Universidade do Minho
Mestrado Integrado em Engenharia Informática

# REDES DE COMPUTADORES
# TP2
# PL 110

Gonçalo de Sá Quental Rosa Medeiros A89514
José Pedro Castro Ferreira A89572
Rui Emanuel Gomes Vieira A89564

25 de novembro de 2020

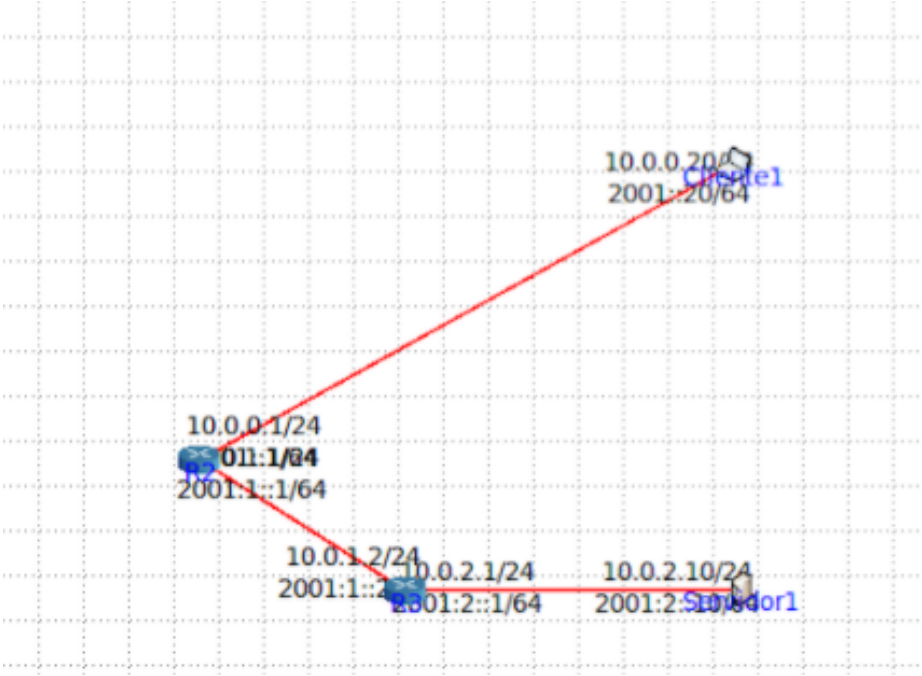

A89514


A89572


A89564

# 1 Protocolo IP: Parte 1

Prepare uma topologia CORE para verificar o comportamento do *traceroute*.



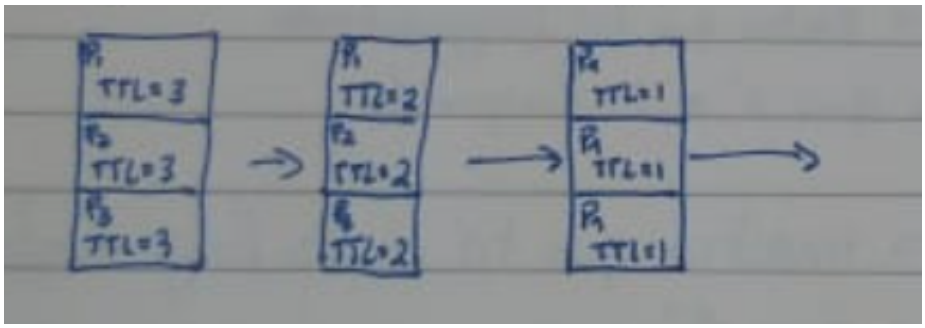

## 1.1 Pergunta 1

### 1.1.1 Active o *wireshark* ou o *tcpdump* no Cliente1. Numa *shell* do Cliente1, execute o comando *traceroute* -I para o endereço IP do Servidor1.

```
root@Cliente1:/tmp/pycore.37311/Cliente1.conf# traceroute -I 10.0.2.10
traceroute to 10.0.2.10 (10.0.2.10), 30 hops max, 60 byte packets
 1  10.0.0.1 (10.0.0.1)  0.046 ms  0.011 ms  0.009 ms
 2  10.0.1.2 (10.0.1.2)  0.023 ms  0.013 ms  0.012 ms
 3  10.0.2.10 (10.0.2.10)  0.024 ms  0.016 ms  0.016 ms
```

### 1.1.2 Registe e analise o tráfego ICMP enviado pelo Cliente1 e o tráfego ICMP recebido como resposta. Comente os resultados face ao comportamento esperado.

São enviados 3 *packets* de cada vez com *TTL* incrementando a cada envio. Seja uma *queue* da seguinte forma:




Entre o cliente e o servidor são precisos 3 saltos, assim sendo, os 6 primeiros *packets* devem ser alvos de *drop* (os 3 primeiros em R2 e os 3 segundos em R3). Apenas os últimos 3 (e todos os seguintes) conseguirão chegar ao servidor, e este vai enviar uma resposta de volta.

#### 1.1.3 Qual deve ser o valor inicial mínimo do campo *TTL* para alcançar o Servidor1?

O *TTL* terá que ter um valor inicial no mínimo de 3 para poder comunicar com o Servidor1.

#### 1.1.4 Calcule o valor médio do tempo de ida-e-volta (*Round-Trip Time*) obtido

$\frac{0.024+0.016+0.016}{3} = \frac{0.056}{3} = 0.01867$ ms

### 1.2 Pergunta 2

Pretende-se agora usar o *traceroute* na sua máquina nativa, e gerar datagramas IP de diferentes tamanhos.

#### 1.2.1 Qual é o endereço IP da *interface* ativa do seu computador?

172.26.77.222

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

O valor do campo do protocolo é 0x01 e identifica o protocolo ICMP.

#### 1.2.3 Quantos *bytes* tem o cabeçalho IP(v4)? Quantos *bytes* tem o campo de dados (*payload*) do datagrama? Como se calcula o tamanho do *payload*?

Há 20 bytes no *header* do IP e o total de *bytes* é de 56, deixando então 36 *bytes* de *payload*.

#### 1.2.4 O datagrama IP foi fragmentado? Justifique

O *bit* de *More fragments* está igual a zero, portanto podemos concluir que os dados não foram fragmentados.

#### 1.2.5 Ordene os pacotes capturados de acordo com o endereço IP fonte, 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 valores que vão alterando de pacote para pacote são a *Identification, header checksum* e *TTL*.

#### 1.2.6 Observa algum padrão nos valores do campo de Identificação do datagrama IP e *TTL*?

*TTL* e ID são incrementados uma vez por cada *echo (ping) request*.

#### 1.2.7 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?

Os primeiros três *packets* têm *ttl* = 255, os segundos *ttl* = 254 e os terceiros *ttl* = 253. Como já vimos anteriormente, os valores do *ttl* variam com o tempo. Esta variação deve-se ao facto que à medida que os packets são enviados pelo *host*, o *ttl* também é incrementado, fazendo com que os mesmos cheguem cada vez mais longe. Quando é preciso emitir uma mensagem de erro, os pacotes vêm de *routers* mais distantes fazendo com que, no caminho de regresso ao *host*, os *packets* passem por mais *routers* intermediários e, por isso, o *ttl* das mensagens é decrementado.

### 1.3 Pergunta 3

Pretende-se agora analisar a fragmentação de pacotes IP.

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

O tamanho do *packet* ultrapassa o tamanho máximo da rede.

#### 1.3.2 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?

*More fragments: set* → Mostra que foi fragmentado
*Offset* = 0 → Mostra que é fragmento
Tamanho = 1500

#### 1.3.3 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?

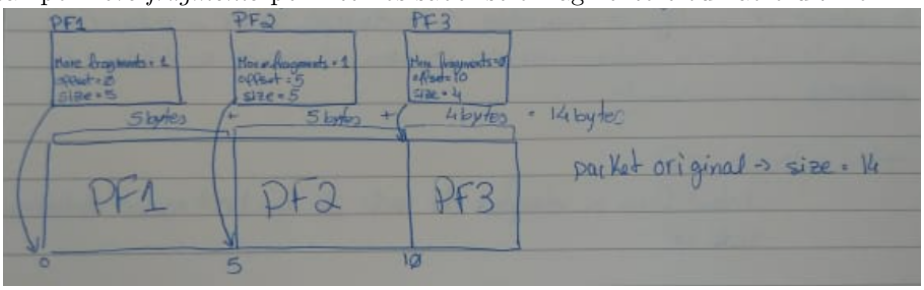
*More fragments: set* → mostra que há mais fragmentos
Como o *Offset* é diferente de zero, concluimos que não se trata do primeiro fragmento.

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

Foram criados 3 fragmentos a partir do datagrama original, o último distingue-se por ter *more fragments* a zero.

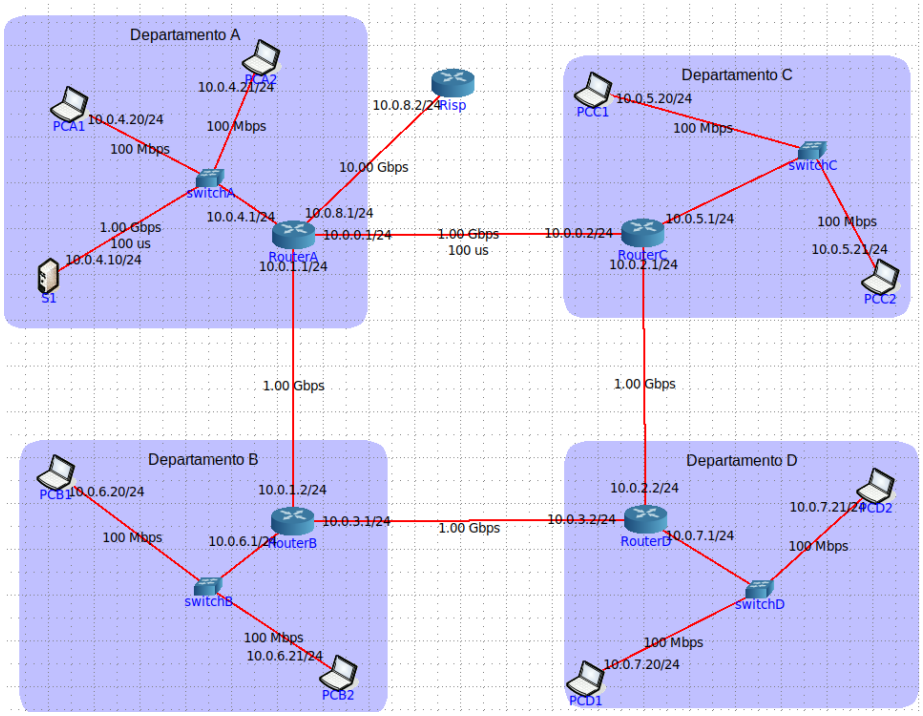
#### 1.3.5 Indique, resumindo, os campos que mudam no cabeçalho IP entre os diferentes fragmentos, e explique a forma como essa informação permite reconstruir o datagrama original.

Os campos sujeitos a mudança são a Identificação, *Fragment offset* e o campo *More fragments*
O *Fragment offset* pertime-nos ter conhecimento da posição inicial do fragmento no *package* original.
O campo *More fragments* permite-nos saber se o fragmento é ou não o último.



# 2 Protocolo IP: Parte 2

Atenda aos endereços IP atribuídos automaticamente pelo CORE aos diversos equipamentos da topologia contruída que reflete a rede local da organização.




## 2.1 Pergunta 1

### 2.1.1 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 endereço usado.

Podemos observar pelo esquema apresentado a cima a topologia definifa e os endereços e, visto que temos todos os endereços a acabar em "/24", a máscara utilizado foi 255.255.255.0.

### 2.1.2 Tratam-se de endereços públicos ou privados? Porquê?

Pelo facto de todos os endereços começarem em 10., e sabendo que todos os endereços entre 10.0.0.0 e 10.255.255.255 são endereços privados, podemos concluir que todos os endereços são privados.

### 2.1.3 Porque razão não é atribuído um endereço IP aos *switches*?

Um *switch* é um equipamento ativo que funciona, normalmente, na camada 2 e tem como principal função interligar equipamentos dentro de uma rede. Quando o *switch* recebe informação por uma porta, transmite essa informação por todas as outras portas e regista o endereço *MAC* dos dispositivos que estão ligados a cada porta. O *switch* analisa o endereço e envia a informação diretamente para o endereço de destino correspondente. Assim sendo, não é atribuído um endereço IP porque não é necessário, dado que os *switches* apenas reencaminham os pacotes para o destino.

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

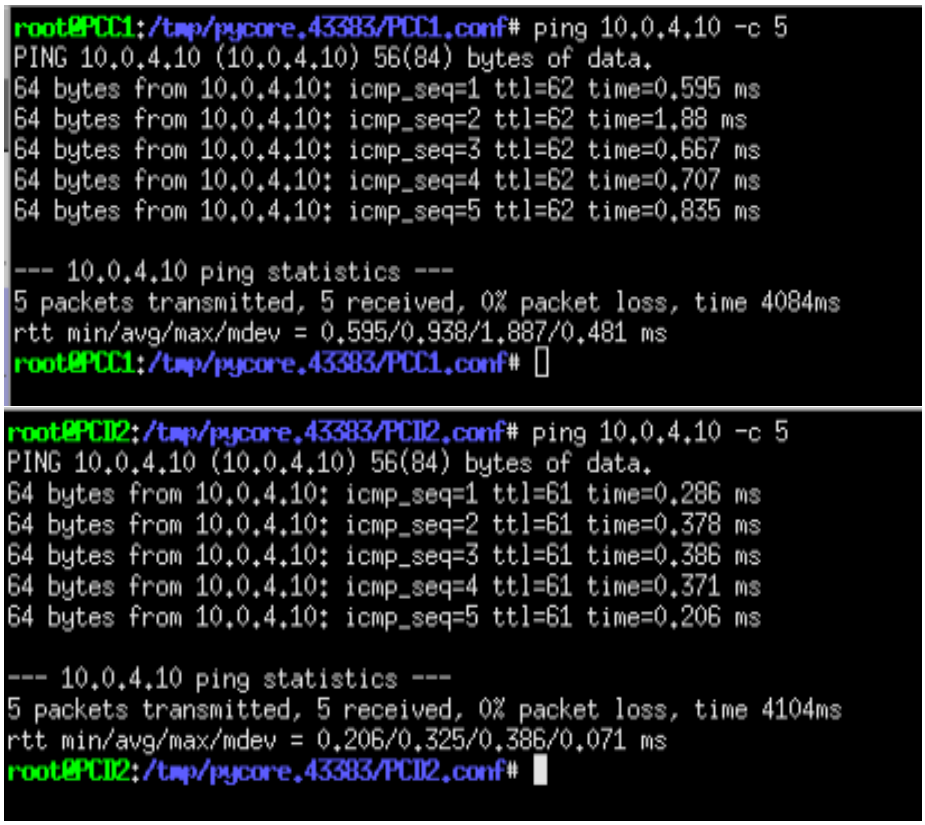
Nesta e na próxima alínea, o comando *ping* foi limitado a enviar 5 *packets* para tornar mais legível e facilitar a compreensão do *output* do comando.

```
root@PCA1:/tmp/pycore.43383/PCA1.conf# ping 10.0.4.10 -c 5
PING 10.0.4.10 (10.0.4.10) 56(84) bytes of data.
64 bytes from 10.0.4.10: icmp_seq=1 ttl=64 time=0.253 ms
64 bytes from 10.0.4.10: icmp_seq=2 ttl=64 time=0.617 ms
64 bytes from 10.0.4.10: icmp_seq=3 ttl=64 time=7.65 ms
64 bytes from 10.0.4.10: icmp_seq=4 ttl=64 time=4.74 ms
64 bytes from 10.0.4.10: icmp_seq=5 ttl=64 time=4.28 ms

--- 10.0.4.10 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4052ms
rtt min/avg/max/mdev = 0.253/3.511/7.654/2.767 ms
root@PCA1:/tmp/pycore.43383/PCA1.conf#
```

```
root@PCB1:/tmp/pycore.43383/PCB1.conf# ping 10.0.4.10 -c 5
PING 10.0.4.10 (10.0.4.10) 56(84) bytes of data.
64 bytes from 10.0.4.10: icmp_seq=1 ttl=62 time=0.652 ms
64 bytes from 10.0.4.10: icmp_seq=2 ttl=62 time=0.178 ms
64 bytes from 10.0.4.10: icmp_seq=3 ttl=62 time=0.201 ms
64 bytes from 10.0.4.10: icmp_seq=4 ttl=62 time=0.222 ms
64 bytes from 10.0.4.10: icmp_seq=5 ttl=62 time=0.397 ms

--- 10.0.4.10 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4098ms
rtt min/avg/max/mdev = 0.178/0.330/0.652/0.178 ms
root@PCB1:/tmp/pycore.43383/PCB1.conf#
```




Como todos os *packets* transmitidos a partir dos *laptops* de cada departamento são recebidos no destino, garantimos que há conetividade entre todos os departamentos.

#### 2.1.5 Verifique se existe conectividade IP do *router* de acesso *Risp* para o servidor S1

```
root@S1:/tmp/pycore.36869/S1.conf# ping 10.0.8.2 -c 5
PING 10.0.8.2 (10.0.8.2) 56(84) bytes of data.
64 bytes from 10.0.8.2: icmp_seq=1 ttl=63 time=0.584 ms
64 bytes from 10.0.8.2: icmp_seq=2 ttl=63 time=22.3 ms
64 bytes from 10.0.8.2: icmp_seq=3 ttl=63 time=36.1 ms
64 bytes from 10.0.8.2: icmp_seq=4 ttl=63 time=19.1 ms
64 bytes from 10.0.8.2: icmp_seq=5 ttl=63 time=10.3 ms

--- 10.0.8.2 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4027ms
rtt min/avg/max/mdev = 0.584/17.726/36.188/11.938 ms
root@S1:/tmp/pycore.36869/S1.conf#
```

Existe e verifica-se através da execução do comando "ping *endereço_de_RISP*", sendo que todos packets enviados são recebidos com sucesso.

### 2.2 Pergunta 2

Para o *router* e um *laptop* do departamento C.

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

```
root@PC1:/tmp/pycore.38215/PC1.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
```

Como podemos por esta tabela referente a um *laptop* C apenas temos a entrada referente ao endereço default (destination 0.0.0.0) sendo este usado em situações onde não é certo o local de envio de um pacote,e a outra com a linha destination 10.0.5.0 que lida com os pacotes que são enviados pelo *laptop* para a própria rede.

```
root@RouterC:/tmp/pycore.38215/RouterC.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
10.0.2.0        0.0.0.0         255.255.255.0   U         0 0          0 eth1
10.0.5.0        0.0.0.0         255.255.255.0   U         0 0          0 eth2
```

Esta tabela refernte ao *router* do departamento C possui as redes a que este tem ligação dando assim os seus *gateways*, consequentemente, quando um pacote tem como destino uma das resdes este é enviado para o valor do *gateway*

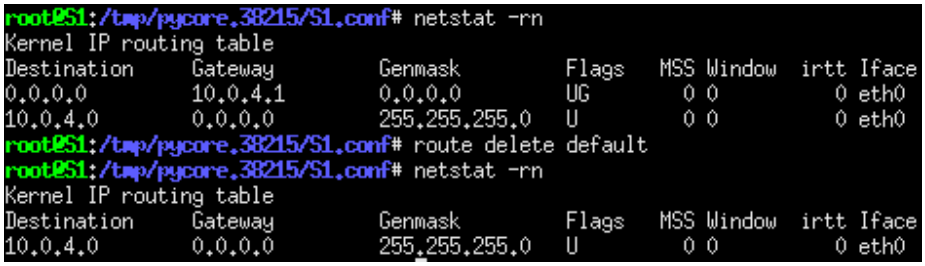
correspondente.

### 2.2.2 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).


Pelo facto de não existirem processos a correr para além dos do próprio *laptop* do departamento C, como podemos observar na imagem a cima, podemos deduzir que as rotas permanecem fixas ao longo do tempo e, por isso, esta máquina está a usar encaminhamento estático.


A presença de protocolos de atualização de rotas ao longo do tempo, como os OSPF's e ZEBRA, possibilita-nos a conclusão de que o *router* do departamento C está a utilizar encaminhamento dinâmico.

### 2.2.3 Admita que, por questões administrativas, a rota por defeito (0.0.0.0 ou *default*) deve ser retirada definitivamente da tabela de encaminhamento do servidor S1 localizado no departamento A. Use o comando route *delete* para o efeito. Que implicações tem esta medida para oa utilizadores da organização *MIEI-RC* que acedem ao servidor. Justifique




Através do uso do comando "route delete default"podemos ver que apenas os laptops do departamento A e continuam a ter acesso ao servidor pelo facto de estarem ligados pelo mesmo *switch*, sendo, deste modo, todas as outras ligações que existiam anteriormente inexistentes.

**2.2.4 Adicione as rotas estáticas necessárias para restaurar a conectividade para o servidos S1, por forma a contornar a restrição imposta na alínea c). Utiliza para o efeito o comando *route add* e registe os comandos que usou.**

```
root@S1:/tmp/pycore.38215/S1.conf# route add -net 10.0.5.0 netmask 255.255.255.0
 gw 10.0.4.1
root@S1:/tmp/pycore.38215/S1.conf# route add -net 10.0.6.0 netmask 255.255.255.0
 gw 10.0.4.1
root@S1:/tmp/pycore.38215/S1.conf# route add -net 10.0.7.0 netmask 255.255.255.0
 gw 10.0.4.1
root@S1:/tmp/pycore.38215/S1.conf# route add -net 10.0.8.0 netmask 255.255.255.0
 gw 10.0.4.1
```

Mediante os comandos utilizados na imagem a cima foram restauradas as rotas entre os vários departamentos (B,C e D) e o router de acesso com o servidor.

**2.2.5 Teste a nova política de encaminhamento garantido que o servidor está novamente acessível, utilizando para o efeito o comando *ping*. Registe a nova tabela de encaminhamento do servidor.**

```
root@PCB1:/tmp/pycore.41567/PCB1.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_seq=1 ttl=62 time=0.285 ms
64 bytes from 10.0.4.10: icmp_seq=2 ttl=62 time=0.459 ms
64 bytes from 10.0.4.10: icmp_seq=3 ttl=62 time=3.38 ms
64 bytes from 10.0.4.10: icmp_seq=4 ttl=62 time=0.249 ms
^C
--- 10.0.4.10 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3062ms
rtt min/avg/max/mdev = 0.249/1.094/3.385/1.325 ms
```

```
root@PCC1:/tmp/pycore.41567/PCC1.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_seq=1 ttl=62 time=5.11 ms
64 bytes from 10.0.4.10: icmp_seq=2 ttl=62 time=14.1 ms
64 bytes from 10.0.4.10: icmp_seq=3 ttl=62 time=9.43 ms
64 bytes from 10.0.4.10: icmp_seq=4 ttl=62 time=4.54 ms
^C
--- 10.0.4.10 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3006ms
rtt min/avg/max/mdev = 4.545/8.303/14.122/3.856 ms
```

```
root@PCD1:/tmp/pycore.41567/PCD1.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_seq=1 ttl=61 time=0.271 ms
64 bytes from 10.0.4.10: icmp_seq=2 ttl=61 time=5.94 ms
64 bytes from 10.0.4.10: icmp_seq=3 ttl=61 time=8.00 ms
64 bytes from 10.0.4.10: icmp_seq=4 ttl=61 time=26.9 ms
^C
--- 10.0.4.10 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3040ms
rtt min/avg/max/mdev = 0.271/10.292/26.953/10.027 ms
```

```
root@Risp:/tmp/pycore.41567/Risp.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_seq=1 ttl=63 time=0.208 ms
64 bytes from 10.0.4.10: icmp_seq=2 ttl=63 time=15.0 ms
64 bytes from 10.0.4.10: icmp_seq=3 ttl=63 time=3.90 ms
64 bytes from 10.0.4.10: icmp_seq=4 ttl=63 time=5.71 ms
^C
--- 10.0.4.10 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3036ms
rtt min/avg/max/mdev = 0.208/6.230/15.091/5.487 ms
```

Pelas imagens anteriormente apresentadas podemos ver que existe trocas de *packets* entre *laptops* de diferentes departamentos e do router Risp com o servidor, comprovando assim o reestabelecimento das ligações.

```
root@S1:/tmp/pycore.38215/S1.conf# netstat -rn
Kernel IP routing table
Destination     Gateway         Genmask         Flags   MSS Window  irtt Iface
10.0.4.0        0.0.0.0         255.255.255.0   U         0 0          0 eth0
10.0.5.0        10.0.4.1        255.255.255.0   UG        0 0          0 eth0
10.0.6.0        10.0.4.1        255.255.255.0   UG        0 0          0 eth0
10.0.7.0        10.0.4.1        255.255.255.0   UG        0 0          0 eth0
10.0.8.0        10.0.4.1        255.255.255.0   UG        0 0          0 eth0
```

## 2.3 Pergunta 3

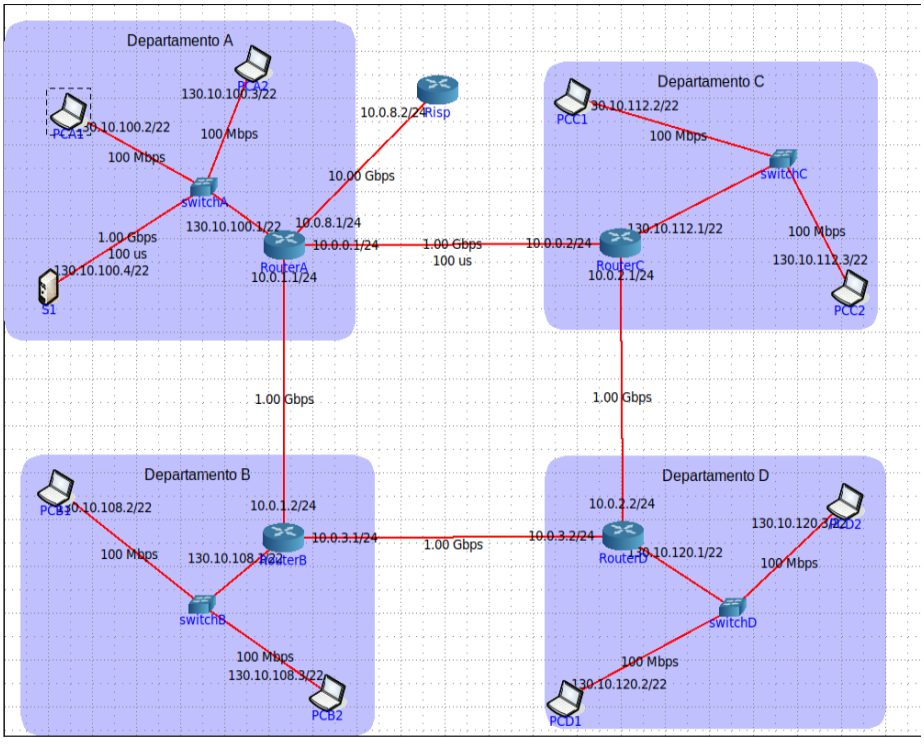
### 2.3.1 Considere que dispõe apenas do endereço de rede IP 130.10.96.0/19. Defina um novo esquema de endereçamento para as redes dos departamentos (mantendo a rede de acesso e core inalteradas) e atribua endereços às *interfaces* dos vários sistemas envolvidos. Assuma que todos os endereços de sub-redes são usáveis. Deve justificar as opções usadas.

O nosso endereço de rede IP é 130.10.96.0/19. Necessitamos criar 4 sub-redes (1 por cada departamento). Para criarmos as sub-redes, utilizamos a expressão 2n-2, tendo de encontrar o menor valor n inteiro para o qual $2n - 24$.
Com n = 2, 22-2 = 2 → não chega
Com n = 3, 23-2 = 6 → chega
Posto isto, vamos precisar de 3 bits para satisfazer as condições necessárias.

A nossa máscara passa a ser /22, devido aos 3 bits que são utilizados para subnetting.

| | | |
|---|---|---|
| 000 | Reservado | |
| 001 | 130.10.100.0/22 | Departamento A |
| 010 | 130.10.104.0/22 | Livre |
| 011 | 130.10.108.0/22 | Departamento B |
| 100 | 130.10.112.0/22 | Departamento C |
| 101 | 130.10.116.0/22 | Livre |
| 110 | 130.10.120/22 | Departamento D |
| 111 | Reservado | |

| Dep. | IP | IP Início | IP Fim |
|---|---|---|---|
| A | 130.10.100.0/22 | 130.10.100.0 | 130.10.103.255 |
| B | 130.10.108.0/22 | 130.10.108.0 | 130.10.111.255 |
| C | 130.10.112.0/22 | 130.10.112.0 | 130.10.115.255 |
| D | 130.10.120.0/22 | 130.10.120.0 | 130.10.123.255 |



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

Tal como foi justificado na alínea anterior, a nossa máscara passou de /19 para /22, uma vez que foram precisos 3 bits para *sub-netting*. Sendo assim, a máscara de rede que usamos, em formato decimal foi: 255.255.252.0. Sabemos então que não podemos mexer nos primeiros 22 *bits*. Restam-nos 10 bits (32-22=10) que podemos alterar. Assim sendo, podemos interligar 210-2=1022 *hosts*. Cada rede tem 2 endereços de reserva: um para *broadcast* e outro que corresponde ao seu endereço IP

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

```
root@PCA1:/tmp/pycore.38911/PCA1.conf# ping 130.10.108.2 -c 5
PING 130.10.108.2 (130.10.108.2) 56(84) bytes of data.
64 bytes from 130.10.108.2: icmp_seq=1 ttl=62 time=0.128 ms
64 bytes from 130.10.108.2: icmp_seq=2 ttl=62 time=0.157 ms
64 bytes from 130.10.108.2: icmp_seq=3 ttl=62 time=0.171 ms
64 bytes from 130.10.108.2: icmp_seq=4 ttl=62 time=0.165 ms
64 bytes from 130.10.108.2: icmp_seq=5 ttl=62 time=0.123 ms

--- 130.10.108.2 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4108ms
rtt min/avg/max/mdev = 0.123/0.148/0.171/0.024 ms
root@PCA1:/tmp/pycore.38911/PCA1.conf# ping 130.10.112.2 -c 5
PING 130.10.112.2 (130.10.112.2) 56(84) bytes of data.
64 bytes from 130.10.112.2: icmp_seq=1 ttl=62 time=0.570 ms
64 bytes from 130.10.112.2: icmp_seq=2 ttl=62 time=2.83 ms
64 bytes from 130.10.112.2: icmp_seq=3 ttl=62 time=4.06 ms
64 bytes from 130.10.112.2: icmp_seq=4 ttl=62 time=1.09 ms
64 bytes from 130.10.112.2: icmp_seq=5 ttl=62 time=11.0 ms

--- 130.10.112.2 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4020ms
rtt min/avg/max/mdev = 0.570/3.913/11.008/3.759 ms
root@PCA1:/tmp/pycore.38911/PCA1.conf# ping 130.10.120.2 -c 5
PING 130.10.120.2 (130.10.120.2) 56(84) bytes of data.
64 bytes from 130.10.120.2: icmp_seq=1 ttl=61 time=0.112 ms
64 bytes from 130.10.120.2: icmp_seq=2 ttl=61 time=0.325 ms
64 bytes from 130.10.120.2: icmp_seq=3 ttl=61 time=0.240 ms
64 bytes from 130.10.120.2: icmp_seq=4 ttl=61 time=0.181 ms
64 bytes from 130.10.120.2: icmp_seq=5 ttl=61 time=0.117 ms

--- 130.10.120.2 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4085ms
rtt min/avg/max/mdev = 0.112/0.195/0.325/0.080 ms
root@PCA1:/tmp/pycore.38911/PCA1.conf#
```

O primeiro *ping* verifica a ligação entre os Departamentos A e B. O segundo igual para A e C e o terceiro de igual modo para A e D. Assim, provamos que existe conexão entre os vários departamentos.