



TP3: Camada de Ligação Lógica: Ethernet e Protocolo ARP

Redes de Computadores

Grupo 1 – PL4



Ana Pereira A81712



Ana Ribeiro A82474



Jéssica Lemos A82061

1 Questões e Respostas

3. Captura e análise de Tramas Ethernet

http						
No.	Time	Source	Destination	Protocol	Length	Info
28	4.820334	192.168.100.223	193.136.19.40	HTTP	648	GET / HTTP/1.1
31	4.828270	193.136.19.40	192.168.100.223	HTTP	247	HTTP/1.1 304 Not Modified
49	4.966583	192.168.100.223	193.136.19.40	HTTP	556	GET /favicon.ico HTTP/1.1
50	4.969003	193.136.19.40	192.168.100.223	HTTP	1514	HTTP/1.1 200 OK (image/vnd.microsoft.icon)

Figura 1 - Captura de Tramas Ethernet de mensagens HTTP

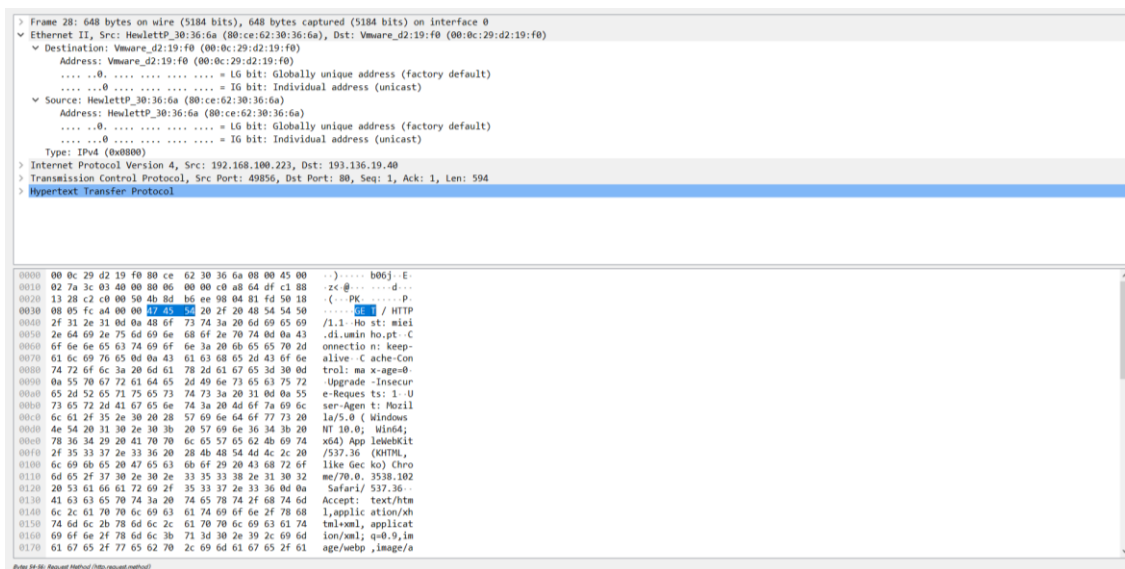


Figura 2 - Trama Ethernet que contém a mensagem HTTP GET

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

Como podemos observar na Figura 2, o endereço MAC de origem é 80:ce:62:30:36:6a e de destino 00:0c:29:d2:19:f0.

2. Identifique a que sistemas se referem. Justifique.

O endereço MAC de origem, que é indicado no campo *Source*, corresponde ao da interface da nossa máquina nativa. Enquanto que o de destino, referente ao campo *Destination*, é o router da rede local à qual estamos ligados.

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

O valor do campo Type da trama Ethernet é 0x0800 e indica qual o protocolo utilizado ao nível de rede, ou seja, 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.

Desde o início da trama até ao caractere ASCII “G” são usados 54 bytes. Como no total, são utilizados 648 bytes, podemos verificar que a sobrecarga introduzida pela pilha é: $(54/648) * 100 \approx 8.333 \%$

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?

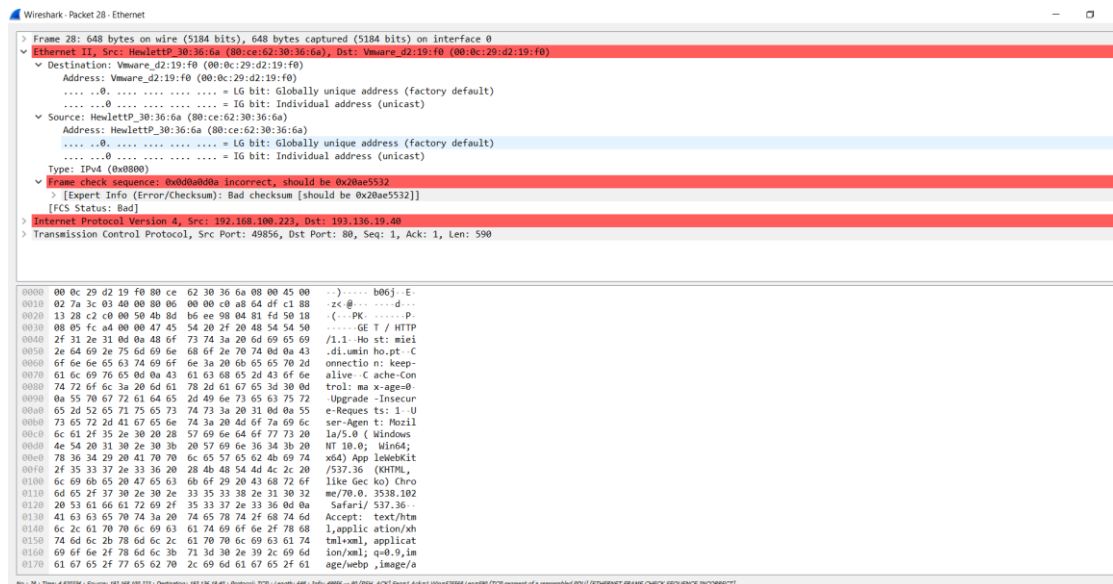


Figura 3 - Visualização da trama em análise

Como podemos verificar na Figura 3, o valor do campo FCS é 0x0d0a0d0a. Desta forma, o Wireshark indica que existe um erro na trama quando estipulamos que este deve interpretar os últimos bytes da trama como FCS. Assim, este campo não está a ser utilizado para controlo de erros. É raro existirem erros no meio cabelado, pelo que não compensa a utilização deste campo.

A seguir responda às seguintes perguntas, baseado no conteúdo da trama Ethernet que contém o primeiro byte da resposta HTTP.

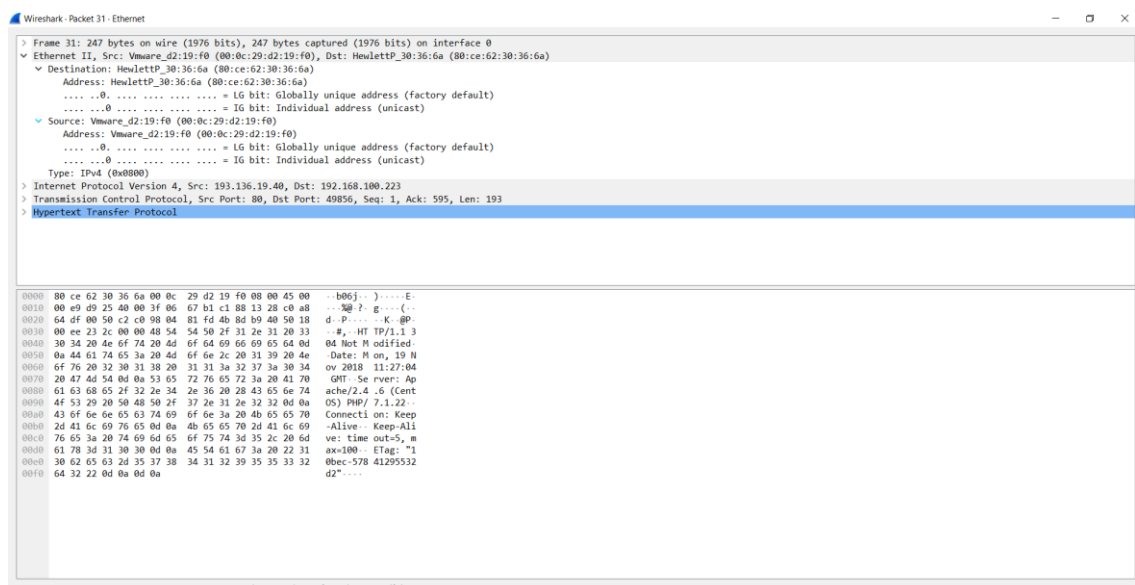


Figura 4 - Trama Ethernet que contém o primeiro byte da resposta HTTP

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

Justifique.

O endereço Ethernet da fonte, que é indicado no campo *Source*, é 00:0c:29:d2:19:f0, que corresponde ao router da rede local à qual estamos ligados.

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

O endereço de destino é 80:ce:62:30:36:6a, que representa a interface ativa da nossa máquina nativa.

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

Como podemos observar na Figura 4, os protocolos contidos na trama recebida são: Ethernet II, IPv4 (Internet Protocol Version 4), TCP (Transmission Control Protocol) e HTTP (Hypertext Transfer Protocol).

4. Protocolo ARP

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

```
Administrator: Linha de comandos
> arp -s 157.55.85.212 00-aa-00-62-c6-09 .... Adds a static entry.
> arp -a .... Displays the arp table.

C:\WINDOWS\system32>arp -a

Interface: 192.168.100.223 --- 0x11
Internet Address      Physical Address      Type
192.168.100.154       cc-2d-8c-06-1e-27     dynamic
192.168.100.161       e8-03-9a-17-4e-54     dynamic
192.168.100.163       2c-60-0c-f6-42-be     dynamic
192.168.100.209       40-6c-8f-3b-d7-52     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
224.0.0.113           01-00-5e-00-00-71     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-fa-fa     static
255.255.255.255       ff-ff-ff-ff-ff-ff     static
```

Figura 5 - Tabela ARP

A tabela ARP mapeia o endereço IP para o endereço MAC dos sistemas que comunicaram recentemente. Assim, a primeira coluna representa o endereço IP, a segunda o endereço MAC e a terceira o tipo de endereçamento que está a ser utilizado.

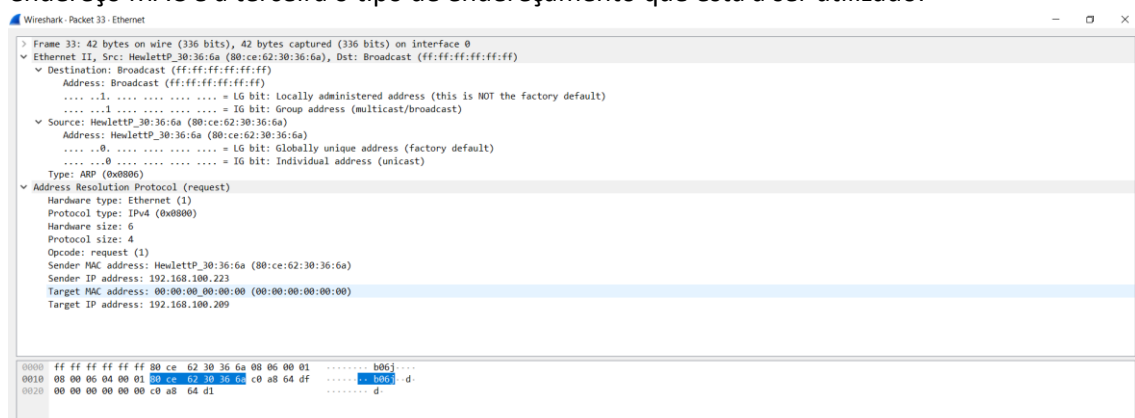


Figura 6 - Trama Ethernet que contém a mensagem com o pedido ARP (ARP Request)

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?

O valor do endereço de origem é 80:ce:62:30:36:6a e o de destino é ff:ff:ff:ff:ff:ff como podemos verificar na Figura 6. Este endereço de destino leva a que todos os endereços conectados à rede recebam a mensagem com o pedido, contudo só o endereço pretendido responde com o seu endereço MAC.

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

O valor do campo tipo da trama é 0x0806, como podemos verificar na Figura 6, sugerindo que o campo de dados pertence ao ARP.

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>.

Como podemos observar na Figura 6, o valor do campo ARP opcode é 0x0001, que indica se é um pedido ou uma resposta. Neste caso, corresponde a um pedido.

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

A mensagem ARP contém os endereços IP do destino e da origem, no entanto apenas é conhecido o endereço MAC correspondente do endereço IP da origem, dado que é um ARP request.

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

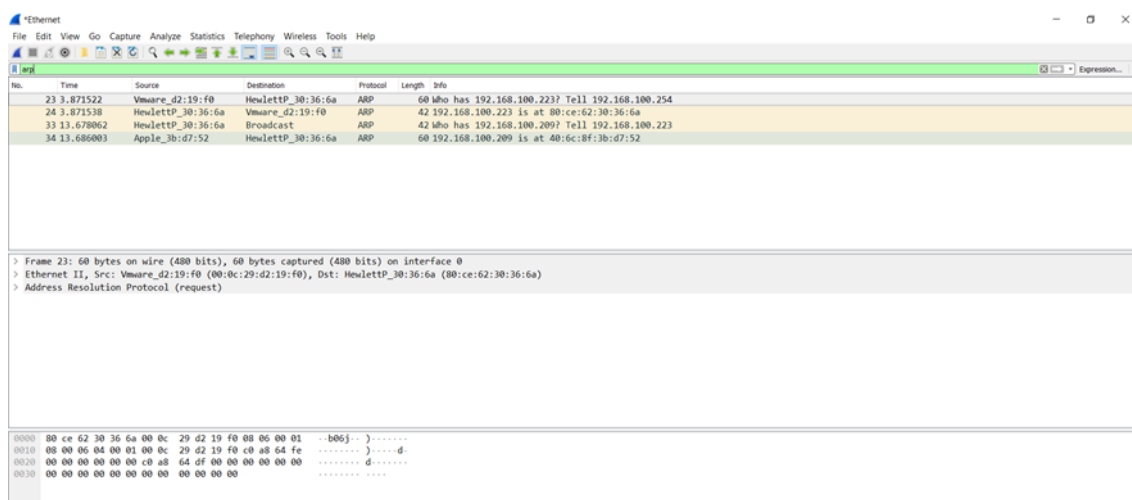


Figura 7 - Tráfego ARP

A nossa máquina pergunta “Quem tem o endereço 192.168.100.209? Diga 192.168.100.223”. Em resposta iremos obter o endereço MAC do equipamento com o endereço indicado.

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

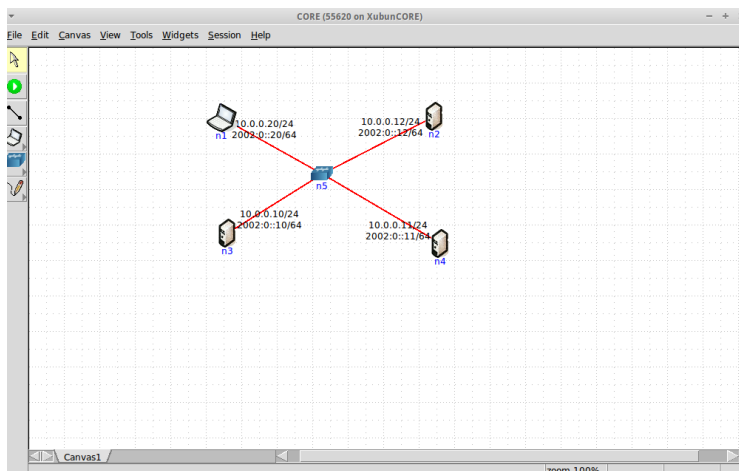


Figura 12 - Topologia CORE

```
root@n1: /tmp/pycore.55621/n1.conf
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=0.045 ms
64 bytes from 10.0.0.12: icmp_req=2 ttl=64 time=0.151 ms
64 bytes from 10.0.0.12: icmp_req=3 ttl=64 time=0.217 ms
64 bytes from 10.0.0.12: icmp_req=4 ttl=64 time=0.108 ms
64 bytes from 10.0.0.12: icmp_req=5 ttl=64 time=0.261 ms
64 bytes from 10.0.0.12: icmp_req=6 ttl=64 time=0.112 ms
64 bytes from 10.0.0.12: icmp_req=7 ttl=64 time=0.010 ms
64 bytes from 10.0.0.12: icmp_req=8 ttl=64 time=0.112 ms
64 bytes from 10.0.0.12: icmp_req=9 ttl=64 time=0.141 ms
64 bytes from 10.0.0.12: icmp_req=10 ttl=64 time=0.111 ms
64 bytes from 10.0.0.12: icmp_req=11 ttl=64 time=0.132 ms
64 bytes from 10.0.0.12: icmp_req=12 ttl=64 time=0.132 ms
64 bytes from 10.0.0.12: icmp_req=13 ttl=64 time=0.133 ms
64 bytes from 10.0.0.12: icmp_req=14 ttl=64 time=0.138 ms
64 bytes from 10.0.0.12: icmp_req=15 ttl=64 time=0.118 ms
64 bytes from 10.0.0.12: icmp_req=16 ttl=64 time=0.134 ms
64 bytes from 10.0.0.12: icmp_req=17 ttl=64 time=0.145 ms
64 bytes from 10.0.0.12: icmp_req=18 ttl=64 time=0.148 ms
64 bytes from 10.0.0.12: icmp_req=19 ttl=64 time=0.137 ms
64 bytes from 10.0.0.12: icmp_req=20 ttl=64 time=0.145 ms
^C
--- 10.0.0.12 ping statistics ---
20 packets transmitted, 20 received, 0% packet loss, time 19004ms
rtt min/avg/max/mdev = 0.010/0.131/0.261/0.051 ms
root@n1: /tmp/pycore.55621/n1.conf#
```

Figura 13 - Ping do laptop n1 para o servidor n2

```
root@n2: /tmp/pycore.55621/n2.conf
11:39:07.277285 IP A11 > 10.0.0.20: ICMP echo reply, id 182, seq 11, length 64
11:39:08.276417 IP 10.0.0.20 > A11: ICMP echo request, id 182, seq 12, length 64
11:39:08.276463 IP A11 > 10.0.0.20: ICMP echo reply, id 182, seq 12, length 64
11:39:09.276404 IP 10.0.0.20 > A11: ICMP echo request, id 182, seq 13, length 64
11:39:09.276449 IP A11 > 10.0.0.20: ICMP echo reply, id 182, seq 13, length 64
11:39:10.277075 IP 10.0.0.20 > A11: ICMP echo request, id 182, seq 14, length 64
11:39:10.277121 IP A11 > 10.0.0.20: ICMP echo reply, id 182, seq 14, length 64
11:39:11.278704 IP 10.0.0.20 > A11: ICMP echo request, id 182, seq 15, length 64
11:39:11.278745 IP A11 > 10.0.0.20: ICMP echo reply, id 182, seq 15, length 64
11:39:12.277722 IP 10.0.0.20 > A11: ICMP echo request, id 182, seq 16, length 64
11:39:12.277768 IP A11 > 10.0.0.20: ICMP echo reply, id 182, seq 16, length 64
11:39:13.276728 IP 10.0.0.20 > A11: ICMP echo request, id 182, seq 17, length 64
11:39:13.276772 IP A11 > 10.0.0.20: ICMP echo reply, id 182, seq 17, length 64
11:39:14.276441 IP 10.0.0.20 > A11: ICMP echo request, id 182, seq 18, length 64
11:39:14.276486 IP A11 > 10.0.0.20: ICMP echo reply, id 182, seq 18, length 64
11:39:15.279802 IP 10.0.0.20 > A11: ICMP echo request, id 182, seq 19, length 64
11:39:15.279846 IP A11 > 10.0.0.20: ICMP echo reply, id 182, seq 19, length 64
11:39:16.281364 IP 10.0.0.20 > A11: ICMP echo request, id 182, seq 20, length 64
11:39:16.281407 IP A11 > 10.0.0.20: ICMP echo reply, id 182, seq 20, length 64

24 packets captured
24 packets received by filter
0 packets dropped by kernel
root@n2: /tmp/pycore.55621/n2.conf#
```

Figura 14 - tcpdump no servidor n2


```
root@n3: /tmp/pycore.55621/n3.conf
root@n3: /tmp/pycore.55621/n3.conf# tcpdump
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), capture size 262144 bytes
^C11:39:10.277091 IP 10.0.0.20 > A11: ICMP echo request, id 182, seq 14, length
64
11:39:10.277157 IP A11 > 10.0.0.20: ICMP echo reply, id 182, seq 14, length 64
11:39:11.278716 IP 10.0.0.20 > A11: ICMP echo request, id 182, seq 15, length 64
11:39:11.278780 IP A11 > 10.0.0.20: ICMP echo reply, id 182, seq 15, length 64
11:39:12.277734 IP 10.0.0.20 > A11: ICMP echo request, id 182, seq 16, length 64
11:39:12.277803 IP A11 > 10.0.0.20: ICMP echo reply, id 182, seq 16, length 64
11:39:13.276740 IP 10.0.0.20 > A11: ICMP echo request, id 182, seq 17, length 64
11:39:13.276818 IP A11 > 10.0.0.20: ICMP echo reply, id 182, seq 17, length 64
11:39:14.276454 IP 10.0.0.20 > A11: ICMP echo request, id 182, seq 18, length 64
11:39:14.276530 IP A11 > 10.0.0.20: ICMP echo reply, id 182, seq 18, length 64
11:39:15.279814 IP 10.0.0.20 > A11: ICMP echo request, id 182, seq 19, length 64
11:39:15.279884 IP A11 > 10.0.0.20: ICMP echo reply, id 182, seq 19, length 64
11:39:16.281376 IP 10.0.0.20 > A11: ICMP echo request, id 182, seq 20, length 64
11:39:16.281449 IP A11 > 10.0.0.20: ICMP echo reply, id 182, seq 20, length 64

14 packets captured
14 packets received by filter
0 packets dropped by kernel
root@n3: /tmp/pycore.55621/n3.conf#
```

Figura 15 - tcpdump no servidor n3

```
root@n4: /tmp/pycore.55621/n4.conf
root@n4: /tmp/pycore.55621/n4.conf# tcpdump
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), capture size 262144 bytes
^C11:39:13.276735 IP 10.0.0.20 > A11: ICMP echo request, id 182, seq 17, length
64
11:39:13.276814 IP A11 > 10.0.0.20: ICMP echo reply, id 182, seq 17, length 64
11:39:14.276449 IP 10.0.0.20 > A11: ICMP echo request, id 182, seq 18, length 64
11:39:14.276526 IP A11 > 10.0.0.20: ICMP echo reply, id 182, seq 18, length 64
11:39:15.279809 IP 10.0.0.20 > A11: ICMP echo request, id 182, seq 19, length 64
11:39:15.279880 IP A11 > 10.0.0.20: ICMP echo reply, id 182, seq 19, length 64
11:39:16.281371 IP 10.0.0.20 > A11: ICMP echo request, id 182, seq 20, length 64
11:39:16.281445 IP A11 > 10.0.0.20: ICMP echo reply, id 182, seq 20, length 64

8 packets captured
8 packets received by filter
0 packets dropped by kernel
root@n4: /tmp/pycore.55621/n4.conf#
```

Figura 16 - tcpdump no servidor s4

Como podemos verificar nas figuras anteriormente apresentadas, ao fazer *ping* do laptop n1 para o servidor n2 e recorrendo ao comando *tcpdump*, constatamos que todos os servidores recebem os pacotes. Tal deve-se ao facto de o *hub* ao receber um pacote de dados reencaminha-os para todos os dispositivos que se encontram 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.

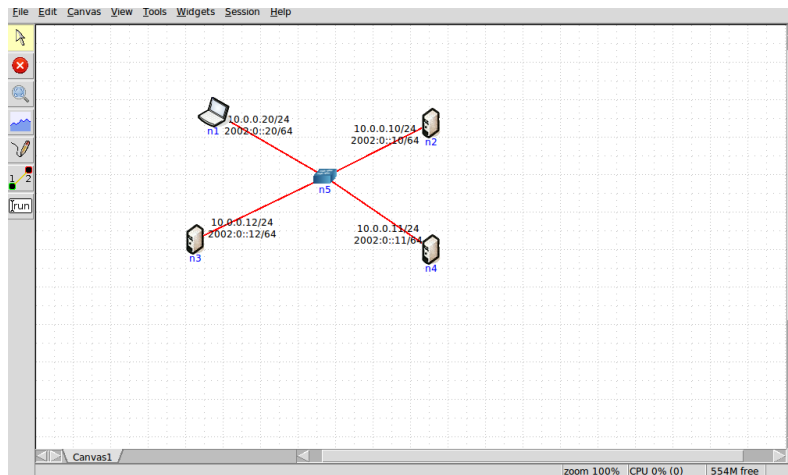


Figura 17 - Topologia CORE

```

root@n1: /tmp/pycore.48514/n1.conf
root@n1:/tmp/pycore.48514/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=0.070 ms
64 bytes from 10.0.0.10: icmp_req=2 ttl=64 time=0.047 ms
64 bytes from 10.0.0.10: icmp_req=3 ttl=64 time=0.064 ms
64 bytes from 10.0.0.10: icmp_req=4 ttl=64 time=0.047 ms
64 bytes from 10.0.0.10: icmp_req=5 ttl=64 time=0.054 ms
64 bytes from 10.0.0.10: icmp_req=6 ttl=64 time=0.042 ms
64 bytes from 10.0.0.10: icmp_req=7 ttl=64 time=0.037 ms
64 bytes from 10.0.0.10: icmp_req=8 ttl=64 time=0.059 ms
64 bytes from 10.0.0.10: icmp_req=9 ttl=64 time=0.050 ms
^C
--- 10.0.0.10 ping statistics ---
9 packets transmitted, 9 received, 0% packet loss, time 7999ms
rtt min/avg/max/mdev = 0.037/0.052/0.070/0.011 ms
root@n1:/tmp/pycore.48514/n1.conf#

```

Figura 18 - Ping no laptop n1 para o servidor n2

```

root@n2: /tmp/pycore.48514/n2.conf
14:51:30.632904 IP 10.0.0.20 > A9: ICMP echo request, id 126, seq 2, length 64
14:51:30.632915 IP A9 > 10.0.0.20: ICMP echo reply, id 126, seq 2, length 64
14:51:31.631909 IP 10.0.0.20 > A9: ICMP echo request, id 126, seq 3, length 64
14:51:31.631930 IP A9 > 10.0.0.20: ICMP echo reply, id 126, seq 3, length 64
14:51:32.630905 IP 10.0.0.20 > A9: ICMP echo request, id 126, seq 4, length 64
14:51:32.630919 IP A9 > 10.0.0.20: ICMP echo reply, id 126, seq 4, length 64
14:51:33.629912 IP 10.0.0.20 > A9: ICMP echo request, id 126, seq 5, length 64
14:51:33.629930 IP A9 > 10.0.0.20: ICMP echo reply, id 126, seq 5, length 64
14:51:34.629700 IP 10.0.0.20 > A9: ICMP echo request, id 126, seq 6, length 64
14:51:34.629712 IP A9 > 10.0.0.20: ICMP echo reply, id 126, seq 6, length 64
14:51:34.637680 ARP, Request who-has 10.0.0.20 tell A9, length 28
14:51:34.637718 ARP, Reply 10.0.0.20 is-at 00:00:00:aa:00:00 (oui Ethernet), length 28
14:51:35.629758 IP 10.0.0.20 > A9: ICMP echo request, id 126, seq 7, length 64
14:51:35.629768 IP A9 > 10.0.0.20: ICMP echo reply, id 126, seq 7, length 64
14:51:36.629814 IP 10.0.0.20 > A9: ICMP echo request, id 126, seq 8, length 64
14:51:36.629832 IP A9 > 10.0.0.20: ICMP echo reply, id 126, seq 8, length 64
14:51:37.630988 IP 10.0.0.20 > A9: ICMP echo request, id 126, seq 9, length 64
14:51:37.631001 IP A9 > 10.0.0.20: ICMP echo reply, id 126, seq 9, length 64

28 packets captured
28 packets received by filter
0 packets dropped by kernel
root@n2:/tmp/pycore.48514/n2.conf#

```

Figura 19 - tcpdump no servidor n2

```
root@n3: /tmp/pycore.48514/n3.conf
root@n3:/tmp/pycore.48514/n3.conf# tcpdump
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), capture size 262144 bytes
^C14:51:29.631227 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.48514/n3.conf#
```

Figura 20 - tcpdump no servidor n3

```
root@n4: /tmp/pycore.48514/n4.conf
root@n4:/tmp/pycore.48514/n4.conf# tcpdump
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), capture size 262144 bytes
^C14:51:29.631229 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.48514/n4.conf#
```

Figura 21 - tcpdump no servidor n4

Nas figuras apresentadas anteriormente, é possível observar que ao fazer *ping* do laptop n1 para o servidor n2, os pacotes são todos entregues ao destino pretendido, que neste caso é o n2. Tal é possível, pois a mensagem é enviada para o *switch* que a envia diretamente para o *host*. Assim nas imagens referentes ao *tcpdump* dos servidores n3 e n4, verificamos que apenas capturam um pacote que corresponde a um pacote *arp-broadcast* enviado pelo switch para todas as interfaces da rede, de modo a conhecer o endereço MAC do *host* destino para o qual a mensagem deve ser enviada. Este procedimento apenas ocorre, quando este endereço ainda não se encontra na tabela de comutação do *switch*. Como os *hubs* transmitem a mensagem recebida por todas os nodos da rede, por apenas um canal de comunicação, as colisões são bastantes frequentes. Para além disto a quantidade de informação desnecessária a circular na rede é considerável. Enquanto que, os *switches* ao limitar o envio da mensagem apenas para o destino pretendido, reduzem a possibilidade de colisão. Assim, podemos concluir que os *switches* são mais viáveis do que os *hubs*.

2 Conclusões

A realização deste relatório permitiu aprofundar os conhecimentos acerca dos Endereços MAC, Address Resolution Protocol (ARP), Ethernet e Interligação de Redes Locais. Tal deve-se ao facto de termos efetuado capturas e as respetivas análises de tramas Ethernet recorrendo à ferramenta Wireshark, que foi essencial para obter todas as informações relativas às tramas, desde os protocolos utilizados até ao código de deteção de erros. Para uma melhor compreensão do mapeamento entre endereços do nível de rede e endereços do nível de ligação lógica foi fundamental o estudo do protocolo ARP.

A utilização da ferramenta de simulação de redes (CORE) proporcionou a criação de topologias que permitiram comparar a eficácia da utilização dos *switches* e *hubs* na diminuição de colisões em tramas Ethernet. Através do trabalho realizado conseguimos verificar que os *switches* são mais viáveis que os *hubs*.