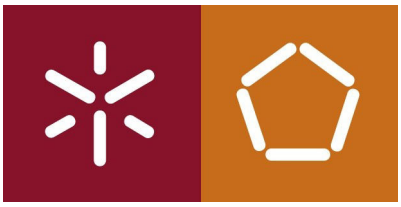


Mestrado Integrado em Engenharia de
Telecomunicações e Informática

Trabalho Prático

Redes de Computadores I

LANs Ethernet e redes TCP/IP usando o CORE

Gilberto Gomes Morim, 65214
Tiago Dourado, 68459

2016/2017

Índice

Índice.....2

1. **Introdução**.....3
2. **Emulação de LANs Ethernet**.....4
3. **DHCP**.....8
4. **Interligação de redes**.....11
5. **Uso das camadas de rede e transporte por parte das aplicações**.....14
6. **Interligação via NAT (Network Address Translator)**.....15
7. **Conclusão**.....17

1. Introdução

No âmbito da unidade curricular de Redes de Computadores I, foi proposta a elaboração de um projeto prático que consiste na implementação de vários tipos de rede e interliga-las entre si utilizando o emulador de redes CORE. Esta ferramenta será utilizada para desenhar as topologias e configurar links e endereços, sendo feita em modo de execução com o objetivo de imitar ao máximo a rede real. Depois de da criação e configuração das topologias será utilizada a ferramenta Wireshark para o diagnóstico de conetividade e análise de capturas de tráfego.

Este projeto tem por objetivo uma melhor aprendizagem de interligação e configuração de máquinas bem como uma melhor compreensão dos protocolos de rede a elas associados.

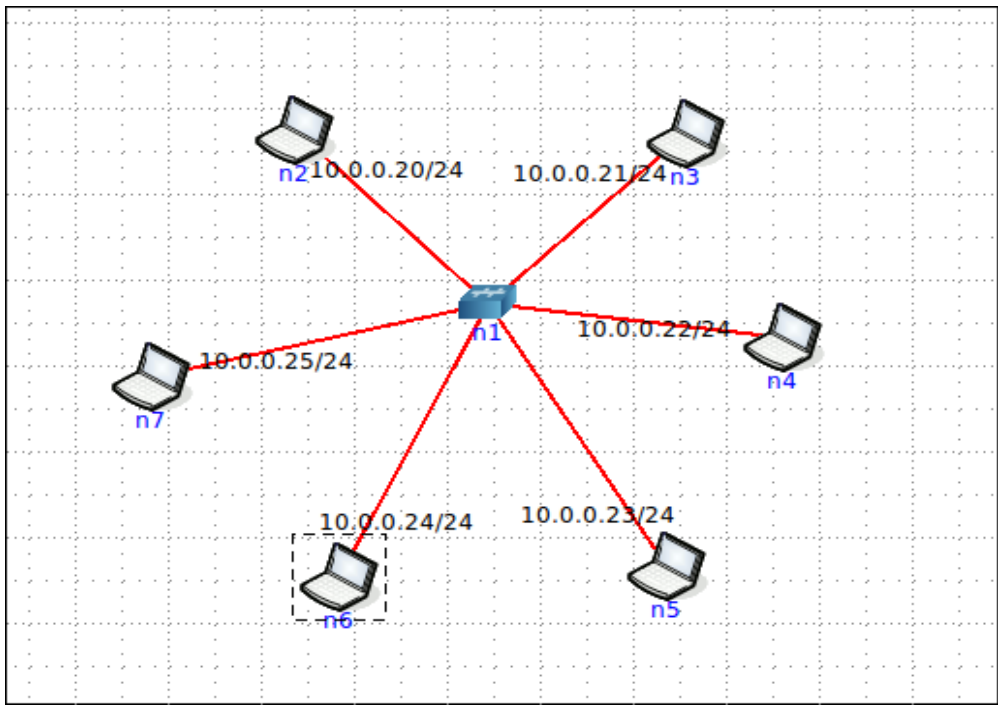
2. Emulação de LANs Ethernet

No primeiro exercício deste enunciado, o objetivo é, após a construção de uma topologia com elementos ligados em estrela ou em árvore, analisar o comportamento dos protocolos ARP e ICMP na ligação.

O primeiro protocolo que iremos analisar, o protocolo ARP, *Address Resolution Protocol,* tem como objetivo mapear um endereço de rede para um endereço físico – o endereço MAC. O segundo protocolo a ser analisado é o protocolo ICMP, *Internet Control Message Protocol,* - que é um protocolo integrante do IP – tem como função enviar relatórios de erros à fonte original, em situações de perdas de pacotes, congestionamento. Qualquer computador que implemente o protocolo IP deve estar preparado para receber os relatórios dos erros e agir adequadamente.

Topologia com HUB

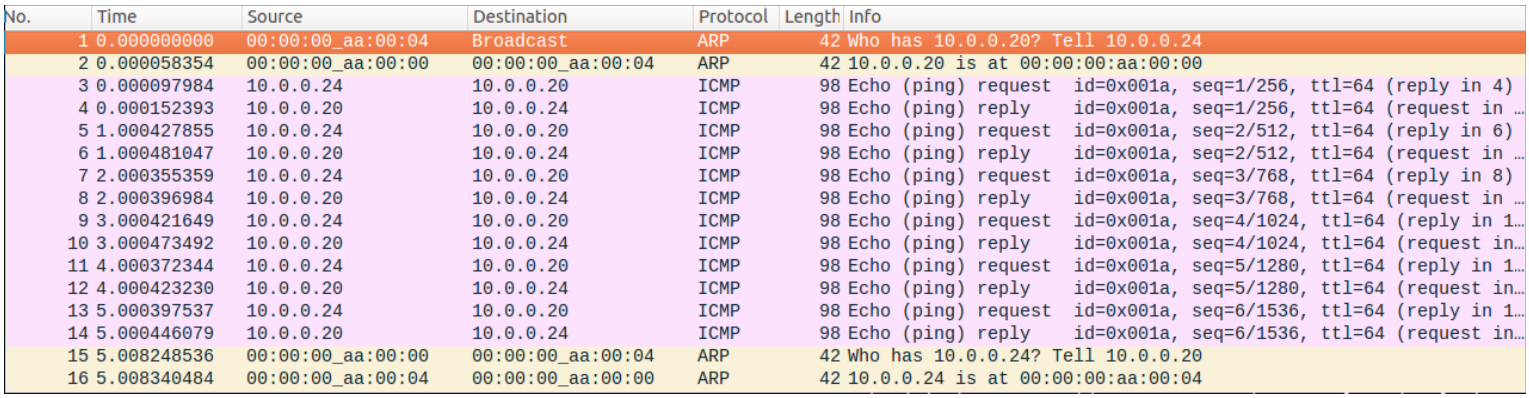
A imagem abaixo representa a topologia em estrela utilizada:



As máquinas foram ligadas utilizando um HUB, dispositivo que ao receber informação, faz um *broadcast* para todas as estações ligadas, possibilitando a estações que não fazem parte da interação terem acesso à informação transmitida. Como a informação é difundida, o desempenho da rede é diminuído e há maior risco de colisões. O HUB não possui tecnologia que lhe permita armazenar informação sobre as estações ligadas.

De modo a testar a conectividade da topologia e inspecionar a interação dos protocolos, foi efetuado um *ping request* da estação **n6** para a **n2**, utilizando o comando "ping 10.0.0.20". Foi conectado à estação **n2** o programa *Wireshark*, que permite fazer a inspeção de tramas, para observarmos a interação.

O resultado obtido no *Wireshark* foi o seguinte:



No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	00:00:00_aa:00:04	Broadcast	ARP	42	Who has 10.0.0.20? Tell 10.0.0.24
2	0.000058354	00:00:00_aa:00:00	00:00:00_aa:00:04	ARP	42	10.0.0.20 is at 00:00:00:aa:00:00
3	0.000097984	10.0.0.24	10.0.0.20	ICMP	98	Echo (ping) request id=0x001a, seq=1/256, ttl=64 (reply in 4)
4	0.000152393	10.0.0.20	10.0.0.24	ICMP	98	Echo (ping) reply id=0x001a, seq=1/256, ttl=64 (request in …
5	1.000427855	10.0.0.24	10.0.0.20	ICMP	98	Echo (ping) request id=0x001a, seq=2/512, ttl=64 (reply in 6)
6	1.000481047	10.0.0.20	10.0.0.24	ICMP	98	Echo (ping) reply id=0x001a, seq=2/512, ttl=64 (request in …
7	2.000355359	10.0.0.24	10.0.0.20	ICMP	98	Echo (ping) request id=0x001a, seq=3/768, ttl=64 (reply in 8)
8	2.000396984	10.0.0.20	10.0.0.24	ICMP	98	Echo (ping) reply id=0x001a, seq=3/768, ttl=64 (request in …
9	3.000421649	10.0.0.24	10.0.0.20	ICMP	98	Echo (ping) request id=0x001a, seq=4/1024, ttl=64 (reply in 1…
10	3.000473492	10.0.0.20	10.0.0.24	ICMP	98	Echo (ping) reply id=0x001a, seq=4/1024, ttl=64 (request in…
11	4.000372344	10.0.0.24	10.0.0.20	ICMP	98	Echo (ping) request id=0x001a, seq=5/1280, ttl=64 (reply in 1…
12	4.000423230	10.0.0.20	10.0.0.24	ICMP	98	Echo (ping) reply id=0x001a, seq=5/1280, ttl=64 (request in…
13	5.000397537	10.0.0.24	10.0.0.20	ICMP	98	Echo (ping) request id=0x001a, seq=6/1536, ttl=64 (reply in 1…
14	5.000446079	10.0.0.20	10.0.0.24	ICMP	98	Echo (ping) reply id=0x001a, seq=6/1536, ttl=64 (request in…
15	5.008248536	00:00:00_aa:00:00	00:00:00_aa:00:04	ARP	42	Who has 10.0.0.24? Tell 10.0.0.20
16	5.008340484	00:00:00_aa:00:04	00:00:00_aa:00:00	ARP	42	10.0.0.24 is at 00:00:00:aa:00:04

Analisando este resultado podemos observar o funcionamento do protocolo ARP:

- No pacote nº1, enviado em *Broadcast* (para todas as estações), observamos o *request* para saber qual das estações possui o endereço 10.0.0.20 ("*Who has 10.0.0.20?*") e para responder para o endereço 10.0.0.24 ("*Tell 10.0.0.24*");

- No pacote nº2, vemos a identificação do endereço MAC da máquina que possui o endereço IP 10.0.0.20;

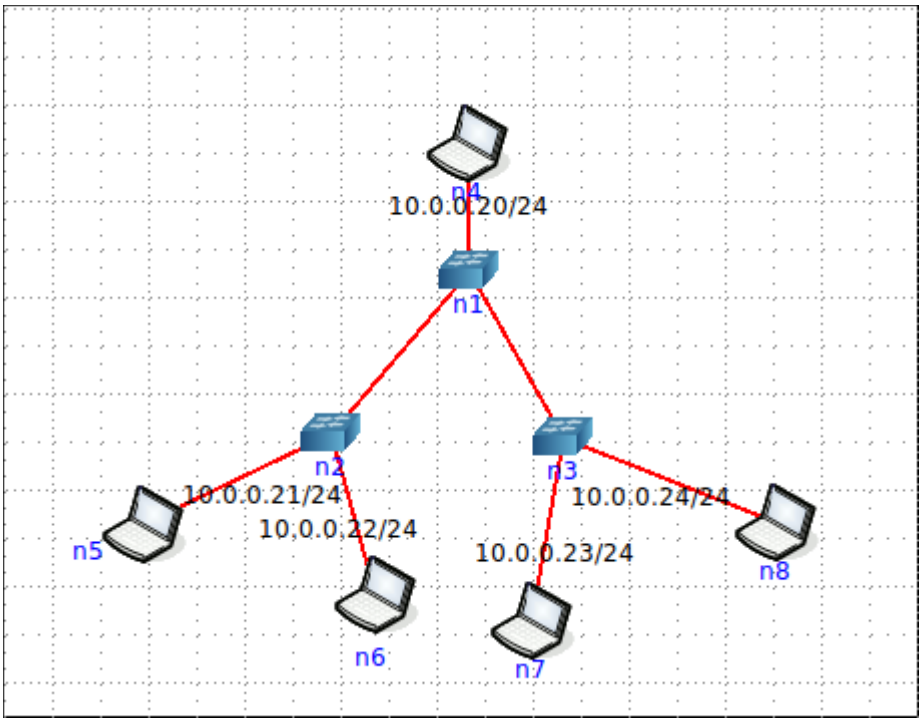
- Mais tarde, no pacote nº15 vemos a estação 10.0.0.20 a enviar um pacote ARP à procura da estação 10.0.0.24, recebendo a resposta no pacote nº16, com o endereço MAC da estação procurada, estabelecendo assim a conexão.

Neste mesmo resultado, observamos também o mecanismo de controlo do protocolo ICMP.

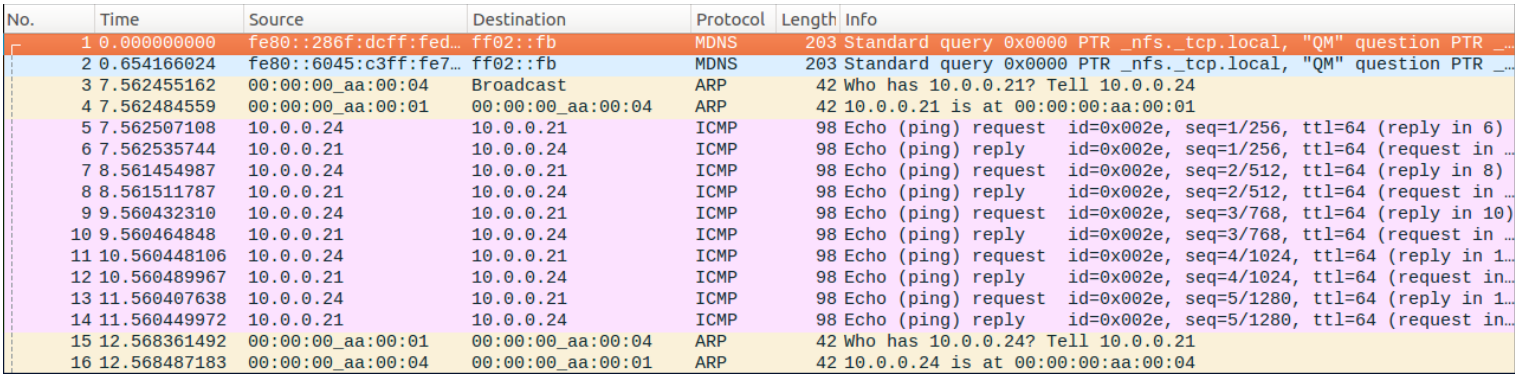
- Nos pacotes nº3 até ao nº14, o protocolo ICMP efetua vários *ping requests*, esperando receber os respetivos *replies*, confirmando o bom funcionamento da ligação.

Topologia com Switch

Foi também criada uma topologia novamente em árvore, mas desta vez conectada com *switches*, como indica a figura abaixo:

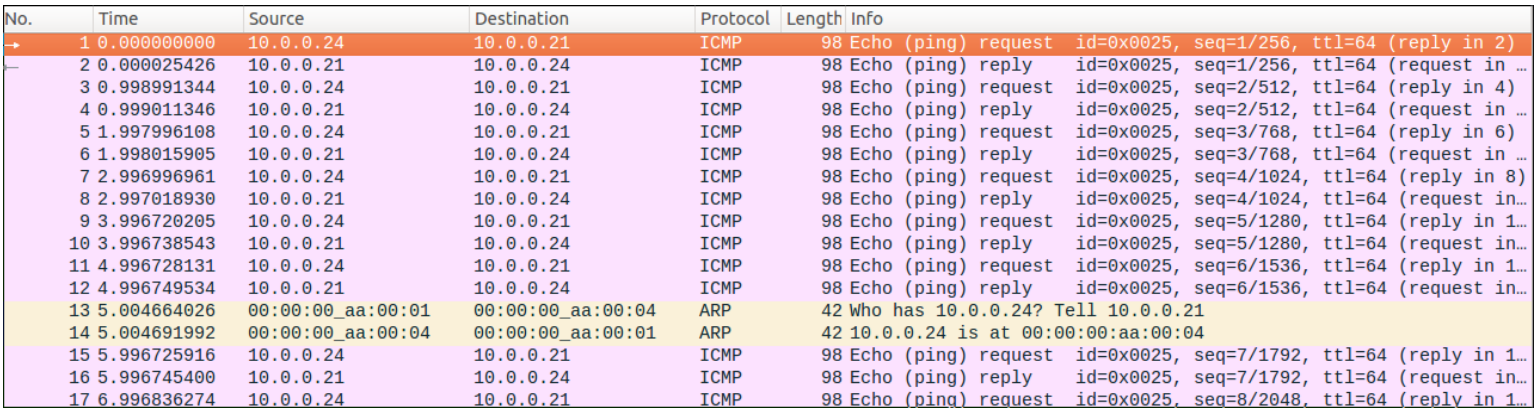


Com o *switch*, o desempenho da rede é substancialmente melhorado em termos de velocidade, e fornece mais privacidade aos utilizadores, devido ao facto de as tramas serem enviadas diretamente para a estação destino. Esta situação pode ser verificada no *Wireshark* novamente. Ao inspecionarmos a estação **n5**, e efetuarmos um ping a partir da estação **n8** obtemos o seguinte:



No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	fe80::286f:dcff:fed…	ff02::fb	MDNS	203	Standard query 0x0000 PTR _nfs._tcp.local, "QM" question PTR _…
2	0.654166024	fe80::6045:c3ff:fe7…	ff02::fb	MDNS	203	Standard query 0x0000 PTR _nfs._tcp.local, "QM" question PTR _…
3	7.562455162	00:00:00_aa:00:04	Broadcast	ARP	42	Who has 10.0.0.21? Tell 10.0.0.24
4	7.562484559	00:00:00_aa:00:01	00:00:00_aa:00:04	ARP	42	10.0.0.21 is at 00:00:00:aa:00:01
5	7.562507108	10.0.0.24	10.0.0.21	ICMP	98	Echo (ping) request id=0x002e, seq=1/256, ttl=64 (reply in 6)
6	7.562535744	10.0.0.21	10.0.0.24	ICMP	98	Echo (ping) reply id=0x002e, seq=1/256, ttl=64 (request in …
7	8.561454987	10.0.0.24	10.0.0.21	ICMP	98	Echo (ping) request id=0x002e, seq=2/512, ttl=64 (reply in 8)
8	8.561511787	10.0.0.21	10.0.0.24	ICMP	98	Echo (ping) reply id=0x002e, seq=2/512, ttl=64 (request in …
9	9.560432310	10.0.0.24	10.0.0.21	ICMP	98	Echo (ping) request id=0x002e, seq=3/768, ttl=64 (reply in 10)
10	9.560464848	10.0.0.21	10.0.0.24	ICMP	98	Echo (ping) reply id=0x002e, seq=3/768, ttl=64 (request in …
11	10.560448106	10.0.0.24	10.0.0.21	ICMP	98	Echo (ping) request id=0x002e, seq=4/1024, ttl=64 (reply in 1…
12	10.560489967	10.0.0.21	10.0.0.24	ICMP	98	Echo (ping) reply id=0x002e, seq=4/1024, ttl=64 (request in…
13	11.560407638	10.0.0.24	10.0.0.21	ICMP	98	Echo (ping) request id=0x002e, seq=5/1280, ttl=64 (reply in 1…
14	11.560449972	10.0.0.21	10.0.0.24	ICMP	98	Echo (ping) reply id=0x002e, seq=5/1280, ttl=64 (request in…
15	12.568361492	00:00:00_aa:00:01	00:00:00_aa:00:04	ARP	42	Who has 10.0.0.24? Tell 10.0.0.21
16	12.568487183	00:00:00_aa:00:04	00:00:00_aa:00:01	ARP	42	10.0.0.24 is at 00:00:00:aa:00:04

À primeira vista, não parece muito diferente do HUB, pois efetua o mesmo broadcast, e aguarda resposta. A grande diferença encontra-se se tentarmos efetuar novamente a comunicação entre estes dois pontos, cujo resultado é o seguinte:



No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	10.0.0.24	10.0.0.21	ICMP	98	Echo (ping) request id=0x0025, seq=1/256, ttl=64 (reply in 2)
2	0.000025426	10.0.0.21	10.0.0.24	ICMP	98	Echo (ping) reply id=0x0025, seq=1/256, ttl=64 (request in …
3	0.998991344	10.0.0.24	10.0.0.21	ICMP	98	Echo (ping) request id=0x0025, seq=2/512, ttl=64 (reply in 4)
4	0.999011346	10.0.0.21	10.0.0.24	ICMP	98	Echo (ping) reply id=0x0025, seq=2/512, ttl=64 (request in …
5	1.997996108	10.0.0.24	10.0.0.21	ICMP	98	Echo (ping) request id=0x0025, seq=3/768, ttl=64 (reply in 6)
6	1.998015905	10.0.0.21	10.0.0.24	ICMP	98	Echo (ping) reply id=0x0025, seq=3/768, ttl=64 (request in …
7	2.996996961	10.0.0.24	10.0.0.21	ICMP	98	Echo (ping) request id=0x0025, seq=4/1024, ttl=64 (reply in 8)
8	2.997018930	10.0.0.21	10.0.0.24	ICMP	98	Echo (ping) reply id=0x0025, seq=4/1024, ttl=64 (request in…
9	3.996720205	10.0.0.24	10.0.0.21	ICMP	98	Echo (ping) request id=0x0025, seq=5/1280, ttl=64 (reply in 1…
10	3.996738543	10.0.0.21	10.0.0.24	ICMP	98	Echo (ping) reply id=0x0025, seq=5/1280, ttl=64 (request in…
11	4.996728131	10.0.0.24	10.0.0.21	ICMP	98	Echo (ping) request id=0x0025, seq=6/1536, ttl=64 (reply in 1…
12	4.996749534	10.0.0.21	10.0.0.24	ICMP	98	Echo (ping) reply id=0x0025, seq=6/1536, ttl=64 (request in…
13	5.004664026	00:00:00_aa:00:01	00:00:00_aa:00:04	ARP	42	Who has 10.0.0.24? Tell 10.0.0.21
14	5.004691992	00:00:00_aa:00:04	00:00:00_aa:00:01	ARP	42	10.0.0.24 is at 00:00:00:aa:00:04
15	5.996725916	10.0.0.24	10.0.0.21	ICMP	98	Echo (ping) request id=0x0025, seq=7/1792, ttl=64 (reply in 1…
16	5.996745400	10.0.0.21	10.0.0.24	ICMP	98	Echo (ping) reply id=0x0025, seq=7/1792, ttl=64 (request in…
17	6.996836274	10.0.0.24	10.0.0.21	ICMP	98	Echo (ping) request id=0x0025, seq=8/2048, ttl=64 (reply in 1…

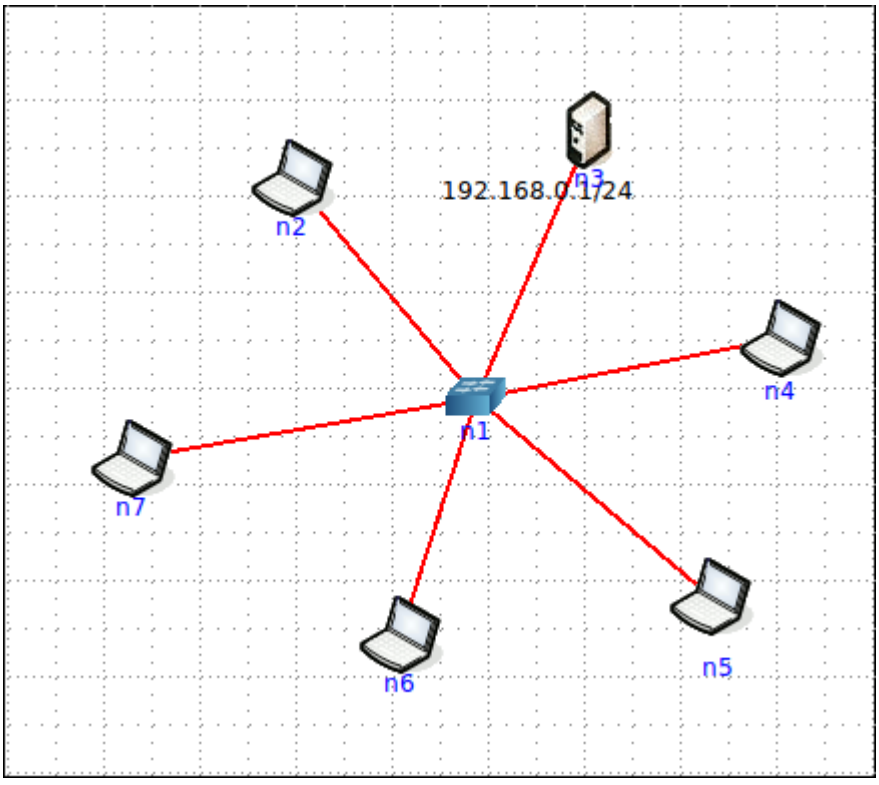
Como podemos verificar, nesta segunda interação entre as estações, os pacotes ICMP de ping são efetuados ainda antes do *broadcast* ser feito, devido à tabela ARP já conter as informações de encaminhamento entre as estações.

3. DHCP

Até agora, os endereços IP foram atribuídos manualmente, o que é o oposto do pretendido no exercício 3. Neste exercício, será usado o protocolo DHCP (*Dynamic Host Configuration Protocol*) entre cliente e servidor, para atribuição dinâmica de endereços IP às estações. Foi também configurado um servidor DHCP no host, de modo a suportar esta operação.

Este protocolo traz muitas vantagens no sentido em que são evitados erros de configuração pelo facto de ser um protocolo em que a atribuição de endereços é automática. Conflitos de endereços que já estejam em utilização também são anulados graças a este protocolo.

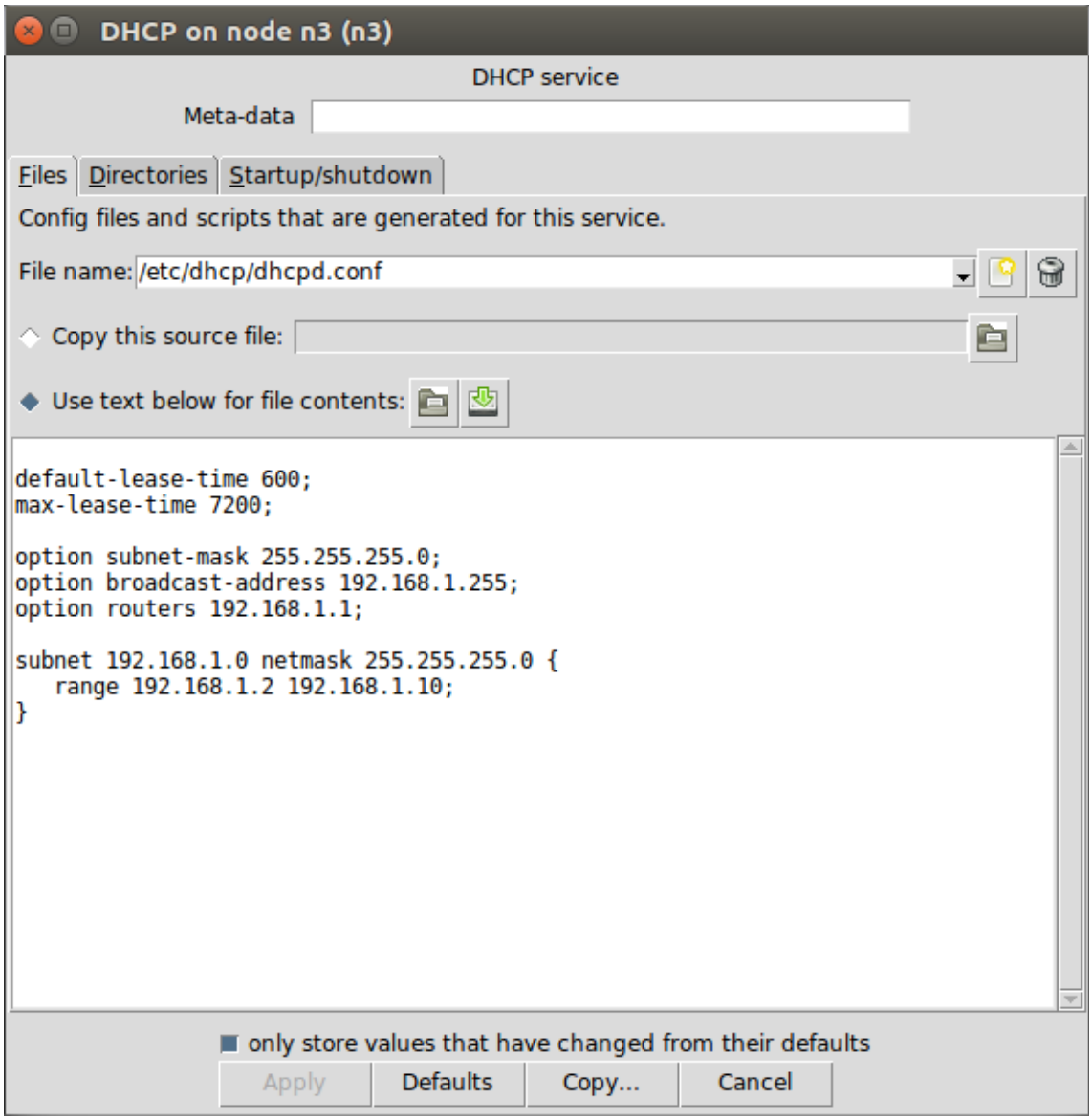
A topologia usada foi a seguinte:



O servidor DHCP, possuidor do endereço IP 192.168.0.1, tem como objetivo atribuir dinamicamente os endereços às quatro estações ligadas a ele. A configuração do servidor é a seguinte:

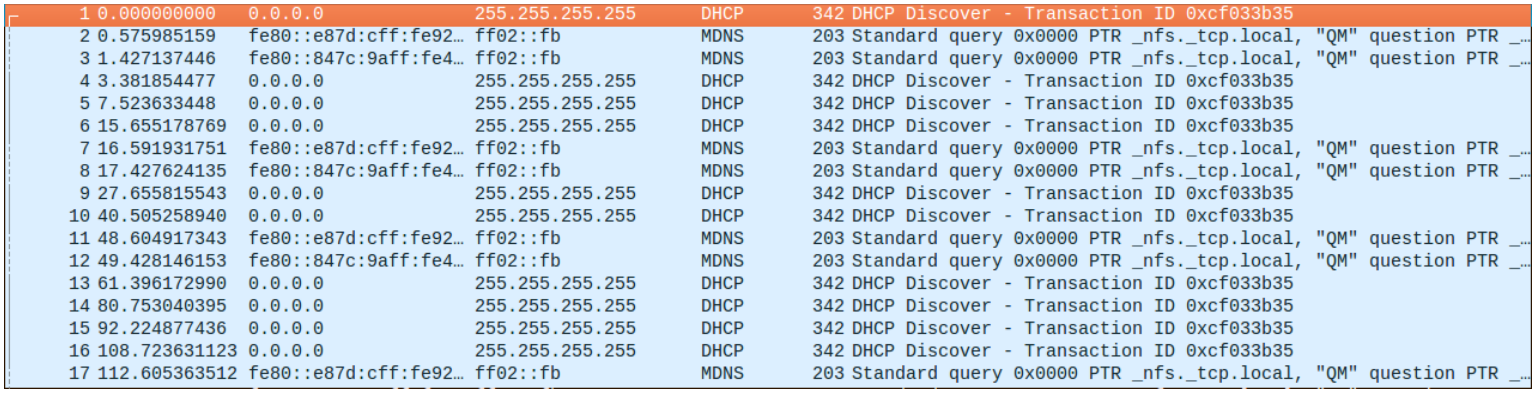
Os parâmetros configurados foram:

- ***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** – 255.255.255.0
- **Endereço de difusão** – 192.168.1.255
- **Encaminhamento** – 192.168.1.1
- **Gama de endereços** – desde 192.168.1.2 até 192.168.1.10



Foram também introduzidos atrasos na estação n2 (configurada como DHCP *client*) e no host de modo a conseguirmos testemunhar toda a interação e utilização do DHCP.

O resultado esperado deste exercício seria obter tramas de *Discover, Offer, Request e ACK,* de modo a que as estações recebessem um endereço da gama definida, e fosse esse utilizado durante as comunicações. No entanto, devido a algum erro na configuração, ou no sistema operativo, que não conseguimos solucionar a tempo, a inspeção do *Wireshark* tanto da estação **n2** como do host **n3** é a seguinte:



1	0.000000000	0.0.0.0	255.255.255.255	DHCP	342	DHCP Discover - Transaction ID 0xcf033b35
2	0.575985159	fe80::e87d:cff:fe92…	ff02::fb	MDNS	203	Standard query 0x0000 PTR _nfs._tcp.local, "QM" question PTR _…
3	1.427137446	fe80::847c:9aff:fe4…	ff02::fb	MDNS	203	Standard query 0x0000 PTR _nfs._tcp.local, "QM" question PTR _…
4	3.381854477	0.0.0.0	255.255.255.255	DHCP	342	DHCP Discover - Transaction ID 0xcf033b35
5	7.523633448	0.0.0.0	255.255.255.255	DHCP	342	DHCP Discover - Transaction ID 0xcf033b35
6	15.655178769	0.0.0.0	255.255.255.255	DHCP	342	DHCP Discover - Transaction ID 0xcf033b35
7	16.591931751	fe80::e87d:cff:fe92…	ff02::fb	MDNS	203	Standard query 0x0000 PTR _nfs._tcp.local, "QM" question PTR _…
8	17.427624135	fe80::847c:9aff:fe4…	ff02::fb	MDNS	203	Standard query 0x0000 PTR _nfs._tcp.local, "QM" question PTR _…
9	27.655815543	0.0.0.0	255.255.255.255	DHCP	342	DHCP Discover - Transaction ID 0xcf033b35
10	40.505258940	0.0.0.0	255.255.255.255	DHCP	342	DHCP Discover - Transaction ID 0xcf033b35
11	48.604917343	fe80::e87d:cff:fe92…	ff02::fb	MDNS	203	Standard query 0x0000 PTR _nfs._tcp.local, "QM" question PTR _…
12	49.428146153	fe80::847c:9aff:fe4…	ff02::fb	MDNS	203	Standard query 0x0000 PTR _nfs._tcp.local, "QM" question PTR _…
13	61.396172990	0.0.0.0	255.255.255.255	DHCP	342	DHCP Discover - Transaction ID 0xcf033b35
14	80.753040395	0.0.0.0	255.255.255.255	DHCP	342	DHCP Discover - Transaction ID 0xcf033b35
15	92.224877436	0.0.0.0	255.255.255.255	DHCP	342	DHCP Discover - Transaction ID 0xcf033b35
16	108.723631123	0.0.0.0	255.255.255.255	DHCP	342	DHCP Discover - Transaction ID 0xcf033b35
17	112.605363512	fe80::e87d:cff:fe92…	ff02::fb	MDNS	203	Standard query 0x0000 PTR _nfs._tcp.local, "QM" question PTR _…

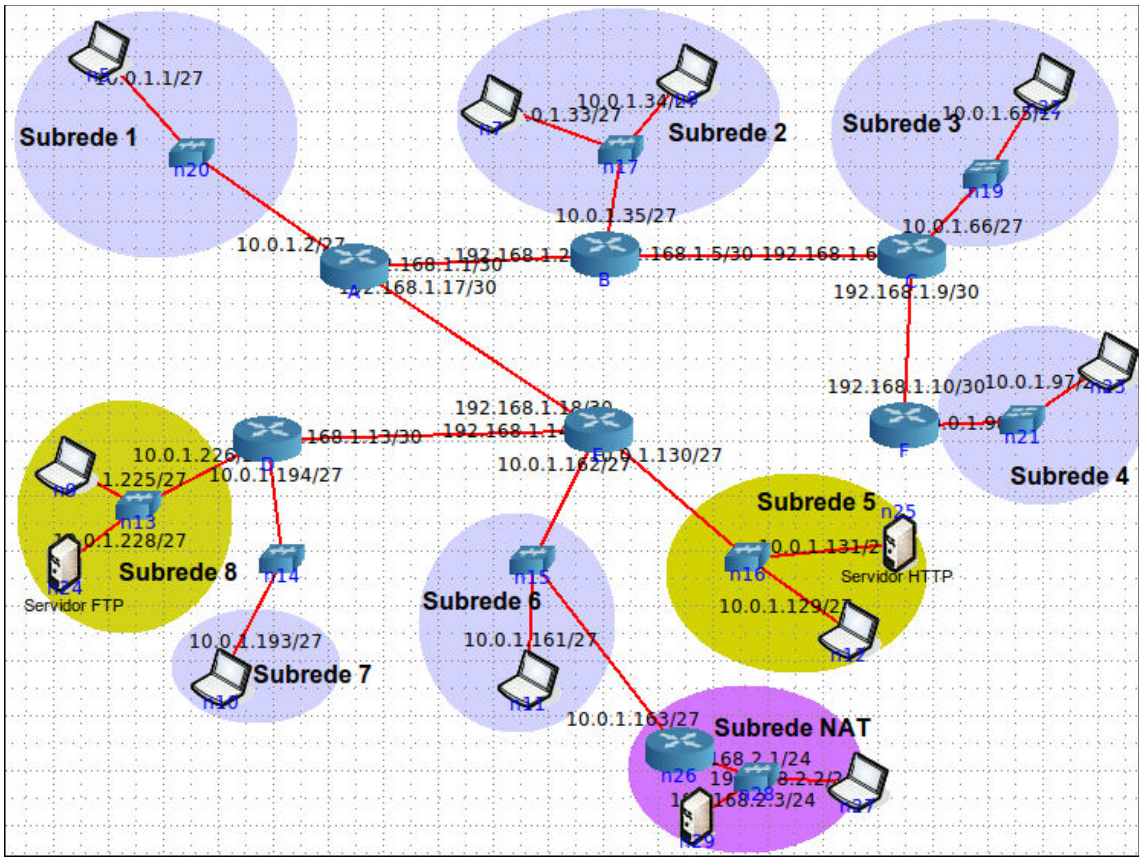
Como podemos observar, o protocolo DHCP fica “preso” no estado *Discover,* não sendo verificadas as outras interações de atribuição de endereço IP. Infelizmente, este exercício encontra-se incompleto.

4. Interligação de redes

Neste exercício é pretendido interligar as redes IP distintas utilizando *routers* capazes de encaminhar o tráfego IP de umas redes para as outras, isto é, ligar as redes criadas e configuradas nos exercícios anteriores.

A topologia desta interligação de redes encontra-se representada na figura seguinte (esta topologia é a mesma usada nos exercícios 5 e 6):

Os endereços utilizados foram os da gama 10.0.1.0/24, e os endereços de ligação foram 192.168.1.0/24, tal como referido no enunciado. Existem também 8 sub-redes, sendo que a sub-rede 8 aloja o servidor FTP e a sub-rede



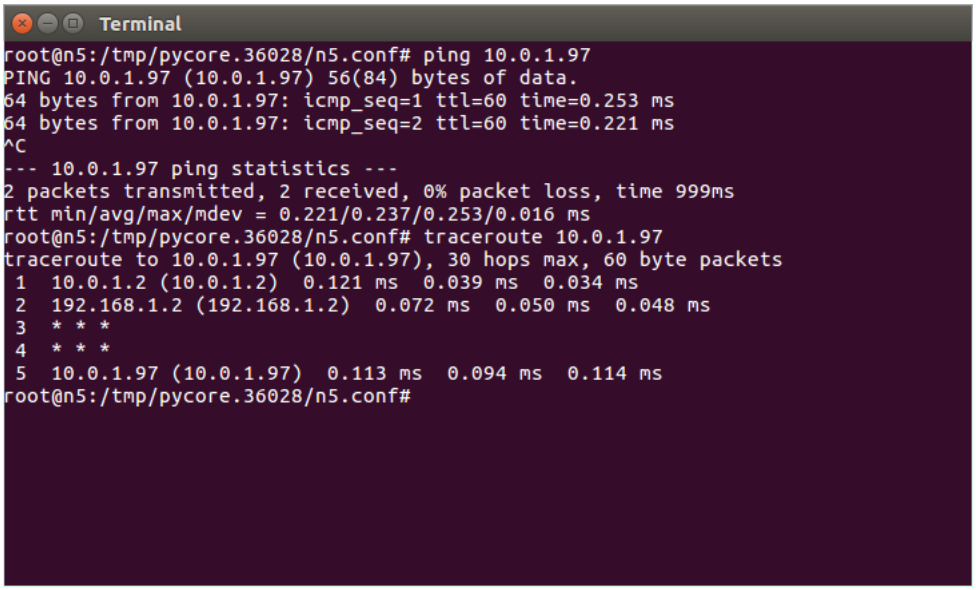
5 aloja o servidor HTTP (para além do servidor FTP na rede NAT, que será abordada mais à frente). As redes possuem os seguintes endereços:

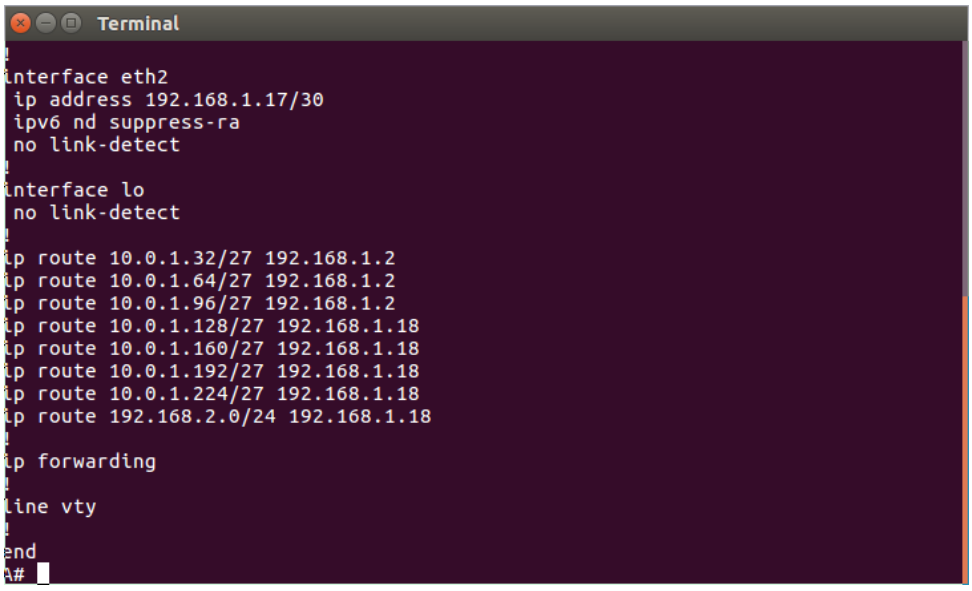
REDE 1	10.0.1.0/27
REDE 2	10.0.1.32/27
REDE 3	10.0.1.64/27
REDE 4	10.0.1.96/27
REDE 5	10.0.1.128/27
REDE 6	10.0.1.160/27
REDE 7	10.0.1.192/27
REDE 8	10.0.1.224/27

Para configurar os routers, foi aberto em cada um deles uma janela *vtysh* no modo de configuração, e foi introduzido manualmente o comando *"ip route [endereço de destino] [próximo nó]"*. Foram introduzidas as ligações para cada uma das outras redes que não estão diretamente conectadas ao router a ser configurado. Após esta personalização, para verificar as tabelas de

encaminhamento, basta introduzir o comando "*show running-config*" que nos dará a informação da tabela de encaminhamento e de cada uma das interfaces do router.

Abaixo está uma imagem exemplificativa de um ping e de um traceroute da estação **n5**, de endereço 10.0.1.1/27 para a estação **n23** 10.0.1.97/27, e de seguida um exemplo do comando *"show running-config"* no router



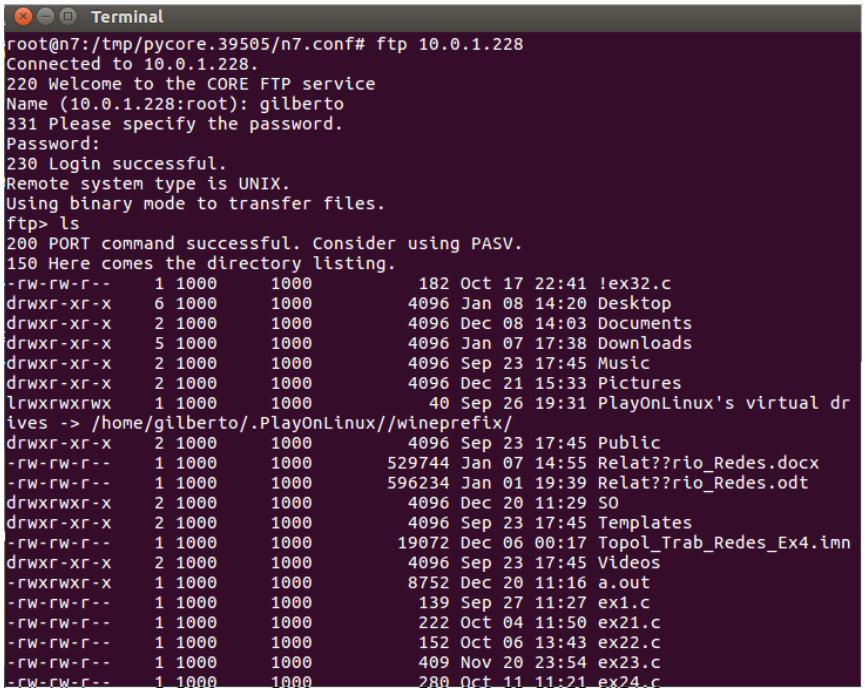


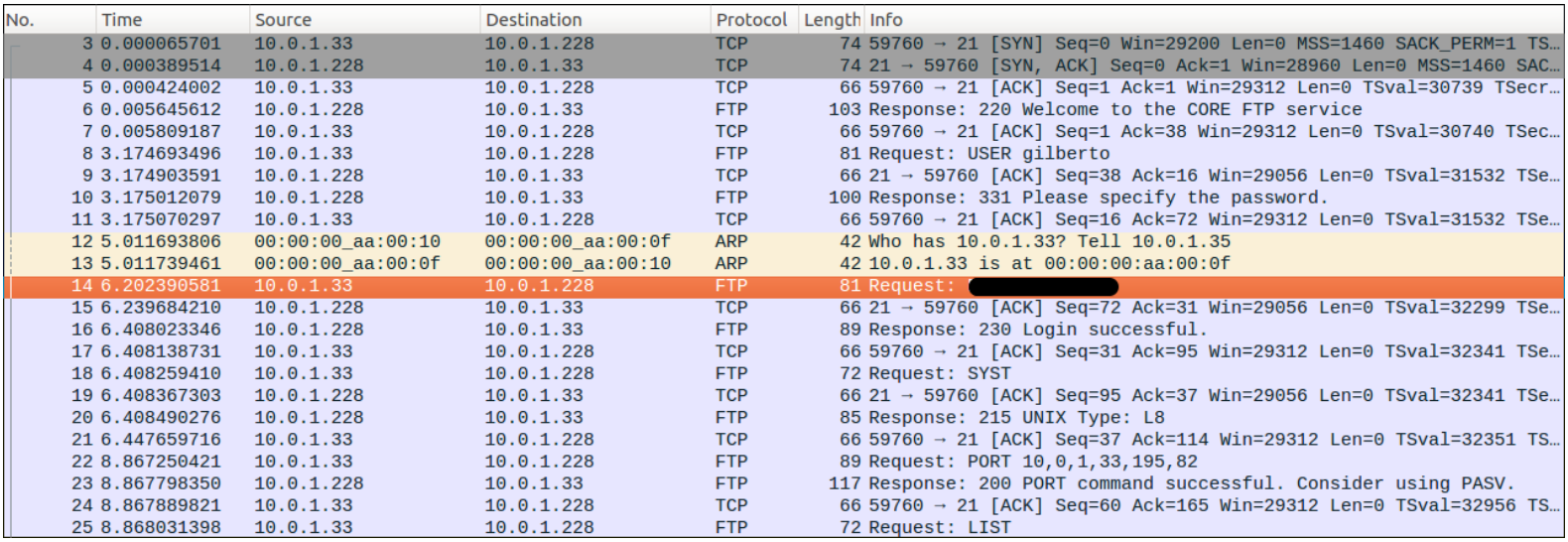
5. Uso das camadas de rede e transporte por parte das aplicações

Depois de criada e configurada, é pretendido que a rede emulada suporte serviços e execute aplicações de rede. Exemplos destes serviços são os serviços HTTP e o FTP. É pedido a ativação de pelo menos um servidor HTTP e um FTP e usá-los depois num *host* da rede emulada no CORE.

Relativamente ao servidor FTP (utilizando o *vsftpd*), este foi conectado à rede 8, com endereço IP 10.0.1.228, sendo ativado e iniciado com os comandos *"chmod a-w /var/run/vsftpd/empty", "chmod a-w /var/ftp" e "vsftpd ./vsftpd.conf".*

De modo a testar o funcionamento do servidor, abrimos uma bash no terminal **n7,** de endereço 10.0.1.33, e foi introduzido o comando *"ftp 10.0.1.228"*. Foi nos apresentada então a mensagem de boas-vindas, e espaço para efetuarmos *login* no servidor com utilizador e palavra-passe. Após efetuado o *login* com sucesso, introduzimos o comando "*ls*", que permite listar os ficheiros de uma determinada diretoria. Conforme esperado foi nos apresentado a lista de ficheiros do *host*, o que prova que o servidor está funcional. Seguem abaixo as imagens do terminal com a invocação dos comandos, e a inspeção do *Wireshark* desta interação.





No.	Time	Source	Destination	Protocol	Length	Info
3	0.000065701	10.0.1.33	10.0.1.228	TCP	74	59760 → 21 [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TS…
4	0.000389514	10.0.1.228	10.0.1.33	TCP	74	21 → 59760 [SYN, ACK] Seq=0 Ack=1 Win=28960 Len=0 MSS=1460 SAC…
5	0.000424002	10.0.1.33	10.0.1.228	TCP	66	59760 → 21 [ACK] Seq=1 Ack=1 Win=29312 Len=0 TSval=30739 TSecr…
6	0.005645612	10.0.1.228	10.0.1.33	FTP	103	Response: 220 Welcome to the CORE FTP service
7	0.005809187	10.0.1.33	10.0.1.228	TCP	66	59760 → 21 [ACK] Seq=1 Ack=38 Win=29312 Len=0 TSval=30740 TSec…
8	3.174693496	10.0.1.33	10.0.1.228	FTP	81	Request: USER gilberto
9	3.174903591	10.0.1.228	10.0.1.33	TCP	66	21 → 59760 [ACK] Seq=38 Ack=16 Win=29056 Len=0 TSval=31532 TSe…
10	3.175012079	10.0.1.228	10.0.1.33	FTP	100	Response: 331 Please specify the password.
11	3.175070297	10.0.1.33	10.0.1.228	TCP	66	59760 → 21 [ACK] Seq=16 Ack=72 Win=29312 Len=0 TSval=31532 TSe…
12	5.011693806	00:00:00_aa:00:10	00:00:00_aa:00:0f	ARP	42	Who has 10.0.1.33? Tell 10.0.1.35
13	5.011739461	00:00:00_aa:00:0f	00:00:00_aa:00:10	ARP	42	10.0.1.33 is at 00:00:00:aa:00:0f
14	6.202390581	10.0.1.33	10.0.1.228	FTP	81	Request: [redacted]
15	6.239684210	10.0.1.228	10.0.1.33	TCP	66	21 → 59760 [ACK] Seq=72 Ack=31 Win=29056 Len=0 TSval=32299 TSe…
16	6.408023346	10.0.1.228	10.0.1.33	FTP	89	Response: 230 Login successful.
17	6.408138731	10.0.1.33	10.0.1.228	TCP	66	59760 → 21 [ACK] Seq=31 Ack=95 Win=29312 Len=0 TSval=32341 TSe…
18	6.408259410	10.0.1.33	10.0.1.228	FTP	72	Request: SYST
19	6.408367303	10.0.1.228	10.0.1.33	TCP	66	21 → 59760 [ACK] Seq=95 Ack=37 Win=29056 Len=0 TSval=32341 TSe…
20	6.408490276	10.0.1.228	10.0.1.33	FTP	85	Response: 215 UNIX Type: L8
21	6.447659716	10.0.1.33	10.0.1.228	TCP	66	59760 → 21 [ACK] Seq=37 Ack=114 Win=29312 Len=0 TSval=32351 TS…
22	8.867250421	10.0.1.33	10.0.1.228	FTP	89	Request: PORT 10,0,1,33,195,82
23	8.867798350	10.0.1.228	10.0.1.33	FTP	117	Response: 200 PORT command successful. Consider using PASV.
24	8.867889821	10.0.1.33	10.0.1.228	TCP	66	59760 → 21 [ACK] Seq=60 Ack=165 Win=29312 Len=0 TSval=32956 TS…
25	8.868031398	10.0.1.33	10.0.1.228	FTP	72	Request: LIST

Na imagem relativa ao *Wireshark* podemos ver a interação e os pacotes FTP, nomeadamente o pacote de boas-vindas (número 6), os requests de *user* e *password* (número 8 e 14), e o comando “ls”, ou *Request: LIST* (número 25).

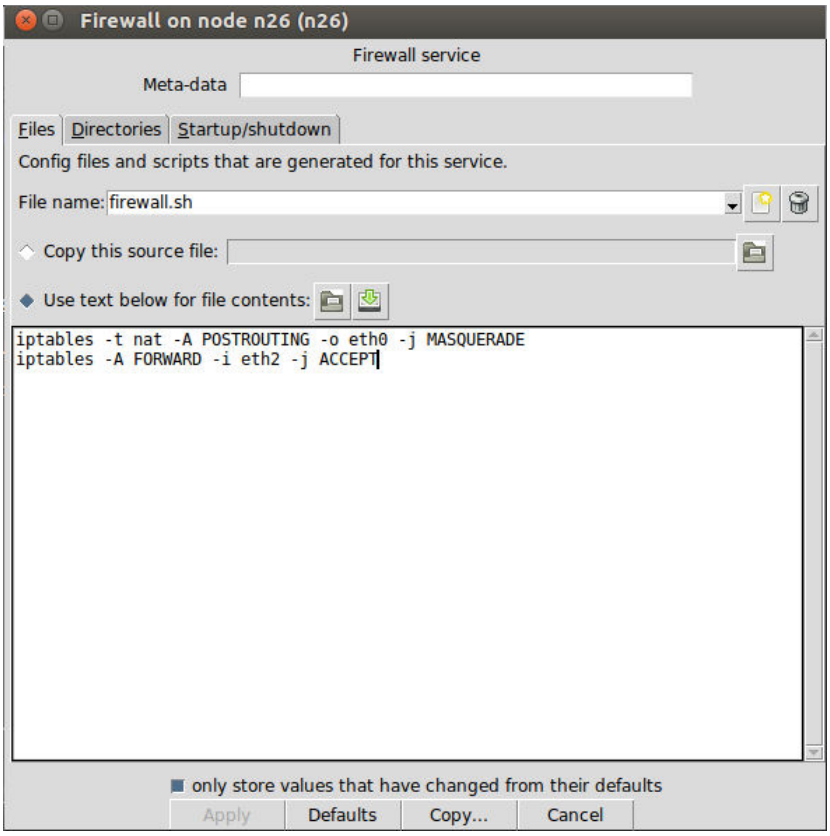
Relativamente ao servidor HTTP, tentamos utilizar o *Apache2*, que foi configurado e ativado, que tentamos testar com o serviço *Lynx*, no entanto não obtivemos sucesso nesta tarefa.

6. Interligação via NAT (Network Address Translator)

Por fim é requerido aos alunos que neste último exercício que seja criada uma interligação via NAT. É acrescentada uma rede local, na qual devem ser utilizando endereços privados da gama 192.168.2.0/24. Para existir conectividade, esta rede terá que ser ligada a uma das redes LAN configuradas anteriormente através de um *router* NAT.

Para este exercício focamo-nos na sub-rede NAT, ligada à sub-rede 6. Durante o seu funcionamento verificaremos que o endereço IP do destino não será anunciado em *broadcast*, mas manter-se-á atingível e conectado durante a interação de pings.

A configuração do router NAT foi efetuada na secção *Firewall* do *CORE*, e é feita da seguinte maneira:



E a captura dos pacotes, usando o *Wireshark*, a partir do ponto onde foi efetuado o *ping* (computador **n7**, com endereço 10.0.1.33).

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	fe80::fc:efff:fea8:…	ff02::fb	MDNS	203	Standard query 0x0000 PTR _nfs._tcp.local, "QM" question PTR _…
2	0.824320890	fe80::c9d:f5ff:fe1b…	ff02::fb	MDNS	203	Standard query 0x0000 PTR _nfs._tcp.local, "QM" question PTR _…
3	25.037174443	00:00:00_aa:00:0f	Broadcast	ARP	42	Who has 10.0.1.35? Tell 10.0.1.33
4	25.037240189	00:00:00_aa:00:10	00:00:00_aa:00:0f	ARP	42	10.0.1.35 is at 00:00:00:aa:00:10
5	25.037249024	10.0.1.33	192.168.2.2	ICMP	98	Echo (ping) request id=0x001a, seq=1/256, ttl=64 (reply in 6)
6	25.037595650	192.168.2.2	10.0.1.33	ICMP	98	Echo (ping) reply id=0x001a, seq=1/256, ttl=60 (request in …
7	26.036163504	10.0.1.33	192.168.2.2	ICMP	98	Echo (ping) request id=0x001a, seq=2/512, ttl=64 (reply in 8)
8	26.036350736	192.168.2.2	10.0.1.33	ICMP	98	Echo (ping) reply id=0x001a, seq=2/512, ttl=60 (request in …
9	27.035165609	10.0.1.33	192.168.2.2	ICMP	98	Echo (ping) request id=0x001a, seq=3/768, ttl=64 (reply in 10)
10	27.035355397	192.168.2.2	10.0.1.33	ICMP	98	Echo (ping) reply id=0x001a, seq=3/768, ttl=60 (request in …
11	28.034734171	10.0.1.33	192.168.2.2	ICMP	98	Echo (ping) request id=0x001a, seq=4/1024, ttl=64 (reply in 1…
12	28.034938032	192.168.2.2	10.0.1.33	ICMP	98	Echo (ping) reply id=0x001a, seq=4/1024, ttl=60 (request in…
13	29.034754416	10.0.1.33	192.168.2.2	ICMP	98	Echo (ping) request id=0x001a, seq=5/1280, ttl=64 (reply in 1…
14	29.034958627	192.168.2.2	10.0.1.33	ICMP	98	Echo (ping) reply id=0x001a, seq=5/1280, ttl=60 (request in…
15	30.034758213	10.0.1.33	192.168.2.2	ICMP	98	Echo (ping) request id=0x001a, seq=6/1536, ttl=64 (reply in 1…
16	30.034960825	192.168.2.2	10.0.1.33	ICMP	98	Echo (ping) reply id=0x001a, seq=6/1536, ttl=60 (request in…
17	30.038743766	00:00:00_aa:00:10	00:00:00_aa:00:0f	ARP	42	Who has 10.0.1.33? Tell 10.0.1.35
18	30.038805316	00:00:00_aa:00:0f	00:00:00_aa:00:10	ARP	42	10.0.1.33 is at 00:00:00:aa:00:0f
19	31.034753854	10.0.1.33	192.168.2.2	ICMP	98	Echo (ping) request id=0x001a, seq=7/1792, ttl=64 (reply in 2…
20	31.034930408	192.168.2.2	10.0.1.33	ICMP	98	Echo (ping) reply id=0x001a, seq=7/1792, ttl=60 (request in…
21	32.034751793	10.0.1.33	192.168.2.2	ICMP	98	Echo (ping) request id=0x001a, seq=8/2048, ttl=64 (reply in 2…
22	32.034953423	192.168.2.2	10.0.1.33	ICMP	98	Echo (ping) reply id=0x001a, seq=8/2048, ttl=60 (request in…

Também colocamos um servidor FTP, que repetindo o mesmo procedimento mencionado acima, está a funcionar corretamente.

7. Conclusão

Após a conclusão deste trabalho, e refletindo sobre os resultados obtidos, podemos afirmar que foi importante conseguirmos ver em acção os conceitos e procedimentos leccionados na aula teórica, que consequentemente se reflete numa melhor compreensão do funcionamento de uma rede. Foi importante testemunhar situações como estabelecimento de conexão entre duas estações, tabelas de encaminhamento dos routers, atribuição estática de endereços IP, protocolo NAT, bem como o funcionamento de algumas aplicações, nomeadamente servidores FTP.

No entanto, deparámo-nos com dificuldades a nível de configuração dos serviços (sendo complicado encontrarmos os comandos e código certos para o correto funcionamento dos serviços) – isso reflete-se no fato de não termos conseguido colocar o servidor HTTP a funcionar.

Contudo, no nosso ponto de vista, o panorama é positivo, e os conhecimentos adquiridos nas aulas da UC foram bastante consolidados.