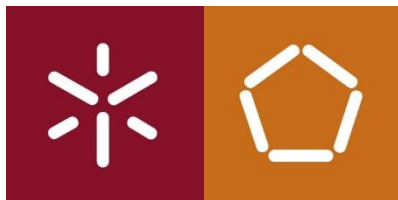


Redes de Computadores



Trabalho prático 2

14 de novembro de 2019

Grupo nº 6

Filipa Alves dos Santos (A83631)

Hugo André Coelho Cardoso (A85006)

João da Cunha e Costa (A84775)



Mestrado Integrado em Engenharia Informática

Universidade do Minho

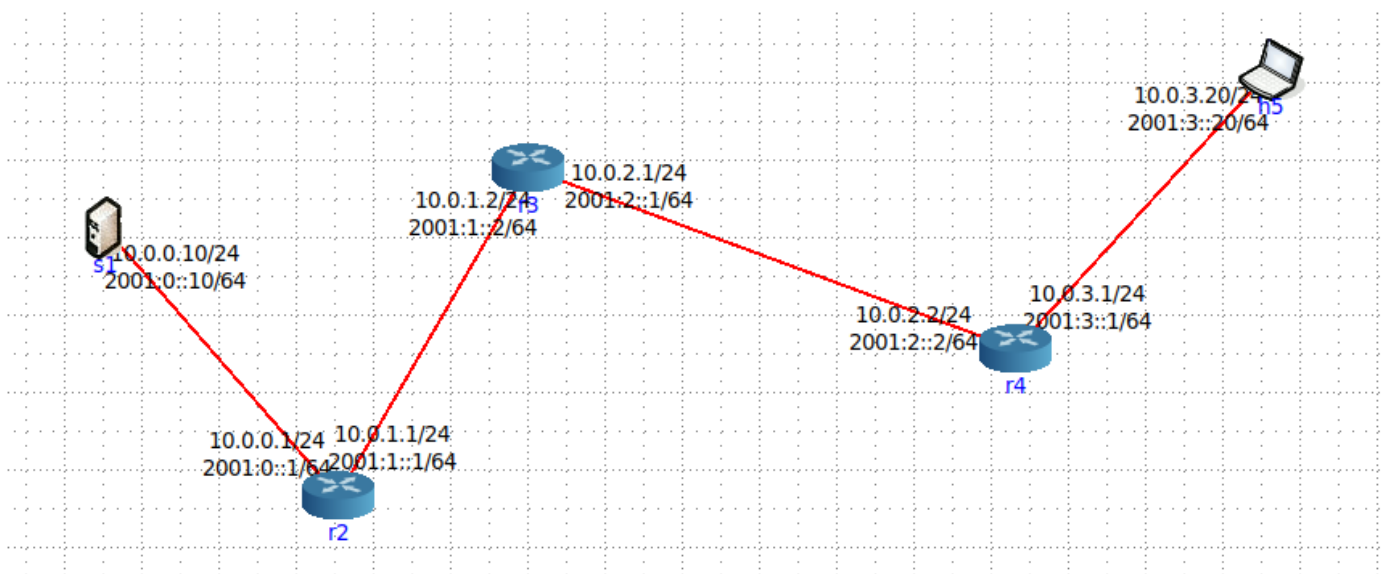
Índice de conteúdos

| | |
|---|-----------|
| 1. Questões e Repostas | 3 |
| 1.1. Parte 1 – Datagramas IP e Fragmentação | 3 |
| 1.2. Parte 2 – Endereço e Encaminhamento IP..... | 11 |
| 2. Conclusões..... | 18 |

1. Questões e Respostas

1.1. Parte 1 – Datagramas IP e Fragmentação

2.1) Prepare uma topologia no CORE para verificar o comportamento do traceroute. Ligue um host (servidor) s1 a um router r2; o router r2 a um router r3, o router r3 a um router r4, que por sua vez, se liga a um host (pc) h5. (Note que pode não existir conectividade IP imediata entre s1 e h5 até que o routing estabilize). Ajuste o nome dos equipamentos atribuídos por defeito para a topologia do enunciado.



a) Active o wireshark ou o tcpdump no pc s1. Numa shell de s1, execute o comando traceroute -I para o endereço IP do host h5.

```
root@s1: /tmp/pycore.38865/s1.conf
root@s1:/tmp/pycore.38865/s1.conf# traceroute -I 10.0.3.20
traceroute to 10.0.3.20 (10.0.3.20), 30 hops max, 60 byte packets
 1  10.0.0.1 (10.0.0.1)  0.074 ms  0.014 ms  0.011 ms
 2  10.0.1.2 (10.0.1.2)  0.077 ms  0.020 ms  0.016 ms
 3  10.0.2.2 (10.0.2.2)  0.029 ms  0.021 ms  0.020 ms
 4  10.0.3.20 (10.0.3.20)  0.040 ms  0.026 ms  0.024 ms
root@s1:/tmp/pycore.38865/s1.conf#
```

b) Registe e analise o tráfego ICMP enviado por s1 e o tráfego ICMP recebido como resposta. Comente os resultados face ao comportamento esperado.

| No. | Time | Source | Destination | Protocol | Length | Info |
|-----|--------------|-----------|-------------|----------|--------|---|
| 9 | 39.959562692 | 10.0.0.10 | 10.0.3.20 | ICMP | 74 | Echo (ping) request id=0x0047, seq=1/256, ttl=1 (no response found!) |
| 10 | 39.959618040 | 10.0.0.1 | 10.0.0.10 | ICMP | 102 | Time-to-live exceeded (Time to live exceeded in transit) |
| 11 | 39.959632376 | 10.0.0.10 | 10.0.3.20 | ICMP | 74 | Echo (ping) request id=0x0047, seq=2/512, ttl=1 (no response found!) |
| 12 | 39.959641315 | 10.0.0.1 | 10.0.0.10 | ICMP | 102 | Time-to-live exceeded (Time to live exceeded in transit) |
| 13 | 39.959648154 | 10.0.0.10 | 10.0.3.20 | ICMP | 74 | Echo (ping) request id=0x0047, seq=3/768, ttl=1 (no response found!) |
| 14 | 39.959655068 | 10.0.0.1 | 10.0.0.10 | ICMP | 102 | Time-to-live exceeded (Time to live exceeded in transit) |
| 15 | 39.959662657 | 10.0.0.10 | 10.0.3.20 | ICMP | 74 | Echo (ping) request id=0x0047, seq=4/1024, ttl=2 (no response found!) |
| 16 | 39.959735201 | 10.0.1.2 | 10.0.0.10 | ICMP | 102 | Time-to-live exceeded (Time to live exceeded in transit) |
| 17 | 39.959748868 | 10.0.0.10 | 10.0.3.20 | ICMP | 74 | Echo (ping) request id=0x0047, seq=5/1280, ttl=2 (no response found!) |
| 18 | 39.959763335 | 10.0.1.2 | 10.0.0.10 | ICMP | 102 | Time-to-live exceeded (Time to live exceeded in transit) |
| 19 | 39.959770145 | 10.0.0.10 | 10.0.3.20 | ICMP | 74 | Echo (ping) request id=0x0047, seq=6/1536, ttl=2 (no response found!) |
| 20 | 39.959782144 | 10.0.1.2 | 10.0.0.10 | ICMP | 102 | Time-to-live exceeded (Time to live exceeded in transit) |
| 21 | 39.959789991 | 10.0.0.10 | 10.0.3.20 | ICMP | 74 | Echo (ping) request id=0x0047, seq=7/1792, ttl=3 (no response found!) |
| 22 | 39.959810078 | 10.0.2.2 | 10.0.0.10 | ICMP | 102 | Time-to-live exceeded (Time to live exceeded in transit) |
| 23 | 39.959823213 | 10.0.0.10 | 10.0.3.20 | ICMP | 74 | Echo (ping) request id=0x0047, seq=8/2048, ttl=3 (no response found!) |
| 24 | 39.959840135 | 10.0.2.2 | 10.0.0.10 | ICMP | 102 | Time-to-live exceeded (Time to live exceeded in transit) |
| 25 | 39.959846514 | 10.0.0.10 | 10.0.3.20 | ICMP | 74 | Echo (ping) request id=0x0047, seq=9/2304, ttl=3 (no response found!) |
| 26 | 39.959863197 | 10.0.2.2 | 10.0.0.10 | ICMP | 102 | Time-to-live exceeded (Time to live exceeded in transit) |
| 27 | 39.959870702 | 10.0.0.10 | 10.0.3.20 | ICMP | 74 | Echo (ping) request id=0x0047, seq=10/2560, ttl=4 (reply in 28) |
| 28 | 39.959906677 | 10.0.3.20 | 10.0.0.10 | ICMP | 74 | Echo (ping) reply id=0x0047, seq=10/2560, ttl=61 (request in 27) |
| 29 | 39.959915384 | 10.0.0.10 | 10.0.3.20 | ICMP | 74 | Echo (ping) request id=0x0047, seq=11/2816, ttl=4 (reply in 30) |
| 30 | 39.959937687 | 10.0.3.20 | 10.0.0.10 | ICMP | 74 | Echo (ping) reply id=0x0047, seq=11/2816, ttl=61 (request in 29) |
| 31 | 39.959944389 | 10.0.0.10 | 10.0.3.20 | ICMP | 74 | Echo (ping) request id=0x0047, seq=12/3072, ttl=4 (reply in 32) |
| 32 | 39.959965637 | 10.0.3.20 | 10.0.0.10 | ICMP | 74 | Echo (ping) reply id=0x0047, seq=12/3072, ttl=61 (request in 31) |
| 33 | 39.959973437 | 10.0.0.10 | 10.0.3.20 | ICMP | 74 | Echo (ping) request id=0x0047, seq=13/3328, ttl=5 (reply in 34) |
| 34 | 39.959994732 | 10.0.3.20 | 10.0.0.10 | ICMP | 74 | Echo (ping) reply id=0x0047, seq=13/3328, ttl=61 (request in 33) |
| 35 | 39.960001210 | 10.0.0.10 | 10.0.3.20 | ICMP | 74 | Echo (ping) request id=0x0047, seq=14/3584, ttl=5 (reply in 36) |
| 36 | 39.960022445 | 10.0.3.20 | 10.0.0.10 | ICMP | 74 | Echo (ping) reply id=0x0047, seq=14/3584, ttl=61 (request in 35) |
| 37 | 39.960028867 | 10.0.0.10 | 10.0.3.20 | ICMP | 74 | Echo (ping) request id=0x0047, seq=15/3840, ttl=5 (reply in 38) |
| 38 | 39.960050229 | 10.0.3.20 | 10.0.0.10 | ICMP | 74 | Echo (ping) reply id=0x0047, seq=15/3840, ttl=61 (request in 37) |
| 39 | 39.960057756 | 10.0.0.10 | 10.0.3.20 | ICMP | 74 | Echo (ping) request id=0x0047, seq=16/4096, ttl=6 (reply in 40) |
| 40 | 39.960080910 | 10.0.3.20 | 10.0.0.10 | ICMP | 74 | Echo (ping) reply id=0x0047, seq=16/4096, ttl=61 (request in 39) |

Podemos observar que, quando TTL < 4, não há resposta e ocorre time out. Obtemos a primeira resposta quando TTL = 4.

c) Qual deve ser o valor inicial mínimo do campo TTL para alcançar o destino h5? Verifique na prática que a sua resposta está correta.

Teoricamente, o valor inicial mínimo do campo TTL para alcançar o destino h5 deve ser 4. O datagrama efetua 4 saltos até chegar ao h5: - do s1 ao r2, do r2 ao r3, do r3 ao r4, do r4 ao h5.

Verificamos na prática que o TTL mínimo é 4, pois, como constatamos na alínea b), quando o TTL é inferior a 4 não se obtém resposta, mas a partir de 4 sim.

d) Qual o valor médio do tempo de ida-e-volta (Round-Trip Time) obtido?

| No. | Time | Source | Destination | Protocol | Length | Info |
|-----|--------------|-----------|-------------|----------|--------|---|
| 10 | 39.959618040 | 10.0.0.1 | 10.0.0.10 | ICMP | 102 | Time-to-live exceeded (Time to live exceeded in transit) |
| 12 | 39.959641315 | 10.0.0.1 | 10.0.0.10 | ICMP | 102 | Time-to-live exceeded (Time to live exceeded in transit) |
| 14 | 39.959655068 | 10.0.0.1 | 10.0.0.10 | ICMP | 102 | Time-to-live exceeded (Time to live exceeded in transit) |
| 9 | 39.959562692 | 10.0.0.10 | 10.0.3.20 | ICMP | 74 | Echo (ping) request id=0x0047, seq=1/256, ttl=1 (no response found!) |
| 11 | 39.959632376 | 10.0.0.10 | 10.0.3.20 | ICMP | 74 | Echo (ping) request id=0x0047, seq=2/512, ttl=1 (no response found!) |
| 13 | 39.959648154 | 10.0.0.10 | 10.0.3.20 | ICMP | 74 | Echo (ping) request id=0x0047, seq=3/768, ttl=1 (no response found!) |
| 15 | 39.959662657 | 10.0.0.10 | 10.0.3.20 | ICMP | 74 | Echo (ping) request id=0x0047, seq=4/1024, ttl=2 (no response found!) |
| 17 | 39.959748868 | 10.0.0.10 | 10.0.3.20 | ICMP | 74 | Echo (ping) request id=0x0047, seq=5/1280, ttl=2 (no response found!) |
| 19 | 39.959770145 | 10.0.0.10 | 10.0.3.20 | ICMP | 74 | Echo (ping) request id=0x0047, seq=6/1536, ttl=2 (no response found!) |
| 21 | 39.959789991 | 10.0.0.10 | 10.0.3.20 | ICMP | 74 | Echo (ping) request id=0x0047, seq=7/1792, ttl=3 (no response found!) |
| 23 | 39.959823213 | 10.0.0.10 | 10.0.3.20 | ICMP | 74 | Echo (ping) request id=0x0047, seq=8/2048, ttl=3 (no response found!) |
| 25 | 39.959846514 | 10.0.0.10 | 10.0.3.20 | ICMP | 74 | Echo (ping) request id=0x0047, seq=9/2304, ttl=3 (no response found!) |
| 27 | 39.959870702 | 10.0.0.10 | 10.0.3.20 | ICMP | 74 | Echo (ping) request id=0x0047, seq=10/2560, ttl=4 (reply in 28) |
| 29 | 39.959915384 | 10.0.0.10 | 10.0.3.20 | ICMP | 74 | Echo (ping) request id=0x0047, seq=11/2816, ttl=4 (reply in 30) |
| 31 | 39.959944389 | 10.0.0.10 | 10.0.3.20 | ICMP | 74 | Echo (ping) request id=0x0047, seq=12/3072, ttl=4 (reply in 32) |
| 33 | 39.959973437 | 10.0.0.10 | 10.0.3.20 | ICMP | 74 | Echo (ping) request id=0x0047, seq=13/3328, ttl=5 (reply in 34) |
| 35 | 39.960001210 | 10.0.0.10 | 10.0.3.20 | ICMP | 74 | Echo (ping) request id=0x0047, seq=14/3584, ttl=5 (reply in 36) |
| 37 | 39.960028867 | 10.0.0.10 | 10.0.3.20 | ICMP | 74 | Echo (ping) request id=0x0047, seq=15/3840, ttl=5 (reply in 38) |
| 39 | 39.960057756 | 10.0.0.10 | 10.0.3.20 | ICMP | 74 | Echo (ping) request id=0x0047, seq=16/4096, ttl=6 (reply in 40) |
| 16 | 39.959735201 | 10.0.1.2 | 10.0.0.10 | ICMP | 102 | Time-to-live exceeded (Time to live exceeded in transit) |
| 18 | 39.959763335 | 10.0.1.2 | 10.0.0.10 | ICMP | 102 | Time-to-live exceeded (Time to live exceeded in transit) |
| 20 | 39.959782144 | 10.0.1.2 | 10.0.0.10 | ICMP | 102 | Time-to-live exceeded (Time to live exceeded in transit) |
| 22 | 39.959810078 | 10.0.2.2 | 10.0.0.10 | ICMP | 102 | Time-to-live exceeded (Time to live exceeded in transit) |
| 24 | 39.959840135 | 10.0.2.2 | 10.0.0.10 | ICMP | 102 | Time-to-live exceeded (Time to live exceeded in transit) |
| 26 | 39.959863197 | 10.0.2.2 | 10.0.0.10 | ICMP | 102 | Time-to-live exceeded (Time to live exceeded in transit) |
| 28 | 39.959906677 | 10.0.3.20 | 10.0.0.10 | ICMP | 74 | Echo (ping) reply id=0x0047, seq=10/2560, ttl=61 (request in 27) |
| 30 | 39.959937687 | 10.0.3.20 | 10.0.0.10 | ICMP | 74 | Echo (ping) reply id=0x0047, seq=11/2816, ttl=61 (request in 29) |
| 32 | 39.959965637 | 10.0.3.20 | 10.0.0.10 | ICMP | 74 | Echo (ping) reply id=0x0047, seq=12/3072, ttl=61 (request in 31) |
| 34 | 39.959994732 | 10.0.3.20 | 10.0.0.10 | ICMP | 74 | Echo (ping) reply id=0x0047, seq=13/3328, ttl=61 (request in 33) |
| 36 | 39.960022445 | 10.0.3.20 | 10.0.0.10 | ICMP | 74 | Echo (ping) reply id=0x0047, seq=14/3584, ttl=61 (request in 35) |
| 38 | 39.960050229 | 10.0.3.20 | 10.0.0.10 | ICMP | 74 | Echo (ping) reply id=0x0047, seq=15/3840, ttl=61 (request in 37) |
| 40 | 39.960080910 | 10.0.3.20 | 10.0.0.10 | ICMP | 74 | Echo (ping) reply id=0x0047, seq=16/4096, ttl=61 (request in 39) |

Para calcular o tempo médio de ida/volta, consideramos apenas as mensagens com um TTL ≥ 4 , pois apenas essas chegam ao destino. Como tal, há 7 mensagens de ida e 7 de volta (assinaladas na imagem).

- Tempo médio de ida = 39.959973437 ms (o tempo varia muito pouco de uma mensagem para outra, aumentado sistematicamente, por isso consideramos o tempo da 4ª mensagem - a do meio - como o tempo médio);
- Tempo médio de volta = 39.959994732 ms (mesma lógica);
- RTT = 39.959973437 (tempo média de ida) + 39.959994732 (tempo médio de chegada) = 79.919968169 ms (aprox. 79.92 ms).

2.2) Usando o wireshark capture o tráfego gerado pelo traceroute para os seguintes tamanhos de pacote: (i) sem especificar, i.e., usando o tamanho por defeito; e (ii) 42XX bytes, em que XX é o seu número de grupo. Utilize como máquina destino o host marco.uminho.pt. Pare a captura.

Com base no tráfego capturado, identifique os pedidos ICMP Echo Request e o conjunto de mensagens devolvidas em resposta a esses pedidos.

Selecione a primeira mensagem ICMP capturada (referente a (i) tamanho por defeito) e centre a análise do nível protocolar IP (expanda o tab correspondente na janela de detalhe do wireshark). Através da análise do cabeçalho IP diga:

| icmp | | | | | | |
|------|----------|-----------------|-----------------|----------|--------|---|
| No. | Time | Source | Destination | Protocol | Length | Info |
| 76 | 7.936126 | 192.168.100.196 | 193.136.9.240 | ICMP | 70 | Echo (ping) request id=0x0001, seq=3/768, ttl=255 (reply in 77) |
| 77 | 7.936941 | 193.136.9.240 | 192.168.100.196 | ICMP | 70 | Echo (ping) reply id=0x0001, seq=3/768, ttl=62 (request in 76) |
| 78 | 7.969676 | 192.168.100.196 | 193.136.9.240 | ICMP | 70 | Echo (ping) request id=0x0001, seq=4/1024, ttl=1 (no response found!) |
| 79 | 7.970456 | 192.168.100.254 | 192.168.100.196 | ICMP | 98 | Time-to-live exceeded (Time to live exceeded in transit) |
| 80 | 8.008195 | 192.168.100.196 | 193.136.9.240 | ICMP | 70 | Echo (ping) request id=0x0001, seq=5/1280, ttl=2 (no response found!) |
| 81 | 8.008971 | 193.136.19.254 | 192.168.100.196 | ICMP | 70 | Time-to-live exceeded (Time to live exceeded in transit) |
| 82 | 8.047151 | 192.168.100.196 | 193.136.9.240 | ICMP | 70 | Echo (ping) request id=0x0001, seq=6/1536, ttl=3 (reply in 83) |
| 83 | 8.048025 | 193.136.9.240 | 192.168.100.196 | ICMP | 70 | Echo (ping) reply id=0x0001, seq=6/1536, ttl=62 (request in 82) |

| |
|---|
| > Frame 78: 70 bytes on wire (560 bits), 70 bytes captured (560 bits) on interface 0 |
| > Ethernet II, Src: HewlettP_93:4b:2b (f4:30:b9:93:4b:2b), Dst: Vmware_d2:19:f0 (00:0c:29:d2:19:f0) |
| > Internet Protocol Version 4, Src: 192.168.100.196, 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: 56 |
| Identification: 0x48f9 (18681) |
| > Flags: 0x0000 |
| 0... .. = Reserved bit: Not set |
| .0.. .. = Don't fragment: Not set |
| ..0. = More fragments: Not set |
| ...0 0000 0000 0000 = Fragment offset: 0 |
| > Time to live: 1 |
| Protocol: ICMP (1) |
| Header checksum: 0x0000 [validation disabled] |
| [Header checksum status: Unverified] |
| Source: 192.168.100.196 |
| Destination: 193.136.9.240 |
| > Internet Control Message Protocol |

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

Para ver o IP da interface ativa do computador, ordenamos os packets por source e, com o filtro icmp aplicado, vemos o source IP do primeiro packet enviado (pelo nosso computador). Neste caso, é 192.168.100.196.

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

O valor do campo protocolo é ICMP (1).

O que identifica?

Identifica o pacote enviado, do tipo ICMP.

c) Quantos bytes tem o cabeçalho IP(v4)?

Cabeçalho IP: 20 bytes (como se pode observar no print)

Quantos bytes tem o campo de dados (payload) do datagrama?

Campo de dados (payload): 36 bytes

Como se calcula o tamanho do payload?

Subtraímos o tamanho do cabeçalho IP - 20 - ao comprimento total (total length) - 56.

d) O datagrama IP foi fragmentado? Justifique.

Não. As flags no print: "More fragments: Not Set" e "Fragment offset: 0", logo o datagrama não foi fragmentado.

e) Ordene os pacotes capturados de acordo com o endereço IP fonte (e.g., selecionando o cabeçalho da coluna Source), 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.

| icmp | | | | | | |
|------|----------|-----------------|-----------------|----------|--------|---|
| No. | Time | Source | Destination | Protocol | Length | Info |
| 76 | 7.936126 | 192.168.100.196 | 193.136.9.240 | ICMP | 70 | Echo (ping) request id=0x0001, seq=3/768, ttl=255 (reply in 77) |
| 78 | 7.969676 | 192.168.100.196 | 193.136.9.240 | ICMP | 70 | Echo (ping) request id=0x0001, seq=4/1024, ttl=1 (no response found!) |
| 80 | 8.008195 | 192.168.100.196 | 193.136.9.240 | ICMP | 70 | Echo (ping) request id=0x0001, seq=5/1280, ttl=2 (no response found!) |
| 82 | 8.047151 | 192.168.100.196 | 193.136.9.240 | ICMP | 70 | Echo (ping) request id=0x0001, seq=6/1536, ttl=3 (reply in 83) |
| 79 | 7.970456 | 192.168.100.254 | 192.168.100.196 | ICMP | 98 | Time-to-live exceeded (Time to live exceeded in transit) |
| 81 | 8.008971 | 193.136.19.254 | 192.168.100.196 | ICMP | 70 | Time-to-live exceeded (Time to live exceeded in transit) |
| 77 | 7.936941 | 193.136.9.240 | 192.168.100.196 | ICMP | 70 | Echo (ping) reply id=0x0001, seq=3/768, ttl=62 (request in 76) |
| 83 | 8.048025 | 193.136.9.240 | 192.168.100.196 | ICMP | 70 | Echo (ping) reply id=0x0001, seq=6/1536, ttl=62 (request in 82) |

> Frame 78: 70 bytes on wire (560 bits), 70 bytes captured (560 bits) on interface 0

> Ethernet II, Src: HewlettP_93:4b:2b (f4:30:b9:93:4b:2b), Dst: Vmware_d2:19:f0 (00:0c:29:d2:19:f0)

> Internet Protocol Version 4, Src: 192.168.100.196, 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: 56

Identification: 0x48f9 (18681)> Flags: 0x0000

0... .. = Reserved bit: Not set

.0... .. = Don't fragment: Not set

..0. = More fragments: Not set

...0 0000 0000 0000 = Fragment offset: 0

> Time to live: 1

Protocol: ICMP (1)

Header checksum: 0x0000 [validation disabled]

[Header checksum status: Unverified]

Source: 192.168.100.196

Destination: 193.136.9.240

> Internet Control Message Protocol

| icmp | | | | | | |
|------|----------|-----------------|-----------------|----------|--------|---|
| No. | Time | Source | Destination | Protocol | Length | Info |
| 76 | 7.936126 | 192.168.100.196 | 193.136.9.240 | ICMP | 70 | Echo (ping) request id=0x0001, seq=3/768, ttl=255 (reply in 77) |
| 78 | 7.969676 | 192.168.100.196 | 193.136.9.240 | ICMP | 70 | Echo (ping) request id=0x0001, seq=4/1024, ttl=1 (no response found!) |
| 80 | 8.008195 | 192.168.100.196 | 193.136.9.240 | ICMP | 70 | Echo (ping) request id=0x0001, seq=5/1280, ttl=2 (no response found!) |
| 82 | 8.047151 | 192.168.100.196 | 193.136.9.240 | ICMP | 70 | Echo (ping) request id=0x0001, seq=6/1536, ttl=3 (reply in 83) |
| 79 | 7.970456 | 192.168.100.254 | 192.168.100.196 | ICMP | 98 | Time-to-live exceeded (Time to live exceeded in transit) |
| 81 | 8.008971 | 193.136.19.254 | 192.168.100.196 | ICMP | 70 | Time-to-live exceeded (Time to live exceeded in transit) |
| 77 | 7.936941 | 193.136.9.240 | 192.168.100.196 | ICMP | 70 | Echo (ping) reply id=0x0001, seq=3/768, ttl=62 (request in 76) |
| 83 | 8.048025 | 193.136.9.240 | 192.168.100.196 | ICMP | 70 | Echo (ping) reply id=0x0001, seq=6/1536, ttl=62 (request in 82) |

| | |
|---|---|
| > | Frame 80: 70 bytes on wire (560 bits), 70 bytes captured (560 bits) on interface 0 |
| > | Ethernet II, Src: HewlettP_93:4b:2b (f4:30:b9:93:4b:2b), Dst: Vmware_d2:19:f0 (00:0c:29:d2:19:f0) |
| > | Internet Protocol Version 4, Src: 192.168.100.196, 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: 56 |
| > | Identification: 0x48fa (18682) |
| > | Flags: 0x0000 |
| > | 0... .. = Reserved bit: Not set |
| > | .0.. .. = Don't fragment: Not set |
| > | ..0. = More fragments: Not set |
| > | ...0 0000 0000 0000 = Fragment offset: 0 |
| > | Time to live: 2 |
| > | Protocol: ICMP (1) |
| > | Header checksum: 0x0000 [validation disabled] |
| > | [Header checksum status: Unverified] |
| > | Source: 192.168.100.196 |
| > | Destination: 193.136.9.240 |
| > | Internet Control Message Protocol |

| icmp | | | | | | |
|------|----------|-----------------|-----------------|----------|--------|---|
| No. | Time | Source | Destination | Protocol | Length | Info |
| 76 | 7.936126 | 192.168.100.196 | 193.136.9.240 | ICMP | 70 | Echo (ping) request id=0x0001, seq=3/768, ttl=255 (reply in 77) |
| 78 | 7.969676 | 192.168.100.196 | 193.136.9.240 | ICMP | 70 | Echo (ping) request id=0x0001, seq=4/1024, ttl=1 (no response found!) |
| 80 | 8.008195 | 192.168.100.196 | 193.136.9.240 | ICMP | 70 | Echo (ping) request id=0x0001, seq=5/1280, ttl=2 (no response found!) |
| 82 | 8.047151 | 192.168.100.196 | 193.136.9.240 | ICMP | 70 | Echo (ping) request id=0x0001, seq=6/1536, ttl=3 (reply in 83) |
| 79 | 7.970456 | 192.168.100.254 | 192.168.100.196 | ICMP | 98 | Time-to-live exceeded (Time to live exceeded in transit) |
| 81 | 8.008971 | 193.136.19.254 | 192.168.100.196 | ICMP | 70 | Time-to-live exceeded (Time to live exceeded in transit) |
| 77 | 7.936941 | 193.136.9.240 | 192.168.100.196 | ICMP | 70 | Echo (ping) reply id=0x0001, seq=3/768, ttl=62 (request in 76) |
| 83 | 8.048025 | 193.136.9.240 | 192.168.100.196 | ICMP | 70 | Echo (ping) reply id=0x0001, seq=6/1536, ttl=62 (request in 82) |

| | |
|---|---|
| > | Frame 82: 70 bytes on wire (560 bits), 70 bytes captured (560 bits) on interface 0 |
| > | Ethernet II, Src: HewlettP_93:4b:2b (f4:30:b9:93:4b:2b), Dst: Vmware_d2:19:f0 (00:0c:29:d2:19:f0) |
| > | Internet Protocol Version 4, Src: 192.168.100.196, 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: 56 |
| > | Identification: 0x48fb (18683) |
| > | Flags: 0x0000 |
| > | 0... .. = Reserved bit: Not set |
| > | .0.. .. = Don't fragment: Not set |
| > | ..0. = More fragments: Not set |
| > | ...0 0000 0000 0000 = Fragment offset: 0 |
| > | Time to live: 3 |
| > | Protocol: ICMP (1) |
| > | Header checksum: 0x0000 [validation disabled] |
| > | [Header checksum status: Unverified] |
| > | Source: 192.168.100.196 |
| > | Destination: 193.136.9.240 |
| > | Internet Control Message Protocol |

O nosso computador envia 4 mensagens ICMP diferentes. A primeira mensagem é um ping enviado pelo pingplotter para testar a conectividade e não deve, portanto, ser considerada juntamente com as outras. Entre as restantes, variam a identificação e o time to live das mensagens.

f) Observa algum padrão nos valores do campo de Identificação do datagrama IP e TTL?

Sim, nas mensagens ICMP enviadas pelo nosso computador estes valores são incrementados com cada datagrama.

g) Ordene o tráfego capturado por endereço destino e encontre a série de respostas ICMP TTL *exceeded* enviadas ao seu computador.

| icmp | | | | | | |
|------|----------|-----------------|-----------------|----------|--------|---|
| No. | Time | Source | Destination | Protocol | Length | Info |
| 77 | 7.936941 | 193.136.9.240 | 192.168.100.196 | ICMP | 70 | Echo (ping) reply id=0x0001, seq=3/768, ttl=62 (request in 76) |
| 79 | 7.970456 | 192.168.100.254 | 192.168.100.196 | ICMP | 98 | Time-to-live exceeded (Time to live exceeded in transit) |
| 81 | 8.008971 | 193.136.19.254 | 192.168.100.196 | ICMP | 70 | Time-to-live exceeded (Time to live exceeded in transit) |
| 83 | 8.048025 | 193.136.9.240 | 192.168.100.196 | ICMP | 70 | Echo (ping) reply id=0x0001, seq=6/1536, ttl=62 (request in 82) |
| 76 | 7.936126 | 192.168.100.196 | 193.136.9.240 | ICMP | 70 | Echo (ping) request id=0x0001, seq=3/768, ttl=255 (reply in 77) |
| 78 | 7.969676 | 192.168.100.196 | 193.136.9.240 | ICMP | 70 | Echo (ping) request id=0x0001, seq=4/1024, ttl=1 (no response found!) |
| 80 | 8.008195 | 192.168.100.196 | 193.136.9.240 | ICMP | 70 | Echo (ping) request id=0x0001, seq=5/1280, ttl=2 (no response found!) |
| 82 | 8.047151 | 192.168.100.196 | 193.136.9.240 | ICMP | 70 | Echo (ping) request id=0x0001, seq=6/1536, ttl=3 (reply in 83) |

| | |
|---|---|
| > | Frame 79: 98 bytes on wire (784 bits), 98 bytes captured (784 bits) on interface 0 |
| > | Ethernet II, Src: Vmware_d2:19:f0 (00:0c:29:d2:19:f0), Dst: HewlettP_93:4b:2b (f4:30:b9:93:4b:2b) |
| > | Internet Protocol Version 4, Src: 192.168.100.254, Dst: 192.168.100.196 |
| > | 0100 = Version: 4 |
| > | 0101 = Header Length: 20 bytes (5) |
| > | Differentiated Services Field: 0xc0 (DSCP: CS6, ECN: Not-ECT) |
| > | Total Length: 84 |
| > | Identification: 0x332f (13103) |
| > | Flags: 0x0000 |
| > | 0... = Reserved bit: Not set |
| > | .0.. = Don't fragment: Not set |
| > | ..0. = More fragments: Not set |
| > | ...0 0000 0000 0000 = Fragment offset: 0 |
| > | Time to live: 64 |
| > | Protocol: ICMP (1) |
| > | Header checksum: 0xfba6 [validation disabled] |
| > | [Header checksum status: Unverified] |
| > | Source: 192.168.100.254 |
| > | Destination: 192.168.100.196 |
| > | Internet Control Message Protocol |

| icmp | | | | | | |
|------|----------|-----------------|-----------------|----------|--------|---|
| No. | Time | Source | Destination | Protocol | Length | Info |
| 77 | 7.936941 | 193.136.9.240 | 192.168.100.196 | ICMP | 70 | Echo (ping) reply id=0x0001, seq=3/768, ttl=62 (request in 76) |
| 79 | 7.970456 | 192.168.100.254 | 192.168.100.196 | ICMP | 98 | Time-to-live exceeded (Time to live exceeded in transit) |
| 81 | 8.008971 | 193.136.19.254 | 192.168.100.196 | ICMP | 70 | Time-to-live exceeded (Time to live exceeded in transit) |
| 83 | 8.048025 | 193.136.9.240 | 192.168.100.196 | ICMP | 70 | Echo (ping) reply id=0x0001, seq=6/1536, ttl=62 (request in 82) |
| 76 | 7.936126 | 192.168.100.196 | 193.136.9.240 | ICMP | 70 | Echo (ping) request id=0x0001, seq=3/768, ttl=255 (reply in 77) |
| 78 | 7.969676 | 192.168.100.196 | 193.136.9.240 | ICMP | 70 | Echo (ping) request id=0x0001, seq=4/1024, ttl=1 (no response found!) |
| 80 | 8.008195 | 192.168.100.196 | 193.136.9.240 | ICMP | 70 | Echo (ping) request id=0x0001, seq=5/1280, ttl=2 (no response found!) |
| 82 | 8.047151 | 192.168.100.196 | 193.136.9.240 | ICMP | 70 | Echo (ping) request id=0x0001, seq=6/1536, ttl=3 (reply in 83) |

| | |
|---|---|
| > | Frame 81: 70 bytes on wire (560 bits), 70 bytes captured (560 bits) on interface 0 |
| > | Ethernet II, Src: Vmware_d2:19:f0 (00:0c:29:d2:19:f0), Dst: HewlettP_93:4b:2b (f4:30:b9:93:4b:2b) |
| > | Internet Protocol Version 4, Src: 193.136.19.254, Dst: 192.168.100.196 |
| > | 0100 = Version: 4 |
| > | 0101 = Header Length: 20 bytes (5) |
| > | Differentiated Services Field: 0xc0 (DSCP: CS6, ECN: Not-ECT) |
| > | Total Length: 56 |
| > | Identification: 0xcc08 (52232) |
| > | Flags: 0x0000 |
| > | 0... = Reserved bit: Not set |
| > | .0.. = Don't fragment: Not set |
| > | ..0. = More fragments: Not set |
| > | ...0 0000 0000 0000 = Fragment offset: 0 |
| > | Time to live: 254 |
| > | Protocol: ICMP (1) |
| > | Header checksum: 0xf508 [validation disabled] |
| > | [Header checksum status: Unverified] |
| > | Source: 193.136.19.254 |
| > | Destination: 192.168.100.196 |
| > | Internet Control Message Protocol |

Qual é o valor do campo TTL?

Há 2 respostas ICMP TTL exceeded enviadas ao computador.

- TTL da primeira resposta: 64 ms
- TTL da segunda resposta: 254 ms

Esse valor permanece constante para todas as mensagens de resposta ICMP TTL exceeded enviados ao seu host? Porquê?

Como podemos observar, o valor não permanece constante para todas as respostas ICMP TTL exceeded enviadas ao computador. Concluimos que é mais eficaz usar TTL's elevados para garantir

que a mensagem chega ao destino, sendo que após a primeira tentativa falhada, o programa atribui um TTL maior à seguinte para tentar garantir que chega ao destino.

2.3) Pretende-se agora analisar a fragmentação de pacotes IP. Reponha a ordem do tráfego capturado usando a coluna do tempo de captura. Observe o tráfego depois do tamanho de pacote ter sido definido para 42XX bytes.

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

| No. | Time | Source | Destination | Protocol | Length | Info |
|-----|----------|-----------------|-----------------|----------|--------|--|
| 3 | 0.483433 | 192.168.100.196 | 159.153.64.176 | TCP | 66 | 62001 → 44325 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1 |
| 13 | 2.040863 | 192.168.100.196 | 193.136.9.240 | ICMP | 1514 | Echo (ping) request id=0x0001, seq=7/1792, ttl=255 (reply in 20) |
| 14 | 2.040863 | 192.168.100.196 | 193.136.9.240 | IPv4 | 1514 | Fragmented IP protocol (proto=ICMP 1, off=1480, ID=1159) |
| 15 | 2.040864 | 192.168.100.196 | 193.136.9.240 | IPv4 | 1260 | Fragmented IP protocol (proto=ICMP 1, off=2960, ID=1159) |
| 16 | 2.041180 | 192.168.100.196 | 192.168.100.254 | DNS | 75 | Standard query 0xd7c8 A marco.uminho.pt |
| 17 | 2.041332 | 192.168.100.196 | 192.168.100.254 | DNS | 75 | Standard query 0xa37d AAAA marco.uminho.pt |
| 18 | 2.042230 | 192.168.100.254 | 192.168.100.196 | DNS | 319 | Standard query response 0xd7c8 A marco.uminho.pt A 193.136.9.240 NS dns2.umi |
| 19 | 2.042230 | 192.168.100.254 | 192.168.100.196 | DNS | 129 | Standard query response 0xa37d AAAA marco.uminho.pt SOA dns.uminho.pt |
| 20 | 2.043459 | 193.136.9.240 | 192.168.100.196 | ICMP | 1514 | Echo (ping) reply id=0x0001, seq=7/1792, ttl=62 (request in 13) |
| 21 | 2.044131 | 193.136.9.240 | 192.168.100.196 | IPv4 | 1514 | Fragmented IP protocol (proto=ICMP 1, off=1480, ID=9008) |
| 22 | 2.044131 | 193.136.9.240 | 192.168.100.196 | IPv4 | 1260 | Fragmented IP protocol (proto=ICMP 1, off=2960, ID=9008) |
| 23 | 2.079856 | 192.168.100.196 | 193.136.9.240 | ICMP | 1514 | Echo (ping) request id=0x0001, seq=8/2048, ttl=1 (no response found!) |
| 24 | 2.079856 | 192.168.100.196 | 193.136.9.240 | IPv4 | 1514 | Fragmented IP protocol (proto=ICMP 1, off=1480, ID=115a) |
| 25 | 2.079857 | 192.168.100.196 | 193.136.9.240 | IPv4 | 1260 | Fragmented IP protocol (proto=ICMP 1, off=2960, ID=115a) |
| 26 | 2.080687 | 192.168.100.254 | 192.168.100.196 | ICMP | 590 | Time-to-live exceeded (Time to live exceeded in transit) |
| 27 | 2.111448 | 192.168.100.196 | 34.194.115.224 | TCP | 66 | 61994 → 5222 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1 |
| 28 | 2.120627 | 192.168.100.196 | 193.136.9.240 | ICMP | 1514 | Echo (ping) request id=0x0001, seq=9/2304, ttl=2 (no response found!) |


```

.... 0101 = Header Length: 20 bytes (5)
> Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
Total Length: 1500
Identification: 0x115a (4442)
▼ Flags: 0x2000, More fragments
  0... .. = Reserved bit: Not set
  .0.. .. = Don't fragment: Not set
  ..1. .. = More fragments: Set
...0 0000 0000 0000 = Fragment offset: 0
> Time to live: 1
Protocol: ICMP (1)
Header checksum: 0x0000 [validation disabled]
[Header checksum status: Unverified]

```

É necessário lembrar que a primeira mensagem ICMP enviada é um ping para testar a conectividade, logo esta e o primeiro bloco de fragmentos, no topo do print, não devem ser consideradas.

O tamanho máximo que se pode enviar num só pacote são 1500 bytes (total length). Como se pretende enviar 4006 bytes de informação, não é possível enviar tudo de uma vez, logo é necessário fragmentar o pacote, neste caso em 3 fragmentos.

b) Que informação no cabeçalho indica que o datagrama foi fragmentado?

O bit "More fragments" está set, indicando que o datagrama foi fragmentado e que ainda estão a chegar mais fragmentos.

Que informação no cabeçalho IP indica que se trata do primeiro fragmento?

O offset do fragmento é 0, logo é o primeiro fragmento.

Qual é o tamanho deste datagrama IP?

1500 bytes, como podemos observar através do parâmetro "Total Length".

c) Imprima o segundo fragmento do datagrama IP original. Que informação do cabeçalho IP indica que não se trata do primeiro fragmento?

| ip.addr == 192.168.100.196 | | | | | | |
|----------------------------|----------|-----------------|-----------------|----------|--------|---|
| No. | Time | Source | Destination | Protocol | Length | Info |
| 21 | 2.044131 | 193.136.9.240 | 192.168.100.196 | IPv4 | 1514 | Fragmented IP protocol (proto=ICMP 1, off=1480, ID=9008) |
| 22 | 2.044131 | 193.136.9.240 | 192.168.100.196 | IPv4 | 1260 | Fragmented IP protocol (proto=ICMP 1, off=2960, ID=9008) |
| 23 | 2.079856 | 192.168.100.196 | 193.136.9.240 | ICMP | 1514 | Echo (ping) request id=0x0001, seq=8/2048, ttl=1 (no response found!) |
| 24 | 2.079856 | 192.168.100.196 | 193.136.9.240 | IPv4 | 1514 | Fragmented IP protocol (proto=ICMP 1, off=1480, ID=115a) |
| 25 | 2.079857 | 192.168.100.196 | 193.136.9.240 | IPv4 | 1260 | Fragmented IP protocol (proto=ICMP 1, off=2960, ID=115a) |
| 26 | 2.080687 | 192.168.100.254 | 192.168.100.196 | ICMP | 590 | Time-to-live exceeded (Time to live exceeded in transit) |
| 27 | 2.111448 | 192.168.100.196 | 34.194.115.224 | TCP | 66 | 61994 → 5222 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1 |
| 28 | 2.120627 | 192.168.100.196 | 193.136.9.240 | ICMP | 1514 | Echo (ping) request id=0x0001, seq=9/2304, ttl=2 (no response found!) |
| 29 | 2.120627 | 192.168.100.196 | 193.136.9.240 | IPv4 | 1514 | Fragmented IP protocol (proto=ICMP 1, off=1480, ID=115b) |
| 30 | 2.120628 | 192.168.100.196 | 193.136.9.240 | IPv4 | 1260 | Fragmented IP protocol (proto=ICMP 1, off=2960, ID=115b) |
| 31 | 2.122326 | 193.136.19.254 | 192.168.100.196 | ICMP | 70 | Time-to-live exceeded (Time to live exceeded in transit) |
| 32 | 2.158496 | 192.168.100.196 | 193.136.9.240 | ICMP | 1514 | Echo (ping) request id=0x0001, seq=10/2560, ttl=3 (reply in 35) |
| 33 | 2.158497 | 192.168.100.196 | 193.136.9.240 | IPv4 | 1514 | Fragmented IP protocol (proto=ICMP 1, off=1480, ID=115c) |
| 34 | 2.158498 | 192.168.100.196 | 193.136.9.240 | IPv4 | 1260 | Fragmented IP protocol (proto=ICMP 1, off=2960, ID=115c) |
| 35 | 2.161240 | 193.136.9.240 | 192.168.100.196 | ICMP | 1514 | Echo (ping) reply id=0x0001, seq=10/2560, ttl=62 (request in 32) |
| 36 | 2.161241 | 193.136.9.240 | 192.168.100.196 | IPv4 | 1514 | Fragmented IP protocol (proto=ICMP 1, off=1480, ID=9039) |
| 37 | 2.161242 | 193.136.9.240 | 192.168.100.196 | IPv4 | 1260 | Fragmented IP protocol (proto=ICMP 1, off=2960, ID=9039) |

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

> Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)

Total Length: 1500

Identification: 0x115a (4442)

▼ Flags: 0x20b9, More fragments

0... .. = Reserved bit: Not set

.0.. .. = Don't fragment: Not set

.1. = More fragments: Set

...0 0101 1100 1000 = Fragment offset: 1480

> Time to live: 1

Protocol: ICMP (1)

Header checksum: 0x0000 [validation disabled]

[Header checksum status: Unverified]

O offset do fragmento em questão é 1480, o que significa que é o segundo fragmento.

Há mais fragmentos? O que nos permite afirmar isso?

Sim, o bit de "More fragments" está set, logo este ainda não é o último fragmento.

d) Quantos fragmentos foram criados a partir do datagrama original?

Foram criados 3 fragmentos, sendo a mensagem ICMP o primeiro, mais as duas mensagens a seguir com a descrição "Fragmented IP protocol" na coluna "Info".

Como se detecta o último fragmento correspondente ao datagrama original?

É o fragmento cuja flag "More fragments" não esteja set.

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

Os campos que mudam são o comprimento da mensagem (total length), o bit "More fragments", que indica se é ou não o último fragmento, e o offset do fragmento. A partir destas informações é possível saber o comprimento do datagrama original, somando os dos fragmentos, bem como o conteúdo, se juntarmos a informação dos fragmentos pela ordem em que eles foram enviados, através do offset de cada um.

1.2. Parte 2 – Datagramas IP e Fragmentação

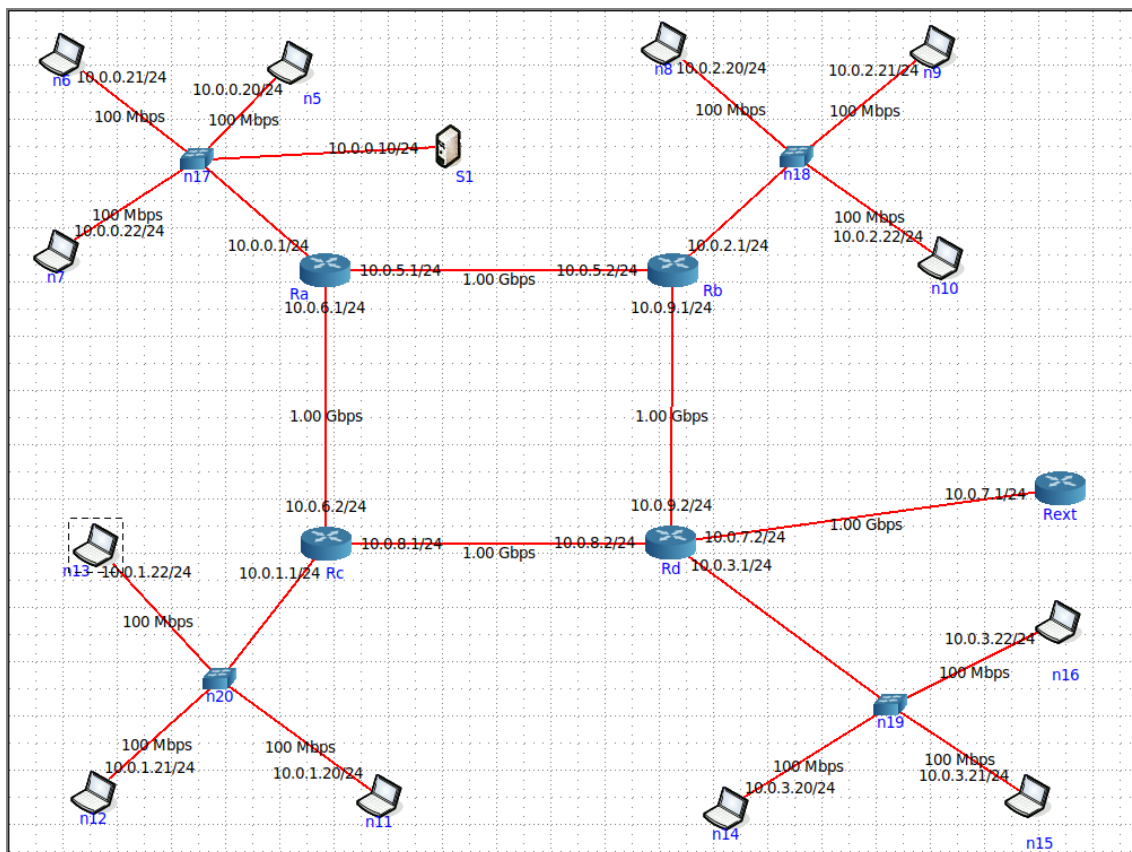
2) Considere que a organização MIEI-RC é constituída por quatro departamentos (A, B, C e D) e cada departamento possui um router de acesso à sua rede local. Estes routers de acesso (Ra, Rb, Rc e Rd) estão interligados entre si por ligações Ethernet a 1 Gbps, formando um anel. Por sua vez, existe um servidor (S1) na rede do departamento A e, pelo menos, três laptops por departamento, interligados ao router respetivo através de um comutador (switch). S1 tem uma ligação a 1 Gbps e os laptops ligações a 100 Mbps. Considere apenas a existência de um comutador por departamento.

A conectividade IP externa da organização é assegurada através de um router de acesso Rext conectado a Rd por uma ligação ponto-a-ponto a 10 Gbps.

Construa uma topologia CORE que reflita a rede local da empresa. Para facilitar a visualização pode ocultar o endereçamento IPv6.

2.1) 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.



Os endereços IP privados encontram-se nas seguintes gamas de valores: 10.0.0.0 - 10.255.255.255, 172.16.0.0 - 172.31.255.255 e 192.168.0.0 - 192.168.255.255. Como podemos observar no print, os nossos endereços são privados, pois estão contidos no primeiro intervalo mencionado.

c) Por que razão não é atribuído um endereço IP aos switches?

Nesta arquitetura, não é atribuído um IP aos switches porque eles são unmanaged, isto é, switches sem opção de configuração, usados para pequenas networks. Neste caso, não há necessidade de ter switches com IP sendo que se trata de uma ligação de apenas 3 dispositivos (PCs) com o router.

d) Usando o comando ping certifique-se que existe conectividade IP entre os laptops dos vários departamentos e o servidor do departamento A (basta certificar-se da conectividade de um laptop por departamento).

The image displays four terminal windows, each showing the output of a ping command from a different host to the IP address 10.0.0.10. The hosts are root@n13, root@n5, root@n16, and root@n8. Each terminal shows the details of four ping packets, including the sequence number, TTL, and round-trip time. The results indicate successful connectivity with 0% packet loss for all hosts.

```
root@n13: /tmp/pycore.42701/n13.conf# ping 10.0.0.10
PING 10.0.0.10 (10.0.0.10) 56(84) bytes of data:
64 bytes from 10.0.0.10: icmp_seq=1 ttl=62 time=0.090 ms
64 bytes from 10.0.0.10: icmp_seq=2 ttl=62 time=0.086 ms
64 bytes from 10.0.0.10: icmp_seq=3 ttl=62 time=0.094 ms
64 bytes from 10.0.0.10: icmp_seq=4 ttl=62 time=0.094 ms
^C
--- 10.0.0.10 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3054ms
rtt min/avg/max/mdev = 0.086/0.091/0.094/0.003 ms
root@n13: /tmp/pycore.42701/n13.conf# c

root@n5: /tmp/pycore.42701/n5.conf# ping 10.0.0.10
PING 10.0.0.10 (10.0.0.10) 56(84) bytes of data:
64 bytes from 10.0.0.10: icmp_seq=1 ttl=64 time=0.123 ms
64 bytes from 10.0.0.10: icmp_seq=2 ttl=64 time=0.059 ms
64 bytes from 10.0.0.10: icmp_seq=3 ttl=64 time=0.061 ms
64 bytes from 10.0.0.10: icmp_seq=4 ttl=64 time=0.063 ms
64 bytes from 10.0.0.10: icmp_seq=5 ttl=64 time=0.061 ms
^Z
[1]+  Stopped                  ping 10.0.0.10
root@n5: /tmp/pycore.42701/n5.conf#

root@n16: /tmp/pycore.42701/n16.conf# ping 10.0.0.10
PING 10.0.0.10 (10.0.0.10) 56(84) bytes of data:
64 bytes from 10.0.0.10: icmp_seq=1 ttl=61 time=0.072 ms
64 bytes from 10.0.0.10: icmp_seq=2 ttl=61 time=0.108 ms
64 bytes from 10.0.0.10: icmp_seq=3 ttl=61 time=0.109 ms
64 bytes from 10.0.0.10: icmp_seq=4 ttl=61 time=0.112 ms
^C
--- 10.0.0.10 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3076ms
rtt min/avg/max/mdev = 0.072/0.100/0.112/0.017 ms
root@n16: /tmp/pycore.42701/n16.conf#

root@n8: /tmp/pycore.42701/n8.conf# ping 10.0.0.10
PING 10.0.0.10 (10.0.0.10) 56(84) bytes of data:
64 bytes from 10.0.0.10: icmp_seq=1 ttl=62 time=0.115 ms
64 bytes from 10.0.0.10: icmp_seq=2 ttl=62 time=0.091 ms
64 bytes from 10.0.0.10: icmp_seq=3 ttl=62 time=0.094 ms
64 bytes from 10.0.0.10: icmp_seq=4 ttl=62 time=0.093 ms
^C
--- 10.0.0.10 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3049ms
rtt min/avg/max/mdev = 0.091/0.098/0.115/0.012 ms
root@n8: /tmp/pycore.42701/n8.conf#
```

Como em cada um dos computadores (n5 do departamento A, n8 do departamento B, n13 do departamento C e n16 do departamento D) são transmitidos pacotes (“4 packets transmitted”), confirmamos a conectividade de todos os departamentos com o servidor.

e) Verifique se existe conectividade IP do router de acesso Rext para o servidor S1.

```
root@Rext: /tmp/pycore.42701/Rext.conf
root@Rext:/tmp/pycore.42701/Rext.conf# ping 10.0.0.10
PING 10.0.0.10 (10.0.0.10) 56(84) bytes of data:
64 bytes from 10.0.0.10: icmp_seq=1 ttl=61 time=0.098 ms
64 bytes from 10.0.0.10: icmp_seq=2 ttl=61 time=0.112 ms
64 bytes from 10.0.0.10: icmp_seq=3 ttl=61 time=0.112 ms
64 bytes from 10.0.0.10: icmp_seq=4 ttl=61 time=0.111 ms
^C
--- 10.0.0.10 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3065ms
rtt min/avg/max/mdev = 0.098/0.108/0.112/0.009 ms
root@Rext:/tmp/pycore.42701/Rext.conf#
```

Tal como na alínea anterior, confirmamos que há conectividade do Rext para o S1 pois enviamos um ping desse router para o servidor e confirmamos que S1 recebe a totalidade dos pacotes enviados.

2.2) Para o router e um laptop do departamento B:

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 respetivo (man netstat).

```
root@Rb: /tmp/pycore.42701/Rb.conf# netstat -rn
Kernel IP routing table
Destination Gateway Genmask Flags MSS Window irtt Iface
10.0.0.0 10.0.5.1 255.255.255.0 UG 0 0 0 eth1
10.0.1.0 10.0.5.1 255.255.255.0 UG 0 0 0 eth1
10.0.2.0 0.0.0.0 255.255.255.0 U 0 0 0 eth0
10.0.3.0 10.0.9.2 255.255.255.0 UG 0 0 0 eth2
10.0.4.0 10.0.5.1 255.255.255.0 UG 0 0 0 eth1
10.0.5.0 0.0.0.0 255.255.255.0 U 0 0 0 eth1
10.0.6.0 10.0.5.1 255.255.255.0 UG 0 0 0 eth1
10.0.7.0 10.0.9.2 255.255.255.0 UG 0 0 0 eth2
10.0.8.0 10.0.9.2 255.255.255.0 UG 0 0 0 eth2
10.0.9.0 0.0.0.0 255.255.255.0 U 0 0 0 eth2
root@Rb: /tmp/pycore.42701/Rb.conf#

root@n9: /tmp/pycore.42701/n9.conf# netstat -rn
Kernel IP routing table
Destination Gateway Genmask Flags MSS Window irtt Iface
0.0.0.0 10.0.2.1 0.0.0.0 UG 0 0 0 eth0
10.0.2.0 0.0.0.0 255.255.255.0 U 0 0 0 eth0
root@n9: /tmp/pycore.42701/n9.conf#
```

Tabela do router b:

- Quando o destino é um IP da rede local do router ou um dos routers adjacentes (A e D), o pacote é enviado diretamente. A gateway nestas linhas da tabela é 0.0.0.0 porque não é necessário fazer nenhum salto intermédio para alcançar o destino.

- Para alcançar os outros destinos já é necessário indicar o próximo salto na gateway, que equivale a um dos IPs da interface dos routers adjacentes a que está ligado.

Tabela do portátil n9:

- O portátil envia pacotes para o router (primeira linha da tabela) através da interface do router b, a que se encontra conectado.
- O router depois reencaminha-os para os outros portáteis da rede local (segunda linha).

b) Diga, justificando, se está a ser usado encaminhamento estático ou dinâmico.

```

root@Ra: /tmp/pycore.45625/Ra.conf# ps -e
PID TTY          TIME CMD
  1 ?            00:00:00 vnoded
 59 ?            00:00:00 zebra
 65 ?            00:00:00 ospf6d
 69 ?            00:00:00 ospfd
 86 pts/2        00:00:00 bash
 94 pts/2        00:00:00 ps
root@Ra: /tmp/pycore.45625/Ra.conf#

```

A atualização dinâmica de rotas é obtida através de protocolos específicos de encaminhamento, nomeadamente os protocolos OSPF observados no print. Logo, está a ser usado encaminhamento dinâmico.

c) 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. Que implicação tem esta medida para os utilizadores da empresa que acedem ao servidor? Justifique.

```

root@S1: /tmp/pycore.42701/S1.conf# netstat -rn
Kernel IP routing table
Destination Gateway Genmask Flags MSS Window irtt Iface
0.0.0.0 10.0.0.1 0.0.0.0 UG 0 0 0 eth0
10.0.0.0 0.0.0.0 255.255.255.0 U 0 0 0 eth0
root@S1: /tmp/pycore.42701/S1.conf# route del default gw 0.0.0.0
root@S1: /tmp/pycore.42701/S1.conf# netstat -rn
Kernel IP routing table
Destination Gateway Genmask Flags MSS Window irtt Iface
10.0.0.0 0.0.0.0 255.255.255.0 U 0 0 0 eth0
root@S1: /tmp/pycore.42701/S1.conf#

```

O servidor S1 continua a ser capaz de comunicar com os restantes portáteis da subrede local do departamento A (10.0.0.0), todavia deixa de conseguir alcançar o router A e, como consequência, todo o resto da rede.

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

```
root@S1:/tmp/pycore.45625/S1.conf# netstat -rn
Kernel IP routing table
Destination        Gateway            Genmask           Flags   MSS Window  irtt Iface
10.0.0.0            0.0.0.0            255.255.255.0     U        0 0          0 eth0
root@S1:/tmp/pycore.45625/S1.conf# route add -net 10.0.1.0 netmask 255.255.255.0 gw 10.0.0.1
root@S1:/tmp/pycore.45625/S1.conf# route add -net 10.0.2.0 netmask 255.255.255.0 gw 10.0.0.1
root@S1:/tmp/pycore.45625/S1.conf# route add -net 10.0.3.0 netmask 255.255.255.0 gw 10.0.0.1
root@S1:/tmp/pycore.45625/S1.conf# netstat -rn
Kernel IP routing table
Destination        Gateway            Genmask           Flags   MSS Window  irtt Iface
10.0.0.0            0.0.0.0            255.255.255.0     U        0 0          0 eth0
10.0.1.0            10.0.0.1           255.255.255.0     UG       0 0          0 eth0
10.0.2.0            10.0.0.1           255.255.255.0     UG       0 0          0 eth0
10.0.3.0            10.0.0.1           255.255.255.0     UG       0 0          0 eth0
```

Com os comandos de route add apresentados no print acima, criamos uma rota estática entre o servidor e cada uma das redes dos departamentos B (10.0.2.0), C (10.0.1.0) e D (10.0.3.0) , isto é, 3 rotas adicionais. Assim, S1 não tem a necessidade de utilizar o router A quando quer enviar e/ou receber pacotes das outras redes.

e) 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.

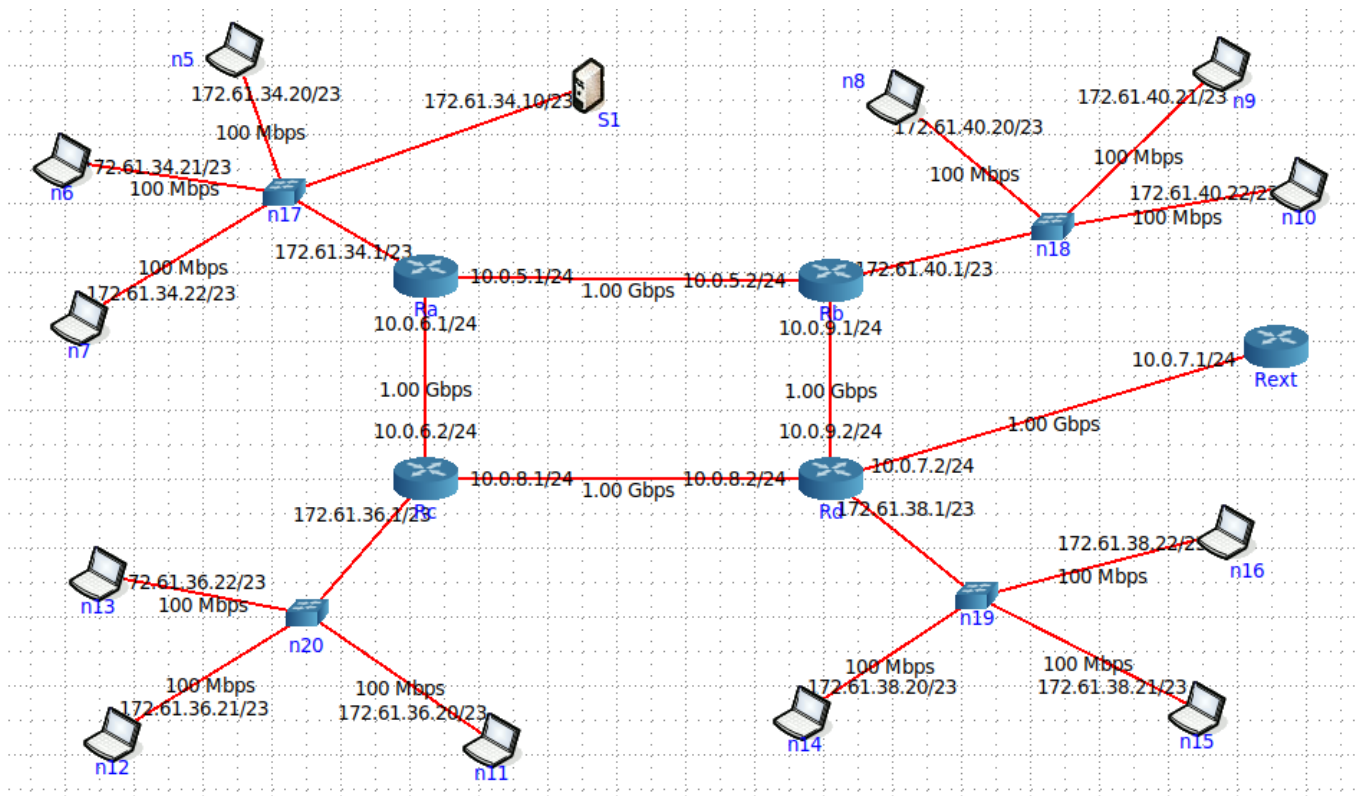
The image displays four terminal windows showing network configuration and connectivity tests:

- Top Left (root@S1):** Shows the kernel IP routing table after adding static routes for 10.0.1.0, 10.0.2.0, and 10.0.3.0 networks, all pointing to gateway 10.0.0.1.
- Top Right (root@n15):** Shows a successful ping test from server n15 to server S1 (10.0.0.10) with 7 packets received and 0% packet loss.
- Bottom Left (root@n12):** Shows a successful ping test from server n12 to server S1 (10.0.0.10) with 5 packets received and 0% packet loss.
- Bottom Right (root@n9):** Shows a successful ping test from server n9 to server S1 (10.0.0.10) with 6 packets received and 0% packet loss.

No canto superior esquerdo vamos a nova tabela do servidor. Efetuando um ping de um portátil de cada uma das redes dos outros departamentos, vemos que o servidor está de novo acessível.

3) Considere a topologia definida anteriormente. Assuma que o endereçamento entre os routers se mantém inalterado, contudo, o endereçamento em cada departamento deve ser redefinido.

3.1) Considere que dispõe apenas do endereço de rede IP 172.yyx.32.0/20, em que “yy” são os dígitos correspondendo ao seu número de grupo (Gyy) e “x” é o dígito correspondente ao seu turno prático (PLx). Defina um novo esquema de endereçamento para as redes dos departamentos (mantendo a rede de acesso e core inalteradas) e atribua endereços às interfaces dos vários sistemas envolvidos. Deve justificar as opções usadas.



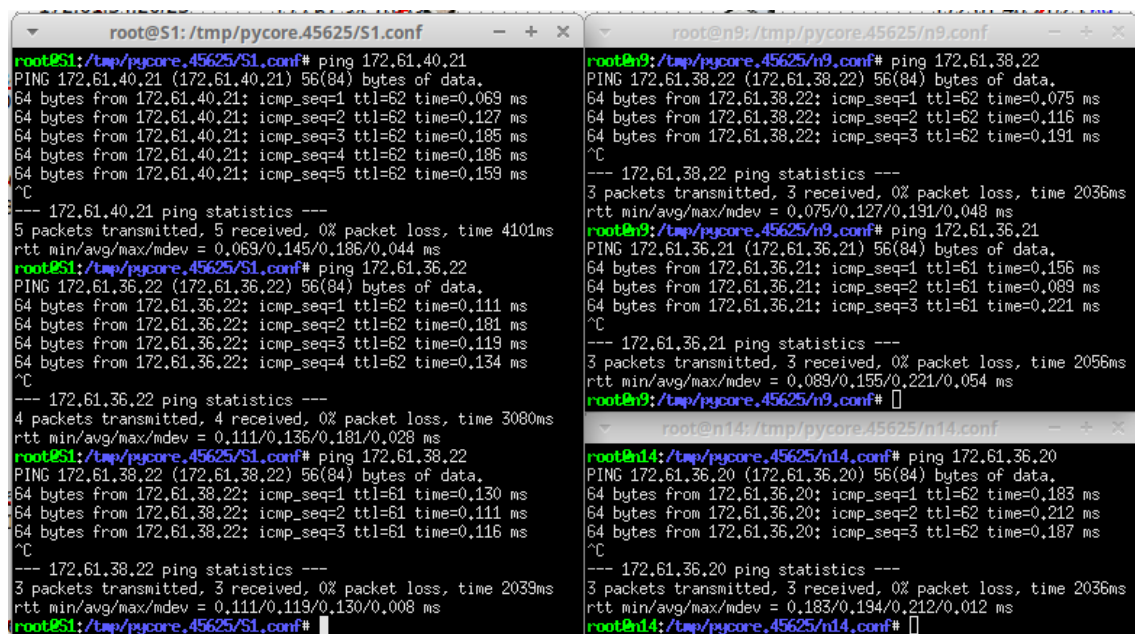
Substituindo yyx pelo nosso respetivo turno e número de grupo, obtivemos o IP 172.61.32.0/20 (yy = 06 e x = 1). A máscara representada por “/20” identifica que 20 bits vão ser reservados para representar a rede, sobrando 12 para poder representar as sub-redes e as interfaces. Sendo que apenas queremos criar 4 sub-redes como no modelo original, averiguamos qual o mínimo de bits necessários para as tais 4 combinações diferentes. Com n bits, temos 2^n combinações possíveis, mas temos que subtrair a este número 2 unidades pois as sequências em que todos os bits são 0’s e 1’s estão reservadas para representarem situações particulares. O menor n que respeita a inequação “ $2^n - 2 \geq 4$ ” é 3, logo vamos necessitar de 3 bits para representarmos estas 4 sub-redes. Neste caso, escolhemos as combinações 001, 010, 011 e 100 que, agrupadas com os bits do número 32, vão gerar os números 34, 36, 38 e 40. Como observamos na imagem acima, cada um destes números vai representar um departamento diferente.

3.2) Qual a máscara de rede que usou (em notação decimal)? Quantos interfaces IP pode interligar em cada departamento? Justifique.

Usamos a máscara de rede 255.255.254.0, que é resultante de adicionar os 3 bits adicionais necessários para representar as sub-redes à máscara da rede dada “/20”, ou em decimal, 255.255.240.0.

Como só 3 bits vão ser utilizados para definir a sub-rede, sobram ainda $32 - 20 - 3 = 9$ bits. Com estes bits, podemos obter 2^9 combinações diferentes. Novamente, retiramos as 2 exceções de quando os bits estão todos a 0 ou todos a 1, podendo cada departamento ter $2^9 - 2$ interfaces.

3.3) Garanta e verifique que a conectividade IP entre as várias redes locais da organização MIEI-RC é mantida. Explique como procedeu.



```
root@S1:/tmp/pycore.45625/S1.conf# ping 172.61.40.21
PING 172.61.40.21 (172.61.40.21) 56(84) bytes of data.
64 bytes from 172.61.40.21: icmp_seq=1 ttl=62 time=0.069 ms
64 bytes from 172.61.40.21: icmp_seq=2 ttl=62 time=0.127 ms
64 bytes from 172.61.40.21: icmp_seq=3 ttl=62 time=0.185 ms
64 bytes from 172.61.40.21: icmp_seq=4 ttl=62 time=0.186 ms
64 bytes from 172.61.40.21: icmp_seq=5 ttl=62 time=0.159 ms
^C
--- 172.61.40.21 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4101ms
rtt min/avg/max/mdev = 0.069/0.145/0.186/0.044 ms
root@S1:/tmp/pycore.45625/S1.conf# ping 172.61.36.22
PING 172.61.36.22 (172.61.36.22) 56(84) bytes of data.
64 bytes from 172.61.36.22: icmp_seq=1 ttl=62 time=0.111 ms
64 bytes from 172.61.36.22: icmp_seq=2 ttl=62 time=0.181 ms
64 bytes from 172.61.36.22: icmp_seq=3 ttl=62 time=0.119 ms
64 bytes from 172.61.36.22: icmp_seq=4 ttl=62 time=0.134 ms
^C
--- 172.61.36.22 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3080ms
rtt min/avg/max/mdev = 0.111/0.136/0.181/0.028 ms
root@S1:/tmp/pycore.45625/S1.conf# ping 172.61.38.22
PING 172.61.38.22 (172.61.38.22) 56(84) bytes of data.
64 bytes from 172.61.38.22: icmp_seq=1 ttl=61 time=0.130 ms
64 bytes from 172.61.38.22: icmp_seq=2 ttl=61 time=0.111 ms
64 bytes from 172.61.38.22: icmp_seq=3 ttl=61 time=0.116 ms
^C
--- 172.61.38.22 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2039ms
rtt min/avg/max/mdev = 0.111/0.119/0.130/0.008 ms
root@S1:/tmp/pycore.45625/S1.conf#

root@n9:/tmp/pycore.45625/n9.conf# ping 172.61.38.22
PING 172.61.38.22 (172.61.38.22) 56(84) bytes of data.
64 bytes from 172.61.38.22: icmp_seq=1 ttl=62 time=0.075 ms
64 bytes from 172.61.38.22: icmp_seq=2 ttl=62 time=0.116 ms
64 bytes from 172.61.38.22: icmp_seq=3 ttl=62 time=0.191 ms
^C
--- 172.61.38.22 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2036ms
rtt min/avg/max/mdev = 0.075/0.127/0.191/0.048 ms
root@n9:/tmp/pycore.45625/n9.conf# ping 172.61.36.21
PING 172.61.36.21 (172.61.36.21) 56(84) bytes of data.
64 bytes from 172.61.36.21: icmp_seq=1 ttl=61 time=0.156 ms
64 bytes from 172.61.36.21: icmp_seq=2 ttl=61 time=0.089 ms
64 bytes from 172.61.36.21: icmp_seq=3 ttl=61 time=0.221 ms
^C
--- 172.61.36.21 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2056ms
rtt min/avg/max/mdev = 0.089/0.155/0.221/0.054 ms
root@n9:/tmp/pycore.45625/n9.conf#

root@n14:/tmp/pycore.45625/n14.conf# ping 172.61.36.20
PING 172.61.36.20 (172.61.36.20) 56(84) bytes of data.
64 bytes from 172.61.36.20: icmp_seq=1 ttl=62 time=0.183 ms
64 bytes from 172.61.36.20: icmp_seq=2 ttl=62 time=0.212 ms
64 bytes from 172.61.36.20: icmp_seq=3 ttl=62 time=0.187 ms
^C
--- 172.61.36.20 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2036ms
rtt min/avg/max/mdev = 0.183/0.194/0.212/0.012 ms
root@n14:/tmp/pycore.45625/n14.conf#
```

Para verificar a conectividade, efetuou-se pings entre todas as sub-redes, verificando que tudo foi enviado e recebido como esperado.

2. Conclusões

Este trabalho prático permitiu a consolidação da matéria dada nas aulas teóricas e provou-se um método de aprendizagem muito eficaz devido à aplicação direta dos conhecimentos, desde a montagem das topologias à experimentação com vários aspetos das mesmas, nomeadamente endereços e máscaras.

Em jeito de resumo, apresenta-se de seguida os resultados mais relevantes da aprendizagem decorrente deste trabalho:

- Montagem de topologias de redes;
- Teste da conectividade entre máquinas da rede e resolução de problemas através de encaminhamento estático;
- Análise de tráfego de mensagens ICMP e respetivas características das mesmas, com ênfase no seu tempo de vida;
- Fragmentação de pacotes e motivação da mesma;
- Tipos de endereçamento e encaminhamento usados numa rede (e como os determinar);
- Análise de tabelas de encaminhamento e rotas entre máquinas.

O grupo considera que foi bem-sucedido no trabalho e que foi uma experiência enriquecedora e fulcral para o desenvolvimento e consolidação de conhecimentos da matéria de Redes de Computadores.