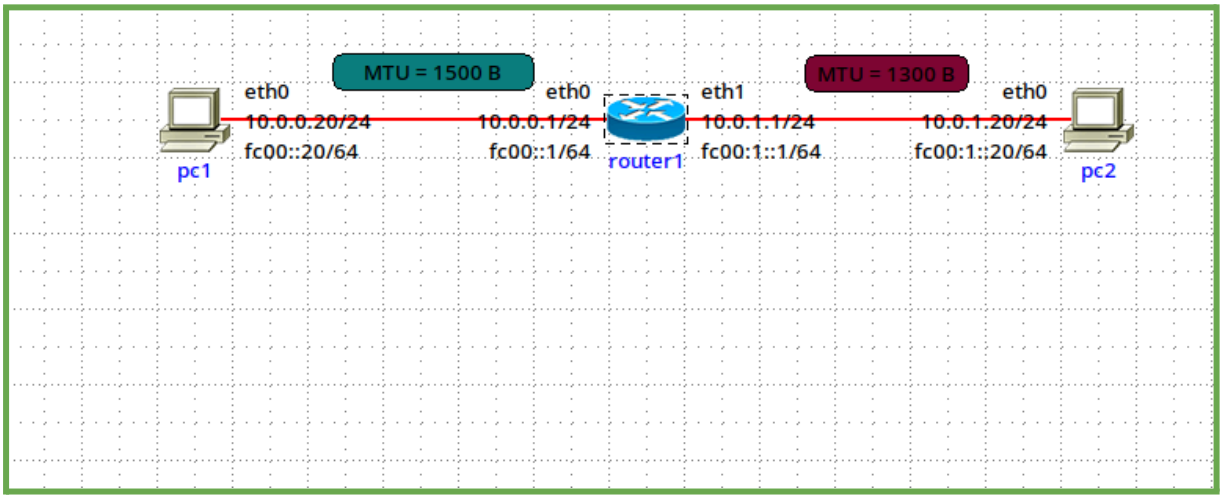
Nome: Nathan Medeiros Cristiano.
Turma: **RED129005**

LABORATÓRIO 13
USANDO MINHA MÁQUINA / IFSC
13 FRAGMENTAÇÃO DE DATAGRAMAS E TUNELAMENTO IPV6 PARA IPV



13.2 Parte 1:Fragmentação de datagramas

- **PC1**

10	75.102575	42:00:aa:00:00…	Broadcast	ARP	42	Who has 10.0.0.1? Tell 10.0.0.20
11	75.102607	42:00:aa:00:00…	42:00:aa:00:00…	ARP	42	10.0.0.1 is at 42:00:aa:00:00:00
12	75.102610	10.0.0.20	10.0.1.20	ICMP	14…	Echo (ping) request id=0x0001, seq=1/256, ttl=64 (reply in 14)
13	75.102696	10.0.1.20	10.0.0.20	IPv4	13…	Fragmented IP protocol (proto=ICMP 1, off=0, ID=b4cf) [Reassembled in #14]
14	75.102698	10.0.1.20	10.0.0.20	ICMP	212	Echo (ping) reply id=0x0001, seq=1/256, ttl=63 (request in 12)
15	76.134747	10.0.0.20	10.0.1.20	ICMP	14…	Echo (ping) request id=0x0001, seq=2/512, ttl=64 (reply in 17)
16	76.134869	10.0.1.20	10.0.0.20	IPv4	13…	Fragmented IP protocol (proto=ICMP 1, off=0, ID=b5be) [Reassembled in #17]
17	76.134872	10.0.1.20	10.0.0.20	ICMP	212	Echo (ping) reply id=0x0001, seq=2/512, ttl=63 (request in 15)
18	77.158833	10.0.0.20	10.0.1.20	ICMP	14…	Echo (ping) request id=0x0001, seq=3/768, ttl=64 (reply in 20)
19	77.158969	10.0.1.20	10.0.0.20	IPv4	13…	Fragmented IP protocol (proto=ICMP 1, off=0, ID=b616) [Reassembled in #20]
20	77.158971	10.0.1.20	10.0.0.20	ICMP	212	Echo (ping) reply id=0x0001, seq=3/768, ttl=63 (request in 18)
21	80.134557	42:00:aa:00:00…	42:00:aa:00:00…	ARP	42	Who has 10.0.0.20? Tell 10.0.0.1
22	80.134571	42:00:aa:00:00…	42:00:aa:00:00…	ARP	42	10.0.0.20 is at 42:00:aa:00:00:01

- **ROUTER**

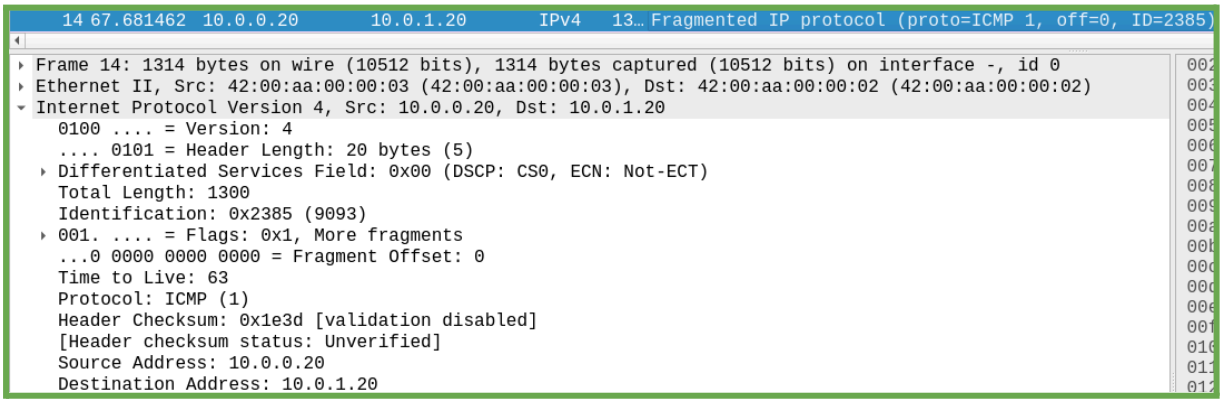
12	67.681446	42:00:aa:00:00…	Broadcast	ARP	42	Who has 10.0.1.20? Tell 10.0.1.1
13	67.681461	42:00:aa:00:00…	42:00:aa:00:00…	ARP	42	10.0.1.20 is at 42:00:aa:00:00:02
14	67.681462	10.0.0.20	10.0.1.20	IPv4	13…	Fragmented IP protocol (proto=ICMP 1, off=0, ID=2385) [Reassembled in #15]
15	67.681463	10.0.0.20	10.0.1.20	ICMP	212	Echo (ping) request id=0x0001, seq=1/256, ttl=63 (reply in 17)
16	67.681489	10.0.1.20	10.0.0.20	IPv4	13…	Fragmented IP protocol (proto=ICMP 1, off=0, ID=b4cf) [Reassembled in #17]
17	67.681491	10.0.1.20	10.0.0.20	ICMP	212	Echo (ping) reply id=0x0001, seq=1/256, ttl=64 (request in 15)
18	68.713586	10.0.0.20	10.0.1.20	IPv4	13…	Fragmented IP protocol (proto=ICMP 1, off=0, ID=23cd) [Reassembled in #19]
19	68.713593	10.0.0.20	10.0.1.20	ICMP	212	Echo (ping) request id=0x0001, seq=2/512, ttl=63 (reply in 21)
20	68.713642	10.0.1.20	10.0.0.20	IPv4	13…	Fragmented IP protocol (proto=ICMP 1, off=0, ID=b5be) [Reassembled in #21]
21	68.713644	10.0.1.20	10.0.0.20	ICMP	212	Echo (ping) reply id=0x0001, seq=2/512, ttl=64 (request in 19)
22	69.737684	10.0.0.20	10.0.1.20	IPv4	13…	Fragmented IP protocol (proto=ICMP 1, off=0, ID=2484) [Reassembled in #23]
23	69.737691	10.0.0.20	10.0.1.20	ICMP	212	Echo (ping) request id=0x0001, seq=3/768, ttl=63 (reply in 25)
24	69.737743	10.0.1.20	10.0.0.20	IPv4	13…	Fragmented IP protocol (proto=ICMP 1, off=0, ID=b616) [Reassembled in #25]
25	69.737745	10.0.1.20	10.0.0.20	ICMP	212	Echo (ping) reply id=0x0001, seq=3/768, ttl=64 (request in 23)

- **Em qual host foi realizada a fragmentação? Como você chegou a esta conclusão?**

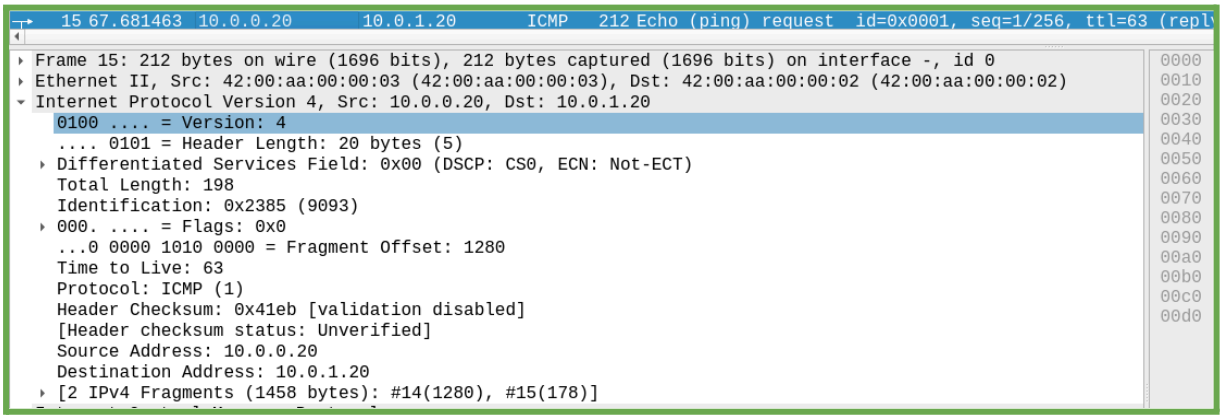
  Nenhum dos dois, o responsável pela fragmentação aqui, é o Router1, ao perceber que pela interface ETH0 chegou um pacote de 1450 Bytes, mas do outro lado a interface eth1, suporta apenas 1300 Bytes.

"*Enxergando*" melhor!

- **PACOTE 14**



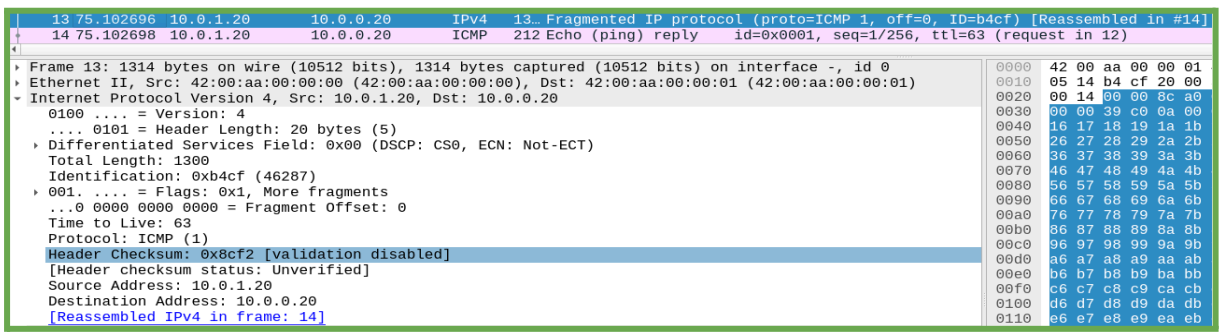
- **PACOTE 15**



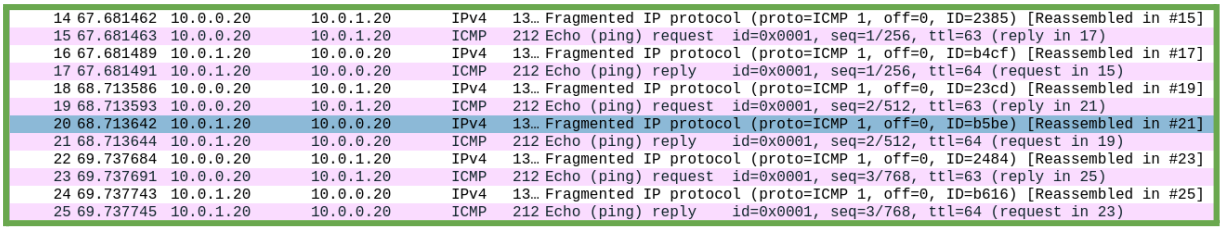
Aqui se observa, que, o primeiro pacote "original" da fila é o "14" vemos que foi fragmentado de 1450 Bytes, para 1280 Bytes. e o pacote "15", ainda pertence a esse mesmo pacote "14" sendo a diferença do 1450-1280.

Observa - se isso na flag do pacote 14, "***Flags: 0x1, More fragments***". Já no pacote 15 não se tem mais essa flag, o que diz que esse pacote foi "finalizado" ali.

- **Faça um print mostrando que, no pc1, o Echo request não é fragmentado e o Echo reply é.**

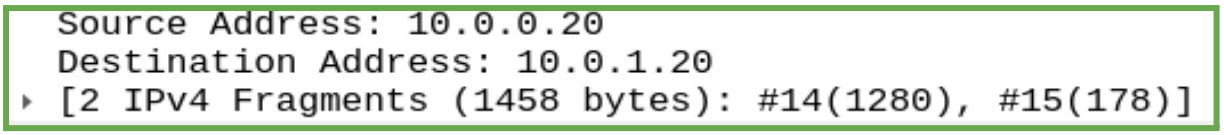


- **Na captura do router1 mostre (print) quantos fragmentos possuem tanto o Echo request quanto o Echo reply (Obs.: Procure esta informação na camada 3 – IP). Qual o tamanho de cada fragmento?**



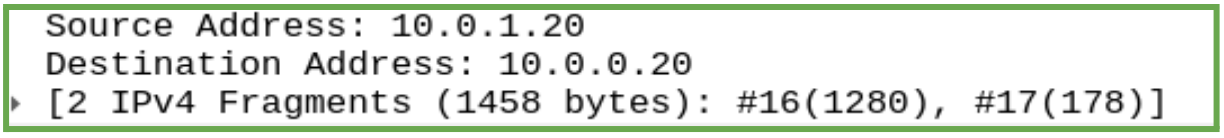
14	67.681462	10.0.0.20	10.0.1.20	IPv4	13…	Fragmented IP protocol (proto=ICMP 1, off=0, ID=2385) [Reassembled in #15]
15	67.681463	10.0.0.20	10.0.1.20	ICMP	212	Echo (ping) request id=0x0001, seq=1/256, ttl=63 (reply in 17)
16	67.681489	10.0.1.20	10.0.0.20	IPv4	13…	Fragmented IP protocol (proto=ICMP 1, off=0, ID=b4cf) [Reassembled in #17]
17	67.681491	10.0.1.20	10.0.0.20	ICMP	212	Echo (ping) reply id=0x0001, seq=1/256, ttl=64 (request in 15)
18	68.713586	10.0.0.20	10.0.1.20	IPv4	13…	Fragmented IP protocol (proto=ICMP 1, off=0, ID=23cd) [Reassembled in #19]
19	68.713593	10.0.0.20	10.0.1.20	ICMP	212	Echo (ping) request id=0x0001, seq=2/512, ttl=63 (reply in 21)
20	68.713642	10.0.1.20	10.0.0.20	IPv4	13…	Fragmented IP protocol (proto=ICMP 1, off=0, ID=b5be) [Reassembled in #21]
21	68.713644	10.0.1.20	10.0.0.20	ICMP	212	Echo (ping) reply id=0x0001, seq=2/512, ttl=64 (request in 19)
22	69.737684	10.0.0.20	10.0.1.20	IPv4	13…	Fragmented IP protocol (proto=ICMP 1, off=0, ID=2484) [Reassembled in #23]
23	69.737691	10.0.0.20	10.0.1.20	ICMP	212	Echo (ping) request id=0x0001, seq=3/768, ttl=63 (reply in 25)
24	69.737743	10.0.1.20	10.0.0.20	IPv4	13…	Fragmented IP protocol (proto=ICMP 1, off=0, ID=b616) [Reassembled in #25]
25	69.737745	10.0.1.20	10.0.0.20	ICMP	212	Echo (ping) reply id=0x0001, seq=3/768, ttl=64 (request in 23)

- **Para request's**



```
Source Address: 10.0.0.20
Destination Address: 10.0.1.20
[2 IPv4 Fragments (1458 bytes): #14(1280), #15(178)]
```

- **Para reply's**



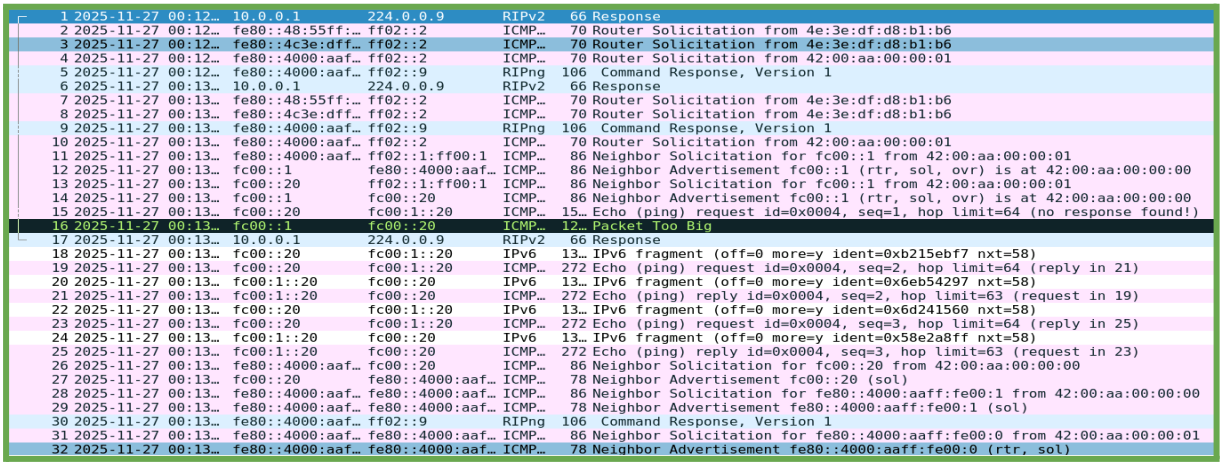
```
Source Address: 10.0.1.20
Destination Address: 10.0.0.20
[2 IPv4 Fragments (1458 bytes): #16(1280), #17(178)]
```

- **Explique a necessidade da fragmentação no Echo Reply.**

  Ela é necessária pois o PC2, tem um enlace de 1300 Bytes, logo para reenviar um pacote de 1500 Bytes, o próprio PC2 entende que precisa fragmentar este pacote, pois é maior que o MTU definido!

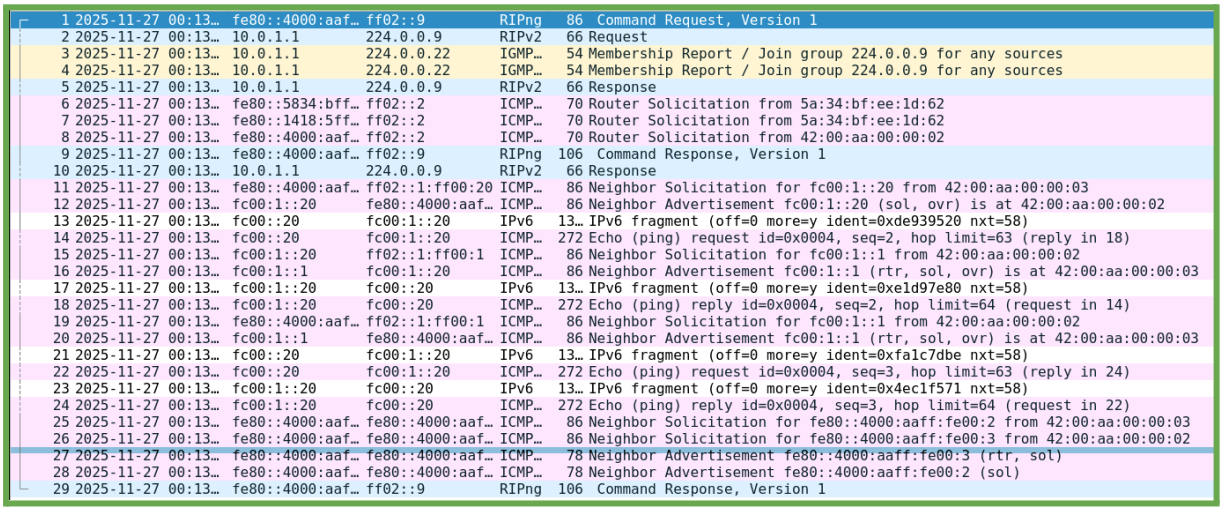
Nova Simulação: Fragmentação De Pacotes Em Ipv6.

- **PC1**



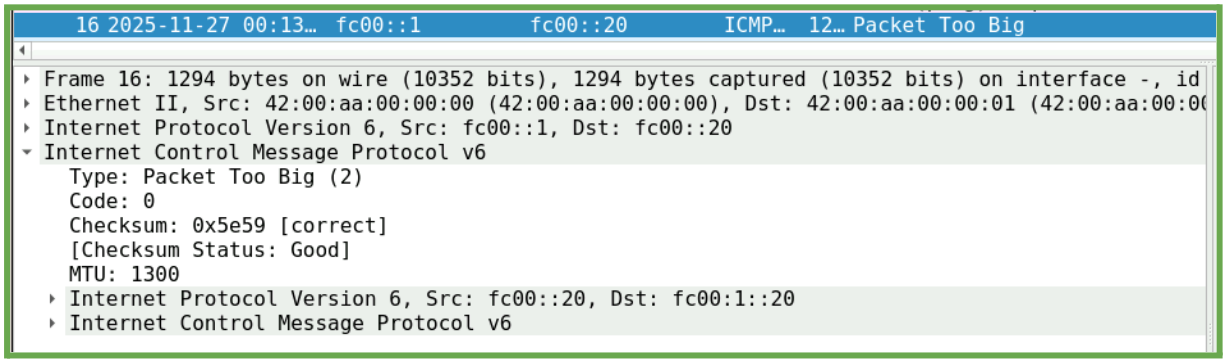
1	2025-11-27 00:12…	10.0.0.1	224.0.0.9	RIPv2	66	Response
2	2025-11-27 00:12…	fe80::48:55ff:…	ff02::2	ICMP…	70	Router Solicitation from 4e:3e:df:d8:b1:b6
3	2025-11-27 00:12…	fe80::4c3e:dff…	ff02::2	ICMP…	70	Router Solicitation from 4e:3e:df:d8:b1:b6
4	2025-11-27 00:12…	fe80::4000:aaf…	ff02::2	ICMP…	70	Router Solicitation from 42:00:aa:00:00:01
5	2025-11-27 00:12…	fe80::4000:aaf…	ff02::9	RIPng	106	Command Response, Version 1
6	2025-11-27 00:13…	10.0.0.1	224.0.0.9	RIPv2	66	Response
7	2025-11-27 00:13…	fe80::48:55ff:…	ff02::2	ICMP…	70	Router Solicitation from 4e:3e:df:d8:b1:b6
8	2025-11-27 00:13…	fe80::4c3e:dff…	ff02::2	ICMP…	70	Router Solicitation from 4e:3e:df:d8:b1:b6
9	2025-11-27 00:13…	fe80::4000:aaf…	ff02::9	RIPng	106	Command Response, Version 1
10	2025-11-27 00:13…	fe80::4000:aaf…	ff02::2	ICMP…	70	Router Solicitation from 42:00:aa:00:00:01
11	2025-11-27 00:13…	fe80::4000:aaf…	ff02::1:ff00:1	ICMP…	86	Neighbor Solicitation for fc00::1 from 42:00:aa:00:00:01
12	2025-11-27 00:13…	fc00::1	fe80::4000:aaf…	ICMP…	86	Neighbor Advertisement fc00::1 (rtr, sol, ovr) is at 42:00:aa:00:00:00
13	2025-11-27 00:13…	fc00::20	ff02::1:ff00:1	ICMP…	86	Neighbor Solicitation for fc00::1 from 42:00:aa:00:00:01
14	2025-11-27 00:13…	fc00::1	fc00::20	ICMP…	86	Neighbor Advertisement fc00::1 (rtr, sol, ovr) is at 42:00:aa:00:00:00
15	2025-11-27 00:13…	fc00::20	fc00:1::20	ICMP…	15…	Echo (ping) request id=0x0004, seq=1, hop limit=64 (no response found!)
16	2025-11-27 00:13…	fc00::1	fc00::20	ICMP…	12…	Packet Too Big
17	2025-11-27 00:13…	10.0.0.1	224.0.0.9	RIPv2	66	Response
18	2025-11-27 00:13…	fc00::20	fc00:1::20	IPv6	13…	IPv6 fragment (off=0 more=y ident=0xb215ebf7 nxt=58)
19	2025-11-27 00:13…	fc00::20	fc00:1::20	ICMP…	272	Echo (ping) request id=0x0004, seq=2, hop limit=64 (reply in 21)
20	2025-11-27 00:13…	fc00:1::20	fc00::20	IPv6	13…	IPv6 fragment (off=0 more=y ident=0x6eb54297 nxt=58)
21	2025-11-27 00:13…	fc00:1::20	fc00::20	ICMP…	272	Echo (ping) reply id=0x0004, seq=2, hop limit=63 (request in 19)
22	2025-11-27 00:13…	fc00::20	fc00:1::20	IPv6	13…	IPv6 fragment (off=0 more=y ident=0x6d241560 nxt=58)
23	2025-11-27 00:13…	fc00::20	fc00:1::20	ICMP…	272	Echo (ping) request id=0x0004, seq=3, hop limit=64 (reply in 25)
24	2025-11-27 00:13…	fc00:1::20	fc00::20	IPv6	13…	IPv6 fragment (off=0 more=y ident=0x58e2a8ff nxt=58)
25	2025-11-27 00:13…	fc00:1::20	fc00::20	ICMP…	272	Echo (ping) reply id=0x0004, seq=3, hop limit=63 (request in 23)
26	2025-11-27 00:13…	fe80::4000:aaf…	fc00::20	ICMP…	86	Neighbor Solicitation for fc00::20 from 42:00:aa:00:00:00
27	2025-11-27 00:13…	fc00::20	fe80::4000:aaf…	ICMP…	78	Neighbor Advertisement fc00::20 (sol)
28	2025-11-27 00:13…	fe80::4000:aaf…	fe80::4000:aaf…	ICMP…	86	Neighbor Solicitation for fe80::4000:aaff:fe00:1 from 42:00:aa:00:00:00
29	2025-11-27 00:13…	fe80::4000:aaf…	fe80::4000:aaf…	ICMP…	78	Neighbor Advertisement fe80::4000:aaff:fe00:1 (sol)
30	2025-11-27 00:13…	fe80::4000:aaf…	ff02::9	RIPng	106	Command Response, Version 1
31	2025-11-27 00:13…	fe80::4000:aaf…	fe80::4000:aaf…	ICMP…	86	Neighbor Solicitation for fe80::4000:aaff:fe00:0 from 42:00:aa:00:00:01
32	2025-11-27 00:13…	fe80::4000:aaf…	fe80::4000:aaf…	ICMP…	78	Neighbor Advertisement fe80::4000:aaff:fe00:0 (rtr, sol)

- **ROUTER 1**



1	2025-11-27 00:13…	fe80::4000:aaf…	ff02::9	RIPng	86	Command Request, Version 1
2	2025-11-27 00:13…	10.0.1.1	224.0.0.9	RIPv2	66	Request
3	2025-11-27 00:13…	10.0.1.1	224.0.0.22	IGMP…	54	Membership Report / Join group 224.0.0.9 for any sources
4	2025-11-27 00:13…	10.0.1.1	224.0.0.22	IGMP…	54	Membership Report / Join group 224.0.0.9 for any sources
5	2025-11-27 00:13…	10.0.1.1	224.0.0.9	RIPv2	66	Response
6	2025-11-27 00:13…	fe80::5834:bff…	ff02::2	ICMP…	70	Router Solicitation from 5a:34:bf:ee:1d:62
7	2025-11-27 00:13…	fe80::1418:5ff…	ff02::2	ICMP…	70	Router Solicitation from 5a:34:bf:ee:1d:62
8	2025-11-27 00:13…	fe80::4000:aaf…	ff02::2	ICMP…	70	Router Solicitation from 42:00:aa:00:00:02
9	2025-11-27 00:13…	fe80::4000:aaf…	ff02::9	RIPng	106	Command Response, Version 1
10	2025-11-27 00:13…	10.0.1.1	224.0.0.9	RIPv2	66	Response
11	2025-11-27 00:13…	fe80::4000:aaf…	ff02::1:ff00:20	ICMP…	86	Neighbor Solicitation for fc00:1::20 from 42:00:aa:00:00:03
12	2025-11-27 00:13…	fc00:1::20	fe80::4000:aaf…	ICMP…	86	Neighbor Advertisement fc00:1::20 (sol, ovr) is at 42:00:aa:00:00:02
13	2025-11-27 00:13…	fc00::20	fc00:1::20	IPv6	13…	IPv6 fragment (off=0 more=y ident=0xde939520 nxt=58)
14	2025-11-27 00:13…	fc00::20	fc00:1::20	ICMP…	272	Echo (ping) request id=0x0004, seq=2, hop limit=63 (reply in 18)
15	2025-11-27 00:13…	fc00:1::20	ff02::1:ff00:1	ICMP…	86	Neighbor Solicitation for fc00:1::1 from 42:00:aa:00:00:02
16	2025-11-27 00:13…	fc00:1::1	fc00:1::20	ICMP…	86	Neighbor Advertisement fc00:1::1 (rtr, sol, ovr) is at 42:00:aa:00:00:03
17	2025-11-27 00:13…	fc00:1::20	fc00::20	IPv6	13…	IPv6 fragment (off=0 more=y ident=0xe1d97e80 nxt=58)
18	2025-11-27 00:13…	fc00:1::20	fc00::20	ICMP…	272	Echo (ping) reply id=0x0004, seq=2, hop limit=64 (request in 14)
19	2025-11-27 00:13…	fe80::4000:aaf…	ff02::1:ff00:1	ICMP…	86	Neighbor Solicitation for fc00:1::1 from 42:00:aa:00:00:02
20	2025-11-27 00:13…	fc00:1::1	fe80::4000:aaf…	ICMP…	86	Neighbor Advertisement fc00:1::1 (rtr, sol, ovr) is at 42:00:aa:00:00:03
21	2025-11-27 00:13…	fc00::20	fc00:1::20	IPv6	13…	IPv6 fragment (off=0 more=y ident=0xfa1c7dbe nxt=58)
22	2025-11-27 00:13…	fc00::20	fc00:1::20	ICMP…	272	Echo (ping) request id=0x0004, seq=3, hop limit=63 (reply in 24)
23	2025-11-27 00:13…	fc00:1::20	fc00::20	IPv6	13…	IPv6 fragment (off=0 more=y ident=0x4ec1f571 nxt=58)
24	2025-11-27 00:13…	fc00:1::20	fc00::20	ICMP…	272	Echo (ping) reply id=0x0004, seq=3, hop limit=64 (request in 22)
25	2025-11-27 00:13…	fe80::4000:aaf…	fe80::4000:aaf…	ICMP…	86	Neighbor Solicitation for fe80::4000:aaff:fe00:2 from 42:00:aa:00:00:03
26	2025-11-27 00:13…	fe80::4000:aaf…	fe80::4000:aaf…	ICMP…	86	Neighbor Solicitation for fe80::4000:aaff:fe00:3 from 42:00:aa:00:00:02
27	2025-11-27 00:13…	fe80::4000:aaf…	fe80::4000:aaf…	ICMP…	78	Neighbor Advertisement fe80::4000:aaff:fe00:3 (rtr, sol)
28	2025-11-27 00:13…	fe80::4000:aaf…	fe80::4000:aaf…	ICMP…	78	Neighbor Advertisement fe80::4000:aaff:fe00:2 (sol)
29	2025-11-27 00:13…	fe80::4000:aaf…	ff02::9	RIPng	106	Command Response, Version 1

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



```
16 2025-11-27 00:13…  fc00::1          fc00::20          ICMP…  12… Packet Too Big

▸ Frame 16: 1294 bytes on wire (10352 bits), 1294 bytes captured (10352 bits) on interface -, id
▸ 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:00:0(
▸ Internet Protocol Version 6, Src: fc00::1, Dst: fc00::20
▾ Internet Control Message Protocol v6
    Type: Packet Too Big (2)
    Code: 0
    Checksum: 0x5e59 [correct]
    [Checksum Status: Good]
    MTU: 1300
  ▸ Internet Protocol Version 6, Src: fc00::20, Dst: fc00:1::20
  ▸ Internet Control Message Protocol v6
```

Qual é o tipo (type) e código (code) do pacote? Está de acordo com a teoria? (iana.org)

- Ela é do tipo 2 e é code 0. Está de acordo com a norma, pois o tipo 2 é designado especificamente para mensagens "Packet Too Big", usadas para descobrir a MTU do enlace.

Procure o campo MTU: 1300, qual o significado desta informação?

- Isso significa que o MTU (Maximum Transmission Unit), do próximo laço da rede pela qual o PC1 quer enviar, e de no máximo 1300 Bytes, quem informa isso ao PC1 é o Router. Isso para o IPV6!

Quantos fragmentos os mesmos possuem? (Obs.: Procure esta informação na camada 3 – IP)

- Observando a captura do PC1 (linhas 18 e 19) e do ROUTER 1 (linhas 13 e 14), podemos afirmar que eles possuem 2 fragmentos.

Qual o tamanho de cada fragmento?

- **1° Fragmento:** Tem o tamanho total de **1300 bytes** (limite máximo do enlace). Isso inclui o cabeçalho IPv6 fixo (40 bytes), o Cabeçalho de Extensão de Fragmentação (8 bytes) e a carga útil de dados (payload).
- **2° Fragmento:** Contém o restante dos dados que não coube no primeiro

Qual host que realizou a fragmentação?

- Quem inicializou a fragmentação e quem envia do enlace com maior MTU, no caso o PC1 (PARA IPV6), fragmentando na origem, e não no roteador.

Como você chegou a esta conclusão

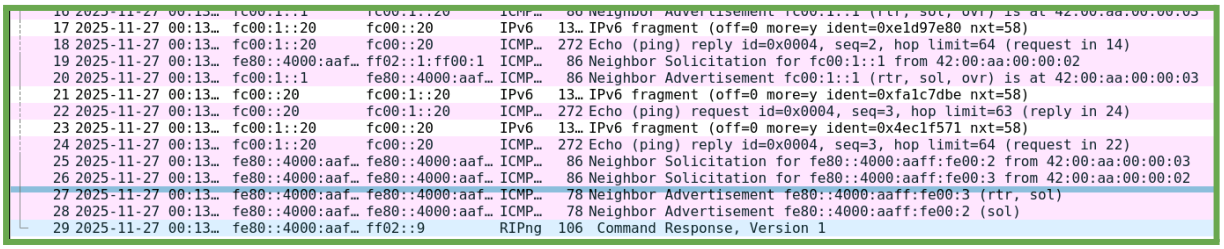
- Basta olhar as capturas realizadas pelo WireShark do PC1, e ver se está de acordo com as normas propostas para o IPV6 "iana.org".

  Na linha 16, o roteador descarta esse pacote e envia de volta uma mensagem ICMPv6 "Packet Too Big", informando ao PC1 que a MTU do próximo salto é 1300.

  A ação é vista pelo PC1, fazendo a fragmentação para enviar os dados novamente!

Procure pelos pacotes Echo (ping) reply. Eles também estão fragmentados? Prove (print).

- Sim, estão. Podemos ver isso na captura do ROUTER 1, especificamente nas linhas 17 e 18 (ou no par posterior 23 e 24).



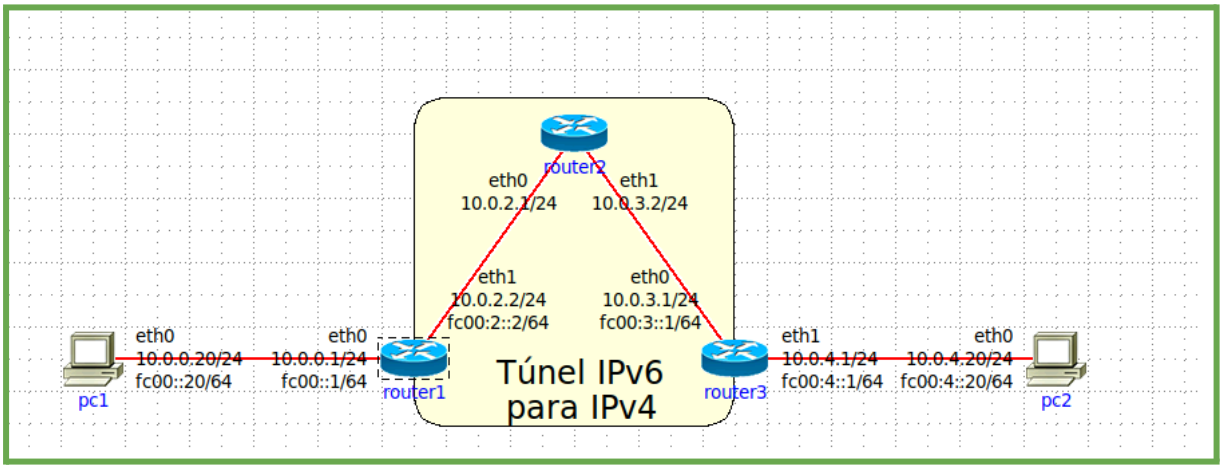
Discuta as diferenças na fragmentação no IPv4 e IPv6

- A principal diferença de fragmentação entre o IPV4 e o IPV6, está a onde ocorre a fragmentação e como isso é feito.

  IPV4: A fragmentação ocorre no roteador entre a comunicação.

  IPV6: O roteador descarta o pacote de tamanho maior que o suportado pelo outro enlace, e avisa o remetente. E quem fica responsável por fragmentar o pacote agora é o remetente.

13.3 Parte 2: Tunelamento IPv6 para IPv4



****Teste a conectividade da rede. Para IPV4 e IPV6.****

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

- Não, apenas o IPV4 teve sucesso. Isso ocorre neste cenário pois um dos roteadores do laço não reconhece o padrão IPV6.

```
root@pc1:/# ping -c2 10.0.4.20
PING 10.0.4.20 (10.0.4.20) 56(84) bytes of data.
64 bytes from 10.0.4.20: icmp_seq=1 ttl=61 time=0.144 ms
64 bytes from 10.0.4.20: icmp_seq=2 ttl=61 time=0.156 ms

--- 10.0.4.20 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1001ms
rtt min/avg/max/mdev = 0.144/0.150/0.156/0.006 ms
root@pc1:/# ping6 -c2 fc00:4::20
PING fc00:4::20(fc00:4::20) 56 data bytes
From fc00::1 icmp_seq=1 Destination unreachable: No route
From fc00::1 icmp_seq=2 Destination unreachable: No route

--- fc00:4::20 ping statistics ---
2 packets transmitted, 0 received, +2 errors, 100% packet loss, time 1020ms
```

CONFIGURANDO UM TÚNEL IPV6 - IPV4

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

```
ip tunnel add toR3 mode sit ttl 64 remote 10.0.3.1 local 10.0.2.2
ip link set dev toR3 up
ip -6 route add fc00:3::1 dev toR3
ip -6 route add fc00:4::0/64 dev toR3
```

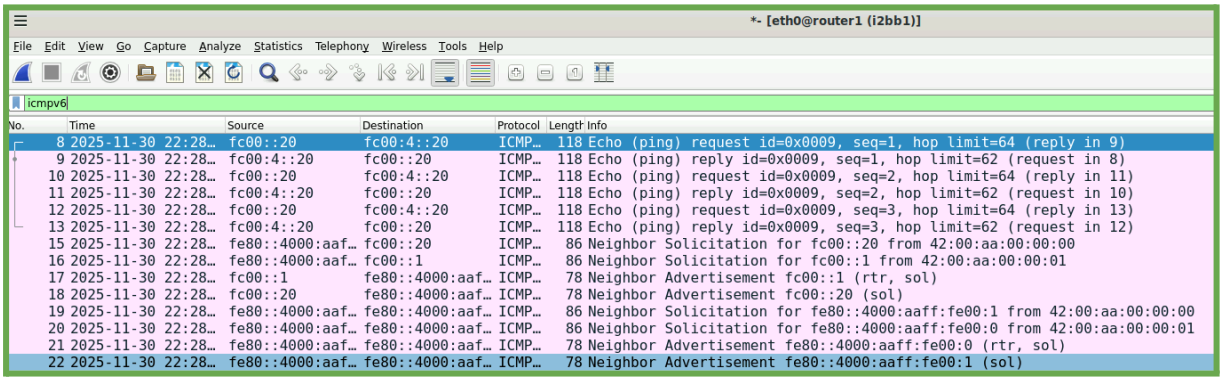
1°- Criar um túnel, onde tudo que entrar nesse túnel deve ser "envelopado" num pacote IPv4 e enviado do meu endereço (10.0.2.2) para o endereço do vizinho distante (10.0.3.1)."

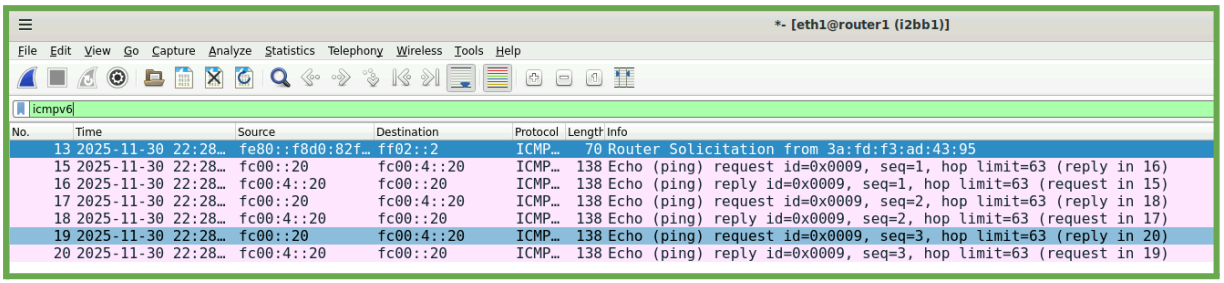
2°- Ele ativa a nova interface criada!

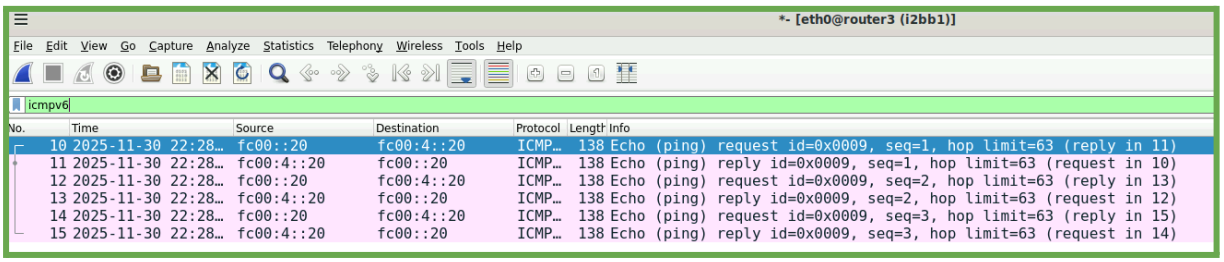
3°- Faz uma rota estática específica para um endereço IPv6. Se você precisar enviar um pacote especificamente para o endereço IPv6 “fc00:3::1” envie-o através do túnel toR3.

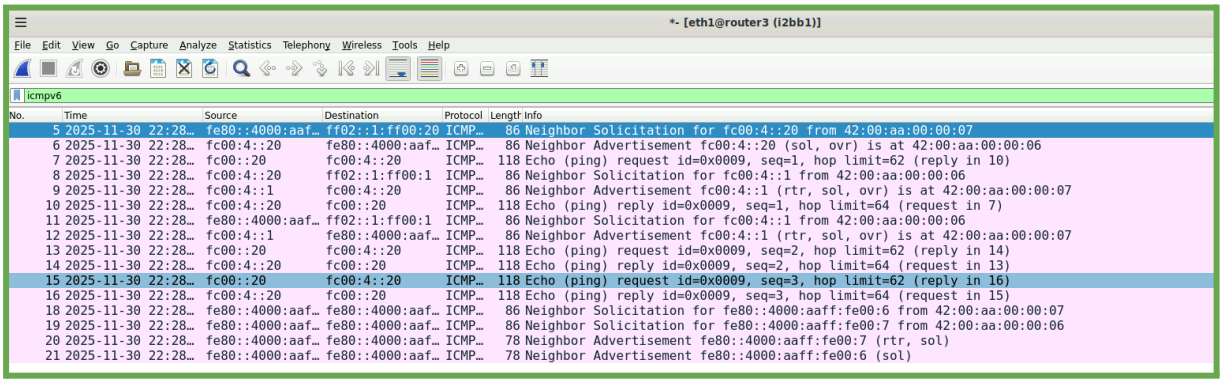
4°- Diz ao roteador: "Se chegar qualquer pacote destinado à rede “fc00:4::/64” (Rede do PC2), jogue esse pacote dentro do túnel “toR3".

****TESTANDO A CONFIGURAÇÃO FEITA****









Procure por pacotes Echo (ping) request e Echo (ping) reply e clique sobre um deles. Observe que as capturas do wireshark são diferentes em cada roteador e em suas interfaces.

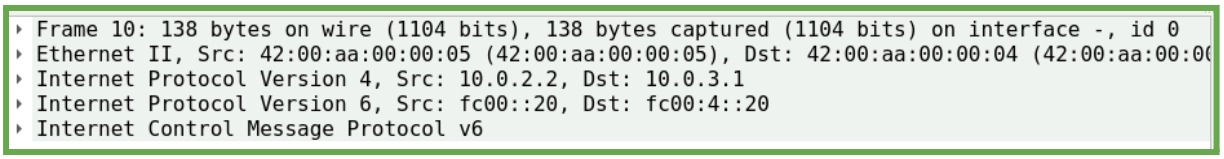
- **Em qual interface e roteador é criado o túnel?**

  O túnel começa no "Router1", na interface "eth1".

- **Em qual interface e roteador é desfeito o túnel?**

  O túnel termina no "Router3", na interface "eth0".

- **Como você concluiu isso?.**



  Basta checar e analisar as capturas feitas pelo wireshark no padrão de entrada e saída da rede.

  Por exemplo aqui se vê exatamente isso, desencapsulamento

  Sendo o IPV6 de origem do PC1 "fc00::20" e o destino o IPV6 do PC2 "fc00:4::20", no meio disso tudo tem um processo de encapsulamento para IPV4.

  "Essa captura é da interface 'eth0' do 'router3', mostrando o pacote chegando encapsulado. Isso comprova que o router3 é o ponto de destino do túnel, onde ocorrerá o desencapsulamento para entregar o pacote à rede IPv6 final."

- **Explique o conceito de túnel, IPv6 dentro do IPv4, baseado nas capturas de pacotes.**

  O tunelamento é feito para garantir a entrega em uma comunicação IPv6, mesmo que no meio tenha algum roteador que só se comunique de forma IPv4.

  Portanto a solução é:

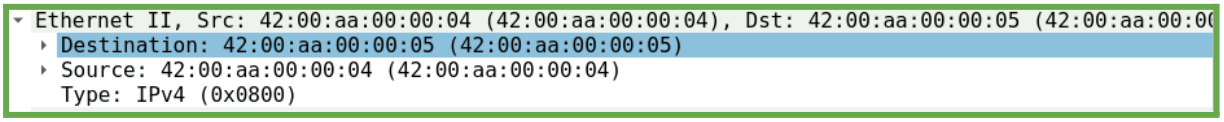
  O Router 1 pega na o pacote IPv6 inteiro e, em vez de a enviar diretamente, coloca-a dentro de um envelope maior, a um pacote IPv4.

  O pacote IPv4 viaja pela rede IPv4 usando endereços IPv4 (origem 10.0.2.2, destino 10.0.3.1). Os roteadores no meio só olham para este envelope externo.

  Quando chega ao Router 3, ele abre o envelope IPv4, retira a mensagem original IPv6 e entrega-a ao PC2.

- **No caso do túnel, qual é o tipo (type), na camada *Ethernet 2*? Está de acordo com a teoria?**

  Sim está de acordo!, (captura do eth0-Router3)



****No caso do túnel, análise um pacote na camada 3****

- **Quantas camadas 3 ele possui?**

  Como visto acima, vemos que tem dois!

- **Quais protocolos?**

  "Internet Protocol Version 4"

  "Internet Protocol Version 6"

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

  Origem: "10.0.2.2"

  Destino: "10.0.3.1"

  Origem (Src): fc00::20 (**PC1**).

  Destino (Dst): fc00:4::20 (**PC2**).

  São condizentes! O IPv4 corresponde aos túneis, enquanto os endereços IPv6 correspondem aos hosts finais que estão trocando a informação .

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

  Protocol: IPv6 (41)

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

  Next Header: ICMPv6 (58)

Explique qual versão do protocolo está "tunelada" em qual outra. Justifique sua resposta.

- A versão IPv6 está tunelada na versão IPv4. O cabeçalho IPv4 possui o campo "**Protocol: 41**" , que é o código específico para dizer "estou carregando um pacote IPv6".