ICMPv6 138 Echo (ping) request id=0x0016, seq=2, hop limit=64 (reply in 11)
11 9.118585 fc00:1::20 fc00::20 ICMPv6 138 Echo (ping) reply id=0x0016, seq=2, hop limit=64 (request in 10)
12 10.142768 fc00::20 fc00:1::20 ICMPv6 138 Echo (ping) request id=0x0016, seq=3, hop limit=64 (reply in 13)
13 10.142832 fc00:1::20 fc00::20 ICMPv6 138 Echo (ping) reply id=0x0016, seq=3, hop limit=64 (request in 12)
```

13. Procure por pacotes **Echo (ping) request** e **Echo (ping) reply** e clique sobre um deles.

1. Veja se os dados contidos nos pacotes conferem com a teoria.

Sim, porque tem tipo e código corretos.

No pacote **Echo Request**, o tipo é 128 e o código é 0. Isso está de acordo com a especificação do ICMPv6, que define esses valores para um pacote de requisição (request).

No pacote **Echo Reply**, o tipo é 129 e o código é 0, também de acordo com a especificação, que define esses valores para um pacote de resposta (reply).

```
- Internet Control Message Protocol v6
Type: Echo (ping) request (128)
Code: 0
Checksum: 0x8f50 [correct]
[Checksum Status: Good]
Identifier: 0x0016
Sequence: 1
[Response In: 9]
Timestamp from Echo data: Jan 15, 2025 22:01:53.149037000 -03
[Timestamp from Echo data (relative): 0.000215000 seconds]
Data (40 bytes)
Data: 101112131415161718191a1b1c1d1e1f202122232425262728292a2b2c2d2e2f3031323334353637
[Length: 40]
```

```
- Internet Control Message Protocol v6
Type: Echo (ping) reply (129)
Code: 0
Checksum: 0x8e50 [correct]
[Checksum Status: Good]
Identifier: 0x0016
Sequence: 1
[Response To: 8]
[Response Time: 0.108 ms]
Timestamp from Echo data: Jan 15, 2025 22:01:53.149037000 -03
[Timestamp from Echo data (relative): 0.000323000 seconds]
Data (40 bytes)
Data: 101112131415161718191a1b1c1d1e1f202122232425262728292a2b2c2d2e2f3031323334353637
[Length: 40]
```

2. Qual é o tipo (type), na camada *Ethernet 2*? Está de acordo com a teoria? ([2])

Sim. o tipo é IPv4. O campo **Type = 0x0800** na camada Ethernet II está correto porque o pacote está encapsulando **IPv4**, que por sua vez transporta o IPv6. Isso é consistente com o comportamento esperado para um túnel IPv6 sobre IPv4. Se fosse tráfego IPv6 diretamente na Ethernet, o campo **Type** seria 0x86DD

```

- Ethernet II, Src: 42:00:aa:00:00:01 (42:00:aa:00:00:01), Dst: 42:00:aa:00:00:00 (42:00:aa:00:00:00)
  ▾ Destination: 42:00:aa:00:00:00 (42:00:aa:00:00:00)
    Address: 42:00:aa:00:00:00 (42:00:aa:00:00:00)
    .... 1. .... = LG bit: Locally administered address (this is NOT the fact)
    .... 0 .... = IG bit: Individual address (unicast)
  ▾ Source: 42:00:aa:00:00:01 (42:00:aa:00:00:01)
    Address: 42:00:aa:00:00:01 (42:00:aa:00:00:01)
    .... 1. .... = LG bit: Locally administered address (this is NOT the fact)
    .... 0 .... = IG bit: Individual address (unicast)
  Type: IPv4 (0x0800)

- Ethernet II, Src: 42:00:aa:00:00:00 (42:00:aa:00:00:00), Dst: 42:00:aa:00:00:01 (42:00:aa:00:00:01)
  ▾ Destination: 42:00:aa:00:00:01 (42:00:aa:00:00:01)
    Address: 42:00:aa:00:00:01 (42:00:aa:00:00:01)
    .... 1. .... = LG bit: Locally administered address (this is NOT the fact)
    .... 0 .... = IG bit: Individual address (unicast)
  ▾ Source: 42:00:aa:00:00:00 (42:00:aa:00:00:00)
    Address: 42:00:aa:00:00:00 (42:00:aa:00:00:00)
    .... 1. .... = LG bit: Locally administered address (this is NOT the fact)
    .... 0 .... = IG bit: Individual address (unicast)
  Type: IPv4 (0x0800)

```

3. Analise um pacote na camada 3:

```

Type: IPv4 (0x0800)
Internet Protocol Version 4, Src: 10.0.0.20, Dst: 10.0.1.20
  0100 .... = Version: 4
  .... 0101 = Header Length: 20 bytes (5)
  ▾ Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
    Total Length: 124
    Identification: 0xe3ae (58286)
  ▾ 010. .... = Flags: 0x2, Don't fragment
    ...0 0000 0000 0000 = Fragment Offset: 0
    Time to Live: 64
    Protocol: IPv6 (41)
    Header Checksum: 0x4183 [validation disabled]
    [Header checksum status: Unverified]
    Source Address: 10.0.0.20
    Destination Address: 10.0.1.20
Internet Protocol Version 6, Src: fc00::20, Dst: fc00:1::20
  0110 .... = Version: 6
  ▾ .... 0000 0000 .... = Traffic Class: 0x00 (DSCP: CS0, ECN: Not-ECT)
    .... 1011 1100 0100 0001 1100 = Flow Label: 0xbcd4c
    Payload Length: 64
    Next Header: ICMPv6 (58)
    Hop Limit: 64
    Source Address: fc00::20
    Destination Address: fc00:1::20
Internet Control Message Protocol v6

Type: IPv6 (0x86dd)
Internet Protocol Version 6, Src: 10.0.1.20, Dst: 10.0.0.20
  0100 .... = Version: 4
  .... 0101 = Header Length: 20 bytes (5)
  ▾ Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
    Total Length: 124
    Identification: 0x82f7 (33527)
  ▾ 010. .... = Flags: 0x2, Don't fragment
    ...0 0000 0000 0000 = Fragment Offset: 0
    Time to Live: 63
    Protocol: IPv6 (41)
    Header Checksum: 0xa33a [validation disabled]
    [Header checksum status: Unverified]
    Source Address: 10.0.1.20
    Destination Address: 10.0.0.20
Internet Protocol Version 6, Src: fc00:1::20, Dst: fc00::20
  0110 .... = Version: 6
  ▾ .... 0000 0000 .... = Traffic Class: 0x00 (DSCP: CS0, ECN: Not-ECT)
    .... 1010 0101 1000 1001 0011 = Flow Label: 0xa5893
    Payload Length: 64
    Next Header: ICMPv6 (58)
    Hop Limit: 64
    Source Address: fc00:1::20
    Destination Address: fc00::20
Internet Control Message Protocol v6

```

1. Quantas camadas 3 ele possui?

Duas camadas.

2. Quais protocolos?

Dois protocolos: IPv4 e IPv6.

3. Quais endereços apresentados em cada versão do IP? São condizentes com os endereços configurados nos hosts?

IPv4

Origem: 10.0.0.20 (endereço do host pc1).

Destino: 10.0.1.20 (endereço do host pc2).

Sim, esses endereços foram configurados no túnel IPv6 sobre IPv4.

IPv6

Origem: fc00::20 (endereço IPv6 de pc1).

Destino: fc00:1::20 (endereço IPv6 de pc2).

Sim, são os endereços configurados nos hosts para o tráfego IPv6.

4. No *Internet Protocol Version 4*, procure o campo do cabeçalho **Protocol**, qual seu conteúdo?

O valor 41 identifica que o protocolo encapsulado dentro do IPv4 é **IPv6**, conforme a especificação IANA.

5. No *Internet Protocol Version 6*, procure o campo do cabeçalho **Next Header**, qual seu conteúdo?

O valor 58 identifica que o protocolo da próxima camada é **ICMPv6**, conforme a especificação do IPv6.

4. Baseado em suas respostas anteriores, explique qual versão do protocolo está "tunelada" em qual outra. Justifique sua resposta.

O protocolo IPv6 está tunelado dentro do IPv4. O cabeçalho IPv4 atua como a camada externa que encapsula o cabeçalho IPv6, e isso é evidenciado pelo campo "Protocol" do IPv4 com o valor 41, que identifica o IPv6 como o protocolo encapsulado.