



Redes de Computadores - Relatório TP2

João Nunes (A82300) Luís Braga (A82088)
Luís Martins (A82298)
Grupo 57

16 de Novembro de 2018

1 Questões e Respostas

1.1 1ª Parte

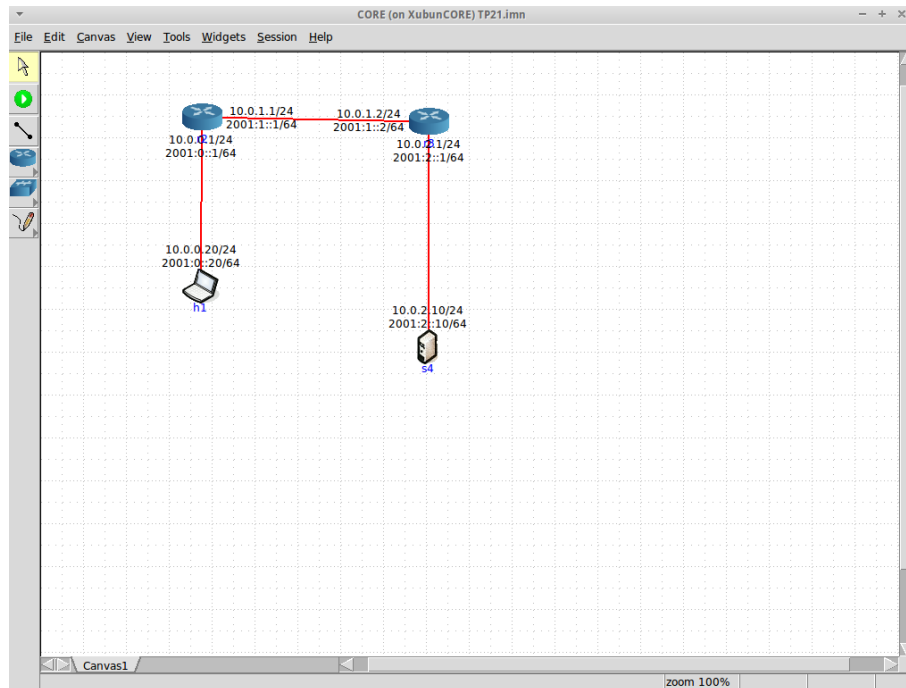


Figura 1: Topologia CORE

1. Prepare uma topologia CORE para verificar o comportamento do *traceroute*. Ligue um host (pc) h1 a um router r2; o router r2 a um router r3 que por sua vez, se liga a um host (servidor) s4. (Note que pode não existir conectividade IP imediata entre h1 e s4 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 h1. Numa shell de h1, execute o comando `traceroute -I` para o endereço IP do host s4.

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

Inicialmente o TTL é inicializado a 1, expirando no r1. Incrementa para dois e volta a falhar no r2, ocorrendo novamente outro time to leave exceeded. Até que por fim, fica com o valor de 3 possibilitando a comunicação entre o h1 e o s4.

The image shows a Wireshark packet capture from interface n1.eth0.82. The filter is set to 'icmp'. The packet list shows several ICMP messages. Packet 12 is a 'Time-to-live exceeded' message. Packet 13 is an 'Echo (ping) request' that also results in a 'Time-to-live exceeded' message. Packet 14 is another 'Time-to-live exceeded' message. Packet 15 is an 'Echo (ping) request' that succeeds. Subsequent packets (16-23) show a series of successful 'Echo (ping) request' and 'Echo (ping) reply' messages.

No.	Time	Source	Destination	Protocol	Length	Info
12	1.718070	10.0.0.1.2	10.0.0.20	ICMP	102	Time-to-live exceeded (Time to live exceeded)
13	1.718073	10.0.0.20	10.0.2.10	ICMP	74	Echo (ping) request id=0x0083, seq=6/1536, tt
14	1.718079	10.0.0.1.2	10.0.0.20	ICMP	102	Time-to-live exceeded (Time to live exceeded)
15	1.718082	10.0.0.20	10.0.2.10	ICMP	74	Echo (ping) request id=0x0083, seq=7/1792, tt
16	1.718093	10.0.2.10	10.0.0.20	ICMP	74	Echo (ping) reply id=0x0083, seq=7/1792, tt
17	1.718097	10.0.0.20	10.0.2.10	ICMP	74	Echo (ping) request id=0x0083, seq=8/2048, tt
18	1.718105	10.0.2.10	10.0.0.20	ICMP	74	Echo (ping) reply id=0x0083, seq=8/2048, tt
19	1.718108	10.0.0.20	10.0.2.10	ICMP	74	Echo (ping) request id=0x0083, seq=9/2304, tt
20	1.718116	10.0.2.10	10.0.0.20	ICMP	74	Echo (ping) reply id=0x0083, seq=9/2304, tt
21	1.718119	10.0.0.20	10.0.2.10	ICMP	74	Echo (ping) request id=0x0083, seq=10/2560, t
22	1.718127	10.0.2.10	10.0.0.20	ICMP	74	Echo (ping) reply id=0x0083, seq=10/2560, t
23	1.718130	10.0.0.20	10.0.2.10	ICMP	74	Echo (ping) request id=0x0083, seq=11/2816, t

The packet details pane for packet 15 shows the following information:

- Total Length: 60
- Identification: 0x211c (8476)
- Flags: 0x00
- Fragment offset: 0
- Time to live: 3
- Protocol: ICMP (1)
- Header checksum: 0x8088 [correct]
- Source: 10.0.0.20 (10.0.0.20)
- Destination: 10.0.2.10 (10.0.2.10)
- Internet Control Message Protocol
 - Type: 8 (Echo (ping) request)
 - Code: 0
 - Checksum: 0x81f0 [correct]
 - Identifier (BE): 131 (0x0083)
 - Identifier (LE): 33536 (0x8300)
 - Sequence number (BE): 7 (0x0007)

The packet bytes pane shows the raw data in hexadecimal and ASCII format.

Figura 2: Tráfego ICMP entre o r1 e o s4

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.

O valor inicial mínimo do campo TTL para estabelecer o contacto entre o h1 e o s4, é de 3. Como se pode verificar na figura 2, ocorre time to leave exceeded com qualquer outro valor no campo TTL, excepto com este a 3.

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

Analisando o resultado do comando do traceroute na linha de comandos e fazendo a média dos valores, obtemos um valor médio de tempo de ida e volta de aproximadamente 0.008ms.

```

traceroute -I 10.0.2.10
), 30 hops max, 60 byte packets
ms 0.007 ms
0.008 ms 0.008 ms
s 0.011 ms 0.011 ms

```

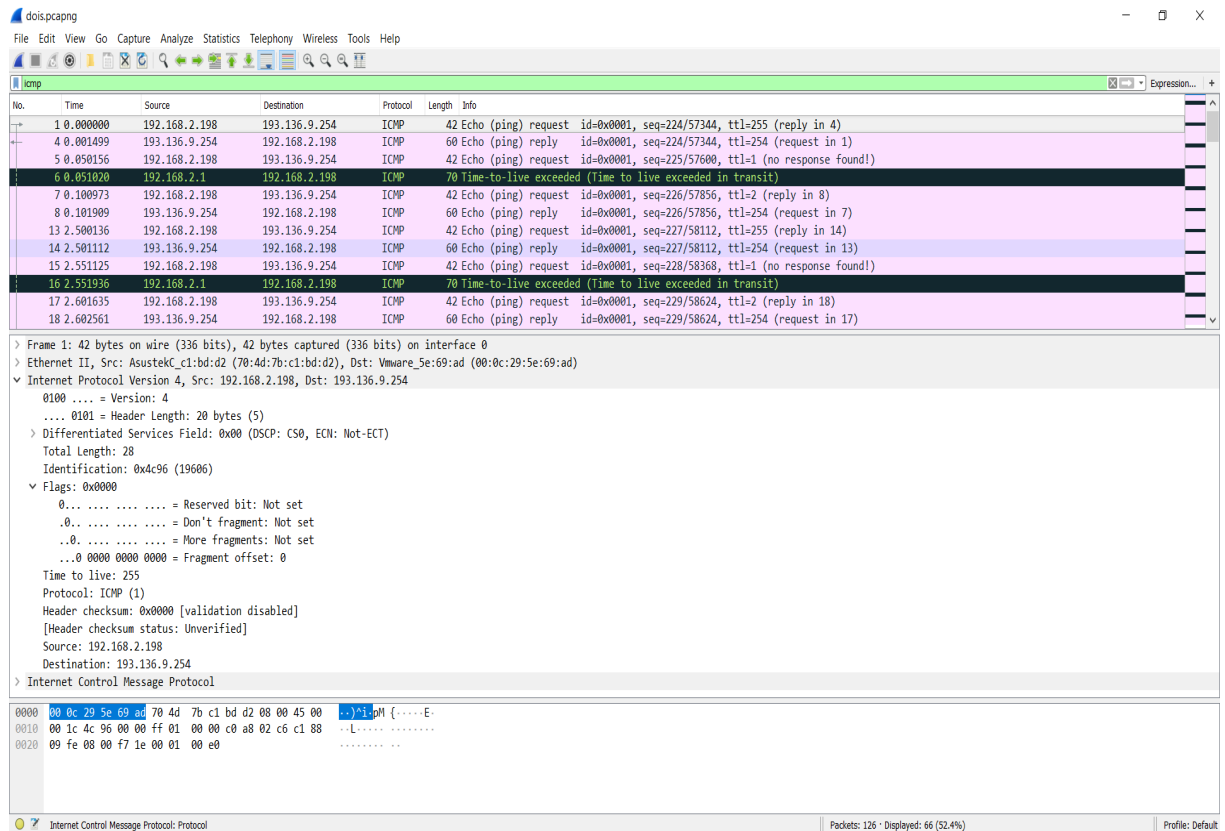


Figura 3: ICMP informação tamanho passado por defeito

2. Pretende-se agora usar o *traceroute* na sua máquina nativa, e gerar datagramas IP de diferentes tamanhos. Selecione a primeira mensagem ICMP capturada (referente a (i) tamanho por defeito) e centre a análise no nível protocolar IP (expandir o tab correspondente na janela de detalhe do Wireshark). Através da análise do cabeçalho IP diga:

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

O endereço IP da interface ativa do computador é 192.168.2.198.

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

O valor do campo protocolo é 1, que indicando o tipo de protocolo a ser utilizado no transporte do pacote, sendo que é 1 porque é ICMP.

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 de tamanho e o tamanho total do datagrama é de 28 bytes, logo o payload é dado por $28-20=8$, ou seja, o payload do datagrama possui 8 bytes de tamanho. O payload é portanto calculado a partir da diferença entre o tamanho total e o tamanho do cabeçalho.

d. O datagrama IP foi fragmentado?

O more fragments bit = 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.

Os campos do tráfego ICMP gerados pela interface da máquina que variam de pacote para pacote ordenados pela coluna source são o time to leave e o campo da identificação.

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

O padrão é que o campo de identificação do datagrama IP incrementa sempre um em um, em cada mensagem ICMP enviada pelo IP da interface atribuído à máquina.

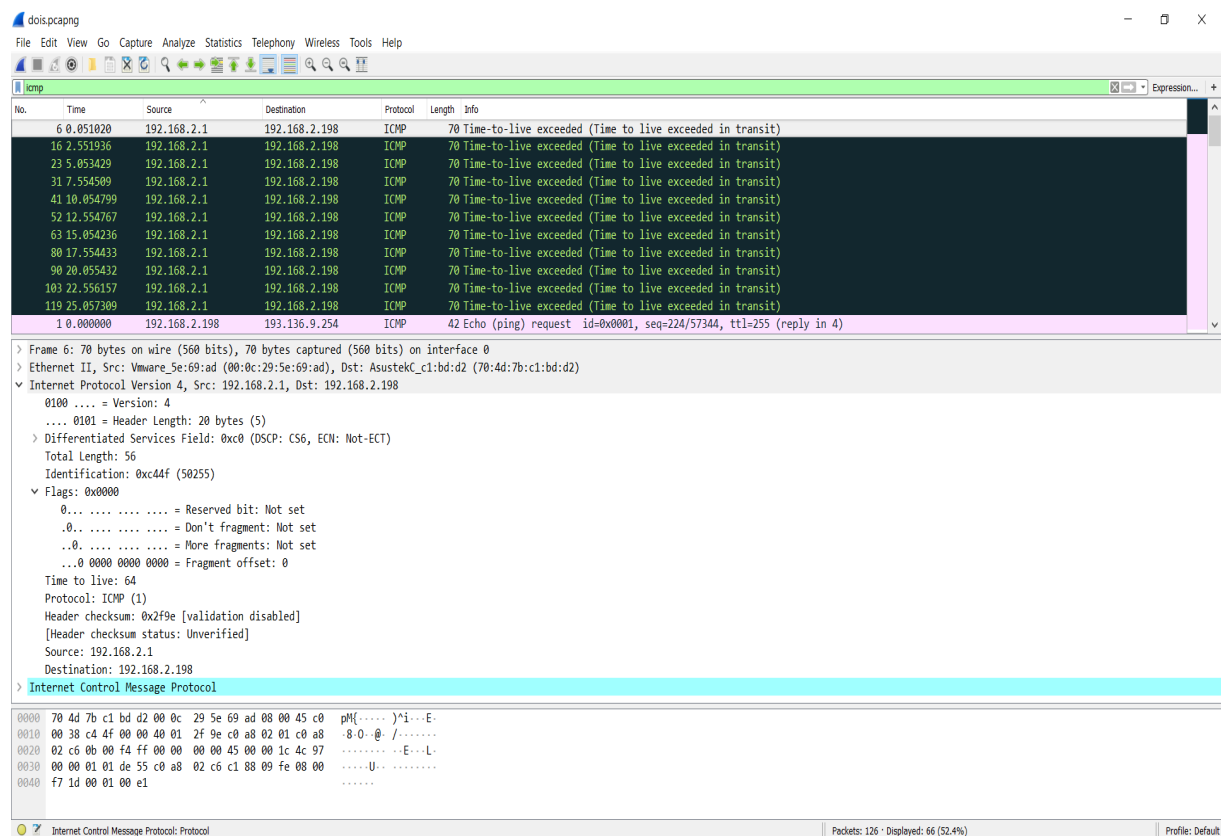


Figura 4: Sequência de mensagens ICMP com TTL exceeded

h. 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 todos as mensagens de resposta ICMP TTL exceeded enviados ao seu host? Porquê?

O valor do campo TTL nas mensagens exceeded enviadas para a máquina é igual a 64, sendo que este valor permanece constante para todas as mensagens exceeded. O valor permanece constante porque o valor de TTL para o primeiro hop router é sempre o mesmo.

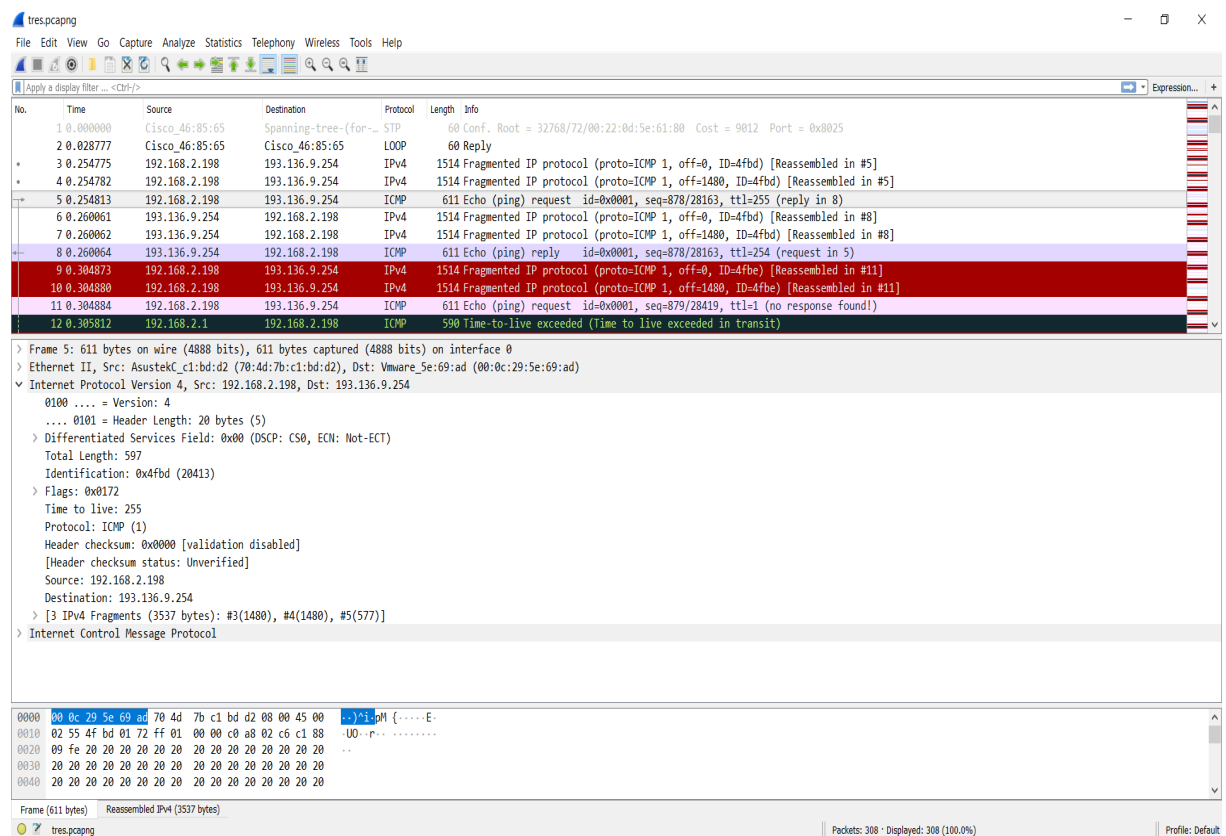


Figura 5: ICMP informação com tamanho de pacote com 3557 bytes.

3. Pretende-se agora analisar a fragmentação de pacotes IP. Reponha a ordem de tráfego capturado usando a coluna do tempo de captura. Observe o tráfego depois do tamanho de pacote ter sido definido para 35XX bytes.

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

Houve necessidade de fragmentar o pacote uma vez que excedeu o tamanho pré definido máximo que é 1500 bytes.

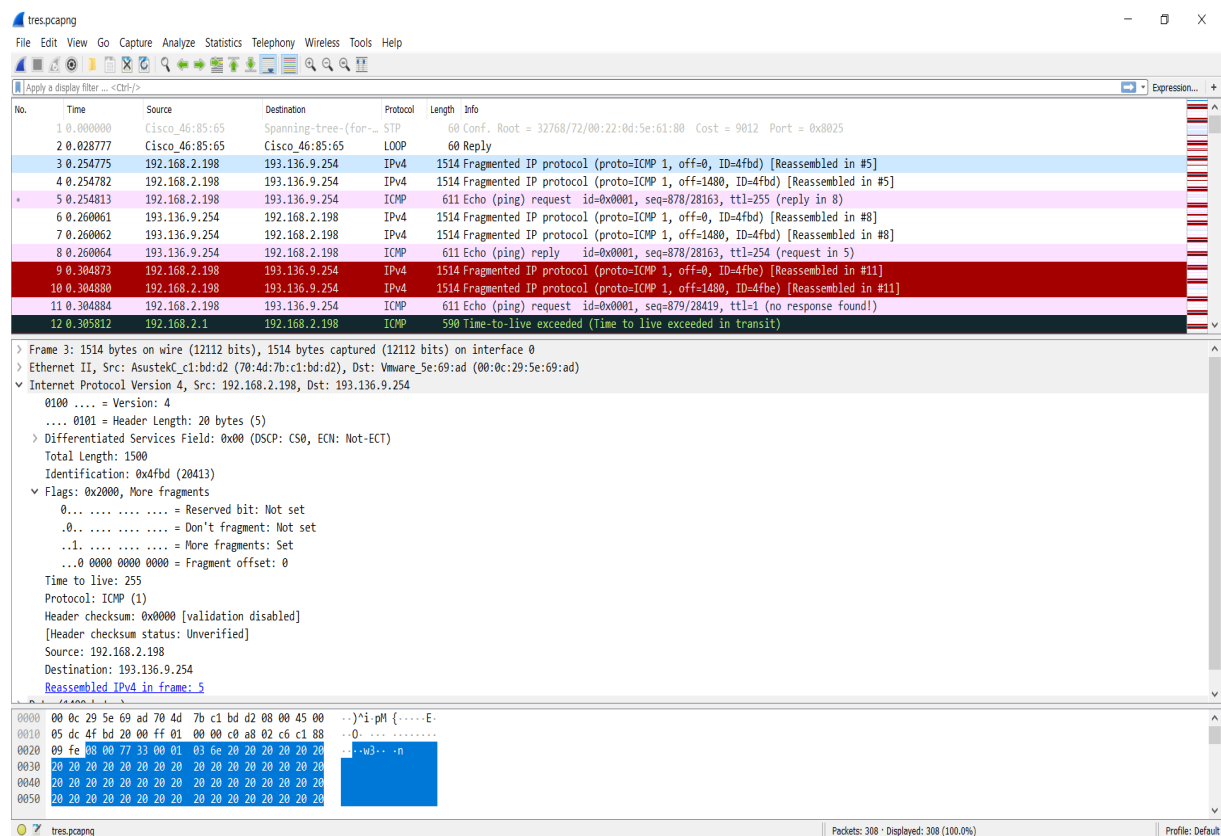


Figura 6: Primeiro datagrama IP fragmentado

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

A informação que indica que o datagrama foi o segmentado é que o bit do more fragments está a 1, e também que o fragment offset está a 0, a combinação dos dois faz com que seja o primeiro fragmento. O tamanho deste datagrama IP é 1500 e é dado pelo total length.

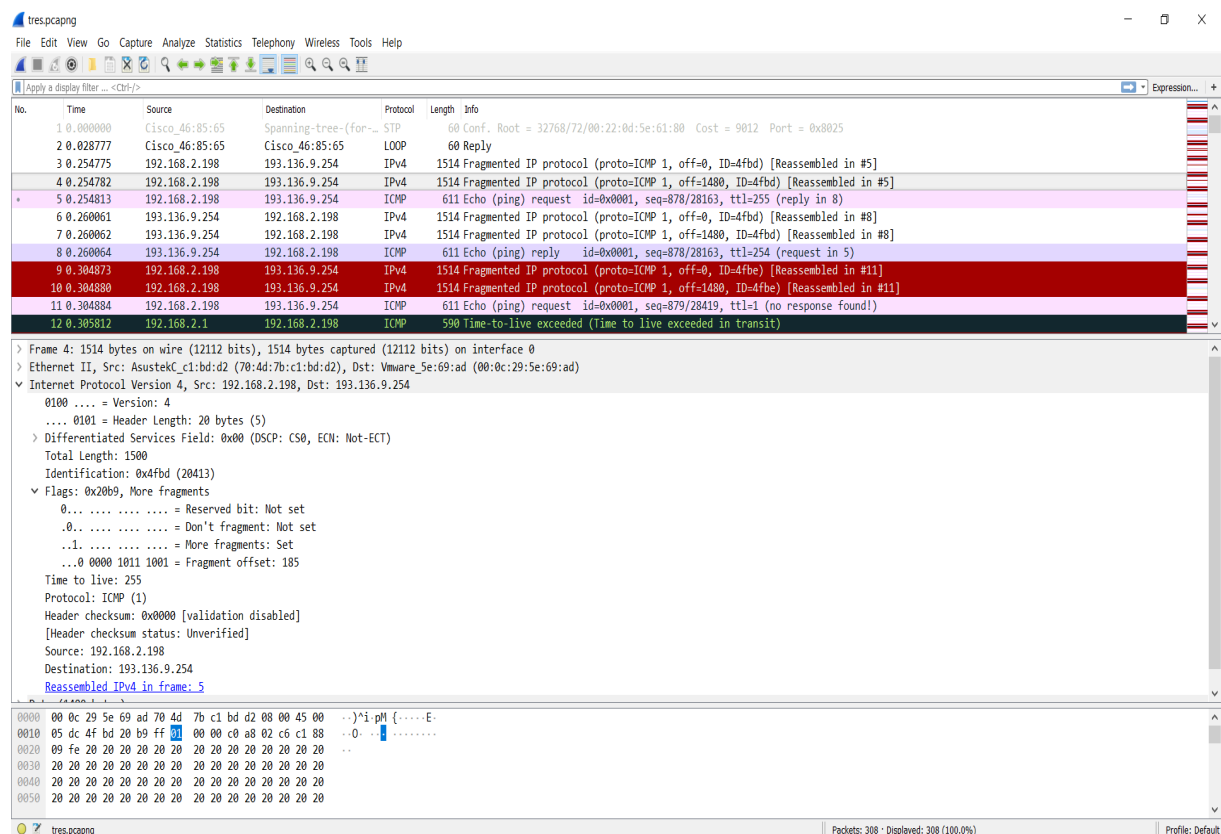


Figura 7: Segundo datagrama IP fragmentado

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

Não se trata do 1º fragmento uma vez que o fragment offset é igual a 185 o que nos indica que houve uma contagem em bytes desde o envio do datagrama original. Há mais fragmentos uma vez que a more fragments flag está a 1, indicando portanto que possui mais fragmentos.

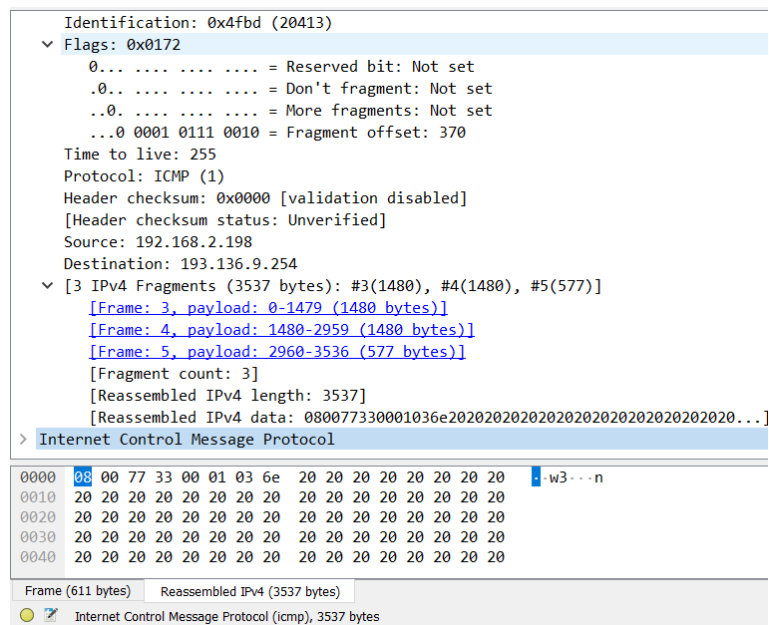


Figura 8: Campo IPv4 Fragments

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

Analisando o campo IPv4 Fragments da mensagem ICMP correspondente à figura 3, é possível verificar que foram criados 3 fragmentos a partir do datagrama original, o último fragmento pode ser detectado ao verificar a more fragments flag, quando esta estiver a 0 é porque é o último fragmento do datagrama original.

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

Os campos que mudam entre os diferentes fragmentos são o fragment offset e a flag. Entre os primeiros dois fragmentos e o último também é possível observar uma diferença no tamanho total, uma vez que nos primeiros dois o tamanho total é 1500 bytes com o bit do more fragments a 1, no último é de 597 bytes com o bit do more fragments a 0.

O datagrama original é reconstruído no princípio que quando o host recebe os fragmentos, estes são armazenados num buffer de remontagem baseado no campo do fragment offset. Quando todos os fragmentos do datagrama original são recebidos o datagrama é montado novamente, o que permite reconstruir o datagrama original.

1.2 2ª Parte

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

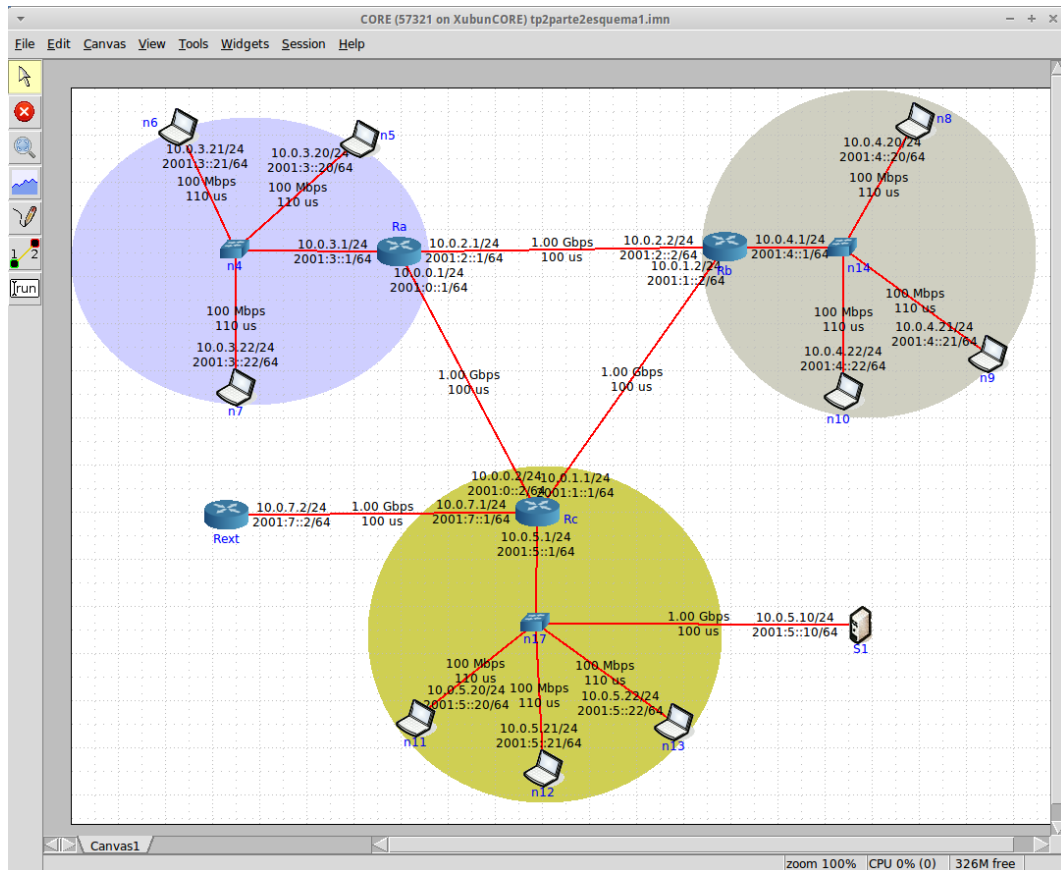


Figura 9: Topologia CORE

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 endereçamentos IP e as máscaras de rede atribuídos pelo CORE a cada equipamento podem ser identificados na figura 9.

b. Tratam-se de endereços públicos ou privados? Porquê?

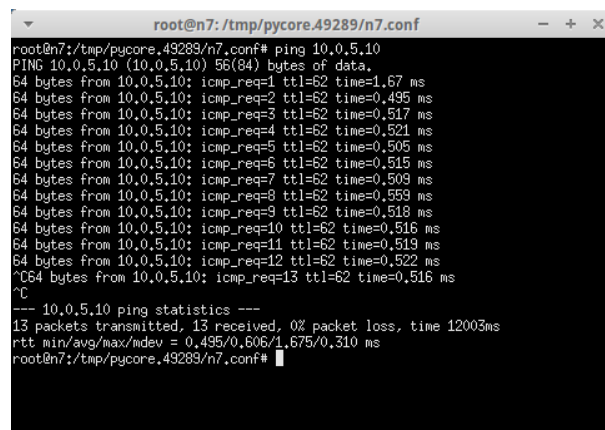
Tratam-se de endereços privados, porque pertencem à gama de IPs reservados à privatização destes mesmos (10.0.0.0 até 10.255.255.255).

c. Porque razão não é atribuído endereço IP aos switches?

Os switches são equipamentos destinados a interligação entre outros equipamentos, não precisando de um endereço IP, uma vez que estes registam os endereços conectados a si, de maneira, a poder fazer um correto redirecionamento não precisando, portanto, de endereço IP.

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).

Os resultados do comando ping em 3 laptops de cada um dos departamentos são os seguintes:

A terminal window titled 'root@n7:/tmp/pycore.49289/n7.conf' displays the output of a 'ping 10.0.5.10' command. The output shows 13 successful ping requests, each receiving 64 bytes of data with a TTL of 62. The response times are listed in milliseconds, ranging from 0.495 ms to 0.522 ms. At the end, a summary line shows '13 packets transmitted, 13 received, 0% packet loss, time 12003ms' and a detailed RTT breakdown: 'rtt min/avg/max/mdev = 0.495/0.506/1.675/0.310 ms'.

```
root@n7:/tmp/pycore.49289/n7.conf# ping 10.0.5.10
PING 10.0.5.10 (10.0.5.10) 56(84) bytes of data:
64 bytes from 10.0.5.10: icmp_req=1 ttl=62 time=1.67 ms
64 bytes from 10.0.5.10: icmp_req=2 ttl=62 time=0.495 ms
64 bytes from 10.0.5.10: icmp_req=3 ttl=62 time=0.517 ms
64 bytes from 10.0.5.10: icmp_req=4 ttl=62 time=0.521 ms
64 bytes from 10.0.5.10: icmp_req=5 ttl=62 time=0.505 ms
64 bytes from 10.0.5.10: icmp_req=6 ttl=62 time=0.515 ms
64 bytes from 10.0.5.10: icmp_req=7 ttl=62 time=0.509 ms
64 bytes from 10.0.5.10: icmp_req=8 ttl=62 time=0.553 ms
64 bytes from 10.0.5.10: icmp_req=9 ttl=62 time=0.518 ms
64 bytes from 10.0.5.10: icmp_req=10 ttl=62 time=0.516 ms
64 bytes from 10.0.5.10: icmp_req=11 ttl=62 time=0.519 ms
64 bytes from 10.0.5.10: icmp_req=12 ttl=62 time=0.522 ms
^C64 bytes from 10.0.5.10: icmp_req=13 ttl=62 time=0.516 ms
^C
--- 10.0.5.10 ping statistics ---
13 packets transmitted, 13 received, 0% packet loss, time 12003ms
rtt min/avg/max/mdev = 0.495/0.506/1.675/0.310 ms
root@n7:/tmp/pycore.49289/n7.conf#
```

Figura 10: Resultar de efetuar ping no laptop n7 do departamento A.

```
root@n10:/tmp/pycore.49289/n10.conf# ping 10.0.5.10
PING 10.0.5.10 (10.0.5.10) 56(84) bytes of data:
64 bytes from 10.0.5.10: icmp_req=1 ttl=62 time=0.915 ms
64 bytes from 10.0.5.10: icmp_req=2 ttl=62 time=0.623 ms
64 bytes from 10.0.5.10: icmp_req=3 ttl=62 time=0.514 ms
64 bytes from 10.0.5.10: icmp_req=4 ttl=62 time=0.548 ms
64 bytes from 10.0.5.10: icmp_req=5 ttl=62 time=0.522 ms
64 bytes from 10.0.5.10: icmp_req=6 ttl=62 time=0.519 ms
64 bytes from 10.0.5.10: icmp_req=7 ttl=62 time=0.560 ms
64 bytes from 10.0.5.10: icmp_req=8 ttl=62 time=0.524 ms
64 bytes from 10.0.5.10: icmp_req=9 ttl=62 time=0.559 ms
64 bytes from 10.0.5.10: icmp_req=10 ttl=62 time=0.515 ms
64 bytes from 10.0.5.10: icmp_req=11 ttl=62 time=0.534 ms
64 bytes from 10.0.5.10: icmp_req=12 ttl=62 time=0.515 ms
^C
--- 10.0.5.10 ping statistics ---
12 packets transmitted, 12 received, 0% packet loss, time 10999ms
rtt min/avg/max/ndev = 0.514/0.570/0.915/0.111 ms
root@n10:/tmp/pycore.49289/n10.conf#
```

Figura 11: Resultar de efetuar ping no laptop n10 do departamento B.

```
root@n12:/tmp/pycore.49289/n12.conf# ping 10.0.5.10
PING 10.0.5.10 (10.0.5.10) 56(84) bytes of data:
64 bytes from 10.0.5.10: icmp_req=1 ttl=64 time=0.490 ms
64 bytes from 10.0.5.10: icmp_req=2 ttl=64 time=0.286 ms
64 bytes from 10.0.5.10: icmp_req=3 ttl=64 time=0.347 ms
64 bytes from 10.0.5.10: icmp_req=4 ttl=64 time=0.339 ms
64 bytes from 10.0.5.10: icmp_req=5 ttl=64 time=0.297 ms
64 bytes from 10.0.5.10: icmp_req=6 ttl=64 time=0.285 ms
64 bytes from 10.0.5.10: icmp_req=7 ttl=64 time=0.270 ms
64 bytes from 10.0.5.10: icmp_req=8 ttl=64 time=0.267 ms
64 bytes from 10.0.5.10: icmp_req=9 ttl=64 time=0.282 ms
64 bytes from 10.0.5.10: icmp_req=10 ttl=64 time=0.267 ms
64 bytes from 10.0.5.10: icmp_req=11 ttl=64 time=0.472 ms
64 bytes from 10.0.5.10: icmp_req=12 ttl=64 time=0.269 ms
^C
--- 10.0.5.10 ping statistics ---
12 packets transmitted, 12 received, 0% packet loss, time 10999ms
rtt min/avg/max/ndev = 0.267/0.322/0.490/0.077 ms
root@n12:/tmp/pycore.49289/n12.conf#
```

Figura 12: Resultar de efetuar ping no laptop n12 do departamento C.

Ou seja, nos três casos apresentados, existe sempre 0% packet loss, pelo que significa que existe conectividade IP entre os laptops dos três departamentos com o servidor S1.

```
root@Rext: /tmp/pycore.57321/Rext.conf
root@Rext:/tmp/pycore.57321/Rext.conf# ping 10.0.5.10
connect: Network is unreachable
root@Rext:/tmp/pycore.57321/Rext.conf#
```

Figura 13: Resultar de efetuar ping.

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

Como é possível verificar na figura 13, não existe conectividade entre os dois nodos.

2) Para o router e um laptop do departamento A:

```
root@n7: /tmp/pycore.57322/n7.conf
root@n7:/tmp/pycore.57322/n7.conf# netstat -rn
Kernel IP routing table
Destination Gateway Genmask Flags MSS Window irtt Iface
0.0.0.0 10.0.3.1 0.0.0.0 UG 0 0 0 eth0
10.0.3.0 0.0.0.0 255.255.255.0 U 0 0 0 eth0
root@n7:/tmp/pycore.57322/n7.conf#
```

Figura 14: Resultado de efetuar o netstat no laptop n7 do departamento A

```
root@Ra:/tmp/pycore.57326/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 0.0.0.0 255.255.255.0 U 0 0 0 eth2
10.0.4.0 10.0.2.2 255.255.255.0 UG 0 0 0 eth1
10.0.5.0 10.0.0.2 255.255.255.0 UG 0 0 0 eth0
10.0.7.0 10.0.0.2 255.255.255.0 UG 0 0 0 eth0
```

Figura 15: Resultado de efetuar o netstat no router Ra do departamento A

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

A entrada da destination representa o destino da mensagem enviada pelo nodo. A gateway representa o ponto de entrada do routing. A genmask representa o destino sendo 0.0.0.0 se a rota é por default. As flags usadas são U ou UG, U se a rota está funcional e UG se está funcional e usa algum gateway para chegar ao destino. A entrada Iface representa a interface pelo qual os pacotes de dados da rota irão ser enviados.

```
root@n7:/tmp/pycore.49289/n7.conf# ps -ef
UID      PID  PPID  C  STIME TTY          TIME CMD
root      1    0    0 17:20 ?        00:00:00 /usr/sbin/vnoded -v -c /tmp/pyco
root     18     1    9 17:20 pts/5    00:00:00 /bin/bash
root     72    18    0 17:20 pts/5    00:00:00 ps -ef
root@n7:/tmp/pycore.49289/n7.conf#
```

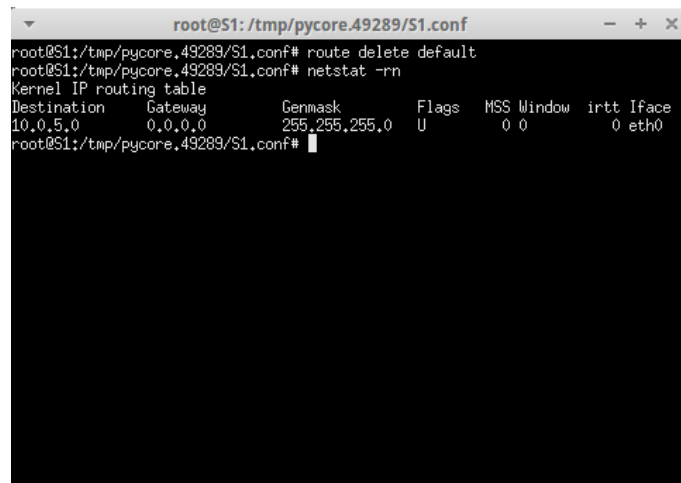
Figura 16: Lista de processos referente ao host n7 do departamento A

```
root@Ra:/tmp/pycore.49289/Ra.conf# ps -ef
UID      PID  PPID  C  STIME TTY          TIME CMD
root      1    0    0  17:20 ?        00:00:00 /usr/sbin/vnoded -v -c /tmp/pyco
root     59    1    0  17:20 ?        00:00:00 /usr/lib/quagga/zebra -u root -g
root     71    1    0  17:20 ?        00:00:00 /usr/lib/quagga/ospfd -u root -g
root     74    1    0  17:20 ?        00:00:00 /usr/lib/quagga/ospf6d -u root -
root     79    1    9  17:20 pts/7    00:00:00 /bin/bash
root    133   79    0  17:20 pts/7    00:00:00 ps -ef
root@Ra:/tmp/pycore.49289/Ra.conf#
```

Figura 17: Lista de processos referente ao router Ra do departamento A

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

O daemon core é inicializado, sendo que, analisando os processos na figura 17 referente ao router, é possível verificar que é utilizado o OSPF sendo isto um protocolo de encaminhamento dinâmico, logo o encaminhamento entre os routers é dinâmico. Entre os hosts (laptop), e analisando os processos da figura 16, não é utilizado nenhum tipo de protocolo de encaminhamento dinâmico, pelo que se conclui portanto que o encaminhamento é estático neste caso.

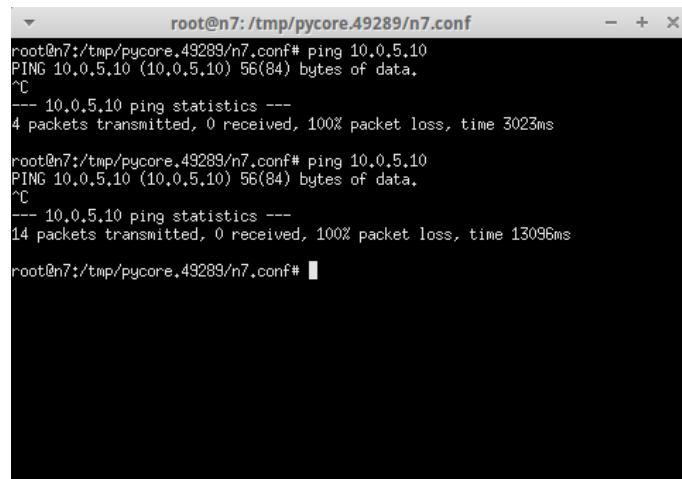


```
root@S1:/tmp/pycore.49289/S1.conf# route delete default
root@S1:/tmp/pycore.49289/S1.conf# netstat -rn
Kernel IP routing table
Destination        Gateway         Genmask         Flags   MSS Window  irtt Iface
10.0.5.0            0.0.0.0        255.255.255.0   U        0  0        0 eth0
root@S1:/tmp/pycore.49289/S1.conf#
```

Figura 18: Comando utilizado no S1 para retirar a rota por defeito

c. Amita 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.

Para os outros utilizadores dos outros departamentos, não irá ser possível estabelecer conexão com o servidor do departamento C. Ao apagar a rota por defeito (figura 18), não será possível encontrar algum caminho a partir das outras subredes (departamento A e B) para o servidor S1 do departamento C, apenas os utilizadores do departamento C poderão estabelecer comunicação com o servidor S1.

A terminal window titled 'root@n7: /tmp/pycore.49289/n7.conf' with standard window controls. The terminal shows two ping attempts to 10.0.5.10. The first attempt shows 4 packets transmitted, 0 received, and 100% packet loss. The second attempt shows 14 packets transmitted, 0 received, and 100% packet loss. Both attempts show a time of approximately 300ms and 1300ms respectively. The prompt 'root@n7: /tmp/pycore.49289/n7.conf#' is visible at the end of each command line.

```
root@n7: /tmp/pycore.49289/n7.conf# ping 10.0.5.10
PING 10.0.5.10 (10.0.5.10) 56(84) bytes of data.
^C
--- 10.0.5.10 ping statistics ---
4 packets transmitted, 0 received, 100% packet loss, time 3023ms

root@n7: /tmp/pycore.49289/n7.conf# ping 10.0.5.10
PING 10.0.5.10 (10.0.5.10) 56(84) bytes of data.
^C
--- 10.0.5.10 ping statistics ---
14 packets transmitted, 0 received, 100% packet loss, time 13096ms

root@n7: /tmp/pycore.49289/n7.conf#
```

Figura 19: Resultado de efetuar ping do host n7 para o servidor S1.

Observando os resultados da figura 19, ao efetuar ping de um laptop de outra subrede, o n7 do departamento A, existe 100% de packet loss, o que significa que não existe conexão para o servidor S1.

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 registre os comandos que usou.

O comando utilizado para adicionar de volta a rota estática necessária para restaurar a conectividade para o servidor S1 foi o *route add default gw 10.0.5.1 eth0*.

```
root@S1:/tmp/pycore.49289/S1.conf# route add default gw 10.0.5.1 eth0
root@S1:/tmp/pycore.49289/S1.conf# netstat -rn
netstat-rn: command not found
root@S1:/tmp/pycore.49289/S1.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
root@S1:/tmp/pycore.49289/S1.conf#
```

Figura 20: Comando utilizado para reestabelecer conectividade.

Foi utilizado o 10.0.5.1 como gateway uma vez que este é o endereço IP do router Rc.

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.

Ao adicionar de novo a rota por defeito, será novamente possível para os utilizadores de outras sub redes (departamento A e B) aceder ao servidor, porque possuem novamente um caminho até ao servidor S1. O mesmo poderá ser verificado na seguinte figura:

```
root@n7: /tmp/pycore.49289/n7.conf# ping 10.0.5.10
PING 10.0.5.10 (10.0.5.10) 56(84) bytes of data:
64 bytes from 10.0.5.10: icmp_req=1 ttl=62 time=1.21 ms
64 bytes from 10.0.5.10: icmp_req=2 ttl=62 time=0.531 ms
64 bytes from 10.0.5.10: icmp_req=3 ttl=62 time=0.514 ms
64 bytes from 10.0.5.10: icmp_req=4 ttl=62 time=0.517 ms
64 bytes from 10.0.5.10: icmp_req=5 ttl=62 time=0.511 ms
64 bytes from 10.0.5.10: icmp_req=6 ttl=62 time=0.509 ms
64 bytes from 10.0.5.10: icmp_req=7 ttl=62 time=0.519 ms
64 bytes from 10.0.5.10: icmp_req=8 ttl=62 time=0.512 ms
64 bytes from 10.0.5.10: icmp_req=9 ttl=62 time=0.532 ms
64 bytes from 10.0.5.10: icmp_req=10 ttl=62 time=0.512 ms
64 bytes from 10.0.5.10: icmp_req=11 ttl=62 time=0.529 ms
64 bytes from 10.0.5.10: icmp_req=12 ttl=62 time=0.536 ms
64 bytes from 10.0.5.10: icmp_req=13 ttl=62 time=0.556 ms
64 bytes from 10.0.5.10: icmp_req=14 ttl=62 time=0.513 ms
64 bytes from 10.0.5.10: icmp_req=15 ttl=62 time=0.512 ms
^C
--- 10.0.5.10 ping statistics ---
15 packets transmitted, 15 received, 0% packet loss, time 14000ms
rtt min/avg/max/mdev = 0.509/0.567/1.214/0.175 ms
root@n7: /tmp/pycore.49289/n7.conf#
```

Figura 21: Ping do laptop n7 (departamento A) para o servidor S1.

Como existe 0% de packet loss, todos os pacotes enviados são recebidos, o que significa que a conexão voltou ao normal.

A nova tabela de encaminhamento do servidor está apresentada também na figura 20.

3. Definição de Sub-redes

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.

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

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

Apartir do endereço de rede IP 172.57.48.0/20 e, sabendo que existem três sub-redes, determina-se que vão ser necessários quatro bits extra, logo fica-se com 172.57.48.0/24 para o eventual aumento do número de departamentos. Daí resulta o novo esquema de endereçamentos:

172.57.0011		0000 .0	– Não é válido (broadcast)
		0001 .0	– Sub-rede A
		0010 .0	– Sub-rede B
		0011 .0	– Sub-rede C
		0100 .0	
		0101 .0	
		0110 .0	
		0111 .0	
		1000 .0	
		1001 .0	
		1010 .0	
		1011 .0	
		1100 .0	
		1101 .0	
		1110 .0	
		1111 .255	– Não é válido

Figura 22: Novo esquema de endereçamento

O departamento A terá o espaço de endereçamento [172.57.49.1 .. 172.57.49.254], da mesma forma o departamento B e C terão os espaços de endereçamento [172.57.50.1 .. 172.57.50.254] e [172.57.51.1 .. 172.57.51.254], respectivamente.

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

A máscara de rede usada é 255.255.255.0 e o número de hosts pode ser calculado da seguinte maneira:

$$hosts = 2^8 - 2 = 254 \quad (1)$$

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

1.3 Conclusão

Na primeira parte do trabalho, foi feita uma análise ao protocolo IPv4. Para isso se concretizar, foi elaborada uma topologia CORE para estudar o comportamento e analisar o tráfego ICMP. Para além disso, o trabalho também foi focado em casos particulares, como a fragmentação de pacotes IP que ocorre devido ao excesso de dimensão, foi necessário identificar e entender o comportamento dos pacotes fragmentados. Todo este trabalho serviu para consolidar os conhecimentos relativamente ao protocolo IPv4 e como o datagrama IP é fragmentado bem como o seu funcionamento.

Na segunda parte do trabalho, o grupo ficou mais esclarecido relativamente ao endereçamento e encaminhamento IP, bem como a sua análise. De modo a concretizar essa análise foi elaborada uma topologia CORE de maior complexidade em relação à topologia elaborada na parte 1. Com essa topologia estabelecida, foi verificado o funcionamento do encaminhamento em três redes diferentes. Nesta parte, foi também manipulado os endereços IP, para subnetting, e o seu encaminhamento.