

Universidade do Minho

MESTRADO INTEGRADO EM ENGENHARIA INFORMÁTICA

22/10/2019
Redes de Computadores
TP2: Protocolo IPv4

PL4 Grupo 9:
António Gonçalves (A85516)
João Araújo (A84306)
João Silva (A81761)

Contents

1	Parte 1	2
1.1	Questão 1 - Comportamento do traceroute na topologia CORE	2
1.2	Questão 2 - Traceroute na máquina nativa e criação de datagramas IP .	3
1.3	Questão 3 - Análise da fragmentação de pacotes de IP	5
2	Parte 2	8
2.1	Questão 1 - Endereçamento e Encaminhamento IP	8
2.2	Questão 2 - Definição de Sub-redes	14
3	Conclusão	16

1 Parte 1

1.1 Questão 1 - Comportamento do traceroute na topologia CORE

A topologia CORE que nos foi pedida para implementar neste parte do trabalho foi a seguinte:

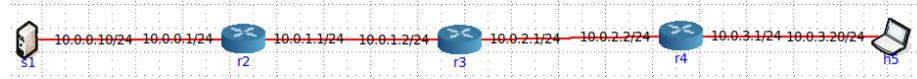


Figure 1: Topologia CORE.

Alínea 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.38519/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)  0.036 ms  0.010 ms  0.007 ms
 2 10.0.1.2 (10.0.1.2)  0.019 ms  0.011 ms  0.010 ms
 3 10.0.2.2 (10.0.2.2)  0.022 ms  0.014 ms  0.015 ms
 4 10.0.3.20 (10.0.3.20)  0.050 ms  0.021 ms  0.017 ms
```

Figure 2: Execução do traceroute -I 10.0.3.20.

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

Após executar o comando “traceroute -I 10.0.2.10” a partir do host s1, são devolvidos 4 tempos relativos aos 4 pacotes enviados para o host h5. Como se observa, o tráfego correspondente a enviar do s1 para o host h5 corresponde a 4 datagramas.

133	78.098260907	10.0.0.10	10.0.3.20	ICMP	74 Echo (ping) request id=0x0025, seq=1/256, ttl=1 (no response found!)
134	78.098260944	10.0.0.10	10.0.3.20	ICMP	102 time-to-live exceeded (time to live exceeded in transit)
135	78.098260959	10.0.0.10	10.0.3.20	ICMP	74 Echo (ping) request id=0x0025, seq=2/256, ttl=1 (no response found!)
136	78.098260974	10.0.0.10	10.0.3.20	ICMP	102 time-to-live exceeded (time to live exceeded in transit)
137	78.098260989	10.0.0.10	10.0.3.20	ICMP	74 Echo (ping) request id=0x0025, seq=3/256, ttl=1 (no response found!)
138	78.098261004	10.0.0.10	10.0.3.20	ICMP	102 time-to-live exceeded (time to live exceeded in transit)
139	78.098261019	10.0.0.10	10.0.3.20	ICMP	74 Echo (ping) request id=0x0025, seq=4/256, ttl=2 (no response found!)
140	78.098261034	10.0.0.10	10.0.3.20	ICMP	102 time-to-live exceeded (time to live exceeded in transit)
141	78.098261049	10.0.0.10	10.0.3.20	ICMP	74 Echo (ping) request id=0x0025, seq=5/256, ttl=2 (no response found!)
142	78.098261064	10.0.0.10	10.0.3.20	ICMP	102 time-to-live exceeded (time to live exceeded in transit)
143	78.098261079	10.0.0.10	10.0.3.20	ICMP	74 Echo (ping) request id=0x0025, seq=6/256, ttl=2 (no response found!)
144	78.098261094	10.0.0.10	10.0.3.20	ICMP	102 time-to-live exceeded (time to live exceeded in transit)
145	78.098261109	10.0.0.10	10.0.3.20	ICMP	74 Echo (ping) request id=0x0025, seq=7/256, ttl=2 (no response found!)
146	78.098261124	10.0.0.10	10.0.3.20	ICMP	102 time-to-live exceeded (time to live exceeded in transit)
147	78.098261139	10.0.0.10	10.0.3.20	ICMP	74 Echo (ping) request id=0x0025, seq=8/256, ttl=3 (no response found!)
148	78.098261154	10.0.0.10	10.0.3.20	ICMP	102 time-to-live exceeded (time to live exceeded in transit)
149	78.098261169	10.0.0.10	10.0.3.20	ICMP	74 Echo (ping) request id=0x0025, seq=9/256, ttl=3 (no response found!)
150	78.098261184	10.0.0.10	10.0.3.20	ICMP	102 time-to-live exceeded (time to live exceeded in transit)
151	78.098261199	10.0.0.10	10.0.3.20	ICMP	74 Echo (ping) request id=0x0025, seq=10/256, ttl=4 (reply in 152)
152	78.098261214	10.0.0.10	10.0.3.20	ICMP	74 Echo (ping) reply id=0x0025, seq=10/256, ttl=4 (request in 151)
153	78.098261229	10.0.0.10	10.0.3.20	ICMP	74 Echo (ping) request id=0x0025, seq=11/256, ttl=4 (reply in 154)
154	78.098261244	10.0.0.10	10.0.3.20	ICMP	74 Echo (ping) reply id=0x0025, seq=11/256, ttl=4 (request in 153)
155	78.098261259	10.0.0.10	10.0.3.20	ICMP	74 Echo (ping) request id=0x0025, seq=12/256, ttl=4 (reply in 156)
156	78.098261274	10.0.0.10	10.0.3.20	ICMP	74 Echo (ping) reply id=0x0025, seq=12/256, ttl=4 (request in 155)

Figure 3: Tráfego ICMP enviado por s1 e tráfego recebido como resposta.

Alínea 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.

O valor mínimo do campo TTL deverá ser 4 para poder alcançar o h5, como podemos ver pela imagem da pergunta anterior.

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

O tempo médio será: $(0.050 + 0.021 + 0.017)/3 \approx 0,029(3)$

1.2 Questão 2 - Traceroute na máquina nativa e criação de data-gramas IP

Alínea a. -Qual é o endereço IP da interface ativa do seu computador?

1 0.000000	192.168.100.196	239.255.255.250	SSDP	216 M-SEARCH * HTTP/1.1
44 6.275072	192.168.100.196	224.0.0.251	MDNS	107 Standard query 0x0000 PTR _googlecast._tcp.local, "Q" question PTR _airplay._tcp.local, "Q" question
19 3.485784	192.168.100.200	239.255.255.250	SSDP	216 M-SEARCH * HTTP/1.1
28 4.486681	192.168.100.200	239.255.255.250	SSDP	216 M-SEARCH * HTTP/1.1
32 5.487927	192.168.100.200	239.255.255.250	SSDP	216 M-SEARCH * HTTP/1.1
45 6.487887	192.168.100.200	239.255.255.250	SSDP	216 M-SEARCH * HTTP/1.1

Figure 4: IP da interface nativa

O IP do interface ativa do PC é 192.168.100.196

Alínea b. -Qual é o valor do campo protocolo? O que identifica?

```

Internet Protocol Version 4, Src: 192.168.100.211, 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: 0x66ba (26298)
  > Flags: 0x0000
    Time to live: 255
    Protocol: ICMP (1)
    Header checksum: 0x0000 [validation disabled]
    [Header checksum status: Unverified]

```

Figure 5: Informações sobre o cabeçalho IPv4.

Como podemos observar, o valor do protocolo é 1 e este identifica o protocolo ICMP.

Alínea c. -Quantos bytes tem o cabeçalho IP(v4)? Quantos bytes tem o campo de dados (payload) do datagrama? Como se calcula o tamanho do payload?

O cabeçalho IP(v4) tem 20 bytes e o campo de dados 56 bytes (como podemos observar na Fig.5). Assim, o tamanho do payload será: $56 - 20 = 36$ bytes

Alínea d. -O datagrama IP foi fragmentado? Justifique.

```
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
```

Figure 6: Flags da mensagem.

Podemos observar na imagem a cima que esta mensagem não foi fragmentada, uma vez que o *Fragment offset* está a 0 e a flag *More fragment* está como Not set.

Alínea 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.

No.	Time	Source	Destination	Protocol	Length	Info
15	1.464990	172.16.115.252	172.26.105.126	ICMP	70	Time-to-live exceeded (Time to live exceeded in transit)
25	3.937456	172.16.115.252	172.26.105.126	ICMP	70	Time-to-live exceeded (Time to live exceeded in transit)
13	1.425498	172.16.2.1	172.26.105.126	ICMP	70	Time-to-live exceeded (Time to live exceeded in transit)
23	3.887298	172.16.2.1	172.26.105.126	ICMP	70	Time-to-live exceeded (Time to live exceeded in transit)
1	0.000000	172.26.105.126	64.233.166.188	TCP	55	50179 → 5228 [ACK] Seq=1 Ack=1 Win=510 Len=1
4	0.908871	172.26.105.126	34.73.232.153	TLSv1.2	92	Application Data
6	1.269678	172.26.105.126	193.137.16.65	DNS	75	Standard query 0x20d3 A marco.uminho.pt
8	1.351732	172.26.105.126	193.136.9.240	ICMP	70	Echo (ping) request id=0x0001, seq=1/256, ttl=255 (reply in 9)
10	1.385555	172.26.105.126	193.136.9.240	ICMP	70	Echo (ping) request id=0x0001, seq=2/512, ttl=1 (no response t
12	1.424267	172.26.105.126	193.136.9.240	ICMP	70	Echo (ping) request id=0x0001, seq=3/768, ttl=2 (no response t

> Frame 15: 70 bytes on wire (560 bits), 70 bytes captured (560 bits) on interface 0	
> Ethernet II, Src: ComdaEnt_ff:94:00 (00:00:03:ff:94:00), Dst: IntelCor_09:5b:2d (1c:1b:b5:09:5b:2d)	
Internet Protocol Version 4, Src: 172.16.115.252, Dst: 172.26.105.126 <ul style="list-style-type: none"> 0100 = Version: 4 0101 = Header Length: 20 bytes (5) > Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT) Total Length: 56 Identification: 0xe508 (58632) > Flags: 0x0000 Time to live: 253 Protocol: ICMP (1) Header checksum: 0xa316 [validation disabled] [Header checksum status: Unverified] Source: 172.16.115.252 Destination: 172.26.105.126 	
> Internet Control Message Protocol	

Figure 7: Pacotes ordenados de acordo com o IP da fonte.

Os campos que variam são o campo de identificação e o TTL(Time to Live).

Alínea f. -Observa algum padrão nos valores do campo de Identificação do datagrama IP e TTL?

Os valores de identificação dos pacotes consecutivos têm um valor de diferença (ordem crescente). Os valores de TTL também são sequenciais, até chegar ao valor 4.

Alínea 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?

O valor do TTL inicial vai diminuindo em 1 valor sempre que é mandada outra resposta.

11	1.387047	172.26.254.254	172.26.105.126	ICMP	70 Time-to-live exceeded (Time to live exceeded in transit)
13	1.425498	172.16.2.1	172.26.105.126	ICMP	70 Time-to-live exceeded (Time to live exceeded in transit)
15	1.464990	172.16.115.252	172.26.105.126	ICMP	70 Time-to-live exceeded (Time to live exceeded in transit)
17	1.503102	193.136.9.240	172.26.105.126	ICMP	70 Echo (ping) reply id=0x0001, seq=5/1280, ttl=61 (request in 16)
19	3.787609	193.136.9.240	172.26.105.126	ICMP	70 Echo (ping) reply id=0x0001, seq=6/1536, ttl=61 (request in 18)

```

> Frame 11: 70 bytes on wire (560 bits), 70 bytes captured (560 bits) on interface 0
> Ethernet II, Src: ComdaEnt_ff:94:00 (00:d0:03:ff:94:00), Dst: IntelCon_09:5b:2d (1c:1b:b5:09:5b:2d)
> Internet Protocol Version 4, Src: 172.26.254.254, Dst: 172.26.105.126
  0100 .... = Version: 4
  .... 0101 = Header Length: 20 bytes (5)
  > Differentiated Services Field: 0xc0 (DSCP: CS6, ECN: Not-ECT)
    Total Length: 56
    Identification: 0xd336 (54070)
  > Flags: 0x0000
    Time to live: 255
    Protocol: ICMP (1)
    Header checksum: 0x271c [validation disabled]
    [Header checksum status: Unverified]
    Source: 172.26.254.254
    Destination: 172.26.105.126
> Internet Control Message Protocol

```

1.3 Questão 3 - Análise da fragmentação de pacotes de IP

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

10	1.918912	172.26.105.126	193.136.9.240	IPv4	1514 Fragmented IP protocol (proto=ICMP 1, off=0, ID=7386) [Reassembled in #12]
11	1.918916	172.26.105.126	193.136.9.240	IPv4	1514 Fragmented IP protocol (proto=ICMP 1, off=1480, ID=7386) [Reassembled in #12]
12	1.918916	172.26.105.126	193.136.9.240	ICMP	1263 Echo (ping) request id=0x0001, seq=22/5632, ttl=1 (no response found)
13	1.920924	172.26.254.254	172.26.105.126	ICMP	70 Time-to-live exceeded (Time to live exceeded in transit)
14	1.957270	172.26.105.126	193.136.9.240	IPv4	1514 Fragmented IP protocol (proto=ICMP 1, off=0, ID=7387) [Reassembled in #16]
15	1.957273	172.26.105.126	193.136.9.240	IPv4	1514 Fragmented IP protocol (proto=ICMP 1, off=1480, ID=7387) [Reassembled in #16]
16	1.957274	172.26.105.126	193.136.9.240	ICMP	1263 Echo (ping) request id=0x0001, seq=23/5688, ttl=2 (no response found)
17	1.959122	172.16.2.1	172.26.105.126	ICMP	70 Time-to-live exceeded (Time to live exceeded in transit)

```

> Frame 10: 1514 bytes on wire (12112 bits), 1514 bytes captured (12112 bits) on interface 0
> Ethernet II, Src: IntelCon_09:5b:2d (1c:1b:b5:09:5b:2d), Dst: ComdaEnt_ff:94:00 (00:d0:03:ff:94:00)
> Internet Protocol Version 4, Src: 172.26.105.126, 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: 0x7386 (29574)
  > 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

```

Figure 8: Primeira mensagem ICMP.

Houve necessidade de fragmentar o pacote inicial uma vez que o novo tamanho do pacote (neste caso, 4209) é demasiado grande para ser enviado num só (tamanho Maximo de um fragmento = 1500 bytes).

Alínea 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?

Como podemos observar na imagem anterior, o *Fragment offset* tem o valor de 0 e a flag *More fragments* como Set, o que nos permite concluir que o datagrama foi fragmentado. Com a imagem seguinte conseguimos observar o tamanho dos fragmentos que fazem parte do datagrama, sendo o tamanho do primeiro fragmento 1480 bytes.

```
[3 IPv4 Fragments (4189 bytes): #10(1480), #11(1480), #12(1229)]
[Frame: 10, payload: 0-1479 (1480 bytes)]
[Frame: 11, payload: 1480-2959 (1480 bytes)]
[Frame: 12, payload: 2960-4188 (1229 bytes)]
[Fragment count: 3]
[Reassembled IPv4 length: 4189]
[Reassembled IPv4 data: 080091a200010016202020202020202020202020202020...]
```

Figure 9: Os fragmentos da mensagem.

Alínea c- Imprima o segundo fragmento do datagrama IP original. Que informação do cabeçalho IP indica que não se trata do 1º fragmento? Há mais fragmentos? O que nos permite afirmar isso?

11 1.918916	172.26.105.126	193.136.9.240	IPv4	1514 Fragmented IP protocol (proto=ICMP 1, off=1480, ID=7386) [Reassembled in #12]
12 1.918916	172.26.105.126	193.136.9.240	ICMP	1263 Echo (ping) request id=0x0001, seq=22/5632, ttl=1 (no response found!)
13 1.920924	172.26.254.254	172.26.105.126	ICMP	70 Time-to-live exceeded (Time to live exceeded in transit)
14 1.957270	172.26.105.126	193.136.9.240	IPv4	1514 Fragmented IP protocol (proto=ICMP 1, off=0, ID=7387) [Reassembled in #16]
15 1.957273	172.26.105.126	193.136.9.240	IPv4	1514 Fragmented IP protocol (proto=ICMP 1, off=1480, ID=7387) [Reassembled in #16]
16 1.957274	172.26.105.126	193.136.9.240	ICMP	1263 Echo (ping) request id=0x0001, seq=23/5888, ttl=2 (no response found!)
17 1.959171	172.16.2.1	172.26.105.126	ICMP	70 Time-to-live exceeded (Time to live exceeded in transit)

```
> Ethernet II, Src: IntelCor_09:5b:2d (1c:1b:b5:09:5b:2d), Dst: ComdaEnt_ff:94:00 (00:d0:03:ff:94:00)
> Internet Protocol Version 4, Src: 172.26.105.126, 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: 0x7386 (29574)
    Flags: 0x20b9, More fragments
        0... .... = Reserved bit: Not set
        .0... .... = Don't fragment: Not set
        ..1. .... = More fragments: Set
        ...0 0000 1011 1001 = Fragment offset: 185
> Time to live: 1
```

Figure 10: Segundo fragmento do datagrama IP original.

Podemos dizer que este fragmento não é o primeiro uma vez que o seu *Fragment offset* é diferente de 0. Para além disso, conseguimos afirmar que há mais fragmentos uma vez que a flag *More fragments* está Set.

**Alínea d- Quantos fragmentos foram criados a partir do datagrama original?
Como se detecta o último fragmento correspondente ao datagrama original?**

11	1.918916	172.26.105.126	193.136.9.240	18v4	1514 fragmented IP protocol (proto-ICMP 1, off=1480, ID=7386) [Reassembled in #12]
12	1.918916	172.26.105.126	193.136.9.240	ICMP	1263 Echo (ping) request id=0x0001, seq=22/5632, ttl=1 (no response found)

```

> Frame 12: 1263 bytes on wire (10104 bits), 1263 bytes captured (10104 bits) on interface 0
> Ethernet II, Src: IntelCor_09:5b:2d (1c:1b:b5:09:5b:2d), Dst: ComdaEnt_ff:94:00 (00:d0:03:ff:94:00)
✓ Internet Protocol Version 4, Src: 172.26.105.126, 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: 1249
  Identification: 0x7386 (29574)
  Flags: 0x0172
    0... .. = Reserved bit: Not set
    .0. .... = Don't fragment: Not set
    ..0. .... = More fragments: Not set
    ...0 0001 0111 0010 = Fragment offset: 370

```

Figure 11: Informações sobre o último fragmento do datagrama.

Foram criados 3 fragmentos ao todo a partir do datagrama original. Podemos afirmar que o datagrama presente na imagem é o último já que a flag *More Fragments* está Not set e o *Fragment offset* tem um valor diferente de 0.

Alínea 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.

Resumindo, os campos do cabeçalho IP que mudam entre os vários fragmentos são o *Fragment offset* e a Flag *More Fragments*.

O *Fragment offset* permite-nos saber que parte da mensagem está a ser enviada, servindo para reconstruir a mensagem original. Assim, quando um dos fragmentos é tratado, o *Fragment offset* vai ser alterado para o início do próximo fragmento.

A Flag *More fragments* indica se existem ou não outros fragmentos que ainda não foram lidos, impedindo que alguns fragmentos sejam perdidos no processo.

2 Parte 2

2.1 Questão 1 - Endereçamento e Encaminhamento IP

2.1.1 - Atenda aos endereços IP atribuídos automaticamente pelo CORE aos diversos equipamentos da topologia.

Alínea 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.

Na imagem seguinte podemos ver os endereços IP atribuídos aos vários equipamentos. Podemos também observar que a Máscara utilizada é a 255.255.255.0 (ou /24 notação CIDR).

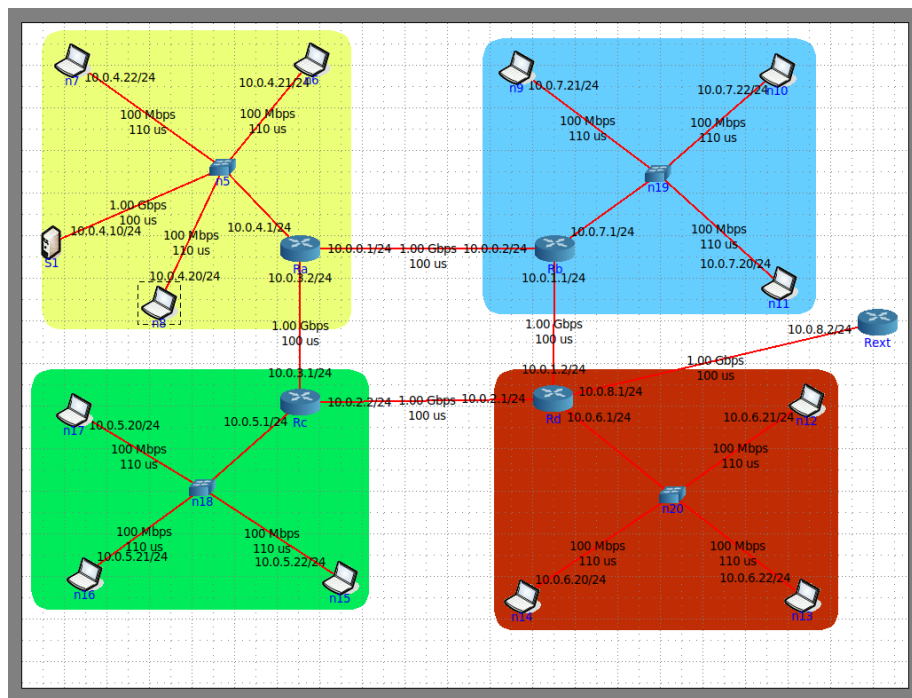


Figure 12: Topologia da Parte 2 do trabalho.

Alínea b. - Trata-se de endereços públicos ou privados? Porquê?

Uma vez que os IPs dos vários equipamentos se encontram no intervalo de 10.0.0.0 e 10.255.255.255, podemos concluir que são privados.

Alínea c. - Por que razão não é atribuído um endereço IP aos switches?

Não são atribuídos IPs aos switches uma vez que estes são switches de Ethernet.

Alínea 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).

As seguintes imagens comprovam que o servidor S1 está ligado a todos os outros laptops, uma vez que os pacotes enviados conseguem chegar ao destino.

```
root@S1:/tmp/pycore.38965/S1.conf# ping 10.0.4.22 -c 5
PING 10.0.4.22 (10.0.4.22) 56(84) bytes of data.
64 bytes from 10.0.4.22: icmp_seq=1 ttl=64 time=1.01 ms
64 bytes from 10.0.4.22: icmp_seq=2 ttl=64 time=2.10 ms
64 bytes from 10.0.4.22: icmp_seq=3 ttl=64 time=0.388 ms
64 bytes from 10.0.4.22: icmp_seq=4 ttl=64 time=1.09 ms
64 bytes from 10.0.4.22: icmp_seq=5 ttl=64 time=1.44 ms

--- 10.0.4.22 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4014ms
rtt min/avg/max/mdev = 0.388/1.206/2.102/0.562 ms
root@S1:/tmp/pycore.38965/S1.conf#
```

Figure 13: Conectividade entre S1 e um laptop do departamento A.

```
root@S1:/tmp/pycore.38965/S1.conf# ping 10.0.7.21 -c 5
PING 10.0.7.21 (10.0.7.21) 56(84) bytes of data.
64 bytes from 10.0.7.21: icmp_seq=1 ttl=62 time=3.27 ms
64 bytes from 10.0.7.21: icmp_seq=2 ttl=62 time=2.37 ms
64 bytes from 10.0.7.21: icmp_seq=3 ttl=62 time=3.39 ms
64 bytes from 10.0.7.21: icmp_seq=4 ttl=62 time=2.31 ms
64 bytes from 10.0.7.21: icmp_seq=5 ttl=62 time=2.87 ms

--- 10.0.7.21 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4005ms
rtt min/avg/max/mdev = 2.319/2.846/3.392/0.447 ms
root@S1:/tmp/pycore.38965/S1.conf#
```

Figure 14: Conectividade entre S1 e um laptop do departamento B.

```

root@S1:/tmp/pycore.38965/S1.conf# ping 10.0.5.21 -c 5
PING 10.0.5.21 (10.0.5.21) 56(84) bytes of data.
64 bytes from 10.0.5.21: icmp_seq=1 ttl=62 time=1.63 ms
64 bytes from 10.0.5.21: icmp_seq=2 ttl=62 time=1.08 ms
64 bytes from 10.0.5.21: icmp_seq=3 ttl=62 time=0.770 ms
64 bytes from 10.0.5.21: icmp_seq=4 ttl=62 time=1.32 ms
64 bytes from 10.0.5.21: icmp_seq=5 ttl=62 time=0.954 ms

--- 10.0.5.21 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4011ms
rtt min/avg/max/mdev = 0.770/1.153/1.634/0.301 ms
root@S1:/tmp/pycore.38965/S1.conf#

```

Figure 15: Conectividade entre S1 e um laptop do departamento C.

```

root@S1:/tmp/pycore.38965/S1.conf# ping 10.0.6.20 -c 5
PING 10.0.6.20 (10.0.6.20) 56(84) bytes of data.
64 bytes from 10.0.6.20: icmp_seq=1 ttl=61 time=3.22 ms
64 bytes from 10.0.6.20: icmp_seq=2 ttl=61 time=4.89 ms
64 bytes from 10.0.6.20: icmp_seq=3 ttl=61 time=2.87 ms
64 bytes from 10.0.6.20: icmp_seq=4 ttl=61 time=2.77 ms
64 bytes from 10.0.6.20: icmp_seq=5 ttl=61 time=4.67 ms

--- 10.0.6.20 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4008ms
rtt min/avg/max/mdev = 2.773/3.686/4.898/0.913 ms
root@S1:/tmp/pycore.38965/S1.conf#

```

Figure 16: Conectividade entre S1 e um laptop do departamento D.

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

Todos os pacotes enviados foram recebidos, logo existe conectividade OP do router Rext e o servidor S1.

```

root@S1:/tmp/pycore.38965/S1.conf# ping 10.0.8.2 -c 5
PING 10.0.8.2 (10.0.8.2) 56(84) bytes of data.
64 bytes from 10.0.8.2: icmp_seq=1 ttl=61 time=2.02 ms
64 bytes from 10.0.8.2: icmp_seq=2 ttl=61 time=1.89 ms
64 bytes from 10.0.8.2: icmp_seq=3 ttl=61 time=2.54 ms
64 bytes from 10.0.8.2: icmp_seq=4 ttl=61 time=2.05 ms
64 bytes from 10.0.8.2: icmp_seq=5 ttl=61 time=4.31 ms

--- 10.0.8.2 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4006ms
rtt min/avg/max/mdev = 1.896/2.566/4.315/0.903 ms
root@S1:/tmp/pycore.38965/S1.conf#

```

Figure 17: Conectividade entre o Router Rext e o servidor S1.

2.1.2 - Para o router e um laptop do departamento B:

Alinea 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@n9:/tmp/pycore.38965/n9.conf# netstat -rn
Kernel IP routing table
Destination Gateway Genmask Flags MSS Window irtt Iface
0.0.0.0 10.0.7.1 0.0.0.0 UG 0 0 0 eth0
10.0.7.0 0.0.0.0 255.255.255.0 U 0 0 0 eth0
root@n9:/tmp/pycore.38965/n9.conf#
```

Figure 18: Resultado do comando `netstat -rn` para um laptop.

Para o laptop existem só duas rotas: a default, que tem como destino o router Rb (0.0.0.0), e a outra tem como destino o departamento B (10.0.7.0).

```
root@Rb:/tmp/pycore.38965/Rb.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 0.0.0.0 255.255.255.0 U 0 0 0 eth1
10.0.2.0 10.0.1.2 255.255.255.0 UG 0 0 0 eth1
10.0.3.0 10.0.0.1 255.255.255.0 UG 0 0 0 eth0
10.0.4.0 10.0.0.1 255.255.255.0 UG 0 0 0 eth0
10.0.5.0 10.0.0.1 255.255.255.0 UG 0 0 0 eth0
10.0.6.0 10.0.1.2 255.255.255.0 UG 0 0 0 eth1
10.0.7.0 0.0.0.0 255.255.255.0 U 0 0 0 eth2
10.0.8.0 10.0.1.2 255.255.255.0 UG 0 0 0 eth1
root@Rb:/tmp/pycore.38965/Rb.conf#
```

Figure 19: Resultado do comando `netstat -rn` para um router.

Para o Router Rb existem várias entradas, como por exemplo, um pacote que tenha de chegar a 10.0.2.0 terá obrigatoriamente de passar por 10.0.1.2.

Alinea b. - Diga, justificando, se está a ser usado encaminhamento estático ou dinâmico (sugestão: analise que processos estão a correr em cada sistema).

```
root@Rb:/tmp/pycore_38965/Rb_conf# ps -A
PID TTY          TIME CMD
  1 ?            00:00:00 vncd
 55 ?            00:00:00 zebra
 65 ?            00:00:00 ospf6d
 68 ?            00:00:00 ospfd
100 pts/3        00:00:00 bash
108 pts/3        00:00:00 ps
```

Figure 20: Processos a decorrer no Router Rb.

Entre os Routers está a ser utilizado encaminhamento dinâmico, uma vez que existem vários caminhos possíveis (podemos observar o protocolo ospfd, que permite ao pacote seguir vários caminhos diferentes quando não lhes é possível seguir a rota predefinida).

```
root@n9:/tmp/pycore_38965/n9_conf# ps -A
PID TTY          TIME CMD
  1 ?            00:00:00 vncd
 26 pts/3        00:00:00 bash
 34 pts/3        00:00:00 ps
```

Figure 21: Processos a decorrer no Laptop n9.

Nos laptops, está a ser utilizado encaminhamento estático, uma vez que só existe uma rota possível. Caso não seja possível seguir esse caminho, o pacote é descartado.

Alinea 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_38965/S1_conf# netstat -rn
Kernel IP routing table
Destination Gateway Genmask Flags MSS Window irtt Iface
0.0.0.0 10.0.4.1 0.0.0.0 UG 0 0 0 eth0
10.0.4.0 0.0.0.0 255.255.255.0 U 0 0 0 eth0
root@S1:/tmp/pycore_38965/S1_conf# route delete default
root@S1:/tmp/pycore_38965/S1_conf# netstat -rn
Kernel IP routing table
Destination Gateway Genmask Flags MSS Window irtt Iface
10.0.4.0 0.0.0.0 255.255.255.0 U 0 0 0 eth0
root@S1:/tmp/pycore_38965/S1_conf#
```

Figure 22: Resultado do comando netstat -rn antes e depois de remover a rota *default*.

Ao remover a rota default, estamos a impossibilitar o servidor S1 de se ligar a equipamentos fora do seu departamento.

Alinea 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.

Será necessário adicionar rotas do servidor S1 até aos routers dos outros departamentos, para que seja possível haver partilha de informação entre eles.

```

root@S1:/tmp/pycore,45/67/S1.conf# route add -net 10.0.5.0/24 gw 10.0.4.1
root@S1:/tmp/pycore,45/67/S1.conf# route add -net 10.0.6.0/24 gw 10.0.4.1
root@S1:/tmp/pycore,45/67/S1.conf# route add -net 10.0.7.0/24 gw 10.0.4.1
root@S1:/tmp/pycore,45/67/S1.conf#

```

Figure 23: Adicionar novas rotas entre S1 e os restantes routers.

Alinea 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.

```

root@S1:/tmp/pycore,356/77/S1.conf# ping 10.0.5.20 -c 3
PING 10.0.5.20 (10.0.5.20) 56(84) bytes of data:
64 bytes from 10.0.5.20: icmp_seq=1 ttl=62 time=2.15 ms
64 bytes from 10.0.5.20: icmp_seq=2 ttl=62 time=1.51 ms
64 bytes from 10.0.5.20: icmp_seq=3 ttl=62 time=0.861 ms

--- 10.0.5.20 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2020ms
rtt min/avg/max/mdev = 0.861/1.510/2.155/0.528 ms
root@S1:/tmp/pycore,356/77/S1.conf# ping 10.0.6.20 -c 3
PING 10.0.6.20 (10.0.6.20) 56(84) bytes of data:
64 bytes from 10.0.6.20: icmp_seq=1 ttl=61 time=11.5 ms
64 bytes from 10.0.6.20: icmp_seq=2 ttl=61 time=27.8 ms
64 bytes from 10.0.6.20: icmp_seq=3 ttl=61 time=31.5 ms

--- 10.0.6.20 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2008ms
rtt min/avg/max/mdev = 11.583/23.674/31.558/8.681 ms
root@S1:/tmp/pycore,356/77/S1.conf# ping 10.0.7.20 -c 3
PING 10.0.7.20 (10.0.7.20) 56(84) bytes of data:
64 bytes from 10.0.7.20: icmp_seq=1 ttl=62 time=7.97 ms
64 bytes from 10.0.7.20: icmp_seq=2 ttl=62 time=11.9 ms
64 bytes from 10.0.7.20: icmp_seq=3 ttl=62 time=37.4 ms

--- 10.0.7.20 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2008ms
rtt min/avg/max/mdev = 7.973/19.102/37.414/13.048 ms
root@S1:/tmp/pycore,356/77/S1.conf#

```

Figure 24: Envio de mensagens do servidor S1 para laptops do departamento C, D e B.

Como podemos observar, após adicionar as rotas para os diversos roteadores do departamento, os seus laptops passaram a ser acessíveis ao servidor S1.

```

root@S1:/tmp/pycore,356/77/S1.conf# netstat -rn
Kernel IP routing table
Destination Gateway Genmask Flags MSS Window irtt Iface
10.0.4.0 0.0.0.0 255.255.255.0 U 0 0 0 eth0
10.0.5.0 10.0.4.1 255.255.255.0 UG 0 0 0 eth0
10.0.6.0 10.0.4.1 255.255.255.0 UG 0 0 0 eth0
10.0.7.0 10.0.4.1 255.255.255.0 UG 0 0 0 eth0

```

Figure 25: Rotas que foram adicionadas 1 S1.

2.2 Questão 2 - Definição de Sub-redes

Alinea a. - 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.

Endereço de IP: 172.94.32.0/20. Uma vez que existem 4 departamentos, será necessário definir pelo menos 4 sub-redes. Dividindo o endereço de IP, definimos $2^n - 2$ sub-redes à medida que andamos n bits para a direita. Escolhendo usar 3 bits, ficamos com $2^3 - 2 = 6$ endereços disponíveis.

Atribuição de endereços:

Sub-rede 1: 000 | default
 Sub-rede 2: 001 | 172.94.32.0/23 | Departamento A
 Sub-rede 3: 010 | 172.94.34.0/23 | livre
 Sub-rede 4: 011 | 172.94.36.0/23 | Departamento B
 Sub-rede 5: 100 | 172.94.38.0/23 | Departamento C
 Sub-rede 6: 101 | 172.94.40.0/23 | livre
 Sub-rede 7: 110 | 172.94.42.0/23 | Departamento D
 Sub-rede 8: 111 | broadcast

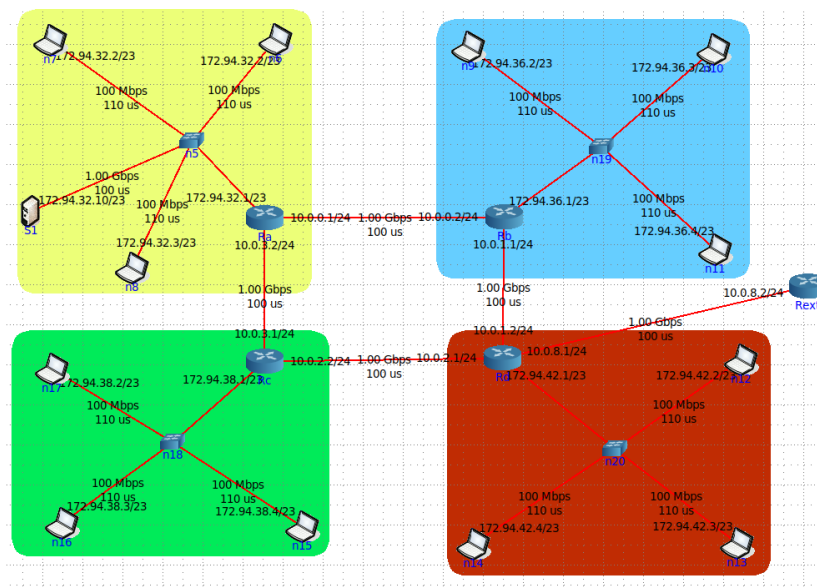


Figure 26: Novos IPs das ligações entre routers e Laptops.

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

A máscara de rede que foi usada foi /23 (255.255.254.0). O número de interfaces IP que se podem interligar em cada departamento é $2^n - 2$ sendo $n = 32 - 23$, ou seja, $2^9 - 2 = 510$.

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

```
root@n7:/tmp/pycore.44273/n7.conf# ping 172.94.38.2 -c 3
PING 172.94.38.2 (172.94.38.2) 56(84) bytes of data.
64 bytes from 172.94.38.2: icmp_seq=1 ttl=62 time=3.26 ms
64 bytes from 172.94.38.2: icmp_seq=2 ttl=62 time=69.8 ms
64 bytes from 172.94.38.2: icmp_seq=3 ttl=62 time=11.7 ms

--- 172.94.38.2 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2011ms
rtt min/avg/max/mdev = 3.265/28.306/69.868/29.594 ms
root@n7:/tmp/pycore.44273/n7.conf# ping 172.94.36.2 -c 3
PING 172.94.36.2 (172.94.36.2) 56(84) bytes of data.
64 bytes from 172.94.36.2: icmp_seq=1 ttl=62 time=1.63 ms
64 bytes from 172.94.36.2: icmp_seq=2 ttl=62 time=0.911 ms
64 bytes from 172.94.36.2: icmp_seq=3 ttl=62 time=1.44 ms

--- 172.94.36.2 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2003ms
rtt min/avg/max/mdev = 0.911/1.330/1.630/0.305 ms
root@n7:/tmp/pycore.44273/n7.conf# ping 172.94.42.2 -c 3
PING 172.94.42.2 (172.94.42.2) 56(84) bytes of data.
64 bytes from 172.94.42.2: icmp_seq=1 ttl=61 time=2.47 ms
64 bytes from 172.94.42.2: icmp_seq=2 ttl=61 time=2.08 ms
64 bytes from 172.94.42.2: icmp_seq=3 ttl=61 time=3.12 ms

--- 172.94.42.2 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2004ms
rtt min/avg/max/mdev = 2.081/2.558/3.123/0.431 ms
root@n7:/tmp/pycore.44273/n7.conf#
```

Figure 27: Envio de pacotes de um laptop do departamento A para vários de outros departamentos.

Podemos observar que existe conectividade entre os vários laptops dos departamentos, uma vez que todos os pacotes enviados foram recebidos sem qualquer problema.

3 Conclusão

Em suma, na primeira parte do trabalho, através de uma topologia CORE, analisou-se o protocolo IPv4, mais propriamente o seu comportamento e o tráfego ICMP recebido.

Já na segunda parte, aprofundamos o estudo dos protocolos IPv4, no que toca ao endereçamento e encaminhamento IP. Utilizando novamente uma topologia CORE, analisamos o encaminhamento entre quatro redes diferentes, isto somado à manipulação de IPs (subnetting).

Posto isto, este trabalho permitiu-nos aprofundar os conhecimentos, não só, em relação, ao funcionamento do CORE e às tipologias deste, mas também aos protocolos IPv4.