

# **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

```
C:\Users\Daniel Maia\Documents\Ghidra\computer-networks\paulinho\parte1\capture_de_tramas.pcapng 501 total packets, 52 shown

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 (800800)
Internet Protocol Version 4, Src: 192.168.100.209, Dst: 193.136.19.20
Transmission Control Protocol, Src Port: 52138, 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

```
C:\Users\Daniel Maia\Documents\GitHub\computer-networks\pabaiho\garter\capture_de_tramas.pcapng 501 total packets, 52 shown

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: lcf09ef6:46:50:4a (08:f7:28:66:05:4a), Dst: Vmware_d2:19:f0 (08:0c:29:d2:19:f0)
Destination: Vmware_d2:19:f0 (08:0c:29:d2:19:f0)
Source: lcf09ef6:46:50:4a (08:f7:28:66:05:4a)
Type: IPv4 (0x0800)
Internet Protocol Version 4, Src: 192.168.100.209, Dst: 193.136.19.20
Transmission Control Protocol, Src Port: 52138, 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

## **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**

### **Realização**

## **2.6 Exercício 6**

### **Questão**

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

### **Resposta**

### **Realização**

## **2.7 Exercício 7**

### **Questão**

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

### **Resposta**

### **Realização**



## **2.8 Exercício 8**

### **Questão**

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

### **Resposta**

### **Realização**

## 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. 3.** 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

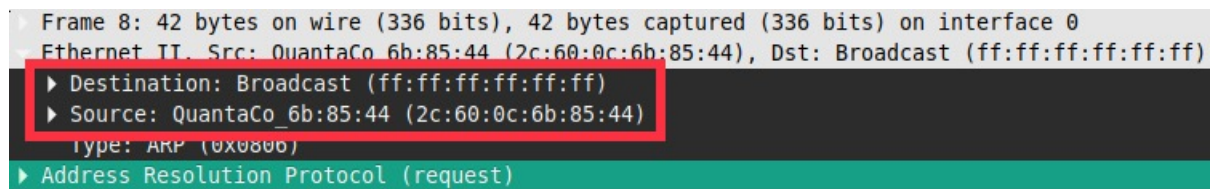
Endereço destino: ff:ff:ff:ff:ff:ff (broadcast)

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

A screenshot of a Wireshark packet capture. The top line shows 'Frame 8: 42 bytes on wire (336 bits), 42 bytes captured (336 bits) on interface 0'. The next line is 'Ethernet II, Src: QuantaCo\_6b:85:44 (2c:60:0c:6b:85:44), Dst: Broadcast (ff:ff:ff:ff:ff:ff)'. Below this, a red rectangle highlights two lines: 'Destination: Broadcast (ff:ff:ff:ff:ff:ff)' and 'Source: QuantaCo\_6b:85:44 (2c:60:0c:6b:85:44)'. The next line is 'Type: ARP (0x0806)'. The bottom line is 'Address Resolution Protocol (request)'.

**Fig. 4.** Pedido ARP enviado em *broadcast*.

## 2.11 Exercício 11

### Questão

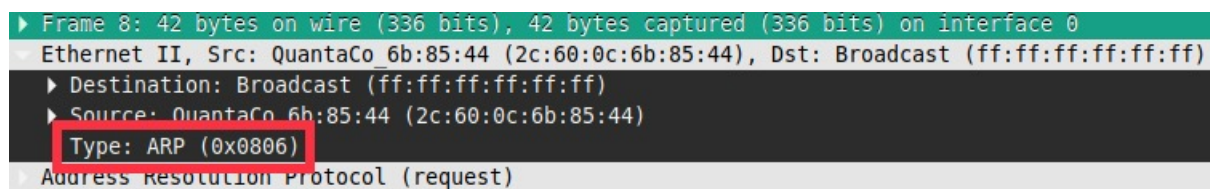
Qual o valor hexadecimal do campo tipo da trama Ethernet? O que indica?

### 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. 5.** 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. 6.** 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. 7.** 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. 8.** Campo *opcode* do ARP *reply*.



## 2.16 Exercício 15.b)

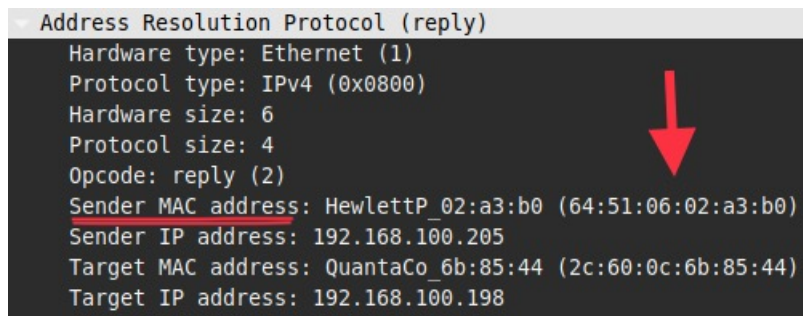
### Questão

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. 9.** 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**

### **Realização**

## **2.18 Exercício 17**

### **Questão**

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

### **Resposta**

### **Realização**

## **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**

### **Realização**

## **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**

### **Realização**

## **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**

### **Realização**

### **3 Parte II - ARP Gratuito**

#### **3.1 Exercício 1**

##### **Questão**

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

##### **Resposta**

##### **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**

#### **Realização**



## 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

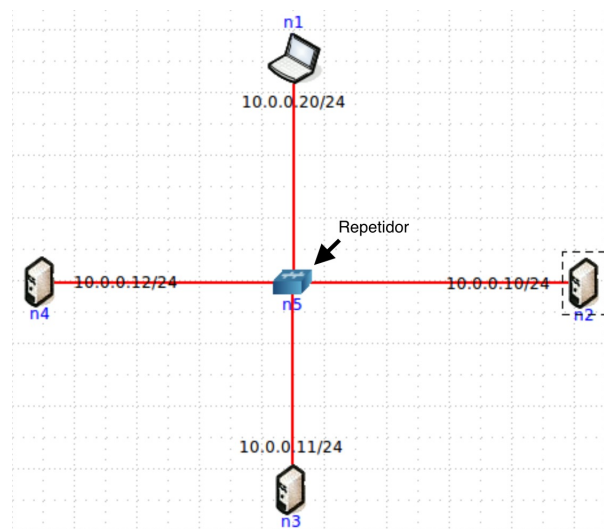


Fig. 10. Topologia de rede que utiliza um repetidor.

```
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. 11. 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. 12. 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. 13.** Tcpdump no *host* n3.

```

th 64
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. 14.** Tcpdump no *host* n4.

## 4.2 Exercício 2

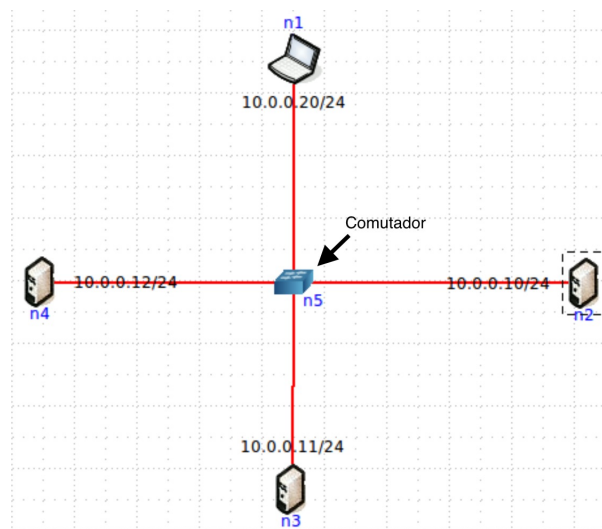
### Questão

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* atua. 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. 15.** 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. 16. 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. 17. 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)? _
smb._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. 18. 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. 19. 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].