

UNIVERSIDADE DO MINHO

MESTRADO INTEGRADO DE ENGENHARIA INFORMÁTICA

## **Redes de Computadores**

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

#### **Autores:**

Frederico Pinto

A73639



Pedro Silva

A78434



Ricardo Leal

A75411



1 de Abril de 2018

# 1 Introdução

Este trabalho prático foi proposto na disciplina de Redes de Computadores e divide-se em duas partes. Na primeira parte foi abordada a tecnologia Ethernet. Esta tecnologia é usada como norma de comunicação e ligação de dados

## 2 Parte I

### 2.1 Captura e análise de Tramas Ethernet

#### 1) Anote os endereços MAC de origem e de destino da trama capturada.

Origem: 60:02:92:10:57:46

Destino: c8:2a:14:58:01:60

Como podemos ver na Figura 1.

```

Ethernet II, Src: Pegatron_10:57:46 (60:02:92:10:57:46), Dst: Apple_58:01:60 (c8:2a:14:58:01:60)
  Destination: Apple_58:01:60 (c8:2a:14:58:01:60)
    Address: Apple_58:01:60 (c8:2a:14:58:01:60)
      ....0. .... = LG bit: Globally unique address (factory default)
      ....0. .... = IG bit: Individual address (unicast)
  Source: Pegatron_10:57:46 (60:02:92:10:57:46)
    Address: Pegatron_10:57:46 (60:02:92:10:57:46)
      ....0. .... = LG bit: Globally unique address (factory default)
      ....0. .... = IG bit: Individual address (unicast)
Type: IPv4 (0x0800)
```

**Figura 1.** Campo Ethernet II da trama que suporta a mensagem HTTP GET.

#### 2) Identifique a que sistemas se referem. Justifique.

O endereço de origem identifica o *laptop*, i.e., a nossa máquina nativa, enquanto que o endereço destino representa o *router* da rede Ethernet da sala de aula.

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

Pela Figura 1 constatamos que vale 0x0800 e identifica o protocolo utilizado a nível de rede, neste caso, IPv4.

#### 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.

Até ao caractere "G" são usados 54 bytes. No total, o tamanho da trama é de 646 bytes. Ora, concluímos que:

$$Overhead(\%) = \frac{54}{646} * 100 \approx 8.36\%$$

Hypertext Transfer Protocol	
GET / HTTP/1.1\r\n	
[Expert Info (Chat/Sequence): GET / HTTP/1.1\r\n]	
[GET / HTTP/1.1\r\n]	
[Severity level: Chat]	
[Group: Sequence]	
Request Method: GET	
Request URI: /	
Request Version: HTTP/1.1	
Host: mlei.di.uminho.pt\r\n	
0000	c8 2a 14 58 01 60 00 02 92 10 57 46 08 00 45 00 .".X."..WF..E.
0010	02 78 06 37 40 00 80 06 00 00 c0 a8 02 07 c1 88 .x.7@... .....
0020	13 14 ea f2 00 50 f7 76 db e8 0f 4f 85 f8 50 18 ....P.v ...O..P.
0030	01 00 99 b6 00 00 47 45 54 20 2f 20 48 54 54 50 .....GE T / HTTP
0040	2f 31 2e 31 0d 0a 48 6f 73 74 3a 20 6d 69 65 69 /1.1..Ho st: mlei
0050	2e 64 69 2e 75 6d 69 6e 68 6f 2e 70 74 0d 0a 43 .di.umin ho.pt..C
0060	6f 6e 6e 65 63 74 69 6f 6e 3a 20 6b 65 65 70 2d onnectio n: keep-
0070	61 6c 69 76 65 0d 0a 43 61 63 68 65 2d 43 6f 6e alive..C ache-Con
0080	74 72 6f 6c 3a 20 6d 61 78 2d 61 67 65 3d 30 0d trol: ma x-age=0.
0090	0a 55 73 65 72 2d 41 67 65 6e 74 3a 20 4d 6f 7a .User-Ag ent: Moz
00a0	69 6c 6c 61 2f 35 2e 30 20 28 57 69 6e 64 6f 77 illa/5.0 (Window
00b0	73 20 4e 54 20 36 2e 31 3b 20 57 69 6e 36 3a 3b s NT 6.1 ; Win64;
00c0	20 78 36 34 29 20 41 70 70 6c 65 57 65 62 4b 69 x64) Ap pleWebK1
00d0	74 2f 35 33 37 2e 33 36 20 28 4b 48 54 4d 4c 2c t/537.36 (KHTML,
00e0	20 6c 69 6b 65 20 47 65 63 6b 6f 29 20 43 68 72 like Ge cko) Chr
00f0	6f 6d 65 2f 36 32 2e 30 2e 33 32 30 32 2e 39 34 ome/62.0 .3202.94
0100	20 53 61 66 61 72 69 2f 35 33 37 2e 33 36 0d 0a Safari/ 537.36..
0110	55 70 67 72 61 64 65 2d 49 6e 73 65 63 75 72 65 Upgrade- Insecure
0120	2d 52 65 71 75 65 73 74 73 3a 20 31 0d 0a 41 63 -Request s: 1..Ac
0130	63 65 70 74 3a 20 74 65 78 74 2f 68 74 6d 6c 2c cept: te xt/html,
0140	61 70 70 6c 69 63 61 74 69 6f 6e 2f 78 68 74 6d applicat ion/xhtm
0150	6c 2b 78 6d 6c 2c 61 70 70 6c 69 63 61 74 69 6f l+xml,ap plicatio
0160	6e 2f 78 6d 6c 3b 71 3d 30 2e 39 2c 69 6d 61 67 n/xml;q= 0.9,imag
0170	65 2f 77 65 62 70 2c 69 6d 61 67 65 2f 61 70 6e e/webp,i mage/apn
0180	67 2c 2a 2f 2a 3b 71 3d 30 2e 38 0d 0a 41 63 63 g,"/*;q= 0.8..Acc
0190	65 70 74 2d 45 6e 63 6f 64 69 6e 67 3a 20 67 7a ept-Enco ding: gz

Figura 2. Parte da trama.

5) Em ligações com fios pouco susceptíveis a erros, nem sempre as NICs geram o código de detecção de erros. Através de visualização direta de uma trama capturada, verifique se o campo FCS está visível, i.e., se está a ser utilizado. Aceda à opção "Edit/Preferences/Protocols/Ethernet" e indique que é assumido o uso do campo FCS. Verifique qual o valor hexadecimal desse campo na trama capturada. Que conclui? Reponha a configuração original.

O campo FCS vale 0x0d0a0d0a, o que nos permite concluir que não foi gerado código de detecção de erros, como observamos na Figura 3.

Frame check sequence: 0x0d0a0d0a incorrect, should be 0x2c20fae9	
[Expert Info (Error/Checksum): Bad checksum [should be 0x2c20fae9]]	
[Bad checksum [should be 0x2c20fae9]]	
[Severity level: Error]	
[Group: Checksum]	
[FCS Status: Bad]	
Internet Protocol Version 4, Src: 192.168.2.7, Dst: 193.136.19.20	
Transmission Control Protocol, Src Port: 60146, Dst Port: 80, Seq: 1, Ack: 1, Len: 588	
0100	20 53 61 66 61 72 69 2f 35 33 37 2e 33 36 0d 0a Safari/ 537.36..
0110	55 70 67 72 61 64 65 2d 49 6e 73 65 63 75 72 65 Upgrade- Insecure
0120	2d 52 65 71 75 65 73 74 73 3a 20 31 0d 0a 41 63 -Request s: 1..Ac
0130	63 65 70 74 3a 20 74 65 78 74 2f 68 74 6d 6c 2c cept: te xt/html,
0140	61 70 70 6c 69 63 61 74 69 6f 6e 2f 78 68 74 6d applicat ion/xhtm
0150	6c 2b 78 6d 6c 2c 61 70 70 6c 69 63 61 74 69 6f l+xml,ap plicatio
0160	6e 2f 78 6d 6c 3b 71 3d 30 2e 39 2c 69 6d 61 67 n/xml;q= 0.9,imag
0170	65 2f 77 65 62 70 2c 69 6d 61 67 65 2f 61 70 6e e/webp,i mage/apn
0180	67 2c 2a 2f 2a 3b 71 3d 30 2e 38 0d 0a 41 63 63 g,"/*;q= 0.8..Acc
0190	65 70 74 2d 45 6e 63 6f 64 69 6e 67 3a 20 67 7a ept-Enco ding: gz
01a0	69 70 2c 20 64 65 66 6c 61 74 65 0d 0a 41 63 63 ip, defl ate..Acc
01b0	65 70 74 2d 4c 61 6e 67 75 61 67 65 3a 20 70 74 ept-Lang uage: pt
01c0	2d 50 54 2c 70 74 3b 71 3d 30 2e 39 2c 65 6e 2d -PT,pt;q =0.9,en-
01d0	55 53 3b 71 3d 30 2e 38 2c 65 6e 3b 71 3d 30 2e US;q=0.8 ,en;q=0.
01e0	37 0d 0a 43 6f 6f 6b 69 65 3a 20 5f 67 61 3d 47 7..Cooki e: _ga=G
01f0	41 31 2e 32 2e 32 30 35 37 34 34 31 31 36 2e 31 A1.2.205 744116.1
0200	35 30 35 34 30 36 32 31 39 3b 20 5f 67 69 64 3d 50540621 9; _gid=
0210	47 41 31 2e 32 2e 32 32 35 32 33 37 37 31 2e 31 GA1.2.22 523771.1
0220	35 31 30 38 32 37 30 39 35 0d 0a 49 6e 2d 4e 6f 51082709 5..If-No
0230	6e 65 2d 4d 61 74 63 68 3a 20 22 31 34 34 36 62 ne-Match : "1446b
0240	66 2d 63 39 38 38 2d 64 63 36 39 66 39 63 30 22 f-c988-d c69f9c0"
0250	0d 0a 49 66 2d 4d 6f 64 69 66 69 65 64 2d 53 69 ..If-Mod ified=51
0260	6e 63 65 3a 20 54 68 75 2c 20 38 32 20 4e 6f 76 nce: Thu , 02 Nov
0270	20 32 30 31 37 20 30 39 3a 34 36 3a 32 33 20 47 2017 09 :46:23 G
0280	4d 54 0d 0a 0d 0a HT....

Figura 3. Frame em análise, vista conforme as configurações especificadas.

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

c8:2a:14:58:01:60, como verificamos na Figura 4, que identifica o *router* de acesso à rede da sala de aula.

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

Constatamos na Figura 4, que o endereço do destino é 60:02:92:10:57:46, associado à nossa máquina.

```

Ethernet II, Src: Apple_58:01:60 (c8:2a:14:58:01:60), Dst: Pegatron_10:57:46 (60:02:92:10:57:46)
  Destination: Pegatron_10:57:46 (60:02:92:10:57:46)
    Address: Pegatron_10:57:46 (60:02:92:10:57:46)
      ....0. .... = LG bit: Globally unique address (factory default)
      ....0. .... = IG bit: Individual address (unicast)
  Source: Apple_58:01:60 (c8:2a:14:58:01:60)
    Address: Apple_58:01:60 (c8:2a:14:58:01:60)
      ....0. .... = LG bit: Globally unique address (factory default)
      ....0. .... = IG bit: Individual address (unicast)
Type: IPv4 (0x0800)
```

Figura 4. Campo Ethernet II da trama que suporta a resposta à mensagem inicial.

8) Atendendo ao conceito de desencapsulamento protocolar, identifique os vários protocolos contidos na trama recebida

Desencapsulando progressivamente esta trama Ethernet, constatamos que os protocolos que a compõem são: Ethernet, IPv4, TCP e HTTP.

```

Encapsulation type: Ethernet (1)
Arrival Time: Nov 16, 2017 10:13:38.953454000 Hora padrão de GMT
[Time shift for this packet: 0.000000000 seconds]
Epoch Time: 1510827218.953454000 seconds
[Time delta from previous captured frame: 0.000804000 seconds]
[Time delta from previous displayed frame: 0.000804000 seconds]
[Time since reference or first frame: 3.730242000 seconds]
Frame Number: 39
Frame Length: 185 bytes (1480 bits)
Capture Length: 185 bytes (1480 bits)
[Frame is marked: False]
[Frame is ignored: False]
[Protocols in frame: eth:ethertype:ip:tcp:http]
[Coloring Rule Name: HTTP]
[Coloring Rule String: http || tcp.port == 80 || http2]
Ethernet II, Src: Apple_58:01:60 (c8:2a:14:58:01:60), Dst: Pegatron_10:57:46 (60:02:92:10:57:46)
Internet Protocol Version 4, Src: 193.136.19.20, Dst: 192.168.2.7
Transmission Control Protocol, Src Port: 80, Dst Port: 60146, Seq: 1, Ack: 593, Len: 131
Hypertext Transfer Protocol

0000  60 02 92 10 57 46 c8 2a 14 58 01 60 08 00 45 00  ...MF..X...E.
0010  00 ab b8 c6 40 00 3f 06 eb 3a c1 88 13 14 c0 a8  ...@.?.....
0020  02 07 00 50 ea f2 0f 4f 85 f8 f7 76 de 38 50 18  ...P...O...v.8P.
0030  19 70 1a 33 00 00 48 54 54 50 2f 31 2e 31 20 33  ..p3..HT TP/1.1 3
0040  30 34 20 4e 6f 74 20 4d 6f 64 69 66 69 65 64 0d  04 Not Modified.
0050  0a 44 61 74 65 3a 20 54 68 75 2c 20 31 36 20 4e  .Date: Thu, 16 N
0060  6f 76 20 32 30 31 37 20 31 30 3a 31 33 3a 34 34  ov 2017 10:13:44
0070  20 47 4d 54 0d 0a 53 65 72 76 65 72 3a 20 41 70  GMT..Se rver: Ap
0080  61 63 68 65 0d 0a 43 6f 6e 6e 65 63 74 69 6f 6e  ache..Co nnection
0090  3a 20 63 6c 6f 73 65 0d 0a 45 54 61 67 3a 20 22  ; close. .ETag: "
00a0  31 34 34 36 62 66 2d 63 39 38 38 2d 64 63 36 39  1446bf-c 988-dc69
00b0  66 39 63 30 22 0d 0a 0d 0a  f9c0"... .
```

Figura 5. Protocolos definidos na trama em estudo.

## 2.2 Protocolo ARP

### 9) Observe o conteúdo da tabela ARP. Diga o que significa cada uma das colunas?

Dado que o protocolo ARP estabelece relações entre endereços do nível de rede e endereços do nível de ligação lógica, a tabela ARP trata de armazenar o mapeamento definido em *soft state*, como podemos ver na Figura 6, onde na primeira coluna se encontra o endereço MAC, na segunda o endereço IP associado e na terceira o tipo de encaminhamento utilizado, estático ou dinâmico.

Interface: 172.26.11.33 --- 0xe		
Endereço Internet	Endereço físico	Tipo
172.26.254.254	00-d0-03-ff-94-00	dinâmico
172.26.255.255	ff-ff-ff-ff-ff-ff	estático
224.0.0.22	01-00-5e-00-00-16	estático
224.0.0.251	01-00-5e-00-00-fb	estático
224.0.0.252	01-00-5e-00-00-fc	estático
239.255.255.250	01-00-5e-7f-ff-fa	estático
255.255.255.255	ff-ff-ff-ff-ff-ff	estático
Interface: 192.168.56.1 --- 0xf		
Endereço Internet	Endereço físico	Tipo
192.168.56.255	ff-ff-ff-ff-ff-ff	estático
224.0.0.22	01-00-5e-00-00-16	estático
224.0.0.251	01-00-5e-00-00-fb	estático
224.0.0.252	01-00-5e-00-00-fc	estático
239.255.255.250	01-00-5e-7f-ff-fa	estático
Interface: 192.168.92.1 --- 0x11		
Endereço Internet	Endereço físico	Tipo
192.168.92.254	00-50-56-f8-38-c4	dinâmico
192.168.92.255	ff-ff-ff-ff-ff-ff	estático
224.0.0.22	01-00-5e-00-00-16	estático
224.0.0.251	01-00-5e-00-00-fb	estático
224.0.0.252	01-00-5e-00-00-fc	estático
239.255.255.250	01-00-5e-7f-ff-fa	estático
255.255.255.255	ff-ff-ff-ff-ff-ff	estático
Interface: 192.168.30.1 --- 0x12		
Endereço Internet	Endereço físico	Tipo
192.168.30.254	00-50-56-f6-1e-11	dinâmico
192.168.30.255	ff-ff-ff-ff-ff-ff	estático
224.0.0.22	01-00-5e-00-00-16	estático
224.0.0.251	01-00-5e-00-00-fb	estático
224.0.0.252	01-00-5e-00-00-fc	estático
239.255.255.250	01-00-5e-7f-ff-fa	estático
255.255.255.255	ff-ff-ff-ff-ff-ff	estático

Figura 6. Tabela ARP.

### 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?

Origem: 60:02:92:10:57:46

Destino: ff:ff:ff:ff:ff:ff

O pedido ARP pretende determinar o endereço físico associado ao endereço IP 192.168.2.1, por forma a preencher a tabela ARP, pelo que necessita de enviar o pedido em *broadcast*, sendo expectável que seja recebida resposta *unicast* do *host* com esse endereço IP associado.

```

Ethernet II, Src: Pegatron_10:57:46 (60:02:92:10:57:46), Dst: Broadcast (ff:ff:ff:ff:ff:ff)
  Destination: Broadcast (ff:ff:ff:ff:ff:ff)
    Address: Broadcast (ff:ff:ff:ff:ff:ff)
      ....1. .... = LG bit: Locally administered address (this is NOT the factory default)
      ....1. .... = IG bit: Group address (multicast/broadcast)
  Source: Pegatron_10:57:46 (60:02:92:10:57:46)
    Address: Pegatron_10:57:46 (60:02:92:10:57:46)
      ....0. .... = LG bit: Globally unique address (factory default)
      ....0. .... = IG bit: Individual address (unicast)
Type: ARP (0x0806)

```

**Figura 7.** Campo Ethernet II do pedido ARP.

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

Pela Figura 7 constatamos que vale 0x0806, identificando ARP.

**12) Qual o valor do campo ARP opcode? O que especifica?**

0x0001, que representa *Request*. Serve para identificar se se trata de um pedido(request) ou de uma resposta a outro pedido(reply).

```

Address Resolution Protocol (request)
  Hardware type: Ethernet (1)
  Protocol type: IPv4 (0x0800)
  Hardware size: 6
  Protocol size: 4
  Opcode: request (1)
  Sender MAC address: Pegatron_10:57:46 (60:02:92:10:57:46)
  Sender IP address: 192.168.2.7
  Target MAC address: 00:00:00_00:00:00 (00:00:00:00:00:00)
  Target IP address: 192.168.2.1

```

**Figura 8.** Informação ARP do pedido.

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

Endereços MAC e IPv4, pela Figura 8, dado que este protocolo define o mapeamento entre esses tipos de endereços.

**14) Explícite que tipo de pedido ou pergunta é feita pelo host de origem?**

2 0.607675	Pegatron_10:57:46	Broadcast	ARP	42 Who has 192.168.2.1? Tell 192.168.2.7
------------	-------------------	-----------	-----	--

**Figura 9.** Pedido efetuado.

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

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

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

Na Figura 10 está representada a resposta ao pedido ARP. O campo opcode vale 0x0002, pelo que identifica que se trata da *reply* ao pedido efetuado, onde no campo *Sender MAC address* se encontra a resposta a este último, c8:2a:14:58:01:60.

```

> Frame 3: 60 bytes on wire (480 bits), 60 bytes captured (480 bits) on interface 0
> Ethernet II, Src: Apple_58:01:60 (c8:2a:14:58:01:60), Dst: Pegatron_10:57:46 (60:02:92:10:57:46)
  Address Resolution Protocol (reply)
    Hardware type: Ethernet (1)
    Protocol type: IPv4 (0x0800)
    Hardware size: 6
    Protocol size: 4
    Opcode: reply (2)
    Sender MAC address: Apple_58:01:60 (c8:2a:14:58:01:60)
    Sender IP address: 192.168.2.1
    Target MAC address: Pegatron_10:57:46 (60:02:92:10:57:46)
    Target IP address: 192.168.2.7

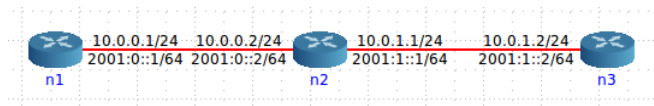
0000  60 02 92 10 57 46 c8 2a 14 58 01 60 08 06 00 01  `...WF.*.X.`...
0010  08 00 06 04 00 02 c8 2a 14 58 01 60 c0 a8 02 01  .....*.X.`...
0020  60 02 92 10 57 46 c0 a8 02 07 00 00 00 00 00 00  `...WF.....
0030  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  .....

```

**Figura 10.** Resposta ao pedido ARP.

### 2.3 ARP numa topologia CORE

Preparamos a topologia CORE conforme o descrito:



**Figura 11.** Topologia CORE.

**16) Com auxílio do comando ifconfig obtenha os endereços Ethernet das interfaces dos diversos routers.**

$n_1$ : eth0 - 00:00:00:aa:00:00  
 $n_2$ : eth0 - 00:00:00:aa:00:01  
       eth1 - 00:00:00:aa:00:02  
 $n_3$ : eth1 - 00:00:00:aa:00:03

```

root@n1:/tmp/pycore.36408/n1.conf# ifconfig
eth0      Link encap:Ethernet  HWaddr 00:00:00:aa:00:00
          inet addr:10.0.0.1  Bcast:0.0.0.0  Mask:255.255.255.0
          inet6 addr: fe80::200::ff:feaa:0/64 Scope:Link
          inet6 addr: 2001::1/64 Scope:Global
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:45 errors:0 dropped:0 overruns:0 frame:0
          TX packets:14 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:8366 (8.3 KB)  TX bytes:1240 (1.2 KB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:16386  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)

root@n2:/tmp/pycore.36408/n2.conf# ifconfig
eth0      Link encap:Ethernet  HWaddr 00:00:00:aa:00:01
          inet addr:10.0.0.2  Bcast:0.0.0.0  Mask:255.255.255.0
          inet6 addr: fe80::200::ff:feaa:1/64 Scope:Link
          inet6 addr: 2001::2/64 Scope:Global
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:76 errors:0 dropped:0 overruns:0 frame:0
          TX packets:50 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:11292 (11.2 KB)  TX bytes:4932 (4.9 KB)

eth1      Link encap:Ethernet  HWaddr 00:00:00:aa:00:02
          inet addr:10.0.0.11 Bcast:0.0.0.0  Mask:255.255.255.0
          inet6 addr: fe80::200::ff:feaa:2/64 Scope:Link
          inet6 addr: 2001::11/64 Scope:Global
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:74 errors:0 dropped:0 overruns:0 frame:0
          TX packets:40 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:11208 (11.2 KB)  TX bytes:4756 (4.7 KB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:16386  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)

root@n3:/tmp/pycore.36408/n3.conf# ifconfig
eth0      Link encap:Ethernet  HWaddr 00:00:00:aa:00:03
          inet addr:10.0.0.12 Bcast:0.0.0.0  Mask:255.255.255.0
          inet6 addr: fe80::200::ff:feaa:3/64 Scope:Link
          inet6 addr: 2001::12/64 Scope:Global
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:79 errors:0 dropped:0 overruns:0 frame:0
          TX packets:44 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:11866 (11.8 KB)  TX bytes:4168 (4.1 KB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:16386  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)

```

**Figura 12.** Resultado de executar o comando ifconfig nos *routers*.



**17) Usando o comando arp obtenha as caches arp dos diversos sistemas.**

Encontra-se o resultado obtido para n1, n2 e n3, nas figuras 13,14 e 15, respetivamente.

```
root@n1:/tmp/pycore.36408/n1.conf# arp
root@n1:/tmp/pycore.36408/n1.conf# arp -a
root@n1:/tmp/pycore.36408/n1.conf#
```

**Figura 13.** Resultado de executar os comandos arp e arp -a em n1.

```
root@n2:/tmp/pycore.36408/n2.conf# arp
Endereço TipoHW EndereçoHW Opções Máscara Interface
A0 ether 00:00:00:aa:00:00 C eth0
10.0.1.2 ether 00:00:00:aa:00:03 C eth1
root@n2:/tmp/pycore.36408/n2.conf# arp -a
A0 (10.0.0.1) em 00:00:00:aa:00:00 [ether] em eth0
? (10.0.1.2) em 00:00:00:aa:00:03 [ether] em eth1
root@n2:/tmp/pycore.36408/n2.conf#
```

**Figura 14.** Resultado de executar os comandos arp e arp -a em n2.

```
root@n3:/tmp/pycore.36408/n3.conf# arp
root@n3:/tmp/pycore.36408/n3.conf# arp -a
root@n3:/tmp/pycore.36408/n3.conf#
```

**Figura 15.** Resultado de executar os comandos arp e arp -a em n3.

**18) Faça ping de n1 para n2. Que modificações observa nas caches ARP desses sistemas? Faça ping de n1 para n3. Consulte as caches ARP. Que conclui?**

Na primeira fase, como mostra a Figura 16, é adicionada à tabela ARP de n1 uma entrada com a associação do endereço MAC de n2 e o seu respectivo endereço IP. Após o ping de n1 e n3, n3 passa a ter informação na sua tabela ARP relativa a n2, que possibilita a comunicação entre n1 e n2, como podemos ver na Figura 17.

```

root@n1:/tmp/pycore.36408/n1.conf# ping 10.0.0.2
PING 10.0.0.2 (10.0.0.2) 56(84) bytes of data.
64 bytes from 10.0.0.2: icmp_req=1 ttl=64 time=0.118 ms
64 bytes from 10.0.0.2: icmp_req=2 ttl=64 time=0.063 ms
64 bytes from 10.0.0.2: icmp_req=3 ttl=64 time=0.074 ms
64 bytes from 10.0.0.2: icmp_req=4 ttl=64 time=0.064 ms
64 bytes from 10.0.0.2: icmp_req=5 ttl=64 time=0.062 ms
^C
--- 10.0.0.2 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 3996ms
rtt min/avg/max/mdev = 0.062/0.076/0.118/0.022 ms
root@n1:/tmp/pycore.36408/n1.conf# arp
Endereço TipoHW EndereçoHW Opções Máscara Interface
A1 ether 00:00:00:aa:00:01 C eth0
root@n1:/tmp/pycore.36408/n1.conf# arp -a
A1 (10.0.0.2) em 00:00:00:aa:00:01 [ether] em eth0
root@n1:/tmp/pycore.36408/n1.conf#

root@n2:/tmp/pycore.36408/n2.conf# arp
Endereço TipoHW EndereçoHW Opções Máscara Interface
A0 ether 00:00:00:aa:00:00 C eth0
root@n2:/tmp/pycore.36408/n2.conf# arp -a
A0 (10.0.0.1) em 00:00:00:aa:00:00 [ether] em eth0
root@n2:/tmp/pycore.36408/n2.conf#

```

**Figura 16.** Alteração na tabela ARP de n1 e na de n2 após fazer ping de n1 para n2.

```

root@n1:/tmp/pycore.36408/n1.conf# ping 10.0.1.2
PING 10.0.1.2 (10.0.1.2) 56(84) bytes of data.
64 bytes from 10.0.1.2: icmp_req=1 ttl=63 time=0.186 ms
64 bytes from 10.0.1.2: icmp_req=2 ttl=63 time=0.086 ms
64 bytes from 10.0.1.2: icmp_req=3 ttl=63 time=0.085 ms
64 bytes from 10.0.1.2: icmp_req=4 ttl=63 time=0.085 ms
64 bytes from 10.0.1.2: icmp_req=5 ttl=63 time=0.098 ms
^C
--- 10.0.1.2 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 3999ms
rtt min/avg/max/mdev = 0.085/0.108/0.186/0.039 ms
root@n1:/tmp/pycore.36408/n1.conf# arp
Endereço TipoHW EndereçoHW Opções Máscara Interface
A1 ether 00:00:00:aa:00:01 C eth0
root@n1:/tmp/pycore.36408/n1.conf# arp -a
A1 (10.0.0.2) em 00:00:00:aa:00:01 [ether] em eth0
root@n1:/tmp/pycore.36408/n1.conf#

root@n2:/tmp/pycore.36408/n2.conf# arp
Endereço TipoHW EndereçoHW Opções Máscara Interface
A0 ether 00:00:00:aa:00:00 C eth0
10.0.1.2 ether 00:00:00:aa:00:03 C eth1
root@n2:/tmp/pycore.36408/n2.conf# arp -a
A0 (10.0.0.1) em 00:00:00:aa:00:00 [ether] em eth0
? (10.0.1.2) em 00:00:00:aa:00:03 [ether] em eth1
root@n2:/tmp/pycore.36408/n2.conf#

root@n3:/tmp/pycore.36408/n3.conf# arp
Endereço TipoHW EndereçoHW Opções Máscara Interface
10.0.1.1 ether 00:00:00:aa:00:02 C eth0
root@n3:/tmp/pycore.36408/n3.conf# arp -a
? (10.0.1.1) em 00:00:00:aa:00:02 [ether] em eth0
root@n3:/tmp/pycore.36408/n3.conf#

```

**Figura 17.** Alterações nas tabelas ARP dos sistemas após fazer ping de n1 para n3.

**19) Em n1 remova a entrada correspondente a n2. Coloque uma nova entrada para n2 com endereço Ethernet inexistente. O que acontece?**

Procedemos conforme o descrito, como mostra a Figura 18. Concluimos que n1 perde a conexão para n2 e, por consequência, para n3, como podemos verificar, por exemplo, na Figura 19.

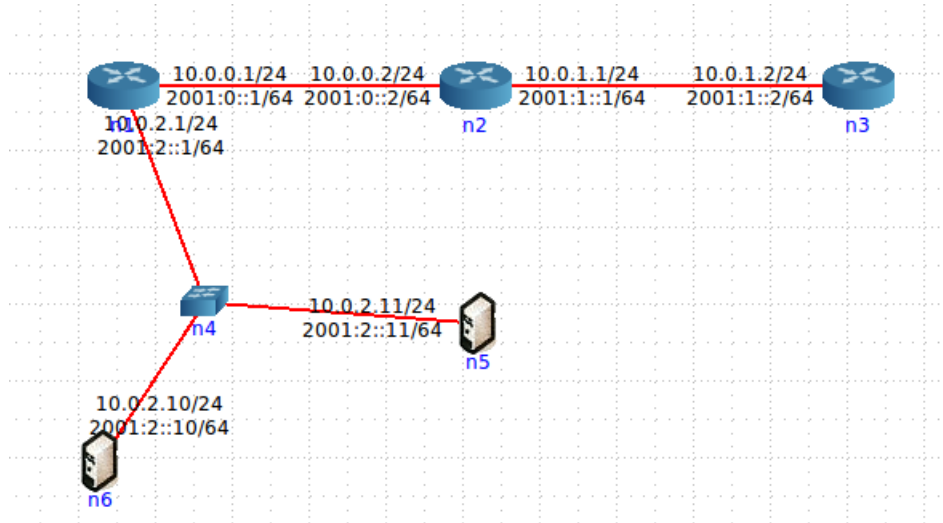
```
root@n1:/tmp/pycore.36408/n1.conf# arp
Endereço TipoHW EndereçoHW Opções Máscara Interface
A1 ether 00:00:00:aa:00:01 C eth0
root@n1:/tmp/pycore.36408/n1.conf# arp -d A1
root@n1:/tmp/pycore.36408/n1.conf# arp
Endereço TipoHW EndereçoHW Opções Máscara Interface
A1 (incompleto) eth0
root@n1:/tmp/pycore.36408/n1.conf# arp -s 00:00:00:aa:00:05
00:00:00:aa:00:05: Host desconhecido
root@n1:/tmp/pycore.36408/n1.conf# man arp -s
No manual entry for -s
root@n1:/tmp/pycore.36408/n1.conf# arp
Endereço TipoHW EndereçoHW Opções Máscara Interface
A1 (incompleto) eth0
root@n1:/tmp/pycore.36408/n1.conf# arp -s A1 00:00:00:aa:00:05
root@n1:/tmp/pycore.36408/n1.conf# arp
Endereço TipoHW EndereçoHW Opções Máscara Interface
A1 ether 00:00:00:aa:00:05 CM eth0
root@n1:/tmp/pycore.36408/n1.conf#
```

**Figura 18.** Remoção da entrada relativa a n2 da tabela ARP de n1 e inserção de um endereço Ethernet inexistente.

```
root@n1:/tmp/pycore.36408/n1.conf# ping 10.0.0.2
PING 10.0.0.2 (10.0.0.2) 56(84) bytes of data.
^C
--- 10.0.0.2 ping statistics ---
7 packets transmitted, 0 received, 100% packet loss, time 6032ms
```

**Figura 19.** Ping de n1 para n2.

Alteramos a topologia conforme o descrito.



**Figura 20.** Topologia CORE após inserir n4, n5 e n6 à rede.

**20) Faça ping de n6 para n5. Sem consultar a tabela ARP anote a entrada que, em sua opinião, é criada na tabela ARP de n6. Verifique, justificando, se a sua interpretação sobre a operação da rede Ethernet e protocolo ARP estava correto.**

Dado que o CORE começou por atribuir endereços MAC aos routers da forma 00:00:aa:00:0x,  $x \in [0, 3]$ , é espectável que para a nova interface de n1 seja atribuído o endereço 00:00:aa:00:04, para n5 00:00:aa:00:05 e para n6 00:00:aa:00:06. Ora, ao fazer ping de n6 para n5, seria de esperar que a entrada criada na tabela ARP de n6 atribuirá ao endereço IP de n5, 10.0.2.11, o endereço MAC 00:00:aa:00:05, como verificamos na Figura 33.

```
root@n6:/tmp/pycore.36409/n6.conf# arp
root@n6:/tmp/pycore.36409/n6.conf# ping 10.0.2.11
PING 10.0.2.11 (10.0.2.11) 56(84) bytes of data:
64 bytes from 10.0.2.11: icmp_req=1 ttl=64 time=0.120 ms
64 bytes from 10.0.2.11: icmp_req=2 ttl=64 time=0.062 ms
64 bytes from 10.0.2.11: icmp_req=3 ttl=64 time=0.061 ms
64 bytes from 10.0.2.11: icmp_req=4 ttl=64 time=0.067 ms
64 bytes from 10.0.2.11: icmp_req=5 ttl=64 time=0.066 ms
^C
--- 10.0.2.11 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 3997ms
rtt min/avg/max/mdev = 0.061/0.075/0.120/0.023 ms
root@n6:/tmp/pycore.36409/n6.conf# arp
Endereço TipoHW EndereçoHW Opções Máscara Interface
10.0.2.11 ether 00:00:00:aa:00:05 C
root@n6:/tmp/pycore.36409/n6.conf# arp -a
? (10.0.2.11) em 00:00:00:aa:00:05 [ether] em eth0
root@n6:/tmp/pycore.36409/n6.conf#
```

**Figura 21.** Tabela ARP de n6 após ping de n6 para n5.

## 3 Parte II

### 3.1 ARP Gratuito

**1) Identifique um pacote de pedido ARP gratuito originado pelo seu sistema. Verifique quantos pacotes ARP gratuito foram enviados e com que intervalo temporal?**

Na Figura 22 podemos ver um exemplo de pacote ARP gratuito enviado pelo nosso sistema. No total foram enviados 3 pacotes, com uma média de  $\mu \approx 56.037552s$  entre eles.

```
262 47.576857 Pegatron_10:57:46 Broadcast ARP 42 Gratuitous ARP for 192.168.2.7 (Request)
```

**Figura 22.** Exemplo de pedido ARP gratuito.

**2) 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?**

Podemos constatar, pela Figura 23, que o pedido ARP gratuito, para além da informação contida nos pedidos ARP habituais, apresenta uma flag *Is gratuitous* a indicar que se trata de um ARP gratuito. É esperado que o pedido não obtenha resposta, pois caso lhe enviassem, significaria que existe algum sistema na rede com o mesmo endereço IP do nosso dispositivo, identificando um conflito na rede local.

```
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: Pegatron_10:57:46 (60:02:92:10:57:46)
  Sender IP address: 192.168.2.7
  Target MAC address: 00:00:00_00:00:00 (00:00:00:00:00:00)
  Target IP address: 192.168.2.7
```

**Figura 23.** Campo ARP do pacote enviado.

### 3.2 Domínios de Colisão

**1) Faça ping de n1 para n4. Verifique com a opção tcpdump como flui o tráfego nas diversas interfaces dos vários dispositivos. Que conclui?**

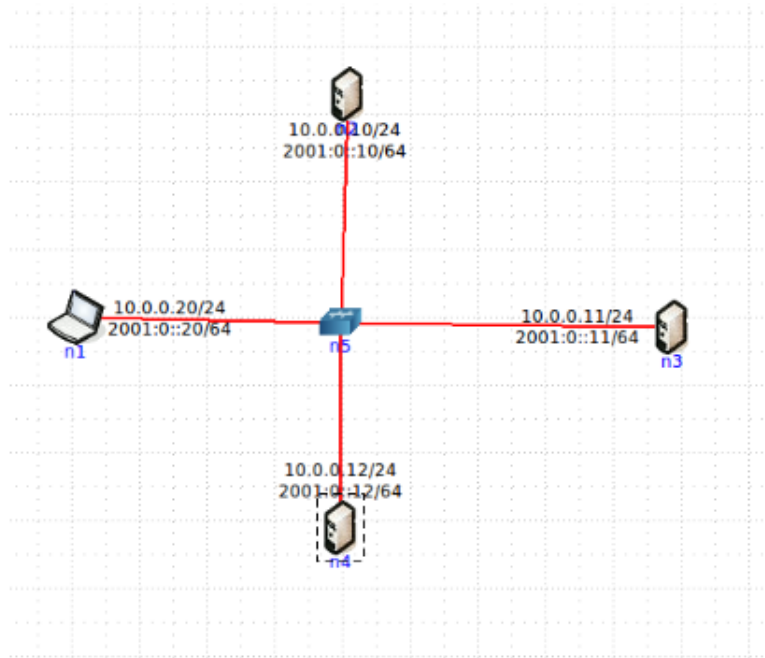


Figura 24. Rede com Hub

```

root@n1:/tmp/pycore.59837/n1.conf# ping 10.0.0.12
PING 10.0.0.12 (10.0.0.12) 56(84) bytes of data:
64 bytes from 10.0.0.12: icmp_req=1 ttl=64 time=0.108 ms
64 bytes from 10.0.0.12: icmp_req=2 ttl=64 time=0.112 ms
64 bytes from 10.0.0.12: icmp_req=3 ttl=64 time=0.102 ms
64 bytes from 10.0.0.12: icmp_req=4 ttl=64 time=0.110 ms
64 bytes from 10.0.0.12: icmp_req=5 ttl=64 time=0.106 ms
64 bytes from 10.0.0.12: icmp_req=6 ttl=64 time=0.110 ms
64 bytes from 10.0.0.12: icmp_req=7 ttl=64 time=0.109 ms
^C
--- 10.0.0.12 ping statistics ---
7 packets transmitted, 7 received, 0% packet loss, time 6002ms
rtt min/avg/max/mdev = 0.102/0.108/0.112/0.006 ms
root@n1:/tmp/pycore.59837/n1.conf#
  
```

Figura 25. ping de n1 para n4.

```

root@n2: /tmp/pycore.59837/n2.conf
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), capture size 65535 bytes
10:05:08.232136 IP 10.0.0.20 > A11: ICMP echo request, id 79, seq 1, length 64
10:05:08.232181 IP A11 > 10.0.0.20: ICMP echo reply, id 79, seq 1, length 64
10:05:09.233649 IP 10.0.0.20 > A11: ICMP echo request, id 79, seq 2, length 64
10:05:09.233689 IP A11 > 10.0.0.20: ICMP echo reply, id 79, seq 2, length 64
10:05:10.234368 IP 10.0.0.20 > A11: ICMP echo request, id 79, seq 3, length 64
10:05:10.234409 IP A11 > 10.0.0.20: ICMP echo reply, id 79, seq 3, length 64
10:05:11.234335 IP 10.0.0.20 > A11: ICMP echo request, id 79, seq 4, length 64
10:05:11.234384 IP A11 > 10.0.0.20: ICMP echo reply, id 79, seq 4, length 64
10:05:12.233353 IP 10.0.0.20 > A11: ICMP echo request, id 79, seq 5, length 64
10:05:12.233400 IP A11 > 10.0.0.20: ICMP echo reply, id 79, seq 5, length 64
10:05:13.233342 IP 10.0.0.20 > A11: ICMP echo request, id 79, seq 6, length 64
10:05:13.233390 IP A11 > 10.0.0.20: ICMP echo reply, id 79, seq 6, length 64
10:05:13.241291 ARP, Request who-has 10.0.0.20 tell A11, length 28
10:05:13.241319 ARP, Reply 10.0.0.20 is-at 00:00:00:aa:00:00 (oui Ethernet), length 28
10:05:14.234555 IP 10.0.0.20 > A11: ICMP echo request, id 79, seq 7, length 64
10:05:14.234600 IP A11 > 10.0.0.20: ICMP echo reply, id 79, seq 7, length 64
16 packets captured
16 packets received by filter
0 packets dropped by kernel
root@n2: /tmp/pycore.59837/n2.conf#

```

Figura 26. Tcpcdump em n2

```

root@n3: /tmp/pycore.59837/n3.conf
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), capture size 65535 bytes
10:05:08.232179 IP A11 > 10.0.0.20: ICMP echo reply, id 79, seq 1, length 64
10:05:09.233646 IP 10.0.0.20 > A11: ICMP echo request, id 79, seq 2, length 64
10:05:09.233696 IP A11 > 10.0.0.20: ICMP echo reply, id 79, seq 2, length 64
10:05:10.234368 IP 10.0.0.20 > A11: ICMP echo request, id 79, seq 3, length 64
10:05:10.234407 IP A11 > 10.0.0.20: ICMP echo reply, id 79, seq 3, length 64
10:05:11.234332 IP 10.0.0.20 > A11: ICMP echo request, id 79, seq 4, length 64
10:05:11.234381 IP A11 > 10.0.0.20: ICMP echo reply, id 79, seq 4, length 64
10:05:12.233349 IP 10.0.0.20 > A11: ICMP echo request, id 79, seq 5, length 64
10:05:12.233397 IP A11 > 10.0.0.20: ICMP echo reply, id 79, seq 5, length 64
10:05:13.233339 IP 10.0.0.20 > A11: ICMP echo request, id 79, seq 6, length 64
10:05:13.233388 IP A11 > 10.0.0.20: ICMP echo reply, id 79, seq 6, length 64
10:05:13.241287 ARP, Request who-has 10.0.0.20 tell A11, length 28
10:05:13.241318 ARP, Reply 10.0.0.20 is-at 00:00:00:aa:00:00 (oui Ethernet), length 28
10:05:14.234551 IP 10.0.0.20 > A11: ICMP echo request, id 79, seq 7, length 64
10:05:14.234598 IP A11 > 10.0.0.20: ICMP echo reply, id 79, seq 7, length 64
16 packets captured
16 packets received by filter
0 packets dropped by kernel
root@n3: /tmp/pycore.59837/n3.conf#

```

Figura 27. Tcpcdump em n3

```
root@n4:/tmp/pycore.59837/n4.conf
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), capture size 65535 bytes
10:05:08.232128 IP 10.0.0.20 > A11: ICMP echo request, id 79, seq 1, length 64
10:05:08.232159 IP A11 > 10.0.0.20: ICMP echo reply, id 79, seq 1, length 64
10:05:09.233642 IP 10.0.0.20 > A11: ICMP echo request, id 79, seq 2, length 64
10:05:09.233672 IP A11 > 10.0.0.20: ICMP echo reply, id 79, seq 2, length 64
10:05:10.234361 IP 10.0.0.20 > A11: ICMP echo request, id 79, seq 3, length 64
10:05:10.234389 IP A11 > 10.0.0.20: ICMP echo reply, id 79, seq 3, length 64
10:05:11.234328 IP 10.0.0.20 > A11: ICMP echo request, id 79, seq 4, length 64
10:05:11.234359 IP A11 > 10.0.0.20: ICMP echo reply, id 79, seq 4, length 64
10:05:12.233345 IP 10.0.0.20 > A11: ICMP echo request, id 79, seq 5, length 64
10:05:12.233376 IP A11 > 10.0.0.20: ICMP echo reply, id 79, seq 5, length 64
10:05:13.233334 IP 10.0.0.20 > A11: ICMP echo request, id 79, seq 6, length 64
10:05:13.233366 IP A11 > 10.0.0.20: ICMP echo reply, id 79, seq 6, length 64
10:05:13.241253 ARP, Request who-has 10.0.0.20 tell A11, length 28
10:05:13.241316 ARP, Reply 10.0.0.20 is-at 00:00:00:aa:00:00 (oui Ethernet), length 28
10:05:14.234547 IP 10.0.0.20 > A11: ICMP echo request, id 79, seq 7, length 64
10:05:14.234576 IP A11 > 10.0.0.20: ICMP echo reply, id 79, seq 7, length 64

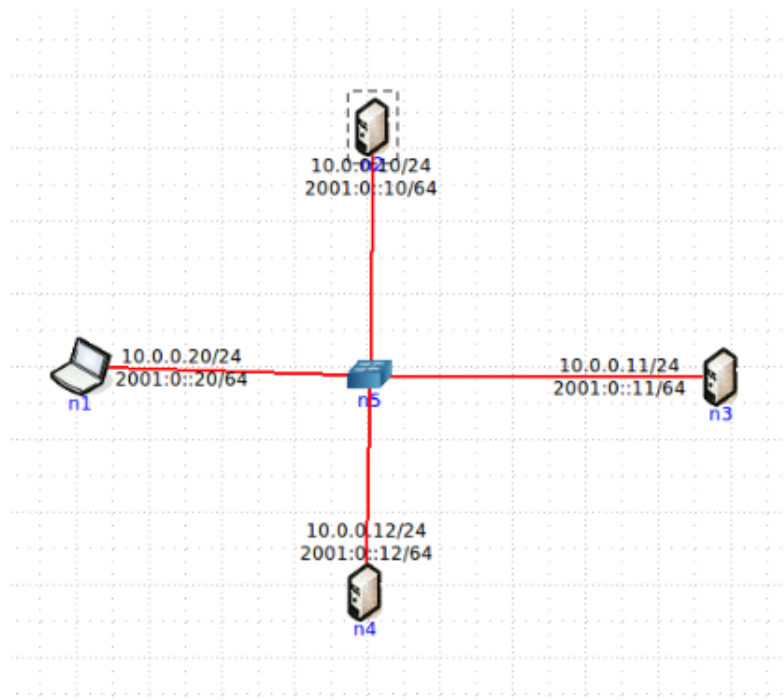
16 packets captured
16 packets received by filter
0 packets dropped by kernel
root@n4:/tmp/pycore.59837/n4.conf#
```

**Figura 28.** Tcpcdump em n4

Um hub ao receber um pacote de dados, reencaminha-o para todos os dispositivos que fazem parte da rede a que ele pertence, fazendo também a detecção de colisões. Através da análise do output resultante do comando tcpdump em cada um dos hosts da rede conclui-se o que foi dito anteriormente. Apesar de ser realizada uma transferência de dados entre os hosts n1 e n4, a informação também é enviada como tráfego nos restantes hosts.



2) 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.



**Figura 29.** Rede com Switch

```
root@n1:/tmp/pycore.59838/n1.conf# ping 10.0.0.12
PING 10.0.0.12 (10.0.0.12) 56(84) bytes of data:
64 bytes from 10.0.0.12: icmp_req=1 ttl=64 time=0.151 ms
64 bytes from 10.0.0.12: icmp_req=2 ttl=64 time=0.082 ms
64 bytes from 10.0.0.12: icmp_req=3 ttl=64 time=0.082 ms
64 bytes from 10.0.0.12: icmp_req=4 ttl=64 time=0.078 ms
64 bytes from 10.0.0.12: icmp_req=5 ttl=64 time=0.081 ms
64 bytes from 10.0.0.12: icmp_req=6 ttl=64 time=0.046 ms
64 bytes from 10.0.0.12: icmp_req=7 ttl=64 time=0.078 ms
^C
--- 10.0.0.12 ping statistics ---
7 packets transmitted, 7 received, 0% packet loss, time 600ms
rtt min/avg/max/ndev = 0.046/0.085/0.151/0.030 ms
root@n1:/tmp/pycore.59838/n1.conf#
```

Figura 30. *ping* de n1 para n4.

```
root@n2:/tmp/pycore.59838/n2.conf# tcpdump
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), capture size 65535 bytes
^C10:08:28.511798 ARP, Request who-has A11 tell 10.0.0.20, length 28

1 packet captured
1 packet received by filter
0 packets dropped by kernel
root@n2:/tmp/pycore.59838/n2.conf#
```

Figura 31. *Tcpdump* em n2

```
root@n3:/tmp/pycore.59838/n3.conf# tcpdump
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), capture size 65535 bytes
^C10:08:28.511795 ARP, Request who-has A11 tell 10.0.0.20, length 28

1 packet captured
1 packet received by filter
0 packets dropped by kernel
root@n3:/tmp/pycore.59838/n3.conf#
```

Figura 32. *Tcpdump* em n3

```
root@n4:/tmp/pycore.59838/n4.conf
^C10:08:28.511792 ARP, Request who-has A11 tell 10.0.0.20, length 28
10:08:28.511834 ARP, Reply A11 is-at 00:00:00:aa:00:03 (oui Ethernet), length 28
10:08:28.511871 IP 10.0.0.20 > A11: ICMP echo request, id 72, seq 1, length 64
10:08:28.511885 IP A11 > 10.0.0.20: ICMP echo reply, id 72, seq 1, length 64
10:08:29.514047 IP 10.0.0.20 > A11: ICMP echo request, id 72, seq 2, length 64
10:08:29.514070 IP A11 > 10.0.0.20: ICMP echo reply, id 72, seq 2, length 64
10:08:30.513316 IP 10.0.0.20 > A11: ICMP echo request, id 72, seq 3, length 64
10:08:30.513340 IP A11 > 10.0.0.20: ICMP echo reply, id 72, seq 3, length 64
10:08:31.513423 IP 10.0.0.20 > A11: ICMP echo request, id 72, seq 4, length 64
10:08:31.513445 IP A11 > 10.0.0.20: ICMP echo reply, id 72, seq 4, length 64
10:08:32.513377 IP 10.0.0.20 > A11: ICMP echo request, id 72, seq 5, length 64
10:08:32.513400 IP A11 > 10.0.0.20: ICMP echo reply, id 72, seq 5, length 64
10:08:33.513306 ARP, Request who-has 10.0.0.20 tell A11, length 28
10:08:33.513349 ARP, Reply 10.0.0.20 is-at 00:00:00:aa:00:00 (oui Ethernet), length 28
10:08:33.513407 IP 10.0.0.20 > A11: ICMP echo request, id 72, seq 6, length 64
10:08:33.513419 IP A11 > 10.0.0.20: ICMP echo reply, id 72, seq 6, length 64
10:08:34.513315 IP 10.0.0.20 > A11: ICMP echo request, id 72, seq 7, length 64
10:08:34.513337 IP A11 > 10.0.0.20: ICMP echo reply, id 72, seq 7, length 64

18 packets captured
18 packets received by filter
0 packets dropped by kernel
root@n4:/tmp/pycore.59838/n4.conf#
```

Figura 33. Tcpcdump em n4

O hub permite detetar, se acontecer, uma colisão entre tramas enviando um sinal, que permite que a trama em conflito volte à máquina de origem e fique à espera até que o hub termine a sua operação atual. Por outro lado, o switch neste domínio, permitem evitar colisões de tramas ao enviar dados que são relevantes a certos dispositivos na rede. Ao analisarmos o tráfego nos dispositivos podemos concluir que nos hosts n1 e n4 se verificou a captura de dados, e no n2 e n3 não se observa qualquer tráfego. Ou seja, como o comando ping foi executado apenas entre n1 e n4, a transferência de dados apenas é relevante para eles próprios, não realizando qualquer tráfego nos restantes. Com isto podemos concluir que o switch é um dispositivo que atenua melhor os conflitos.

## 4 Conclusão

Executada a II podemos concluir que houve um aumento da compreensão sobre a execução dos processos e objetivo predefinido, (relacionar e fornecer informação entre a camada L2 e L3), do protocolo ARP que contribuiu também para uma melhor assimilação dos conceitos da abordados na parte I, Ethernet e ARP. Este trabalho prático permitiu-nos ainda aprofundar o nosso conhecimento sobre as ferramentas utilizadas, ferramenta de simulação de redes (CORE) e de captura e análise de tramas (Wireshark). A utilização do Wireshark foi fundamental para este trabalho pois permitiu-nos verificar quais os protocolos envolvidos, qual o encapsulamento correto no processo de transferência e verificar outras propriedades importantes, que nos permitem saber informações como quais os endereços envolvidos, tipo da mensagem ARP e a análise de um pedido ARP.