

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

Diogo Afonso Costa, Daniel Maia, and Vitor Castro

University of Minho, Department of Informatics, 4710-057 Braga, Portugal  
e-mail: {a78034,a77531,a77870}@alunos.uminho.pt

**Abstract.** Resumo...

## **1 Introdução**

## 2 Parte I - Captura e análise de Tramas Ethernet

### 2.1 Exercício 1

#### Questão

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

#### Resposta

*Destination:* Vmware\_d2:19:f0 (00:0c:29:d2:19:f0)

*Source:* LcfcHefe\_66:65:4a (68:f7:28:66:65:4a)

#### Realização

```
236 7.772854      192.168.100.209      193.136.19.20      HTTP      386      GET / HTTP/1.1
Frame 236: 386 bytes on wire (3088 bits), 386 bytes captured (3088 bits) on interface 0
Ethernet II, Src: LcfcHefe_66:65:4a (68:f7:28:66:65:4a), Dst: Vmware_d2:19:f0 (00:0c:29:d2:19:f0)
  Destination: Vmware_d2:19:f0 (00:0c:29:d2:19:f0)
  Source: LcfcHefe_66:65:4a (68:f7:28:66:65:4a)
  Type: IPv4 (0x0800)
Internet Protocol Version 4, Src: 192.168.100.209, Dst: 193.136.19.20
Transmission Control Protocol, Src Port: 52130, Dst Port: 80, Seq: 1, Ack: 1, Len: 332
Hypertext Transfer Protocol
```

Fig. 1: Endereços MAC do HTTP GET.

## **2.2 Exercício 2**

### **Questão**

Identifique a que sistemas se referem. Justifique.

### **Resposta**

O MAC *destination* refere ao comutador da rede local. O MAC *source* refere à máquina que enviou o pedido; neste caso, o computador utilizado.

### **Realização**

Ao nível da ligação de dados, as máquinas apenas comunicam com máquinas adjacentes.

## 2.3 Exercício 3

### Questão

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

### Resposta

*Type*: 0x0800; Este valor indica o tipo IPv4.

### Realização

```
236 7.772854      192.168.100.209      193.136.19.20      HTTP      386      GET / HTTP/1.1
Frame 236: 386 bytes on wire (3088 bits), 386 bytes captured (3088 bits) on interface 0
Ethernet II, Src: LcfcHefe_66:65:4a (68:f7:28:66:65:4a), Dst: Vmware_d2:19:f0 (00:0c:29:d2:19:f0)
  Destination: Vmware_d2:19:f0 (00:0c:29:d2:19:f0)
  Source: LcfcHefe_66:65:4a (68:f7:28:66:65:4a)
  Type: IPv4 (0x0800)
Internet Protocol Version 4, Src: 192.168.100.209, Dst: 193.136.19.20
Transmission Control Protocol, Src Port: 52130, Dst Port: 80, Seq: 1, Ack: 1, Len: 332
Hypertext Transfer Protocol
```

Fig. 2: O campo *Type* da trama.

## 2.4 Exercício 4

### Questão

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.

### Resposta

Até ao caractere "G", existem 54 bytes dos 386 bytes do método, resultando num overhead percentual de  $(54/386) * 100 = 14\%$

### Realização

```
0000 00 0c 29 d2 19 f8 68 f7 28 66 65 4a 08 00 45 00 ..)...h.(fe).E.
0010 01 74 3c 8c 40 00 00 06 00 00 c9 a8 64 d1 c1 88 .t:8...d...
0020 13 14 c0 a2 00 50 99 8c 41 16 62 96 0b 8f 50 18 .....P..A...P.
0030 80 00 f5 7c 00 00 45 54 20 2f 20 48 54 54 50 ...].G[/ HTTP
0040 2f 31 2e 31 0d 0a 41 63 63 65 70 74 3a 20 74 65 /1.1..Ac cept: te
0050 70 74 2f 68 74 6d 6c 2c 20 61 70 70 6c 69 63 61 xt/html, applica
0060 74 69 6f 6e 2f 70 68 74 6d 6c 2b 78 6d 6c 2c 20 tion/xht m+xml),
0070 69 6d 61 67 65 2f 6a 70 72 2c 20 2a 2f 2a 8d 0a image/jv r, /*..
0080 41 63 63 65 70 74 2d 4c 61 6e 67 75 61 67 65 3a Accept-L anguage:
0090 20 70 74 2d 50 54 0d 0a 55 73 65 72 2d 41 67 65 pt-PT.. User-Age
00a0 6e 74 3a 20 4d 6f 7a 69 6c 6c 61 2f 35 2e 30 20 nt: Mozil la/5.0
00b0 20 57 69 6e 6d 6f 77 73 20 4e 54 20 31 30 2e 30 (Windows NT 10.0
00c0 3b 20 57 69 6e 36 34 3b 20 78 36 34 29 20 41 70 ; Win64; x64) Ap
00d0 70 6c 65 57 65 62 4b 69 74 2f 35 33 37 2e 33 36 pleiebel t/537.36
00e0 20 2b 4b 54 4d 4c 2c 20 6c 69 6b 65 20 47 65 (OTM), like Ge
00f0 63 6b 6f 29 20 43 68 72 6f 6d 65 2f 35 32 2e 30 cko) Chr om/52.0
0100 2e 32 37 34 33 2e 31 31 36 20 53 61 66 61 72 69 .2743.11 6 Safari
0110 2f 35 33 37 2e 33 36 20 45 64 67 65 2f 31 35 2e /537.36 Edge/15.
0120 31 35 30 36 33 0d 0a 41 63 63 65 70 74 2d 45 6e 15063..A ccept-En
0130 63 6f 64 69 6e 67 3a 20 67 7a 69 70 2c 20 64 65 coding: grip, de
0140 66 6c 61 74 65 0d 0a 4b 6f 73 74 3a 20 6d 69 65 flate..H ost: sia
0150 69 2e 64 69 2e 75 6d 69 6e 68 6f 2e 70 74 0d 0a i.di.umi nho.pt..
```

Bytes 54-56: Request Method (http.request.method)

Fig. 3: O byte com o caractere "G" aparece após 54 outros.

## 2.5 Exercício 5

### Questão

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.

### Resposta

O valor hexadecimal FCS da trama é 0x0d0a0d0a, mas deveria ser 0x971613cf. Disto conclui que a NIC não gerou código de verificação de erros, visto que, se a trama estivesse incorreta, o pacote teria sido enviado.

### Realização

```
236 7.772854      192.168.100.209      193.136.19.20      TCP      386      52130 → 80 [PSH, ACK] Seq=1 Ack=1 Win=262144
Len=328 [TCP segment of a reassembled PDU] [ETHERNET FRAME CHECK SEQUENCE INCORRECT]
Frame 236: 386 bytes on wire (3088 bits), 386 bytes captured (3088 bits) on interface 0
Ethernet II, Src: LcfcHefe_66:65:4a (68:f7:28:66:65:4a), Dst: Vmware_d2:19:f0 (00:0c:29:d2:19:f0)
Destination: Vmware_d2:19:f0 (00:0c:29:d2:19:f0)
Source: LcfcHefe_66:65:4a (68:f7:28:66:65:4a)
Type: IPv4 (0x0800)
Frame check sequence: 0x0d0a0d0a incorrect, should be 0x971613cf
[Expert Info (Error/Checksum): Bad checksum [should be 0x971613cf]]
[FCS Status: Bad]
Internet Protocol Version 4, Src: 192.168.100.209, Dst: 193.136.19.20
Transmission Control Protocol, Src Port: 52130, Dst Port: 80, Seq: 1, Ack: 1, Len: 328
```

Fig. 4: Se o pacote tivesse um erro, não seria enviado e, por extensão, capturado.

## 2.6 Exercício 6

### Questão

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

### Resposta

*Address:* Vmware\_d2:19:f0 (00:0c:29:d2:19:f0)

Este endereço corresponde ao comutador da rede local visto que, ao nível dos dados, máquinas apenas comunicam com as máquinas adjacentes através do protocolo ARP.

### Realização

```
299 7.782820      193.136.19.20      192.168.100.209      HTTP      783      HTTP/1.1 200 OK (text/html)
Frame 299: 783 bytes on wire (6264 bits), 783 bytes captured (6264 bits) on interface 0
Ethernet II, Src: Vmware_d2:19:f0 (00:0c:29:d2:19:f0), Dst: LcfcHefe_66:65:4a (68:f7:28:66:65:4a)
  Destination: LcfcHefe_66:65:4a (68:f7:28:66:65:4a)
  Source: Vmware_d2:19:f0 (00:0c:29:d2:19:f0)
  Type: IPv4 (0x0800)
Internet Protocol Version 4, Src: 193.136.19.20, Dst: 192.168.100.209
Transmission Control Protocol, Src Port: 80, Dst Port: 52130, Seq: 51101, Ack: 333, Len: 729
[36 Reassembled TCP Segments (51829 bytes): #238(1460), #240(1460), #242(1460), #243(1460), #245(1460), #246(1460), #248(1460),
#249(1460), #250(1460), #253(1460), #254(1460), #255(1460), #257(1460), #258(1460), #259(1460), #260(1460), #266]
Hypertext Transfer Protocol
Line-based text data: text/html
```

Fig. 5: O endereço Ethernet da fonte da resposta.

## 2.7 Exercício 7

### Questão

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

### Resposta

*Address:* LcfcHefe\_66:65:4a (68:f7:28:66:65:4a)

Este endereço corresponde ao computador utilizado, que originalmente enviou o pedido HTTP GET.

### Realização

```
299 7.782820      193.136.19.20      192.168.100.209      HTTP      783      HTTP/1.1 200 OK (text/html)
Frame 299: 783 bytes on wire (6264 bits), 783 bytes captured (6264 bits) on interface 0
Ethernet II, Src: Vmware_d2:19:f0 (00:0c:29:d2:19:f0), Dst: LcfcHefe_66:65:4a (68:f7:28:66:65:4a)
  Destination: LcfcHefe_66:65:4a (68:f7:28:66:65:4a)
  Source: Vmware_d2:19:f0 (00:0c:29:d2:19:f0)
  Type: IPv4 (0x0800)
Internet Protocol Version 4, Src: 193.136.19.20, Dst: 192.168.100.209
Transmission Control Protocol, Src Port: 80, Dst Port: 52130, Seq: 51101, Ack: 333, Len: 729
[36 Reassembled TCP Segments (51829 bytes): #238(1460), #240(1460), #242(1460), #243(1460), #245(1460), #246(1460), #248(1460),
#249(1460), #250(1460), #253(1460), #254(1460), #255(1460), #257(1460), #258(1460), #259(1460), #260(1460), #266]
Hypertext Transfer Protocol
Line-based text data: text/html
```

Fig. 6: O endereço Ethernet do destino da resposta.



## 2.8 Exercício 8

### Questão

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

### Resposta

Na trama capturada está contido um protocolo Ethernet II, no qual está encapsulado um protocolo IPv4 que, por sua vez, transporta um protocolo TCP, no qual está contido um protocolo Hypertext Transfer Protocol (HTTP).

### Realização

```
299 7.782820      193.136.19.20      192.168.100.209      HTTP      783      HTTP/1.1 200 OK (text/html)
Frame 299: 783 bytes on wire (6264 bits), 783 bytes captured (6264 bits) on interface 0
Ethernet II, Src: Vmware_d2:19:f0 (00:0c:29:d2:19:f0), Dst: LcfcHefe_66:65:4a (68:f7:28:66:65:4a)
Internet Protocol Version 4, Src: 193.136.19.20, Dst: 192.168.100.209
Transmission Control Protocol, Src Port: 80, Dst Port: 52130, Seq: 51101, Ack: 333, Len: 729
[36 Reassembled TCP Segments (51829 bytes): #238(1460), #240(1460), #242(1460), #243(1460), #245(1460), #246(1460), #248(1460),
#249(1460), #250(1460), #253(1460), #254(1460), #255(1460), #257(1460), #258(1460), #259(1460), #260(1460), #266]
Hypertext Transfer Protocol
Line-based text data: text/html
```

Fig. 7: Os protocolos contidos na trama de resposta.

## 2.9 Exercício 9

### Questão

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

### Resposta

A versão linux instalada na máquina na qual este exercício foi realizado não tem o pacote *ARP* disponível pois este foi deprecado. Contudo, o comando `ip neigh`, realiza a mesma operação, isto é, permite observar o conteúdo da tabela *ARP*.

Desta forma, consultando o manual referente ao comando a cima mencionado (`man ip-neighbour`), percebe-se que o *output* é segmentado em 4 colunas. A primeira indica o endereço IP da interface da máquina na vizinhança, à qual se refere a entrada na tabela. A segunda coluna, traduz a interface na qual o *host* da vizinhança se encontra ligado. A terceira coluna apresenta o endereço MAC do *host* a que esta entrada na tabela se refere. Por último, a quarta coluna indica o estado da ligação entre os dois sistemas.

### Realização

```
to ADDRESS (default)
    the protocol address of the neighbour. It is either an IPv4 or IPv6 address.

dev NAME
    the interface to which this neighbour is attached.

lladdr LLADDRESS
    the link layer address of the neighbour. LLADDRESS can also be null.

nud STATE
    the state of the neighbour entry. nud is an abbreviation for 'Neighbour Unreachability Detection'. The state can take one of the following values:

    permanent
        the neighbour entry is valid forever and can be only be removed administratively.

    noarp
        the neighbour entry is valid. No attempts to validate this entry will be made but it can be removed when its lifetime expires.

    reachable
        the neighbour entry is valid until the reachability timeout expires.

    stale
        the neighbour entry is valid but suspicious. This option to ip neigh does not change the neighbour state if it was valid and the address is not changed by this command.

    none
        this is a pseudo state used when initially creating a neighbour entry or after trying to remove it before it becomes free to do so.

    incomplete
        the neighbour entry has not (yet) been validated/resolved.

    delay
        neighbor entry validation is currently delayed.

    probe
        neighbor is being probed.

    failed
        max number of probes exceeded without success, neighbor validation has ultimately failed.
```

Fig. 8: Manual do comando `ip neigh`.

## 2.10 Exercício 10

### Questão

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?

### Resposta

Endereço origem: 2c:60:0c:6b:85:44

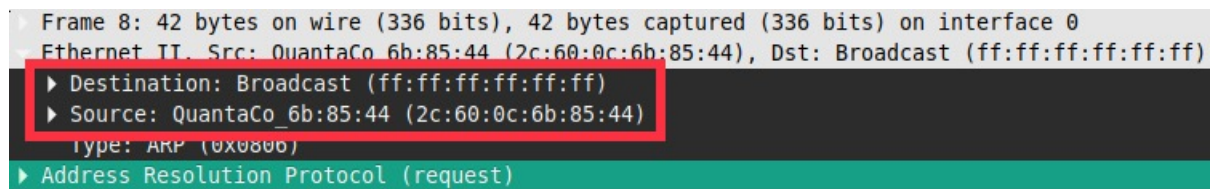
Endereço destino: ff:ff:ff:ff:ff:ff (broadcast)

A estratégia utilizada para resolver o exercício foi fazendo *ping* para outra máquina do laboratório, nomeadamente, 192.168.100.205.

Desta forma, a máquina que estava a relizar o *ping* apenas tinha a informação do IP da interface da máquina destino. No entanto, dado que o *host* destino se encontra na vizinhança da máquina de origem, é possível enviar a trama sem que esta vá ao AP. Para tal é usado o protocolo ARP por forma a descobrir o MAC do *host* para que a comunicação seja feita diretamente.

Deste modo, é enviado para todos os sistemas que se encontram na mesma rede local um ARP *request*, por forma a descobrir o endereço MAC a que pertence o IP 192.168.100.205, daí o endereço de destino ser ff:ff:ff:ff:ff:ff.

### Realização



```
▶ Frame 8: 42 bytes on wire (336 bits), 42 bytes captured (336 bits) on interface 0
  Ethernet II, Src: QuantaCo_6b:85:44 (2c:60:0c:6b:85:44), Dst: Broadcast (ff:ff:ff:ff:ff:ff)
    ▶ Destination: Broadcast (ff:ff:ff:ff:ff:ff)
    ▶ Source: QuantaCo_6b:85:44 (2c:60:0c:6b:85:44)
      type: ARP (0x0806)
    ▶ Address Resolution Protocol (request)
```

Fig. 9: Pedido *ARP* enviado em *broadcast*.

## 2.11 Exercício 11

### Questão

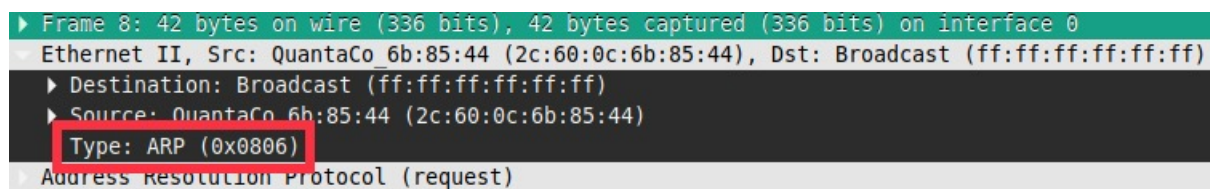
Qual o valor hexadecimal do campo tipo da trama Ethernet? O que indica?

### Resposta

Type: ARP (0x0806)

Significa que a trama *Ethernet* leva encapsulado o protocolo ARP.

### Realização



```
▶ Frame 8: 42 bytes on wire (336 bits), 42 bytes captured (336 bits) on interface 0
  ▶ Ethernet II, Src: QuantaCo_6b:85:44 (2c:60:0c:6b:85:44), Dst: Broadcast (ff:ff:ff:ff:ff:ff)
    ▶ Destination: Broadcast (ff:ff:ff:ff:ff:ff)
    ▶ Source: QuantaCo_6b:85:44 (2c:60:0c:6b:85:44)
    ▶ Type: ARP (0x0806)
  ▶ Address Resolution Protocol (request)
```

Fig. 10: Tipo da trama *Ethernet*.

## 2.12 Exercício 12

### Questão

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

### Resposta

Opcode: request (1)

Segundo o padrão descrito no RFC826 [1]: *"The opcode is to determine if this is a request (which may cause a reply) or a reply to a previous request. 16 bits for this is overkill, but a flag (field) is needed."*

Desta forma, pode-se afirmar que a trama em questão representa um ARP *request*.

### Realização

```
▼ Address Resolution Protocol (request)
  Hardware type: Ethernet (1)
  Protocol type: IPv4 (0x0800)
  Hardware size: 6
  Protocol size: 4
  Opcode: request (1)
  Sender MAC address: QuantaCo_6b:85:44 (2c:60:0c:6b:85:44)
  Sender IP address: 192.168.100.198
  Target MAC address: 00:00:00_00:00:00 (00:00:00:00:00:00)
  Target IP address: 192.168.100.205
```

Fig. 11: Valor do *opcode* do ARP *request*.

## 2.13 Exercício 13

### Questão

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

### Resposta

Os endereços contidos na mensagem ARP são os seguintes:

- Sender MAC address: 2c:60:0c:6b:85:44
- Sender IP address: 192.168.100.198
- Target MAC address: 00:00:00:00:00:00
- Target IP address: 192.168.100.254

Efetivamente, o facto de o *Target MAC address* estar todo a zero significa que está à espera de ser preenchido, assim que o *Target IP address* faça correspondência na máquina de destino.

### Realização

```
Address Resolution Protocol (request)
  Hardware type: Ethernet (1)
  Protocol type: IPv4 (0x0800)
  Hardware size: 6
  Protocol size: 4
  Opcode: request (1)
  Sender MAC address: QuantaCo_6b:85:44 (2c:60:0c:6b:85:44)
  Sender IP address: 192.168.100.198
  Target MAC address: 00:00:00_00:00:00 (00:00:00:00:00:00)
  Target IP address: 192.168.100.205
```

Fig. 12: Endereços contidos na mensagem ARP.

## **2.14 Exercício 14**

### **Questão**

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

### **Resposta**

O *host* de origem quer saber qual o endereço MAC da máquina que tem como IP da interface 192.168.100.205.

## 2.15 Exercício 15.a)

### Questão

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

### Resposta

Opcode: reply (2)

Este valor significa que esta trama é a resposta a uma trama ARP *request* recebida anteriormente.

### Realização

```
Address Resolution Protocol (reply)
  Hardware type: Ethernet (1)
  Protocol type: IPv4 (0x0800)
  Hardware size: 6
  Protocol size: 4
  Opcode: reply (2)
  Sender MAC address: HewlettP_02:a3:b0 (64:51:06:02:a3:b0)
  Sender IP address: 192.168.100.205
  Target MAC address: QuantaCo_6b:85:44 (2c:60:0c:6b:85:44)
  Target IP address: 192.168.100.198
```

Fig. 13: Campo *opcode* do ARP *reply*.



## 2.16 Exercício 15.b)

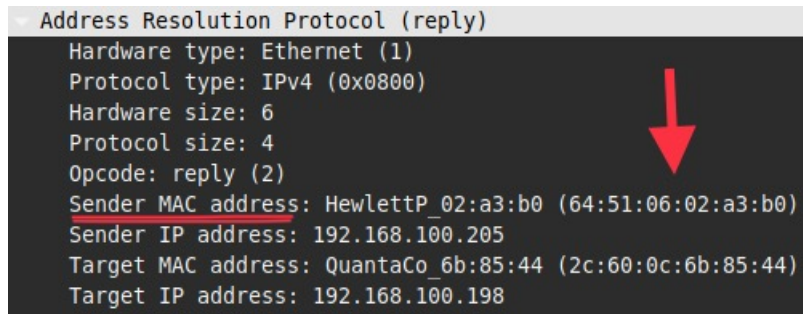
### Questão

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

### Resposta

– Sender MAC address: 64:51:06:02:a3:b0

### Realização



```
- Address Resolution Protocol (reply)
  Hardware type: Ethernet (1)
  Protocol type: IPv4 (0x0800)
  Hardware size: 6
  Protocol size: 4
  Opcode: reply (2)
  Sender MAC address: HewlettP_02:a3:b0 (64:51:06:02:a3:b0)
  Sender IP address: 192.168.100.205
  Target MAC address: QuantaCo_6b:85:44 (2c:60:0c:6b:85:44)
  Target IP address: 192.168.100.198
```

Fig. 14: Campo com a resposta ao pedido ARP.

## 2.17 Exercício 16

### Questão

Com auxílio do comando `ifconfig` obtenha os endereços Ethernet das interfaces dos diversos routers.

### Resposta

- MAC Router n1 (eth0): 00:00:00:aa:00:00
- MAC Router n2 (eth0): 00:00:00:aa:00:01
- MAC Router n2 (eth1): 00:00:00:aa:00:02
- MAC Router n3 (eth0): 00:00:00:aa:00:03

Pode-se observar que as interfaces ligadas são a eth0 (de n1) e eth0 (de n2), relativas ao caminho entre **n1** e **n2**, tal como eth1 (de n2) e eth0 (de n3), correspondentes ao caminho entre **n2** e **n3**.

### Realização

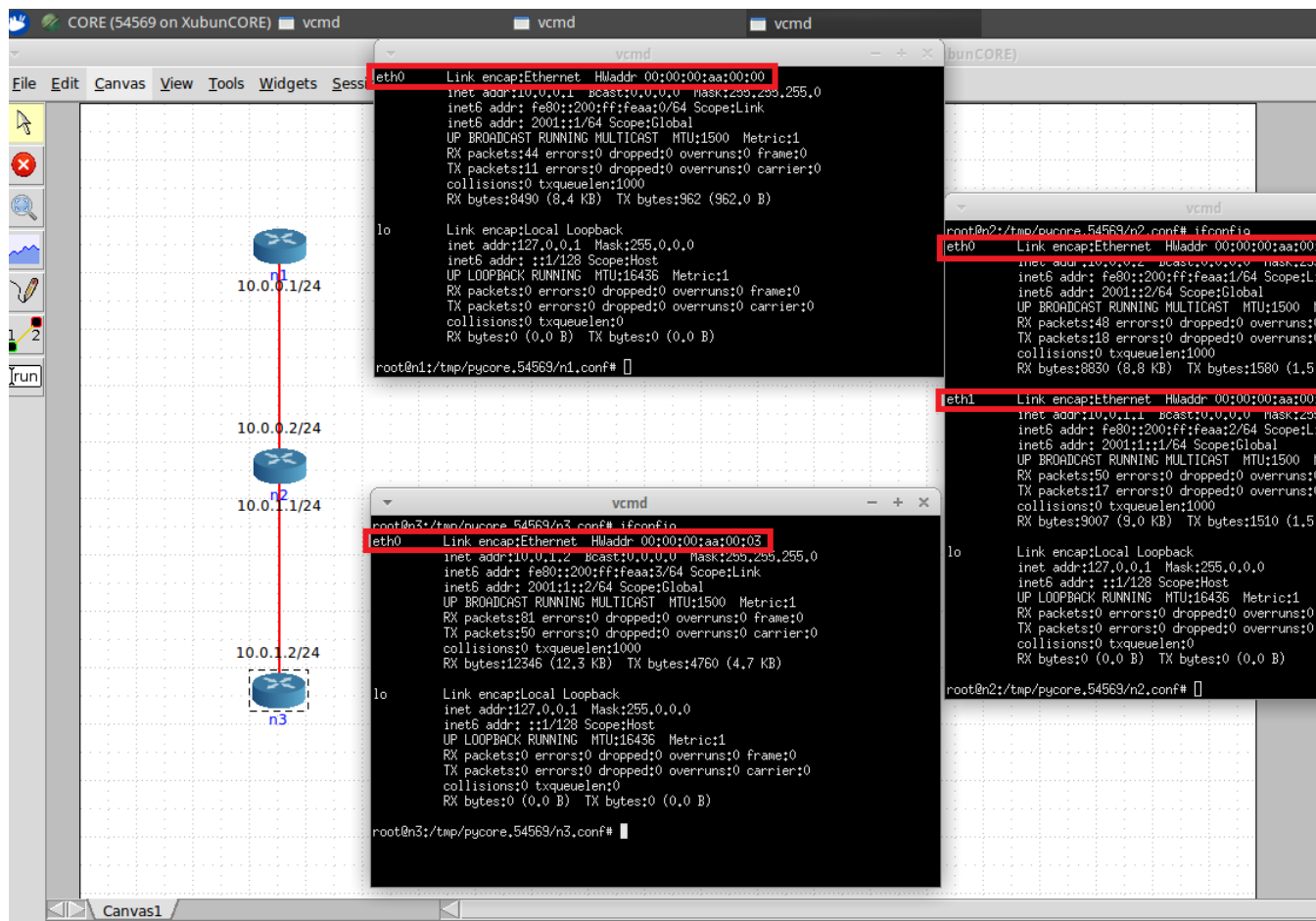


Fig. 15: Endereços de interfaces.

## 2.18 Exercício 17

### Questão

Usando o comando arp obtenha as caches arp dos diversos sistemas.

### Resposta

As caches ARP inicialmente estão vazias, como seria de esperar, pois ainda não houve qualquer tipo de comunicação.

### Realização

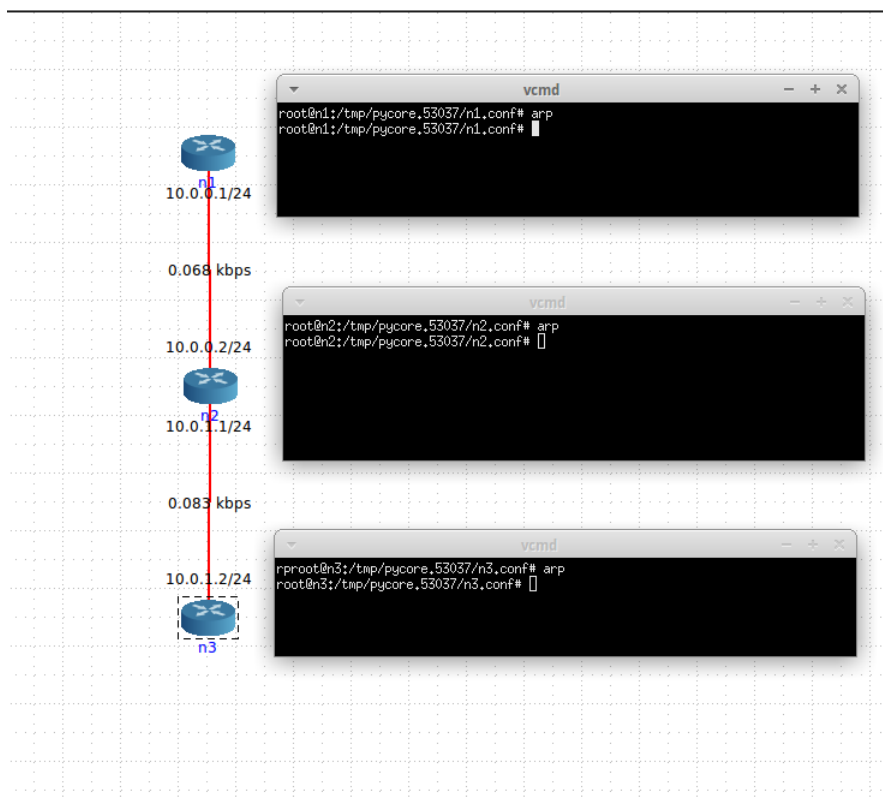


Fig. 16: Caches ARP vazias.

## 2.19 Exercício 18

### Questão

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?

### Resposta

Depois de feito o ping de **n1** para **n2**, a tabela ARP de **n1** passou a conhecer o MAC address relativo à interface eth0 do router **n2**. Do mesmo modo, o router **n2** passou a conhecer o router **n1** na interface eth0.

Depois de fazer ping de **n1** para **n3**, a tabela ARP de **n1** manteve-se, uma vez que o protocolo ARP só permite conhecer as máquinas adjacentes. No entanto, o router **n2**, como foi utilizado para encaminhamento, passou a conhecer a interface eth0 do router **n3** que, por sua vez passou a conhecer a interface eth1 do router **n2**.

### Realização

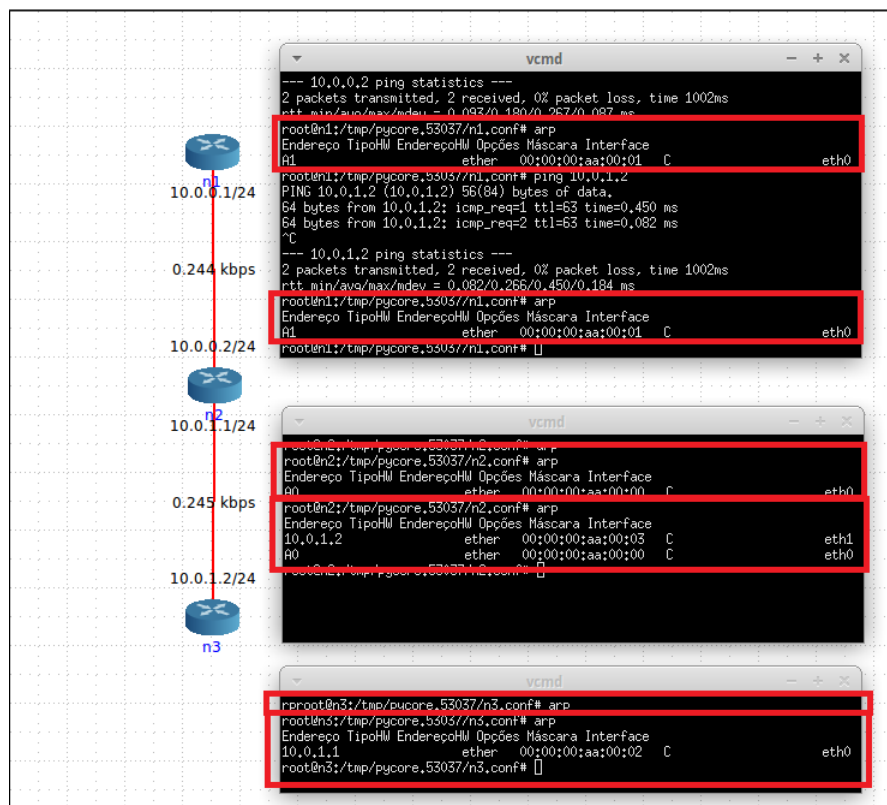


Fig. 17: Atualização de caches ARP.

## 2.20 Exercício 19

### Questão

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

### Resposta

**n1** conhecia, pelo ARP, o endereço IP da interface do router **n2**. Quando, na tabela ARP, removemos a entrada conhecida e adicionamos outra, com o mesmo IP mas com um endereço Ethernet inexistente, os pacotes são enviados para o endereço inexistente (no caso, utilizou-se 00:00:00:aa:00:04). Este envio é errado pois aquele endereço não existe, havendo perda total de pacotes. Este erro não é corrigido pois é suposto a tabela ARP estar correta.

### Realização

```
root@n1:/tmp/pycore.44730/n1.conf#  
root@n1:/tmp/pycore.44730/n1.conf# arp  
Endereço TipoHW EndereçoHW Opções Máscara Interface  
#1 ether 00:00:00:aa:00:01 C eth0  
root@n1:/tmp/pycore.44730/n1.conf# arp -d 10.0.0.2  
root@n1:/tmp/pycore.44730/n1.conf# arp  
Endereço TipoHW EndereçoHW Opções Máscara Interface  
#1 (incomplete) eth0  
root@n1:/tmp/pycore.44730/n1.conf# arp -s 10.0.0.2 00:00:00:aa:00:04  
root@n1:/tmp/pycore.44730/n1.conf# arp  
Endereço TipoHW EndereçoHW Opções Máscara Interface  
#1 ether 00:00:00:aa:00:04 CM eth0  
root@n1:/tmp/pycore.44730/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 ---  
4 packets transmitted, 0 received, 100% packet loss, time 3024ms  
root@n1:/tmp/pycore.44730/n1.conf#  
root@n1:/tmp/pycore.44730/n1.conf#
```

Fig. 18: Tabela ARP com entradas erradas.

## 2.21 Exercício 20

### Questão

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.

### Resposta

Como esperado, a tabela ARP de **n6** passou a conter o endereço Ethernet e IP do host 5. Como fazem parte da mesma sub-rede **10.0.2.x/24** e estão diretamente ligados, seria expectável que isto acontecesse. De facto, o *switch* não tem qualquer impacto nas tabelas ARP, uma vez que funciona apenas como redirecionador de pacotes, não tendo interferência àquele nível de rede.

### Realização

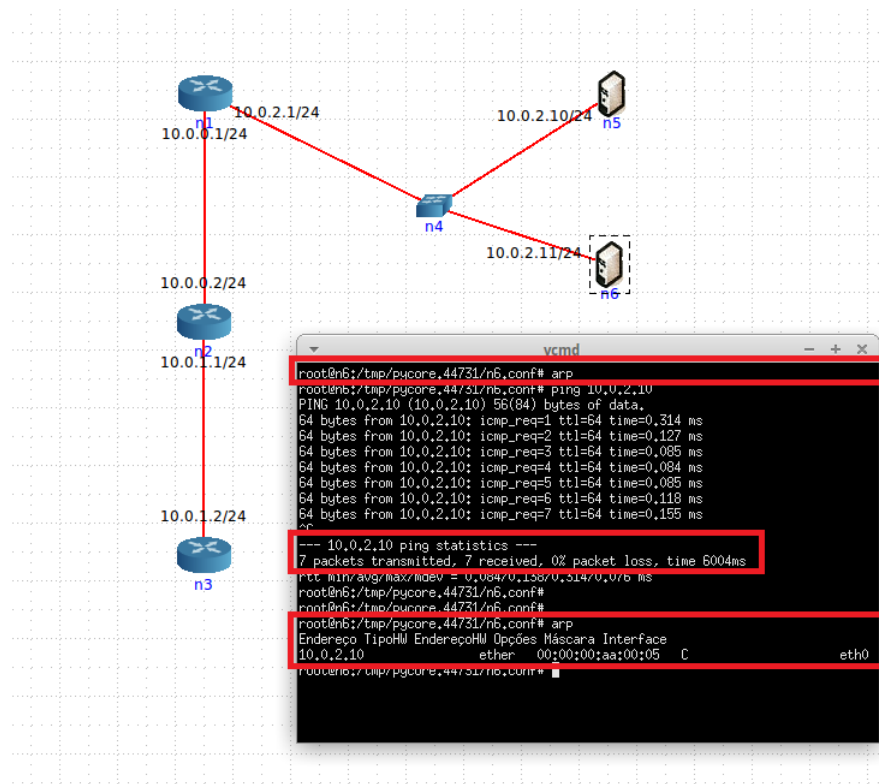


Fig. 19: Tabela ARP atualizada com endereço de host.

## Realização



## 3.2 Exercício 2

### Questão

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?

### Resposta

O pacote de pedido ARP Gratuito difere dos restantes em que o IP de envio é igual ao IP destino. Como o destino da trama Ethernet é ff:ff:ff:ff:ff:ff (endereço de broadcast), este irá enviar o pacote a todos os hosts da rede. Estes, por sua vez, ao receber o pacote - e identificando o pacote como ARP Gratuito - irão atualizar as respetivas tabelas ARP no MAC em questão com o novo IP.

### Realização

No.	Time	Source	Destination	Protocol	Length	Info
13	3.777434	AsustekC_08:3a:95	Broadcast	ARP	60	Who has 192.168.100.254? Tell 192.168.100.219
30	12.402798	LcfcHefe_66:65:4a	Broadcast	ARP	42	Who has 192.168.100.254? Tell 192.168.100.211
31	12.403659	Vmware_d2:19:f0	LcfcHefe_66:65:4a	ARP	60	192.168.100.254 is at 00:0c:29:d2:19:f0
38	12.458810	LcfcHefe_66:65:4a	Broadcast	ARP	42	Who has 192.168.100.254? Tell 192.168.100.211
41	12.459670	Vmware_d2:19:f0	LcfcHefe_66:65:4a	ARP	60	192.168.100.254 is at 00:0c:29:d2:19:f0
53	12.630741	LcfcHefe_66:65:4a	Broadcast	ARP	42	Who has 192.168.100.211? Tell 0.0.0.0
127	13.630832	LcfcHefe_66:65:4a	Broadcast	ARP	42	Who has 192.168.100.211? Tell 0.0.0.0
153	13.841777	AsustekC_08:3a:95	Broadcast	ARP	60	Who has 192.168.100.254? Tell 192.168.100.219
226	14.631248	LcfcHefe_66:65:4a	Broadcast	ARP	42	Who has 192.168.100.211? Tell 0.0.0.0
254	15.631251	LcfcHefe_66:65:4a	Broadcast	ARP	42	Gratuitous ARP for 192.168.100.211 (Request)
478	17.388852	Compalin_3a:23:c3	Broadcast	ARP	60	Who has 192.168.100.254? Tell 192.168.100.166
649	23.885174	AsustekC_08:3a:95	Broadcast	ARP	60	Who has 192.168.100.254? Tell 192.168.100.219
712	34.032549	AsustekC_08:3a:95	Broadcast	ARP	60	Who has 192.168.100.254? Tell 192.168.100.219
718	38.689541	Vmware_d2:19:f0	LcfcHefe_66:65:4a	ARP	60	Who has 192.168.100.211? Tell 192.168.100.254
719	38.689588	LcfcHefe_66:65:4a	Vmware_d2:19:f0	ARP	42	192.168.100.211 is at 68:f7:28:66:65:4a
735	44.229150	AsustekC_08:3a:95	Broadcast	ARP	60	Who has 192.168.100.254? Tell 192.168.100.219
771	54.365026	AsustekC_08:3a:95	Broadcast	ARP	60	Who has 192.168.100.254? Tell 192.168.100.219
784	64.394518	AsustekC_08:3a:95	Broadcast	ARP	60	Who has 192.168.100.254? Tell 192.168.100.219
812	74.442720	AsustekC_08:3a:95	Broadcast	ARP	60	Who has 192.168.100.254? Tell 192.168.100.219

> Frame 254: 42 bytes on wire (336 bits), 42 bytes captured (336 bits) on interface 0

> Ethernet II, Src: LcfcHefe\_66:65:4a (68:f7:28:66:65:4a), Dst: Broadcast (ff:ff:ff:ff:ff:ff)

▼ 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: LcfcHefe\_66:65:4a (68:f7:28:66:65:4a)

Sender IP address: 192.168.100.211

Target MAC address: 00:00:00\_00:00:00 (00:00:00:00:00:00)

Target IP address: 192.168.100.211

Fig. 22: O IP destino é igual ao IP fonte e o destino Ethernet é *Broadcast*.

## 4 Parte II - Domínios de colisão

### 4.1 Exercício 1

#### Questão

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

#### Resposta

A rede criada no CORE encontra-se ligada por um **repetidor**, que quando recebe tramas de um determinado *host* apenas reproduz essa informação para todas as redes a qual se encontram ligadas e como tal pode existir colisões entre diferentes tramas. Deste modo, quando é feito *ping de n1 para n4* todos os restantes *hosts* ficam à espera que as ligações fiquem livres para poderem voltar a enviar tramas. Efetivamente, ficam à escuta do que vai passando na rede. Consequentemente o *tcpdump* apresenta os pacotes que estão a ser enviados do n1 para o n4, além de que mais nenhuma trama circula na rede a não ser as trocadas entre o n1 e o n2.

Concluindo, os resultados apresentados verificam o modo como as colisões são evitadas, ou seja, é utilizado o protocolo CSMA/CD para resolver as colisões neste domínio de colisão em específico [2].

#### Realização

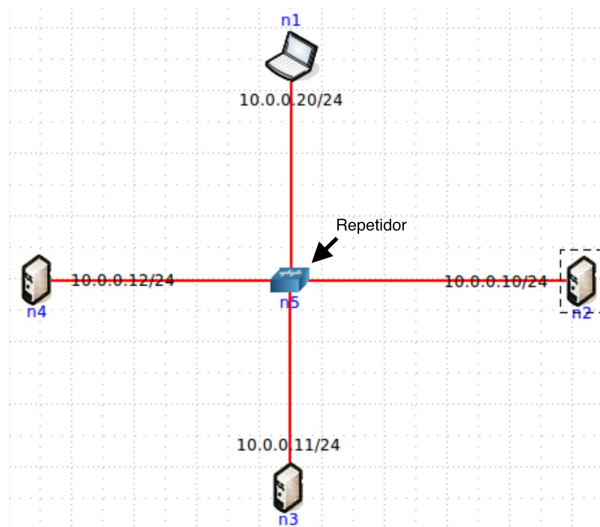


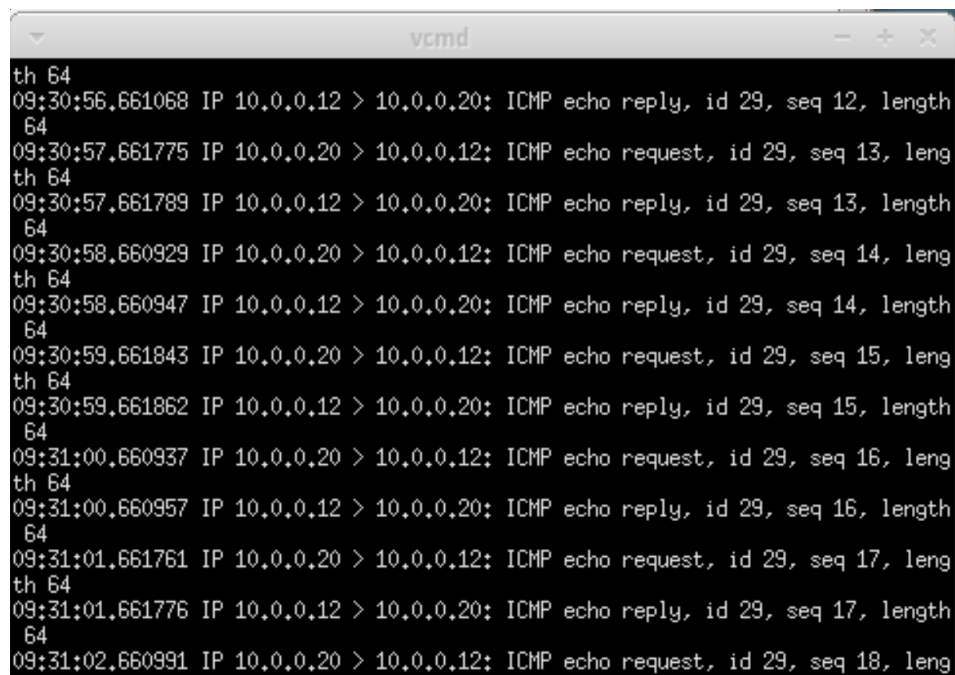
Fig. 23: Topologia de rede que utiliza um repetidor.

```
vcmd
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=0.040 ms
64 bytes from 10.0.0.12: icmp_req=2 ttl=64 time=0.035 ms
64 bytes from 10.0.0.12: icmp_req=3 ttl=64 time=0.036 ms
64 bytes from 10.0.0.12: icmp_req=4 ttl=64 time=0.040 ms
64 bytes from 10.0.0.12: icmp_req=5 ttl=64 time=0.047 ms
64 bytes from 10.0.0.12: icmp_req=6 ttl=64 time=0.042 ms
64 bytes from 10.0.0.12: icmp_req=7 ttl=64 time=0.046 ms
64 bytes from 10.0.0.12: icmp_req=8 ttl=64 time=0.042 ms
64 bytes from 10.0.0.12: icmp_req=9 ttl=64 time=0.042 ms
64 bytes from 10.0.0.12: icmp_req=10 ttl=64 time=0.050 ms
64 bytes from 10.0.0.12: icmp_req=11 ttl=64 time=0.042 ms
64 bytes from 10.0.0.12: icmp_req=12 ttl=64 time=0.094 ms
64 bytes from 10.0.0.12: icmp_req=13 ttl=64 time=0.038 ms
64 bytes from 10.0.0.12: icmp_req=14 ttl=64 time=0.042 ms
64 bytes from 10.0.0.12: icmp_req=15 ttl=64 time=0.046 ms
64 bytes from 10.0.0.12: icmp_req=16 ttl=64 time=0.048 ms
64 bytes from 10.0.0.12: icmp_req=17 ttl=64 time=0.040 ms
64 bytes from 10.0.0.12: icmp_req=18 ttl=64 time=0.094 ms
^C
--- 10.0.0.12 ping statistics ---
18 packets transmitted, 18 received, 0% packet loss, time 16996ms
rtt min/avg/max/mdev = 0.035/0.048/0.094/0.016 ms
root@n1:/tmp/pycore.58529/n1.conf#
```

Fig. 24: Ping realizado do *host* n1 para o *host* n4.

```
vcmd
09:30:55.660930 IP 10.0.0.20 > 10.0.0.12: ICMP echo request, id 29, seq 11, length 64
09:30:55.660947 IP 10.0.0.12 > 10.0.0.20: ICMP echo reply, id 29, seq 11, length 64
09:30:56.661030 IP 10.0.0.20 > 10.0.0.12: ICMP echo request, id 29, seq 12, length 64
09:30:56.661070 IP 10.0.0.12 > 10.0.0.20: ICMP echo reply, id 29, seq 12, length 64
09:30:57.661776 IP 10.0.0.20 > 10.0.0.12: ICMP echo request, id 29, seq 13, length 64
09:30:57.661790 IP 10.0.0.12 > 10.0.0.20: ICMP echo reply, id 29, seq 13, length 64
09:30:58.660930 IP 10.0.0.20 > 10.0.0.12: ICMP echo request, id 29, seq 14, length 64
09:30:58.660947 IP 10.0.0.12 > 10.0.0.20: ICMP echo reply, id 29, seq 14, length 64
09:30:59.661845 IP 10.0.0.20 > 10.0.0.12: ICMP echo request, id 29, seq 15, length 64
09:30:59.661863 IP 10.0.0.12 > 10.0.0.20: ICMP echo reply, id 29, seq 15, length 64
09:31:00.660939 IP 10.0.0.20 > 10.0.0.12: ICMP echo request, id 29, seq 16, length 64
09:31:00.660957 IP 10.0.0.12 > 10.0.0.20: ICMP echo reply, id 29, seq 16, length 64
```

Fig. 25: Tcpdump no *host* n2.

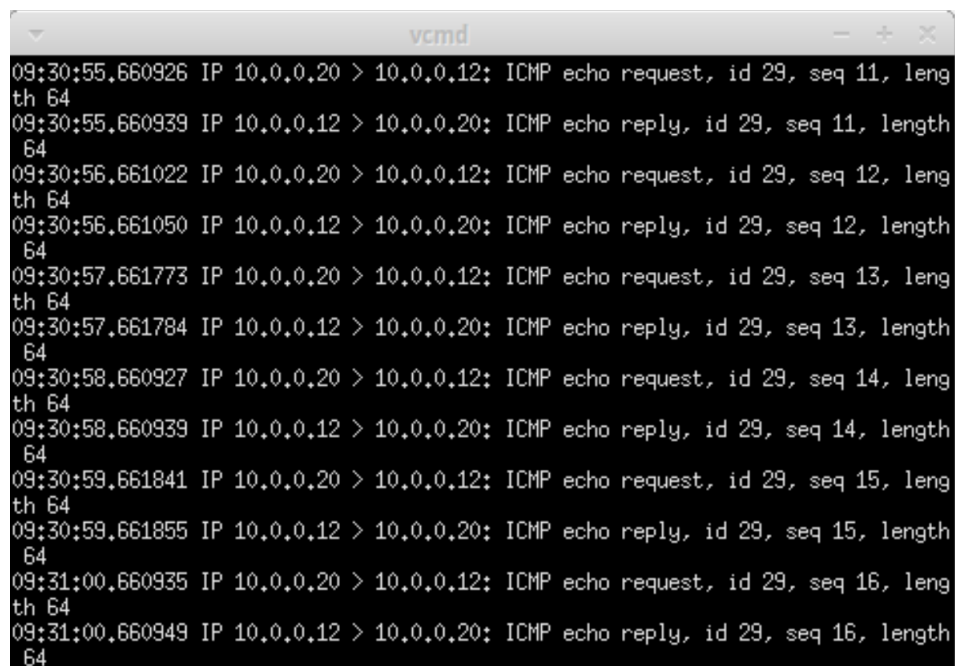


```

th 64
09:30:56.661068 IP 10.0.0.12 > 10.0.0.20: ICMP echo reply, id 29, seq 12, length
64
09:30:57.661775 IP 10.0.0.20 > 10.0.0.12: ICMP echo request, id 29, seq 13, leng
th 64
09:30:57.661789 IP 10.0.0.12 > 10.0.0.20: ICMP echo reply, id 29, seq 13, length
64
09:30:58.660929 IP 10.0.0.20 > 10.0.0.12: ICMP echo request, id 29, seq 14, leng
th 64
09:30:58.660947 IP 10.0.0.12 > 10.0.0.20: ICMP echo reply, id 29, seq 14, length
64
09:30:59.661843 IP 10.0.0.20 > 10.0.0.12: ICMP echo request, id 29, seq 15, leng
th 64
09:30:59.661862 IP 10.0.0.12 > 10.0.0.20: ICMP echo reply, id 29, seq 15, length
64
09:31:00.660937 IP 10.0.0.20 > 10.0.0.12: ICMP echo request, id 29, seq 16, leng
th 64
09:31:00.660957 IP 10.0.0.12 > 10.0.0.20: ICMP echo reply, id 29, seq 16, length
64
09:31:01.661761 IP 10.0.0.20 > 10.0.0.12: ICMP echo request, id 29, seq 17, leng
th 64
09:31:01.661776 IP 10.0.0.12 > 10.0.0.20: ICMP echo reply, id 29, seq 17, length
64
09:31:02.660991 IP 10.0.0.20 > 10.0.0.12: ICMP echo request, id 29, seq 18, leng

```

Fig. 26: Tcpdump no *host* n3.



```

09:30:55.660926 IP 10.0.0.20 > 10.0.0.12: ICMP echo request, id 29, seq 11, leng
th 64
09:30:55.660939 IP 10.0.0.12 > 10.0.0.20: ICMP echo reply, id 29, seq 11, length
64
09:30:56.661022 IP 10.0.0.20 > 10.0.0.12: ICMP echo request, id 29, seq 12, leng
th 64
09:30:56.661050 IP 10.0.0.12 > 10.0.0.20: ICMP echo reply, id 29, seq 12, length
64
09:30:57.661773 IP 10.0.0.20 > 10.0.0.12: ICMP echo request, id 29, seq 13, leng
th 64
09:30:57.661784 IP 10.0.0.12 > 10.0.0.20: ICMP echo reply, id 29, seq 13, length
64
09:30:58.660927 IP 10.0.0.20 > 10.0.0.12: ICMP echo request, id 29, seq 14, leng
th 64
09:30:58.660939 IP 10.0.0.12 > 10.0.0.20: ICMP echo reply, id 29, seq 14, length
64
09:30:59.661841 IP 10.0.0.20 > 10.0.0.12: ICMP echo request, id 29, seq 15, leng
th 64
09:30:59.661855 IP 10.0.0.12 > 10.0.0.20: ICMP echo reply, id 29, seq 15, length
64
09:31:00.660935 IP 10.0.0.20 > 10.0.0.12: ICMP echo request, id 29, seq 16, leng
th 64
09:31:00.660949 IP 10.0.0.12 > 10.0.0.20: ICMP echo reply, id 29, seq 16, length
64

```

Fig. 27: Tcpdump no *host* n4.

## 4.2 Exercício 2

### Questão

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.

### Resposta

Substituindo o repetidor pelo *switch* as colisões são evitadas. Essa diferença é resultado da metodologia na qual o *switch* aceta. Efetivamente, este aprende a rede na qual se encontra inserido e constrói uma tabela com os endereços MAC correspondendo-os às respectivas portas a que os sistemas se encontram ligados. Assim quando o *host* n1 realiza *ping* no *host* n4, o *switch* reencaminha o tráfego diretamente na porta na qual o n4 está ligado. Desta forma colisões entre diferentes segmentos deixam de ser possíveis visto estarem efetivamente separados. Como resultado, o *tcpdump* que se encontra a correr no *host* n2 e n3, não apresenta qualquer tráfego resultante do *ping* entre o *host* n1 e n4, pois o tráfego encontra-se isolado destes sistemas [3].

### Realização

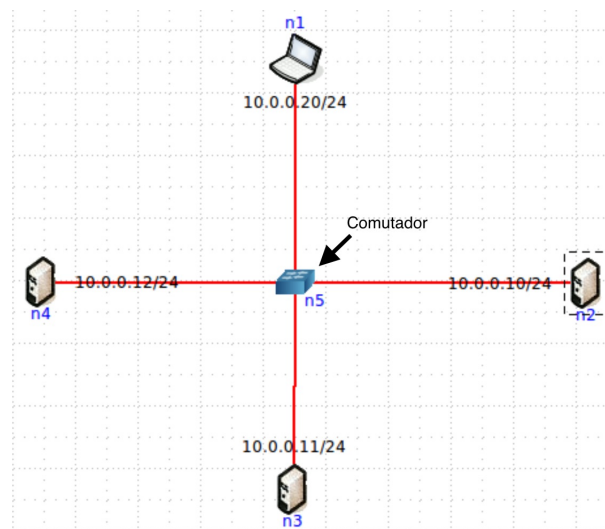


Fig. 28: Topologia de rede que utiliza um comutador.

```
vcmd
root@n1:/tmp/pycore.58530/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=0.058 ms
64 bytes from 10.0.0.12: icmp_req=2 ttl=64 time=0.033 ms
64 bytes from 10.0.0.12: icmp_req=3 ttl=64 time=0.030 ms
64 bytes from 10.0.0.12: icmp_req=4 ttl=64 time=0.055 ms
64 bytes from 10.0.0.12: icmp_req=5 ttl=64 time=0.071 ms
64 bytes from 10.0.0.12: icmp_req=6 ttl=64 time=0.033 ms
64 bytes from 10.0.0.12: icmp_req=7 ttl=64 time=0.032 ms
64 bytes from 10.0.0.12: icmp_req=8 ttl=64 time=0.046 ms
64 bytes from 10.0.0.12: icmp_req=9 ttl=64 time=0.030 ms
64 bytes from 10.0.0.12: icmp_req=10 ttl=64 time=0.035 ms
64 bytes from 10.0.0.12: icmp_req=11 ttl=64 time=0.030 ms
64 bytes from 10.0.0.12: icmp_req=12 ttl=64 time=0.032 ms
^C
--- 10.0.0.12 ping statistics ---
12 packets transmitted, 12 received, 0% packet loss, time 10999ms
rtt min/avg/max/mdev = 0.030/0.040/0.071/0.014 ms
root@n1:/tmp/pycore.58530/n1.conf#
```

Fig. 29: Ping realizado do *host* n1 para o *host* n4.

```
vcmd
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), capture size 262144 bytes
09:48:25.337616 IP6 fe80::f0fb:63ff:fe2d:648a.5353 > ff02::fb.5353: 0 [6q] PTR (
QM)? _afpovertcp._tcp.local. PTR (QM)? _ftp._tcp.local. PTR (QM)? _webdav._tcp.l
ocal. PTR (QM)? _webdavs._tcp.local. PTR (QM)? _sftp-ssh._tcp.local. PTR (QM)? _
smb._tcp.local. (107)
09:48:26.007125 IP6 fe80::6ca0:63ff:fed6:772d.5353 > ff02::fb.5353: 0 [6q] PTR (
QM)? _afpovertcp._tcp.local. PTR (QM)? _ftp._tcp.local. PTR (QM)? _webdav._tcp.l
ocal. PTR (QM)? _webdavs._tcp.local. PTR (QM)? _sftp-ssh._tcp.local. PTR (QM)? _
smb._tcp.local. (107)
09:48:40.045340 ARP, Request who-has 10.0.0.12 tell 10.0.0.20, length 28

```

Fig. 30: Tcpdump no *host* n2.

```
vcmd
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), capture size 262144 bytes
09:48:25.337615 IP6 fe80::f0fb:63ff:fe2d:648a.5353 > ff02::fb.5353: 0 [6q] PTR (
QM)? _afpovertcp._tcp.local. PTR (QM)? _ftp._tcp.local. PTR (QM)? _webdav._tcp.l
ocal. PTR (QM)? _webdavs._tcp.local. PTR (QM)? _sftp-ssh._tcp.local. PTR (QM)? _s
mb._tcp.local. (107)
09:48:25.912749 IP6 fe80::8b4:3eff:fe8e:94e3.5353 > ff02::fb.5353: 0 [6q] PTR (Q
M)? _afpovertcp._tcp.local. PTR (QM)? _ftp._tcp.local. PTR (QM)? _webdav._tcp.lo
cal. PTR (QM)? _webdavs._tcp.local. PTR (QM)? _sftp-ssh._tcp.local. PTR (QM)? _s
mb._tcp.local. (107)
09:48:40.045339 ARP, Request who-has 10.0.0.12 tell 10.0.0.20, length 28
█
```

Fig. 31: Tcpdump no *host* n3.

```
vcmd
h 64
09:48:46.044941 IP 10.0.0.12 > 10.0.0.20: ICMP echo reply, id 28, seq 7, length
64
09:48:47.044962 IP 10.0.0.20 > 10.0.0.12: ICMP echo request, id 28, seq 8, lengt
h 64
09:48:47.044975 IP 10.0.0.12 > 10.0.0.20: ICMP echo reply, id 28, seq 8, length
64
09:48:48.045743 IP 10.0.0.20 > 10.0.0.12: ICMP echo request, id 28, seq 9, lengt
h 64
09:48:48.045751 IP 10.0.0.12 > 10.0.0.20: ICMP echo reply, id 28, seq 9, length
64
09:48:49.044900 IP 10.0.0.20 > 10.0.0.12: ICMP echo request, id 28, seq 10, leng
th 64
09:48:49.044911 IP 10.0.0.12 > 10.0.0.20: ICMP echo reply, id 28, seq 10, length
64
09:48:50.045761 IP 10.0.0.20 > 10.0.0.12: ICMP echo request, id 28, seq 11, leng
th 64
09:48:50.045769 IP 10.0.0.12 > 10.0.0.20: ICMP echo reply, id 28, seq 11, length
64
09:48:51.044926 IP 10.0.0.20 > 10.0.0.12: ICMP echo request, id 28, seq 12, leng
th 64
09:48:51.044936 IP 10.0.0.12 > 10.0.0.20: ICMP echo reply, id 28, seq 12, length
64
█
```

Fig. 32: Tcpdump no *host* n4.

## 5 Conclusões

Neste trabalho...

## References

1. Plummer, D.C.: RFC 826: Ethernet Address Resolution Protocol: Or converting network protocol addresses to 48.bit Ethernet address for transmission on Ethernet hardware (1982)
2. Wikipedia: Csmacd. <https://pt.wikipedia.org/wiki/CSMA/CD> (2017) [Online; acedido a 1-Dezembro-2017].
3. Wikipedia: Comutador (redes). [https://pt.wikipedia.org/wiki/Comutador\\_\(redes\)](https://pt.wikipedia.org/wiki/Comutador_(redes)) (2017) [Online; acedido a 1-Dezembro-2017].