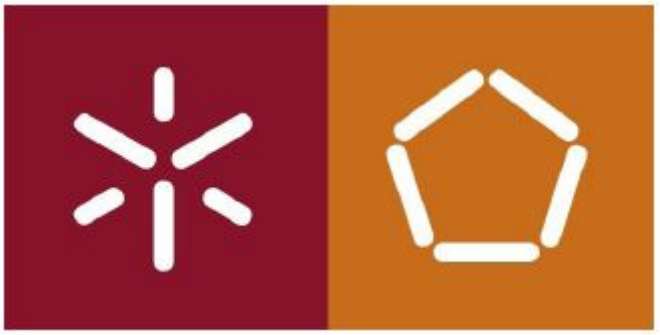
Mestrado Integrado em Engenharia de
Telecomunicações e Informática



Trabalho prático: LANs Ethernet e redes TCP/IP usando o CORE

David Alves, A79625
Raul Chamusca, A44043

2017/2018

Índice

1.Introdução

No contexto da unidade curricular de Rede de Computadores I, foi proposto a elaboração de um trabalho prático que consiste na implementação de várias redes locais e interligá-las entre si, utilizando um emulador de redes CORE (Common Open Research Emulator). Através desta ferramenta podemos desenhar e configurar várias topologias e executá-las com o objectivo de "imitar" a rede real. Utilizaremos também a ferramenta Wireshark para podermos visualizar e diagnosticar a conectividade entre as redes, e também proceder à captura do seu tráfego.

Este projecto tem também como objectivo, por em prática os conceitos e fundamentos que foram lecionados na unidade curricular ao longo do semestre bem como uma melhor compreensão dos protocolos de redes e a elas associados.

2.Emulação de LANs Ethernet

Neste primeiro exercício pretendia-se que se construísse e emulasse uma pequena rede local, recorrendo ao emulador de redes CORE, e depois com recurso a uma outra ferramenta, o *Wireshark,* observar os pacotes enviados e recebidos dentro da rede local criada, e perceber melhor o funcionamento dos protocolos ARP e ICMP. A topologia criada teria que ser a de uma rede local em árvore usando para isso dois HUB's e um SWITCH, tal como é possível observar na figura seguinte(esquema da rede local implementada).

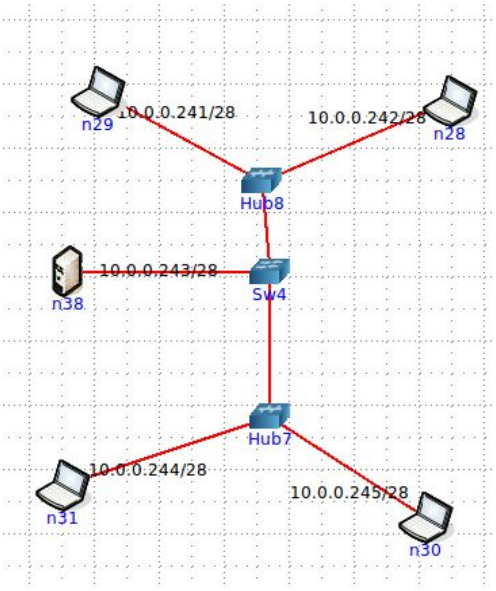


Fig.1 Topologia de rede local criada

O protocolo ARP, *Adress Resolution Protocol,* tem como principal função o mapeamento dos endereços de IP, utilizando para isso um ARP *request* e ARP *reply.*

O protocolo ICMP, *Internet Control Message Protocol,* é um protocolo utilizado pelos *routers*, *hosts*(terminais) e outros dispositivos de comunicação, para verificar a existência de erros, o que é fundamental para a implementação IP. Os dispositivos enviam mensagens ICMP á origem do pacote que recebeu, como por exemplo, tempo de transmissão do pacote excedido,mensagem de dispositivo ocupado ou, simplesmente, quando o dispositivo pode enviar o pacote por um caminho mais curto.

Para verificar a existência e funcionamento destes protocolos no nosso projeto recorremos a comandos ping e traceroute, e também ainda recorremos à ferramenta *Wireshark* para visualizá-los numa interface mais gráfica.

Em relação ao HUB e SWITCH sabe-se que:

-O HUB logo após receber dados numa dada porta,reencaminha esses dados para todas as outras portas, diminuindo assim a performance substancialmente, uma vez que será possível, em qualquer sistema terminal fora da ligação, ter acesso a informação transmitida no HUB, além disso, não consegue, nem tem funcionalidades, que lhe permitam armazenar informação dos hosts a ele conectadas.

- O SWITCH, pelo contrário, permite que logo após uma primeira ligação entre sistemas, fique registado o endereço MAC destes conectados a cada porta. Depois o SWITCH comuta o pacote diretamente para a porta de destino da mesma.

Para testar a conectividade da topologia, e a utilização dos protocolos ARP e ICMP, efectuámos um *ping request* da estação n29 para a n31 utilizando o comando ”ping 10.0.0.244”, tal como mostrado na seguinte figura.

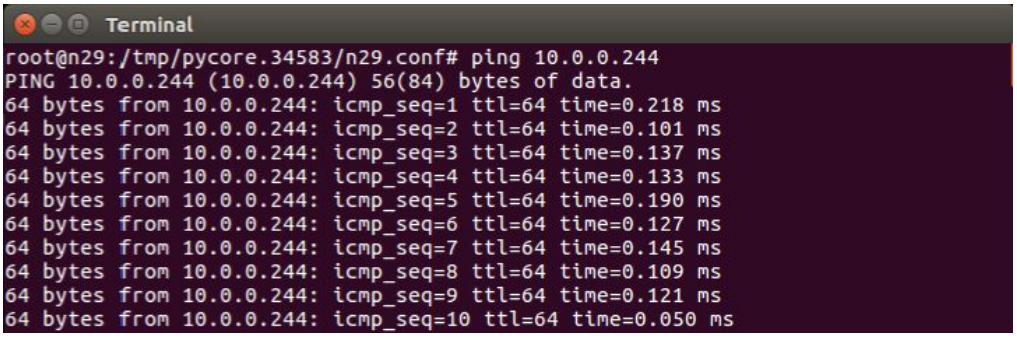
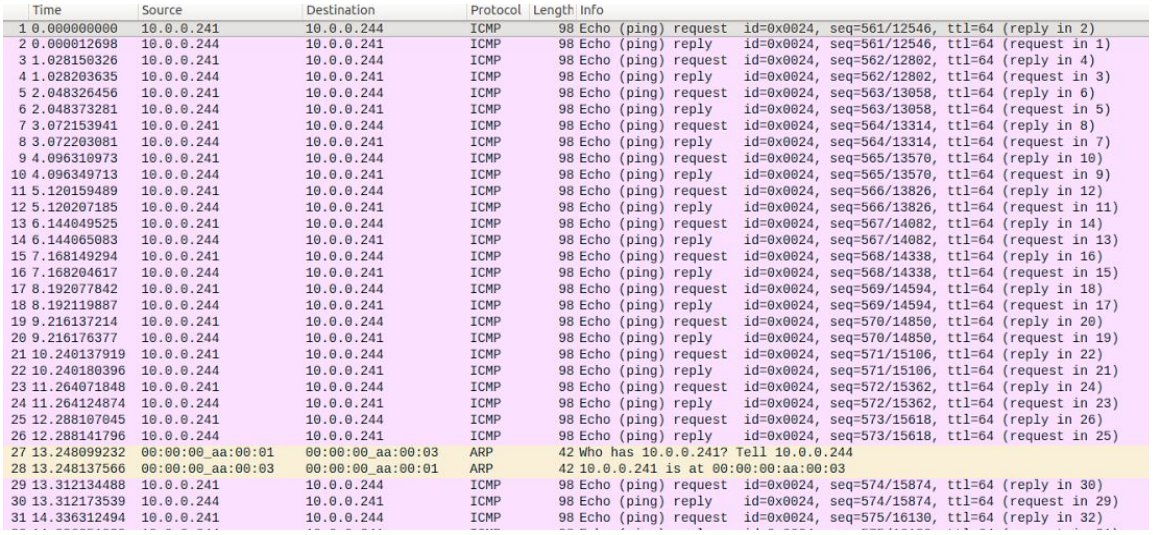


Fig.2- Comando ping 10.0.0.244

Como se pode observar o ping foi efectuado com sucesso e podemos ver que está a receber os pacotes de dados com sucesso do endereço 10.0.0.244. De seguida, para visualizarmos os pacotes enviados e recebidos e os protocolos utilizados com uma interface gráfica, abrimos na estação n31, o programa *Wireshark* , e verificámos a utilização dos protocolos ICMP e ARP, tal como se pode ver na seguinte figura.



	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	10.0.0.241	10.0.0.244	ICMP	98	Echo (ping) request id=0x0024, seq=561/12546, ttl=64 (reply in 2)
2	0.000012698	10.0.0.244	10.0.0.241	ICMP	98	Echo (ping) reply id=0x0024, seq=561/12546, ttl=64 (request in 1)
3	1.028150326	10.0.0.241	10.0.0.244	ICMP	98	Echo (ping) request id=0x0024, seq=562/12802, ttl=64 (reply in 4)
4	1.028203635	10.0.0.244	10.0.0.241	ICMP	98	Echo (ping) reply id=0x0024, seq=562/12802, ttl=64 (request in 3)
5	2.048326456	10.0.0.241	10.0.0.244	ICMP	98	Echo (ping) request id=0x0024, seq=563/13058, ttl=64 (reply in 6)
6	2.048373281	10.0.0.244	10.0.0.241	ICMP	98	Echo (ping) reply id=0x0024, seq=563/13058, ttl=64 (request in 5)
7	3.072153941	10.0.0.241	10.0.0.244	ICMP	98	Echo (ping) request id=0x0024, seq=564/13314, ttl=64 (reply in 8)
8	3.072203081	10.0.0.244	10.0.0.241	ICMP	98	Echo (ping) reply id=0x0024, seq=564/13314, ttl=64 (request in 7)
9	4.096310973	10.0.0.241	10.0.0.244	ICMP	98	Echo (ping) request id=0x0024, seq=565/13570, ttl=64 (reply in 10)
10	4.096349713	10.0.0.244	10.0.0.241	ICMP	98	Echo (ping) reply id=0x0024, seq=565/13570, ttl=64 (request in 9)
11	5.120159489	10.0.0.241	10.0.0.244	ICMP	98	Echo (ping) request id=0x0024, seq=566/13826, ttl=64 (reply in 12)
12	5.120207185	10.0.0.244	10.0.0.241	ICMP	98	Echo (ping) reply id=0x0024, seq=566/13826, ttl=64 (request in 11)
13	6.144049525	10.0.0.241	10.0.0.244	ICMP	98	Echo (ping) request id=0x0024, seq=567/14082, ttl=64 (reply in 14)
14	6.144065083	10.0.0.244	10.0.0.241	ICMP	98	Echo (ping) reply id=0x0024, seq=567/14082, ttl=64 (request in 13)
15	7.168149294	10.0.0.241	10.0.0.244	ICMP	98	Echo (ping) request id=0x0024, seq=568/14338, ttl=64 (reply in 16)
16	7.168204617	10.0.0.244	10.0.0.241	ICMP	98	Echo (ping) reply id=0x0024, seq=568/14338, ttl=64 (request in 15)
17	8.192077842	10.0.0.241	10.0.0.244	ICMP	98	Echo (ping) request id=0x0024, seq=569/14594, ttl=64 (reply in 18)
18	8.192119887	10.0.0.244	10.0.0.241	ICMP	98	Echo (ping) reply id=0x0024, seq=569/14594, ttl=64 (request in 17)
19	9.216137214	10.0.0.241	10.0.0.244	ICMP	98	Echo (ping) request id=0x0024, seq=570/14850, ttl=64 (reply in 20)
20	9.216176377	10.0.0.244	10.0.0.241	ICMP	98	Echo (ping) reply id=0x0024, seq=570/14850, ttl=64 (request in 19)
21	10.240137919	10.0.0.241	10.0.0.244	ICMP	98	Echo (ping) request id=0x0024, seq=571/15106, ttl=64 (reply in 22)
22	10.240180396	10.0.0.244	10.0.0.241	ICMP	98	Echo (ping) reply id=0x0024, seq=571/15106, ttl=64 (request in 21)
23	11.264071848	10.0.0.241	10.0.0.244	ICMP	98	Echo (ping) request id=0x0024, seq=572/15362, ttl=64 (reply in 24)
24	11.264124874	10.0.0.244	10.0.0.241	ICMP	98	Echo (ping) reply id=0x0024, seq=572/15362, ttl=64 (request in 23)
25	12.288107045	10.0.0.241	10.0.0.244	ICMP	98	Echo (ping) request id=0x0024, seq=573/15618, ttl=64 (reply in 26)
26	12.288141796	10.0.0.244	10.0.0.241	ICMP	98	Echo (ping) reply id=0x0024, seq=573/15618, ttl=64 (request in 25)
27	13.248099232	00:00:00_aa:00:01	00:00:00_aa:00:03	ARP	42	Who has 10.0.0.241? Tell 10.0.0.244
28	13.248137566	00:00:00_aa:00:03	00:00:00_aa:00:01	ARP	42	10.0.0.241 is at 00:00:00:aa:00:03
29	13.312134488	10.0.0.241	10.0.0.244	ICMP	98	Echo (ping) request id=0x0024, seq=574/15874, ttl=64 (reply in 30)
30	13.312173539	10.0.0.244	10.0.0.241	ICMP	98	Echo (ping) reply id=0x0024, seq=574/15874, ttl=64 (request in 29)
31	14.336312494	10.0.0.241	10.0.0.244	ICMP	98	Echo (ping) request id=0x0024, seq=575/16130, ttl=64 (reply in 32)

Fig.3-Captura da utilização dos protocolo ARP e ICMP no Wireshark

Como se pode ver, os pacotes ICMP de ping são efetuados ainda antes do *broadcast* (ARP) ser feito, uma vez que a tabela ARP já contém as informações de encaminhamento entre as estações.Também podemos observar que é efectuado um ping request do 10.0.0.241 para o 10.0.0.244 e depois logo consecutivamente um ping reply do 10.0.0.244 para o 10.0.0.241.

3.Interligação de redes

Depois de criar 4 topologias de redes locais tal como pedido, nesta fase era ainda necessário que se interligassem as 4 redes locais, para isso teria que se recorrer a *routers* (neste caso, 7 *routers*, sendo 4 ligados a cada rede local e 3 para comunicação entre *routers*) capazes de encaminhar o tráfego IP de umas redes para outras, tal como mostrado na fig.5. Para que se pudesse simular esse encaminhamento era necessário que se instalasse uma ferramenta chamada *quagga.*Os routers deveriam estar interligados por sub-redes com máscara de 30 bits, da rede 192.168.0.0/24 e quanto às redes locais, estas deveriam estar na gama de endereços 10.0.0.0/24, sendo que cada uma redes associada ao respetivo *router* tinha limites diferentes de utilização:

- R1:Suporta 150 computadores;
- R2:Suporta 40 computadores;
- R3:Suporta 20 computadores;
- R4:Suporta 10 computadores.

Para atribuir endereços IP dentro das redes locais, fizemos uma tabela de endereçamentos, de acordo com os limites para cada rede:

Nº de *Host's*	Endereço de Rede	Endereço de Difusão	Máscara de rede	Gama de endereços válidos DESDE | ATÉ	
121(Rede1)	10.0.0.0	10.0.0.127	255.255.255.128/25	10.0.0.1	10.0.0.126
27(Rede2)	10.0.0.128	10.0.0.159	255.255.255.224/27	10.0.0.129	10.0.0.158
25(Rede1)	10.0.0.160	10.0.0.191	255.255.255.224/27	10.0.0.161	10.0.0190
20 (Rede3)	10.0.0.192	10.0.0.223	255.255.255.224/27	10.0.0.193	10.0.0.222
13(Rede2)	10.0.0.224	10.0.0.239	255.255.255.240/28	10.0.0.225	10.0.0.238
10(Rede4)	10.0.0.240	10.0.0.255	255.255.255.240/28	10.0.0.241	10.0.0.254

Tabela 1- Tabela de encaminhamento IP das redes locais

Tal como se pode se pode observar na tabela acima, as redes 1 e 2 foram divididas, cada uma, em duas sub-redes(R1, dividida em sub-redes que suportam 121 estações e outra que suporta 25 estações, R2, dividida em sub-redes, uma que suporta 27 estações e outra que suporta 13 estações) isto aconteceu para que houvesse uma otimização dos endereços de IP disponíveis para atribuição. Ainda se acrescenta que o endereço que para ligação entre a rede local e o router é o mais alto endereço possível desse rede local (No caso das redes 1 e 2 os endereços são os mais alto endereço possivel de cada uma das subredes, tal como se pode ver nas figuras 6 e 7).

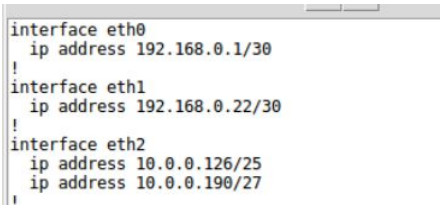


Fig.6-Interfaces ligadas ao router 1(pormenor da eth2-subredes)

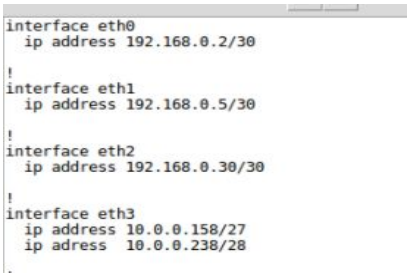


Fig.7-Interfaces ligadas ao router 2(pormenor da eth3-subredes)

Para o correto encaminhamento IP entre *router's* e as redes locais recorremos à interface gráfica do CORE colocando em cada *router* os "caminhos" a seguir para cada endereço de destino que lhe era pedido. Este passo também poderia ter sido usado recorrendo ao terminal de cada um dos *router´s* e adicionando caminho a caminho com o comando iproute, no entanto escolhemos a interface gráfica do CORE para minimizar o tempo na inserção da rotas/caminhos.
As rotas que colocamos em cada router foram as seguintes:

R1

```
!
ip route 10.0.0.240/28 192.168.0.21
ip route 10.0.0.128/27 192.168.0.2
ip route 10.0.0.224/28 192.168.0.2
ip route 10.0.0.192/27 192.168.0.21
```

Fig.8- *iproutes* do *router1*

R2

```
!
ip route 10.0.0.240/28 192.168.0.6
ip route 10.0.0.0/25 192.168.0.1
ip route 10.0.0.160/27 192.168.0.1
ip route 10.0.0.192/27 192.168.0.29
```

Fig.9-*iproutes* do *router2*

R3

```
!
ip route 10.0.0.0/25   192.168.0.17
ip route 10.0.0.160/27 192.168.0.17
ip route 10.0.0.128/27 192.168.0.26
ip route 10.0.0.224/28 192.168.0.26
ip route 10.0.0.240/28 192.168.0.14
```

Fig.10-*iproutes* do *router3*

R4

```
!
ip route 10.0.0.192/27 192.168.0.13
ip route 10.0.0.0/25   192.168.0.13
ip route 10.0.0.160/27 192.168.0.13
ip route 10.0.0.128/27 192.168.0.9
ip route 10.0.0.224/28 192.168.0.9
```

Fig.11-*iproutes* do *router4*

R5

```
!
ip route 10.0.0.192/27 192.168.0.18
ip route 10.0.0.0/25   192.168.0.22
ip route 10.0.0.160/27 192.168.0.22
ip route 10.0.0.128/27 192.168.0.22
ip route 10.0.0.240/28 192.168.0.18
ip route 10.0.0.224/28 192.168.0.22
```

Fig.12-*iproutes* do *router 5*

R6

```
ip route 10.0.0.192/27 192.168.0.25
ip route 10.0.0.128/27 192.168.0.30
ip route 10.0.0.224/28 192.168.0.30
ip route 10.0.0.0/25   192.168.0.30
ip route 10.0.0.160/27 192.168.0.30
ip route 10.0.0.240/28 192.168.0.25
```

Fig.13-*iproutes* do *router 6*

R7

```
!
ip route 10.0.0.192/27 192.168.0.10
ip route 10.0.0.240/28 192.168.0.10
ip route 10.0.0.128/27 192.168.0.5
ip route 10.0.0.224/28 192.168.0.5
ip route 10.0.0.0/25   192.168.0.5
ip route 10.0.0.160/27 192.168.0.5
```

Fig.14-*iproutes* do *router 7*

4.DHCP

Nas fases anteriormente apresentadas os endereços são definidos manualmente, recorrendo à uma tabela de encaminhamento IP. Neste exercício pretende-se que uma das redes locais criada anteriormente atribua automaticamente os endereços IP para cada novo host que se conecta à rede, para isso devemos usar o protocolo DHCP*(Dynamic Host Configuration Protocol*), configurando o servidor da rede local de modo a que cumpra esse protocolo.

Ao utilizar este protocolo existem vários factores que aumentam a eficiência da configuração e melhora a utilização das redes locais, tais como: a inserção de um novo endereço IP automaticamente cada vez que se quer adicionar um host à rede(o que pode antes podia levar a erros na inserção); Evita conflitos de endereços já em utilização por outra rede; Diminui o tempo usado na configuração e reconfiguração individual das redes.

A topologia da rede local usada(Rede 3) foi a seguinte:

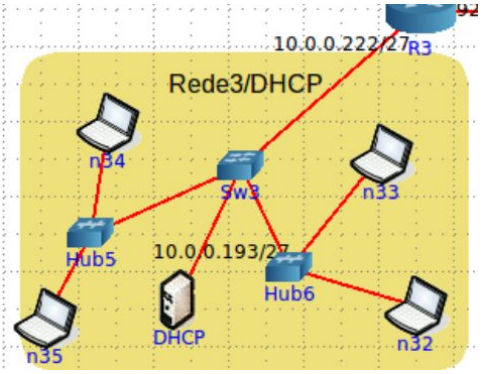


Fig.15-Topologia da rede local usada para o servidor DHCP

O servidor DHCP, na figura com o endereço 10.0.0.193/27, tem como principal objetivo atribuir dinamicamente os endereços ao hosts a ele ligados.Para isso usou-se seguinte configuração:

Default lease time (tempo base de atribuição do endereço) – 600 segundos
Max lease time (tempo máximo de atribuição do endereço) – 7200 segundos
Máscara da sub-rede – 255.255.255.224
Endereço de difusão-10.0.0.222
Gama de endereços – desde 10.0.0.194 até 10.0.0.221

```
# auto-generated by DHCP service (utility.py)
# NOTE: move these option lines into the desired pool { } block(s) below
#option domain-name "test.com";
#option domain-name-servers 10.0.0.1;
#option routers 10.0.0.1;

log-facility local6;

default-lease-time 600;
max-lease-time 7200;

ddns-update-style none;

subnet 10.0.0.192 netmask 255.255.255.224 {
  pool {
    range 10.0.0.194 10.0.0.221;
    default-lease-time 600;
    option routers 10.0.0.222;
  }
}
```

Fig.16-Configuração do router DHCP

Considerou-se n34 como DHCP client, de seguida executou-se num terminal dessa estação o comando ifconfig, tal como se pode ver na seguinte figura, e ficou-se a saber assim qual o endereço atribuído pelo servidor DHCP, neste caso o endereço obtido foi 10.0.0.200

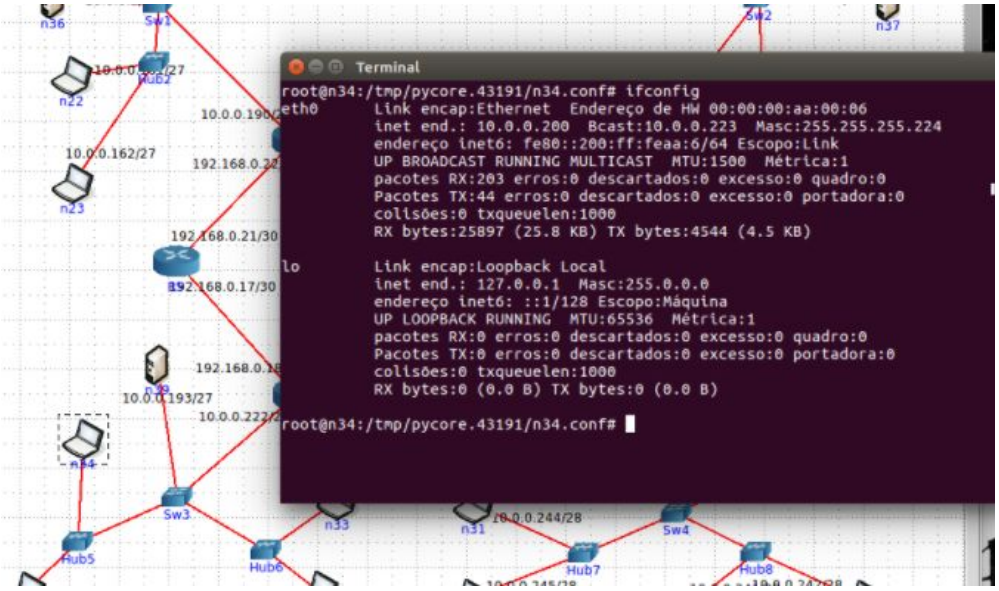


Fig.17-Visualização do endereço atribuído ao DHCP client

Depois, abriu-se um terminal de uma estação externa(n23) à rede local do servidor DHCP, e de seguida, fez-se um ping para o endereço que foi atribuido ao DHCP

client(10.0.0.200) e tal como se pode ver na seguinte figura obteve-se um transmissão de pacotes de dados com sucesso.

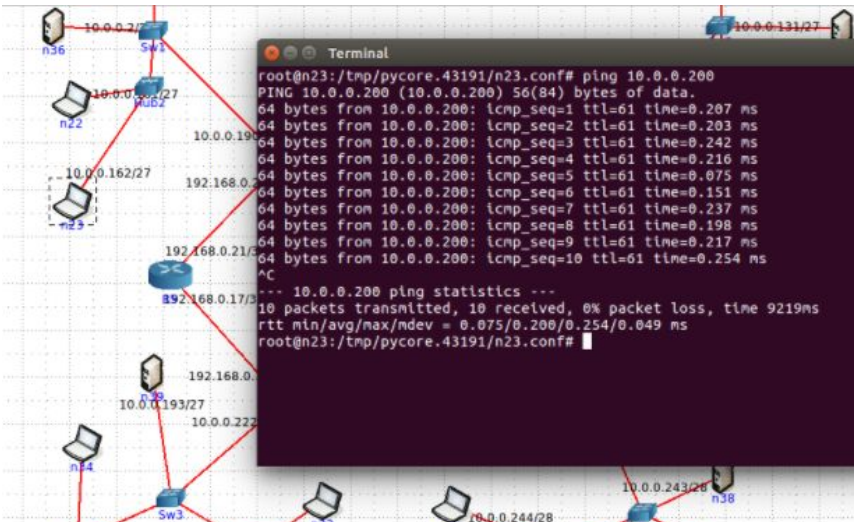


Fig.18-Ping para um DHCP client

5.Uso das camadas de rede e transporte por parte das aplicações

Tal como pedido, neste exercício escolhemos uma rede local da nossa topologia e inserimos um servidor FTP, (neste caso com o endereço 10.0.0.243, situado na rede 4) numa delas e em outra rede um servidor HTTP(neste caso com o endereço 10.0.0.131, situado na rede 2). Para configurar o servidor FTP usamos a interface gráfica do core ativando o serviço de FTP, de seguida abriu-se um terminal de uma estação externa à rede local desse servidor e executou-se comando ftp 10.0.0.243 e tal como se pode ver na seguinte figura, após a inserção do nome do utilizador e respetiva password e da execução do comando ls podemos observar o ficheiros presentes no nosso servidor ftp.

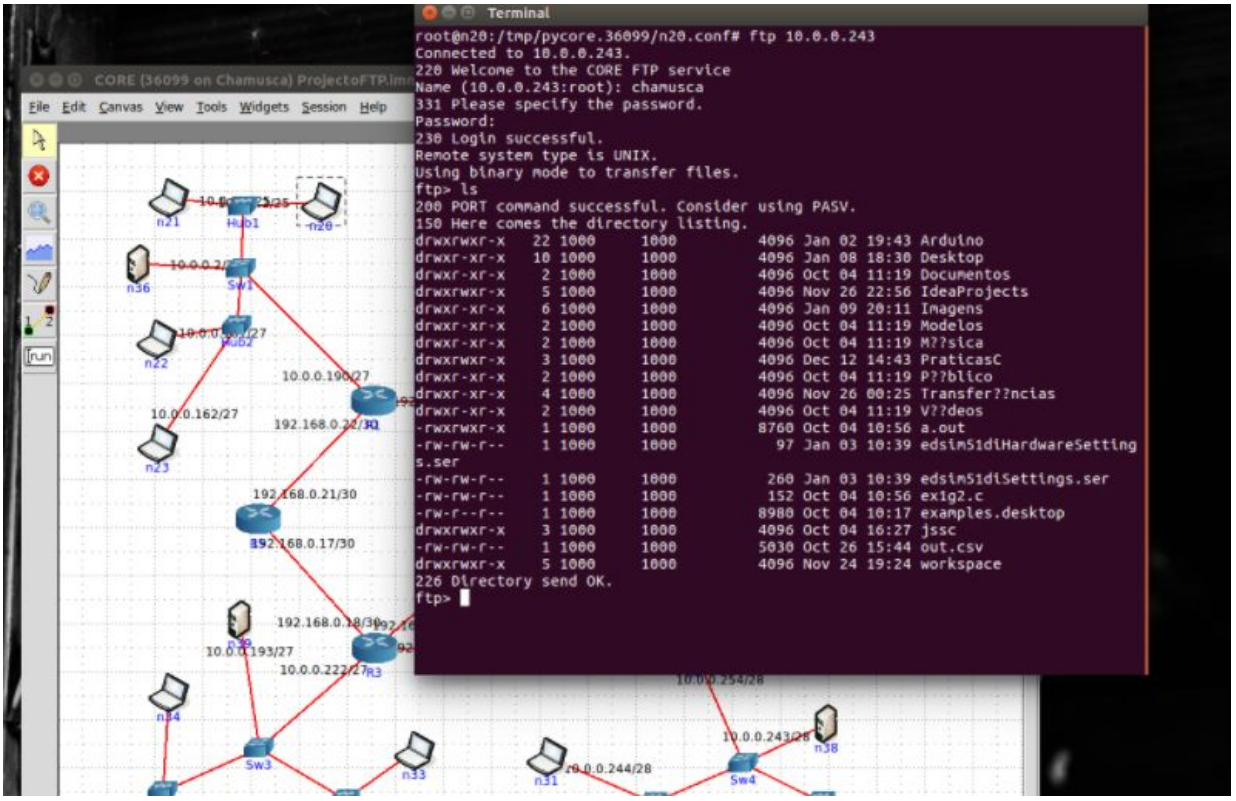


Fig.19-Funcionamento do servidor FTP(10.0.0.243)

Para configurar o servidor HTTP, ativamos o serviço de HTTP no servidor escolhido e de seguida abrimos um terminal de estação externa à rede do servidor (n20) e executamos o comando o “wget -S 10.0.0.131”, depois disso verificámos que foi transferido com sucesso para o PC onde estávamos a executar o comando um ficheiro de teste chamado “index.html”, o qual verificámos que se encontrava presente nesse PC executando o comando ls, tal como se pode ver na seguinte figura.

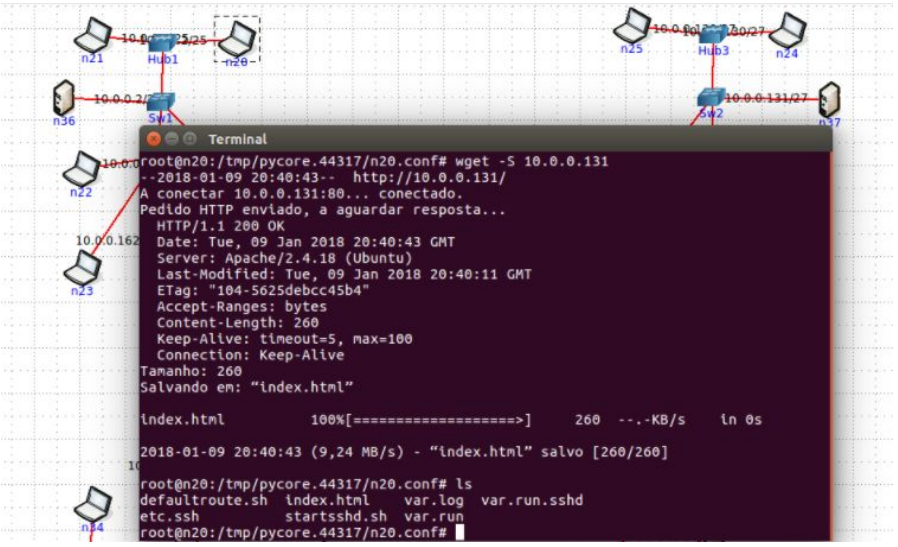


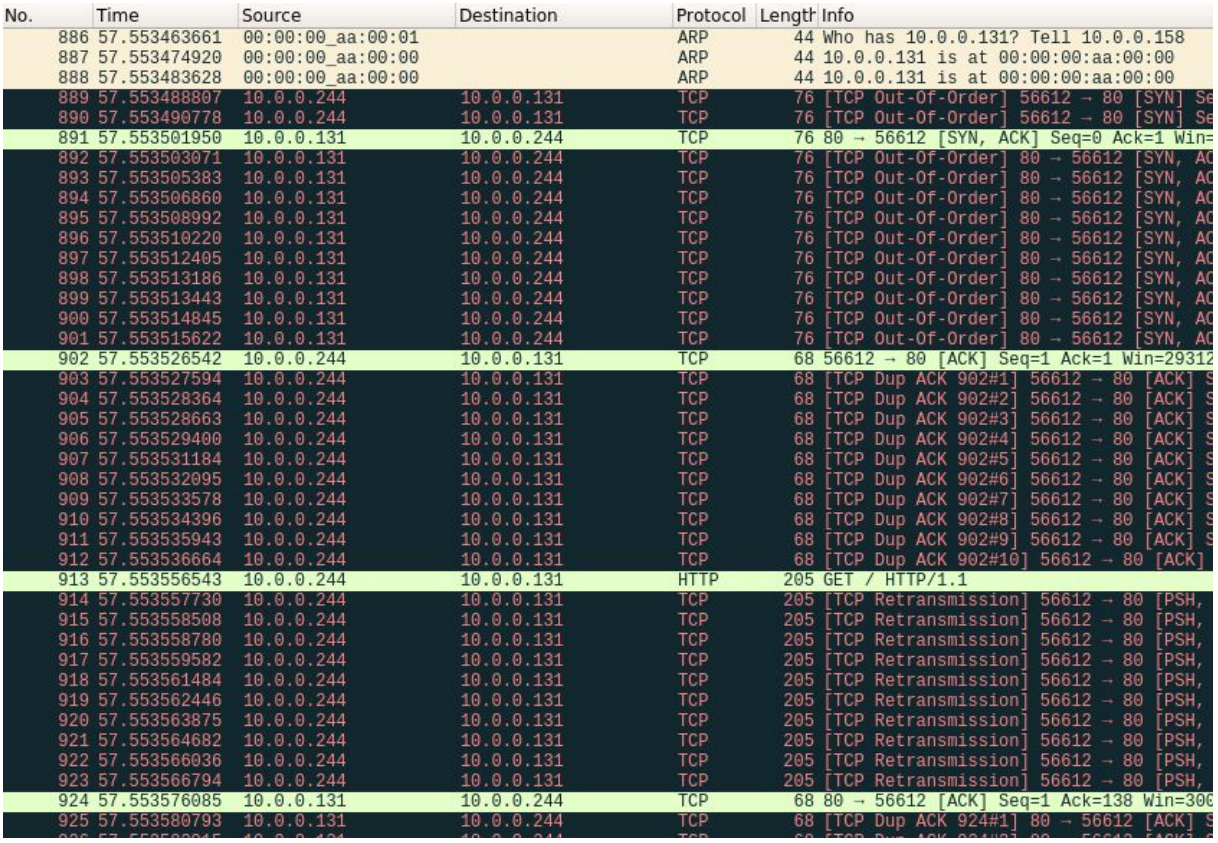
Fig.20-Funcionamento do servidor HTTP(10.0.0.131)

Recorrendo ao Wireshark para o servidor ftp abrimos o terminal de uma estação da rede 2 (10.0.0.226) executamos o comando ftp 10.0.0.243 e de seguida fomos ao wireshark e observamos o correto funcionamento dos protocolos TCP e FTP, tal como visto na amostra representada na seguinte figura, onde podemos ver um ACK,ou seja a confirmação de envio com sucesso de informação.

No.	Time	Source	Destination	Protocol	Length	Info
492	41.291700077	10.0.0.226	10.0.0.243	TCP	86	[TCP Retransmission] 56680 → 21 [PSH, ACK] Seq=1 Ack…
527	41.697186601	10.0.0.226	10.0.0.243	TCP	68	56680 → 21 [ACK] Seq=19 Ack=24 Win=229 Len=0 TSval=2…
528	41.697192515	10.0.0.226	10.0.0.243	TCP	68	[TCP Dup ACK 527#1] 56680 → 21 [ACK] Seq=19 Ack=24 W…
529	41.697194346	10.0.0.226	10.0.0.243	TCP	68	[TCP Dup ACK 527#2] 56680 → 21 [ACK] Seq=19 Ack=24 W…
530	41.697194879	10.0.0.226	10.0.0.243	TCP	68	[TCP Dup ACK 527#3] 56680 → 21 [ACK] Seq=19 Ack=24 W…
531	41.697197418	10.0.0.226	10.0.0.243	TCP	68	[TCP Dup ACK 527#4] 56680 → 21 [ACK] Seq=19 Ack=24 W…
532	41.697203670	10.0.0.226	10.0.0.243	TCP	68	[TCP Dup ACK 527#5] 56680 → 21 [ACK] Seq=19 Ack=24 W…
533	41.697205805	10.0.0.226	10.0.0.243	TCP	68	[TCP Dup ACK 527#6] 56680 → 21 [ACK] Seq=19 Ack=24 W…
534	41.697209834	10.0.0.226	10.0.0.243	TCP	68	[TCP Dup ACK 527#7] 56680 → 21 [ACK] Seq=19 Ack=24 W…
535	41.697212049	10.0.0.226	10.0.0.243	TCP	68	[TCP Dup ACK 527#8] 56680 → 21 [ACK] Seq=19 Ack=24 W…
536	41.697216039	10.0.0.226	10.0.0.243	TCP	68	[TCP Dup ACK 527#9] 56680 → 21 [ACK] Seq=19 Ack=24 W…
537	41.697218013	10.0.0.226	10.0.0.243	TCP	68	[TCP Dup ACK 527#10] 56680 → 21 [ACK] Seq=19 Ack=24 …
538	41.697255736	10.0.0.226	10.0.0.243	FTP	74	Request: SYST
539	41.697257559	10.0.0.226	10.0.0.243	FTP	74	[TCP Fast Retransmission] Request: SYST
540	41.697258766	10.0.0.226	10.0.0.243	FTP	74	[TCP Fast Retransmission] Request: SYST

Fig.21-Captura do Wireshark do servidor ftp

Recorrendo ao Wireshark para o servidor HTTP abrimos o terminal de uma estação da rede4(10.0.0.244), executamos o comando wget -S 10.0.0.131, e de seguida abrimos o wireshark para ver o funcionamento dos protocolos HTTP e TCP, tal como é possível na seguinte captura do wireshark.



No.	Time	Source	Destination	Protocol	Length	Info
886	57.553463661	00:00:00_aa:00:01		ARP	44	Who has 10.0.0.131? Tell 10.0.0.158
887	57.553474920	00:00:00_aa:00:00		ARP	44	10.0.0.131 is at 00:00:00:aa:00:00
888	57.553483628	00:00:00_aa:00:00		ARP	44	10.0.0.131 is at 00:00:00:aa:00:00
889	57.553488807	10.0.0.244	10.0.0.131	TCP	76	[TCP Out-Of-Order] 56612 → 80 [SYN] Se
890	57.553490778	10.0.0.244	10.0.0.131	TCP	76	[TCP Out-Of-Order] 56612 → 80 [SYN] Se
891	57.553501950	10.0.0.131	10.0.0.244	TCP	76	80 → 56612 [SYN, ACK] Seq=0 Ack=1 Win=
892	57.553503071	10.0.0.131	10.0.0.244	TCP	76	[TCP Out-Of-Order] 80 → 56612 [SYN, AC
893	57.553505383	10.0.0.131	10.0.0.244	TCP	76	[TCP Out-Of-Order] 80 → 56612 [SYN, AC
894	57.553506860	10.0.0.131	10.0.0.244	TCP	76	[TCP Out-Of-Order] 80 → 56612 [SYN, AC
895	57.553508992	10.0.0.131	10.0.0.244	TCP	76	[TCP Out-Of-Order] 80 → 56612 [SYN, AC
896	57.553510220	10.0.0.131	10.0.0.244	TCP	76	[TCP Out-Of-Order] 80 → 56612 [SYN, AC
897	57.553512405	10.0.0.131	10.0.0.244	TCP	76	[TCP Out-Of-Order] 80 → 56612 [SYN, AC
898	57.553513186	10.0.0.131	10.0.0.244	TCP	76	[TCP Out-Of-Order] 80 → 56612 [SYN, AC
899	57.553513443	10.0.0.131	10.0.0.244	TCP	76	[TCP Out-Of-Order] 80 → 56612 [SYN, AC
900	57.553514845	10.0.0.131	10.0.0.244	TCP	76	[TCP Out-Of-Order] 80 → 56612 [SYN, AC
901	57.553515622	10.0.0.131	10.0.0.244	TCP	76	[TCP Out-Of-Order] 80 → 56612 [SYN, AC
902	57.553526542	10.0.0.244	10.0.0.131	TCP	68	56612 → 80 [ACK] Seq=1 Ack=1 Win=29312
903	57.553527594	10.0.0.244	10.0.0.131	TCP	68	[TCP Dup ACK 902#1] 56612 → 80 [ACK] S
904	57.553528364	10.0.0.244	10.0.0.131	TCP	68	[TCP Dup ACK 902#2] 56612 → 80 [ACK] S
905	57.553528663	10.0.0.244	10.0.0.131	TCP	68	[TCP Dup ACK 902#3] 56612 → 80 [ACK] S
906	57.553529400	10.0.0.244	10.0.0.131	TCP	68	[TCP Dup ACK 902#4] 56612 → 80 [ACK] S
907	57.553531184	10.0.0.244	10.0.0.131	TCP	68	[TCP Dup ACK 902#5] 56612 → 80 [ACK] S
908	57.553532095	10.0.0.244	10.0.0.131	TCP	68	[TCP Dup ACK 902#6] 56612 → 80 [ACK] S
909	57.553533578	10.0.0.244	10.0.0.131	TCP	68	[TCP Dup ACK 902#7] 56612 → 80 [ACK] S
910	57.553534396	10.0.0.244	10.0.0.131	TCP	68	[TCP Dup ACK 902#8] 56612 → 80 [ACK] S
911	57.553535943	10.0.0.244	10.0.0.131	TCP	68	[TCP Dup ACK 902#9] 56612 → 80 [ACK] S
912	57.553536664	10.0.0.244	10.0.0.131	TCP	68	[TCP Dup ACK 902#10] 56612 → 80 [ACK]
913	57.553556543	10.0.0.244	10.0.0.131	HTTP	205	GET / HTTP/1.1
914	57.553557730	10.0.0.244	10.0.0.131	TCP	205	[TCP Retransmission] 56612 → 80 [PSH,
915	57.553558508	10.0.0.244	10.0.0.131	TCP	205	[TCP Retransmission] 56612 → 80 [PSH,
916	57.553558780	10.0.0.244	10.0.0.131	TCP	205	[TCP Retransmission] 56612 → 80 [PSH,
917	57.553559582	10.0.0.244	10.0.0.131	TCP	205	[TCP Retransmission] 56612 → 80 [PSH,
918	57.553561484	10.0.0.244	10.0.0.131	TCP	205	[TCP Retransmission] 56612 → 80 [PSH,
919	57.553562446	10.0.0.244	10.0.0.131	TCP	205	[TCP Retransmission] 56612 → 80 [PSH,
920	57.553563875	10.0.0.244	10.0.0.131	TCP	205	[TCP Retransmission] 56612 → 80 [PSH,
921	57.553564682	10.0.0.244	10.0.0.131	TCP	205	[TCP Retransmission] 56612 → 80 [PSH,
922	57.553566036	10.0.0.244	10.0.0.131	TCP	205	[TCP Retransmission] 56612 → 80 [PSH,
923	57.553566794	10.0.0.244	10.0.0.131	TCP	205	[TCP Retransmission] 56612 → 80 [PSH,
924	57.553576085	10.0.0.131	10.0.0.244	TCP	68	80 → 56612 [ACK] Seq=1 Ack=138 Win=300
925	57.553580793	10.0.0.131	10.0.0.244	TCP	68	[TCP Dup ACK 924#1] 80 → 56612 [ACK] S

Fig.22- Captura do Wireshark do servidor HTTP

6.Interligação via NAT (Network Address Translator)

Nesta fase, era pedido que se criasse uma rede privada dentro de uma das redes que criámos, essa rede teria que ter endereços na gama de 192.168.1.0/24, para atribuirmos os endereços às respetivas estações usamos o mesmo tipo de procedimento usado para a determinação dos endereços das redes locais, sendo que neste caso é nos dado a máscara e não o número de hosts da rede.Assim sendo a rede NAT criada ficou associada à rede 4 e tem a seguinte topologia.

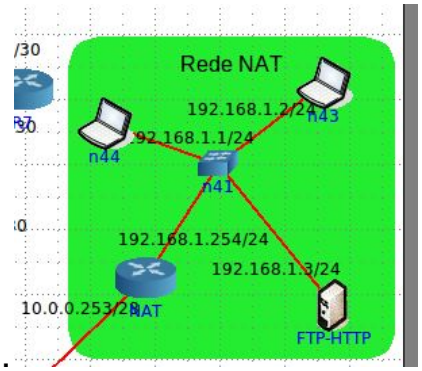


Fig.23-Topologia da Rede NAT

Para configurar o nosso router NAT, utilizamos a interface gráfica do CORE, e ativamos o serviço Firewall, configurando esse serviço com os seguintes comandos iptables.

Fig.24-Configuração do router NAT

Fundamentalmente, esta rede NAT é uma rede que não está acessível aos outros utilizadores de redes externas, para comprovar que isso acontece e assim comprovar que a configuração do nosso router NAT era a correta, abrimos um terminal de uma estação externa à rede NAT (n20), e executamos o comando ping 192.168.1.1 e tal como previsto não é possível estabelecer uma conexão de dados entre uma estação externa e a rede NAT, tal como se pode ver na seguinte figura.

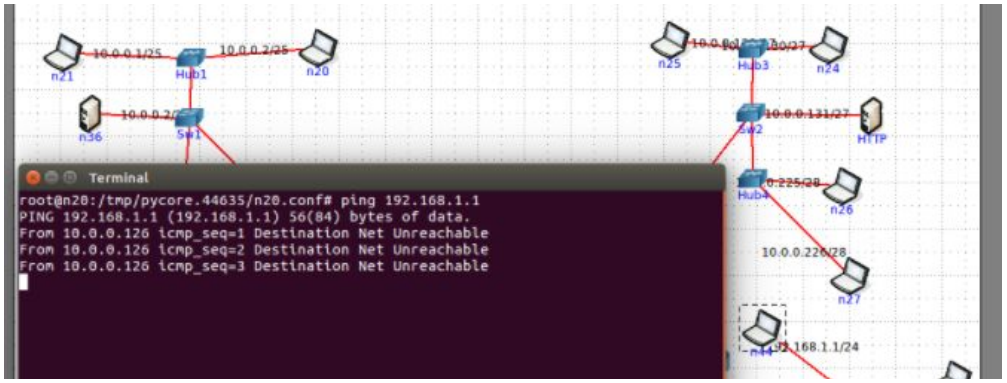


Fig.25-Conexão entre rede externa e rede NAT

No entanto se o ping for efectuado dentro da rede NAT, aí sim, tal como previsto, já é possível efectuar uma conexão de dados com êxito, tal como se pode ver na seguinte figura, na qual abrimos um terminal numa estação dentro da rede NAT(n44) e efectuamos um ping entre duas estações da rede NAT, e verificamos que se efectuou com sucesso uma conexão de dados.

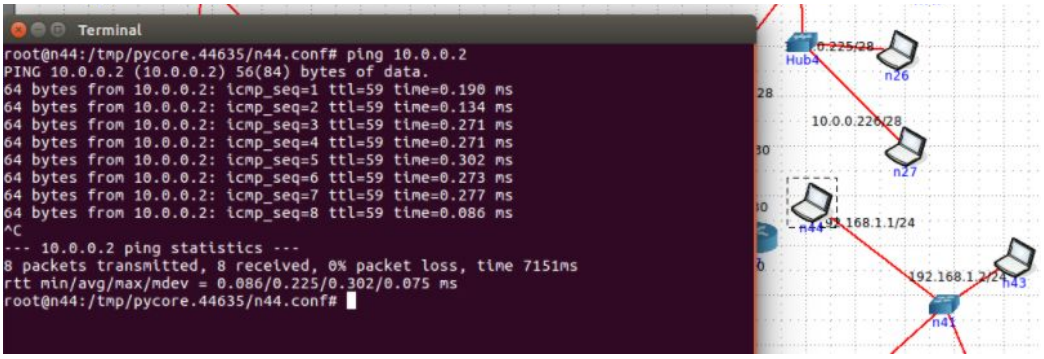


Fig.26- Conexão entre estações interna da rede NAT

De seguida ainda era pedido que criássemos um servidor HTTP e FTP dentro da rede NAT. Para isso usamos o mesmo tipo de configuração que usamos aquando da inserção dos servidores FTP e HTTP nas redes 4 e 2. De seguida tal como previsto, verificámos que foi possível conectar aos servidores com êxito dentro e fora da rede NAT.

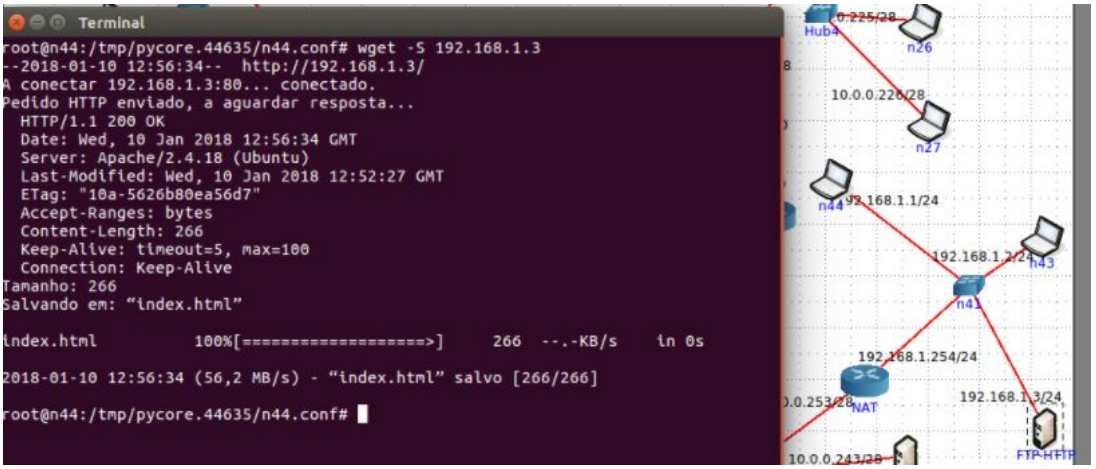


Fig.27-Conexão entre estação e servidor HTTP na rede NAT

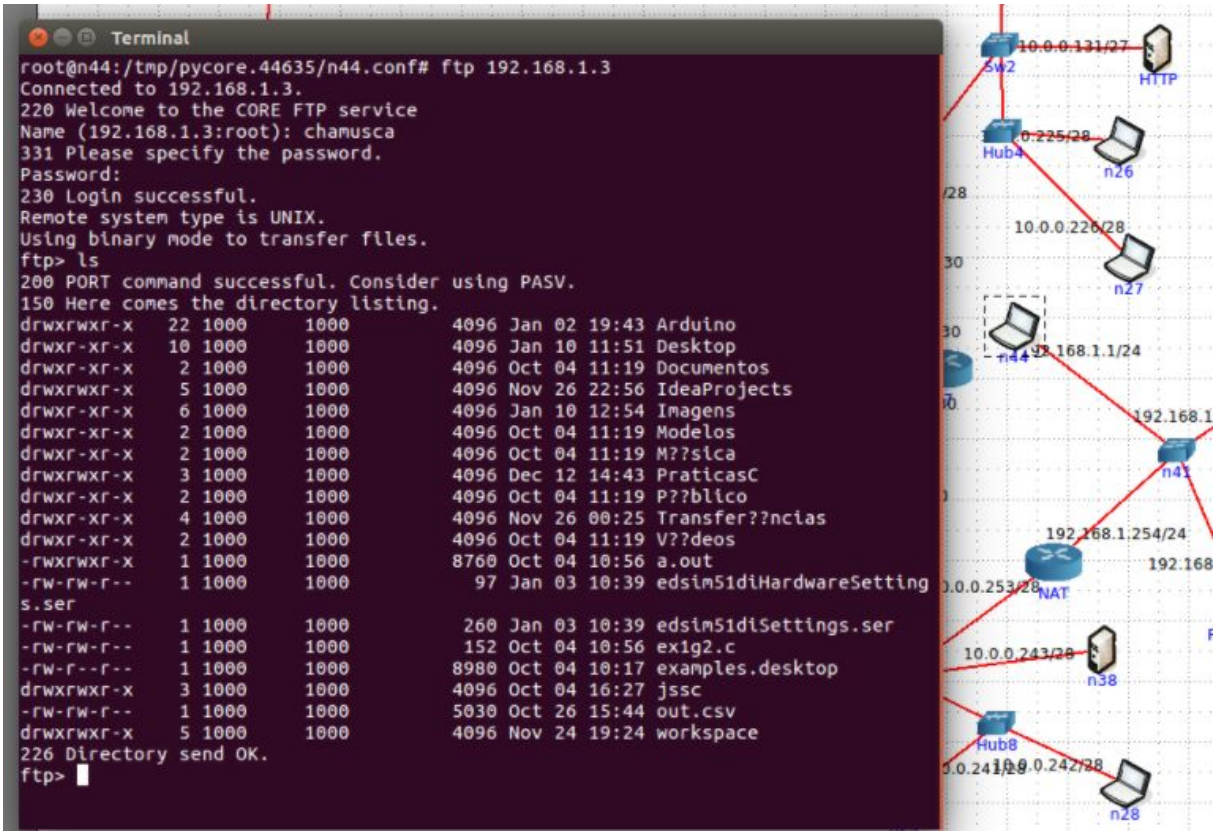


Fig.28-Conexão entre estação e servidor FTP na rede NAT

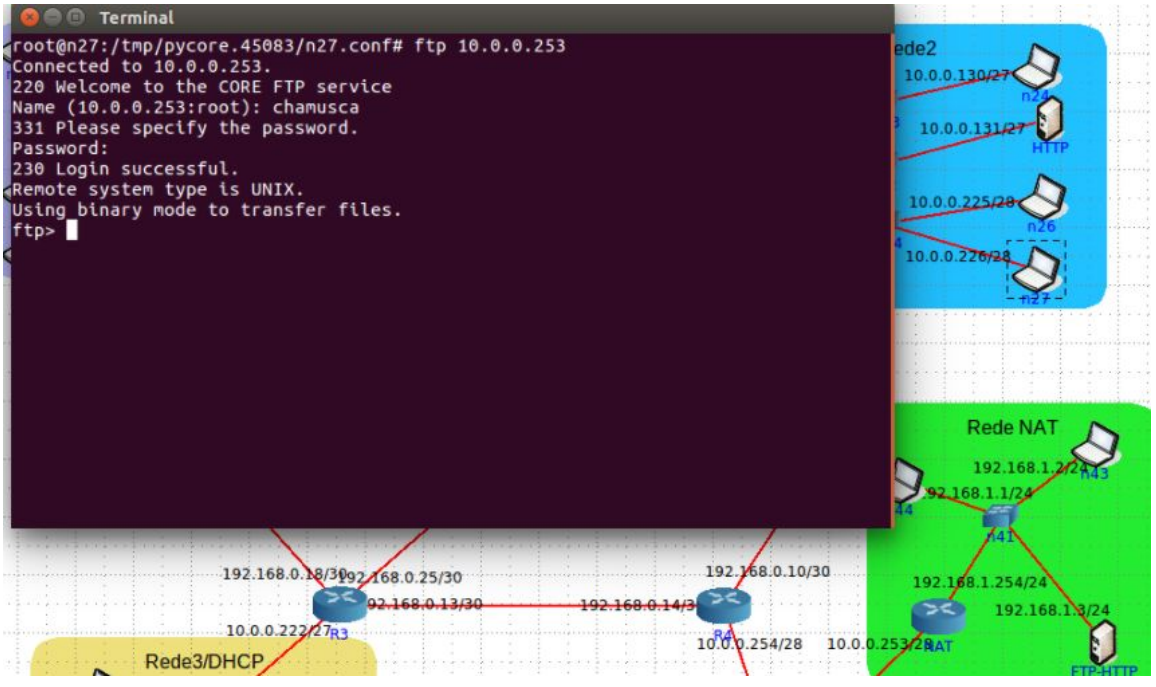


Fig.29-Conexão entre estação externa e servidor FTP NAT

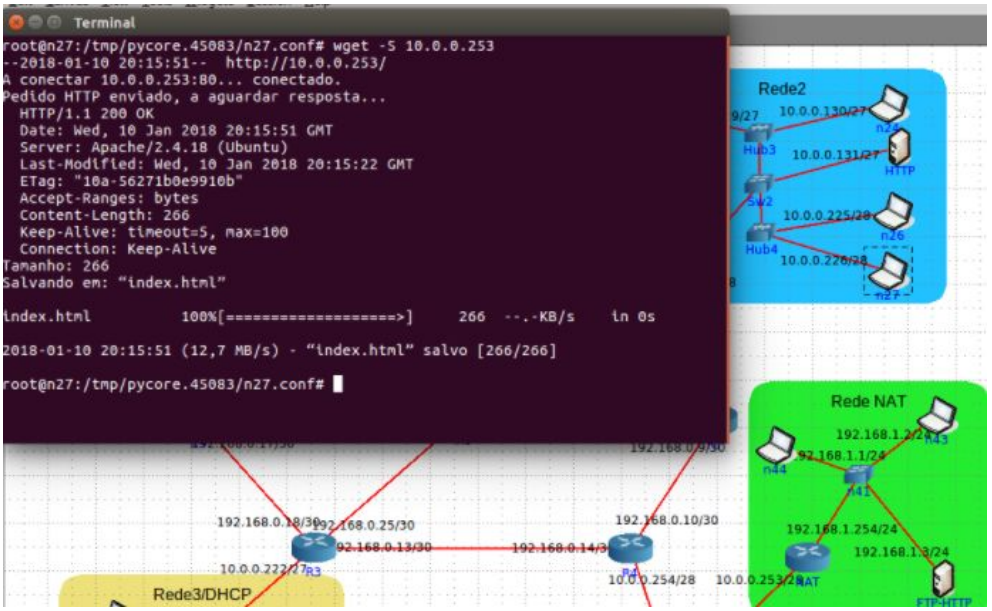


Fig.30-Conexão entre estação externa e servidor HTTP NAT

Para que isto acontecesse alterámos no router NAT dentro do serviço Firewall para que cada vez que haja uma chamada do servidor ftp ou http com o endereço do router NAT este reencaminhe para o servidor FTP/HTTP existente na rede NAT, como se pode ver na seguinte figura.

```
iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE
iptables -A FOWARD -i eth1 -j ACCEPT

#HTTP
iptables -t nat -A PREROUTING -p tcp --dport 80 -j DNAT --to-destination 192.168.1.3
iptables -t nat -A POSTROUTING -p tcp -d 192.168.1.3 --dport 80 -j SNAT --to-source 10.0.0.254

#FTP
iptables -t nat -A PREROUTING -p tcp --dport 21 -j DNAT --to-destination 192.168.1.3
iptables -t nat -A POSTROUTING -p tcp -d 192.168.1.3 --dport 21 -j SNAT --to-source 10.0.0.254
```

Fig.31-Configuração Firewall router NAT para servidor HTTP/FTP

Ainda se acrescenta que no router NAT foi configurado de modo a que se conecte por defeito ao router da rede 4, usando para isso o mesmo tipo de configuração defaultRoute que se usou para as estações da rede 4, tal como se pode ver na seguinte figura.

```
#!/bin/sh
# auto-generated by DefaultRoute service (utility.py)
ip route add default via 10.0.0.254
```

Fig.32-Configuração defaultroute no router NAT

Para finalizarmos, mostramos agora na seguinte figura a nossa topologia final.

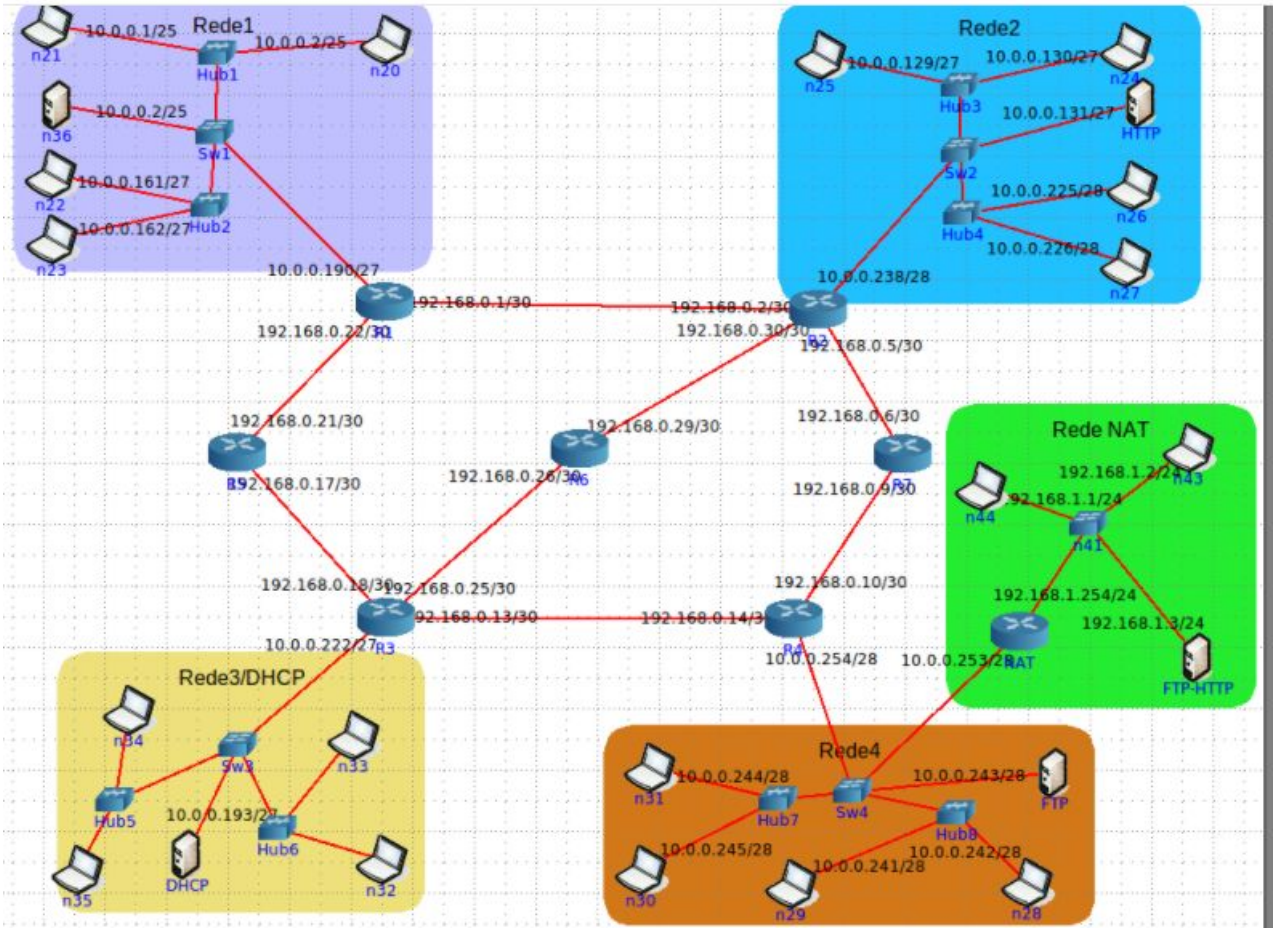


Fig.33-Topologia final

7.Conclusão

Concluído o trabalho prático, e refletindo sobre os resultados obtidos, podemos afirmar que conseguimos verificar e executar todas as fases que nos foram propostas, isto é, todos os conceitos e procedimentos que foram lecionados nas aulas teóricas foram aplicados com sucesso no projecto prático. Este projecto levou a uma melhor perceção sobre o funcionamento de uma rede, estabelecimento de conexões entre duas estações, tabelas de encaminhamento, atribuição de endereços de IP, funcionamento protocolo NAT, DHCP, FTP e HTTP.

A maior dificuldade com que nos deparámos em relação ao projecto foi, na visualização do tráfego e dos protocolos utilizados na conexão entre as várias estações, através da ferramenta *wireshark*, devido a um problemas nas permissões do próprio programa.

Contudo, podemos concluir que o grupo conseguiu com sucesso atingir os objectivos propostos e aprimorar os conhecimentos obtidos na unidade curricular.