

Universidade do Minho
Mestrado Integrado em Engenharia Informática

Redes de Computadores

TP3: Camada de Ligação Lógica

Henrique Paz (a84372), Joana Afonso Gomes (A84912), João Neves (a81366)

November 2019

Parte I

3. Captura e análise de tramas Ethernet

No.	Time	Source	Destination	Protocol	Length	Info
179	8.168809	192.168.100.156	193.136.19.40	HTTP	568	GET / HTTP/1.1
181	8.170292	193.136.19.40	192.168.100.156	HTTP	535	HTTP/1.1 301 Moved Permanently (text/html)

Figura 1: Captura de mensagens HTTP de Tramas *Ethernet*

> Frame 179: 568 bytes on wire (4544 bits), 568 bytes captured (4544 bits) on interface 0	
v Ethernet II, Src: AsustekC_35:e0:34 (2c:4d:54:35:e0:34), Dst: Vmware_d2:19:f0 (00:0c:29:d2:19:f0)	
> Destination: Vmware_d2:19:f0 (00:0c:29:d2:19:f0)	
> Source: AsustekC_35:e0:34 (2c:4d:54:35:e0:34)	
Type: IPv4 (0x0800)	
> Internet Protocol Version 4, Src: 192.168.100.156, Dst: 193.136.19.40	
v Transmission Control Protocol, Src Port: 50896, Dst Port: 80, Seq: 1, Ack: 1, Len: 514	
Source Port: 50896	
Destination Port: 80	
[Stream index: 11]	
[TCP Segment Len: 514]	
Source number: 1 (relative sequence number)	
0030	10 0a fc 11 00 00 47 45 54 20 2f 20 48 54 54 50GET / HTTP
0040	2f 31 2e 31 0d 0a 48 6f 73 74 3a 20 6d 69 65 69 /1.1-Host: miei
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 onnection: keep-
0070	61 6c 69 76 65 0d 0a 55 70 67 72 61 64 65 2d 49 alive-U pgrade-I
0080	6e 73 65 63 75 72 65 2d 52 65 71 75 65 73 74 73 nsecure- Requests
0090	3a 20 31 0d 0a 55 73 65 72 2d 41 67 65 6e 74 3a : 1-Use r-Agent:

Figura 2: Trama *Ethernet* com a mensagem HTTP GET

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

Origem: 2c : 4d : 54 : 35 : e0 : 34

Destino: 00 : 0c : 29 : d2 : 19 : f0

- (2) Identifique a que sistemas se referem. Justifique.

- **Endereço de origem (2c : 4d : 54 : 35 : e0 : 34)** : endereço MAC da máquina nativa.
- **Endereço de destino (00 : 0c : 29 : d2 : 19 : f0)**: endereço MAC da rede a que estamos ligados.

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

v Ethernet II, Src: AsustekC_35:e0:34 (2c:4d:54:35:e0:34), Dst: Vmware_d2:19:f0 (00:0c:29:d2:19:f0)	
> Destination: Vmware_d2:19:f0 (00:0c:29:d2:19:f0)	
> Source: AsustekC_35:e0:34 (2c:4d:54:35:e0:34)	
Type: IPv4 (0x0800)	

Figura 3: *Header* da trama *Ethernet*

O valor do campo *Type* da trama *Ethernet* é **0x0800**.

Este campo diz que tipo de dados a são transportados. Isto significa portanto que há um pacote 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. O comprimento da trama são 568 bytes.

Em percentagem, a sobrecarga é aproximadamente 9.5% (54 / 568).

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

O campo FCS não nos aparece na trama capturada, o que nos leva a concluir que não está a ser usado.

Este campo é usado controlo de erros e, como não existe uma significativa probabilidade de ocorrer erros, não se justifica a sua utilização.

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

```
▼ Ethernet II, Src: AsustekC_35:e0:34 (2c:4d:54:35:e0:34), Dst: Vmware_d2:19:f0 (00:0c:29:d2:19:f0)
  > Destination: Vmware_d2:19:f0 (00:0c:29:d2:19:f0)
  > Source: AsustekC_35:e0:34 (2c:4d:54:35:e0:34)
```

Figura 4: Endereço *Source* da *Ethernet*

Como vemos na figura acima, o endereço Ethernet da fonte é **2c:4d:54:35:e0:34**. Este corresponde ao *router* da rede a que estamos ligados.

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

```
▼ Ethernet II, Src: AsustekC_35:e0:34 (2c:4d:54:35:e0:34), Dst: Vmware_d2:19:f0 (00:0c:29:d2:19:f0)
  > Destination: Vmware_d2:19:f0 (00:0c:29:d2:19:f0)
  > Source: AsustekC_35:e0:34 (2c:4d:54:35:e0:34)
```

Figura 4: Endereço MAC *Destination* da *Ethernet*

O endereço MAC do destino é **00:0c:29:d2:19:f0**. Este corresponde à interface ativa do nosso computador.

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

```
▼ Ethernet II, Src: AsustekC_35:e0:34 (2c:4d:54:35:e0:34), Dst: Vmware_d2:19:f0 (00:0c:29:d2:19:f0)
  > Destination: Vmware_d2:19:f0 (00:0c:29:d2:19:f0)
  > Source: AsustekC_35:e0:34 (2c:4d:54:35:e0:34)
```

Figura 5: Protocolos na trama recebida.

Como podemos observar na Figura 5, os protocolos contidos na trama

recebida são **Ethernet II**, **IPv4** (Internet Protocol Version 4), **HTTP** (Hypertext Transfer Protocol), **TCP** (Transmission Control Protocol).

4. Protocolo ARP

- (9) Observe o conteúdo da tabela ARP. Explique o significado de cada uma das colunas.

```
C:\>arp -a

Interface: 192.168.100.207 --- 0x10
Internet Address      Physical Address      Type
192.168.100.187       fc-45-96-a6-0e-1d     dynamic
192.168.100.202       88-d7-f6-db-80-db     dynamic
192.168.100.254       00-0c-29-d2-19-f0     dynamic
192.168.100.255       ff-ff-ff-ff-ff-ff     static
224.0.0.22            01-00-5e-00-00-16     static
224.0.0.251           01-00-5e-00-00-fb     static
224.0.0.252           01-00-5e-00-00-fc     static
239.255.255.250       01-00-5e-7f-ff-fa     static
255.255.255.255       ff-ff-ff-ff-ff-ff     static
```

Figura 6: Tabela ARP.

```
> Ethernet II, Src: VMware_d2:19:f0 (00:0c:29:d2:19:f0), Dst: Broadcast (ff:ff:ff:ff:ff:ff)
v Address Resolution Protocol (request)
  Hardware type: Ethernet (1)
  Protocol type: IPv4 (0x0800)
  Hardware size: 6
  Protocol size: 4
  Opcode: request (1)
  Sender MAC address: VMware_d2:19:f0 (00:0c:29:d2:19:f0)
  Sender IP address: 192.168.100.254
  Target MAC address: 00:00:00_00:00:00 (00:00:00:00:00:00)
  Target IP address: 192.168.100.204
```

Figura 7: Campo ARP da trama capturada.

A primeira coluna representa o endereço. A segunda coluna o endereço MAC. A terceira o tipo de endereçamento usado.

```

C:\WINDOWS\system32>ping 192.168.100.202

Pinging 192.168.100.202 with 32 bytes of data:
Reply from 192.168.100.202: bytes=32 time<1ms TTL=128
Reply from 192.168.100.202: bytes=32 time<1ms TTL=128
Reply from 192.168.100.202: bytes=32 time=1ms TTL=128
Reply from 192.168.100.202: bytes=32 time=1ms TTL=128

Ping statistics for 192.168.100.202:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 0ms, Maximum = 1ms, Average = 0ms

```

Figura 8: Ping.

- (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?

```

> Ethernet II, Src: Vmware_d2:19:f0 (00:0c:29:d2:19:f0), Dst: Broadcast (ff:ff:ff:ff:ff:ff)
  Address Resolution Protocol (request)
    Hardware type: Ethernet (1)
    Protocol type: IPv4 (0x0800)
    Hardware size: 6
    Protocol size: 4
    Opcode: request (1)
    Sender MAC address: Vmware_d2:19:f0 (00:0c:29:d2:19:f0)
    Sender IP address: 192.168.100.254
    Target MAC address: 00:00:00_00:00:00 (00:00:00:00:00:00)
    Target IP address: 192.168.100.204

```

Figura 9: Trama Ethernet com o pedido ARP.

Como podemos ver na figura acima, o endereço de origem da trama Ethernet é 00:0c:29:d2:19:f0 e o de destino ff:ff:ff:ff:ff:ff.

Este endereço de destino é usado para ser identificado por todas as máquinas ligadas à rede, para assim todas conseguirem receber o pedido ARP. No entanto só a máquina que tiver o endereço que nós pedimos, no nosso caso, 192.168.100.202, é que responde o seu endereço MAC.

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

```

> Frame 5: 60 bytes on wire (480 bits), 60 bytes captured (480 bits) on interface 0
✓ Ethernet II, Src: Vmware_d2:19:f0 (00:0c:29:d2:19:f0), Dst: Broadcast (ff:ff:ff:ff:ff:ff)
  > Destination: Broadcast (ff:ff:ff:ff:ff:ff)
  > Source: Vmware_d2:19:f0 (00:0c:29:d2:19:f0)
    Type: ARP (0x0806)
    Padding: 0000000000000000000000000000000000000000000000000000000000000000
  > Address Resolution Protocol (request)

```

Figura 10: Campo *type* da trama Ethernet.

O valor hexadecimal do campo tipo da trama Ethernet é 0x0806.

- (12) Qual o valor do campo ARP *opcode*? O que especifica? Se necessário, consulte a RFC do protocolo ARP.

```

> Ethernet II, Src: Vmware_d2:19:f0 (00:0c:29:d2:19:f0), Dst: Broadcast (ff:ff:ff:ff:ff:ff)
✓ Address Resolution Protocol (request)
  Hardware type: Ethernet (1)
  Protocol type: IPv4 (0x0800)
  Hardware size: 6
  Protocol size: 4
  Opcode: request (1)
  Sender MAC address: Vmware_d2:19:f0 (00:0c:29:d2:19:f0)
  Sender IP address: 192.168.100.254
  Target MAC address: 00:00:00_00:00:00 (00:00:00:00:00:00)
  Target IP address: 192.168.100.220

```

Figura 11: Campo ARP *opcode*.

Como podemos ver na figura 11, o valor do campo ARP é 1 (0x0001). Este campo indica se é um pedido ou uma resposta. Como é 1, é um pedido (se fosse 2, era uma resposta).

- (13) Identifique que tipo de endereços está contido na mensagem ARP? Que conclui?

Como podemos ver na figura 11, a mensagem ARP contém os endereços de tipo IP e MAC, com origem e destino. Constatamos que não é possível saber o endereço MAC de destino, dado que ainda é o *request* (esse endereço é o que nós procuramos obter).

- (14) Explícite que tipo de pedido ou pergunta é feito pelo *host* de origem?

```

108 12.374138 AsustekC_35:e0:34 AsustekC_db:80:db ARP 42 Who has 192.168.100.202? Tell 192.168.100.207

```

Figura 12.

O *host* de origem pergunta "Who has 192.168.100.202?", endereço para o qual fizemos *ping* e diz qual é o seu endereço IP. A ideia é que a máquina com esse endereço responda com o seu endereço MAC.

- (15) Localize a mensagem ARP que é a resposta ao pedido ARP

efectuado.

- (a) Qual o valor do campo ARP opcode? O que especifica?

```

v Address Resolution Protocol (reply)
  Hardware type: Ethernet (1)
  Protocol type: IPv4 (0x0800)
  Hardware size: 6
  Protocol size: 4
  Opcode: reply (2)
  Sender MAC address: AsustekC_db:80:db (88:d7:f6:db:80:db)
  Sender IP address: 192.168.100.202
  Target MAC address: AsustekC_35:e0:34 (2c:4d:54:35:e0:34)
  Target IP address: 192.168.100.207

```

Figura 13: Trama ARP.

O valor do campo ARP opcode é 2 (0x0002). Especifica que se trata de uma resposta ao pedido ARP.

- (b) Em que posição da mensagem ARP está a resposta ao pedido ARP?

```

v Address Resolution Protocol (reply)
  Hardware type: Ethernet (1)
  Protocol type: IPv4 (0x0800)
  Hardware size: 6
  Protocol size: 4
  Opcode: reply (2)
  Sender MAC address: AsustekC_db:80:db (88:d7:f6:db:80:db)
  Sender IP address: 192.168.100.202
  Target MAC address: AsustekC_35:e0:34 (2c:4d:54:35:e0:34)
  Target IP address: 192.168.100.207

```

0000	2c 4d 54 35 e0 34 88 d7 f6 db 80 db 08 06 00 01	,MT5-4..
0010	08 00 06 04 00 02 88 d7 f6 db 80 db c0 a8 64 ca-d-
0020	2c 4d 54 35 e0 34 c0 a8 64 cf 00 00 00 00 00 00	,MT5-4- d-.....
0030	00 00 00 00 00 00 00 00 00 00 00 00

Figura 14.

A mensagem ARP, como podemos ver na Figura 14, encontra-se entre os bytes 23 e 28 .

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

270	9.093832	Vmware_d2:19:f0	Broadcast	ARP	60 Who has 192.168.100.218? Tell 192.168.100.254
273	9.233853	Vmware_d2:19:f0	Broadcast	ARP	60 Who has 192.168.100.178? Tell 192.168.100.254
318	9.540296	AsustekC_35:e0:34	Broadcast	ARP	42 Gratuitous ARP for 192.168.100.207 (Request)
365	10.234639	Vmware_d2:19:f0	Broadcast	ARP	60 Who has 192.168.100.178? Tell 192.168.100.254

Padding: 00000000000000000000000000000000

▼ Address Resolution Protocol (request)

Hardware type: Ethernet (1)
Protocol type: IPv4 (0x0800)
Hardware size: 6
Protocol size: 4
Opcode: request (1)
Sender MAC address: Vmware_d2:19:f0 (00:0c:29:d2:19:f0)
Sender IP address: 192.168.100.254
Target MAC address: 00:00:00_00:00:00 (00:00:00:00:00:00)
Target IP address: 192.168.100.178

Figura 15: Pedido ARP.

273	9.233853	Vmware_d2:19:f0	Broadcast	ARP	60 Who has 192.168.100.178? Tell 192.168.100.254
318	9.540296	AsustekC_35:e0:34	Broadcast	ARP	42 Gratuitous ARP for 192.168.100.207 (Request)
365	10.234639	Vmware_d2:19:f0	Broadcast	ARP	60 Who has 192.168.100.178? Tell 192.168.100.254

▼ 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: AsustekC_35:e0:34 (2c:4d:54:35:e0:34)
Sender IP address: 192.168.100.207
Target MAC address: 00:00:00_00:00:00 (00:00:00:00:00:00)
Target IP address: 192.168.100.207

Figura 16: Pedido ARP gratuito.

Como podemos ver na figura 16, o pedido ARP gratuito tem uma flag **"Is gratuitous = True"** . Para além disso, reparamos que o endereço de destino é igual ao de origem.

Um ARP Gratuito tem como objetivo detetar conflitos de endereços IP na rede local, sendo usado para determinar se existe algum equipamento na rede com um IP igual ao da máquina que envia a mensagem.

É, portanto, esperado que não haja resposta ao pedido ARP gratuito, pois isso significaria que existiria algum equipamento com o endereço IP igual ao da nossa máquina, e portanto, um conflito na rede.

5. Domínios de colisão

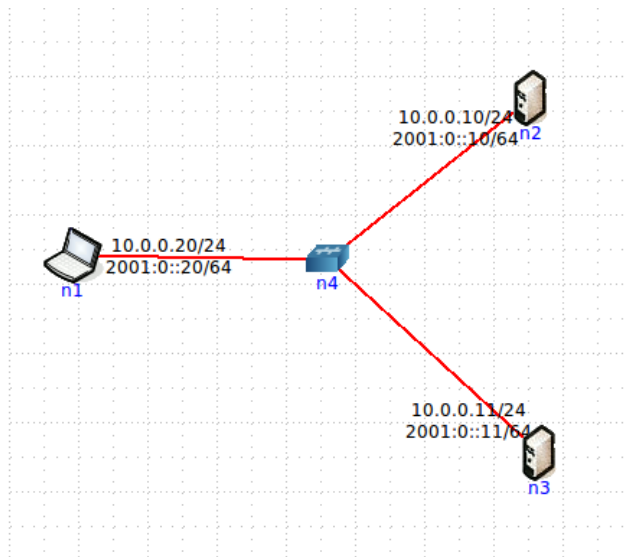


Figura 17: Topologia do CORE.

- (17) Faça *ping* de n1 para n2. Verifique com a opção *tcpdump* como flui o tráfego nas diversas interfaces dos vários dispositivos. Que conclui?

```
root@n1:/tmp/pycore.49658/n1.conf# ping 10.0.1.10
PING 10.0.1.10 (10.0.1.10) 56(84) bytes of data.
From 10.0.0.20 icmp_seq=1 Destination Host Unreachable
From 10.0.0.20 icmp_seq=2 Destination Host Unreachable
From 10.0.0.20 icmp_seq=3 Destination Host Unreachable
From 10.0.0.20 icmp_seq=4 Destination Host Unreachable
From 10.0.0.20 icmp_seq=5 Destination Host Unreachable
From 10.0.0.20 icmp_seq=6 Destination Host Unreachable
From 10.0.0.20 icmp_seq=7 Destination Host Unreachable
From 10.0.0.20 icmp_seq=8 Destination Host Unreachable
From 10.0.0.20 icmp_seq=9 Destination Host Unreachable
From 10.0.0.20 icmp_seq=10 Destination Host Unreachable
From 10.0.0.20 icmp_seq=11 Destination Host Unreachable
From 10.0.0.20 icmp_seq=12 Destination Host Unreachable
^C
--- 10.0.1.10 ping statistics ---
15 packets transmitted, 0 received, +12 errors, 100% packet loss, time 14012ms
pipe 4
root@n1:/tmp/pycore.49658/n1.conf#
```

Figura 18: Ping do *laptop* n1 para o *server* n2.

```

root@n2:/tmp/pycore.49658/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
^C17:24:05.305196 IP6 fe80::200:ff:feaa:2 > ip6-allrouters: ICMP6, router solici
tation, length 16
17:24:05.805333 ARP, Request who-has A0 tell 10.0.0.20, length 28
17:24:05.841395 IP6 fe80::a0b0:9ff:feb1:e2de > ff02::16: HBH ICMP6, multicast li
stener report v2, 1 group record(s), length 28
17:24:06.805238 ARP, Request who-has A0 tell 10.0.0.20, length 28
17:24:07.805857 ARP, Request who-has A0 tell 10.0.0.20, length 28
17:24:08.805801 ARP, Request who-has A0 tell 10.0.0.20, length 28
17:24:09.805857 ARP, Request who-has A0 tell 10.0.0.20, length 28
17:24:10.805552 ARP, Request who-has A0 tell 10.0.0.20, length 28
17:24:11.805578 ARP, Request who-has A0 tell 10.0.0.20, length 28
17:24:12.805437 ARP, Request who-has A0 tell 10.0.0.20, length 28
17:24:13.805361 ARP, Request who-has A0 tell 10.0.0.20, length 28

```

Figura 19: tcpdump no servidor n2.

```

root@n3:/tmp/pycore.49658/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
^C17:24:09.805853 ARP, Request who-has A0 tell 10.0.0.20, length 28
17:24:10.805549 ARP, Request who-has A0 tell 10.0.0.20, length 28
17:24:11.805574 ARP, Request who-has A0 tell 10.0.0.20, length 28
17:24:12.805433 ARP, Request who-has A0 tell 10.0.0.20, length 28
17:24:13.805357 ARP, Request who-has A0 tell 10.0.0.20, length 28

5 packets captured
5 packets received by filter
0 packets dropped by kernel
root@n3:/tmp/pycore.49658/n3.conf# █

```

Figura 20: tcpdump no servidor n3.

Assim como é visível nas figuras 19 e 20, onde usamos o comando `tcpdump`, os servidores recebem ambas mensagens ARP, apesar de a mensagem ser destinada apenas a n2.

Isto acontece porque, quando fazemos ping do *laptop* n1 para o servidor n2, a mensagem é transmitida para o *hub* e daí reencaminhado para n2 e n3, visto que o *hub* apenas repete aquilo que recebe e faz *broadcast* para todos os dispositivos na rede.

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

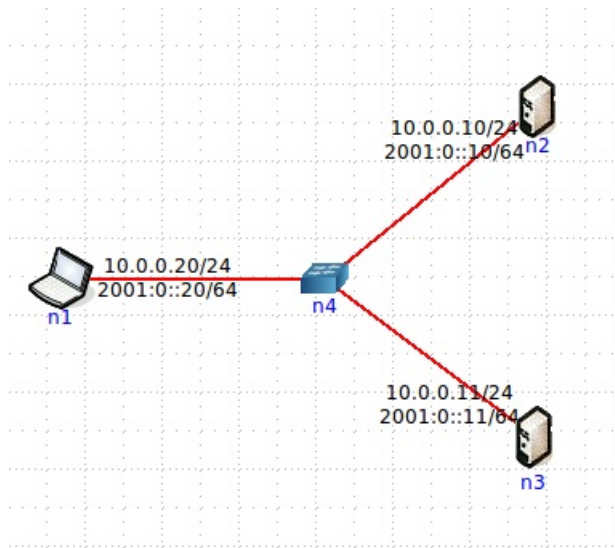


Figura 21: Topologia do CORE com o *switch*.

```

root@n1:/tmp/pycore.49674/n1.conf# ping 10.0.0.10
PING 10.0.0.10 (10.0.0.10) 56(84) bytes of data:
64 bytes from 10.0.0.10: icmp_req=1 ttl=64 time=0.044 ms
64 bytes from 10.0.0.10: icmp_req=2 ttl=64 time=0.137 ms
64 bytes from 10.0.0.10: icmp_req=3 ttl=64 time=0.061 ms
64 bytes from 10.0.0.10: icmp_req=4 ttl=64 time=0.064 ms
64 bytes from 10.0.0.10: icmp_req=5 ttl=64 time=0.063 ms
64 bytes from 10.0.0.10: icmp_req=6 ttl=64 time=0.090 ms
64 bytes from 10.0.0.10: icmp_req=7 ttl=64 time=0.029 ms
64 bytes from 10.0.0.10: icmp_req=8 ttl=64 time=0.074 ms
64 bytes from 10.0.0.10: icmp_req=9 ttl=64 time=0.074 ms
64 bytes from 10.0.0.10: icmp_req=10 ttl=64 time=0.072 ms
^C
--- 10.0.0.10 ping statistics ---
10 packets transmitted, 10 received, 0% packet loss, time 8997ms
rtt min/avg/max/mdev = 0.029/0.070/0.137/0.029 ms
root@n1:/tmp/pycore.49674/n1.conf# 
  
```

Figura 22: Ping do *laptop* n1 para o *server* n2.

```

root@n2:/tmp/pycore.49674/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
^C18:20:43.573172 IP 10.0.0.20 > A9: ICMP echo request, id 128, seq 6, length 64
18:20:43.573200 IP A9 > 10.0.0.20: ICMP echo reply, id 128, seq 6, length 64
18:20:43.585200 ARP, Request who-has 10.0.0.20 tell A9, length 28
18:20:43.585270 ARP, Reply 10.0.0.20 is-at 00:00:00:aa:00:00 (oui Ethernet), len
gth 28
18:20:44.573146 IP 10.0.0.20 > A9: ICMP echo request, id 128, seq 7, length 64
18:20:44.573155 IP A9 > 10.0.0.20: ICMP echo reply, id 128, seq 7, length 64
18:20:45.573800 IP 10.0.0.20 > A9: ICMP echo request, id 128, seq 8, length 64
18:20:45.573820 IP A9 > 10.0.0.20: ICMP echo reply, id 128, seq 8, length 64
18:20:46.573205 IP 10.0.0.20 > A9: ICMP echo request, id 128, seq 9, length 64
18:20:46.573227 IP A9 > 10.0.0.20: ICMP echo reply, id 128, seq 9, length 64
18:20:47.573185 IP 10.0.0.20 > A9: ICMP echo request, id 128, seq 10, length 64
18:20:47.573207 IP A9 > 10.0.0.20: ICMP echo reply, id 128, seq 10, length 64

12 packets captured
12 packets received by filter
0 packets dropped by kernel
root@n2:/tmp/pycore.49674/n2.conf# █

```

Figura 23: tcpdump no servidor n2..

```

root@n3:/tmp/pycore.49674/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
^C
0 packets captured
0 packets received by filter
0 packets dropped by kernel
root@n3:/tmp/pycore.49674/n3.conf# █

```

Figura 24: tcpdump no servidor n3.

Agora que o *hub* foi substituído por um *switch*, o que observamos é que, quando é feito o ping do laptop n1 para o server n2, os pacotes são todos entregues a ele, e o outro server não regista qualquer tráfego, como podemos ver na Figura 24. Isto significa que a mensagem enviada por n1 ao *switch* é enviada por este diretamente ao server pretendido, e não para todos os nodos da rede.

Concluimos deste modo que os switch são úteis para evitar as chamadas colisões (interferências mútuas que detioram as tramas que são enviadas), limitando o envio da mensagem ao destino que se pretende. Isto contrasta com o que acontece quando são usados hubs, em que as colisões são bastantes frequentes, tendo em vista que transmitem mensagens para todos os elementos da rede por um único canal de comunicação e levando, simultaneamente, a haver uma quantidade relevante de informação desnecessária em circulação na rede.

Conclusões

Achamos unanimemente que este trabalho contribui para nos conceder um maior entendimento no que toca, por um lado à análise de tramas Ethernet, ajudando a perceber o seu funcionamento. Por outro lado, fomos familiarizados com o protocolo ARP, essencial para percebermos o conceito de mapeamento entre endereços do nível de rede e endereços do nível de ligação lógica.

De notar também a utilização do CORE para entendermos o conceito de *hubs* e switches, o funcionamento característico dos mesmos e qual as vantagens em usar um comparativamente ao outro.