

TP2 - SCR



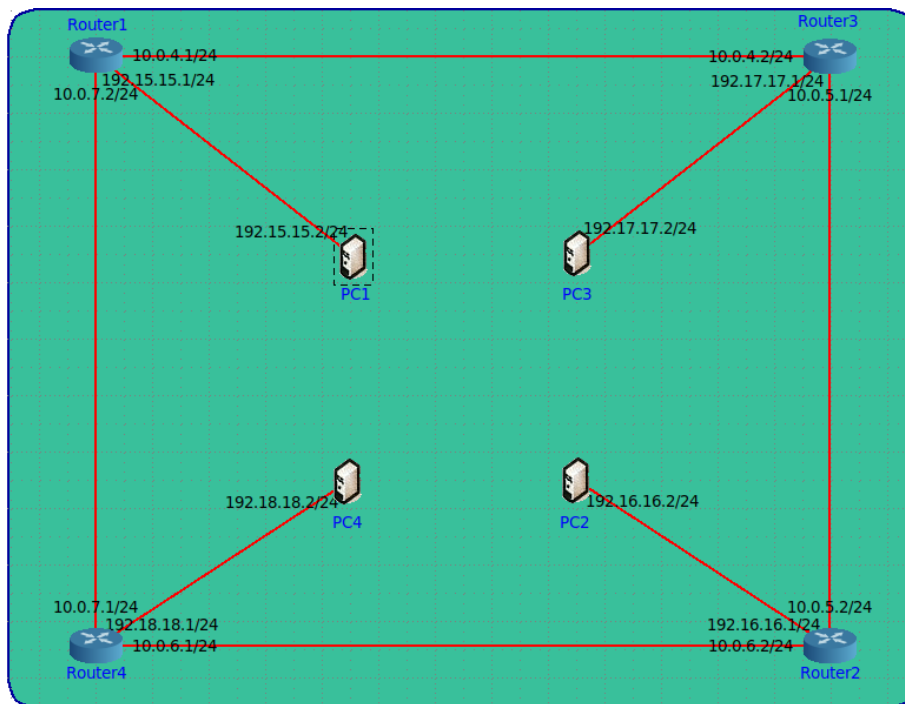
Diogo Aires, A91685

João Silva, A91638

Eduardo Pereira, A70619

PARTE 1

2. Captura do tráfego IP



Pergunta 1)

(a) Registe e analise o tráfego ICMP enviado pelo PC1 e o tráfego ICMP recebido como resposta. Comente os resultados face ao comportamento esperado.

| No. | Time | Source | Destination | Protocol | Length | Info |
|-----|--------------|-------------|-------------|----------|--------|---|
| 27 | 27.661575051 | 192.15.15.2 | 192.16.16.2 | ICMP | 74 | Echo (ping) request id=0x0024, seq=1/256, ttl=1 (no response... |
| 28 | 27.661590722 | 192.15.15.1 | 192.15.15.2 | ICMP | 102 | Time-to-live exceeded (Time to live exceeded in transit) |
| 29 | 27.661599939 | 192.15.15.2 | 192.16.16.2 | ICMP | 74 | Echo (ping) request id=0x0024, seq=2/512, ttl=1 (no response... |
| 30 | 27.661604619 | 192.15.15.1 | 192.15.15.2 | ICMP | 102 | Time-to-live exceeded (Time to live exceeded in transit) |
| 31 | 27.661608287 | 192.15.15.2 | 192.16.16.2 | ICMP | 74 | Echo (ping) request id=0x0024, seq=3/768, ttl=1 (no response... |
| 32 | 27.661612014 | 192.15.15.1 | 192.15.15.2 | ICMP | 102 | Time-to-live exceeded (Time to live exceeded in transit) |
| 33 | 27.661615901 | 192.15.15.2 | 192.16.16.2 | ICMP | 74 | Echo (ping) request id=0x0024, seq=4/1024, ttl=2 (no respons... |
| 34 | 27.661633403 | 10.0.4.2 | 192.15.15.2 | ICMP | 102 | Time-to-live exceeded (Time to live exceeded in transit) |
| 35 | 27.661637307 | 192.15.15.2 | 192.16.16.2 | ICMP | 74 | Echo (ping) request id=0x0024, seq=5/1280, ttl=2 (no respons... |
| 36 | 27.661644282 | 10.0.4.2 | 192.15.15.2 | ICMP | 102 | Time-to-live exceeded (Time to live exceeded in transit) |
| 37 | 27.661647694 | 192.15.15.2 | 192.16.16.2 | ICMP | 74 | Echo (ping) request id=0x0024, seq=6/1536, ttl=2 (no respons... |
| 38 | 27.661654021 | 10.0.4.2 | 192.15.15.2 | ICMP | 102 | Time-to-live exceeded (Time to live exceeded in transit) |
| 39 | 27.661657902 | 192.15.15.2 | 192.16.16.2 | ICMP | 74 | Echo (ping) request id=0x0024, seq=7/1792, ttl=3 (no respons... |
| 40 | 27.661675485 | 10.0.5.2 | 192.15.15.2 | ICMP | 102 | Time-to-live exceeded (Time to live exceeded in transit) |
| 41 | 27.661679057 | 192.15.15.2 | 192.16.16.2 | ICMP | 74 | Echo (ping) request id=0x0024, seq=8/2048, ttl=3 (no respons... |
| 42 | 27.661688221 | 10.0.5.2 | 192.15.15.2 | ICMP | 102 | Time-to-live exceeded (Time to live exceeded in transit) |
| 43 | 27.661691757 | 192.15.15.2 | 192.16.16.2 | ICMP | 74 | Echo (ping) request id=0x0024, seq=9/2304, ttl=3 (no respons... |
| 44 | 27.661700455 | 10.0.5.2 | 192.15.15.2 | ICMP | 102 | Time-to-live exceeded (Time to live exceeded in transit) |
| 45 | 27.661704384 | 192.15.15.2 | 192.16.16.2 | ICMP | 74 | Echo (ping) request id=0x0024, seq=10/2560, ttl=4 (reply in ... |
| 46 | 27.661735024 | 192.16.16.2 | 192.15.15.2 | ICMP | 74 | Echo (ping) reply id=0x0024, seq=10/2560, ttl=61 (request ... |

Cada router no percurso até ao destino decremente 1 ao TTL, se este atingir o valor 0, esse router descarta o datagrama e envia uma mensagem ao host de origem indicando que o TTL foi excedido. Ou seja, o datagrama com o TTL=1 faz com que o router a um salto de distância envie essa mensagem para o host de origem. O datagrama com TTL=2 provoca esse comportamento no router a 2 saltos de distância e assim sucessivamente. Quando a mensagem ICMP Echo request atinge o host destino, este responde com a mensagem ICMP Echo Reply (Type=0).

Como podemos ver, é exatamente isso que acontece na prática, sendo que só com TTL=4 é que o host de destino responde com uma mensagem ICMP Echo Reply.

(b) Qual deve ser o valor inicial mínimo do campo TTL para alcançar o PC2? Verifique na prática que a sua resposta está correta.

O valor inicial mínimo do TTL para alcançar o PC2 deve ser 4, de maneira a conseguir percorrer os 4 nodos existentes no caminho de PC1 para PC2 (Router1, Router3, Router2 e, finalmente, PC2).

```

vcmd
root@PC1:/tmp/pycore.38903/PC1.conf# traceroute -I 192.16.16.2
traceroute to 192.16.16.2 (192.16.16.2), 30 hops max, 60 byte packets
 1 192.15.15.1 (192.15.15.1) 0.043 ms 0.003 ms 0.002 ms
 2 10.0.4.2 (10.0.4.2) 0.018 ms 0.004 ms 0.004 ms
 3 10.0.5.2 (10.0.5.2) 0.012 ms 0.005 ms 0.006 ms
 4 192.16.16.2 (192.16.16.2) 0.018 ms 0.007 ms 0.008 ms
root@PC1:/tmp/pycore.38903/PC1.conf#

```

| No. | Time | Source | Destination | Protocol | Length | Info |
|-----|--------------|-------------|-------------|----------|--------|--|
| 27 | 27.661575051 | 192.15.15.2 | 192.16.16.2 | ICMP | 74 | Echo (ping) request id=0x0024, seq=1/256, ttl=1 (no response...) |
| 28 | 27.661590722 | 192.15.15.1 | 192.15.15.2 | ICMP | 102 | Time-to-live exceeded (Time to live exceeded in transit) |
| 29 | 27.661599939 | 192.15.15.2 | 192.16.16.2 | ICMP | 74 | Echo (ping) request id=0x0024, seq=2/512, ttl=1 (no response...) |
| 30 | 27.661604619 | 192.15.15.1 | 192.15.15.2 | ICMP | 102 | Time-to-live exceeded (Time to live exceeded in transit) |
| 31 | 27.661608287 | 192.15.15.2 | 192.16.16.2 | ICMP | 74 | Echo (ping) request id=0x0024, seq=3/768, ttl=1 (no response...) |
| 32 | 27.661612014 | 192.15.15.1 | 192.15.15.2 | ICMP | 102 | Time-to-live exceeded (Time to live exceeded in transit) |
| 33 | 27.661615901 | 192.15.15.2 | 192.16.16.2 | ICMP | 74 | Echo (ping) request id=0x0024, seq=4/1024, ttl=2 (no respons... |
| 34 | 27.661633403 | 10.0.4.2 | 192.15.15.2 | ICMP | 102 | Time-to-live exceeded (Time to live exceeded in transit) |
| 35 | 27.661637307 | 192.15.15.2 | 192.16.16.2 | ICMP | 74 | Echo (ping) request id=0x0024, seq=5/1280, ttl=2 (no respons... |
| 36 | 27.661644282 | 10.0.4.2 | 192.15.15.2 | ICMP | 102 | Time-to-live exceeded (Time to live exceeded in transit) |
| 37 | 27.661647694 | 192.15.15.2 | 192.16.16.2 | ICMP | 74 | Echo (ping) request id=0x0024, seq=6/1536, ttl=2 (no respons... |
| 38 | 27.661654021 | 10.0.4.2 | 192.15.15.2 | ICMP | 102 | Time-to-live exceeded (Time to live exceeded in transit) |
| 39 | 27.661657902 | 192.15.15.2 | 192.16.16.2 | ICMP | 74 | Echo (ping) request id=0x0024, seq=7/1792, ttl=3 (no respons... |
| 40 | 27.661675485 | 10.0.5.2 | 192.15.15.2 | ICMP | 102 | Time-to-live exceeded (Time to live exceeded in transit) |
| 41 | 27.661679057 | 192.15.15.2 | 192.16.16.2 | ICMP | 74 | Echo (ping) request id=0x0024, seq=8/2048, ttl=3 (no respons... |
| 42 | 27.661688221 | 10.0.5.2 | 192.15.15.2 | ICMP | 102 | Time-to-live exceeded (Time to live exceeded in transit) |
| 43 | 27.661691757 | 192.15.15.2 | 192.16.16.2 | ICMP | 74 | Echo (ping) request id=0x0024, seq=9/2304, ttl=3 (no respons... |
| 44 | 27.661700455 | 10.0.5.2 | 192.15.15.2 | ICMP | 102 | Time-to-live exceeded (Time to live exceeded in transit) |
| 45 | 27.661704384 | 192.15.15.2 | 192.16.16.2 | ICMP | 74 | Echo (ping) request id=0x0024, seq=10/2560, ttl=4 (reply in ...) |
| 46 | 27.661735024 | 192.16.16.2 | 192.15.15.2 | ICMP | 74 | Echo (ping) reply id=0x0024, seq=10/2560, ttl=61 (request ...) |

Como podemos ver pelas imagens acima, quando TTL=1, TTL=2 ou TTL=3, o pacote não chega ao destino. Apenas quando TTL=4, é que o pacote chega ao PC2 e este emite uma resposta. Portanto confirma-se que o valor do campo TTL tem de ser no mínimo 4 para alcançar o destino.

(c) Calcule o valor médio do tempo de ida-e-volta (Round-Trip Time) obtido? (Para melhorar a média, convém alterar o número de probe packets com a opção -q).

```

vcmd
root@PC1:/tmp/pycore.38903/PC1.conf# traceroute -I 192.16.16.2 -q 10
traceroute to 192.16.16.2 (192.16.16.2), 30 hops max, 60 byte packets
 1 192.15.15.1 (192.15.15.1) 0.034 ms 0.006 ms 0.005 ms 0.005 ms 0.004 ms
 0.005 ms * * * *
 2 10.0.4.2 (10.0.4.2) 0.016 ms 0.009 ms 0.007 ms 0.008 ms 0.007 ms 0.007
 ms * * * *
 3 10.0.5.2 (10.0.5.2) 0.021 ms 0.011 ms 0.010 ms 0.010 ms 0.010 ms 0.010
 ms * * * *
 4 192.16.16.2 (192.16.16.2) 0.020 ms 0.013 ms 0.012 ms 0.012 ms 0.013 ms
 0.013 ms 0.012 ms 0.012 ms 0.013 ms 0.013 ms
root@PC1:/tmp/pycore.38903/PC1.conf#

```

O tempo médio de ida-e-volta pode se calcular através do comando

traceroute -I 192.16.16.2 -q 10

Então utilizando os valores presentes no passo 4 do caminho do Host de origem até ao destino, pois apenas no passo 4 é que o destino retorna uma resposta à origem, calcula-se: $(0.020+0.013+0.012+0.012+0.013+0.013+0.012+0.012+0.013+0.013)/10 = 0.013$.

Então, o valor médio do tempo de ida-e-volta é de 0,013 ms.

Pergunta 2) Pretende-se agora usar o traceroute na sua máquina nativa.

(a) Qual é o endereço IP da interface ativa do seu computador?

| No. | Time | Source | Destination | Protocol | Length | Info |
|-----|-----------|----------------|---------------|----------|--------|--|
| 23 | 17.771271 | 172.26.0.175 | 193.136.9.240 | ICMP | 106 | Echo (ping) request id=0x0001, seq=10/2560, ttl=1 (no response found!) |
| 24 | 17.780123 | 172.26.254.254 | 172.26.0.175 | ICMP | 70 | Time-to-live exceeded (Time to live exceeded in transit) |
| 25 | 17.782696 | 172.26.0.175 | 193.136.9.240 | ICMP | 106 | Echo (ping) request id=0x0001, seq=11/2816, ttl=1 (no response found!) |
| 26 | 17.785533 | 172.26.254.254 | 172.26.0.175 | ICMP | 70 | Time-to-live exceeded (Time to live exceeded in transit) |
| 27 | 17.787446 | 172.26.0.175 | 193.136.9.240 | ICMP | 106 | Echo (ping) request id=0x0001, seq=12/3072, ttl=1 (no response found!) |
| 28 | 17.789846 | 172.26.254.254 | 172.26.0.175 | ICMP | 70 | Time-to-live exceeded (Time to live exceeded in transit) |
| 58 | 23.308666 | 172.26.0.175 | 193.136.9.240 | ICMP | 106 | Echo (ping) request id=0x0001, seq=13/3328, ttl=2 (no response found!) |
| 59 | 23.317121 | 172.16.2.1 | 172.26.0.175 | ICMP | 70 | Time-to-live exceeded (Time to live exceeded in transit) |
| 60 | 23.319653 | 172.26.0.175 | 193.136.9.240 | ICMP | 106 | Echo (ping) request id=0x0001, seq=14/3584, ttl=2 (no response found!) |
| 61 | 23.322111 | 172.16.2.1 | 172.26.0.175 | ICMP | 70 | Time-to-live exceeded (Time to live exceeded in transit) |
| 62 | 23.324838 | 172.26.0.175 | 193.136.9.240 | ICMP | 106 | Echo (ping) request id=0x0001, seq=15/3840, ttl=2 (no response found!) |
| 63 | 23.328891 | 172.16.2.1 | 172.26.0.175 | ICMP | 70 | Time-to-live exceeded (Time to live exceeded in transit) |
| 92 | 28.890738 | 172.26.0.175 | 193.136.9.240 | ICMP | 106 | Echo (ping) request id=0x0001, seq=16/4096, ttl=3 (no response found!) |
| 93 | 28.894455 | 172.16.115.252 | 172.26.0.175 | ICMP | 70 | Time-to-live exceeded (Time to live exceeded in transit) |
| 94 | 28.901851 | 172.26.0.175 | 193.136.9.240 | ICMP | 106 | Echo (ping) request id=0x0001, seq=17/4352, ttl=3 (no response found!) |
| 95 | 28.904568 | 172.16.115.252 | 172.26.0.175 | ICMP | 70 | Time-to-live exceeded (Time to live exceeded in transit) |
| 96 | 28.908490 | 172.26.0.175 | 193.136.9.240 | ICMP | 106 | Echo (ping) request id=0x0001, seq=18/4608, ttl=3 (no response found!) |
| 97 | 28.911528 | 172.16.115.252 | 172.26.0.175 | ICMP | 70 | Time-to-live exceeded (Time to live exceeded in transit) |
| 111 | 34.456710 | 172.26.0.175 | 193.136.9.240 | ICMP | 106 | Echo (ping) request id=0x0001, seq=19/4864, ttl=4 (reply in 112) |
| 112 | 34.465858 | 193.136.9.240 | 172.26.0.175 | ICMP | 106 | Echo (ping) reply id=0x0001, seq=19/4864, ttl=4 (request in 111) |
| 113 | 34.468446 | 172.26.0.175 | 193.136.9.240 | ICMP | 106 | Echo (ping) request id=0x0001, seq=20/5120, ttl=4 (reply in 114) |

IP: 172.26.0.175

(b) Qual é o valor do campo protocolo? O que identifica?

| No. | Time | Source | Destination | Protocol | Length | Info |
|-----|-----------|----------------|---------------|----------|--------|--|
| 23 | 17.771271 | 172.26.0.175 | 193.136.9.240 | ICMP | 106 | Echo (ping) request id=0x0001, seq=10/2560, ttl=1 (no response found!) |
| 24 | 17.780123 | 172.26.254.254 | 172.26.0.175 | ICMP | 70 | Time-to-live exceeded (Time to live exceeded in transit) |
| 25 | 17.782096 | 172.26.0.175 | 193.136.9.240 | ICMP | 106 | Echo (ping) request id=0x0001, seq=11/2816, ttl=1 (no response found!) |
| 26 | 17.785533 | 172.26.254.254 | 172.26.0.175 | ICMP | 70 | Time-to-live exceeded (Time to live exceeded in transit) |
| 27 | 17.787446 | 172.26.0.175 | 193.136.9.240 | ICMP | 106 | Echo (ping) request id=0x0001, seq=12/3072, ttl=1 (no response found!) |
| 28 | 17.789846 | 172.26.254.254 | 172.26.0.175 | ICMP | 70 | Time-to-live exceeded (Time to live exceeded in transit) |
| 58 | 23.308666 | 172.26.0.175 | 193.136.9.240 | ICMP | 106 | Echo (ping) request id=0x0001, seq=13/3328, ttl=2 (no response found!) |
| 59 | 23.317121 | 172.16.2.1 | 172.26.0.175 | ICMP | 70 | Time-to-live exceeded (Time to live exceeded in transit) |
| 60 | 23.319653 | 172.26.0.175 | 193.136.9.240 | ICMP | 106 | Echo (ping) request id=0x0001, seq=14/3584, ttl=2 (no response found!) |

| Frame 23: 106 bytes on wire (848 bits), 106 bytes captured (848 bits) on interface \Device\NPF{...} | |
|---|--|
| Ethernet II, Src: AzureWav_81:c8:ab (48:e7:da:81:c8:ab), Dst: ComdaEnt_ff:94:00 (00:d0:03:ff:94:00) | |
| Internet Protocol Version 4, Src: 172.26.0.175, Dst: 193.136.9.240 | |
| 0100 = Version: 4 | |
| 0101 = Header Length: 20 bytes (5) | |
| > Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT) | |
| Total Length: 92 | |
| Identification: 0xf4a3 (62627) | |
| > 0000 = Flags: 0x0 | |
| ...0 0000 0000 0000 = Fragment Offset: 0 | |
| > Time to Live: 1 | |
| Protocol: ICMP (1) | |
| Header Checksum: 0x4bc [validation disabled] | |
| [Header checksum status: Unverified] | |
| Source Address: 172.26.0.175 | |

Protocolo: ICMP (1)

Este valor - 1 - identifica o protocolo usado na transmissão da mensagem - ICMP.

(c) Quantos bytes tem o cabeçalho IPv4? Quantos bytes tem o campo de dados (payload) do datagrama? Como se calcula o tamanho do payload?

| No. | Time | Source | Destination | Protocol | Length | Info |
|-----|-----------|----------------|---------------|----------|--------|--|
| 23 | 17.771271 | 172.26.0.175 | 193.136.9.240 | ICMP | 106 | Echo (ping) request id=0x0001, seq=10/2560, ttl=1 (no response found!) |
| 24 | 17.780123 | 172.26.254.254 | 172.26.0.175 | ICMP | 70 | Time-to-live exceeded (Time to live exceeded in transit) |
| 25 | 17.782096 | 172.26.0.175 | 193.136.9.240 | ICMP | 106 | Echo (ping) request id=0x0001, seq=11/2816, ttl=1 (no response found!) |
| 26 | 17.785533 | 172.26.254.254 | 172.26.0.175 | ICMP | 70 | Time-to-live exceeded (Time to live exceeded in transit) |
| 27 | 17.787446 | 172.26.0.175 | 193.136.9.240 | ICMP | 106 | Echo (ping) request id=0x0001, seq=12/3072, ttl=1 (no response found!) |
| 28 | 17.789846 | 172.26.254.254 | 172.26.0.175 | ICMP | 70 | Time-to-live exceeded (Time to live exceeded in transit) |
| 58 | 23.308666 | 172.26.0.175 | 193.136.9.240 | ICMP | 106 | Echo (ping) request id=0x0001, seq=13/3328, ttl=2 (no response found!) |
| 59 | 23.317121 | 172.16.2.1 | 172.26.0.175 | ICMP | 70 | Time-to-live exceeded (Time to live exceeded in transit) |
| 60 | 23.319653 | 172.26.0.175 | 193.136.9.240 | ICMP | 106 | Echo (ping) request id=0x0001, seq=14/3584, ttl=2 (no response found!) |

| Frame 23: 106 bytes on wire (848 bits), 106 bytes captured (848 bits) on interface \Device\NPF{...} | |
|---|--|
| Ethernet II, Src: AzureWav_81:c8:ab (48:e7:da:81:c8:ab), Dst: ComdaEnt_ff:94:00 (00:d0:03:ff:94:00) | |
| Internet Protocol Version 4, Src: 172.26.0.175, Dst: 193.136.9.240 | |
| 0100 = Version: 4 | |
| 0101 = Header Length: 20 bytes (5) | |
| > Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT) | |
| Total Length: 92 | |
| Identification: 0xf4a3 (62627) | |
| > 0000 = Flags: 0x0 | |
| ...0 0000 0000 0000 = Fragment Offset: 0 | |
| > Time to Live: 1 | |
| Protocol: ICMP (1) | |
| Header Checksum: 0x4bc [validation disabled] | |
| [Header checksum status: Unverified] | |
| Source Address: 172.26.0.175 | |

O cabeçalho IPv4 tem 20 bytes.

O número total de bytes do campo é de 92 bytes.

O payload tem $92 - 20 = 72$ bytes.

(d) O datagrama IP foi fragmentado? Justifique

```

  000. .... = Flags: 0x0
    0... .... = Reserved bit: Not set
    .0... .... = Don't fragment: Not set
    ..0. .... = More fragments: Not set
    ...0 0000 0000 0000 = Fragment Offset: 0

```

O Fragment Offset tem valor 0, logo este datagrama não foi fragmentado.
O Fragment Offset tendo valor 0, pode também indicar que este é o primeiro fragmento do datagrama, no entanto não existem mais fragmentos (More fragments tem valor 0), logo o datagrama não foi fragmentado.

(e) Ordene os pacotes capturados de acordo com o endereço IP fonte, e analise a sequência de tráfego ICMP gerado a partir do endereço IP atribuído à interface da sua máquina. Para a sequência de mensagens ICMP enviadas pelo seu computador, indique que campos do cabeçalho IP variam de pacote para pacote.

| No. | Time | Source | Destination | Protocol | Length | Info |
|-----|-----------|----------------|---------------|----------|--------|--|
| 23 | 17.771271 | 172.26.0.175 | 193.136.9.240 | ICMP | 106 | Echo (ping) request id=0x0001, seq=10/2560, ttl=1 (no response found!) |
| 24 | 17.780123 | 172.26.254.254 | 172.26.0.175 | ICMP | 70 | Time-to-live exceeded (Time to live exceeded in transit) |
| 25 | 17.782096 | 172.26.0.175 | 193.136.9.240 | ICMP | 106 | Echo (ping) request id=0x0001, seq=11/2816, ttl=1 (no response found!) |
| 26 | 17.785533 | 172.26.254.254 | 172.26.0.175 | ICMP | 70 | Time-to-live exceeded (Time to live exceeded in transit) |
| 27 | 17.787446 | 172.26.0.175 | 193.136.9.240 | ICMP | 106 | Echo (ping) request id=0x0001, seq=12/3072, ttl=1 (no response found!) |
| 28 | 17.789846 | 172.26.254.254 | 172.26.0.175 | ICMP | 70 | Time-to-live exceeded (Time to live exceeded in transit) |
| 58 | 23.308666 | 172.26.0.175 | 193.136.9.240 | ICMP | 106 | Echo (ping) request id=0x0001, seq=13/3328, ttl=2 (no response found!) |
| 59 | 23.317121 | 172.16.2.1 | 172.26.0.175 | ICMP | 70 | Time-to-live exceeded (Time to live exceeded in transit) |
| 60 | 23.319653 | 172.26.0.175 | 193.136.9.240 | ICMP | 106 | Echo (ping) request id=0x0001, seq=14/3584, ttl=2 (no response found!) |
| 61 | 23.322111 | 172.16.2.1 | 172.26.0.175 | ICMP | 70 | Time-to-live exceeded (Time to live exceeded in transit) |
| 62 | 23.324838 | 172.26.0.175 | 193.136.9.240 | ICMP | 106 | Echo (ping) request id=0x0001, seq=15/3840, ttl=2 (no response found!) |
| 63 | 23.328891 | 172.16.2.1 | 172.26.0.175 | ICMP | 70 | Time-to-live exceeded (Time to live exceeded in transit) |
| 92 | 28.890738 | 172.26.0.175 | 193.136.9.240 | ICMP | 106 | Echo (ping) request id=0x0001, seq=16/4096, ttl=3 (no response found!) |
| 93 | 28.899445 | 172.16.115.252 | 172.26.0.175 | ICMP | 70 | Time-to-live exceeded (Time to live exceeded in transit) |
| 94 | 28.901851 | 172.26.0.175 | 193.136.9.240 | ICMP | 106 | Echo (ping) request id=0x0001, seq=17/4352, ttl=3 (no response found!) |
| 95 | 28.904560 | 172.16.115.252 | 172.26.0.175 | ICMP | 70 | Time-to-live exceeded (Time to live exceeded in transit) |
| 96 | 28.908490 | 172.26.0.175 | 193.136.9.240 | ICMP | 106 | Echo (ping) request id=0x0001, seq=18/4608, ttl=3 (no response found!) |
| 97 | 28.911528 | 172.16.115.252 | 172.26.0.175 | ICMP | 70 | Time-to-live exceeded (Time to live exceeded in transit) |
| 111 | 34.456710 | 172.26.0.175 | 193.136.9.240 | ICMP | 106 | Echo (ping) request id=0x0001, seq=19/4864, ttl=4 (reply in 112) |
| 112 | 34.465858 | 193.136.9.240 | 172.26.0.175 | ICMP | 106 | Echo (ping) reply id=0x0001, seq=19/4864, ttl=61 (request in 111) |
| 113 | 34.468446 | 172.26.0.175 | 193.136.9.240 | ICMP | 106 | Echo (ping) request id=0x0001, seq=20/5120, ttl=4 (reply in 114) |
| 114 | 34.471115 | 193.136.9.240 | 172.26.0.175 | ICMP | 106 | Echo (ping) reply id=0x0001, seq=20/5120, ttl=61 (request in 113) |
| 115 | 34.473705 | 172.26.0.175 | 193.136.9.240 | ICMP | 106 | Echo (ping) request id=0x0001, seq=21/5376, ttl=4 (reply in 116) |
| 116 | 34.478294 | 193.136.9.240 | 172.26.0.175 | ICMP | 106 | Echo (ping) reply id=0x0001, seq=21/5376, ttl=61 (request in 115) |

Campos: 1) "Time to Live"
2) "Header Checksum"
3) "Identification"

(f) Indique o padrão observado nos valores do campo de Identificação do datagrama IP e TTL.

Podemos observar que o campo de Identificação incrementa uma unidade. Ao fim de três incrementos (do Identification), o campo TTL aumenta uma unidade.

(g) Ordene o tráfego capturado por endereço destino e encontre a série de respostas ICMP TTL exceeded enviadas ao seu computador. Qual é o valor do campo TTL? Esse valor permanece constante para todas as mensagens de resposta ICMP TTL exceeded enviados ao seu host? Porquê?

Usando o filtro “icmp && ip.dst==172.26.0.175”, obtivemos:

| icmp && ip.dst==172.26.0.175 | | | | | | | |
|------------------------------|-----------|----------------|--------------|----------|--------|---|--|
| No. | Time | Source | Destination | Protocol | Length | Info | |
| 4 | 4.356263 | 172.26.254.254 | 172.26.0.175 | ICMP | 70 | Time-to-live exceeded (Time to live exceeded in transit) | |
| 6 | 4.360240 | 172.26.254.254 | 172.26.0.175 | ICMP | 70 | Time-to-live exceeded (Time to live exceeded in transit) | |
| 8 | 4.364035 | 172.26.254.254 | 172.26.0.175 | ICMP | 70 | Time-to-live exceeded (Time to live exceeded in transit) | |
| 41 | 9.897978 | 172.16.2.1 | 172.26.0.175 | ICMP | 70 | Time-to-live exceeded (Time to live exceeded in transit) | |
| 43 | 9.900999 | 172.16.2.1 | 172.26.0.175 | ICMP | 70 | Time-to-live exceeded (Time to live exceeded in transit) | |
| 45 | 9.904008 | 172.16.2.1 | 172.26.0.175 | ICMP | 70 | Time-to-live exceeded (Time to live exceeded in transit) | |
| 207 | 15.449332 | 172.16.115.252 | 172.26.0.175 | ICMP | 70 | Time-to-live exceeded (Time to live exceeded in transit) | |
| 209 | 15.453161 | 172.16.115.252 | 172.26.0.175 | ICMP | 70 | Time-to-live exceeded (Time to live exceeded in transit) | |
| 211 | 15.457424 | 172.16.115.252 | 172.26.0.175 | ICMP | 70 | Time-to-live exceeded (Time to live exceeded in transit) | |
| 280 | 20.998858 | 193.136.9.240 | 172.26.0.175 | ICMP | 106 | Echo (ping) reply id=0x0001, seq=19/4864, ttl=61 (request in 279) | |
| 282 | 21.002042 | 193.136.9.240 | 172.26.0.175 | ICMP | 106 | Echo (ping) reply id=0x0001, seq=20/5120, ttl=61 (request in 281) | |
| 284 | 21.005144 | 193.136.9.240 | 172.26.0.175 | ICMP | 106 | Echo (ping) reply id=0x0001, seq=21/5376, ttl=61 (request in 283) | |

Daqui, de três em três, retiramos as seguintes informações:

✓ Internet Protocol Version 4, Src: 172.26.254.254, Dst: 172.26.0.175

0100 = Version: 4

.... 0101 = Header Length: 20 bytes (5)

> Differentiated Services Field: 0xc0 (DSCP: CS6, ECN: Not-ECT)

Total Length: 56

Identification: 0x0735 (1845)

> 000. = Flags: 0x0

...0 0000 0000 0000 = Fragment Offset: 0

Time to Live: 255

Protocol: ICMP (1)

Header Checksum: 0x5bed [validation disabled]

[Header checksum status: Unverified]

Source Address: 172.26.254.254

Destination Address: 172.26.0.175

> Internet Control Message Protocol

```

0000  48 e7 da 81 c8 ab 00 d0  03 ff 94 00 08 00 45 c0  H.....E.
0010  00 38 07 35 00 00 ff 01  5b ed ac 1a fe fe ac 1a  .8.5....[.....
0020  00 af 0b 00 f4 ff 00 00  00 00 45 00 00 5c be 49  .....E..\I
0030  00 00 01 01 83 16 ac 1a  00 af c1 88 09 f0 08 00  .....
0040  f7 f4 00 01 00 0a

```



```

> Ethernet II, Src: ComdaEnt_ff:94:00 (00:d0:03:ff:94:00), Dst: AzureWav_81:c8:ab (4
✓ Internet Protocol Version 4, Src: 172.16.2.1, Dst: 172.26.0.175
    0100 .... = Version: 4
    .... 0101 = Header Length: 20 bytes (5)
> Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
    Total Length: 56
    Identification: 0x3360 (13152)
> 000. .... = Flags: 0x0
    ...0 0000 0000 0000 = Fragment Offset: 0
    Time to Live: 254
    Protocol: ICMP (1)
    Header Checksum: 0x2e8a [validation disabled]
    [Header checksum status: Unverified]
    Source Address: 172.16.2.1
    Destination Address: 172.26.0.175

```

| | | |
|------|---|------------|
| 0000 | 48 e7 da 81 c8 ab 00 d0 03 ff 94 00 08 00 45 00 | H.....E. |
| 0010 | 00 38 33 60 00 00 fe 01 2e 8a ac 10 02 01 ac 1a | .83`... .. |
| 0020 | 00 af 0b 00 f4 ff 00 00 00 00 45 00 00 5c be 4c |E..\L |
| 0030 | 00 00 01 01 83 13 ac 1a 00 af c1 88 09 f0 08 00 | |
| 0040 | f7 f1 00 01 00 0d | |

```

> Frame 207: 70 bytes on wire (560 bits), 70 bytes captured (560 bits) on interface \
> Ethernet II, Src: ComdaEnt_ff:94:00 (00:d0:03:ff:94:00), Dst: AzureWav_81:c8:ab (48
✓ Internet Protocol Version 4, Src: 172.16.115.252, Dst: 172.26.0.175
    0100 .... = Version: 4
    .... 0101 = Header Length: 20 bytes (5)
> Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
    Total Length: 56
    Identification: 0x463a (17978)
> 000. .... = Flags: 0x0
    ...0 0000 0000 0000 = Fragment Offset: 0
    Time to Live: 253
    Protocol: ICMP (1)
    Header Checksum: 0xaab4 [validation disabled]
    [Header checksum status: Unverified]
    Source Address: 172.16.115.252

```

| | | |
|------|---|--------------|
| 0000 | 48 e7 da 81 c8 ab 00 d0 03 ff 94 00 08 00 45 00 | H.....E. |
| 0010 | 00 38 46 3a 00 00 fd 01 aa b4 ac 10 73 fc ac 1a | .8F:....s... |
| 0020 | 00 af 0b 00 f4 ff 00 00 00 00 45 00 00 5c be 4f |E..\O |
| 0030 | 00 00 01 01 83 10 ac 1a 00 af c1 88 09 f0 08 00 | |
| 0040 | f7 ee 00 01 00 10 | |

Conseguimos observar que o valor do TTL, não só não permanece constante, como decrementa um valor. Isto, cada vez que a mensagem/o pacote “passa” por um router, isto é, quando “dá um salto”, o valor TTL, decrementa.

O valor TTL inicialmente é algo que “grande”, pois assim assegura que o pacote chegará sempre ao seu destinatário, independentemente da quantidade de saltos que este terá de dar.

Pergunta 3) Pretende-se agora analisar a fragmentação de pacotes IP. Capture com o Wireshark o tráfego gerado pelo comando ping -l 5215 marco.uminho.pt

Utilizada a opção -l, por termos examinado a captura em Windows e 5215 referente ao grupo 15.

(a) Localize a primeira mensagem ICMP. Porque é que houve necessidade de fragmentar o pacote inicial?

Sempre que o tamanho do datagrama é superior a 1480 bytes (correspondente a 1500 bytes menos os 20 bytes do cabeçalho), o pacote é fragmentado. Neste caso o tamanho do datagrama está definido para 5215 bytes, daí a necessidade de fragmentar o pacote inicial.

(b) Imprima o primeiro fragmento do datagrama IP segmentado. Que informação no cabeçalho indica que o datagrama foi fragmentado? Que informação no cabeçalho IP indica que se trata do primeiro fragmento? Qual é o tamanho deste datagrama IP?

```
Frame 7: 1514 bytes on wire (12112 bits), 1514 bytes captured (12112 bits) on interface \Device\NPF_{4FBFF88F-2175-4C99-8F88-8881AF02FDEB}, id 0
Ethernet II, Src: IntelCor_eb:8f:34 (40:ec:99:eb:8f:34), Dst: ComdaEnt_ff:94:00 (00:d0:03:ff:94:00)
Internet Protocol Version 4, Src: 172.26.13.120, Dst: 193.136.9.240
  0100 .... = Version: 4
  .... 0101 = Header Length: 20 bytes (5)
  > Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
  Total Length: 1500
  Identification: 0xa590 (42384)
  v Flags: 0x20, More fragments
    0... .... = Reserved bit: Not set
    .0... .... = Don't fragment: Not set
    ..1. .... = More fragments: Set
  Fragment Offset: 0
  Time to Live: 128
  Protocol: ICMP (1)
  Header Checksum: 0x0000 [validation disabled]
  [Header checksum status: Unverified]
  Source Address: 172.26.13.120
  Destination Address: 193.136.9.240
  [Reassembled IPv4 in frame: 10]
Data (1480 bytes)
```

Se, no campo More Fragments, o bit MF estiver a 0, significa que este é o último fragmento. Como podemos ver na imagem acima, o campo *More Fragments* está definido como *Set* e o bit está a 1. Isto mostra que este não é o último fragmento, logo o datagrama foi fragmentado.

Se se tratar do primeiro fragmento, o campo *Fragment Offset* deve estar a 0, pois este valor teria de ser (0+1480) caso se tratasse do segundo fragmento

e $(0+1480+1480)$ caso se tratasse do terceiro fragmento. Analisando a imagem, podemos confirmar que este valor está a 0, e por isso, comprova-se que este é o primeiro fragmento.

O tamanho do datagrama IP, como podemos ver na imagem acima, no campo *Total Length*, é 1500 bytes, pois cada fragmento, exceto o último, leva o máximo de bytes possível.

(c) Imprima o segundo fragmento do datagrama IP original. Que informação do cabeçalho IP indica que não se trata do primeiro fragmento? Há mais fragmentos? O que nos permite afirmar isso?

```
Frame 8: 1514 bytes on wire (12112 bits), 1514 bytes captured (12112 bits) on interface \Device\NPF_{4FBFF88F-2175-4C99-8F88-8881AF02FDEB}, id 0
Ethernet II, Src: IntelCor_eb:8f:34 (40:ec:99:eb:8f:34), Dst: ComdaEnt_ff:94:00 (00:d0:03:ff:94:00)
Internet Protocol Version 4, Src: 172.26.13.120, Dst: 193.136.9.240
  0100 .... = Version: 4
  .... 0101 = Header Length: 20 bytes (5)
  > Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
  Total Length: 1500
  Identification: 0xa590 (42384)
  ✓ Flags: 0x20, More fragments
    0... .... = Reserved bit: Not set
    .0.. .... = Don't fragment: Not set
    ..1. .... = More fragments: Set
  Fragment Offset: 1480
  Time to Live: 128
  Protocol: ICMP (1)
  Header Checksum: 0x0000 [validation disabled]
  [Header checksum status: Unverified]
  Source Address: 172.26.13.120
  Destination Address: 193.136.9.240
  [Reassembled IPv4 in frame: 10]
Data (1480 bytes)
```

Se fosse o primeiro fragmento, o valor do campo *Fragment Offset* teria de ser 0, podemos ver na imagem que é 1480, que corresponde ao segundo fragmento.

Sim, existem mais fragmentos, porque no campo *More Fragments*, o último bit está a 1, para ser o último, teria de estar a 0.

(d) Quantos fragmentos foram criados a partir do datagrama original? Como se detecta o último fragmento correspondente ao datagrama original? Estabeleça um filtro no Wireshark que permita listar o último fragmento do datagrama IP segmentado.

```
✓ [4 IPv4 Fragments (5223 bytes): #7(1480), #8(1480), #9(1480), #10(783)]
[Frame: 7, payload: 0-1479 (1480 bytes)]
[Frame: 8, payload: 1480-2959 (1480 bytes)]
[Frame: 9, payload: 2960-4439 (1480 bytes)]
[Frame: 10, payload: 4440-5222 (783 bytes)]
[Fragment count: 4]
[Reassembled IPv4 length: 5223]
[Reassembled IPv4 data: 080082ce000100246162636465666768696a6b6c6d6e6f70717273747576776162636465...]
```

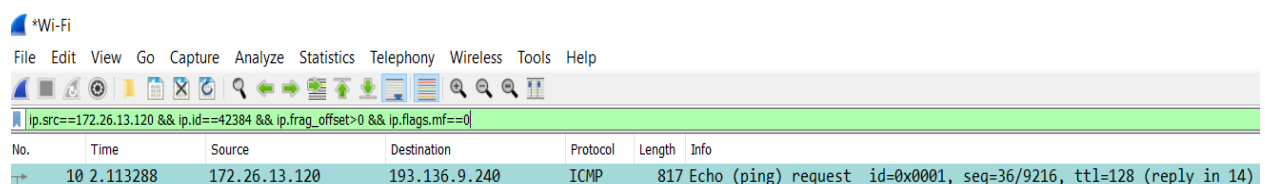
Foram criados 4 fragmentos a partir do datagrama original.

```
✓ Flags: 0x02
  0... .... = Reserved bit: Not set
  .0.. .... = Don't fragment: Not set
  ..0. .... = More fragments: Not set
Fragment Offset: 4440
```

O último bit do campo *More Fragments* está a 0 e o campo *Fragment Offset* está a 4440 (ou seja, maior que 0), que é o resultado de somar 1480 ao valor de cada campo *Fragment Offset* desde o primeiro fragmento até ao quarto.

$$0+1480=1480+1480=2960+1480=4440$$

Logo, este é o último fragmento do datagrama original.



Através do filtro:

```
ip.src==172.26.13.120 && ip.id==42384 && ip.frag_offset>0 &&
ip.flags.mf==0
```

Conseguimos ver que este é o último fragmento do datagrama enviado do IP 172.26.13.120 para o IP id=42384. Pois está filtrado para ter o Offset superior a 0, e o Bit MF igual a 0.

(e) Indique, resumindo, os campos que mudam no cabeçalho IP entre os diferentes fragmentos, e explique a forma como essa informação permite reconstruir o datagrama original.

Os campos que mudam no cabeçalho IP entre os fragmentos, é o valor do Campo Offset, que aumenta 1480 em cada fragmento e, no caso do último fragmento, altera-se o bit do Campo More Fragments, que fica a 0. No último fragmento, também tem a informação sobre a quantidade de fragmentos, incluindo o tamanho de cada fragmento.

Como o campo Identification é igual em todos os fragmentos, é possível saber quais dos fragmentos vão pertencer a um datagrama.

Quando chega ao último fragmento, utiliza a informação do campo Offset para remontar o datagrama.

(f) Determine o valor máximo de SIZE sem que ocorra fragmentação do pacote. Justifique o valor obtido.

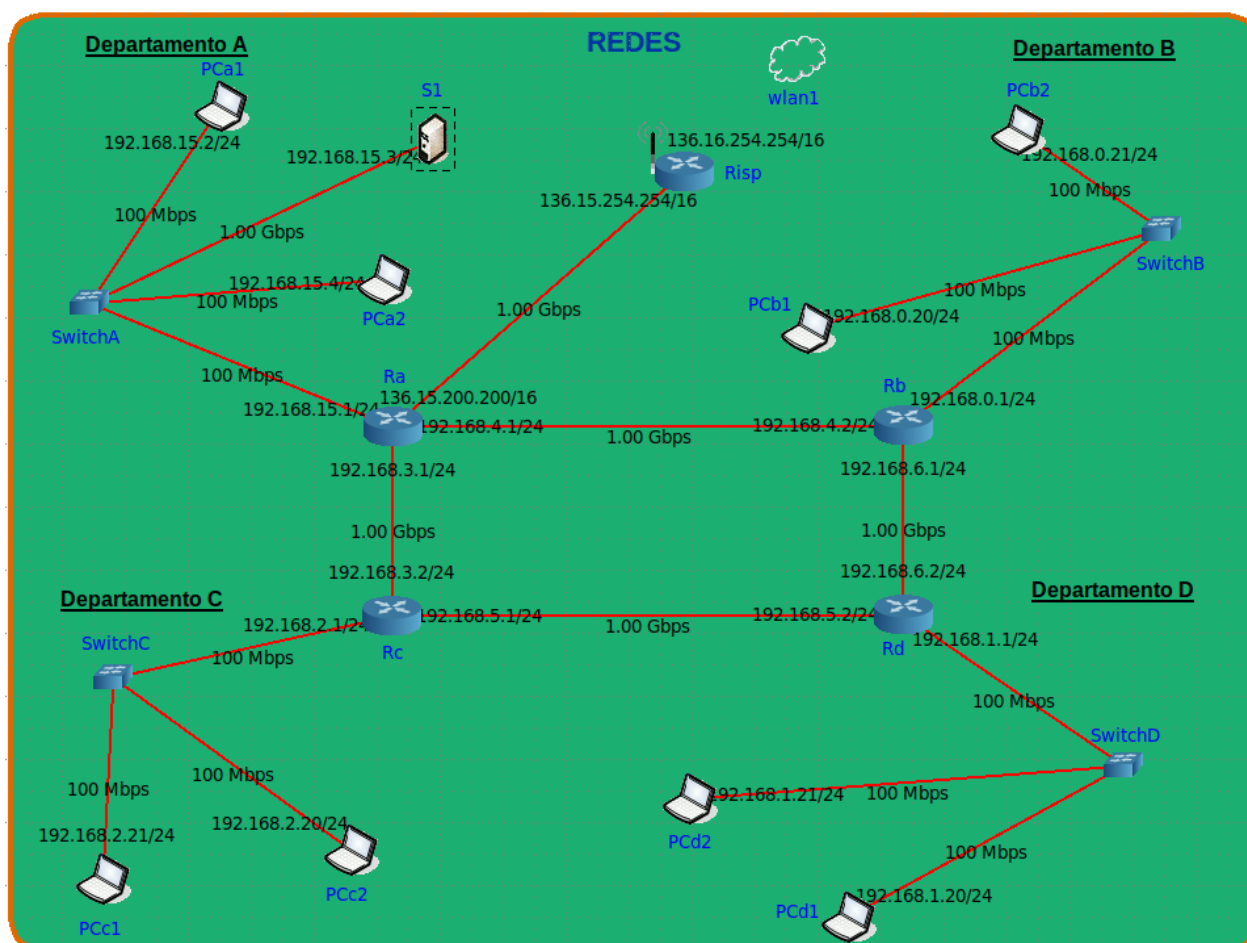
Como o máximo é 1500 bytes, temos de tirar os 20 bytes do cabeçalho, e temos ainda de tirar 8 bytes do cabeçalho ICMP que resulta em 1472 bytes. Por isso, este é o valor máximo de SIZE para que não ocorra fragmentação do pacote.

PARTE 2

2. Endereçamento e Encaminhamento IP

Pergunta 1) Atenda aos endereços IP atribuídos automaticamente pelo CORE aos diversos equipamentos da topologia.

(a) Indique que endereços IP e máscaras de rede foram atribuídos pelo CORE a cada equipamento. Para simplificar, pode incluir uma imagem que ilustre de forma clara a topologia definida e o endereçamento usado.



(b) *Tratam-se de endereços públicos ou privados? Porquê?*

Os endereços do tipo 192.168.../24 são da Classe C, com um Range de IP Privado.

Os endereços do tipo 136.15.../16 são da classe B, com um Range de IP Público.

Os departamentos A,B,C e D tratam-se de redes privadas, pois são redes de Classe C e os endereços vão de 192.168.0.0 até 192.168.255.255 .

Por outro lado, a cloud é uma rede pública, de Classe B, uma vez que o seu endereço está entre 128.0.0.0 to 191.255.0.0 .

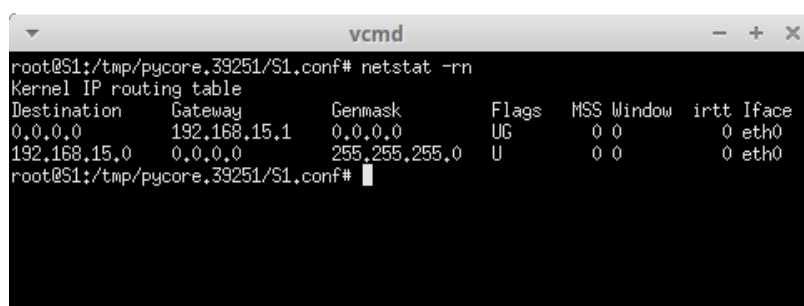
(c) *Por que razão o CORE não atribui um endereço IP aos switches? Faz sentido na prática um switch ter um endereço IP? Justifique.*

O core não atribui endereços IP aos switches pois estes são equipamentos de nível 2, não reconhecendo endereços IP's, apenas trabalham com endereços MAC.

Dispositivos como hubs, switches e pontos de acesso sem fio não precisam de endereços IPv4 para operar como dispositivos intermediários.

Pergunta 2) *Para o router RA e o servidor S1 do departamento A:*

(a) *Execute o comando netstat -rn por forma a poder consultar a tabela de encaminhamento unicast (IPv4). Inclua no seu relatório as tabelas de encaminhamento obtidas. Interprete as várias entradas de cada tabela. Se necessário, consulte o manual respectivo (man netstat).*



```
vcmd
root@S1:/tmp/pycore.39251/S1.conf# netstat -rn
Kernel IP routing table
Destination    Gateway         Genmask         Flags   MSS Window  irtt  Iface
0.0.0.0        192.168.15.1   0.0.0.0         UG      0 0        0     eth0
192.168.15.0   0.0.0.0        255.255.255.0   U       0 0        0     eth0
root@S1:/tmp/pycore.39251/S1.conf#
```

Tabela de encaminhamento de S1

```
vcmd
root@Ra:/tmp/pycore.39251/Ra.conf# netstat -rn
Kernel IP routing table
Destination      Gateway          Genmask         Flags   MSS Window  irtt  Iface
136.15.0.0        0.0.0.0          255.255.0.0     U        0 0        0 eth3
136.16.0.0        136.15.254.254  255.255.0.0     UG       0 0        0 eth3
192.168.0.0        192.168.4.2     255.255.255.0   UG       0 0        0 eth2
192.168.1.0        192.168.3.2     255.255.255.0   UG       0 0        0 eth1
192.168.2.0        192.168.3.2     255.255.255.0   UG       0 0        0 eth1
192.168.3.0        0.0.0.0          255.255.255.0   U        0 0        0 eth1
192.168.4.0        0.0.0.0          255.255.255.0   U        0 0        0 eth2
192.168.5.0        192.168.3.2     255.255.255.0   UG       0 0        0 eth1
192.168.6.0        192.168.4.2     255.255.255.0   UG       0 0        0 eth2
192.168.15.0       0.0.0.0          255.255.255.0   U        0 0        0 eth0
root@Ra:/tmp/pycore.39251/Ra.conf#
```

Tabela de encaminhamento de Ra

A entrada **Destination** nas tabelas de encaminhamento, mostram o IP da rede destino.

O **Gateway** indica o endereço IP que permite chegar à rede destino.

A **Genmask** é a máscara de cada rede. Esta é 0.0.0.0 para o *Default*.

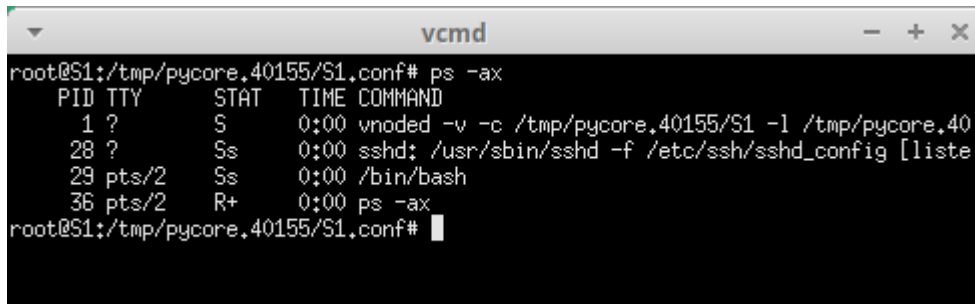
As **Flags**, neste caso, podem ser U ou G, a flag U indica que a rede está disponível e a flag G indica que é para usar o Gateway, essencialmente informa que o próximo nó é um Router.

O **MSS** é o tamanho máximo do segmento padrão para conexões TCP ao longo deste caminho e a **Window** é o tamanho da janela padrão para conexões TCP neste caminho.

O **IRTT** é o tempo de Round-Trip inicial. É útil para determinar os melhores parâmetros do protocolo TCP sem delay.

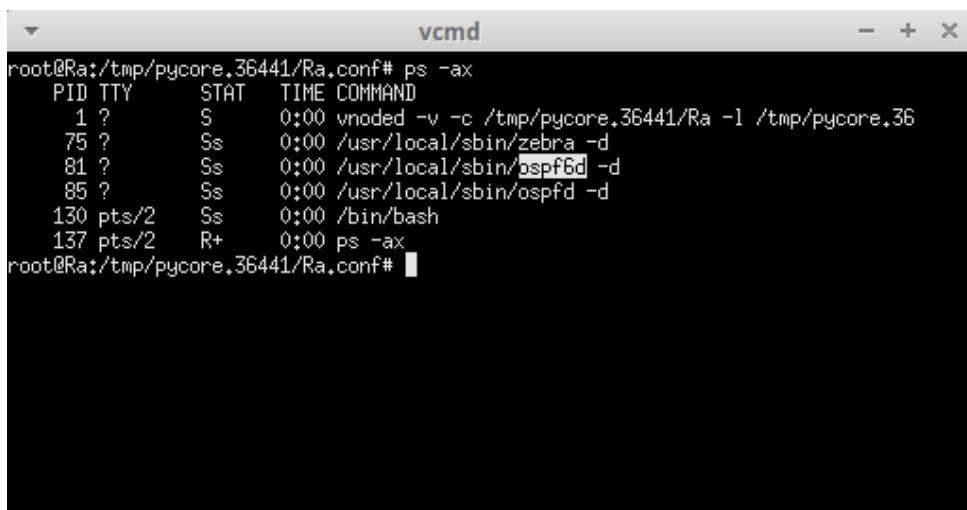
O **Iface** é a interface para onde serão enviados os pacotes desta rota.

(b) Diga, justificando, se está a ser usado encaminhamento estático ou dinâmico (sugestão: analise os processos que estão a correr em cada sistema (hosts, routers) usando, por exemplo, o comando ps -ax).

A terminal window titled 'vcmd' showing the output of the 'ps -ax' command on Host S1. The output lists four processes: PID 1 (vnode), PID 28 (sshd), PID 29 (bash), and PID 36 (ps -ax).

```
vcmd
root@S1:/tmp/pycore.40155/S1.conf# ps -ax
  PID TTY          STAT       TIME COMMAND
    1 ?            S          0:00 vnode -v -c /tmp/pycore.40155/S1 -l /tmp/pycore.40
   28 ?            Ss         0:00 sshd: /usr/sbin/sshd -f /etc/ssh/sshd_config [liste
   29 pts/2        Ss         0:00 /bin/bash
   36 pts/2        R+         0:00 ps -ax
root@S1:/tmp/pycore.40155/S1.conf#
```

Como podemos ver, quando se analisa os processos que estão a correr no Host S1, entende-se que está a ser usado um encaminhamento estático.

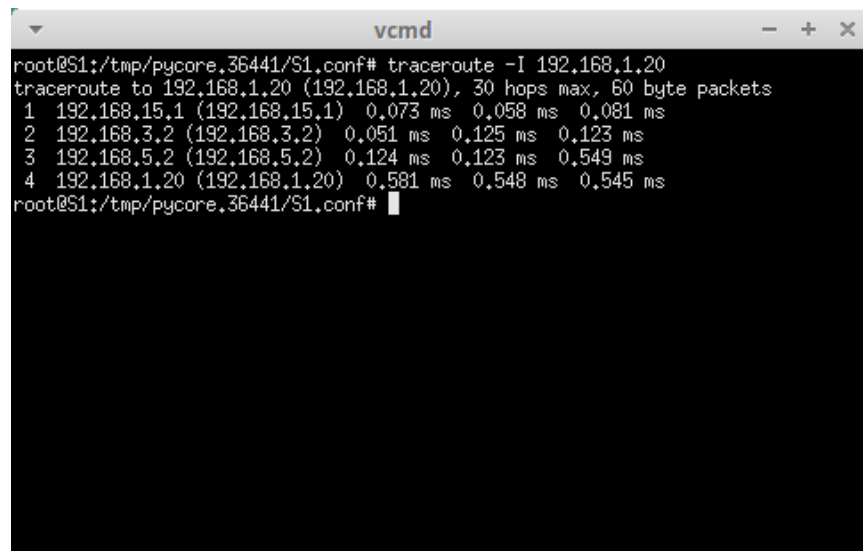
A terminal window titled 'vcmd' showing the output of the 'ps -ax' command on Router Ra. The output lists six processes: PID 1 (vnode), PID 75 (zebra), PID 81 (ospf6d), PID 85 (ospfd), PID 130 (bash), and PID 137 (ps -ax).

```
vcmd
root@Ra:/tmp/pycore.36441/Ra.conf# ps -ax
  PID TTY          STAT       TIME COMMAND
    1 ?            S          0:00 vnode -v -c /tmp/pycore.36441/Ra -l /tmp/pycore.36
   75 ?            Ss         0:00 /usr/local/sbin/zebra -d
   81 ?            Ss         0:00 /usr/local/sbin/ospf6d -d
   85 ?            Ss         0:00 /usr/local/sbin/ospfd -d
  130 pts/2        Ss         0:00 /bin/bash
  137 pts/2        R+         0:00 ps -ax
root@Ra:/tmp/pycore.36441/Ra.conf#
```

Analisando os processos no router Ra, como há OSPF (Open Shortest Path First), significa que está a ser usado encaminhamento dinâmico, pois os routers trocam informação entre eles. Os caminhos são calculados dinamicamente, e adaptam-se a possíveis alterações na rede.

(c) Escolha um PC da organização REDES que esteja o mais distante possível em termos de saltos IP do servidor S1. Recorrendo apenas ao comando `traceroute -I`, responda às seguintes questões, justificando:

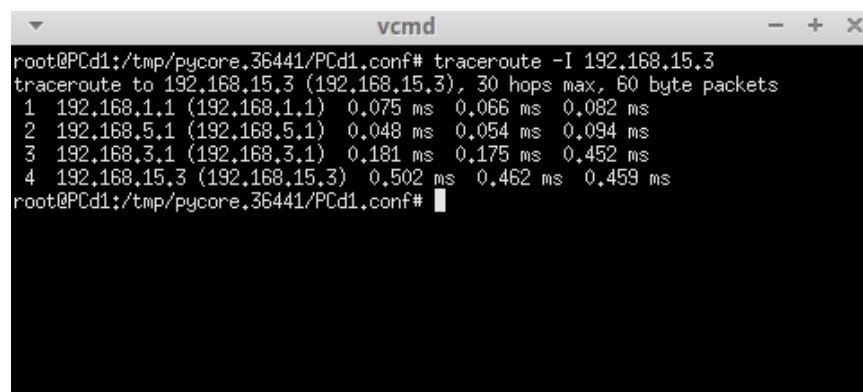
- i) Existe conectividade IP desse PC para o servidor S1? A quantos saltos IP está o PC selecionado do servidor S1?**



```
vcmd
root@S1:/tmp/pycore.36441/S1.conf# traceroute -I 192.168.1.20
traceroute to 192.168.1.20 (192.168.1.20), 30 hops max, 60 byte packets
 1 192.168.15.1 (192.168.15.1) 0.073 ms 0.058 ms 0.081 ms
 2 192.168.3.2 (192.168.3.2) 0.051 ms 0.125 ms 0.123 ms
 3 192.168.5.2 (192.168.5.2) 0.124 ms 0.123 ms 0.549 ms
 4 192.168.1.20 (192.168.1.20) 0.581 ms 0.548 ms 0.545 ms
root@S1:/tmp/pycore.36441/S1.conf#
```

De acordo com a imagem acima, existe conectividade entre o Servidor S1 e o PCd1. Este PC está a 4 saltos de distância. Na imagem podemos ver, respectivamente em cada salto, os IPs correspondentes a: Ra, Rc, Rd e PCd1.

- ii) As rotas dos pacotes ICMP echo reply são as mesmas, mas em sentido inverso, das rotas dos pacotes ICMP echo request trocadas entre esse PC e o servidor S1? (Sugestão: trace a rota de S1 para o PC selecionado).**

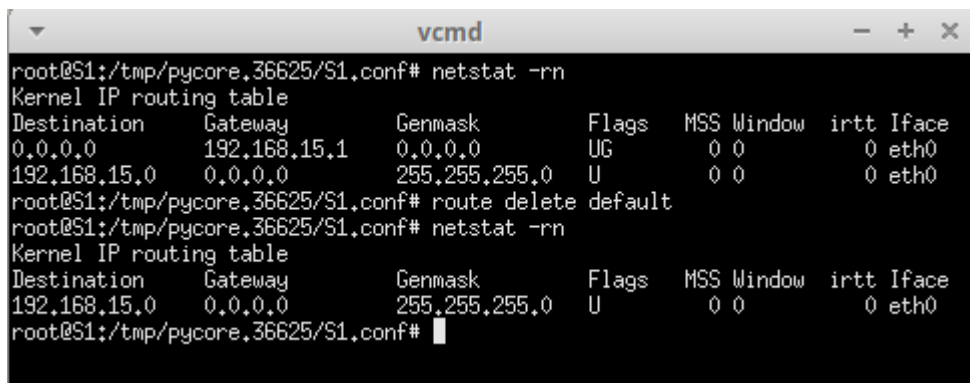


```
vcmd
root@PCd1:/tmp/pycore.36441/PCd1.conf# traceroute -I 192.168.15.3
traceroute to 192.168.15.3 (192.168.15.3), 30 hops max, 60 byte packets
 1 192.168.1.1 (192.168.1.1) 0.075 ms 0.066 ms 0.082 ms
 2 192.168.5.1 (192.168.5.1) 0.048 ms 0.054 ms 0.094 ms
 3 192.168.3.1 (192.168.3.1) 0.181 ms 0.175 ms 0.452 ms
 4 192.168.15.3 (192.168.15.3) 0.502 ms 0.462 ms 0.459 ms
root@PCd1:/tmp/pycore.36441/PCd1.conf#
```

Como podemos ver na imagem acima, as rotas utilizadas pelos pacotes ICMP echo reply são as mesmas mas em sentido inverso.

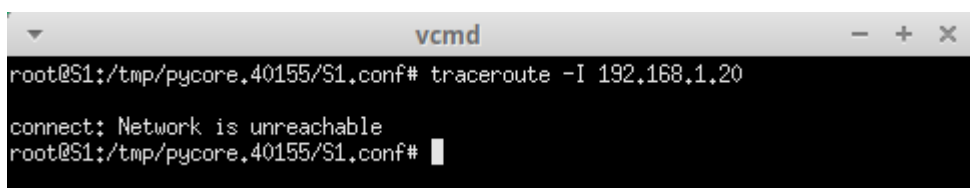
Como está a ser utilizado encaminhamento dinâmico, as rotas utilizadas poderiam ser diferentes dependendo do estado da rede, dado que estas são calculadas dinamicamente.

(d) Admita que, por questões administrativas, a rota por defeito (0.0.0.0 ou default) deve ser retirada definitivamente da tabela de encaminhamento do servidor S1 localizado no departamento A. Use o comando `route delete` para o efeito. Se necessário, consulte o manual respectivo (`man route`). Que implicações tem esta medida para os utilizadores da organização REDES que acedem ao servidor. Justifique.



```
vcmd
root@S1:/tmp/pycore.36625/S1.conf# netstat -rn
Kernel IP routing table
Destination Gateway Genmask Flags MSS Window irtt Iface
0.0.0.0 192.168.15.1 0.0.0.0 UG 0 0 0 eth0
192.168.15.0 0.0.0.0 255.255.255.0 U 0 0 0 eth0
root@S1:/tmp/pycore.36625/S1.conf# route delete default
root@S1:/tmp/pycore.36625/S1.conf# netstat -rn
Kernel IP routing table
Destination Gateway Genmask Flags MSS Window irtt Iface
192.168.15.0 0.0.0.0 255.255.255.0 U 0 0 0 eth0
root@S1:/tmp/pycore.36625/S1.conf#
```

Se retirarmos o default, os utilizadores só conseguem aceder a máquinas dentro da mesma rede, dado que a rota de acesso a máquinas ligadas a outras redes, é retirada.



```
vcmd
root@S1:/tmp/pycore.40155/S1.conf# traceroute -I 192.168.1.20
connect: Network is unreachable
root@S1:/tmp/pycore.40155/S1.conf#
```

Como podemos verificar, depois de retirar o default, partindo do S1, é impossível chegar ao IP 192.168.1.20 (PCd1).

(e) Adicione as rotas estáticas necessárias para restaurar a conectividade para o servidor S1, por forma a contornar a restrição imposta na alínea anterior. Utilize para o efeito o comando route add e registe os comandos que usou.

```
vcmd
root@S1:/tmp/pycore.35687/S1.conf# route add -net 192.168.0.0 gw 192.168.15.1 netmask 255.255.255.0
root@S1:/tmp/pycore.35687/S1.conf#
root@S1:/tmp/pycore.35687/S1.conf# route add -net 192.168.1.0 gw 192.168.15.1 netmask 255.255.255.0
root@S1:/tmp/pycore.35687/S1.conf#
root@S1:/tmp/pycore.35687/S1.conf# route add -net 192.168.2.0 gw 192.168.15.1 netmask 255.255.255.0
root@S1:/tmp/pycore.35687/S1.conf#
root@S1:/tmp/pycore.35687/S1.conf# route add -net 192.168.3.0 gw 192.168.15.1 netmask 255.255.255.0
root@S1:/tmp/pycore.35687/S1.conf#
root@S1:/tmp/pycore.35687/S1.conf# route add -net 192.168.4.0 gw 192.168.15.1 netmask 255.255.255.0
root@S1:/tmp/pycore.35687/S1.conf#
root@S1:/tmp/pycore.35687/S1.conf# route add -net 192.168.5.0 gw 192.168.15.1 netmask 255.255.255.0
root@S1:/tmp/pycore.35687/S1.conf#
root@S1:/tmp/pycore.35687/S1.conf# route add -net 192.168.6.0 gw 192.168.15.1 netmask 255.255.255.0
root@S1:/tmp/pycore.35687/S1.conf#
root@S1:/tmp/pycore.35687/S1.conf# route add -net 136.15.0.0 gw 192.168.15.1 netmask 255.255.0.0
root@S1:/tmp/pycore.35687/S1.conf#
root@S1:/tmp/pycore.35687/S1.conf# route add -net 136.16.0.0 gw 192.168.15.1 netmask 255.255.0.0
root@S1:/tmp/pycore.35687/S1.conf#
```

(f) Teste a nova política de encaminhamento garantindo que o servidor está novamente acessível, utilizando para o efeito o comando ping. Registe a nova tabela de encaminhamento do servidor S1

```
vcmd
root@S1:/tmp/pycore.35687/S1.conf# ping 192.168.0.21
PING 192.168.0.21 (192.168.0.21) 56(84) bytes of data.
64 bytes from 192.168.0.21: icmp_seq=1 ttl=62 time=0.407 ms
64 bytes from 192.168.0.21: icmp_seq=2 ttl=62 time=0.174 ms
64 bytes from 192.168.0.21: icmp_seq=3 ttl=62 time=0.169 ms
64 bytes from 192.168.0.21: icmp_seq=4 ttl=62 time=0.172 ms
^C
--- 192.168.0.21 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3068ms
rtt min/avg/max/mdev = 0.169/0.230/0.407/0.101 ms
root@S1:/tmp/pycore.35687/S1.conf#
root@S1:/tmp/pycore.35687/S1.conf#
root@S1:/tmp/pycore.35687/S1.conf# ping 192.168.2.20
PING 192.168.2.20 (192.168.2.20) 56(84) bytes of data.
64 bytes from 192.168.2.20: icmp_seq=1 ttl=62 time=0.527 ms
64 bytes from 192.168.2.20: icmp_seq=2 ttl=62 time=0.171 ms
64 bytes from 192.168.2.20: icmp_seq=3 ttl=62 time=0.156 ms
64 bytes from 192.168.2.20: icmp_seq=4 ttl=62 time=0.236 ms
^C
--- 192.168.2.20 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3058ms
rtt min/avg/max/mdev = 0.156/0.272/0.527/0.149 ms
root@S1:/tmp/pycore.35687/S1.conf#
root@S1:/tmp/pycore.35687/S1.conf#
root@S1:/tmp/pycore.35687/S1.conf# ping 192.168.1.21
PING 192.168.1.21 (192.168.1.21) 56(84) bytes of data.
64 bytes from 192.168.1.21: icmp_seq=1 ttl=61 time=0.554 ms
64 bytes from 192.168.1.21: icmp_seq=2 ttl=61 time=0.205 ms
64 bytes from 192.168.1.21: icmp_seq=3 ttl=61 time=0.207 ms
64 bytes from 192.168.1.21: icmp_seq=4 ttl=61 time=0.219 ms
^C
--- 192.168.1.21 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3069ms
rtt min/avg/max/mdev = 0.205/0.296/0.554/0.148 ms
root@S1:/tmp/pycore.35687/S1.conf#
```

Como se pode ver na imagem acima, fazendo pings para do host S1 para PCs que estão ligados a outras subredes, podemos confirmar que o servidor está de novo acessível.

```

root@S1:/tmp/pycore.35687/S1.conf# ping 136.16.254.254
PING 136.16.254.254 (136.16.254.254) 56(84) bytes of data.
64 bytes from 136.16.254.254: icmp_seq=1 ttl=63 time=0.412 ms
64 bytes from 136.16.254.254: icmp_seq=2 ttl=63 time=0.132 ms
64 bytes from 136.16.254.254: icmp_seq=3 ttl=63 time=0.132 ms
64 bytes from 136.16.254.254: icmp_seq=4 ttl=63 time=0.131 ms
^C
--- 136.16.254.254 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3053ms
rtt min/avg/max/mdev = 0.131/0.201/0.412/0.121 ms
root@S1:/tmp/pycore.35687/S1.conf#
root@S1:/tmp/pycore.35687/S1.conf#
root@S1:/tmp/pycore.35687/S1.conf# ping 136.15.254.254
PING 136.15.254.254 (136.15.254.254) 56(84) bytes of data.
64 bytes from 136.15.254.254: icmp_seq=1 ttl=63 time=0.602 ms
64 bytes from 136.15.254.254: icmp_seq=2 ttl=63 time=0.122 ms
64 bytes from 136.15.254.254: icmp_seq=3 ttl=63 time=0.131 ms
64 bytes from 136.15.254.254: icmp_seq=4 ttl=63 time=0.133 ms
^C
--- 136.15.254.254 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3055ms
rtt min/avg/max/mdev = 0.122/0.247/0.602/0.205 ms
root@S1:/tmp/pycore.35687/S1.conf#

```

Podemos também confirmar por esta imagem, que S1 consegue comunicar com o RISP.

```

vcmd
root@S1:/tmp/pycore.35687/S1.conf# netstat -rn
Kernel IP routing table

```

| Destination | Gateway | Genmask | Flags | MSS Window | irrt | Iface |
|--------------|--------------|---------------|-------|------------|------|-------|
| 136.15.0.0 | 192.168.15.1 | 255.255.0.0 | UG | 0 0 | 0 | eth0 |
| 136.16.0.0 | 192.168.15.1 | 255.255.0.0 | UG | 0 0 | 0 | eth0 |
| 192.168.0.0 | 192.168.15.1 | 255.255.255.0 | UG | 0 0 | 0 | eth0 |
| 192.168.1.0 | 192.168.15.1 | 255.255.255.0 | UG | 0 0 | 0 | eth0 |
| 192.168.2.0 | 192.168.15.1 | 255.255.255.0 | UG | 0 0 | 0 | eth0 |
| 192.168.3.0 | 192.168.15.1 | 255.255.255.0 | UG | 0 0 | 0 | eth0 |
| 192.168.4.0 | 192.168.15.1 | 255.255.255.0 | UG | 0 0 | 0 | eth0 |
| 192.168.5.0 | 192.168.15.1 | 255.255.255.0 | UG | 0 0 | 0 | eth0 |
| 192.168.6.0 | 192.168.15.1 | 255.255.255.0 | UG | 0 0 | 0 | eth0 |
| 192.168.15.0 | 0.0.0.0 | 255.255.255.0 | U | 0 0 | 0 | eth0 |

```

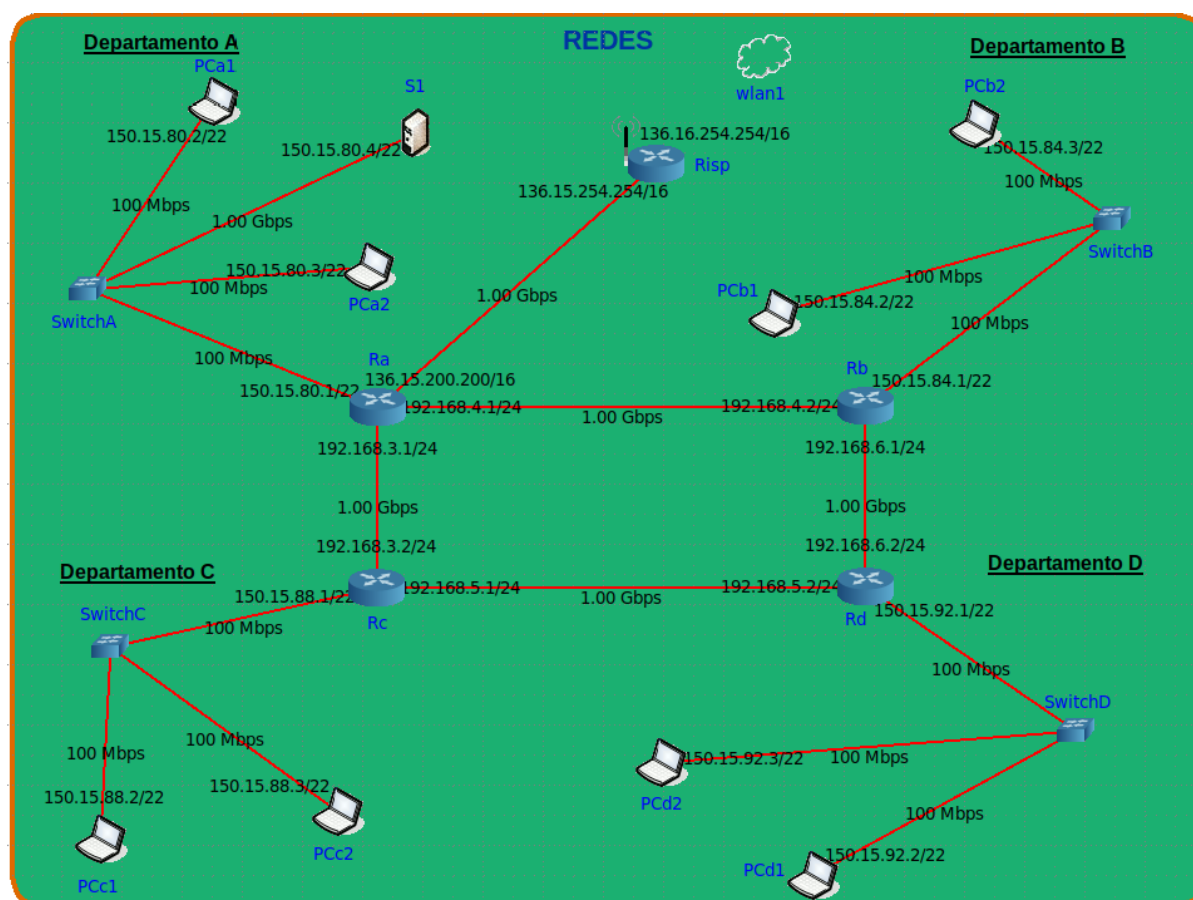
root@S1:/tmp/pycore.35687/S1.conf#

```

Nova Tabela de Encaminhamento de S1

3. Definição de Sub-redes

Pergunta 1) Considere que dispõe apenas do endereço de rede IP 150.XX.80.0/20, em que XX é o decimal correspondendo ao seu número de grupo (GXX). Defina um novo esquema de endereçamento para as redes dos departamentos (mantendo as redes de acesso e central (core) inalteradas) e atribua endereços às interfaces dos vários sistemas envolvidos. Assuma que todos os endereços de sub-redes são usáveis.



150.15.80.0/20 <- endereço fornecido
/20 -> 20 bits de rede, 12 host (32-20)
150.15.0101 0000.0/20

150.15.0101 00 00 .0/22 - 150.15.80.0/22 - end. max. ->83.254/22; end. min.
->80.1/22

150.15.0101 01 00 .0/22 - 150.15.84.0/22

150.15.0101 10 00 .0/22 - 150.15.88.0/22

150.15.0101 11 00 .0/22 - 150.15.92.0/22

(...)

Pergunta 2) Qual a máscara de rede que usou (em formato decimal)? Quantos hosts IP podem interligar no máximo em cada departamento? Quantos prefixos de sub-rede ficam disponíveis para uso futuro? Justifique.

A máscara de rede usada foi a /22 em formato Cidr, que equivale à máscara, 255.255.252.0, em formato decimal.

Podemos interligar 254 hosts (end. min. -> 150.15.xx.1/22; end. max. -> 150.15.xx.254/22)

Como usamos 2 bits para endereçar 4 sub-redes ($2^2 = 4$), não temos mais bits disponíveis para um uso futuro.

Pergunta 3) Garanta e verifique que conectividade IP entre as várias redes locais da organização REDES é mantida. Explique como procedeu.

Para provar a conectividade IP entre as várias redes, efetuamos ping das seguintes formas:

- PCa1 -> PCb1, PCc1, PCd1
- PCb1 -> PCc1, PCd1
- PCc1 -> PCd1

É necessário afirmar, que, se existe conectividade entre a máquina A e a máquina B, então, existe conectividade entre a máquina B e a máquina A.

```
root@PCa1:/tmp/pycore.37511/PCa1.conf# ping 150.15.84.2
PING 150.15.84.2 (150.15.84.2) 56(84) bytes of data.
64 bytes from 150.15.84.2: icmp_seq=1 ttl=62 time=0.806 ms
64 bytes from 150.15.84.2: icmp_seq=2 ttl=62 time=0.440 ms
64 bytes from 150.15.84.2: icmp_seq=3 ttl=62 time=0.466 ms
64 bytes from 150.15.84.2: icmp_seq=4 ttl=62 time=0.458 ms
64 bytes from 150.15.84.2: icmp_seq=5 ttl=62 time=0.452 ms
64 bytes from 150.15.84.2: icmp_seq=6 ttl=62 time=0.519 ms
^C
--- 150.15.84.2 ping statistics ---
6 packets transmitted, 6 received, 0% packet loss, time 5125ms
rtt min/avg/max/mdev = 0.440/0.523/0.806/0.128 ms
root@PCa1:/tmp/pycore.37511/PCa1.conf# ping 150.15.88.2
PING 150.15.88.2 (150.15.88.2) 56(84) bytes of data.
64 bytes from 150.15.88.2: icmp_seq=1 ttl=62 time=0.980 ms
64 bytes from 150.15.88.2: icmp_seq=2 ttl=62 time=0.453 ms
64 bytes from 150.15.88.2: icmp_seq=3 ttl=62 time=0.478 ms
64 bytes from 150.15.88.2: icmp_seq=4 ttl=62 time=0.540 ms
^C
--- 150.15.88.2 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3032ms
rtt min/avg/max/mdev = 0.453/0.612/0.980/0.214 ms
root@PCa1:/tmp/pycore.37511/PCa1.conf# ping 150.15.92.2
PING 150.15.92.2 (150.15.92.2) 56(84) bytes of data.
64 bytes from 150.15.92.2: icmp_seq=1 ttl=61 time=0.859 ms
64 bytes from 150.15.92.2: icmp_seq=2 ttl=61 time=0.573 ms
64 bytes from 150.15.92.2: icmp_seq=3 ttl=61 time=0.571 ms
64 bytes from 150.15.92.2: icmp_seq=4 ttl=61 time=0.576 ms
64 bytes from 150.15.92.2: icmp_seq=5 ttl=61 time=0.551 ms
^C
--- 150.15.92.2 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4088ms
rtt min/avg/max/mdev = 0.551/0.626/0.859/0.116 ms
root@PCa1:/tmp/pycore.37511/PCa1.conf#
```

```

root@PCb1:/tmp/pycore.37511/PCb1.conf# ping 150.15.88.2
PING 150.15.88.2 (150.15.88.2) 56(84) bytes of data.
64 bytes from 150.15.88.2: icmp_seq=1 ttl=61 time=0.801 ms
64 bytes from 150.15.88.2: icmp_seq=2 ttl=61 time=0.574 ms
64 bytes from 150.15.88.2: icmp_seq=3 ttl=61 time=0.206 ms
64 bytes from 150.15.88.2: icmp_seq=4 ttl=61 time=0.556 ms
^C
--- 150.15.88.2 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3062ms
rtt min/avg/max/mdev = 0.206/0.534/0.801/0.212 ms
root@PCb1:/tmp/pycore.37511/PCb1.conf# ping 150.15.92.2
PING 150.15.92.2 (150.15.92.2) 56(84) bytes of data.
64 bytes from 150.15.92.2: icmp_seq=1 ttl=62 time=0.776 ms
64 bytes from 150.15.92.2: icmp_seq=2 ttl=62 time=0.512 ms
64 bytes from 150.15.92.2: icmp_seq=3 ttl=62 time=0.462 ms
64 bytes from 150.15.92.2: icmp_seq=4 ttl=62 time=0.650 ms
^C
--- 150.15.92.2 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3075ms
rtt min/avg/max/mdev = 0.462/0.600/0.776/0.122 ms
root@PCb1:/tmp/pycore.37511/PCb1.conf# █

```

```

root@PCc1:/tmp/pycore.37511/PCc1.conf# ping 150.15.92.2
PING 150.15.92.2 (150.15.92.2) 56(84) bytes of data.
64 bytes from 150.15.92.2: icmp_seq=1 ttl=62 time=0.759 ms
64 bytes from 150.15.92.2: icmp_seq=2 ttl=62 time=0.454 ms
64 bytes from 150.15.92.2: icmp_seq=3 ttl=62 time=0.421 ms
64 bytes from 150.15.92.2: icmp_seq=4 ttl=62 time=1.54 ms
^C
--- 150.15.92.2 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3074ms
rtt min/avg/max/mdev = 0.421/0.793/1.540/0.450 ms
root@PCc1:/tmp/pycore.37511/PCc1.conf# █

```

CONCLUSÃO

Com este trabalho conseguimos perceber melhor todo o processo do fluxo de informação existente entre o tráfego de redes (e sub-redes). Conseguimos mitigar a exaustão dos endereços IPv4 e também reduzir os recursos de memória necessários nos routers para manter as tabelas de encaminhamento. Conseguimos concluir, também, que a definição de sub-redes permite uma melhor organização do espaço de endereçamento das redes em questão.