TP4: Protocolo IPv4

Frederico Mendes, João Rocha, and Paulo Sousa

Universidade do Minho, Departamento de Informática, 4710-057 Braga, Portugal
e-mail: {a64331,a64340,a64348}@alunos.uminho.pt

1 Captura de tráfego IP

1.1 Prepare uma topologia CORE para verificar o comportamento do traceroute. Ligue um cliente n1 ao router n2; o n2 a um router n3 que se liga a um router n4 que, por sua vez, se liga ao servidor n5.

Active o *wireshark* ou o *tcpdump* no nó 1.

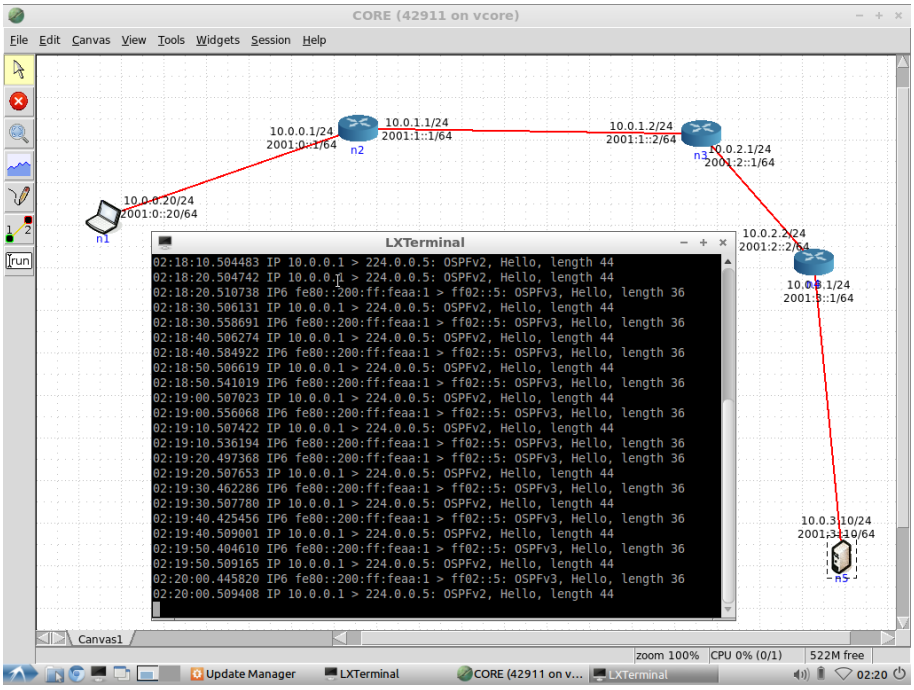

Figura 1. *Print screen* da topologia *CORE*

Numa *shell* do nó 1, execute $traceroute -I$ para o endereço IP do servidor.

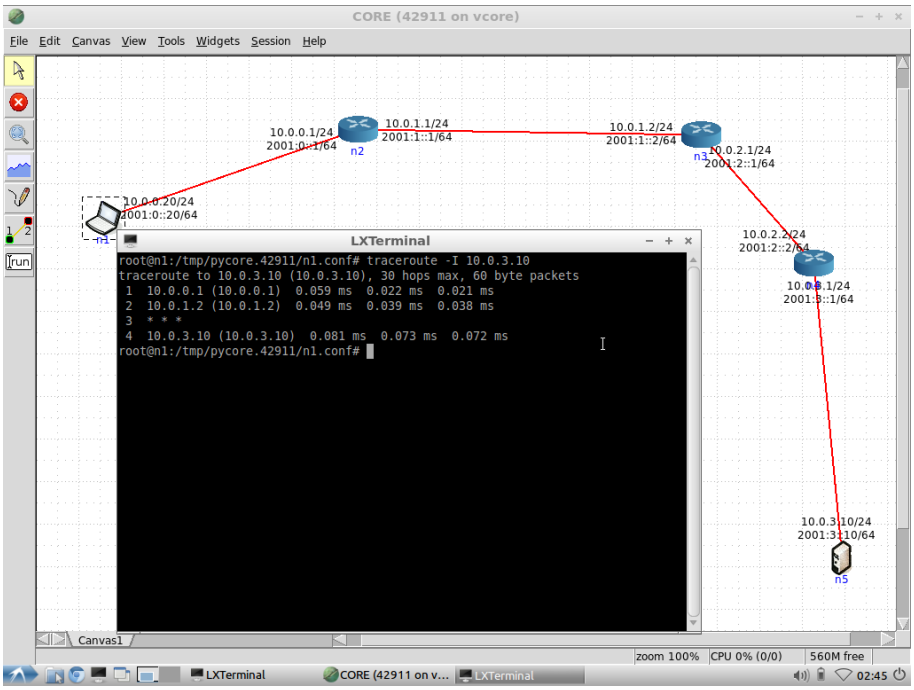

Figura 2. *Print screen* da topologia *CORE*

Analise o tráfego ICMP enviado por n1 e o tráfego ICMP recebido em resposta. Comente os resultados face ao comportamento esperado.

```
10.0.0.20 > 10.0.3.10: ICMP echo request, id 31, seq 1, length 40
10.0.0.1 > 10.0.0.20: ICMP time exceeded in-transit, length 68
```

O cliente envia um datagrama com o campo TTL igual a 1, pelo que tal como se esperava, o primeiro router, após decrementar o TTL, devolve uma mensagem de controlo ICMP por este ter sido excedido.

```
10.0.0.20 > 10.0.3.10: ICMP echo request, id 31, seq 2, length 40
10.0.0.1 > 10.0.0.20: ICMP time exceeded in-transit, length 68
```

Aqui não se esperava que o router devolvesse a mensagem novamente.

```
10.0.0.20 > 10.0.3.10: ICMP echo request, id 31, seq 3, length 40
10.0.0.1 > 10.0.0.20: ICMP time exceeded in-transit, length 68
10.0.0.20 > 10.0.3.10: ICMP echo request, id 31, seq 4, length 40
10.0.1.2 > 10.0.0.20: ICMP time exceeded in-transit, length 68
10.0.0.20 > 10.0.3.10: ICMP echo request, id 31, seq 5, length 40
10.0.1.2 > 10.0.0.20: ICMP time exceeded in-transit, length 68
10.0.0.20 > 10.0.3.10: ICMP echo request, id 31, seq 6, length 40
10.0.1.2 > 10.0.0.20: ICMP time exceeded in-transit, length 68
10.0.0.20 > 10.0.3.10: ICMP echo request, id 31, seq 7, length 40
10.0.0.20 > 10.0.3.10: ICMP echo request, id 31, seq 8, length 40
10.0.0.20 > 10.0.3.10: ICMP echo request, id 31, seq 9, length 40
10.0.0.20 > 10.0.3.10: ICMP echo request, id 31, seq 10, length 40
10.0.3.10 > 10.0.0.20: ICMP echo reply, id 31, seq 10, length 40
10.0.0.20 > 10.0.3.10: ICMP echo request, id 31, seq 11, length 40
```

```
10.0.3.10 > 10.0.0.20: ICMP echo reply, id 31, seq 11, length 40
10.0.0.20 > 10.0.3.10: ICMP echo request, id 31, seq 12, length 40
10.0.3.10 > 10.0.0.20: ICMP echo reply, id 31, seq 12, length 40
10.0.0.20 > 10.0.3.10: ICMP echo request, id 31, seq 13, length 40
10.0.3.10 > 10.0.0.20: ICMP echo reply, id 31, seq 13, length 40
10.0.0.20 > 10.0.3.10: ICMP echo request, id 31, seq 14, length 40
10.0.3.10 > 10.0.0.20: ICMP echo reply, id 31, seq 14, length 40
10.0.0.20 > 10.0.3.10: ICMP echo request, id 31, seq 15, length 40
10.0.3.10 > 10.0.0.20: ICMP echo reply, id 31, seq 15, length 40
10.0.0.20 > 10.0.3.10: ICMP echo request, id 31, seq 16, length 40
10.0.3.10 > 10.0.0.20: ICMP echo reply, id 31, seq 16, length 40
10.0.0.20 > 10.0.3.10: ICMP echo request, id 31, seq 19, length 40
10.0.0.20 > 10.0.3.10: ICMP echo request, id 31, seq 17, length 40
10.0.3.10 > 10.0.0.20: ICMP echo reply, id 31, seq 17, length 40
10.0.0.20 > 10.0.3.10: ICMP echo request, id 31, seq 18, length 40
10.0.3.10 > 10.0.0.20: ICMP echo reply, id 31, seq 18, length 40
10.0.3.10 > 10.0.0.20: ICMP echo reply, id 31, seq 19, length 40
10.0.0.20 > 10.0.3.10: ICMP echo request, id 31, seq 20, length 40
10.0.3.10 > 10.0.0.20: ICMP echo reply, id 31, seq 20, length 40
10.0.0.20 > 10.0.3.10: ICMP echo request, id 31, seq 21, length 40
```

Pode também saber-se quando atingimos o último router quando surgem os *replys*.

Qual deve ser o valor final do campo TTL para alcançar o destino n5? Qual o tempo médio de ida-e-volta (RTT - round-trip time)?

```
root@n1:/tmp/pycore.42917/n1.conf# traceroute -I 10.0.3.10
traceroute to 10.0.3.10 (10.0.3.10), 30 hops max, 60 byte packets
 1  10.0.0.1 (10.0.0.1)  0.093 ms  0.021 ms  0.018 ms
 2  10.0.1.2 (10.0.1.2)  0.040 ms  0.034 ms  0.031 ms
 3  * * *
 4  10.0.3.10 (10.0.3.10)  0.062 ms  0.054 ms  0.054 ms
```

Para chegar ao destino n5 o valor final do campo TTL deve ser 4 para que o valor 0 seja lá atingido.

O tempo médio (ms) entre o envio e a recepção de um pacote entre o cliente e o servidor é dado pela média dos valores lidos no último resultado do comando $traceroute$:

$$\frac{0.062 + 0.054 + 0.054}{3} = 0.0567$$

```
traceroute to piano.dsi.uminho.pt (193.137.8.93), 64 hops max, 2000 byte packets
 1  * * *
 2  * * *
 3  194.65.8.109 (194.65.8.109)  921.151 ms  399.243 ms  410.592 ms
 4  fccn.as1930.gigapix.pt (193.136.251.1)  349.689 ms  339.546 ms  329.670 ms
 5  router3.10ge.cr1.lisboa.fccn.pt (193.137.0.10)  319.839 ms  319.917 ms  329.664
 6  router2.10ge.dwdm.porto.fccn.pt (193.136.1.2)  339.957 ms  359.608 ms  340.285 m
 7  uminho.braga.fccn.pt (193.136.4.100)  339.445 ms  339.388 ms  341.208 ms
 8  * * *
 9  * * *
10  piano.dsi.uminho.pt (193.137.8.93)  334.040 ms  347.648 ms  371.752 ms
```

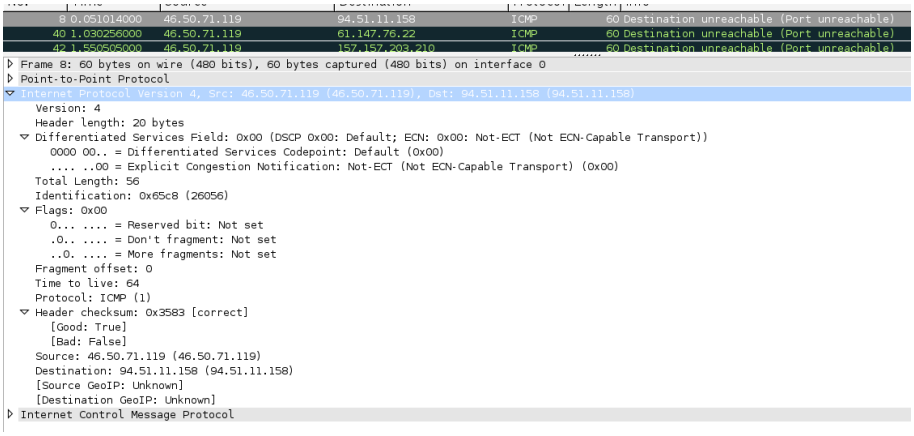

Figura 3. *Print screen* do nível IP da primeira mensagem ICMP capturada pelo *Wireshark*.

Qual é o endereço IP da interface ativa do seu computador? O endereço é o 46.50.71.119.

Qual é o valor do campo protocolo? O que identifica? O valor é 1, que identifica o protocolo ICMP.

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 do IP(v4) tem 20 bytes.

O tamanho definido para os pacotes foi de 2000 bytes, mas este foi fragmentado, correspondendo este datagrama a 1500 bytes.

O payload calcula-se subtraindo o overhead utilizado para o cabeçalho.

$$1500 - 20 = 1480$$

Como são dois datagramas possui dois headers, um em cada datagrama com 20 bytes.

O datagrama foi fragmentado? Justifique. Sim. A transmissão dos 2000 bytes foi feita recorrendo a dois datagramas, pois o MTU[1] foi excedido.

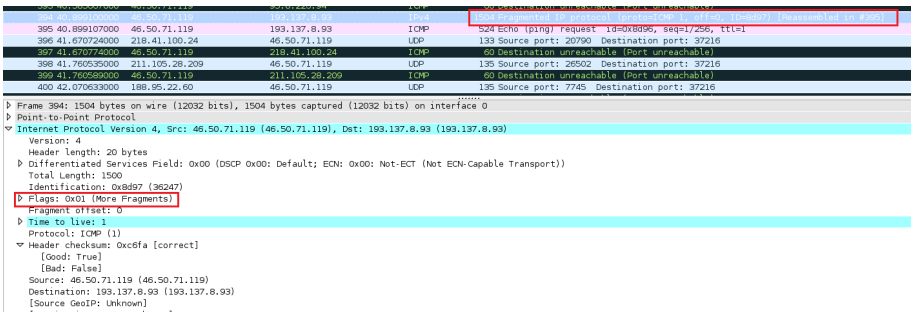

Figura 4. *Print screen* do datagrama fragmentado.

[1] Maximum Transmission Unit

Para transmitir o pacote de dados com o tamanho de predefinição não foi utilizada fragmentação. Para o pacote de tamanho 3500 bytes foram utilizados 3 fragmentos.

A seguir 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 com base no IP gerado na sua máquina. Que campos do datagrama IP mudam sempre na série de mensagens ICMP enviadas pelo seu computador? Os campos do datagrama IP que mudam sempre na série de mensagens ICMP enviadas são o ID, o Header checksum, o seq e a identificação dos fragmentos.

Que campos se mantêm constantes? Que campos se devem manter, preferencialmente, constantes? Porquê? Mantêm-se constantes os campos da versão do protocolo IP, o header length, total length, o tipo do protocolo transportado, *differentiated services field*, e os endereços de destino e fonte do pacote de dados.

É fundamental que os endereços de destino e fonte não mudem para que o encaminhamento seja feito sempre através dos mesmos routers nas diferentes iterações do comando $traceroute$.

Observa algum padrão nos valores do campo de Identificação do datagrama IP? Sim, estes valores são sucessivos incrementos uns em relação aos outros.

A seguir (com os pacotes ordenados por endereço destino) encontre a série de respostas ICMP TTL *exceeded* enviadas ao seu computador pelo primeiro router. Qual é o valor dos campos Identificação e TTL?

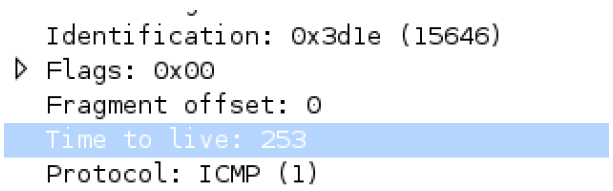

Figura 5. *Print screen* dos valores dos campos de Identificação e TTL.

Esses valores permanecem constantes para todas as mensagens de resposta ICMP TTL *exceeded* enviados pelo primeiro *router* ao seu *host*? Porquê? Os valores do campo de identificação variam entre as mensagens enviadas. Já os campos de TTL mantêm-se constantes para as mensagens provenientes do mesmo *router*.

Os valores de identificação variam (quando não correspondem a fragmentos) pois os datagramas são identificadas univocamente entre si. Já o TTL mantém-se constante pois o *router* é sempre o mesmo, logo o número de saltos dados por essas as mensagens não se altera.

1.2 Pretende-se agora analisar a possível fragmentação de pacotes IP. Reponha a ordem do tráfego capturado usando a coluna do tempo de captura.

Localize a primeira mensagem ICMP depois do tamanho de pacote ter sido definido em 2000 bytes. A mensagem foi fragmentada? Porque é que houve (ou não) necessidade de o fazer? Sim, a mensagem foi fragmentada pois esta excede o MTU, pelo que foi necessário utilizar mais do que um datagrama (neste caso dois) para a sua transmissão.

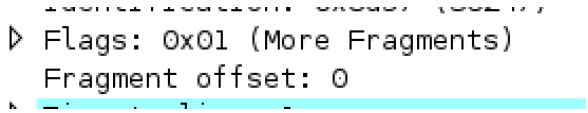

Figura 6. *Print screen* dos campos que identificam a fragmentação.

Imprima o primeiro fragmento do datagrama IP segmentado. Que informação no cabeçalho indica que o datagrama foi fragmentado? Que informação no cabeçalho IP indica se trata do primeiro fragmento? Qual é o tamanho deste datagrama IP? Nas flags é nos indicado se o datagrama corresponde a um fragmento. No campo *Fragment offset* é nos indicada a posição desse fragmento na mensagem a agrupar. Neste caso esse valor está a zero, pelo que nos indica que é o primeiro fragmento.

O datagrama IP tem 20 bytes.

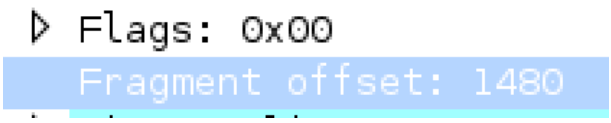

Figura 7. *Print screen* dos campos que identificam o fim da fragmentação.

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? No campo *Fragment offset* é indicado que este fragmento deve ser acoplado ao 1480 byte, pelo que podemos concluir que a mensagem foi fragmentada.

Podemos concluir que não existem mais fragmentos pois o campo das *Flags* está a zero (não é *may fragment* nem *more fragments*).

Procure o primeiro datagrama IP depois do tamanho das mensagens ter sido estabelecido em 3500 bytes. Quantos fragmentos foram criados a partir do datagrama original? Foram criados três fragmentos.

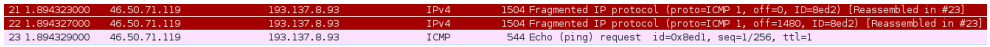

Figura 8. *Print screen* da fragmentação observada aquando da transmissão de um pacote de dados de 3500 bytes.

Indique os campos que mudam no cabeçalho IP entre os diferentes fragmentos, e verifique a forma como essa informação permite reconstruir o datagrama original. O campo *Total length* varia no último datagrama (pois esse fica com o resto da divisão pela MTU). O campo das *Flags* também muda no último fragmento (por ser o último).

O *Fragment offset* também muda pois é este que indica a posição que os dados transportados pelo datagrama devem tomar aquando da reconstituição do datagrama original.

Consequentemente o *Header checksum* também muda.

2 Endereçamento e Encaminhamento IP

2.1 Endereçamento e Encaminhamento IP

Atente aos endereços IP atribuídos automaticamente pelo CORE aos diversos equipamentos da topologia.

Indique que endereços IP e máscaras de rede foram atribuídos automaticamente pelo CORE a cada equipamento. (Pode incluir uma imagem que ilustre de forma clara a topologia e o endereçamento).

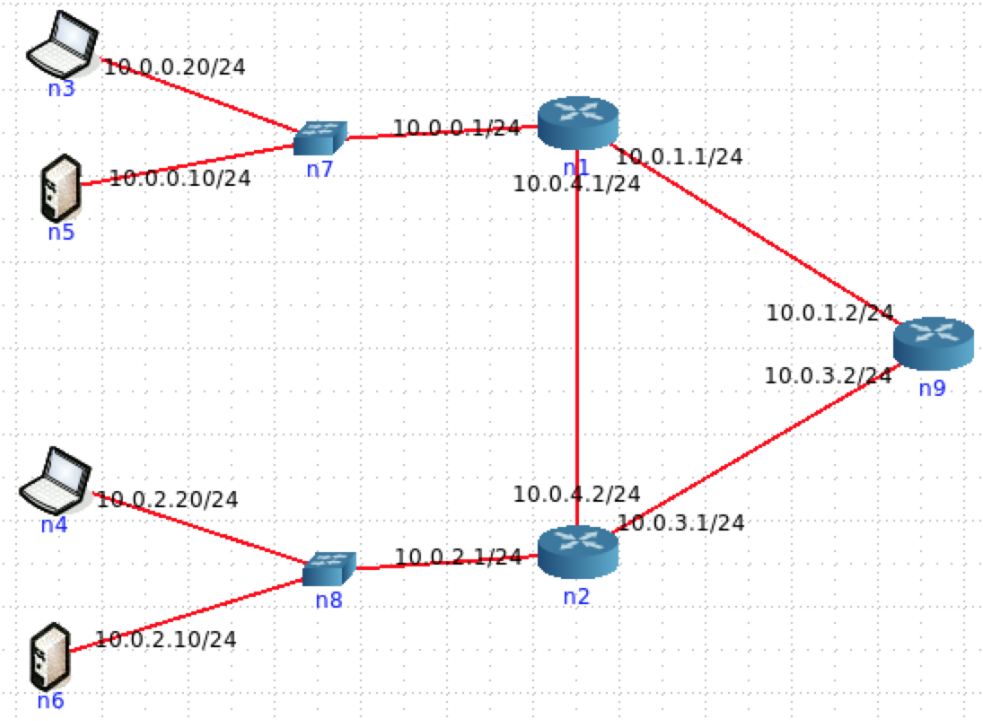

Figura 9. Esquema em *CORE* da topologia descrita.

Tratam-se de endereços públicos ou privados? Porquê? Os endereços são privados pois pertencem ao intervalo [10.0.0.0, 10.255.255.255], que são atribuídos a *Internets* privadas.

Porque razão não é atribuído um endereço IP aos *switches*? Porque os *switches* encaminham pacotes *ethernet* e neste nível não existem endereços IP.

Usando o comando $ping$ certifique-se que existe conectividade total entre os sistemas em ligados em rede.

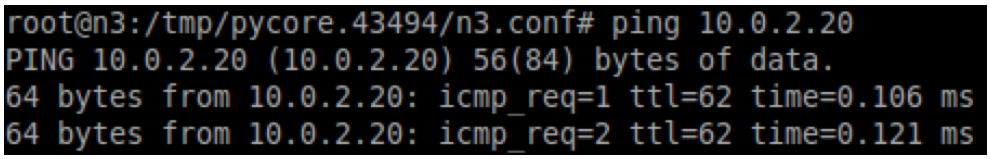

Figura 10. $ping$ do *laptop* n3 para o *laptop* n4.

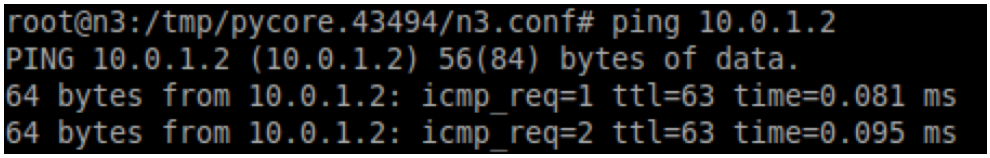

Figura 11. $ping$ do *laptop* n3 para o *router* n9.

Para o router e o laptop de um dos departamentos:

Execute o comando $netstat\ \breve{}rn$ por forma a poder consultar a tabela de encaminhamento. Inclua no seu relatório as tabelas de encaminhamento obtidas; interprete as várias entradas de cada tabela. Se necessário, consulte o manual respetivo ($man\ netstat$).

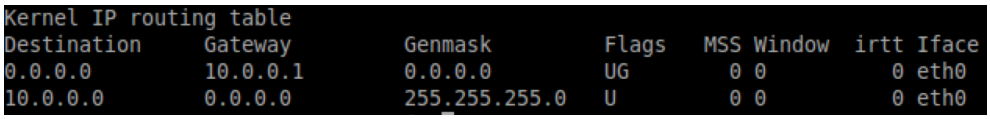

Figura 12. Resultado do comando $netstat\ -rn$ executado no *laptop* n3.

A primeira entrada, relativa ao IP 0.0.0.0 (caminho por defeito quando não há encaminhamento definido na tabela) é encaminhada para a interface do primeiro *router* encontrado (n1).

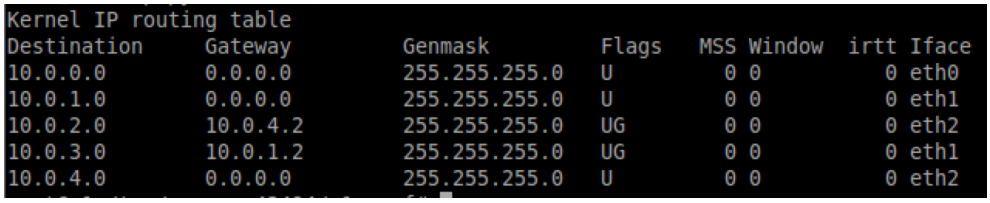

Figura 13. Resultado do comando $netstat$ -rn executado no *router* n1.

Um datagrama destinado à rede 10.0.0.0 será entregue pelo próprio *router*, pois este já tem acesso a ela através da interface *eth1*.

O mesmo se passa para datagramas destinados às redes 10.0.1.0 e 10.0.4.0, pelas interfaces *eth1* e *eth2*, respectivamente.

Já os datagramas destinados à rede 10.0.2.0 deverão ser entregues na interface de endereço 10.0.4.2, pela interface *eth2*.

Os datagramas destinados à rede 10.0.3.0 deverão ser entregues na interface de endereço 10.0.1.2, pela interface *eth1*.

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

```
root@n1:/tmp/pycore.43778/n1.conf# ps -ax
Warning: bad ps syntax, perhaps a bogus '-'? See http://procps.sf.net/faq.html
  PID TTY      STAT   TIME COMMAND
    1 ?        S      0:00 /usr/sbin/vnoded -v -c /tmp/pycore.43778/n1 -l /tmp/p
   50 ?        Ss     0:00 /usr/lib/quagga/zebra -u root -g root -d
   69 ?        Ss     0:00 /usr/lib/quagga/ospf6d -u root -g root -d
   70 ?        Ss     0:00 /usr/lib/quagga/ospfd -u root -g root -d
   76 pts/6    Ss     0:00 /bin/bash
   94 pts/6    R+     0:00 ps -ax
```

Figura 14. Resultado do comando $ps\ -ax$ executado no *router* n1.

Podemos verificar que no *router* existe encaminhamento dinâmico (quer através dos processos que executam, quer na população gradual das tabelas de encaminhamento).

```
root@n3:/tmp/pycore.43778/n3.conf# ps -ax
Warning: bad ps syntax, perhaps a bogus '-'? See http://procps.sf.net/faq.html
  PID TTY      STAT   TIME COMMAND
    1 ?        S      0:00 /usr/sbin/vnoded -v -c /tmp/pycore.43778/n3 -l /tmp/p
   16 pts/4    Ss+    0:00 /bin/bash
   29 pts/10   Ss     0:00 /bin/bash
   40 pts/10   R+     0:00 ps -ax
```

Figura 15. Resultado do comando $ps\ -ax$ executado no *laptop* n3.

Aqui verificámos que os *laptops* apenas utilizam encaminhamento estático.

Admita que, por questões administrativas, a rota por defeito deve ser retirada das tabelas de encaminhamento dos *laptops*. Use o comando $route\ delete$ para esse efeito. Como é afectada a conectividade para cada um dos servidores. Justifique.

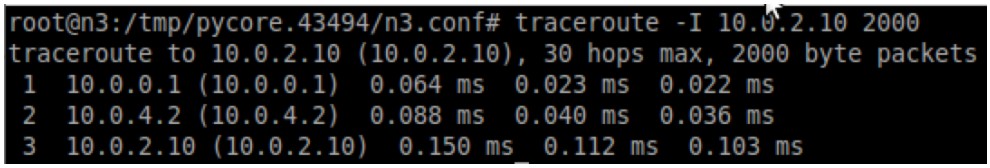

Figura 16. $traceroute$ exemplificativo do encaminhamento feito por defeito entre o *laptop* n3 e o *host* n6.

Depois de eliminar a entrada *default* do *laptop* n3, é impossível comunicarmos com o *host* n6.

```
$ route del -net 0.0.0.0 netmask 0.0.0.0 gw 10.0.0.1
```

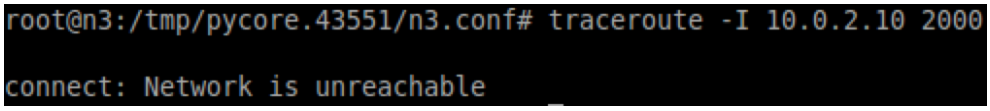

Figura 17. $traceroute$ após eliminar a entrada na tabela de encaminhamento.

Mas o *host* que pertence ao mesmo departamento mantém-se contactável pois a entrada da sua rede não foi eliminada.

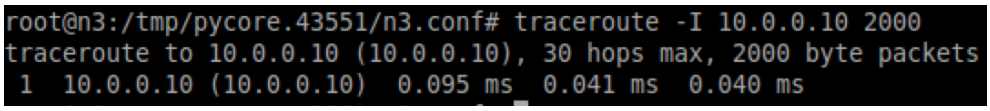

Figura 18. $traceroute$ entre o *laptop* n3 e o *host* n5 (mesmo departamento).

Adicione as rotas estáticas necessárias para repor a conectividade entre os departamentos. Utilize para o efeito o comando $route\ add$. Registe o comando completo que usou.

```
$ route add -net 10.0.2.0 netmask 255.255.255.0 gw 10.0.0.1
```

Assim adicionámos uma entrada que encaminha todo o trafego destinado à rede 10.0.2.0 para o *router* n1, que é capaz de encaminhar os datagramas até ao outro departamento.

Teste a nova política de encaminhamento garantindo que ambos os servidores estão acessíveis, utilizando para o efeito o comando $ping$. Inclua as novas tabelas de encaminhamento dos *laptops*.

```
root@n3:/tmp/pycore.43551/n3.conf# route
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
10.0.0.0        *               255.255.255.0   U     0      0        0 eth0
10.0.2.0        10.0.0.1        255.255.255.0   UG    0      0        0 eth0
```

Figura 19. Tabela de encaminhamento do *laptop* n3 após alterações.

```
root@n3:/tmp/pycore.43551/n3.conf# ping 10.0.2.10
PING 10.0.2.10 (10.0.2.10) 56(84) bytes of data.
64 bytes from 10.0.2.10: icmp_req=1 ttl=62 time=0.144 ms
64 bytes from 10.0.2.10: icmp_req=2 ttl=62 time=0.130 ms
```

Figura 20. $ping$ desde o *laptop* n3 para o *host* n6 (departamentos diferentes).

A conexão é estabelecida.

```
root@n3:/tmp/pycore.43551/n3.conf# ping 10.0.0.10
PING 10.0.0.10 (10.0.0.10) 56(84) bytes of data.
64 bytes from 10.0.0.10: icmp_req=1 ttl=64 time=0.120 ms
64 bytes from 10.0.0.10: icmp_req=2 ttl=64 time=0.077 ms
```

Figura 21. $ping$ desde o *laptop* n3 para o *host* n5 (mesmo departamento).

A conexão é estabelecida.

Que conclui face à atual conectividade externa e interna? A entrada para encaminhamento adicionada ao *laptop* n3 apenas contempla os datagramas destinados à rede 10.0.2.0, pelo que as redes 10.0.1.0 e 10.0.3.0 não estão acessíveis através do mesmo, tornando o *router* n9 inalcancável.

Como este *router* é o que faz o acesso ao exterior, esta conectividade vê-se comprometida.

A conectividade entre os departamentos é, no entanto, garantida.

```
root@n3:/tmp/pycore.43551/n3.conf# traceroute -I 10.0.1.0 2000

connect: Network is unreachable
```

Figura 22. $traceroute$ desde o *laptop* n3 para o *router* n9, que mostra a falta de conectividade.

2.2 Definição de Sub-redes

Assumindo que dispõe apenas de um único endereço de rede IP classe C 192.168.128.0/24, defina um novo esquema de endereçamento para as redes dos departamentos e atribua endereços às interfaces dos vários sistemas envolvidos. Deve justificar as opções usadas.

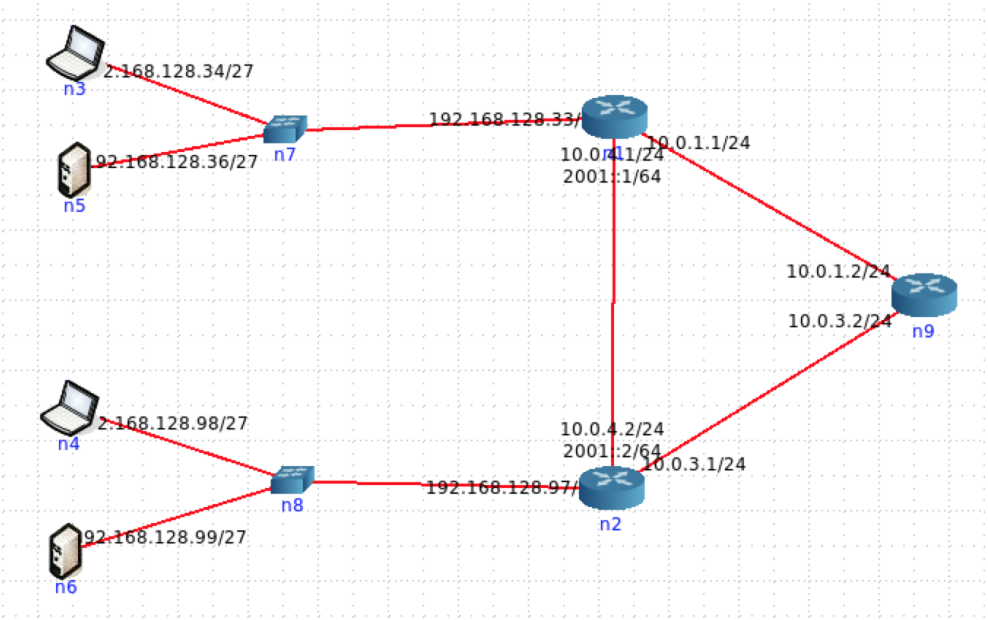

Figura 23. Novo esquema de endereçamento para as redes, e atribuição de endereços às interfaces dos vários sistemas.

Para mapear as sub-redes utilizámos 3 bits, podendo ser os restantes utilizados para o endereçamento dos hosts dentro da sub-rede.

Foram utilizados 3 bits para que uma eventual expansão da empresa (criação de novos departamentos) seja possível, não comprometendo em demasia a escalabilidade do próprio departamento.

Com esta opção podemos endereçar 8 sub-redes e 30 hosts por cada (perfazendo 240 dispositivos).

Qual a máscara de rede a usar (em formato decimal)? 255.255.255.224 (/27)

Com base no novo endereçamento, será possível ao encaminhador de saída anunciar um único prefixo de rede que agregue as redes dos departamentos? Sim, pois um dos três bits é igual em todas as sub-redes.

Que prefixo de rede pode ser exportada para o exterior? O primeiro bit (dos três) não está a ser utilizado pelo endereçamento de sub-rede, pelo que pode ser exportado para o exterior, utilizando a máscara de rede 255.255.255.128 (/25).

Quantos hosts pode interligar em cada departamento? Cada departamento dispõe de 5 bits para endereçar as suas máquinas, pelo que pode interligar 30 hosts (31-1).

Garanta que a conectividade total na rede é mantida.

```
root@n3:/tmp/pycore.43782/n3.conf# traceroute -I 192.168.128.98 2000
traceroute to 192.168.128.98 (192.168.128.98), 30 hops max, 2000 byte packets
 1  192.168.128.33 (192.168.128.33)  0.110 ms  0.023 ms  0.020 ms
 2  10.0.4.2 (10.0.4.2)  0.083 ms  0.039 ms  0.038 ms
 3  192.168.128.98 (192.168.128.98)  0.153 ms  0.116 ms  0.104 ms
```

Figura 24. $traceroute$ entre o *laptop* n3 e o *laptop* n4.

Podemos verificar que existe conectividade entre o *laptop* n3 e o *router* n1, e entre este e o *router* n2, e finalmente entre este e o *laptop* n4.

```
root@n3:/tmp/pycore.43782/n3.conf# traceroute -I 192.168.128.99 2000
traceroute to 192.168.128.99 (192.168.128.99), 30 hops max, 2000 byte packets
 1  192.168.128.33 (192.168.128.33)  0.062 ms  0.056 ms  0.022 ms
 2  10.0.4.2 (10.0.4.2)  0.055 ms  0.038 ms  0.037 ms
 3  192.168.128.99 (192.168.128.99)  0.167 ms  0.135 ms  0.108 ms
```

Figura 25. $traceroute$ entre o *laptop* n3 e o *host* n6.

```
root@n3:/tmp/pycore.43782/n3.conf# traceroute -I 10.0.3.2 2000
traceroute to 10.0.3.2 (10.0.3.2), 30 hops max, 2000 byte packets
 1  192.168.128.33 (192.168.128.33)  0.066 ms  0.023 ms  0.022 ms
 2  10.0.3.2 (10.0.3.2)  0.135 ms  0.076 ms  0.074 ms
```

Figura 26. $traceroute$ entre o *laptop* n3 e o *router* n9 (conexão com o exterior).

3 Conclusões

Com a **Parte I** foram-nos introduzidos os conceitos relacionados com protocolo IP. Com a ajuda do *wireshark* pudemos ver com detalhe a implementação e características de aspectos como a fragmentação de pacotes IP, overhead do protocolo, entre outros. Compreendemos as políticas de encaminhamento através do programa $traceroute$, que nos permitiu ver os vários saltos que um datagrama realiza, entendendo para isso detalhes acerca do protocolo *ICMP*.

Na **Parte II**, através de um problema real, pudemos ver o real impacto e implementação do endereçamento IP e de toda a lógica associada. Pudemos ver como o *CORE* distribuiu de forma automática os IPs e de como o encaminhamento funcionava nos vários dispositivos presentes na rede (*laptops*, *routers* e *switches*).

Transformando as tabelas de encaminhamento IP dos diferentes dispositivos, pudemos perceber experimentalmente os efeitos de um bom endereçamento, e do controlo que se pode ter numa rede mudando estes valores.