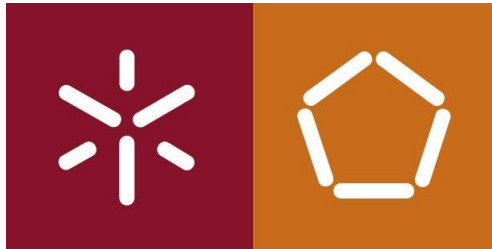


UNIVERSIDADE DO MINHO

MESTRADO INTEGRADO EM ENGENHARIA INFORMÁTICA



---

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

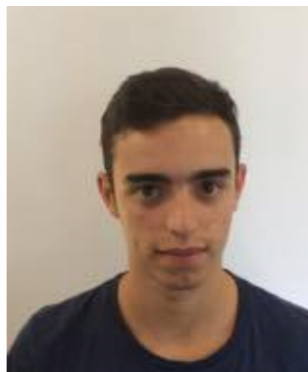
---

REDES DE COMPUTADORES

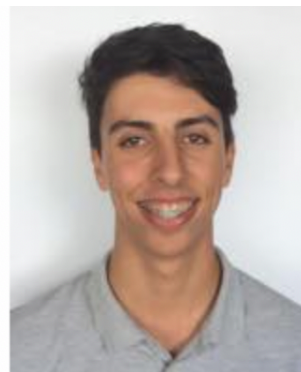
PL 4 GRUPO 8



Carla Cruz  
A80564



Diogo Sobral  
A82523



Pedro Freitas  
A80975

November 26, 2018

# Captura e análise de Tramas Ethernet

Assegure-se que a cache do seu browser está vazia e está conectado em rede através da interface Ethernet.

Ative o Wireshark na sua máquina nativa.

No seu browser, aceda ao URL `http://miei.di.uminho.pt`.

Pare a captura do Wireshark.

Obtenha o número de ordem da sequência de bytes capturada (coluna da esquerda da janela do Wireshark) correspondente à mensagem HTTP GET enviada pelo seu computador para o servidor Web, bem como o começo da respetiva mensagem HTTP Response proveniente do servidor.

No sentido de proceder à análise do tráfego, selecione a trama Ethernet que contém a mensagem HTTP GET. Recorde-se que a mensagem GET do HTTP está no interior de um segmento TCP que é transportado num datagrama IP que, por sua vez, está encapsulado no campo de dados de uma trama Ethernet. Expand a informação do nível da ligação de dados (Ethernet II) e observe o conteúdo da trama Ethernet (cabeçalho e dados (payload)).

Responda às perguntas seguintes com base no conteúdo da trama Ethernet que contém a mensagem HTTP GET.

Sempre que aplicável, deve incluir a impressão dos dados relativa ao pacote capturado (ou parte dele) necessária para fundamentar a resposta à questão colocada. Para imprimir um pacote, use File->Print, escolha Selected packet only e Packet summary line, ou use qualquer outro método que lhe pareça adequado para a captura desses dados. Selecione o mínimo detalhe necessário para responder à pergunta.

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

40	3.835741	192.168.2.171	193.136.19.40	TCP	66	50165 → 80	[ACK] Seq=1 Ack=1 Win=131712 Len=0 TSval=50148278 TSecr=222
41	3.835755	192.168.2.171	193.136.19.40	TCP	66	50166 → 80	[ACK] Seq=1 Ack=1 Win=131712 Len=0 TSval=50148278 TSecr=222
42	3.840377	192.168.2.171	193.136.19.40	HTTP	479	GET / HTTP/1.1	
43	3.841074	193.136.19.40	192.168.2.171	TCP	66	80 → 50165	[ACK] Seq=1 Ack=414 Win=30080 Len=0 TSval=2228060857 TSecr=
44	3.844601	193.136.19.40	192.168.2.171	TCP	1514	80 → 50165	[ACK] Seq=1 Ack=414 Win=30080 Len=1448 TSval=2228060860 TSecr=
45	3.844704	193.136.19.40	192.168.2.171	TCP	1514	80 → 50165	[ACK] Seq=1449 Ack=414 Win=30080 Len=1448 TSval=2228060860 TSecr=
46	3.844740	192.168.2.171	193.136.19.40	TCP	66	50165 → 80	[ACK] Seq=414 Ack=2897 Win=129600 Len=0 TSval=50148287 TSecr=
47	3.844827	193.136.19.40	192.168.2.171	TCP	1514	80 → 50165	[ACK] Seq=2897 Ack=414 Win=30080 Len=1448 TSval=2228060860 TSecr=
48	3.844951	193.136.19.40	192.168.2.171	TCP	1514	80 → 50165	[ACK] Seq=4345 Ack=414 Win=30080 Len=1448 TSval=2228060860 TSecr=
49	3.844979	192.168.2.171	193.136.19.40	TCP	66	50165 → 80	[ACK] Seq=414 Ack=5793 Win=126720 Len=0 TSval=50148287 TSecr=
50	3.845074	193.136.19.40	192.168.2.171	TCP	1514	80 → 50165	[ACK] Seq=5793 Ack=414 Win=30080 Len=1448 TSval=2228060860 TSecr=
51	3.845197	193.136.19.40	192.168.2.171	TCP	1514	80 → 50165	[ACK] Seq=7241 Ack=414 Win=30080 Len=1448 TSval=2228060860 TSecr=
52	3.845225	192.168.2.171	193.136.19.40	TCP	66	50165 → 80	[ACK] Seq=414 Ack=8689 Win=123776 Len=0 TSval=50148287 TSecr=

Figure 1: Pacote TCP enviados ao aceder ao site

```

▼ Ethernet II, Src: CelLink_17:bc:b8 (a0:ce:c8:17:bc:b8), Dst: Vmware_5e:69:ad (00:0c:29:5e:69:ad)
► Destination: Vmware_5e:69:ad (00:0c:29:5e:69:ad)
► Source: CelLink_17:bc:b8 (a0:ce:c8:17:bc:b8)
Type: IPv4 (0x0800)

```

Figure 2: Campo Ethernet II da trama selecionado

Endereço MAC destino -> 00:0c:29:5e:69:ad

Endereço MAC origem -> a0:ce:c8:17:bc:b8

2. Identifique a que sistemas se referem. Justifique.

O endereço mac refere-se ao endereço físico da interface ativa de uma máquina. Neste caso a origem refere-se ao endereço físico do nosso computador e o destino refere-se ao endereço físico do router com que se está a comunicar.

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

Como podemos ver na figura 2, o campo do Type tem o valor 0x0800 que significa que a camada superior está a utilizar o protocolo IPv4.

4. Quanto bytes são usados desde o início de trama de caractere ASCII “G” do método HTTP GET? Calcule e indique, em percentagem, a sobrecarga (overhead) introduzida pela pilha protocolar no envio do HTTP GET.

```

0000 00 0c 29 5e 69 ad a0 ce c8 17 bc b8 08 00 45 02  ..)^i... ..E.
0010 01 d1 00 00 40 00 40 06 a1 21 c0 a8 02 ab c1 88  ...@.@.!. ....
0020 13 28 c3 f5 00 50 e0 e0 84 1f 7f c0 c6 e9 80 18  .(...P... ..3...
0030 08 0a fe 88 00 00 01 01 08 0a 02 fd 33 bb 84 cd  ..GET / HTTP/1.1
0040 82 b3 47 45 54 20 2f 20 48 54 54 50 2f 31 2e 31  ..Host: miei.di.
0050 0d 0a 48 6f 73 74 3a 20 6d 69 65 69 2e 64 69 2e

```

Figure 3: Valor dos bytes da trama

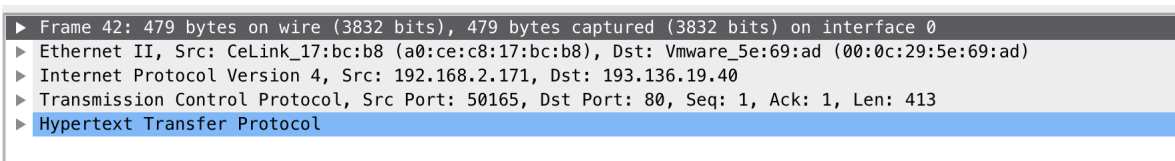
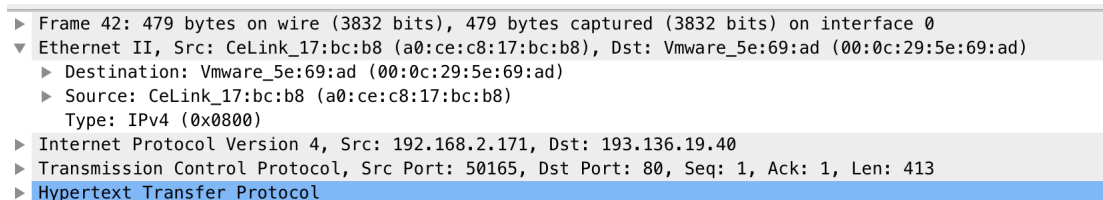


Figure 4: Descrição da trama

Como podemos ver na figura 3, até ao GET temos  $8 \times 2 \times 4 + 2$  (66) bytes. Pela análise da figura 4 vemos que a trama tem 479 bytes ficando assim com uma percentagem com valor 13.78% ( $(66 * 100) / 479$ ).

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?



Ao analisar a imagem acima, podemos concluir que o campo FCS não foi utilizado, visto que este não aparece na parte da Ethernet. Do nosso ponto de vista deve-se ao facto de as ligações por cabo serem ligações muito estáveis e pouco suscetíveis a acumularem erros.

111	3.850182	193.136.19.40	192.168.2.171	TCP	1514	80 → 50165	[ACK] Seq=65161 Ack=414 Win=30080 Len=1448 TSval=2228060866
112	3.850216	192.168.2.171	193.136.19.40	TCP	66	50165 → 80	[ACK] Seq=414 Ack=66609 Win=65856 Len=0 TSval=50148292 TSecr
113	3.850309	193.136.19.40	192.168.2.171	TCP	1514	80 → 50165	[ACK] Seq=66609 Ack=414 Win=30080 Len=1448 TSval=2228060866
114	3.850378	193.136.19.40	192.168.2.171	HTTP	912	HTTP/1.1 200 OK	(text/html)
115	3.850410	192.168.2.171	193.136.19.40	TCP	66	50165 → 80	[ACK] Seq=414 Ack=68903 Win=63616 Len=0 TSval=50148292 TSecr
116	3.855211	192.168.2.171	193.136.19.40	TCP	66	[TCP Window Update] 50165 → 80	[ACK] Seq=414 Ack=68903 Win=96384 Len=0
117	3.856976	192.168.2.171	193.136.19.40	TCP	66	[TCP Window Update] 50165 → 80	[ACK] Seq=414 Ack=68903 Win=129152 Len=0
118	3.987120	192.168.2.171	193.136.19.40	TCP	78	50167 → 80	[SYN, ECN, CWR] Seq=0 Win=65535 Len=0 MSS=1460 WS=64 TSval=5

Figure 5: Pacote TCP selecionado

▼ Ethernet II, Src: Vmware_5e:69:ad (00:0c:29:5e:69:ad), Dst: CeLink_17:bc:b8 (a0:ce:c8:17:bc:b8)
► Destination: CeLink_17:bc:b8 (a0:ce:c8:17:bc:b8)
► Source: Vmware_5e:69:ad (00:0c:29:5e:69:ad)
Type: IPv4 (0x0800)

Figure 6: Campo Ethernet II da resposta HTTP

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

Com a ajuda da figura 6, o endereço ethernet da fonte é 00:0c:29:5e:69:ad e corresponde ao endereço físico do router com que estamos a comunicar.

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

A figura 6 mostra-nos que o endereço ethernet do destino é a0:ce:c8:17:bc:b8 e corresponde ao endereço físico da interface ativo do nosso computador.

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

Os protocolos HTTP(Hypertext Transfer Protocol), IPv4(Internet Protocol Version 4), Ethernet e TCP (Transmition Control Protocol).

# Protocolo ARP

Nesta secção, pretende-se analisar a operação do protocolo ARP.

Verifique o conteúdo da cache ARP do seu computador.

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

```
$ arp -a
? (192.168.2.1) at 0:c:29:5e:69:ad on en5 ifscope [ethernet]
? (224.0.0.251) at 1:0:5e:0:0:fb on en5 ifscope permanent [ethernet]
? (239.255.255.250) at 1:0:5e:7f:ff:fa on en5 ifscope permanent [ethernet]
```

Figure 7: Tabela ARP

R: A primeira coluna tem os endereços IP e a segunda coluna tem os respetivos endereços MAC para os nodos conhecidos em LAN.

Para observar o protocolo ARP em operação, apague novamente a cache ARP e assegure-se que a cache do browser está vazia.

Inicie a captura de tráfego com o Wiresharke e aceda a <http://miei.di.uminho.pt>. Efetue também um ping para um host da sala de aula que esteja a ser usado por outro grupo. Pare a captura de tráfego e tente localizar o tráfego ARP. Se necessário limite os protocolos visíveis apenas a protocolos abaixo do nível IP. Para tal selecione Analyse->Enabled Protocols e remova a opção IPv4 e IPv6.

```

$ sudo arp -d -a
Password:
192.168.2.1 (192.168.2.1) deleted
224.0.0.251 (224.0.0.251) deleted
239.255.255.250 (239.255.255.250) deleted

# Ambrosiny @ MBP-de-Diogo in ~/Desktop/Un
$ arp -a

```

Figure 8: Delete da tabela ARP

No.	Time	Source	Destination	Protocol	Length	Info
5	2.333233	CeLink_17:bc:b8	Broadcast	ARP	42	Who has 192.168.2.1? Tell 192.168.2.171
6	2.333596	Vmware_5e:69:ad	CeLink_17:bc:b8	ARP	60	192.168.2.1 is at 00:0c:29:5e:69:ad
30	7.377833	Vmware_5e:69:ad	CeLink_17:bc:b8	ARP	60	Who has 192.168.2.171? Tell 192.168.2.1
31	7.377884	CeLink_17:bc:b8	Vmware_5e:69:ad	ARP	42	192.168.2.171 is at a0:ce:c8:17:bc:b8
699	37.182334	CeLink_17:bc:b8	Broadcast	ARP	42	Who has 192.168.2.161? Tell 192.168.2.171
700	37.182670	CompalIn_cc:91:76	CeLink_17:bc:b8	ARP	60	192.168.2.161 is at 20:89:84:cc:91:76
714	42.333281	CompalIn_cc:91:76	CeLink_17:bc:b8	ARP	60	Who has 192.168.2.171? Tell 192.168.2.161
715	42.333323	CeLink_17:bc:b8	CompalIn_cc:91:76	ARP	42	192.168.2.171 is at a0:ce:c8:17:bc:b8

Figure 9: Tráfego ARP

```

$ ping 192.168.2.161
PING 192.168.2.161 (192.168.2.161): 56 data bytes
64 bytes from 192.168.2.161: icmp_seq=0 ttl=64 time=0.675 ms
64 bytes from 192.168.2.161: icmp_seq=1 ttl=64 time=0.433 ms
64 bytes from 192.168.2.161: icmp_seq=2 ttl=64 time=0.358 ms
64 bytes from 192.168.2.161: icmp_seq=3 ttl=64 time=0.361 ms
^C

```

Figure 10: Ping para 192.168.2.161

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

A origem tem o valor a0:ce:c8:17:bc:b8. O destino tem o valor ff:ff:ff:ff:ff:ff. Este valor deve-se ao facto de a nossa tabela arp não ter o valor do endereço mac associado ao endereço ip para o qual mandamos o ping. Assim é preciso enviar para todos os dispositivos na rede para que o destino possa responder e assim guardar o valor do endereço mac. Para isto usamos o endereço de broadcast (ff:ff:ff:ff:ff:ff).

```
▶ Frame 699: 42 bytes on wire (336 bits), 42 bytes captured (336 bits) on interface 0
▼ Ethernet II, Src: CeLink_17:bc:b8 (a0:ce:c8:17:bc:b8), Dst: Broadcast (ff:ff:ff:ff:ff:ff)
  ▶ Destination: Broadcast (ff:ff:ff:ff:ff:ff)
  ▶ Source: CeLink_17:bc:b8 (a0:ce:c8:17:bc:b8)
    Type: ARP (0x0806)
▶ Address Resolution Protocol (request)
```

Figure 11: Campo Ethernet

11. Qual o valor hexadecimal do campo tipo de trama Ethernet? O que indica?

Como mostra na figura 12, o campo type tem o valor 0x0806 e indica que a camada acima está a usar o protocolo ARP (Address Resolution Protocol).

```
▶ Frame 699: 42 bytes on wire (336 bits), 42 bytes captured (336 bits) on interface 0
▼ Ethernet II, Src: CeLink_17:bc:b8 (a0:ce:c8:17:bc:b8), Dst: Broadcast (ff:ff:ff:ff:ff:ff)
  ▶ Destination: Broadcast (ff:ff:ff:ff:ff:ff)
  ▶ Source: CeLink_17:bc:b8 (a0:ce:c8:17:bc:b8)
    Type: ARP (0x0806)
▶ Address Resolution Protocol (request)
```

Figure 12: Type

12. Qual o valor do campo ARP opcode? O que especifica? Se necessário, consulte a RFC do protocolo ARP <http://tools.ietf.org/html/rfc826.html>.

Na figura 13, podemos ver que o opcode tem o valor 1 que representa um request. Assim podemos concluir que a nossa máquina está a pedir aos dispositivos em rede para que respondam se o seu ip for o pretendido.

```
▼ Address Resolution Protocol (request)
  Hardware type: Ethernet (1)
  Protocol type: IPv4 (0x0800)
  Hardware size: 6
  Protocol size: 4
  Opcode: request (1)
  Sender MAC address: CeLink_17:bc:b8 (a0:ce:c8:17:bc:b8)
  Sender IP address: 192.168.2.171
```

Figure 13: Opcode do ARP



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

Uma mensagem ARP contém endereços MAC e IP. Com isto concluímos que o protocolo ARP serve para converter um endereço IP no endereço mac da interface ativa respectiva.

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

Quando o ping é feito, a nossa tabela arp não tem uma associação entre o IP para o qual o ping foi executado e o respectivo endereço mac. Assim é enviado uma mensagem ARP para todos os dispositivos na rede para que o endereço IP pretendido, caso receba a mensagem, responda com o seu endereço mac.

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

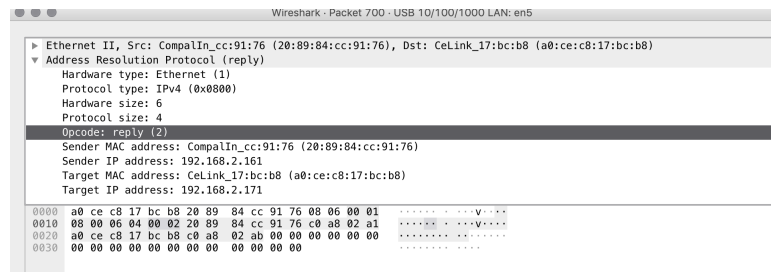


Figure 14: Mensagem de resposta

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

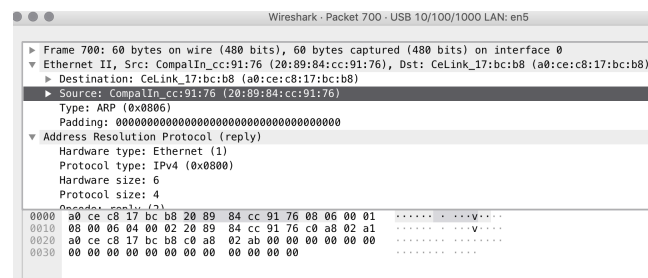


Figure 15: Opcode

O campo opcode tem o valor 2 e significa que o endereço 192.168.2.161 recebem a mensagem de request e está a comunicar o seu endereço mac de volta.

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

A resposta à mensagem está nos bytes entre 23 e 28 da trama, ou seja, é o sender mac address da source na mensagem de resposta.

# ARP Gratuito

Arranque Wireshark na sua máquina nativa e inicie a captura de dados.

Desligue e volte a ligar a sua ligação à rede local Ethernet, ou force o pedido de atribuição de um novo endereço IP à interface em uso. Pare a captura de tráfego. Utilize o filtro de visualização ARP para facilitar a identificação dos pacotes respetivos.

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?

91	14.856914	CeLink_17:bc:b8	Vmware_5e:69:ad	ARP	42 Who has 192.168.2.1? Tell 192.168.2.171
95	14.879460	CeLink_17:bc:b8	Vmware_5e:69:ad	ARP	42 Who has 192.168.2.1? Tell 192.168.2.171
96	14.924724	CeLink_17:bc:b8	Vmware_5e:69:ad	ARP	42 Who has 192.168.2.1? Tell 192.168.2.171
97	15.010003	CeLink_17:bc:b8	Vmware_5e:69:ad	ARP	42 Who has 192.168.2.1? Tell 192.168.2.171
98	15.010359	Vmware_5e:69:ad	CeLink_17:bc:b8	ARP	60 192.168.2.1 is at 00:0c:29:5e:69:ad
99	15.010803	CeLink_17:bc:b8	Broadcast	ARP	42 Gratuitous ARP for 192.168.2.171 (Request)
100	15.015431	CeLink_17:bc:b8	Broadcast	ARP	42 Who has 169.254.255.255? Tell 192.168.2.171
101	15.041272	CeLink_17:bc:b8	Broadcast	ARP	42 Who has 192.168.2.1? Tell 192.168.2.171
102	15.041548	Vmware_5e:69:ad	CeLink_17:bc:b8	ARP	60 192.168.2.1 is at 00:0c:29:5e:69:ad
109	15.340510	CeLink_17:bc:b8	Broadcast	ARP	42 Who has 169.254.255.255? Tell 192.168.2.171
123	15.662060	CeLink_17:bc:b8	Broadcast	ARP	42 Who has 169.254.255.255? Tell 192.168.2.171
152	15.983521	CeLink_17:bc:b8	Broadcast	ARP	42 Who has 169.254.255.255? Tell 192.168.2.171
216	16.305604	CeLink_17:bc:b8	Broadcast	ARP	42 Who has 169.254.255.255? Tell 192.168.2.171

Figure 16: ARP Gratuito

▶ Frame 99: 42 bytes on wire (336 bits), 42 bytes captured (336 bits) on interface 0			
▼ Ethernet II, Src: CeLink_17:bc:b8 (a0:ce:c8:17:bc:b8), Dst: Broadcast (ff:ff:ff:ff:ff:ff)			
▶ Destination: Broadcast (ff:ff:ff:ff:ff:ff)			
▶ Source: CeLink_17:bc:b8 (a0:ce:c8:17:bc:b8)			
Type: ARP (0x0806)			
▼ Address Resolution Protocol (request/gratuitous ARP)			
Hardware type: Ethernet (1)			
Protocol type: IPv4 (0x0800)			
Hardware size: 6			
Protocol size: 4			
Opcode: request (1)			
[Target IP address: 0.0.0.0]			
0000	ff ff ff ff ff ff a0 ce	c8 17 bc b8 08 06 00 01	.....
0010	08 00 06 04 00 01 a0 ce	c8 17 bc b8 c0 a8 02 ab	.....
0020	00 00 00 00 00 00 c0 a8	02 ab	.....

Figure 17: ARP valores

A grande diferença entre os valores do ARP gratuito e os outros é o valor do Target IP address que aqui tem o mesmo valor que o Target IP address e nos outros tem valor diferente. Como não temos um target IP não é suposto outros dispositivos, para além do router, responderem aos ARP. Como podemos ver na figura 16, apenas o router a que estamos ligados respondeu.

# Domínios de colisão

Construa uma topologia no emulador CORE com um Laptop (n1) e três servidores (n2,n3,n4) interligados através de um hub(repetidor).

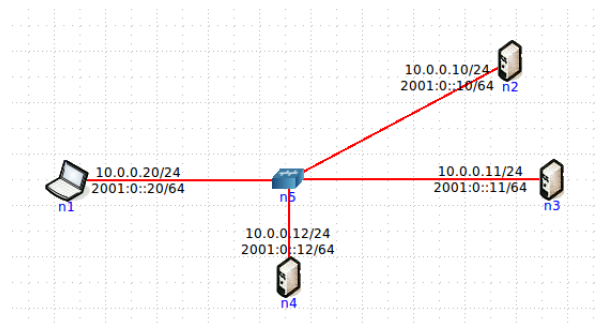


Figure 18: Topologia implementada

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.34389/n1.conf
root@n1:/tmp/pycore.34389/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=0.046 ms
64 bytes from 10.0.0.10: icmp_req=2 ttl=64 time=0.065 ms
64 bytes from 10.0.0.10: icmp_req=3 ttl=64 time=0.041 ms
64 bytes from 10.0.0.10: icmp_req=4 ttl=64 time=0.044 ms
64 bytes from 10.0.0.10: icmp_req=5 ttl=64 time=0.048 ms
64 bytes from 10.0.0.10: icmp_req=6 ttl=64 time=0.044 ms
^C
--- 10.0.0.10 ping statistics ---
6 packets transmitted, 6 received, 0% packet loss, time 5000ms
rtt min/avg/max/mdev = 0.041/0.048/0.065/0.007 ms
root@n1:/tmp/pycore.34389/n1.conf#
```

Figure 19: N1

```
root@n2:/tmp/pycore.34389/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
12:26:15.860996 IP 10.0.0.20 > 89: ICMP echo request, id 183, seq 1, length 64
12:26:16.862697 IP 10.0.0.20 > 89: ICMP echo request, id 183, seq 2, length 64
12:26:16.862714 IP 10.0.0.20 > 89: ICMP echo request, id 183, seq 2, length 64
12:26:17.861690 IP 10.0.0.20 > 89: ICMP echo request, id 183, seq 3, length 64
12:26:17.861694 IP 10.0.0.20 > 89: ICMP echo request, id 183, seq 3, length 64
12:26:18.861254 IP 10.0.0.20 > 89: ICMP echo request, id 183, seq 4, length 64
12:26:19.861248 IP 10.0.0.20 > 89: ICMP echo request, id 183, seq 4, length 64
12:26:19.861253 IP 10.0.0.20 > 89: ICMP echo request, id 183, seq 5, length 64
12:26:19.861258 IP 10.0.0.20 > 89: ICMP echo request, id 183, seq 5, length 64
12:26:20.861252 IP 10.0.0.20 > 89: ICMP echo request, id 183, seq 6, length 64
12:26:20.861265 IP 10.0.0.20 > 89: ICMP echo request, id 183, seq 6, length 64
12:26:20.866318 ARP, Request who-has 10.0.0.20 tell 89, length 28
12:26:20.866356 ARP, Reply 10.0.0.20 is-at 00:00:00:aa:00:00 (oui Ethernet), length 28

14 packets captured
14 packets received by filter
0 packets dropped by kernel
root@n2:/tmp/pycore.34389/n2.conf#
```

Figure 20: N2

```
root@n3:/tmp/pycore.34389/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
12:26:15.860994 IP 10.0.0.20 > 89: ICMP echo request, id 183, seq 1, length 64
12:26:16.862695 IP 10.0.0.20 > 89: ICMP echo request, id 183, seq 1, length 64
12:26:16.862720 IP 10.0.0.20 > 89: ICMP echo request, id 183, seq 2, length 64
12:26:17.861678 IP 10.0.0.20 > 89: ICMP echo request, id 183, seq 3, length 64
12:26:17.861697 IP 10.0.0.20 > 89: ICMP echo request, id 183, seq 3, length 64
12:26:18.861233 IP 10.0.0.20 > 89: ICMP echo request, id 183, seq 4, length 64
12:26:18.861251 IP 10.0.0.20 > 89: ICMP echo request, id 183, seq 4, length 64
12:26:19.861252 IP 10.0.0.20 > 89: ICMP echo request, id 183, seq 5, length 64
12:26:19.861271 IP 10.0.0.20 > 89: ICMP echo request, id 183, seq 5, length 64
12:26:20.861250 IP 10.0.0.20 > 89: ICMP echo request, id 183, seq 6, length 64
12:26:20.861259 IP 10.0.0.20 > 89: ICMP echo request, id 183, seq 6, length 64
12:26:20.866339 ARP, Request who-has 10.0.0.20 tell 89, length 28
12:26:20.866355 ARP, Reply 10.0.0.20 is-at 00:00:00:aa:00:00 (oui Ethernet), length 28

14 packets captured
14 packets received by filter
0 packets dropped by kernel
root@n3:/tmp/pycore.34389/n3.conf#
```

Figure 21: N3

```
root@n4:/tmp/pycore.34389/n4.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
12:26:15.860992 IP 10.0.0.20 > 89: ICMP echo request, id 183, seq 1, length 64
12:26:16.862692 IP 10.0.0.20 > 89: ICMP echo request, id 183, seq 1, length 64
12:26:16.862719 IP 10.0.0.20 > 89: ICMP echo request, id 183, seq 2, length 64
12:26:17.861677 IP 10.0.0.20 > 89: ICMP echo request, id 183, seq 3, length 64
12:26:17.861696 IP 10.0.0.20 > 89: ICMP echo request, id 183, seq 3, length 64
12:26:18.861231 IP 10.0.0.20 > 89: ICMP echo request, id 183, seq 4, length 64
12:26:18.861250 IP 10.0.0.20 > 89: ICMP echo request, id 183, seq 4, length 64
12:26:19.861250 IP 10.0.0.20 > 89: ICMP echo request, id 183, seq 5, length 64
12:26:19.861270 IP 10.0.0.20 > 89: ICMP echo request, id 183, seq 5, length 64
12:26:20.861248 IP 10.0.0.20 > 89: ICMP echo request, id 183, seq 6, length 64
12:26:20.861268 IP 10.0.0.20 > 89: ICMP echo request, id 183, seq 6, length 64
12:26:20.866337 ARP, Request who-has 10.0.0.20 tell 89, length 28
12:26:20.866354 ARP, Reply 10.0.0.20 is-at 00:00:00:aa:00:00 (oui Ethernet), length 28

14 packets captured
14 packets received by filter
0 packets dropped by kernel
root@n4:/tmp/pycore.34389/n4.conf#
```

Figure 22: N4

Um hub é um dispositivo que trabalha a nível físico juntando várias portas num único segmento de rede. Quando é enviado algo entre duas portas, o hub envia o input que recebeu para todas as portas ligadas à exceção da porta que enviou o input. Como podemos ver pelas imagens, o tráfego que passa em n2,n3 e n4 é o mesmo, visto que o hub redireciona para todas as portas. Com isto concluímos que acabam por chegar mensagens que não são supostas a outros dispositivos.

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.

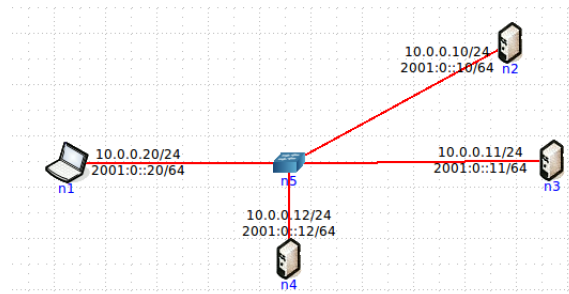


Figure 23: Topologia implementada

```
root@n1:/tmp/pycore.34390/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=0.063 ms
64 bytes from 10.0.0.10: icmp_req=2 ttl=64 time=0.031 ms
64 bytes from 10.0.0.10: icmp_req=3 ttl=64 time=0.076 ms
^C
--- 10.0.0.10 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 1998ms
rtt min/avg/max/mdev = 0.031/0.056/0.076/0.020 ms
root@n1:/tmp/pycore.34390/n1.conf#
```

Figure 24: N1

```
root@n2:/tmp/pycore.34390/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
12:30:53.271632 ARP, Request who-has A9 tell 10.0.0.20, length 28
12:30:53.271717 ARP, Reply A9 is-at 00:00:00:aa:00:01 (oui Ethernet), length 28
12:30:53.271724 IP 10.0.0.20 > A9: ICMP echo request, id 72, seq 1, length 64
12:30:53.271729 IP A9 > 10.0.0.20: ICMP echo reply, id 72, seq 1, length 64
12:30:54.270686 IP 10.0.0.20 > A9: ICMP echo request, id 72, seq 2, length 64
12:30:54.270696 IP A9 > 10.0.0.20: ICMP echo reply, id 72, seq 2, length 64
12:30:55.270357 IP 10.0.0.20 > A9: ICMP echo request, id 72, seq 3, length 64
12:30:55.270380 IP A9 > 10.0.0.20: ICMP echo reply, id 72, seq 3, length 64
8 packets captured
8 packets received by filter
0 packets dropped by kernel
root@n2:/tmp/pycore.34390/n2.conf#
```

Figure 25: N2

```
root@n3:/tmp/pycore.34390/n3.conf
root@n3:/tmp/pycore.34390/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
^C12:30:53.271690 ARP, Request who-has A9 tell 10.0.0.20, length 28

1 packet captured
1 packet received by filter
0 packets dropped by kernel
root@n3:/tmp/pycore.34390/n3.conf#
```

Figure 26: N3

```
root@n4:/tmp/pycore.34390/n4.conf
root@n4:/tmp/pycore.34390/n4.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
^C12:30:53.271689 ARP, Request who-has A9 tell 10.0.0.20, length 28

1 packet captured
1 packet received by filter
0 packets dropped by kernel
root@n4:/tmp/pycore.34390/n4.conf#
```

Figure 27: N4

Quando usamos a topologia com hub, o tráfego é enviado para todos os dispositivos. Ao mudar para switch, o tráfego capturado pelo tcpdump muda. Ao contrário da primeira topologia, agora o tráfego é apenas direcionado para o n2. Os outros dispositivos n3 e n4, possuem apenas uma captura que é a resultante do envio ARP Broadcast como podemos ver nas figura 26 e 27. Com isto concluímos que os switches ao evitarem enviar a informação para todos os hosts fazem com que o risco de haver colisões seja menor. Em contra partida, o hub como junta tudo num canal de transmissão único e repete muita da informação está mais propício a colisões. Um switch possui uma tabela de endereçamento que permite fazer o redirecionamento apenas para o nó pretendido.

# Conclusão

Neste trabalho prático realizado foi-nos possível aprofundar o conhecimento acerca as tramas de Ethernet e que estas estão organizadas em bytes, bem como, conhecer um melhor o protocolo ARP. Para realização deste utilizamos o simulador de redes (CORE) e ainda um de captura e análise de tramas (Wireshark).

A utilização desta segunda ferramenta foi essencial pois permitiu a observação dos protocolos envolvidos, qual o encapsulamento a ter aquando a transferência de processos e verificar outras propriedades importantes, que permitem ter conhecimento e informações sobre os endereços envolvidos, o tipo da mensagem ARP, assim como a análise de um pedido ARP.

Outro ponto abordado foi a análise de um ARP Gratuito, verificando assim que apesar de ser algo que acontece sem que se tenha pedido para o fazer este torna-se útil na verificação de um host ter o mesmo endereço IP que o originador do pedido, assim como permite informar hosts e/ou switches novos endereços MAC para que todos os sistemas da rede possam atualizar as suas tabela ARP.

Desta forma pudemos consolidar conceitos, protocolos e comportamento dos dados transmitidos em Ethernet.