

# TP3: Camada de Ligação Lógica: Ethernet e Protocolo ARP

Diogo Braga, João Silva, and Ricardo Caçador

University of Minho, Department of Informatics, 4710-057 Braga, Portugal  
e-mail: {a82547,a82005,a81064}@alunos.uminho.pt  
PL4, Grupo 7

## 1 Captura e análise de Tramas Ethernet

### 1.1 Exercício 1

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

```
▼ Ethernet II, Src: BelkinIn_7f:f4:5c (58:ef:68:7f:f4:5c), Dst: Vmware_d2:19:f0 (00:0c:29:d2:19:f0)
  ► Destination: Vmware_d2:19:f0 (00:0c:29:d2:19:f0)
  ► Source: BelkinIn_7f:f4:5c (58:ef:68:7f:f4:5c)
    Type: IPv4 (0x0800)
```

**Fig. 1.** Ethernet II

**R:** Como se pode observar na figura 1, o endereço MAC de origem da trama capturada é **58:ef:68:7f:f4:5c**, enquanto o endereço de destino é **00:0c:29:d2:19:f0**.

### 1.2 Exercício 2

*Identifique a que sistemas se referem. Justifique.*

**R:** Como se pode observar na figura 1, o endereço de origem refere-se à interface de comunicação da nossa máquina. Neste caso, estamos conectados com um adaptador *belkin*, daí a parte do endereço de origem atribuída ao fabricante estar assim designada. O endereço de destino refere-se ao router de acesso da nossa rede local. Neste caso, o fabricante é a *Vmware*.

### 1.3 Exercício 3

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

**R:** Como se pode observar na figura 1, o valor é **0x0800**. Este campo é usado para indicar o protocolo que é encapsulado no payload do frame, sendo que neste caso é o **IPv4**.

### 1.4 Exercício 4

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

```
▼ Hypertext Transfer Protocol
▼ GET / HTTP/1.1\r\n
  ▶ [Expert Info (Chat/Sequence): GET / HTTP/1.1\r\n]
    Request Method: GET
    Request URI: /
    Request Version: HTTP/1.1
    Host: miei.di.uminho.pt\r\n
    User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.14; rv:63.0) Gecko/20100101 Firefox/63.0\r\n
    Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8\r\n
    Accept-Language: pt-PT,pt;q=0.8,en;q=0.5,en-US;q=0.3\r\n
    Accept-Encoding: gzip, deflate\r\n
    Connection: keep-alive\r\n
    Upgrade-Insecure-Requests: 1\r\n
    \r\n
    0000 00 0c 29 d2 19 f0 58 ef 68 7f f4 5c 08 00 45 00 ..)...X.h...\E.
    0010 01 94 00 00 40 00 40 06 3f 34 c0 a8 64 d7 c1 88 ...@.@. ?4..d...
    0020 13 28 eb 99 00 50 aa 30 69 7e 91 04 6c 3d 80 18 ...{...P.0 i~...l=...
    0030 08 0a 77 bc 00 00 01 01 08 0a 0c b8 3b 5e 84 d8 ...w.....;^...
    0040 0b f4 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 6e 74 3a 20 4d 6f 7a 69 6c 6c 61 2f 35 Agent: M ozilla/5
    0080 2e 30 20 28 4d 61 63 69 6e 74 6f 73 68 3b 20 49 .0 (Maci ntosh; I
    0090 6e 74 65 6c 20 4d 61 63 20 4f 53 20 58 20 31 30 ntel Mac OS X 10
    00a0 2e 31 34 3b 20 72 76 3a 36 33 2e 30 29 20 47 65 .14; rv: 63.0) Ge
    00b0 63 6b 6f 2f 32 30 31 30 30 31 30 31 20 46 69 72 cko/2010 0101 Fir
    00c0 65 66 6f 78 2f 36 33 2e 30 0d 0a 41 63 63 65 70 efox/63. 0..Accep
    00d0 74 3a 20 74 65 78 74 2f 68 74 6d 6c 2c 61 70 70 t: text/ html,app
    00e0 6c 69 63 61 74 69 6f 6e 2f 78 68 74 6d 6c 2b 78 lication /xhtml+x
    00f0 6d 6c 2c 61 70 70 6c 69 63 61 74 69 6f 6e 2f 78 ml,appli cation/x
    0100 6d 6c 3b 71 3d 30 2e 39 2c 2a 2f 2a 3b 71 3d 30 ml;q=0.9 ,*/*;q=0
    0110 2e 38 0d 0a 41 63 63 65 70 74 2d 4c 61 6e 67 75 .8..Acce pt-Langu
    0120 61 67 65 3a 20 70 74 2d 50 54 2c 70 74 3b 71 3d age: pt- PT,pt;q=
    0130 30 2e 38 2c 65 6e 3b 71 3d 30 2e 35 2c 65 6e 2d 0.8,en;q =0.5,en-
    0140 55 53 3b 71 3d 30 2e 33 0d 0a 41 63 63 65 70 74 US;q=0.3 ..Accept
    0150 2d 45 6e 63 6f 64 69 6e 67 3a 20 67 7a 69 70 2c -Encodin g: gzip,
    0160 20 64 65 66 6c 61 74 65 0d 0a 43 6f 6e 6e 65 63 deflate ..Connec
    0170 74 69 6f 6e 3a 20 6b 65 65 70 2d 61 6c 69 76 65 tion: ke ep-alive
    0180 0d 0a 55 70 67 72 61 64 65 2d 49 6e 73 65 63 75 ..Upgrad e-Insecu
    0190 72 65 2d 52 65 71 75 65 73 74 73 3a 20 31 0d 0a re-Reque sts: 1..
    01a0 0d 0a ..
```

Fig. 2. HTTP GET

**R:** Até ao carácter ASCII "G" são usados **66 bytes**, enquanto no total são usados **418 bytes**. Deste modo, o overhead introduzido pela pilha protocolar é calculado dividindo 66 por 418, e multiplicando por 100, resulta em **15.79%**. Tal pode ser verificado na figura 2.

### 1.5 Exercício 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?*

**R:** Como estamos perante uma ligação Ethernet, e como este tipo de ligação especifica que uma trama danificada deve ser descartada, não é necessário que exista o campo FCS, uma vez que este género de ligação não especifica nenhuma ação que faça com que a trama seja retransmitida. Neste caso a ocorrência de erros é ínfima, mas por exemplo, no caso da ligação Wireless o campo FCS certamente estaria presente na trama, visto este tipo de ligação ser mais sucestível a ruído e erros.

### 1.6 Exercício 6

*Qual é o endereço Ethernet da fonte? A que sistema de rede corresponde? Justifique.*

**R:** O endereço da fonte é **00:0c:29:d2:19:f0**, que corresponde ao router da rede local. Uma vez que recebemos a resposta do endereço da interface IP 193.136.19.40, a nível de

ligação de dados o router tem uma tabela que permite fazer o mapeamento entre endereços de nível de rede e endereços de nível de ligação lógica. Como o nível de ligação lógica apenas conhece os hosts da rede local, a trama é entregue no destino 58:ef:68:7f:f4:5c, que corresponde ao endereço de IP 192.168.100.215.

### 1.7 Exercício 7

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

**R:** O endereço de destino é **58:ef:68:7f:f4:5c**, e corresponde à interface de comunicação do nosso computador.

### 1.8 Exercício 8

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

**R:** Os protocolos contidos na trama recebida são o HTTP a nível aplicacional, o TCP a nível de transporte e o IPv4 a nível de rede.

## 2 Protocolo ARP

### 2.1 Exercício 9

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

```
$ arp -a
? (192.168.100.204) at 0:e0:4c:68:6d on en5 ifscope [ethernet]
server6.sa.di.uminho.pt (192.168.100.242) at 0:c:29:98:ac:62 on en5 ifscope [ethernet]
gw.sa.di.uminho.pt (192.168.100.254) at 0:c:29:d2:19:f0 on en5 ifscope [ethernet]
? (192.168.100.255) at ff:ff:ff:ff:ff:ff on en5 ifscope [ethernet]
? (224.0.0.251) at 1:0:5e:0:0:fb on en5 ifscope permanent [ethernet]
```

**Fig. 3.** Tabela ARP

**R:** As tabelas ARP fazem o mapeamento entre endereços de rede e endereços de nível de ligação de dados. Como se pode observar na figura 3, a primeira coluna da tabela corresponde aos endereços de nível 3, enquanto a segunda coluna corresponde aos endereços de nível 2.

### 2.2 Exercício 10

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

```
► Frame 960: 60 bytes on wire (480 bits), 60 bytes captured (480 bits) on interface 0
► Ethernet II, Src: Vmware_d2:19:f0 (00:0c:29:d2:19:f0), Dst: BelkinIn_7f:f4:5c (58:ef:68:7f:f4:5c)
► Address Resolution Protocol (request)
```

**Fig. 4.** Origem e Destino da Trama Ethernet

**R:** O endereço de origem da trama Ethernet encontra-se sublinhado a vermelho na figura 4, cujo valor é **00:0c:29:d2:19:f0**. Por outro lado, o endereço de destino é **58:ef:68:7f:f4:5c**, e encontra-se sublinhado a azul na mesma figura. O endereço de destino identifica o nosso computador, tal é conclusível porque estamos perante uma situação em que o router tenta ter conexão a nível de ligação de dados com este mesmo, visto já saber o seu endereço IP.

## 2.3 Exercício 11

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

```
▼ Ethernet II, Src: Vmware_d2:19:f0 (00:0c:29:d2:19:f0), Dst: BelkinIn_7f:f4:5c (58:ef:68:7f:f4:5c)
  ► Destination: BelkinIn_7f:f4:5c (58:ef:68:7f:f4:5c)
  ► Source: Vmware_d2:19:f0 (00:0c:29:d2:19:f0)
    Type: ARP (0x0806)
    Padding: 000000000000000000000000000000000000000000000000
  ► Address Resolution Protocol (request)
```

**Fig. 5.** Tipo da Trama Ethernet

**R:** Como se pode visualizar na figura 5 sublinhado a azul, o valor do campo tipo é **0x0806**, e indica o protocolo que vai encapsulado dentro da trama Ethernet, neste caso ARP.

## 2.4 Exercício 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>.*

```
▼ Address Resolution Protocol (request)
  Hardware type: Ethernet (1)
  Protocol type: IPv4 (0x0800)
  Hardware size: 6
  Protocol size: 4
  Opcode: request (1)
  Sender MAC address: Vmware_d2:19:f0 (00:0c:29:d2:19:f0)
  Sender IP address: 192.168.100.254
  Target MAC address: 00:00:00_00:00:00 (00:00:00:00:00:00)
  Target IP address: 192.168.100.203
```

**Fig. 6.** ARP Request

**R:** O valor do campo ARP opcode é **request (1)**, tal como pode ser verificável na figura 6 sublinhado a azul. O opcode serve para determinar se a mensagem ARP é um pedido ou uma resposta ao pedido anterior.

## 2.5 Exercício 13

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

**R:** Existem dois endereços do tipo MAC Address e dois do tipo IP. A nível de rede, sabemos que o endereço de origem, sublinhado a vermelho, pretende comunicar com o endereço destino, sublinhado a verde. A nível de ligação de dados, conhecemos o endereço de origem sublinhado a preto, mas não temos conhecimento de qual seja o endereço de destino, pelo que enviamos para o endereço broadcast, sublinhado a castanho. Tal pode ser também verificável na figura 6.

## 2.6 Exercício 14

*Explicite que tipo de pedido ou pergunta é feita pelo host de origem?*

960	26.122486	Vmware_d2:19:f0	BelkinIn_7f:f4:5c	ARP	60	<u>Who has 192.168.100.203? Tell 192.168.100.254</u>
961	26.122515	BelkinIn_7f:f4:5c	Vmware_d2:19:f0	ARP	42	192.168.100.203 is at 58:ef:68:7f:f4:5c

**Fig. 7.** Request ARP

**R:** Como se pode visualizar na figura 7 sublinhado a preto, o host de origem pergunta quem tem o endereço 192.168.100.203, e diz para o tal o comunicar ao endereço 192.168.100.254.

## 2.7 Exercício 15

*Localize a mensagem ARP que é a resposta ao pedido ARP efectuado.*

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

```
▼ Address Resolution Protocol (reply)
  Hardware type: Ethernet (1)
  Protocol type: IPv4 (0x0800)
  Hardware size: 6
  Protocol size: 4
  Opcode: reply (2)
  Sender MAC address: BelkinIn_7f:f4:5c (58:ef:68:7f:f4:5c)
  Sender IP address: 192.168.100.203
  Target MAC address: Vmware_d2:19:f0 (00:0c:29:d2:19:f0)
  Target IP address: 192.168.100.254
```

**Fig. 8.** ARP Reply

**R:** Como se pode visualizar na figura 8 sublinhado a preto, o valor do campo ARP opcode é **reply (2)**. Uma vez que o endereço do destino do pedido é igual ao endereço do nosso computador, este envia um ARP Reply.

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

**R:** A resposta ao pedido ARP encontra-se no campo **Sender MAC address**, uma vez que este tem o endereço que o router procura, e portanto responde ao request.

### 3 ARP Gratuito

#### 3.1 Exercício 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?*

```
▼ Address Resolution Protocol (request/gratuitous ARP)
  Hardware type: Ethernet (1)
  Protocol type: IPv4 (0x0800)
  Hardware size: 6
  Protocol size: 4
  Opcode: request (1)
  [Is gratuitous: True]
  Sender MAC address: BelkinIn_7f:f4:5c (58:ef:68:7f:f4:5c)
  Sender IP address: 192.168.100.203
  Target MAC address: 00:00:00_00:00:00 (00:00:00:00:00:00)
  Target IP address: 192.168.100.203
```

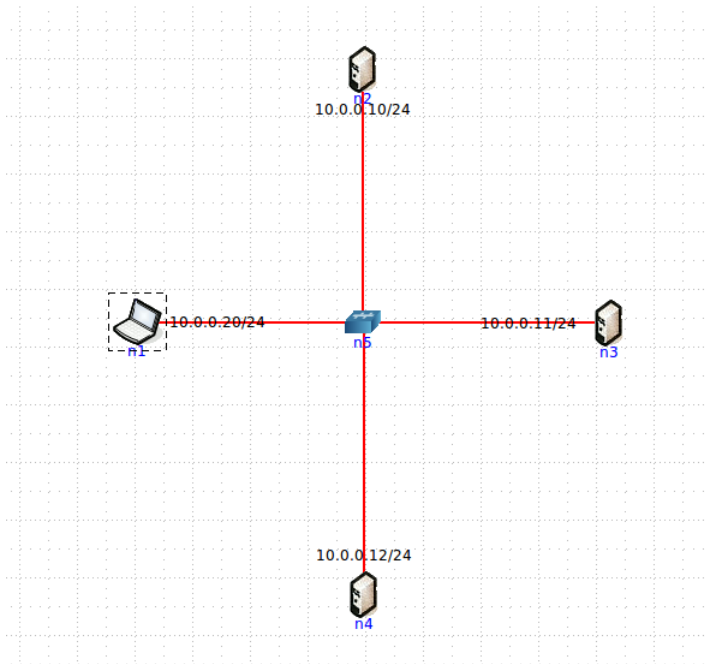
**Fig. 9.** ARP Request Gratuito

```
▼ Address Resolution Protocol (request)
  Hardware type: Ethernet (1)
  Protocol type: IPv4 (0x0800)
  Hardware size: 6
  Protocol size: 4
  Opcode: request (1)
  Sender MAC address: BelkinIn_7f:f4:5c (58:ef:68:7f:f4:5c)
  Sender IP address: 192.168.100.203
  Target MAC address: 00:00:00_00:00:00 (00:00:00:00:00:00)
  Target IP address: 192.168.100.254
```

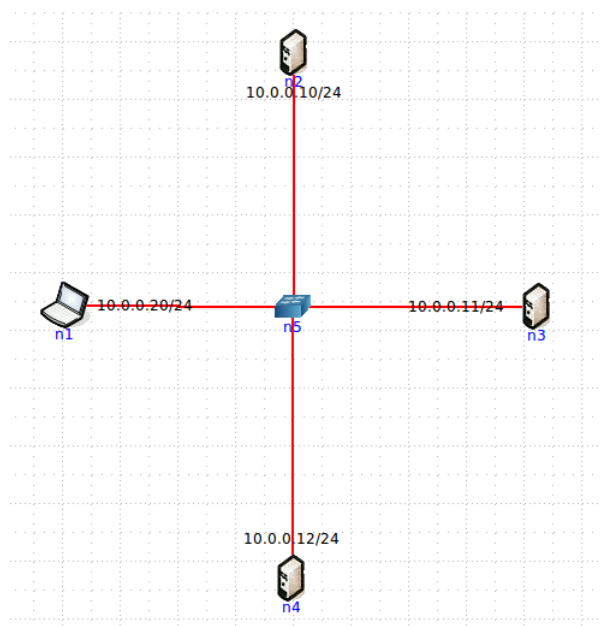
**Fig. 10.** ARP Request Normal

**R:** Como pode ser verificado nas figuras 9 e 10, os endereços IP origem e destino do ARP Request gratuito são iguais, algo que não se verifica nos ARP Request normais. Isto porque nos ARP Request normais pretende-se saber qual o MAC Address de um determinado endereço IP. Por outro lado, no ARP Request gratuito pretende-se anunciar um endereço MAC para que todos os sistemas na rede local possam atualizar as suas tabelas ARP.

#### 4 Domínios de colisão



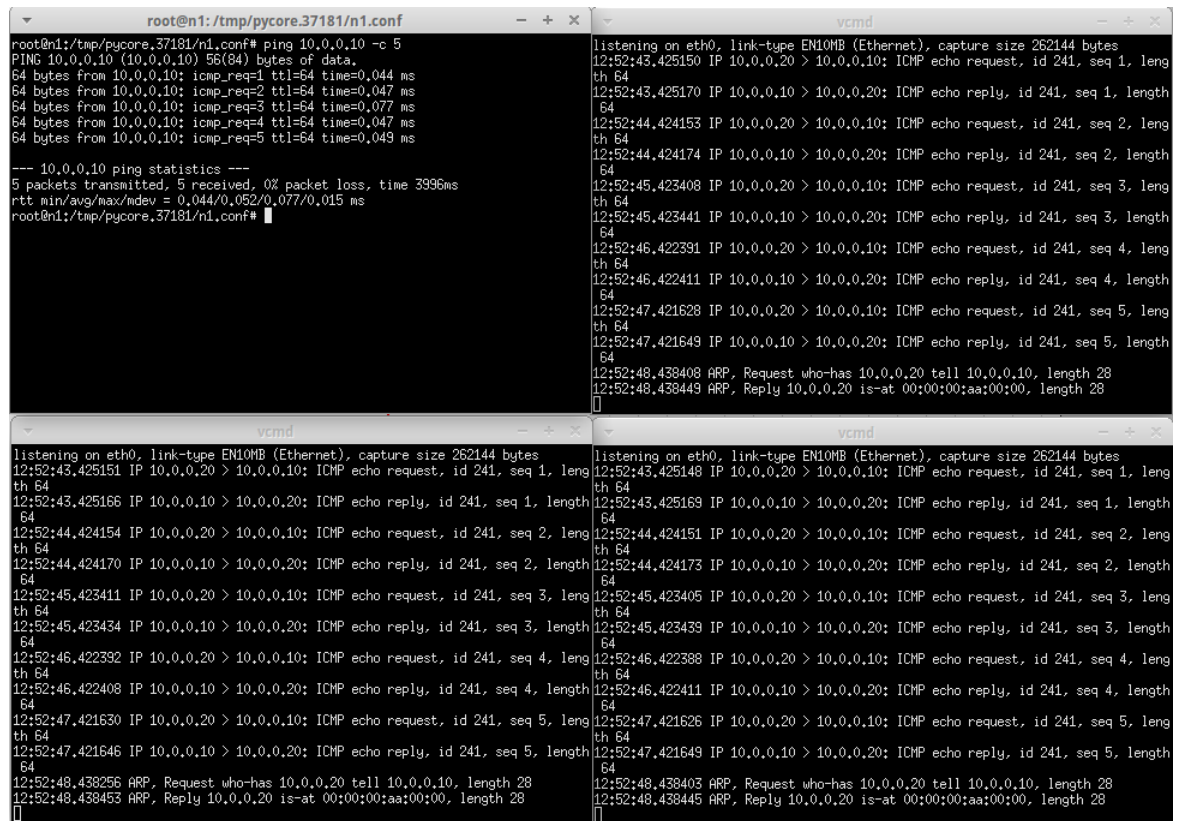
**Fig. 11.** Topologia com Hub



**Fig. 12.** Topologia com Switch

## 4.1 Exercício 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.37181/n1.conf# ping 10.0.0.10 -c 5
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=0.044 ms
64 bytes from 10.0.0.10: icmp_req=2 ttl=64 time=0.047 ms
64 bytes from 10.0.0.10: icmp_req=3 ttl=64 time=0.077 ms
64 bytes from 10.0.0.10: icmp_req=4 ttl=64 time=0.047 ms
64 bytes from 10.0.0.10: icmp_req=5 ttl=64 time=0.049 ms

--- 10.0.0.10 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 3996ms
rtt min/avg/max/mdev = 0.044/0.052/0.077/0.015 ms
root@n1:/tmp/pycore.37181/n1.conf#
```

```
listening on eth0, link-type EN10MB (Ethernet), capture size 262144 bytes
12:52:43.425150 IP 10.0.0.20 > 10.0.0.10: ICMP echo request, id 241, seq 1, length 64
12:52:43.425170 IP 10.0.0.10 > 10.0.0.20: ICMP echo reply, id 241, seq 1, length 64
12:52:44.424153 IP 10.0.0.20 > 10.0.0.10: ICMP echo request, id 241, seq 2, length 64
12:52:44.424174 IP 10.0.0.10 > 10.0.0.20: ICMP echo reply, id 241, seq 2, length 64
12:52:45.423408 IP 10.0.0.20 > 10.0.0.10: ICMP echo request, id 241, seq 3, length 64
12:52:45.423441 IP 10.0.0.10 > 10.0.0.20: ICMP echo reply, id 241, seq 3, length 64
12:52:46.422391 IP 10.0.0.20 > 10.0.0.10: ICMP echo request, id 241, seq 4, length 64
12:52:46.422411 IP 10.0.0.10 > 10.0.0.20: ICMP echo reply, id 241, seq 4, length 64
12:52:47.421628 IP 10.0.0.20 > 10.0.0.10: ICMP echo request, id 241, seq 5, length 64
12:52:47.421649 IP 10.0.0.10 > 10.0.0.20: ICMP echo reply, id 241, seq 5, length 64
12:52:48.438408 ARP, Request who-has 10.0.0.20 tell 10.0.0.10, length 28
12:52:48.438449 ARP, Reply 10.0.0.20 is-at 00:00:00:aa:00:00, length 28
```

```
listening on eth0, link-type EN10MB (Ethernet), capture size 262144 bytes
12:52:43.425151 IP 10.0.0.20 > 10.0.0.10: ICMP echo request, id 241, seq 1, length 64
12:52:43.425166 IP 10.0.0.10 > 10.0.0.20: ICMP echo reply, id 241, seq 1, length 64
12:52:44.424154 IP 10.0.0.20 > 10.0.0.10: ICMP echo request, id 241, seq 2, length 64
12:52:44.424170 IP 10.0.0.10 > 10.0.0.20: ICMP echo reply, id 241, seq 2, length 64
12:52:45.423411 IP 10.0.0.20 > 10.0.0.10: ICMP echo request, id 241, seq 3, length 64
12:52:45.423434 IP 10.0.0.10 > 10.0.0.20: ICMP echo reply, id 241, seq 3, length 64
12:52:46.422392 IP 10.0.0.20 > 10.0.0.10: ICMP echo request, id 241, seq 4, length 64
12:52:46.422408 IP 10.0.0.10 > 10.0.0.20: ICMP echo reply, id 241, seq 4, length 64
12:52:47.421630 IP 10.0.0.20 > 10.0.0.10: ICMP echo request, id 241, seq 5, length 64
12:52:47.421646 IP 10.0.0.10 > 10.0.0.20: ICMP echo reply, id 241, seq 5, length 64
12:52:48.438256 ARP, Request who-has 10.0.0.20 tell 10.0.0.10, length 28
12:52:48.438453 ARP, Reply 10.0.0.20 is-at 00:00:00:aa:00:00, length 28
```

```
listening on eth0, link-type EN10MB (Ethernet), capture size 262144 bytes
12:52:43.425148 IP 10.0.0.20 > 10.0.0.10: ICMP echo request, id 241, seq 1, length 64
12:52:43.425169 IP 10.0.0.10 > 10.0.0.20: ICMP echo reply, id 241, seq 1, length 64
12:52:44.424151 IP 10.0.0.20 > 10.0.0.10: ICMP echo request, id 241, seq 2, length 64
12:52:44.424173 IP 10.0.0.10 > 10.0.0.20: ICMP echo reply, id 241, seq 2, length 64
12:52:45.423405 IP 10.0.0.20 > 10.0.0.10: ICMP echo request, id 241, seq 3, length 64
12:52:45.423439 IP 10.0.0.10 > 10.0.0.20: ICMP echo reply, id 241, seq 3, length 64
12:52:46.422388 IP 10.0.0.20 > 10.0.0.10: ICMP echo request, id 241, seq 4, length 64
12:52:46.422411 IP 10.0.0.10 > 10.0.0.20: ICMP echo reply, id 241, seq 4, length 64
12:52:47.421626 IP 10.0.0.20 > 10.0.0.10: ICMP echo request, id 241, seq 5, length 64
12:52:47.421649 IP 10.0.0.10 > 10.0.0.20: ICMP echo reply, id 241, seq 5, length 64
12:52:48.438403 ARP, Request who-has 10.0.0.20 tell 10.0.0.10, length 28
12:52:48.438445 ARP, Reply 10.0.0.20 is-at 00:00:00:aa:00:00, length 28
```

Fig. 13. Fluxo com Hub

**R:** Como pode ser verificado na figura 13, usando um Hub e executando o campo ping a partir do laptop n1, o Hub replica o fluxo para todos os servidores, tal é possível ver na figura uma vez que utilizamos o tcpdump em cada servidor para verificar o fluxo de tráfego. Um Hub é por consequência mais suscetível a ocorrência de colisões.

## 4.2 Exercício 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.37182/n1.conf# ping 10.0.0.10 -c 5
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=0.041 ms
64 bytes from 10.0.0.10: icmp_req=2 ttl=64 time=0.032 ms
64 bytes from 10.0.0.10: icmp_req=3 ttl=64 time=0.055 ms
64 bytes from 10.0.0.10: icmp_req=4 ttl=64 time=0.053 ms
64 bytes from 10.0.0.10: icmp_req=5 ttl=64 time=0.042 ms

--- 10.0.0.10 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 399ms
rtt min/avg/max/mdev = 0.032/0.044/0.055/0.011 ms
root@n1: /tmp/pycore.37182/n1.conf#
```

```
vcmd
listening on eth0, link-type EN10MB (Ethernet), capture size 262144 bytes
12:57:45.510617 IP 10.0.0.20 > 10.0.0.10: ICMP echo request, id 74, seq 1, length 64
12:57:45.510629 IP 10.0.0.10 > 10.0.0.20: ICMP echo reply, id 74, seq 1, length 64
12:57:46.509613 IP 10.0.0.20 > 10.0.0.10: ICMP echo request, id 74, seq 2, length 64
12:57:46.509623 IP 10.0.0.10 > 10.0.0.20: ICMP echo reply, id 74, seq 2, length 64
12:57:47.510879 IP 10.0.0.20 > 10.0.0.10: ICMP echo request, id 74, seq 3, length 64
12:57:47.510893 IP 10.0.0.10 > 10.0.0.20: ICMP echo reply, id 74, seq 3, length 64
12:57:48.509876 IP 10.0.0.20 > 10.0.0.10: ICMP echo request, id 74, seq 4, length 64
12:57:48.509891 IP 10.0.0.10 > 10.0.0.20: ICMP echo reply, id 74, seq 4, length 64
12:57:49.509616 IP 10.0.0.20 > 10.0.0.10: ICMP echo request, id 74, seq 5, length 64
12:57:49.509627 IP 10.0.0.10 > 10.0.0.20: ICMP echo reply, id 74, seq 5, length 64
12:57:50.518664 ARP, Request who-has 10.0.0.20 tell 10.0.0.10, length 28
12:57:50.518695 ARP, Reply 10.0.0.20 is-at 00:00:00:aa:00:00, length 28
```

```
vcmd
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), capture size 262144 bytes
```

```
vcmd
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), capture size 262144 bytes
```

Fig. 14. Fluxo com Switch

**R:** Como pode ser verificado na figura 14, usando um Switch e executando o campo ping a partir do laptop n1, o Switch tem capacidade para enviar o fluxo só para o servidor pretendido, diminuindo assim a ocorrência de colisões, ao contrário dum Hub. Na figura, com auxílio do tcpdump aberto em cada servidor, é possível verificar tal ocorrência.

## 5 Conclusão

Neste trabalho prático dividido em 4 partes, abordamos principalmente temas como a Ethernet e o Protocolo ARP.

Na primeira secção trabalhamos com tramas Ethernet, e analisamos a fundo a questão dos endereços de nível lógico, também conhecidos com **MAC Address**

Na segunda secção abordamos o protocolo ARP, que inclui a análise de tabelas ARP presentes tanto nos end-systems como nos routers da rede local. Aprendemos também de uma forma muito prático como é feita a comunicação entre o router e um end-system a nível de ligação de dados.

Na terceira secção abordamos a questão do ARP gratuito, e foi de facto interessante perceber que quando um novo host entra numa rede local envia uma mensagem ARP a todos os hosts, para que estes possam atualizar as suas tabelas.

Na quarta e última secção tratamos da questão dos domínios de colisões, onde pusemos lado a lado um Hub e um Switch de forma a comparar o seu funcionamento e questões relacionadas com colisões na transmissão de tramas Ethernet.

Concluindo, com este guião exploramos a fundo as questões do nível de ligação de dados, e conseguimos de forma muito prática aprender os conceitos mais teóricos do funcionamento dos mecanismos desta camada.