



UNIVERSIDADE DO MINHO
DEPARTAMENTO DE INFORMÁTICA

Trabalho 2
Redes de Computadores
Grupo 45

Catarina Machado (a81047) João Vilaça (a82339)
Ricardo Milhazes Veloso (a81919)

12 de Novembro de 2018

Conteúdo

1	TP2: Protocolo IP (Parte I)	3
1.1	Exercício 1	3
1.1.1	a) Active o wireshark ou o tcpdump no pc h1. Numa shell de h1, execute o comando traceroute -I para o endereço IP do host s4.	3
1.1.2	b) Registe e analise o tráfego ICMP enviado por h1 e o tráfego ICMP recebido como resposta. Comente os resultados face ao comportamento esperado.	4
1.1.3	c) Qual deve ser o valor inicial mínimo do campo TTL para alcançar o destino s4? Verifique na prática que a sua resposta está correta.	4
1.1.4	d) Qual o valor médio do tempo de ida-e-volta (Round-Trip Time) obtido?	4
1.2	Exercício 2	5
1.2.1	a) Qual é o endereço IP da interface ativa do seu computador?	5
1.2.2	b) Qual é o valor do campo protocolo? O que identifica?	5
1.2.3	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?	5
1.2.4	d) O datagrama IP foi fragmentado? Justifique.	5
1.2.5	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 a 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.	6
1.2.6	f) Observa algum padrão nos valores do campo de Identificação do datagrama IP e TTL??	6
1.2.7	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?	6
1.3	Exercício 3	7
1.3.1	a) Localize a primeira mensagem ICMP. Porque é que houve necessidade de fragmentar o pacote inicial?	7
1.3.2	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?	7
1.3.3	c) Imprima o segundo fragmento do datagrama IP original. Que informação do cabeçalho IP indica que não se trata do 1.º fragmento? Há mais fragmentos? O que nos permite afirmar isso?	8
1.3.4	d) Quantos fragmentos foram criados a partir do datagrama original? Como se detecta o último fragmento correspondente ao datagrama original?	8
1.3.5	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.	8
2	TP2: Protocolo IP (Parte II)	9
2.1	Exercício 1	9
2.1.1	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.	9
2.1.2	b) Tratam-se de endereços públicos ou privados? Porquê?	9
2.1.3	c) Porque razão não é atribuído um endereço IP aos switches?	9
2.1.4	d) Usando o comando ping certifique-se que existe conectividade IP entre os laptops dos vários departamentos e o servidor do departamento C (basta certificar-se da conectividade de um laptop por departamento).	10

2.1.5	e) Verifique se existe conectividade IP do router de acesso Rext para o servidor S1.	10
2.2	Exercício 2	11
2.2.1	a) Execute o comando <code>netstat -rn</code> 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 (<code>man netstat</code>).	11
2.2.2	b) Diga, justificando, se está a ser usado encaminhamento estático ou dinâmico (sugestão: analise que processos estão a correr em cada sistema).	12
2.2.3	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 C. Use o comando <code>route delete</code> para o efeito. Que implicações tem esta medida para os utilizadores da empresa que acedem ao servidor. Justifique.	12
2.2.4	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 <code>route add</code> e registe os comandos que usou.	12
2.2.5	e) Teste a nova política de encaminhamento garantindo que o servidor está novamente acessível, utilizando para o efeito o comando <code>ping</code> . Registe a nova tabela de encaminhamento do servidor.	13
2.3	Exercício 3	14
2.3.1	1) Considere que dispõe apenas do endereço de rede IP 172.XX.48.0/20, em que XX é o decimal correspondendo ao seu número de grupo (PLXX). 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.	14
2.3.2	2) Qual a máscara de rede que usou (em formato decimal)? Quantos hosts IP pode interligar em cada departamento? Justifique.	14
2.3.3	3) Garanta e verifique que conectividade IP entre as várias redes locais da organização MIEL-RC é mantida. Explique como procedeu.	15
3	Conclusão	16

1 TP2: Protocolo IP (Parte I)

1.1 Exercício 1

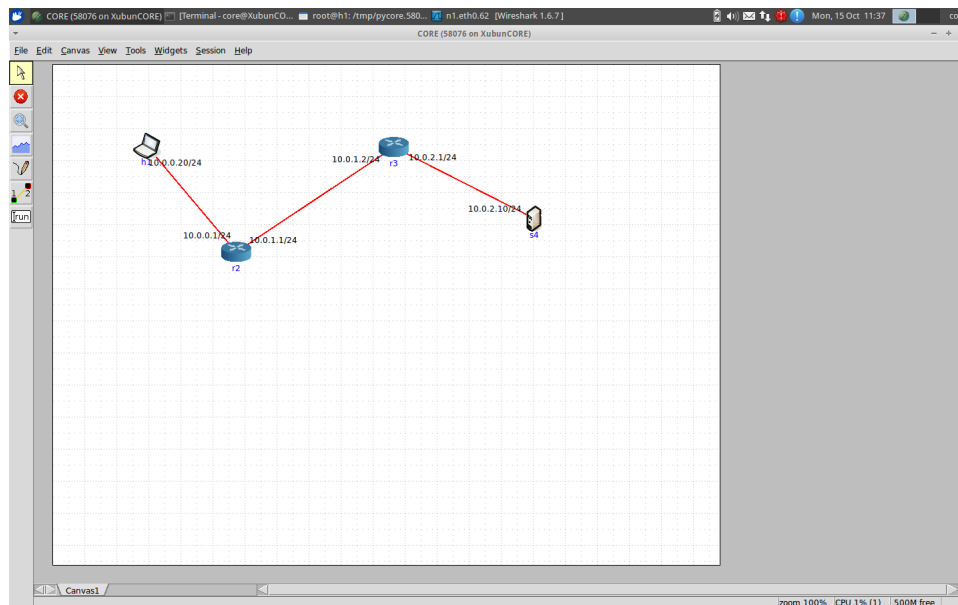


Figura 1: Topologia CORE.

- 1.1.1 a) Active o wireshark ou o tcpdump no pc h1. Numa shell de h1, execute o comando `traceroute -I` para o endereço IP do host s4.

```
root@h1:/tmp/pycore.58076/h1.conf# traceroute -I 10.0.2.10
traceroute to 10.0.2.10 (10.0.2.10), 30 hops max, 60 byte packets
 1  h0 (10.0.0.1)  0.041 ms  0.004 ms  0.003 ms
 2  10.0.1.2 (10.0.1.2)  0.022 ms  0.006 ms  0.005 ms
 3  10.0.2.10 (10.0.2.10)  0.020 ms  0.007 ms  0.006 ms
root@h1:/tmp/pycore.58076/h1.conf#
```

Figura 2: Resultados de execução do traceroute.

- 1.1.2 b) Registe e analise o tráfego ICMP enviado por h1 e o tráfego ICMP recebido como resposta. Comente os resultados face ao comportamento esperado.

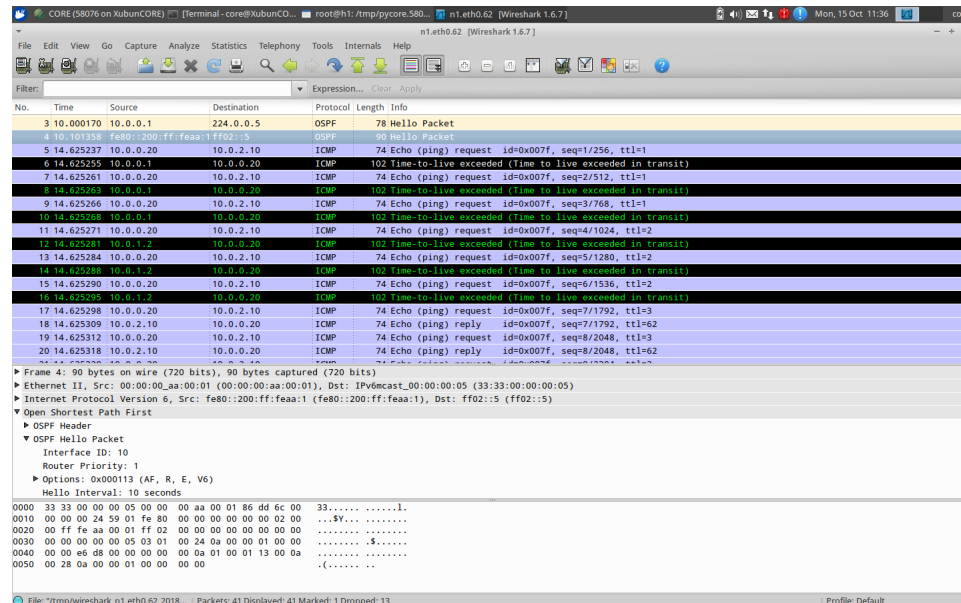


Figura 3: Análise tráfego wireshark.

Face aos resultados analisados, observamos que se verificou o comportamento esperado tendo sido enviado de h1 vários pacotes, "Echo (ping) request". Numa primeira fase, foram enviados vários pacotes com TTL = 1, descartados por r2. Em seguida, foram enviados pacotes com TTL = 2, também descartados por r3 e, por fim, foram enviados pacotes com TTL = 3 que chegaram ao seu destino, s4. Para cada pacote descartado foi recebido um outro pacote do router que o descartou, "Time-to-live exceeded". Como resposta aos pacotes que alcançaram o destino foi recebido um pacote "Echo (ping) reply".

- 1.1.3 c) Qual deve ser o valor inicial mínimo do campo TTL para alcançar o destino s4? Verifique na prática que a sua resposta está correta.

TTL = 3

17	14.625298	10.0.0.20	10.0.2.10	ICMP	74 Echo (ping) request	id=0x007f, seq=7/1792, ttl=3
18	14.625309	10.0.2.10	10.0.0.20	ICMP	74 Echo (ping) reply	id=0x007f, seq=7/1792, ttl=62
19	14.625312	10.0.0.20	10.0.2.10	ICMP	74 Echo (ping) request	id=0x007f, seq=8/2048, ttl=3
20	14.625318	10.0.2.10	10.0.0.20	ICMP	74 Echo (ping) reply	id=0x007f, seq=8/2048, ttl=62

Figura 4: Comunicação com sucesso entre as máquinas.

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

$$(0.020\text{ms} + 0.007\text{ms} + 0.006\text{ms}) / 3 = 0.011\text{ms}$$

1.2 Exercício 2

```
► Frame 42: 86 bytes on wire (688 bits), 86 bytes captured (688 bits) on interface 0
► Ethernet II, Src: Apple_71:99:c5 (8c:85:90:71:99:c5), Dst: ComdaEnt_ff:94:00 (00:d0:03:ff:94:00)
▼ Internet Protocol Version 4, Src: 172.26.30.3, 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: 72
      Identification: 0x96e2 (38626)
    ▼ 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: 4
      Protocol: ICMP (1)
      Header checksum: 0x8a3d [validation disabled]
      [Header checksum status: Unverified]
      Source: 172.26.30.3
      Destination: 193.136.9.240
    ► Internet Control Message Protocol
```

Figura 5: Cabeçalho da comunicação.

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

172.26.30.3

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

ICMP (1).

Identifica Internet Protocol.

1.2.3 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 tem 20 bytes.

Payload = 72-20 = 52 bytes.

O tamanho do campo de dados (payload) é igual ao tamanho total do pacote subtraindo o tamanho do cabeçalho.

1.2.4 d) O datagrama IP foi fragmentado? Justifique.

Não. No cabeçalho, o "Fragment offset" está definido como sendo 0, logo estamos no início do pacote e, como o campo "More Fragments" não está definido este é a última "parte" do pacote, logo ele não está fragmentado.

1.2.5 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 a 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.

1	0.000000	172.26.30.3	193.136.9.240	ICMP	86 Echo (ping) request	id=0x96d8, seq=1/256, ttl=1 (no response found!)
3	0.088613	172.26.30.3	193.136.9.240	ICMP	86 Echo (ping) request	id=0x96d8, seq=2/512, ttl=1 (no response found!)
5	0.095071	172.26.30.3	193.136.9.240	ICMP	86 Echo (ping) request	id=0x96d8, seq=3/768, ttl=1 (no response found!)
7	0.101824	172.26.30.3	193.136.9.240	ICMP	86 Echo (ping) request	id=0x96d8, seq=4/1024, ttl=2 (no response found!)
9	0.103935	172.26.30.3	193.137.16.65	DNS	83 Standard query 0x658f PTR 1.2.16.172.in-addr.arpa	
11	0.105756	172.26.30.3	193.137.16.145	DNS	83 Standard query 0x658f PTR 1.2.16.172.in-addr.arpa	
13	0.108691	172.26.30.3	193.137.16.75	DNS	83 Standard query 0x658f PTR 1.2.16.172.in-addr.arpa	
15	0.678780	172.26.30.3	157.240.1.35	TLSv1...	138 Application Data	
16	0.680354	172.26.30.3	157.240.1.35	TLSv1...	105 Application Data	
17	0.680354	172.26.30.3	157.240.1.35	TLSv1...	751 Application Data	
20	0.716845	172.26.30.3	157.240.1.35	TCP	66 53330 → 443 [ACK] Seq=797 Ack=36 Win=2685 Len=0 TSval=694081229 TSecr=2471197707	
21	0.716846	172.26.30.3	157.240.1.35	TCP	66 53330 → 443 [ACK] Seq=797 Ack=75 Win=2685 Len=0 TSval=694081229 TSecr=2471197707	
24	0.733120	172.26.30.3	157.240.1.35	TCP	66 53330 → 443 [ACK] Seq=797 Ack=348 Win=2682 Len=0 TSval=694081245 TSecr=2471197723	
25	0.733120	172.26.30.3	157.240.1.35	TCP	66 53330 → 443 [ACK] Seq=797 Ack=380 Win=2681 Len=0 TSval=694081245 TSecr=2471197723	
26	1.114834	172.26.30.3	193.136.9.240	ICMP	86 Echo (ping) request	id=0x96d8, seq=5/1280, ttl=2 (no response found!)
28	1.118095	172.26.30.3	193.136.9.240	ICMP	86 Echo (ping) request	id=0x96d8, seq=6/1536, ttl=2 (no response found!)
30	1.119619	172.26.30.3	193.136.9.240	ICMP	86 Echo (ping) request	id=0x96d8, seq=7/1792, ttl=3 (no response found!)
32	1.122059	172.26.30.3	193.137.16.65	DNS	87 Standard query 0x8f18 PTR 252.115.16.172.in-addr.arpa	
34	1.124026	172.26.30.3	193.137.16.145	DNS	87 Standard query 0x8f18 PTR 252.115.16.172.in-addr.arpa	
36	1.126430	172.26.30.3	193.137.16.75	DNS	87 Standard query 0x8f18 PTR 252.115.16.172.in-addr.arpa	
38	2.134051	172.26.30.3	193.136.9.240	ICMP	86 Echo (ping) request	id=0x96d8, seq=8/2048, ttl=3 (no response found!)
40	2.138151	172.26.30.3	193.136.9.240	ICMP	86 Echo (ping) request	id=0x96d8, seq=9/2304, ttl=3 (no response found!)
42	2.140657	172.26.30.3	193.136.9.240	ICMP	86 Echo (ping) request	id=0x96d8, seq=10/2560, ttl=4 (reply in 43)
44	2.143158	172.26.30.3	193.136.9.240	ICMP	86 Echo (ping) request	id=0x96d8, seq=11/2816, ttl=4 (reply in 45)
46	2.145305	172.26.30.3	193.136.9.240	ICMP	86 Echo (ping) request	id=0x96d8, seq=12/3072, ttl=4 (reply in 47)

Figura 6: Tráfego wireshark ordenado por endereço de fonte.

Os campos do cabeçalho IP que variam de pacote para pacote são os seguintes: TTL, Header checksum e identification.

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

Identificacao do datagrama IP: os primeiros 8 bits são iguais (0x96...).

TTL: É incrementado sequencialmente.

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

2	0.087613	172.26.254.254	172.26.30.3	ICMP	70 Time-to-live exceeded (Time to live exceeded in transit)	
4	0.094875	172.26.254.254	172.26.30.3	ICMP	70 Time-to-live exceeded (Time to live exceeded in transit)	
6	0.101665	172.26.254.254	172.26.30.3	ICMP	70 Time-to-live exceeded (Time to live exceeded in transit)	
8	0.103291	172.16.2.1	172.26.30.3	ICMP	70 Time-to-live exceeded (Time to live exceeded in transit)	
10	0.105548	193.137.16.65	172.26.30.3	DNS	83 Standard query response 0x658f Refused PTR 1.2.16.172.in-addr.arpa	
12	0.108451	193.137.16.145	172.26.30.3	DNS	83 Standard query response 0x658f Refused PTR 1.2.16.172.in-addr.arpa	
14	0.110327	193.137.16.75	172.26.30.3	DNS	83 Standard query response 0x658f Refused PTR 1.2.16.172.in-addr.arpa	
18	0.716773	157.240.1.35	172.26.30.3	TLSv1...	101 Application Data	
19	0.716776	157.240.1.35	172.26.30.3	TLSv1...	105 Application Data	
22	0.733050	157.240.1.35	172.26.30.3	TLSv1...	339 Application Data	
23	0.733054	157.240.1.35	172.26.30.3	TLSv1...	98 Application Data	
27	1.117965	172.16.2.1	172.26.30.3	ICMP	70 Time-to-live exceeded (Time to live exceeded in transit)	
29	1.119534	172.16.2.1	172.26.30.3	ICMP	70 Time-to-live exceeded (Time to live exceeded in transit)	
31	1.121424	172.16.115.252	172.26.30.3	ICMP	70 Time-to-live exceeded (Time to live exceeded in transit)	
33	1.123851	193.137.16.65	172.26.30.3	DNS	87 Standard query response 0x8f18 Refused PTR 252.115.16.172.in-addr.arpa	
35	1.126312	193.137.16.145	172.26.30.3	DNS	87 Standard query response 0x8f18 Refused PTR 252.115.16.172.in-addr.arpa	
37	1.127861	193.137.16.75	172.26.30.3	DNS	87 Standard query response 0x8f18 Refused PTR 252.115.16.172.in-addr.arpa	
39	2.137996	172.16.115.252	172.26.30.3	ICMP	70 Time-to-live exceeded (Time to live exceeded in transit)	
41	2.140506	172.16.115.252	172.26.30.3	ICMP	70 Time-to-live exceeded (Time to live exceeded in transit)	
43	2.142515	193.136.9.240	172.26.30.3	ICMP	86 Echo (ping) reply	id=0x96d8, seq=10/2560, ttl=61 (request in 42)
45	2.145122	193.136.9.240	172.26.30.3	ICMP	86 Echo (ping) reply	id=0x96d8, seq=11/2816, ttl=61 (request in 44)
47	2.147863	193.136.9.240	172.26.30.3	ICMP	86 Echo (ping) reply	id=0x96d8, seq=12/3072, ttl=61 (request in 46)

Figura 7: Tráfego wireshark ordenado por endereço de destino.

O valor do campo TTL é 61.

Sim, esse valor permanece constante para todas as mensagens de resposta ICMP TTL exceeded enviados ao nosso host. O TTL predefinido (hardcoded) pelo destino é 64 (garante que o pacote chega da origem ao destino). No final, quando o pacote chega ao seu destino como passou por 3 routers intermediários o valor foi decrementado 3 vezes, daí o TTL ser 61.

1.3 Exercício 3

35	2.436016	172.26.30.3	193.136.9.240	IPv4	1514	Fragmented IP protocol (proto:ICMP 1, offset=0, ID=96c0) [Reassembled in #37]
36	2.437621	172.26.30.3	193.136.9.240	IPv4	1514	Fragmented IP protocol (proto:ICMP 1, offset=1480, ID=96c0) [Reassembled in #37]
37	2.437621	172.26.30.3	193.136.9.240	ICMP	599	Echo (ping) request id=b96bf, seq=1250, ttl=1 (no response found)
38	2.501616	172.26.124.254	172.26.30.3	ICMP	70	Time-to-live exceeded (time to live exceeded in transit)
39	2.502144	172.26.30.3	193.136.9.240	IPv4	1514	Fragmented IP protocol (proto:ICMP 1, offset=0, ID=96c1) [Reassembled in #41]
40	2.502144	172.26.30.3	193.136.9.240	IPv4	1514	Fragmented IP protocol (proto:ICMP 1, offset=1480, ID=96c1) [Reassembled in #41]
41	2.502144	172.26.30.3	193.136.9.240	ICMP	599	Echo (ping) request id=b96bf, seq=27512, ttl=1 (no response found)
42	2.518076	172.26.124.254	172.26.30.3	ICMP	70	Time-to-live exceeded (time to live exceeded in transit)
43	2.518086	172.26.30.3	193.136.9.240	IPv4	1514	Fragmented IP protocol (proto:ICMP 1, offset=0, ID=96c2) [Reassembled in #45]
44	2.518077	172.26.30.3	193.136.9.240	IPv4	1514	Fragmented IP protocol (proto:ICMP 1, offset=1480, ID=96c2) [Reassembled in #45]
45	2.518077	172.26.30.3	193.136.9.240	ICMP	599	Echo (ping) request id=b96bf, seq=3768, ttl=1 (no response found)
46	2.518138	172.26.124.254	172.26.30.3	ICMP	70	Time-to-live exceeded (time to live exceeded in transit)
47	2.518292	172.26.30.3	193.136.9.240	IPv4	1514	Fragmented IP protocol (proto:ICMP 1, offset=0, ID=96c3) [Reassembled in #49]
48	2.518293	172.26.30.3	193.136.9.240	IPv4	1514	Fragmented IP protocol (proto:ICMP 1, offset=1480, ID=96c3) [Reassembled in #49]
49	2.518293	172.26.30.3	193.136.9.240	ICMP	599	Echo (ping) request id=b96bf, seq=47824, ttl=2 (no response found)
50	2.518504	172.26.30.3	193.136.9.240	ICMP	70	Time-to-live exceeded (time to live exceeded in transit)
51	2.518517	172.26.30.3	193.136.9.240	IPv4	1514	Fragmented IP protocol (proto:ICMP 1, offset=0, ID=96c4) [Reassembled in #53]
52	2.518518	172.26.30.3	193.136.9.240	IPv4	1514	Fragmented IP protocol (proto:ICMP 1, offset=1480, ID=96c4) [Reassembled in #53]
53	2.518518	172.26.30.3	193.136.9.240	ICMP	599	Echo (ping) request id=b96bf, seq=57120, ttl=2 (no response found)
54	2.518530	172.26.30.3	193.136.9.240	ICMP	70	Time-to-live exceeded (time to live exceeded in transit)
55	2.517780	172.26.30.3	193.136.9.240	IPv4	1514	Fragmented IP protocol (proto:ICMP 1, offset=0, ID=96c5) [Reassembled in #57]
56	2.517781	172.26.30.3	193.136.9.240	IPv4	1514	Fragmented IP protocol (proto:ICMP 1, offset=1480, ID=96c5) [Reassembled in #57]
57	2.517782	172.26.30.3	193.136.9.240	ICMP	599	Echo (ping) request id=b96bf, seq=67152, ttl=2 (no response found)
58	2.518473	172.16.1.1	172.26.30.3	ICMP	70	Time-to-live exceeded (time to live exceeded in transit)
59	2.518876	172.26.30.3	193.136.9.240	IPv4	1514	Fragmented IP protocol (proto:ICMP 1, offset=0, ID=96c6) [Reassembled in #61]
60	2.518877	172.26.30.3	193.136.9.240	IPv4	1514	Fragmented IP protocol (proto:ICMP 1, offset=1480, ID=96c6) [Reassembled in #61]
61	2.519879	172.26.30.3	193.136.9.240	ICMP	599	Echo (ping) request id=b96bf, seq=77192, ttl=3 (no response found)
62	2.522440	172.16.115.252	172.26.30.3	ICMP	70	Time-to-live exceeded (time to live exceeded in transit)
63	2.522108	172.26.30.3	193.136.9.240	IPv4	1514	Fragmented IP protocol (proto:ICMP 1, offset=0, ID=96c7) [Reassembled in #65]
64	2.523189	172.26.30.3	193.136.9.240	IPv4	1514	Fragmented IP protocol (proto:ICMP 1, offset=1480, ID=96c7) [Reassembled in #65]
65	2.523110	172.26.30.3	193.136.9.240	ICMP	599	Echo (ping) request id=b96bf, seq=87200, ttl=3 (no response found)
66	2.525084	172.16.115.252	172.26.30.3	ICMP	70	Time-to-live exceeded (time to live exceeded in transit)
67	2.525210	172.26.30.3	193.136.9.240	IPv4	1514	Fragmented IP protocol (proto:ICMP 1, offset=0, ID=96c8) [Reassembled in #69]
68	2.525211	172.26.30.3	193.136.9.240	IPv4	1514	Fragmented IP protocol (proto:ICMP 1, offset=1480, ID=96c8) [Reassembled in #69]
69	2.525212	172.26.30.3	193.136.9.240	ICMP	599	Echo (ping) request id=b96bf, seq=97280, ttl=3 (no response found)
70	2.527210	172.16.115.252	172.26.30.3	ICMP	70	Time-to-live exceeded (time to live exceeded in transit)
71	2.527410	172.26.30.3	193.136.9.240	IPv4	1514	Fragmented IP protocol (proto:ICMP 1, offset=0, ID=96c9) [Reassembled in #73]
72	2.527411	172.26.30.3	193.136.9.240	IPv4	1514	Fragmented IP protocol (proto:ICMP 1, offset=1480, ID=96c9) [Reassembled in #73]
73	2.527412	172.26.30.3	193.136.9.240	ICMP	599	Echo (ping) request id=b96bf, seq=107296, ttl=4 (reply in 80)
74	2.531143	193.136.9.240	172.26.30.3	IPv4	1514	Fragmented IP protocol (proto:ICMP 1, offset=0, ID=a5d1) [Reassembled in #76]
75	2.531148	193.136.9.240	172.26.30.3	IPv4	599	Fragmented IP protocol (proto:ICMP 1, offset=2960, ID=a5d2) [Reassembled in #76]
76	2.531149	193.136.9.240	172.26.30.3	ICMP	1514	Echo (ping) reply id=b96bf, seq=107296, ttl=4 (request in 73)
77	2.531150	172.26.30.3	193.136.9.240	ICMP	1514	Fragmented IP protocol (proto:ICMP 1, offset=0, ID=a5d3) [Reassembled in #79]
78	2.531858	172.26.30.3	193.136.9.240	IPv4	1514	Fragmented IP protocol (proto:ICMP 1, offset=1480, ID=96ca) [Reassembled in #79]
79	2.531859	172.26.30.3	193.136.9.240	ICMP	599	Echo (ping) request id=b96bf, seq=117216, ttl=4 (reply in 82)
80	2.534069	193.136.9.240	172.26.30.3	IPv4	1514	Fragmented IP protocol (proto:ICMP 1, offset=0, ID=a5d4) [Reassembled in #82]
81	2.534072	193.136.9.240	172.26.30.3	IPv4	599	Fragmented IP protocol (proto:ICMP 1, offset=2960, ID=a5d4) [Reassembled in #82]
82	2.534073	193.136.9.240	172.26.30.3	ICMP	1514	Echo (ping) reply id=b96bf, seq=117216, ttl=4 (request in 79)
83	2.535580	172.26.30.3	193.136.9.240	ICMP	1514	Fragmented IP protocol (proto:ICMP 1, offset=0, ID=96cb) [Reassembled in #85]
84	2.535101	172.26.30.3	193.136.9.240	IPv4	1514	Fragmented IP protocol (proto:ICMP 1, offset=1480, ID=96cb) [Reassembled in #85]
85	2.535101	172.26.30.3	193.136.9.240	ICMP	599	Echo (ping) request id=b96bf, seq=1273072, ttl=4 (reply in 88)
86	2.538026	193.136.9.240	172.26.30.3	IPv4	1514	Fragmented IP protocol (proto:ICMP 1, offset=0, ID=a5d5) [Reassembled in #88]
87	2.538031	193.136.9.240	172.26.30.3	IPv4	599	Fragmented IP protocol (proto:ICMP 1, offset=2960, ID=a5d5) [Reassembled in #88]
88	2.538032	193.136.9.240	172.26.30.3	ICMP	1514	Echo (ping) reply id=b96bf, seq=1273072, ttl=4 (request in 85)

Figura 8: Fragmentos do datagrama IP.

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

Mensagem com o número 35 (Figura 8).

Porque o tamanho do PDU é maior do que o máximo permitido pelo protocolo IPv4.

1.3.2 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 35: 1514 bytes on wire (12112 bits), 1514 bytes captured (12112 bits) on interface 0
► Ethernet II, Src: Apple_71:99:c5 (8c:85:90:71:99:c5), Dst: ComdaEnt_ff:94:00 (00:d0:03:ff:94:00)
▼ Internet Protocol Version 4, Src: 172.26.30.3, 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: 0x96c0 (38592)
  ▼ 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: 0x67cb [validation disabled]
    [Header checksum status: Unverified]
    Source: 172.26.30.3
    Destination: 193.136.9.240
    Reassembled IPv4 in frame: 37
  ► Data (1480 bytes)

```

Figura 9: Cabeçalho do primeiro fragmento do datagrama IP.

No campo das Flags, o bit correspondente a "More fragments" ter o valor 1 (Set), o que indica que o datagrama foi fragmentado.

É o primeiro fragmento porque o offset é 0 ("Fragment offset").

O tamanho do datagrama IP é 1500 bytes ("Total Length").

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

```
► Frame 36: 1514 bytes on wire (12112 bits), 1514 bytes captured (12112 bits) on interface 0
► Ethernet II, Src: Apple_71:99:c5 (8c:85:90:71:99:c5), Dst: ComdaEnt_ff:94:00 (00:d0:03:ff:94:00)
▼ Internet Protocol Version 4, Src: 172.26.30.3, 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: 0x96c0 (38592)
  ▼ 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
  Protocol: ICMP (1)
  Header checksum: 0x6712 [validation disabled]
  [Header checksum status: Unverified]
  Source: 172.26.30.3
  Destination: 193.136.9.240
  Reassembled IPv4 in frame: 37
► Data (1480 bytes)
```

Figura 10: Cabeçalho do segundo fragmento do datagrama IP.

Não é o 1.º fragmento do pacote porque o offset é diferente de 0 ("Fragment offset" = 185).

Sim, há mais fragmentos porque o bit correspondente a "More fragments" é 1.

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

```
► Frame 37: 599 bytes on wire (4792 bits), 599 bytes captured (4792 bits) on interface 0
► Ethernet II, Src: Apple_71:99:c5 (8c:85:90:71:99:c5), Dst: ComdaEnt_ff:94:00 (00:d0:03:ff:94:00)
▼ Internet Protocol Version 4, Src: 172.26.30.3, 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: 585
  Identification: 0x96c0 (38592)
  ▼ 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
  ► Time to live: 1
  Protocol: ICMP (1)
  Header checksum: 0x89ec [validation disabled]
  [Header checksum status: Unverified]
  Source: 172.26.30.3
  Destination: 193.136.9.240
  ► [3 IPv4 Fragments (3525 bytes): #35(1480), #36(1480), #37(565)]
► Internet Control Message Protocol
```

Figura 11: Cabeçalho do terceiro fragmento do datagrama IP.

Foram criados 3 fragmentos, o número 35, 36 e o 37 (Figura 8).

O bit correspondente a "More fragments" é 0 (Not Set), ou seja, não há mais fragmentos.

- 1.3.5 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.**

Entre os cabeçalhos IP dos 3 fragmentos varia o "More fragments" e o "Fragment offset".

Ordena-se os fragmentos por ordem crescente do fragment offset até que o bit de more fragment seja 0, isto é, estejamos no último fragmento.

2 TP2: Protocolo IP (Parte II)

2.1 Exercício 1

- 2.1.1 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.

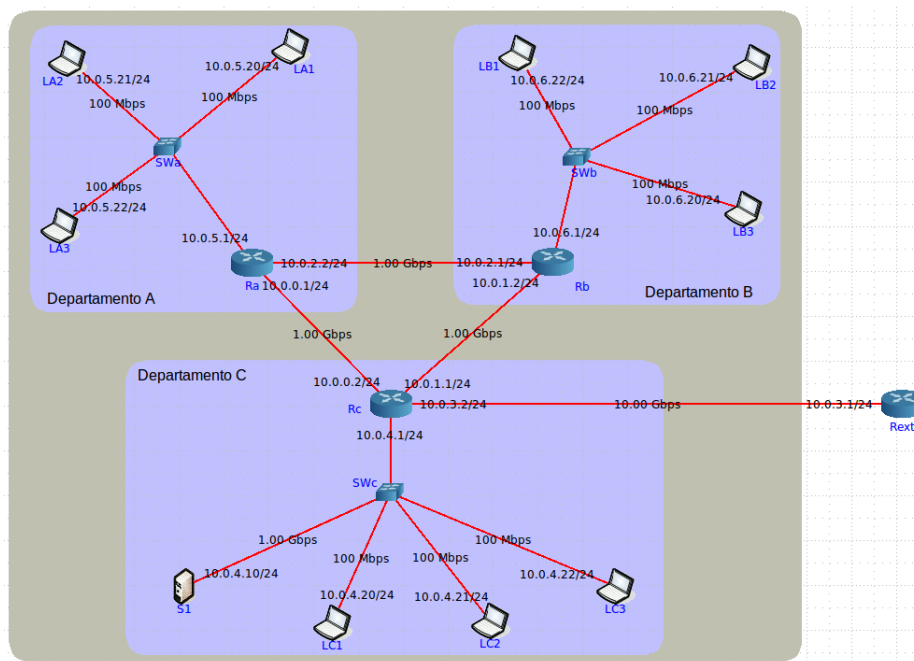


Figura 12: Topologia CORE.

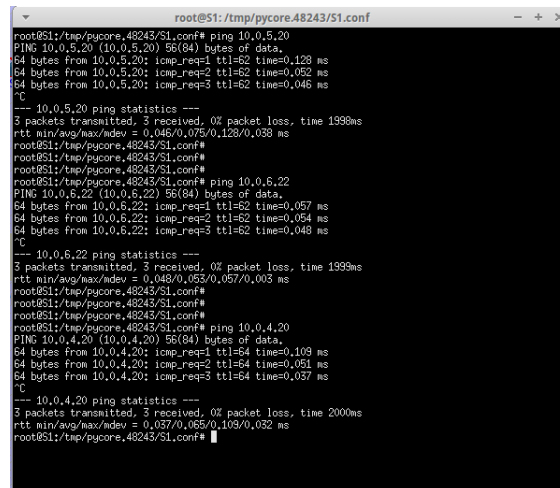
- 2.1.2 b) Tratam-se de endereços públicos ou privados? Porquê?

São endereços privados porque utilizam como prefixo um dos blocos reservados a endereços privados na norma RFC 1918 pela IANA ("10.0.0.0 - 10.255.255.255 (10/8 prefix)").

- 2.1.3 c) Porque razão não é atribuído um endereço IP aos switches?

Os switches não têm um IP atribuído porque operam numa camada abaixo (Layer 2).

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

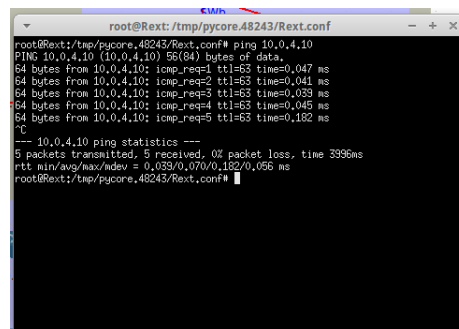


```
root@S1:/tmp/pycore.48243/S1.conf# ping 10.0.5.20
PING 10.0.5.20 (10.0.5.20) 56(84) bytes of data:
64 bytes from 10.0.5.20: icmp_req=1 ttl=62 time=0.128 ms
64 bytes from 10.0.5.20: icmp_req=2 ttl=62 time=0.052 ms
64 bytes from 10.0.5.20: icmp_req=3 ttl=62 time=0.046 ms
--- 10.0.5.20 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 1998ms
rtt min/avg/max/ndev = 0.046/0.075/0.128/0.038 ms
root@S1:/tmp/pycore.48243/S1.conf#
root@S1:/tmp/pycore.48243/S1.conf# ping 10.0.6.22
PING 10.0.6.22 (10.0.6.22) 56(84) bytes of data:
64 bytes from 10.0.6.22: icmp_req=1 ttl=62 time=0.057 ms
64 bytes from 10.0.6.22: icmp_req=2 ttl=62 time=0.054 ms
64 bytes from 10.0.6.22: icmp_req=3 ttl=62 time=0.048 ms
--- 10.0.6.22 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 1999ms
rtt min/avg/max/ndev = 0.048/0.053/0.057/0.003 ms
root@S1:/tmp/pycore.48243/S1.conf#
root@S1:/tmp/pycore.48243/S1.conf# ping 10.0.4.20
PING 10.0.4.20 (10.0.4.20) 56(84) bytes of data:
64 bytes from 10.0.4.20: icmp_req=1 ttl=64 time=0.109 ms
64 bytes from 10.0.4.20: icmp_req=2 ttl=64 time=0.051 ms
64 bytes from 10.0.4.20: icmp_req=3 ttl=64 time=0.037 ms
--- 10.0.4.20 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2000ms
rtt min/avg/max/ndev = 0.037/0.065/0.109/0.032 ms
root@S1:/tmp/pycore.48243/S1.conf#
```

Figura 13: Ping para laptops a partir de S1 (servidor do departamento C).

Observando a Figura 13 nota-se que existe conectividade IP entre os laptops dos vários departamentos e o servidor do departamento C.

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



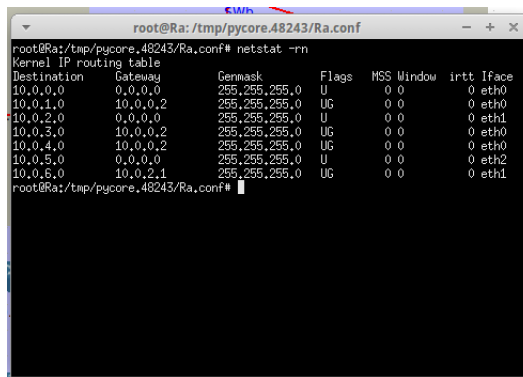
```
root@Rext:/tmp/pycore.48243/Rext.conf# ping 10.0.4.10
PING 10.0.4.10 (10.0.4.10) 56(84) bytes of data:
64 bytes from 10.0.4.10: icmp_req=1 ttl=63 time=0.047 ms
64 bytes from 10.0.4.10: icmp_req=2 ttl=63 time=0.041 ms
64 bytes from 10.0.4.10: icmp_req=3 ttl=63 time=0.039 ms
64 bytes from 10.0.4.10: icmp_req=4 ttl=63 time=0.046 ms
64 bytes from 10.0.4.10: icmp_req=5 ttl=63 time=0.182 ms
--- 10.0.4.10 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 3396ms
rtt min/avg/max/ndev = 0.039/0.070/0.182/0.066 ms
root@Rext:/tmp/pycore.48243/Rext.conf#
```

Figura 14: Ping para S1 a partir de Rext.

Observando a Figura 14 verifica-se que existe conectividade IP do router de acesso (Rext) para o servidor.

2.2 Exercício 2

- 2.2.1 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@Ra:/tmp/pycore.48243/Ra.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 10.0.0.2 255.255.255.0 UG 0 0 0 eth0
10.0.2.0 0.0.0.0 255.255.255.0 U 0 0 0 eth1
10.0.3.0 10.0.0.2 255.255.255.0 UG 0 0 0 eth0
10.0.4.0 10.0.0.2 255.255.255.0 UG 0 0 0 eth0
10.0.5.0 0.0.0.0 255.255.255.0 U 0 0 0 eth2
10.0.6.0 10.0.2.1 255.255.255.0 UG 0 0 0 eth1
```

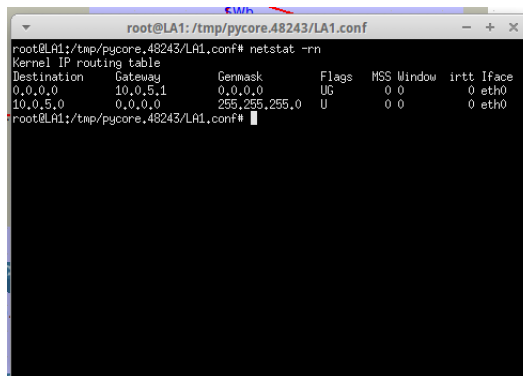
Figura 15: Tabela de encaminhamento do router do departamento A.

Analisando as entradas da tabela da Figura 15:

Forma de leitura de cada linha: Um datagrama destinado à rede "Destination" será entregue na interface de endereço "Gateway" saindo pela interface local "Iface". É obrigatório a máscara ser mencionada (uma vez que estamos a utilizar Classless), que neste caso será sempre 24 (255.255.255.0).

Analisando agora a primeira e a segunda entrada da tabela verificamos que diferem no Gateway (a primeira tem o endereço default, ao contrário da segunda que tem Gateway definido). Isto acontece porque se os dois endereços estão ligados diretamente o Gateway não precisa de estar definido, enquanto que se não estiverem ligados diretamente é necessário saber qual é o próximo salto.

As flags apenas acrescentam informação adicional, sendo U (a route é válida) utilizada quando o Gateway não está definido, e UG (a route é válida mas redireciona para um gateway e não diretamente a uma rede ou host) caso contrário.



```
root@LA1:/tmp/pycore.48243/LA1.conf# netstat -rn
Kernel IP routing table
Destination Gateway Genmask Flags MSS Window irtt Iface
0.0.0.0 10.0.5.1 0.0.0.0 UG 0 0 0 eth0
10.0.5.0 0.0.0.0 255.255.255.0 U 0 0 0 eth0
```

Figura 16: Tabela de encaminhamento de um laptop do departamento A.

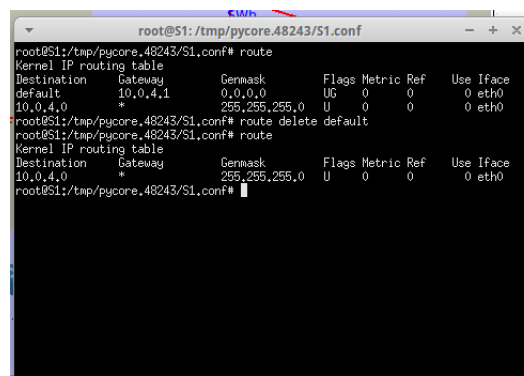
Analisando as duas entradas da tabela da Figura 16:

Como a máscara é 0 o destino 0.0.0.0 identifica todas as redes possíveis, ou seja, caso o tráfego não seja para um host de 10.0.5.0 ele é redirecionado, por defeito, para o Gateway 10.0.5.1.

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

É usado um encaminhamento dinâmico porque as rotas são definidas automaticamente através da troca de informação de routing entre routers.

- 2.2.3 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 C. Use o comando `route delete` para o efeito. Que implicações tem esta medida para os utilizadores da empresa que acedem ao servidor. Justifique.

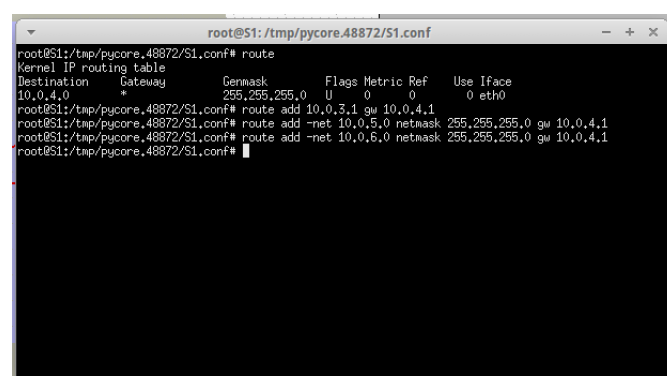


```
root@S1:/tmp/pycore.48243/S1.conf# route
Kernel IP routing table
Destination    Gateway         Genmask         Flags Metric Ref    Use Iface
default        10.0.4.1        0.0.0.0         UG    0     0          0 eth0
10.0.4.0       *              255.255.255.0   U     0     0          0 eth0
root@S1:/tmp/pycore.48243/S1.conf# route delete default
root@S1:/tmp/pycore.48243/S1.conf# route
Kernel IP routing table
Destination    Gateway         Genmask         Flags Metric Ref    Use Iface
10.0.4.0       *              255.255.255.0   U     0     0          0 eth0
root@S1:/tmp/pycore.48243/S1.conf#
```

Figura 17: Tabela de encaminhamento depois de `delete route` por defeito.

O servidor S1 perde a conectividade com todos os hosts que não pertencem à sua rede local (Departamento C) isto porque, removendo a rota por defeito, o Servidor S1 não tem definida a rota de envio de tráfego para redes não locais. Assim, os utilizadores da empresa conseguem enviar dados para o servidor mas não conseguem receber.

- 2.2.4 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.



```
root@S1:/tmp/pycore.48872/S1.conf# route
Kernel IP routing table
Destination    Gateway         Genmask         Flags Metric Ref    Use Iface
10.0.4.0       *              255.255.255.0   U     0     0          0 eth0
root@S1:/tmp/pycore.48872/S1.conf# route add 10.0.3.1 gw 10.0.4.1
root@S1:/tmp/pycore.48872/S1.conf# route add -net 10.0.5.0 netmask 255.255.255.0 gw 10.0.4.1
root@S1:/tmp/pycore.48872/S1.conf# route add -net 10.0.6.0 netmask 255.255.255.0 gw 10.0.4.1
root@S1:/tmp/pycore.48872/S1.conf#
```

Figura 18: Adição de static routes.

A partir do S1:
`route add 10.0.3.1 gw 10.0.4.1`

```
route add -net 10.0.5.0 netmask 255.255.255.0 gw 10.0.4.1
route add -net 10.0.6.0 netmask 255.255.255.0 gw 10.0.4.1
```

2.2.5 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.48872/S1.conf# ping 10.0.5.21
PING 10.0.5.21 (10.0.5.21) 56(84) bytes of data:
64 bytes from 10.0.5.21: icmp_req=1 ttl=62 time=0.044 ms
64 bytes from 10.0.5.21: icmp_req=2 ttl=62 time=0.047 ms
64 bytes from 10.0.5.21: icmp_req=3 ttl=62 time=0.047 ms
^C
--- 10.0.5.21 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 1998ms
rtt min/avg/max/ndev = 0.044/0.046/0.047/0.001 ms
root@S1:/tmp/pycore.48872/S1.conf# ping 10.0.6.22
PING 10.0.6.22 (10.0.6.22) 56(84) bytes of data:
64 bytes from 10.0.6.22: icmp_req=1 ttl=62 time=0.049 ms
64 bytes from 10.0.6.22: icmp_req=2 ttl=62 time=0.047 ms
64 bytes from 10.0.6.22: icmp_req=3 ttl=62 time=0.073 ms
^C
--- 10.0.6.22 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 1998ms
rtt min/avg/max/ndev = 0.047/0.056/0.073/0.013 ms
root@S1:/tmp/pycore.48872/S1.conf# ping 10.0.3.1
PING 10.0.3.1 (10.0.3.1) 56(84) bytes of data:
64 bytes from 10.0.3.1: icmp_req=1 ttl=63 time=0.839 ms
64 bytes from 10.0.3.1: icmp_req=2 ttl=63 time=0.046 ms
64 bytes from 10.0.3.1: icmp_req=3 ttl=63 time=0.042 ms
^C
--- 10.0.3.1 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2000ms
rtt min/avg/max/ndev = 0.042/0.309/0.839/0.374 ms
root@S1:/tmp/pycore.48872/S1.conf# route
Kernel IP routing table
Destination Gateway Genmask Flags Metric Ref Use Iface
10.0.3.1 10.0.4.1 255.255.255.255 UGH 0 0 0 eth0
10.0.4.0 * 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
root@S1:/tmp/pycore.48872/S1.conf#
```

Figura 19: Teste de conectividade e tabela de encaminhamento.

Através da Figura 19 verificamos, através do comando ping, que o servidor S1 está novamente acessível.

2.3 Exercício 3

- 2.3.1 1) Considere que dispõe apenas do endereço de rede IP 172.XX.48.0/20, em que XX é o decimal correspondendo ao seu número de grupo (PLXX). 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.

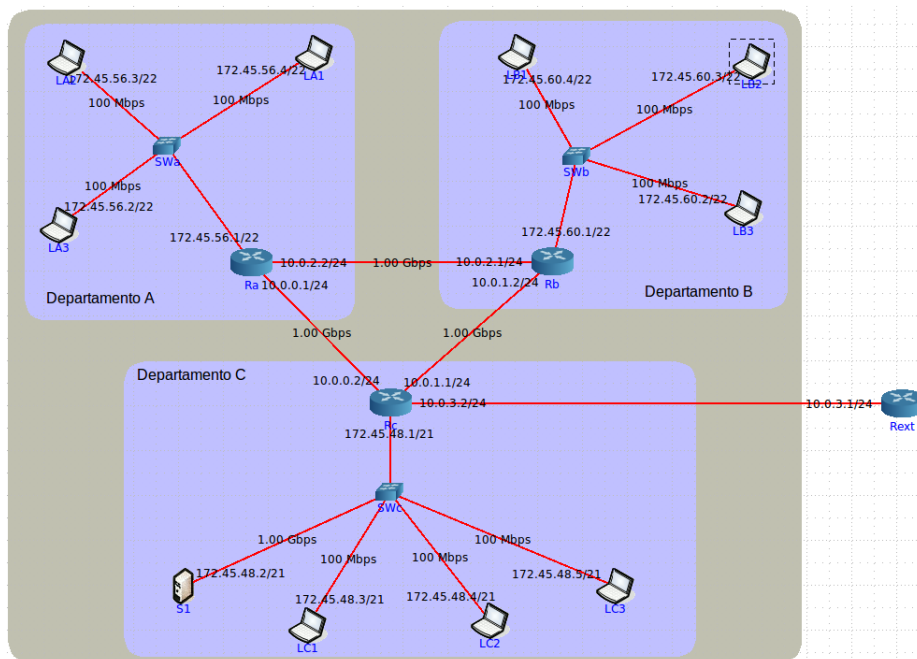


Figura 20: Divisão de endereços 172.45.48.0/20

Nós dividimos o endereço de rede dado em 3, um /21 e dois /22. A escolha foi feita de forma a obter o mínimo de perda de capacidade para hosts que queiram usufruir da rede.

- 2.3.2 2) Qual a máscara de rede que usou (em formato decimal)? Quantos hosts IP pode interligar em cada departamento? Justifique.

Para /21: 255.255.248.0

Para /22: 255.255.252.0

O número de hosts IP para cada departamento é 2 elevado ao número de bits não utilizados como identificador de rede, menos 2, o endereço de broadcast e o endereço universal da rede.

Departamento A e B:

$$2^{10} - 2 = 1022$$

Departamento C:

$$2^{11} - 2 = 2046$$

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

```
root@Ra:/tmp/pycore.48876/Ra.conf# ping 172.45.48.2
PING 172.45.48.2 (172.45.48.2) 56(84) bytes of data:
64 bytes from 172.45.48.2: icmp_req=1 ttl=63 time=0.088 ms
64 bytes from 172.45.48.2: icmp_req=2 ttl=63 time=0.048 ms
64 bytes from 172.45.48.2: icmp_req=3 ttl=63 time=0.042 ms
^C
--- 172.45.48.2 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 1938ms
rtt min/avg/max/ndev = 0.042/0.053/0.088/0.021 ms
root@Ra:/tmp/pycore.48876/Ra.conf#
root@Ra:/tmp/pycore.48876/Ra.conf# ping 172.45.60.3
PING 172.45.60.3 (172.45.60.3) 56(84) bytes of data:
64 bytes from 172.45.60.3: icmp_req=1 ttl=63 time=0.067 ms
64 bytes from 172.45.60.3: icmp_req=2 ttl=63 time=0.039 ms
64 bytes from 172.45.60.3: icmp_req=3 ttl=63 time=0.039 ms
^C
--- 172.45.60.3 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 1998ms
rtt min/avg/max/ndev = 0.039/0.048/0.067/0.014 ms
root@Ra:/tmp/pycore.48876/Ra.conf#
root@Ra:/tmp/pycore.48876/Ra.conf# ping 10.0.3.1
PING 10.0.3.1 (10.0.3.1) 56(84) bytes of data:
64 bytes from 10.0.3.1: icmp_req=1 ttl=63 time=0.063 ms
64 bytes from 10.0.3.1: icmp_req=2 ttl=63 time=0.042 ms
64 bytes from 10.0.3.1: icmp_req=3 ttl=63 time=0.188 ms
^C
--- 10.0.3.1 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2001ms
rtt min/avg/max/ndev = 0.042/0.097/0.188/0.065 ms
root@Ra:/tmp/pycore.48876/Ra.conf#
```

Figura 21: Conetividade a partir do Departamento A

```
root@LB2:/tmp/pycore.48876/LB2.conf# ping 172.45.48.3
PING 172.45.48.3 (172.45.48.3) 56(84) bytes of data:
64 bytes from 172.45.48.3: icmp_req=1 ttl=62 time=0.103 ms
64 bytes from 172.45.48.3: icmp_req=2 ttl=62 time=0.254 ms
64 bytes from 172.45.48.3: icmp_req=3 ttl=62 time=0.050 ms
^C
--- 172.45.48.3 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2000ms
rtt min/avg/max/ndev = 0.050/0.135/0.254/0.087 ms
root@LB2:/tmp/pycore.48876/LB2.conf#
root@LB2:/tmp/pycore.48876/LB2.conf# ping 10.0.3.1
PING 10.0.3.1 (10.0.3.1) 56(84) bytes of data:
64 bytes from 10.0.3.1: icmp_req=1 ttl=62 time=0.054 ms
64 bytes from 10.0.3.1: icmp_req=2 ttl=62 time=0.052 ms
64 bytes from 10.0.3.1: icmp_req=3 ttl=62 time=0.047 ms
^C
--- 10.0.3.1 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 1998ms
rtt min/avg/max/ndev = 0.047/0.051/0.054/0.003 ms
root@LB2:/tmp/pycore.48876/LB2.conf#
```

Figura 22: Conetividade a partir do Departamento B

```
root@LC3:/tmp/pycore.48876/LC3.conf# ping 10.0.3.1
PING 10.0.3.1 (10.0.3.1) 56(84) bytes of data:
64 bytes from 10.0.3.1: icmp_req=1 ttl=63 time=0.072 ms
64 bytes from 10.0.3.1: icmp_req=2 ttl=63 time=0.038 ms
64 bytes from 10.0.3.1: icmp_req=3 ttl=63 time=0.042 ms
^C
--- 10.0.3.1 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 1998ms
rtt min/avg/max/ndev = 0.038/0.050/0.072/0.017 ms
root@LC3:/tmp/pycore.48876/LC3.conf#
```

Figura 23: Conetividade a partir do Departamento C

Teste de conectividade dos vários departamentos entre si e cada um com o exterior.

3 Conclusão

Este trabalho prático serviu de complemento às aulas teóricas e ajudou a consolidar a matéria lecionada nas mesmas.

Através da máquina virtual disponibilizada, tivemos a oportunidade de desenvolver as nossas capacidades de construção de Topologias CORE. Foi bastante esclarecedor para a nossa eficiente aprendizagem poder construir um caso virtual de tráfego ICMP.

Relativamente ao capítulo de IP: Internet Protocol, ficamos a perceber melhor como é que o TTL funciona uma vez que tivemos a oportunidade de analisar vários exemplos (Secção 1.1.2, 1.1.3 e 1.2.7), assim como a importância de tentar ter o menor TTL possível (para se não houver conectividade o pacote não andar em loop), mas que consiga chegar ao destino desejado.

Quanto ao formato de um datagrama IP, a par das aulas teóricas, identificamos os dois campos constituintes no datagrama da Figura 5: campo de dados (payload) e cabeçalho (Secção 1.2.3).

Averiguámos também a fragmentação de datagramas (Secção 1.2.4 e 1.3), e consequentemente, fomos confrontados com a importância do campo relativo à fragmentation, com respetivas flags (reserved bit, don't fragment, more fragments e fragment offset) (IPv4).

Relembramos ainda a definição de endereços públicos e privados na Secção 2.1.2 e de switches, na Secção 2.1.3.

Analisamos tabelas de encaminhamento (Secção 2.3.3) e recordamos a definição de encaminhamento estático e dinâmico (Secção 2.2.2). Implicitamente, os conceitos de classfull e classless também foram utilizados.

A definição de rotas estáticas e manipulação das mesmas foi realizada na Secção 2.2.4.

O conceito de subnetting foi também posto à prova com a necessidade da divisão de endereços e a importância das máscaras de rede foi também, mais uma vez, realçada (Secção 2.3).

Resumindo, basicamente todo o capítulo de Protocolo IP foi abrangido e lembrado, e os conceitos inerentes ao mesmo foram consolidados.