

## **Protocolo IPv4 - Trabalho Prático 2**

Tiago Alves a80872, Francisco Costa a95227, Cláudio Bessa a97063

Universidade do Minho

## 1 Trabalho Prático 2 - Parte 1

### 1.1 Topologia CORE

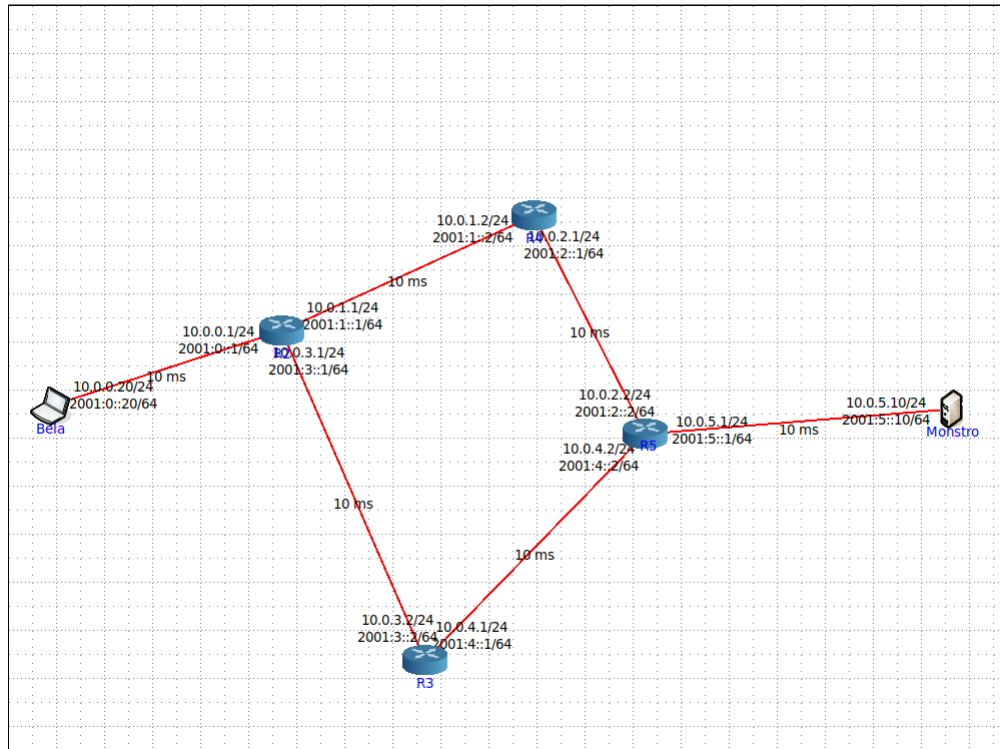
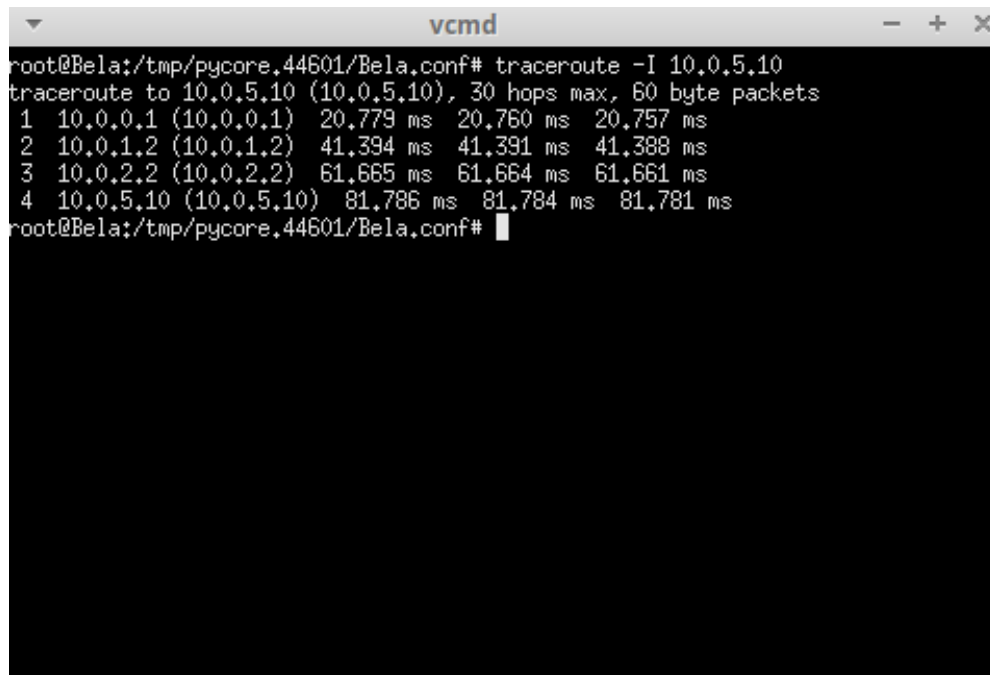


Figura 1. Topologia CORE efetuada

a) Ative o wireshark ou o tcpdump no host Bela. Numa shell de Bela execute o comando *traceroute -I* para o endereço IP do Monstro.



```
vcmd
root@Bela:/tmp/pycore.44601/Bela.conf# traceroute -I 10.0.5.10
traceroute to 10.0.5.10 (10.0.5.10), 30 hops max, 60 byte packets
 1 10.0.0.1 (10.0.0.1) 20.779 ms 20.760 ms 20.757 ms
 2 10.0.1.2 (10.0.1.2) 41.394 ms 41.391 ms 41.388 ms
 3 10.0.2.2 (10.0.2.2) 61.665 ms 61.664 ms 61.661 ms
 4 10.0.5.10 (10.0.5.10) 81.786 ms 81.784 ms 81.781 ms
root@Bela:/tmp/pycore.44601/Bela.conf#
```

Figura 2. Execução do comando *traceroute -I*

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

**R:** O tráfego ICMP vai alterando o valor do TTL uma vez que terá que fazer vários testes até ser suficiente para chegar ao Monstro.

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

**R:** O valor inicial mínimo do TTL será 40, devido aos *delays* inseridos para a sua propagação.

d) Calcule o valor médio do tempo de ida-e-volta (RTT - *Round-Trip Time*) obtido no acesso ao servidor. Para melhorar a média, poderá alterar o número pacotes de prova com a opção -q

**R:** O valor médio do RTT será de 81.777ms.

e) O valor médio do atraso num sentido (*One-Way Delay*) poderia ser calculado com precisão dividindo o RTT por dois? O que torna difícil o cálculo desta métrica?

**R:** O *One-way Delay* pode ser calculado dividindo o **RTT** por dois, mas para obtermos valores fiáveis, os caminhos de A a B e de B a A, tem que ser similares, com congestão, número de saltos e qualidades equivalentes, caso contrário teríamos que o calcular por estimativa.

## 1.2 Traceroute na máquina nativa

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

**R:** O computador obtém um endereço de 172.26.9.57 na sua ligação ao *router*. Sendo o IP da máquina 193.136.9.254.

```
Wireless LAN adapter Wi-Fi:

    Connection-specific DNS Suffix  . : eduroam.uminho.pt
    Link-local IPv6 Address . . . . . : fe80::1963:243a:9817:17fd%3
    IPv4 Address. . . . . : 172.26.9.57
    Subnet Mask . . . . . : 255.255.0.0
    Default Gateway . . . . . : 172.26.254.254
```

**Figura 3.** Endereço IPv4

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

**R:** O valor no campo de protocolo é 1 (ou 0x01 em hexadecimal) uma vez que é utilizado o protocolo do ICMP.

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

**R:** O tamanho do pacote é de 56 bytes. Onde 20 bytes são referentes ao cabeçalho e 36 ao *payload*.

$$\text{Payloadsize} = \text{packetlength} - \text{headerlength}$$

d) O datagrama IP foi fragmentado? Justifique.

**R:** O datagrama IP não se encontra fragmentado já que não se verifica um *offset*, ou seja o tamanho do *payload* mais o do header equivalem ao tamanho total do pacote. E o *ip.frag\_offset* é 0.

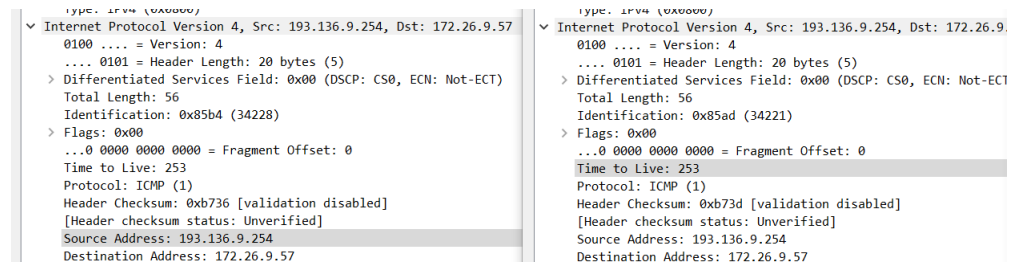
$$20\text{headerlength} + 36\text{payloadlength} (28\text{databytes} + 8\text{bytesICMPheader}) = 56\text{packetlength}$$

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.

**R:** Os campos que sofrem alterações no cabeçalho IP são o de identificação e o *header checksum*, respetivamente.

ip.src==193.136.9.254						
No.	Time	Source	Destination	Protocol	Length	Info
1891	173.232304	193.136.9.254	172.26.9.57	ICMP	70	Echo (ping) reply id=0x0001, seq=1591/14086, ttl=253 (request in 1889)
1907	175.643713	193.136.9.254	172.26.9.57	ICMP	70	Echo (ping) reply id=0x0001, seq=1594/14854, ttl=253 (request in 1899)
1910	175.695700	193.136.9.254	172.26.9.57	ICMP	70	Echo (ping) reply id=0x0001, seq=1597/15622, ttl=253 (request in 1906)
1916	178.001178	193.136.9.254	172.26.9.57	ICMP	70	Echo (ping) reply id=0x0001, seq=1600/16390, ttl=253 (request in 1915)
1925	178.192576	193.136.9.254	172.26.9.57	ICMP	70	Echo (ping) reply id=0x0001, seq=1603/17158, ttl=253 (request in 1920)
1948	180.505306	193.136.9.254	172.26.9.57	ICMP	70	Echo (ping) reply id=0x0001, seq=1606/17926, ttl=253 (request in 1947)
1956	180.650277	193.136.9.254	172.26.9.57	ICMP	70	Echo (ping) reply id=0x0001, seq=1609/18694, ttl=253 (request in 1955)
2032	183.065454	193.136.9.254	172.26.9.57	ICMP	70	Echo (ping) reply id=0x0001, seq=1612/19462, ttl=253 (request in 2030)
2037	183.156194	193.136.9.254	172.26.9.57	ICMP	70	Echo (ping) reply id=0x0001, seq=1615/20230, ttl=253 (request in 2036)
2052	185.515098	193.136.9.254	172.26.9.57	ICMP	70	Echo (ping) reply id=0x0001, seq=1618/20998, ttl=253 (request in 2051)
2058	185.650242	193.136.9.254	172.26.9.57	ICMP	70	Echo (ping) reply id=0x0001, seq=1621/21766, ttl=253 (request in 2057)
2073	187.995573	193.136.9.254	172.26.9.57	ICMP	70	Echo (ping) reply id=0x0001, seq=1624/22534, ttl=253 (request in 2071)
2082	188.149371	193.136.9.254	172.26.9.57	ICMP	70	Echo (ping) reply id=0x0001, seq=1627/23302, ttl=253 (request in 2081)
2097	190.511137	193.136.9.254	172.26.9.57	ICMP	70	Echo (ping) reply id=0x0001, seq=1630/24070, ttl=253 (request in 2096)
2103	190.656100	193.136.9.254	172.26.9.57	ICMP	70	Echo (ping) reply id=0x0001, seq=1633/24838, ttl=253 (request in 2102)
2121	193.000854	193.136.9.254	172.26.9.57	ICMP	70	Echo (ping) reply id=0x0001, seq=1636/25606, ttl=253 (request in 2120)
2127	193.170103	193.136.9.254	172.26.9.57	ICMP	70	Echo (ping) reply id=0x0001, seq=1639/26374, ttl=253 (request in 2126)

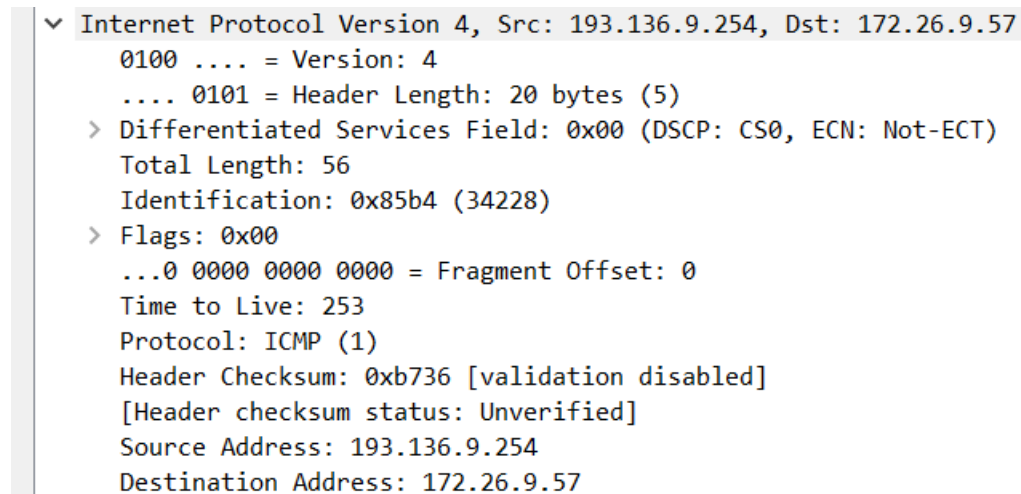
Figura 4. Pacotes capturados com *source* 193.136.9.254



**Figura 5.** Verificação da alteração dos campos

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

**R:** O valor de identificação do datagrama IP é *0x85b4* (34228 em decimal) e o valor de identificação do datagrama TTL é 253. O valor do TTL mantém um valor constante, como verificado na Figura 4.



**Figura 6.** Identificação dos datagramas

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 enviadas ao seu host? Porquê?

**R:** Os pacotes são enviados pelos routers em que o **TTL** falha. Por *default* o **TTL** de envio ronda por volta de 65, por isso observam-se disparidades dependendo do número de *hops* a retornar. Contudo o campo de **TTL** tem 8 bits, havendo alguns dispositivos configurados para devolver usando os 8 bits a 1 (255 em base decimal).

### 1.3 Fragmentação de pacotes IP

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

**R:** Há necessidade de fragmentar o pacote inicial pois os *routers* não são capazes de receber toda a informação simultaneamente, então dividem o pacote em fragmentos mais pequenos.

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?

**R:** Sabemos que o datagrama foi fragmentado, pois ele indica-nos que tem mais fragmentos, e sabemos que é o primeiro porque o *fragment offset* é 0, o tamanho deste datagrama é 1500.

```

Type: IPv4 (0x0800)
▼ Internet Protocol Version 4, Src: 193.136.9.240, Dst: 172.26.9.57
  0100 .... = Version: 4
  .... 0101 = Header Length: 20 bytes (5)
  > Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
  Total Length: 1500
  Identification: 0x79dc (31196)
  > Flags: 0x20, More fragments
  ...0 0000 0000 0000 = Fragment Offset: 0
  Time to Live: 61
  Protocol: ICMP (1)
  Header Checksum: 0x5d79 [validation disabled]
  [Header checksum status: Unverified]
  Source Address: 193.136.9.240
  Destination Address: 172.26.9.57
  [Reassembled IPv4 in frame: 43]
▼ Data (1480 bytes)
  Data: 00002f170001376e2020202020202020202020202020202020202020202020202020...
  [Length: 1480]

```

**Figura 7.** Primeiro fragmento

```

> Flags: 0x20, More fragments
...0 0000 0000 0000 = Fragment Offset: 0

```

**Figura 8.** Primeiro fragmento *fragment offset*

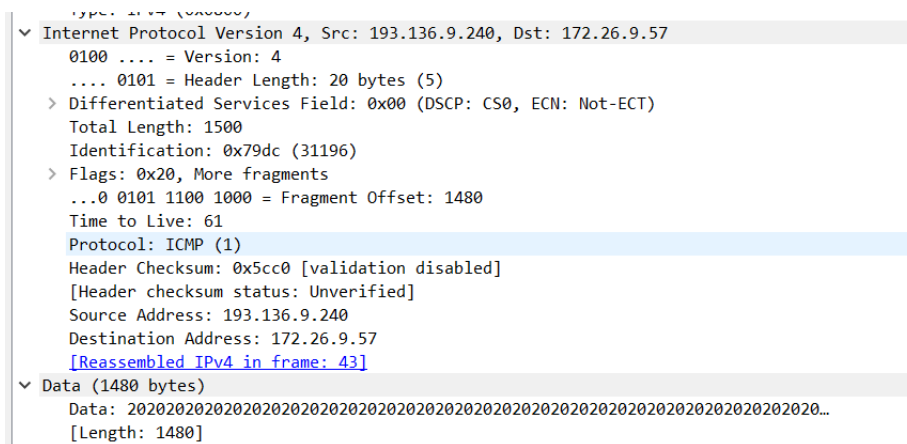
Total Length: 1500

**Figura 9.** Tamanho do datagrama do primeiro fragmento



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?

**R:** Não se trata do primeiro fragmentos visto que possui um *offset* que possui o mesmo tamanho do fragmento anterior, sabendo assim que tem mais fragmentos, uma vez que os datagramas indicam-nos precisamente isso nas *flags*.



**Figura 10.** Segundo fragmento

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

**R:** Foram criados um total de 3 fragmentos em resultado 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.

**R:** Os campos que sofrem alterações são o IP, o tamanho e as *fragment offset*, para o fragmento cujo *offset* for 0. Este será o primeiro fragmento. O segundo fragmento terá como *offset* o valor do fragmento anterior, o terceiro obterá um *offset* a soma da data dos dois fragmentos anteriores e por fim, o último dirá que não existe mais fragmentos (*more frags = 0*).

```

... 0100 0000 = 10 bits: individual address (unicast)
Type: IPv4 (0x0800)
v Internet Protocol Version 4, Src: 193.136.9.240, Dst: 172.26.9.57
  0100 .... = Version: 4
  .... 0101 = Header Length: 20 bytes (5)
  > Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
    Total Length: 1045
    Identification: 0x79dc (31196)
  > Flags: 0x01
    ...0 1011 1001 0000 = Fragment Offset: 2960
    Time to Live: 61
    Protocol: ICMP (1)
    Header Checksum: 0x7dce [validation disabled]
    [Header checksum status: Unverified]
    Source Address: 193.136.9.240
    Destination Address: 172.26.9.57
  > [3 IPv4 Fragments (3985 bytes): #41(1480), #42(1480), #43(1025)]
v Internet Control Message Protocol

```

Figura 11. Cardinalidade de fragmentos

f) Verifique o processo de fragmentação através de um processo de cálculo.

**R:** O processo de remontagem é calculado pela adição dos *offsets* com o tamanho do ultimo fragmento a ser enviado, neste caso,  $1480 + 1480 + 1025 = 3985$ . Uma vez que 20 bytes são reservados para o *header*, obtemos assim o valor inicial de 4005 bytes.

O cálculo pode ser designado pela fórmula:

$$offset + (bytesdados) = tamanhototal$$

g) Escreva uma expressão lógica que permita detetar o último fragmento correspondente ao datagrama original.

**R:** O último fragmento tem necessariamente que ter a *flag* "More Fragments" a 0 obtendo assim um tamanho final de :

$$tamanhooriginal - 1500 * (nfragmentos - 1) - 20$$

## 2 Trabalho Prático 2 - Parte 2

### 2.1 IP atribuídos pelo CORE

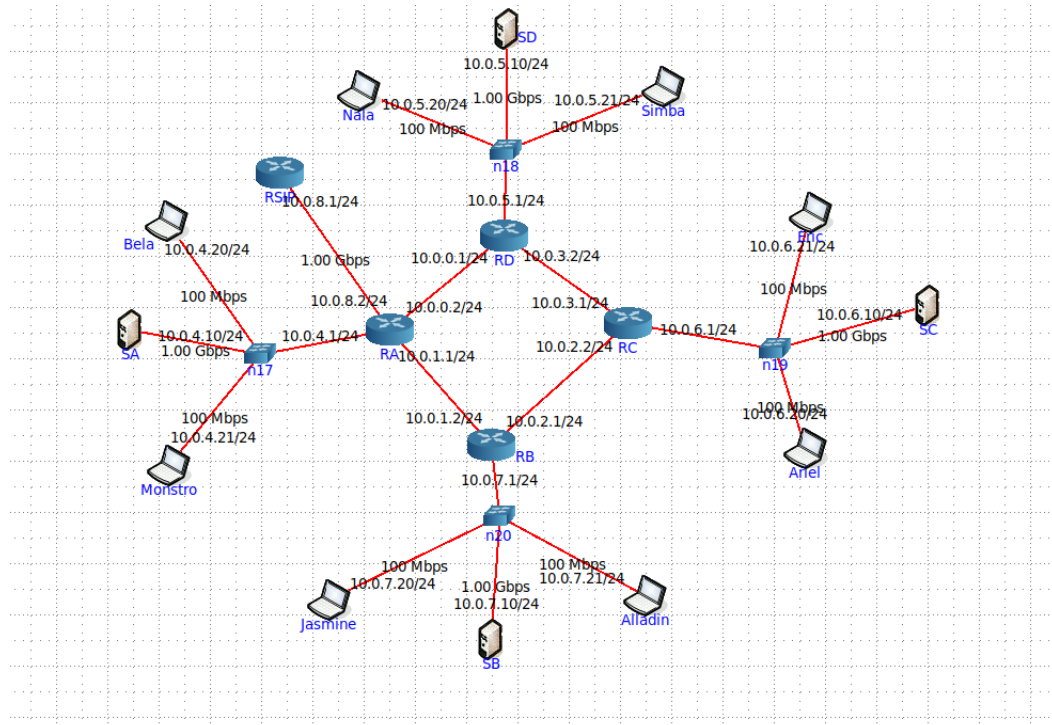


Figura 12. Topologia Core 2

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.

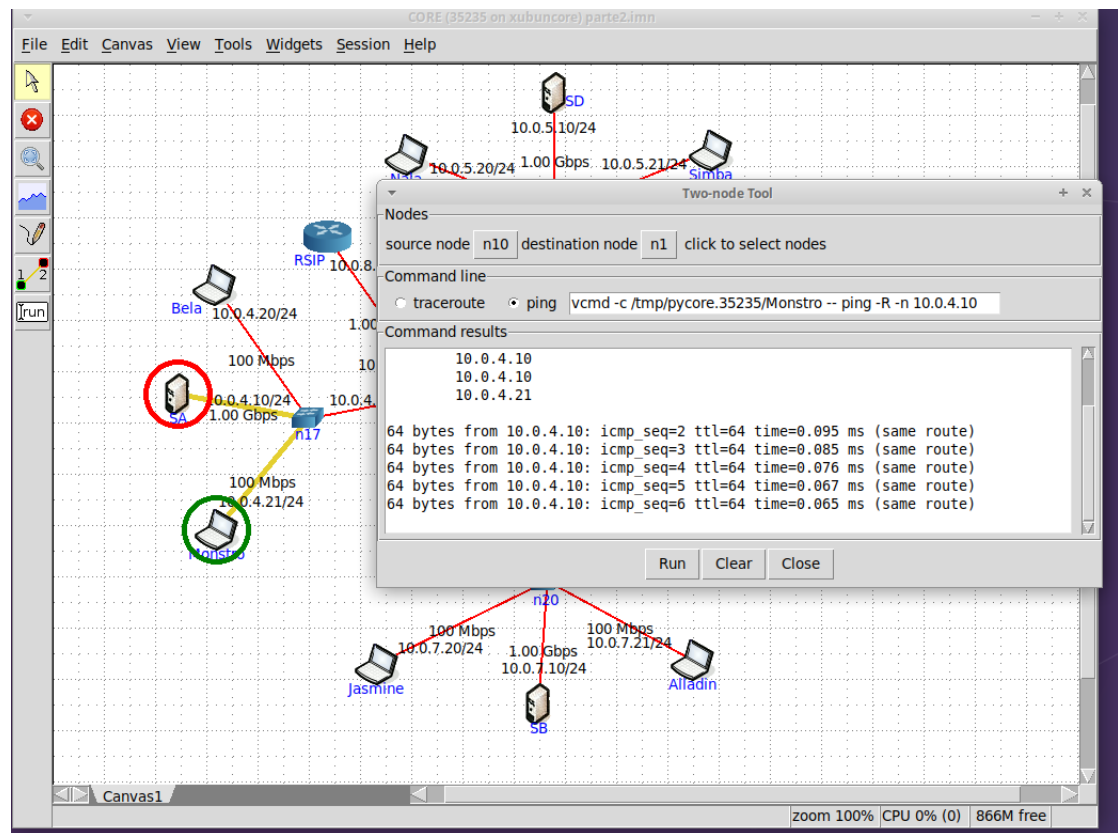
**R:** A máscara de rede presente é de 24 bits. Verificar figura 12.

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

**R:** Os IP's definidos pelo CORE são privados uma vez que fazem parte do grupo de IP's reservados pelo IANA (*Internet Assigned Numbers Authority*) para redes privadas. (Sendo inicializados por 10.0.0.0 com seu prefixo 10/8). Verificar figura 12.

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

**R:** Os *switches* não têm um endereço IP uma vez que apenas precisam de saber o **MAC address** de cada dispositivo conectado, ou seja apenas opera na *layer* 2 do **TCP/IP**.

d) Usando o comando *ping* certifique-se que existe conectividade IP interna a cada departamento (e.g. entre um laptop e o servidor respetivo).

**Figura 13.** Teste de conectividade utilizando o comando *ping*

e) Execute o número mínimo de comandos ping que lhe permite verificar a existência de conectividade ICP entre departamentos.

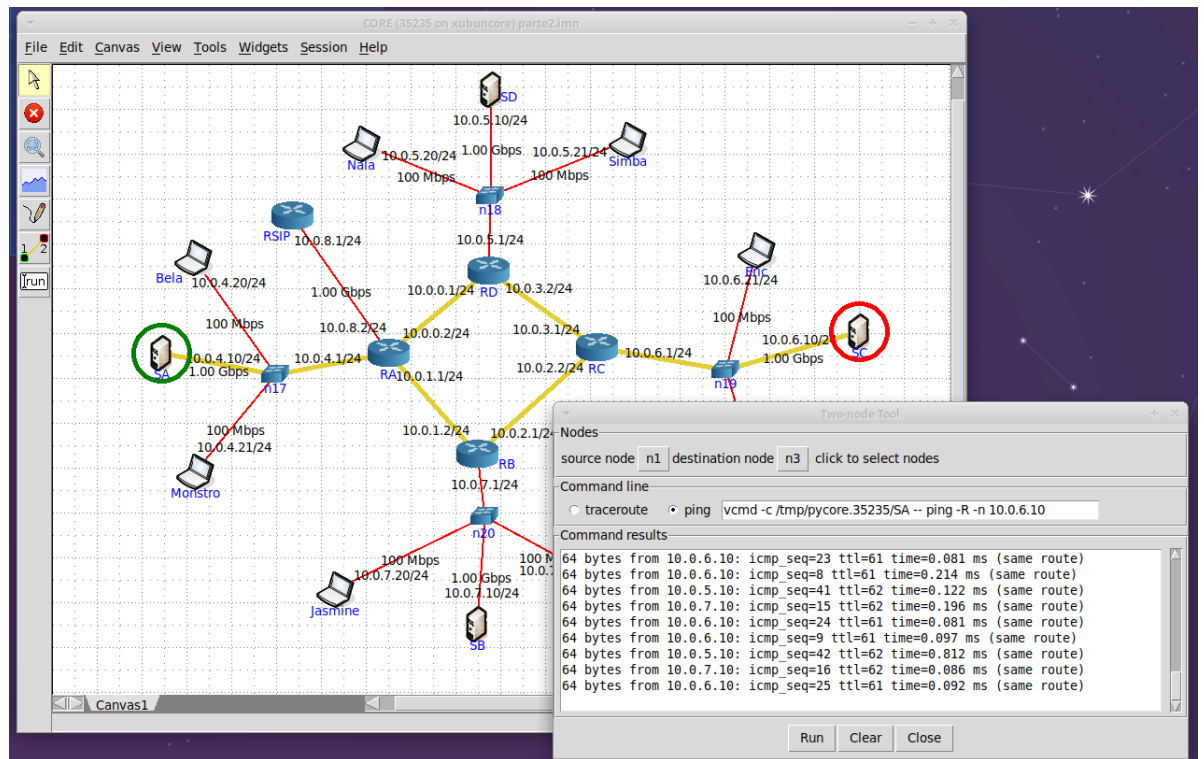


Figura 14. Conectividade entre SA e SC

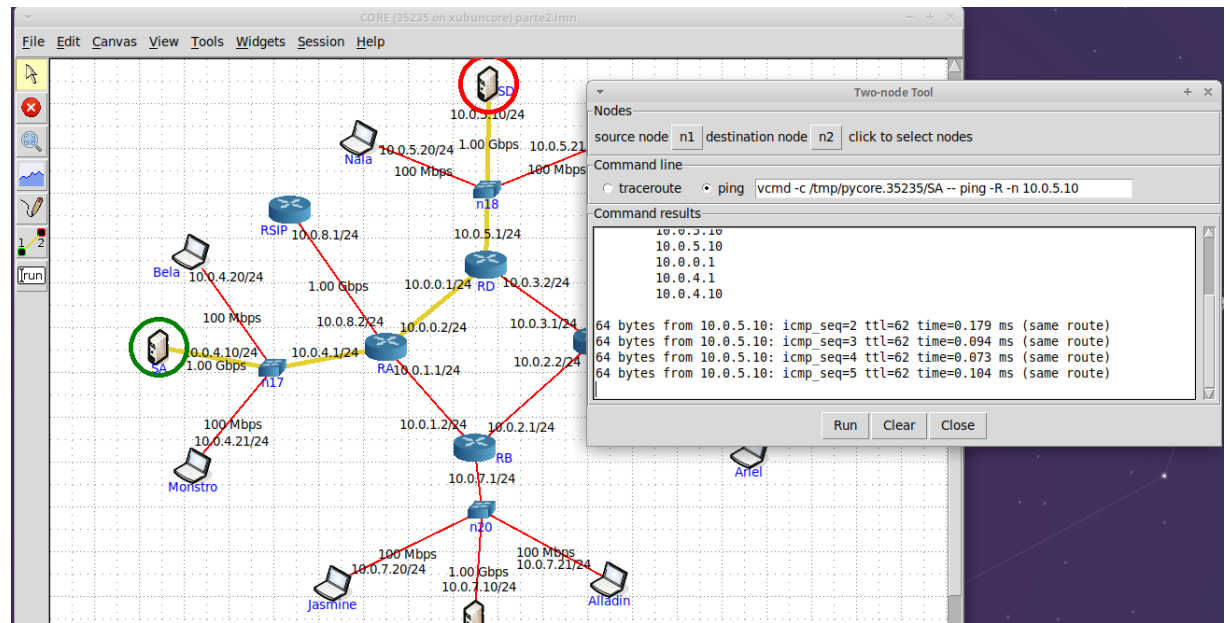


Figura 15. Conectividade entre SA e SD

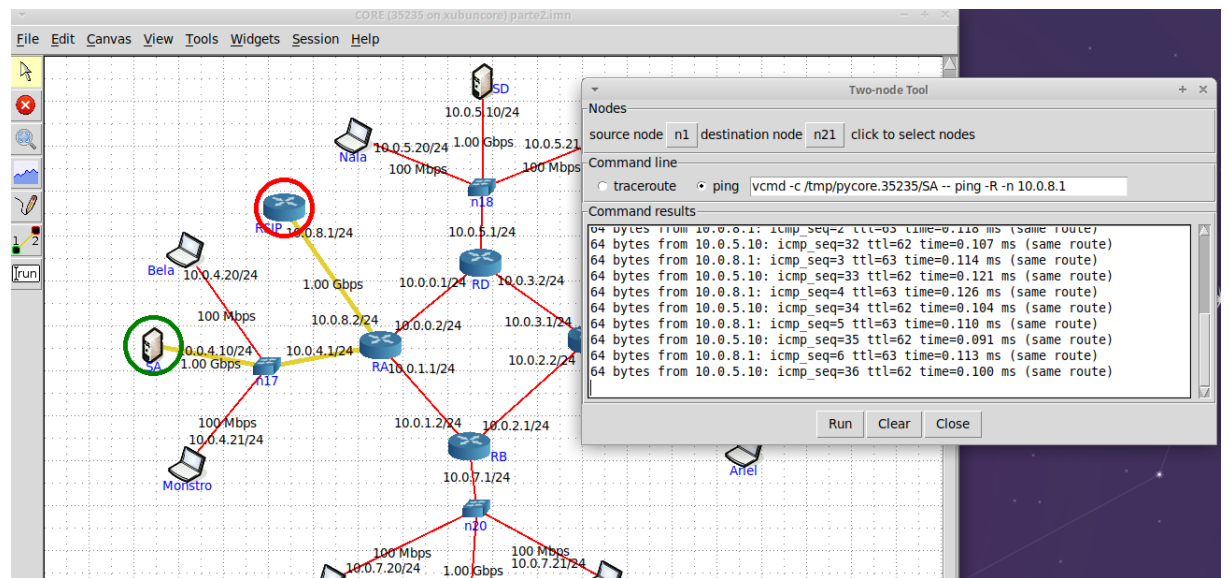


Figura 16. Conectividade entre SA e  $RSIP$

f) Verifique se existe conectividade IP do portátil Bela para o router de acesso  $R_{ISP}$ .

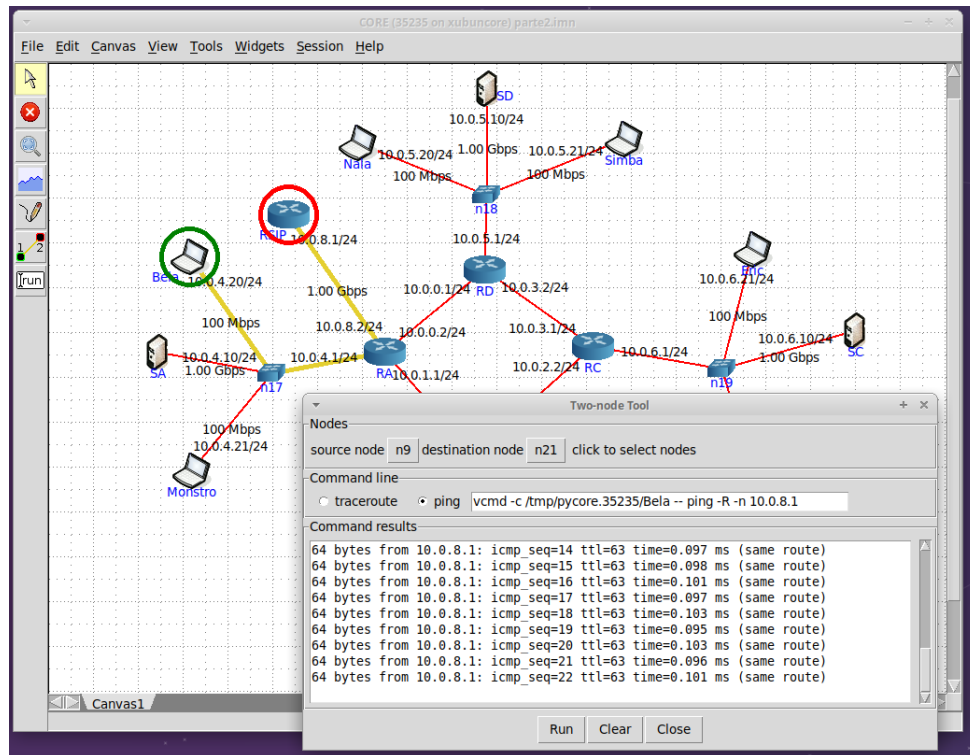


Figura 17. Conectividade entre Bela e  $R_{ISP}$

## 2.2 Router $R_A$ e o portátil Bela

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`).

```
> netstat -rn
> n6 > 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          0.0.0.0          255.255.255.0    U           0 0           0 eth2
10.0.5.0          10.0.0.1         255.255.255.0    UG          0 0           0 eth0
10.0.6.0          10.0.0.1         255.255.255.0    UG          0 0           0 eth0
10.0.7.0          10.0.1.2         255.255.255.0    UG          0 0           0 eth1
10.0.8.0          0.0.0.0          255.255.255.0    U           0 0           0 eth3
> n9 > 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
```

**Figura 18.** Tabelas de encaminhamentos respetivamente  $R_A$  e Bela

**R:** Nesta(s) tabela(s) de encaminhamento(s) podemos verificar um conjunto de colunas e/ou campos que representam os diferentes IP's ligados na mesma rede que o equipamento em que foi efetuado o comando *netstat*. Na figura 18, o n6 representa o *router*  $R_A$  e o n9 representa o *host* Bela.

### Destination

Coluna encarregue de identificar todas as redes ligadas ao equipamento, verificado por *netstat*.

### Gateway

O *Gateway* representa o *next hop* a efetuar de forma a conseguir chegar ao destino pretendido.

### GenMask

Representa a máscara que é utilizada pela rede identificando a rede e o seu *host*. Respetivamente os 3 primeiros e seguidamente o último *byte*.

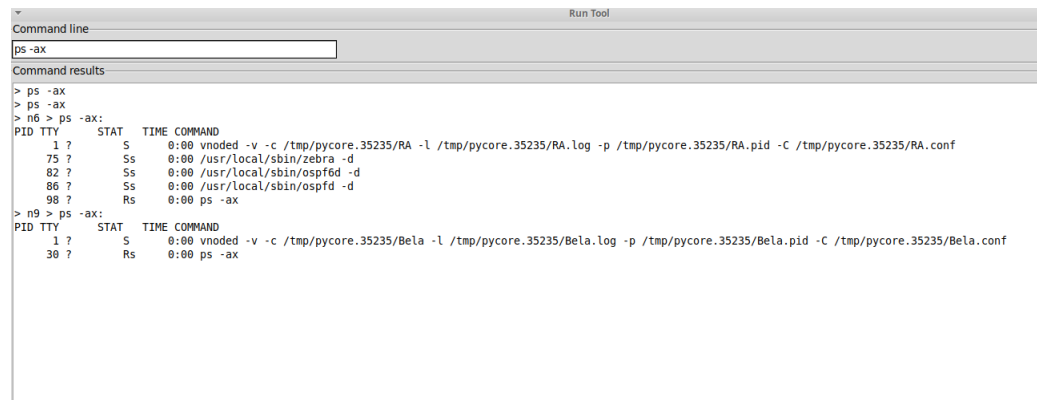


## Flags

O campo de *flags* possui dois tipos de possíveis valores. Sendo eles **U** e **UG** sendo que **U** representa os destinos dos quais são possíveis obter o destino diretamente, sendo assim, inversamente **UG** necessitará de efetuar *hops* para tal.

Podemos verificar, exemplificando, que através de  $R_A$  conseguiríamos aceder a uma rede 10.0.4.0 que será entre por default saindo na *interface* por eth2.

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



```

Command line
ps -ax

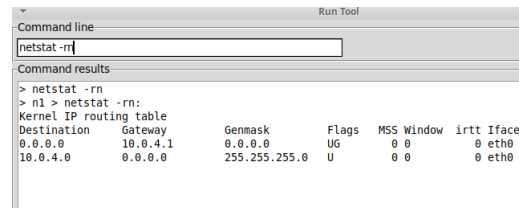
Command results
> ps -ax
> ps -ax
> n6 > ps -ax:
PID TTY STAT TIME COMMAND
1 ? S 0:00 vnodes -v -c /tmp/pycore.35235/RA -l /tmp/pycore.35235/RA.log -p /tmp/pycore.35235/RA.pid -C /tmp/pycore.35235/RA.conf
75 ? Ss 0:00 /usr/local/sbin/zebra -d
82 ? Ss 0:00 /usr/local/sbin/ospf6d -d
86 ? Ss 0:00 /usr/local/sbin/ospf6d -d
98 ? Rs 0:00 ps -ax
> n9 > ps -ax:
PID TTY STAT TIME COMMAND
1 ? S 0:00 vnodes -v -c /tmp/pycore.35235/Bela -l /tmp/pycore.35235/Bela.log -p /tmp/pycore.35235/Bela.pid -C /tmp/pycore.35235/Bela.conf
30 ? Rs 0:00 ps -ax

```

**Figura 19.** Tipos de encaminhamento

**R:** No encaminhamento do  $R_A$  podemos verificar que usufruem de "ospf6d" e de "ospfd" que são tipos de encaminhamento dinâmicos, algo que não conseguimos visualizar em Bela. Na figura 19, o n6 representa o *router*  $R_A$  e o n9 representa o *host* Bela.

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 SA. Use o comando `route delete` para o efeito. Que implicações tem esta medida para os utilizadores da LEI-RC que acedem ao servidor. Justifique.



```

Command line
netstat -rn

Command results
> netstat -rn
> nl > 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

```

**Figura 20.** Antes de apagar a rota por defeito



```

> netstat -rn
> nl > 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

```

**Figura 21.** Depois de apagar a rota por defeito

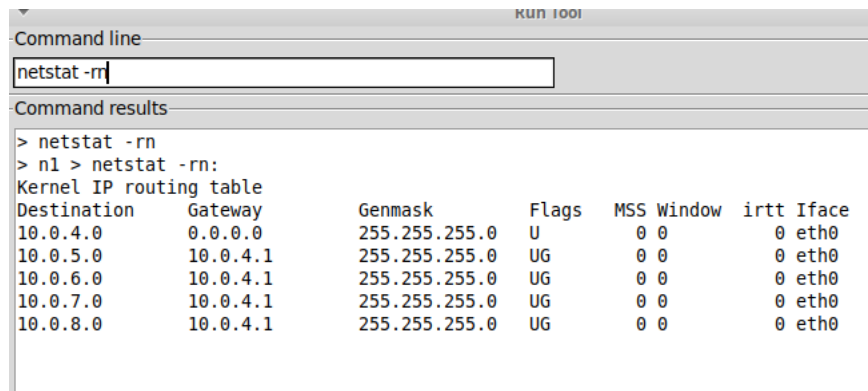
**R:** Eliminando a rota por defeito, estamos a criar vários problemas para os utilizadores. Continua a ser de igual forma possível obter conectividade entre o mesmo departamento porém fora do mesmo tornou-se assim impossível não havendo mais identificadores de rede.

d) Não volte a repor a rota por defeito. Adicione todas as rotas estáticas necessárias para restaurar a conectividade para o servidor SA, por forma a contornar a restrição imposta na alínea c). Utilize para o efeito o comando `route add` e registe os comandos que usou.

**R:** Foram utilizados os comandos:

`route add -net 10.0.x.0 netmask 255.255.255.0 gw 10.0.4.1,  $x \in \{5, 6, 7, 8\}$`

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.



```

Command line
netstat -rn

Command results
> netstat -rn
> n1 > 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
10.0.8.0         10.0.4.1       255.255.255.0   UG      0  0      0 eth0
  
```

Figura 22. Tabela de encaminhamentos SA

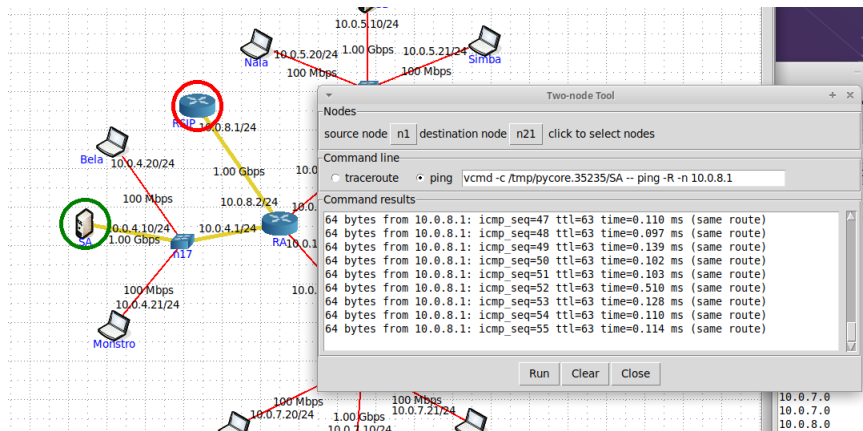


Figura 23. Ping do SA ao  $R_{ISP}$

### 2.3 Definição de Sub-redes

1) Considere que dispõe apenas do endereço de rede IP 192.168.XXX.128/25, em que XXX é o decimal correspondendo ao seu número de grupo (PLXX). Defina um novo esquema de endereçamento para as redes dos departamentos (mantendo as redes de acesso externo e backbone inalteradas), sabendo que o número de departamentos pode vir a aumentar no curto prazo. Atribua endereços às interfaces dos vários sistemas envolvidos. Assuma que todos os endereços de sub-redes são usáveis. Justifique as opções tomadas no planeamento.

**R:** O nosso valor de **XX** é consequentemente 65 .

Uma vez que uma subrede é definida com 32 bits e atualmente estamos a usufruir de 25,  $32 - 25 = 7$ . Sobrando 7 subredes para subredes sabendo que estamos a utilizar 4 subredes, i.e. 4 departamentos, sobram nos 3 subredes, algo possível portanto de representar em apenas 2 bits. Uma vez que há a visão de posteriormente poderá ser preciso mais do que 3 subredes poderá ser reservado mais 1 bit.

2) Qual a máscara de rede que usou (em formato decimal)? Quantos hosts IP pode interligar em cada departamento? Quantos prefixos de sub-rede ficam disponíveis para uso futuro? Justifique.

**R:** O IP seria 192.168.65.128/25.

Com os restantes 4 bits para representar os hosts, ficamos com  $2^4$  endereços. Com estes 16 endereços, ainda nos resta retirar 2 endereços, os quais reservados, finalizando assim com 14 endereços utilizáveis. Quanto aos prefixos de sub-rede, podendo usar 3 bits ficamos com  $2^3 - 1$  subredes, pelo que a nossa máscara obterá um resultado final de: 192.168.65.128/25.

3) Verifique e garanta que a conectividade IP interna na rede local LEI-RC é mantida. No caso de não existência de conectividade, reveja a atribuição de endereços efetuada e eventuais erros de encaminhamento por forma a realizar as correções necessárias. Explique como procedeu.

**R:** Para verificar a conectividade IP da rede apenas fazemos *ping* de um computador para outro, estando ambos em departamentos diferentes, verificamos assim que não ocorreram qualquer tipo de problemas.

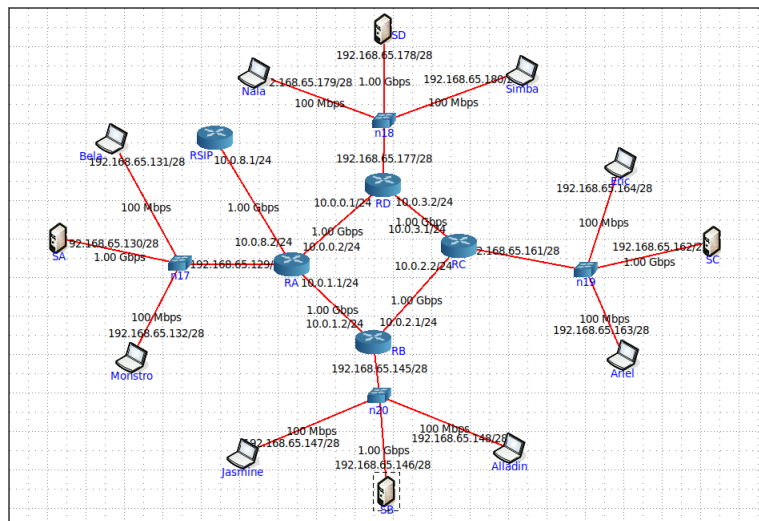


Figura 24. Topologia proveniente do *subnetting*

### 3 Conclusão

Na realização deste TP2, sentimos alguma dificuldade principalmente no entendimento do enunciado. Não que o mesmo fosse complexo mas por uma mistura de conceitos e métodos em alcançar o pretendido. Após obter essa percepção, conjugado com as aulas práticas tornou-se tudo bastante mais claro ajudando assim a entender na prática o pretendido pela parte teórica.

Outra grande dificuldade foi inicialmente aprender a usar o Core e o Wireshark. Sentimos mais dificuldade na organização e compreensão de alguns dos resultados obtidos no CORE, principalmente a tabela de processos do `ps -ax`, com a ajuda dos docentes essa compreensão foi crucial na resolução dos exercícios.