

UNIVERSIDADE DO MINHO
DEPARTAMENTO DE INFORMÁTICA

TP3 - Camada de ligação lógica: Ethernet e
Protocolo ARP
Redes e Computadores
Grupo 56

Bruno Martins (a80410) Filipe Monteiro (a80229)
Márcio Sousa (a82400)

10 de Dezembro de 2018

Conteúdo

1	Captura e Análise de Tramas Ethernet	2
1.1	Anote os endereços MAC de origem e de destino da trama capturada.	2
1.2	Identifique a que sistemas se referem. Justifique.	3
1.3	Qual o valor hexadecimal do campo Type da trama Ethernet? O que significa? . .	3
1.4	Quantos bytes são usados desde o início da trama até ao caractere ASCII "G" do método HTTP GET? Calcule e indique, em percentagem, a sobrecarga (overhead) introduzida pela pilha protocolar no envio do HTTP GET.	4
1.5	Através de visualização direta de uma trama capturada, verifique que, possivelmente, o campo FCS (Frame Check Sequence) usado para deteção de erros não está a ser usado. Em sua opinião, porque será?	4
1.6	Qual é o endereço Ethernet da fonte? A que sistema de rede corresponde? Justifique	4
1.7	Qual é o endereço MAC do destino? A que sistema corresponde?	4
1.8	Atendendo ao conceito de desencapsulamento protocolar, identifique os vários protocolos contidos na trama recebida.	5
2	Protocolo Arp	5
2.9	Observe o conteúdo da tabela ARP. Diga o que significa cada uma das colunas . .	5
2.10	Qual é o valor hexadecimal dos endereços origem e destino na trama Ethernet que contém a mensagem com o pedido ARP (ARP Request)? Como interpreta e justifica o endereço destino usado?	5
2.11	Qual o valor hexadecimal do campo tipo da trama Ethernet? O que indica?	6
2.12	Qual o valor do campo ARP opcode? O que especifica? Se necessário, consulte a RFC do protocolo ARP http://tools.ietf.org/html/rfc826.html	6
2.13	Identifique que tipo de endereços estão contidos na mensagem ARP? Que conclui?	6
2.14	Explicita que tipo de pedido ou pergunta é feita pelo host de origem?	6
2.15	Localize a mensagem ARP que é a resposta ao pedido ARP efectuado.	6
3	ARP Gratuito	7
3.16	Identifique um pacote de pedido ARP gratuito originado pelo seu sistema. Analise o conteúdo de um pedido ARP gratuito e identifique em que se distingue dos restantes pedidos ARP. Registe a trama Ethernet correspondente. Qual o resultado esperado face ao pedido ARP gratuito enviado?	7
4	Domínio de Colisão	8
4.17	Faça ping de n1 para n2. Verifique com a opção tcpdump como flui o tráfego nas diversas interfaces dos vários dispositivos. Que conclui?	8
4.18	Na topologia de rede substitua o hub por um switch. Repita os procedimentos que realizou na pergunta anterior. Comente os resultados obtidos quanto à utilização de hubs e switches no contexto de controlar ou dividir domínios de colisão. Documente as suas observações e conclusões com base no tráfego observado/capturado.	9
5	Conclusão	9

1 Captura e Análise de Tramas Ethernet

1.1 Anote os endereços MAC de origem e de destino da trama capturada.

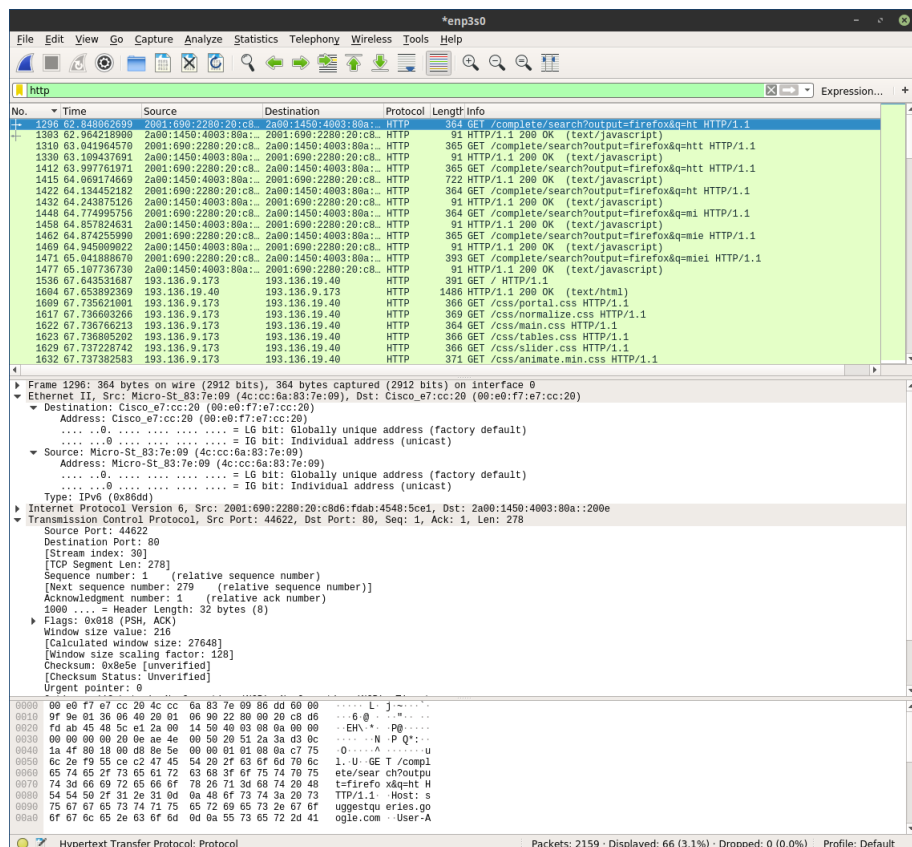


Figura 1: Primeiro trama capturada ao aceder à página miei.di.uminho.pt

O endereço MAC de origem é 4c:cc:6a:83:7e:09.
O endereço MAC do destino é 00:0a:8a:97:74:80.

1.2 Identifique a que sistemas se referem. Justifique.

```
pimonteiro@msi-fuck:~$ ifconfig
enp3s0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 193.136.9.173 netmask 255.255.255.0 broadcast 193.136.9.255
    inet6 fe80::cdf1:3f82:555c:55ad prefixlen 64 scopeid 0x20<link>
    inet6 fd78:2b3:e7b0:4:c8d6:fdab:4548:5ce1 prefixlen 64 scopeid 0x0<global>
    inet6 2001:690:2280:20:c8d6:fdab:4548:5ce1 prefixlen 64 scopeid 0x0<global>
    inet6 2001:690:2280:20:6e05:f02a:225f:6979 prefixlen 64 scopeid 0x0<global>
    inet6 fd78:2b3:e7b0:4:526d:2a31:219e:a1f7 prefixlen 64 scopeid 0x0<global>
    ether 4c:cc:6a:83:7e:09 txqueuelen 1000 (Ethernet)
    RX packets 70334 bytes 60873450 (60.8 MB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 34829 bytes 4076276 (4.0 MB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
    device interrupt 19

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
    inet 127.0.0.1 netmask 255.0.0.0
    inet6 ::1 prefixlen 128 scopeid 0x10<host>
    loop txqueuelen 1000 (Local Loopback)
    RX packets 4401 bytes 405384 (405.3 KB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 4401 bytes 405384 (405.3 KB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

pimonteiro@msi-fuck:~$ host miei.di.uminho.pt
miei.di.uminho.pt is an alias for www6.di.uminho.pt.
www6.di.uminho.pt has address 193.136.19.40
```

(a) Em cima, o IP da nossa máquina (inet) assim como o endereço MAC (ether). Em baixo apresenta o IP correspondente à página da *web* em questão.

```
▼ Ethernet II, Src: Micro-St_83:7e:09 (4c:cc:6a:83:7e:09), Dst: Cisco_97:74:80 (00:0a:8a:97:74:80)
  ▼ Destination: Cisco_97:74:80 (00:0a:8a:97:74:80)
    Address: Cisco_97:74:80 (00:0a:8a:97:74:80)
    ....0. .... = LG bit: Globally unique address (factory default)
    ....0. .... = IG bit: Individual address (unicast)
  ▼ Source: Micro-St_83:7e:09 (4c:cc:6a:83:7e:09)
    Address: Micro-St_83:7e:09 (4c:cc:6a:83:7e:09)
    ....0. .... = LG bit: Globally unique address (factory default)
    ....0. .... = IG bit: Individual address (unicast)
  Type: IPv4 (0x0800)
▶ Internet Protocol Version 4, Src: 193.136.9.173, Dst: 193.136.19.40
```

(b) Endereços MAC do *host* (nossa máquina) e do servidor.

Figura 2: Validação dos endereços de cada máquina.

O endereço MAC de origem corresponde à nossa máquina. O endereço MAC de destino refere-se ao servidor destino (*website*).

1.3 Qual o valor hexadecimal do campo Type da trama Ethernet? O que significa?

```
.....
Type: IPv4 (0x0800)
```

Figura 3: Detalhes do campo *Type* do trama *Ethernet*.

O valor hexadecimal é 0x0800 e significa que o tipo é IPv4.

- 1.4 Quantos bytes são usados desde o início da trama até ao caractere ASCII “G” do método HTTP GET? Calcule e indique, em percentagem, a sobrecarga (overhead) introduzida pela pilha protocolar no envio do HTTP GET.

0000	00 0a 8a 97 74 80 4c cc 6a 83 7e 09 08 00 45 00	...t.L.j~...E.
0010	01 79 e7 b7 40 00 40 06 b1 e1 c1 88 09 ad c1 88	y..@.@.
0020	13 28 e2 d0 00 50 77 f2 96 14 b6 a9 1a 69 80 18	(...Pw.i..
0030	00 e5 74 c0 00 00 01 01 08 0a 08 ff 2c 48 99 17	..t.... ..,H..
0040	97 e8 47 45 54 20 2f 20 48 54 54 50 2f 31 2e 31	GET / HTTP/1.1
0050	0d 0a 48 6f 73 74 3a 20 6d 69 65 69 2e 64 69 2e	..Host: miei.di.
0060	75 6d 69 6e 68 6f 2e 70 74 0d 0a 55 73 65 72 2d	uminho.p t..User-
0070	41 67 65 6a 74 3a 20 4d 6f 7a 60 6c 6c 61 2f 35	Agent: Mozilla/5

Figura 4: Frame da trama. A seguir ao selecionado começam os dados.

Existem 66 bytes de *overhead*. No total do pacote existem 391 bytes. A percentagem de overhead é aproximadamente 16,8%.

- 1.5 Através de visualização direta de uma trama capturada, verifique que, possivelmente, o campo FCS (Frame Check Sequence) usado para deteção de erros não está a ser usado. Em sua opinião, porque será?

O campo FCS não está a ser usado porque como o campo FCS ocupa bastante espaço no datagrama e o protocolo Ethernet já é atualmente mais fiável que no passado, não se envia este, sendo que se for preciso reenvia-se um novo pacote.

- 1.6 Qual é o endereço Ethernet da fonte? A que sistema de rede corresponde? Justifique

▼ Ethernet II, Src: Cisco_97:74:80 (00:0a:8a:97:74:80), Dst: Micro-St_83:7e:09 (4c:cc:6a:83:7e:09)
► Destination: Micro-St_83:7e:09 (4c:cc:6a:83:7e:09)
► Source: Cisco_97:74:80 (00:0a:8a:97:74:80)
Type: IPv4 (0x0800)
► Internet Protocol Version 4, Src: 193.136.19.40, Dst: 193.136.9.173

Figura 5: Trama *Ethernet* com os endereços MAC da fonte e do destino, assim como os IP's respetivos.

O endereço Ethernet da fonte é 00:0a:8a:97:74:80 e corresponde ao router que faz a ligação a uma rede exterior, pois o servidor não se encontra na nossa sub-rede, logo o MAC Address não pode pertencer ao servidor, mas sim ao aparelho de conexão à rede onde se encontra o servidor.

- 1.7 Qual é o endereço MAC do destino? A que sistema corresponde?

▼ Ethernet II, Src: Cisco_97:74:80 (00:0a:8a:97:74:80), Dst: Micro-St_83:7e:09 (4c:cc:6a:83:7e:09)
► Destination: Micro-St_83:7e:09 (4c:cc:6a:83:7e:09)
► Source: Cisco_97:74:80 (00:0a:8a:97:74:80)
Type: IPv4 (0x0800)
► Internet Protocol Version 4, Src: 193.136.19.40, Dst: 193.136.9.173

Figura 6: Trama *Ethernet* com os endereços MAC da fonte e do destino, assim como os IP's respetivos.

O endereço MAC destino é 4c:cc:6a:83:7e:09 e corresponde à nossa máquina.

1.8 Atendendo ao conceito de desencapsulamento protocolar, identifique os vários protocolos contidos na trama recebida.

```

No.      Time            Source            Destination      Protocol Length Info
 89 1.718876454      193.136.19.40      193.136.9.173      HTTP      1486      HTTP/1.1 200 OK (text/html)
Frame 89: 1486 bytes on wire (11888 bits), 1486 bytes captured (11888 bits) on interface 0
Ethernet II, Src: Cisco_97:74:80 (00:0a:8a:97:74:80), Dst: Micro-St_83:7e:09 (4c:cc:6a:83:7e:09)
Internet Protocol Version 4, Src: 193.136.19.40, Dst: 193.136.9.173
Transmission Control Protocol, Src Port: 80, Dst Port: 58064, Seq: 68057, Ack: 326, Len: 1420
[34 Reassembled TCP Segments (69476 bytes): #23(1448), #25(2896), #27(1448), #29(2896), #31(1448), #33(1448), #35(1448), #37(1448), #39(1448), #41(1448), #43(2896), #45(2896), #47(1448), #49(2896), #51(2896), #53(1448), #55(1448), #57(1448)]
Hypertext Transfer Protocol

```

Figura 7: Diferentes encapsulamentos protocolares da trama recebida.

Temos TCP, IPv4, HTTP, Ethernet II.

2 Protocolo Arp

2.9 Observe o conteúdo da tabela ARP. Diga o que significa cada uma das colunas

Address	HWtype	HWaddress	Flags Mask	Iface
gateway	ether	00:d0:03:ff:94:00	C	wlp2s0

Figura 8: Tabela ARP da nossa máquina.

1. *Address* - IP/endereço de uma máquina;
2. *HWtype* - tipo de ligação com essa máquina;
3. *HWaddress* - endereço MAC correspondente;
4. *Flags* - indicam o estado da entrada, ou seja, se foi deduzido, inserido manualmente ou incompleto;
5. *Iface* - indica o nome da *interface* a que o dispositivo está conectado.

2.10 Qual é o valor hexadecimal dos endereços origem e destino na trama Ethernet que contém a mensagem com o pedido ARP (ARP Request)? Como interpreta e justifica o endereço destino usado?

Address	HWtype	HWaddress	Flags Mask	Iface
gw.sa.di.uminho.pt	ether	00:0c:29:d2:19:f0	C	enp3s0
177.143.198.104.bc.goog		(incomplete)		enp3s0

Figura 9: Tabela ARP da nossa máquina após conexão ao servidor.

Endereço origem: 4c:cc:6a:83:7e:09 Endereço destino: ff:ff:ff:ff:ff:ff

O endereço destino é este pois o pedido *ARP Request* usa um endereço MAC de broadcast para todas as máquinas receberem o pacote e eventualmente alguma aceitar, tratar e responder (por ser o IP destino do pacote).

Neste caso, fizemos um **ping 192.168.100.198**, não estando este guardado na tabela de ARP antes do ping.

2.11 Qual o valor hexadecimal do campo tipo da trama Ethernet? O que indica?

```
▼ Ethernet II, Src: Micro-St_83:7e:09 (4c:cc:6a:83:7e:09), Dst: Broadcast (ff:ff:ff:ff:ff:ff)
  ▶ Destination: Broadcast (ff:ff:ff:ff:ff:ff)
  ▶ Source: Micro-St_83:7e:09 (4c:cc:6a:83:7e:09)
  Type: ARP (0x0806)
▼ Address Resolution Protocol (request)
  Hardware type: Ethernet (1)
  Protocol type: IPv4 (0x0800)
  Hardware size: 6
  Protocol size: 4
  Opcode: request (1)
  Sender MAC address: Micro-St_83:7e:09 (4c:cc:6a:83:7e:09)
  Sender IP address: 192.168.100.165
  Target MAC address: 00:00:00_00:00:00 (00:00:00:00:00:00)
  Target IP address: 192.168.100.198
```

Figura 10: Trama c

O valor do campo tipo da trama é de 0x0806 que indica ser um ARP, sendo um ARP Request se analisarmos o *opcode*.

2.12 Qual o valor do campo ARP opcode? O que especifica? Se necessário, consulte a RFC do protocolo ARP <http://tools.ietf.org/html/rfc826.html>

O valor do campo ARP *opcode* é 1 e significa que se trata de um *Arp Request*.

2.13 Identifique que tipo de endereços estão contidos na mensagem ARP? Que conclui?

Na mensagem ARP estão contidos dois tipos de endereços: endereços MAC e endereços IP, que serão usados para encaminhar o pacote para o destino correto e para mais tarde serem adicionados à tabela ARP.

2.14 Explícite que tipo de pedido ou pergunta é feita pelo host de origem?

O *host* de origem pede a quem tiver o IP XXX.XXX.XXX.XXX para lhe responder, ficando assim a saber o endereço MAC da máquina destino.

2.15 Localize a mensagem ARP que é a resposta ao pedido ARP efetuado.

1. Qual o valor do campo ARP opcode? O que especifica?

O valor do campo ARP *opcode* é de 2 que significa ser uma resposta da máquina a um pedido ARP.

2. Em que posição da mensagem ARP está a resposta ao pedido ARP?

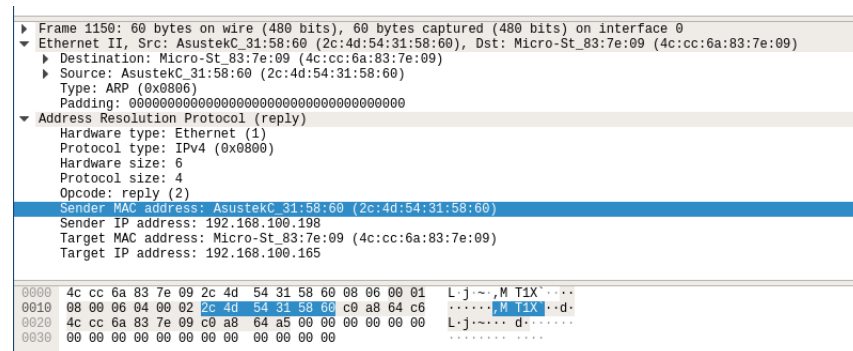


Figura 11: Ping de n1 para n2 (canto superior esquerdo), *tcpdump* de n2 (canto superior direito), *tcpdump* de n3 (canto inferior esquerdo), *tcpdump* de n4 (canto inferior direito)

A resposta encontra-se não entre os *bytes* 23 e 28 (campo *MAC Address*).

3 ARP Gratuito

3.16 Identifique um pacote de pedido ARP gratuito originado pelo seu sistema. Analise o conteúdo de um pedido ARP gratuito e identifique em que se distingue dos restantes pedidos ARP. Registe a trama Ethernet correspondente. Qual o resultado esperado face ao pedido ARP gratuito enviado?

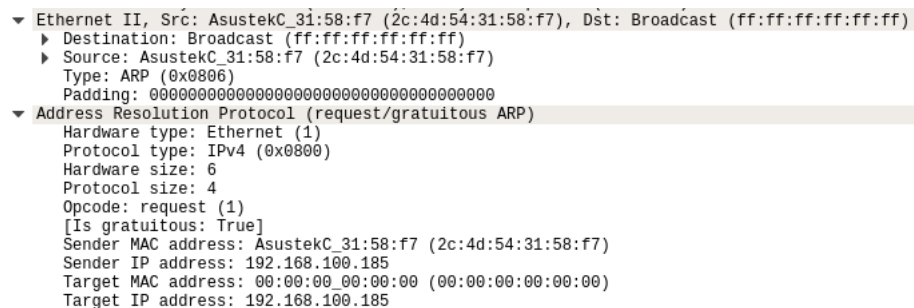


Figura 12: Pedido ARP gratuito

O pedido ARP gratuito distingue-se dos outros porque o Sender IP e o Target IP são o mesmo, o que não acontece normalmente.

4 Domínio de Colisão

4.17 Faça ping de n1 para n2. Verifique com a opção `tcpdump` como flui o tráfego nas diversas interfaces dos vários dispositivos. Que conclui?

```

root@n1: /tmp/pycore.37563/n1.conf
64 bytes from 10.0.0.10: icmp_req=260 ttl=64 time=0.044 ms
64 bytes from 10.0.0.10: icmp_req=261 ttl=64 time=0.126 ms
64 bytes from 10.0.0.10: icmp_req=262 ttl=64 time=0.045 ms
64 bytes from 10.0.0.10: icmp_req=263 ttl=64 time=0.045 ms
64 bytes from 10.0.0.10: icmp_req=264 ttl=64 time=0.073 ms
64 bytes from 10.0.0.10: icmp_req=265 ttl=64 time=0.095 ms
64 bytes from 10.0.0.10: icmp_req=266 ttl=64 time=0.091 ms
64 bytes from 10.0.0.10: icmp_req=267 ttl=64 time=0.086 ms
64 bytes from 10.0.0.10: icmp_req=268 ttl=64 time=0.049 ms
64 bytes from 10.0.0.10: icmp_req=269 ttl=64 time=0.153 ms
64 bytes from 10.0.0.10: icmp_req=270 ttl=64 time=0.104 ms
64 bytes from 10.0.0.10: icmp_req=271 ttl=64 time=0.099 ms
64 bytes from 10.0.0.10: icmp_req=272 ttl=64 time=0.102 ms
64 bytes from 10.0.0.10: icmp_req=273 ttl=64 time=0.101 ms
64 bytes from 10.0.0.10: icmp_req=274 ttl=64 time=0.102 ms
64 bytes from 10.0.0.10: icmp_req=275 ttl=64 time=0.109 ms
64 bytes from 10.0.0.10: icmp_req=276 ttl=64 time=0.149 ms
64 bytes from 10.0.0.10: icmp_req=277 ttl=64 time=0.102 ms
64 bytes from 10.0.0.10: icmp_req=278 ttl=64 time=0.096 ms
64 bytes from 10.0.0.10: icmp_req=279 ttl=64 time=0.108 ms
64 bytes from 10.0.0.10: icmp_req=280 ttl=64 time=0.102 ms
64 bytes from 10.0.0.10: icmp_req=281 ttl=64 time=0.102 ms
64 bytes from 10.0.0.10: icmp_req=282 ttl=64 time=0.096 ms

vcmd
gth 64
10:06:56.233552 IP 10.0.0.10 > 10.0.0.20: ICMP echo reply, id 75, seq 277, len
h 64
10:06:57.234754 IP 10.0.0.20 > 10.0.0.10: ICMP echo request, id 75, seq 278, len
gth 64
10:06:57.234782 IP 10.0.0.10 > 10.0.0.20: ICMP echo reply, id 75, seq 278, len
h 64
10:06:58.234651 IP 10.0.0.20 > 10.0.0.10: ICMP echo request, id 75, seq 279, len
gth 64
10:06:58.234686 IP 10.0.0.10 > 10.0.0.20: ICMP echo reply, id 75, seq 279, len
h 64
10:06:59.234427 IP 10.0.0.20 > 10.0.0.10: ICMP echo request, id 75, seq 280, len
gth 64
10:06:59.234460 IP 10.0.0.10 > 10.0.0.20: ICMP echo reply, id 75, seq 280, len
h 64
10:07:00.233486 IP 10.0.0.20 > 10.0.0.10: ICMP echo request, id 75, seq 281, len
gth 64
10:07:00.233519 IP 10.0.0.10 > 10.0.0.20: ICMP echo reply, id 75, seq 281, len
h 64
10:07:01.234834 IP 10.0.0.20 > 10.0.0.10: ICMP echo request, id 75, seq 282, len
gth 64
10:07:01.234862 IP 10.0.0.10 > 10.0.0.20: ICMP echo reply, id 75, seq 282, len
h 64

vcmd
gth 64
10:06:56.233560 IP 10.0.0.10 > 10.0.0.20: ICMP echo reply, id 75, seq 277, len
h 64
10:06:57.234745 IP 10.0.0.20 > 10.0.0.10: ICMP echo request, id 75, seq 278, len
gth 64
10:06:57.234790 IP 10.0.0.10 > 10.0.0.20: ICMP echo reply, id 75, seq 278, len
h 64
10:06:58.234643 IP 10.0.0.20 > 10.0.0.10: ICMP echo request, id 75, seq 279, len
gth 64
10:06:58.234697 IP 10.0.0.10 > 10.0.0.20: ICMP echo reply, id 75, seq 279, len
h 64
10:06:59.234420 IP 10.0.0.20 > 10.0.0.10: ICMP echo request, id 75, seq 280, len
gth 64
10:06:59.234468 IP 10.0.0.10 > 10.0.0.20: ICMP echo reply, id 75, seq 280, len
h 64
10:07:00.233478 IP 10.0.0.20 > 10.0.0.10: ICMP echo request, id 75, seq 281, len
gth 64
10:07:00.233527 IP 10.0.0.10 > 10.0.0.20: ICMP echo reply, id 75, seq 281, len
h 64
10:07:01.234826 IP 10.0.0.20 > 10.0.0.10: ICMP echo request, id 75, seq 282, len
gth 64
10:07:01.234871 IP 10.0.0.10 > 10.0.0.20: ICMP echo reply, id 75, seq 282, len
h 64

vcmd
gth 64
10:06:56.233563 IP 10.0.0.10 > 10.0.0.20: ICMP echo reply, id 75, seq 277, len
h 64
10:06:57.234750 IP 10.0.0.20 > 10.0.0.10: ICMP echo request, id 75, seq 278, len
gth 64
10:06:57.234793 IP 10.0.0.10 > 10.0.0.20: ICMP echo reply, id 75, seq 278, len
h 64
10:06:58.234647 IP 10.0.0.20 > 10.0.0.10: ICMP echo request, id 75, seq 279, len
gth 64
10:06:58.234701 IP 10.0.0.10 > 10.0.0.20: ICMP echo reply, id 75, seq 279, len
h 64
10:06:59.234424 IP 10.0.0.20 > 10.0.0.10: ICMP echo request, id 75, seq 280, len
gth 64
10:06:59.234472 IP 10.0.0.10 > 10.0.0.20: ICMP echo reply, id 75, seq 280, len
h 64
10:07:00.233483 IP 10.0.0.20 > 10.0.0.10: ICMP echo request, id 75, seq 281, len
gth 64
10:07:00.233530 IP 10.0.0.10 > 10.0.0.20: ICMP echo reply, id 75, seq 281, len
h 64
10:07:01.234831 IP 10.0.0.20 > 10.0.0.10: ICMP echo request, id 75, seq 282, len
gth 64
10:07:01.234873 IP 10.0.0.10 > 10.0.0.20: ICMP echo reply, id 75, seq 282, len
h 64

```

Figura 13: Ping de n1 para n2 (canto superior esquerdo), *tcpdump* de n2 (canto superior direito), *tcpdump* de n3 (canto inferior esquerdo), *tcpdump* de n4 (canto inferior direito)

Analisando o tráfego, concluímos que com *hub*, apesar de estarmos a enviar tráfego de n1 para n2, os outros dispositivos conseguem ver este tráfego. Existe também a possibilidade de colisões, podendo haver a eliminação de pacotes, sendo depois necessário o seu reenvio. Para prevenir esta colisões existe o protocolo CSMA/CD que obriga dispositivos a esperar que outro acabe de comunicar antes de tentarem mandar tráfego.

- 4.18 Na topologia de rede substitua o hub por um switch. Repita os procedimentos que realizou na pergunta anterior. Comente os resultados obtidos quanto à utilização de hubs e switches no contexto de controlar ou dividir domínios de colisão. Documente as suas observações e conclusões com base no tráfego observado/capturado.

```

root@n1: /tmp/pycore.37564/n1.conf
64 bytes from 10.0.0.10: icmp_req=155 ttl=64 time=0,077 ms
64 bytes from 10.0.0.10: icmp_req=156 ttl=64 time=0,076 ms
64 bytes from 10.0.0.10: icmp_req=157 ttl=64 time=0,075 ms
64 bytes from 10.0.0.10: icmp_req=158 ttl=64 time=0,075 ms
64 bytes from 10.0.0.10: icmp_req=159 ttl=64 time=0,075 ms
64 bytes from 10.0.0.10: icmp_req=160 ttl=64 time=0,121 ms
64 bytes from 10.0.0.10: icmp_req=161 ttl=64 time=0,075 ms
64 bytes from 10.0.0.10: icmp_req=162 ttl=64 time=0,075 ms
64 bytes from 10.0.0.10: icmp_req=163 ttl=64 time=0,075 ms
64 bytes from 10.0.0.10: icmp_req=164 ttl=64 time=0,078 ms
64 bytes from 10.0.0.10: icmp_req=165 ttl=64 time=0,072 ms
64 bytes from 10.0.0.10: icmp_req=166 ttl=64 time=0,077 ms
64 bytes from 10.0.0.10: icmp_req=167 ttl=64 time=0,073 ms
64 bytes from 10.0.0.10: icmp_req=168 ttl=64 time=0,075 ms
64 bytes from 10.0.0.10: icmp_req=169 ttl=64 time=0,075 ms
64 bytes from 10.0.0.10: icmp_req=170 ttl=64 time=0,077 ms
64 bytes from 10.0.0.10: icmp_req=171 ttl=64 time=0,075 ms
64 bytes from 10.0.0.10: icmp_req=172 ttl=64 time=0,076 ms
64 bytes from 10.0.0.10: icmp_req=173 ttl=64 time=0,076 ms
64 bytes from 10.0.0.10: icmp_req=174 ttl=64 time=0,076 ms
64 bytes from 10.0.0.10: icmp_req=175 ttl=64 time=0,077 ms
64 bytes from 10.0.0.10: icmp_req=176 ttl=64 time=0,076 ms
64 bytes from 10.0.0.10: icmp_req=177 ttl=64 time=0,075 ms

vcmid
gth 64
10:14:21,298414 IP 10.0.0.10 > 10.0.0.20: ICMP echo reply, id 72, seq 172, len
h 64
10:14:22,297581 IP 10.0.0.20 > 10.0.0.10: ICMP echo request, id 72, seq 173, len
gth 64
10:14:22,297602 IP 10.0.0.10 > 10.0.0.20: ICMP echo reply, id 72, seq 173, len
h 64
10:14:23,297437 IP 10.0.0.20 > 10.0.0.10: ICMP echo request, id 72, seq 174, len
gth 64
10:14:23,297459 IP 10.0.0.10 > 10.0.0.20: ICMP echo reply, id 72, seq 174, len
h 64
10:14:24,297562 IP 10.0.0.20 > 10.0.0.10: ICMP echo request, id 72, seq 175, len
gth 64
10:14:24,297583 IP 10.0.0.10 > 10.0.0.20: ICMP echo reply, id 72, seq 175, len
h 64
10:14:25,297437 IP 10.0.0.20 > 10.0.0.10: ICMP echo request, id 72, seq 176, len
gth 64
10:14:25,297458 IP 10.0.0.10 > 10.0.0.20: ICMP echo reply, id 72, seq 176, len
h 64
10:14:26,297458 IP 10.0.0.20 > 10.0.0.10: ICMP echo request, id 72, seq 177, len
gth 64
10:14:26,297479 IP 10.0.0.10 > 10.0.0.20: ICMP echo reply, id 72, seq 177, len
h 64

vcmid
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), capture size 65535 bytes

vcmid
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), capture size 65535 bytes

```

Figura 14: Ping de n1 para n2 (canto superior esquerdo), *tcpdump* de n2 (canto superior direito), *tcpdump* de n3 (canto inferior esquerdo), *tcpdump* de n4 (canto inferior direito)

Utilizando um *switch* ao invés de *hub*, a rede o tráfego de um dispositivo para outro passa a ir diretamente da origem para o destino, sendo que os outros dispositivos na rede não conseguem ver tráfego entre estes dois (o *switch* encaminha o pacote diretamente para a porta onde se encontra o destino).

5 Conclusão

Este relatório tem como objetivo apresentar as respostas à ficha apresentada nas aulas práticas. A resolução desta ficha permitiu-nos adquirir conhecimento acerca da arquitetura Ethernet fazendo com que, na prática, vissemos os endereços MAC de origem e destino das tramas que capturávamos utilizando o software *Wireshark* que nos ofereceu uma visão mais realista. Um dos grandes focos desta ficha foi também o protocolo ARP encontrado ao nível da ligação de dados, em que numa primeira fase foi analisado o conteúdo da tabela ARP, bem como os endereços lá especificados e o seu processo de encaminhamento de tramas. Já numa fase mais avançada foi forçado um pedido ARP gratuito de forma a conseguir visualizar as diferenças entre ele e os restantes pedidos. No final, através de uma topologia modelada no *CORE*, foi possível analisar as diferenças entre *hubs* *switches* dentro de uma rede, como funcionam e o porquê de os *switches* serem uma melhor escolha para usar.