Universidade do Minho

Relatório do Projeto 2016/17

Universidade do Minho
Escola de Engenharia

Redes de Computadores 1

Docente: Maria João Mesquita Rodrigues Cunha Nicolau Pinto

Mestrado Integrado de Engenharia de Telecomunicações e Informática – 3ºAno

26 de novembro de 2016

Constituição do Grupo

Francisco Peixoto Silva, A68491

a68491@alunos.uminho.pt

Luís Pedro Lobo de Araújo, A73232

a73232@alunos.uminho.pt

Nuno Frederico Oliveira Tavares, A73985

a73985@alunos.uminho.pt

Índice

Índice de Figuras e Tabelas

Glossário

ARP – *Address Resolution Protocol*

DHCP – *Dynamic Host Configuration Protocol*

FTP – *File Transfer Protocol*

HTTP – *Hypertext Transfer Protocol*

ICMP – *Internet Control Message Protocol*

IP – *Internet Protocol*

LAN – *Local Area Networks*

NAT – *Network Address Translation*

TCP – *Transmission Control Protocol*

UC – Unidade Curricular

1. Introdução

Este relatório enquadra-se no âmbito da UC (Unidade Curricular) Redes de Computadores 1, o relatório relata todo o desenvolvimento por parte dos elementos deste grupo dos vários problemas propostos pela docente neste trabalho prático. Esses mesmos problemas envolvem a emulação de vários tipos de redes, interligando-as entre si. Para sustentar este processo o grupo utilizou as ferramentas: *CORE* (utilizada para a edição das várias topologias e as respetivas simulações) e *Wireshark* (utilizada para diagnosticar a conectividade e analisar as capturas de tráfego entre os vários nodos das redes emuladas).

Os vários tipos de topologias implementadas no seguimento deste relatório terão diferentes objetivos de aprendizagem, cada uma delas com uma configuração e diferentes tipos de máquinas associadas a essas topologias. Ao longo deste trabalho prático executamos comandos que serão frequentes durante a análise da performance dos exercícios implementados são o *ping* e *traceroute*.

2. Emulação de LANs Ethernet

O primeiro exercício é pedido que emulemos pequenas redes locais (LAN – *Local Area Networks*) no *CORE.* O grupo decidiu implementar nestas redes 2 tipos de topologias, em estrela e em árvore, sendo que elaboramos 2 exemplos para a topologia em estrela (usando um com 1 *HUB* e o outro com 1 *SWITCH*) e 1 exemplo para uma topologia em árvore (implementando com os dois tipos de equipamentos, *HUBs* e *SWITCHs*). Para desenhar as várias topologias o grupo utilizou a interface gráfica do *CORE* e também configurou as ligações entre os vários nodos, os endereços de cada máquina e definir o papel que esta irá desempenhar. Todos os endereços atribuídos aos equipamentos inseridos nas redes emuladas no *CORE* neste exercício serão na base do endereço 10.0.0.0/24 com uma máscara de 24 bits.

Na Figura 1 podemos visualizar a primeira topologia desenhada num formato em estrela com um *HUB* a servir de meio de comunicação entre os vários terminais (2 computadores e 2 *hosts*).

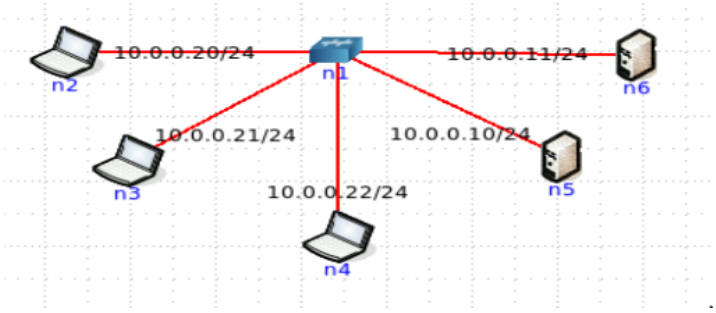


Figura 1 - Topologia em estrela com um HUB a ligar os vários terminais

A rede seguinte foi desenhada com a mesma topologia, substituíndo o *HUB* por um *SWITCH*, presente na Figura 2.

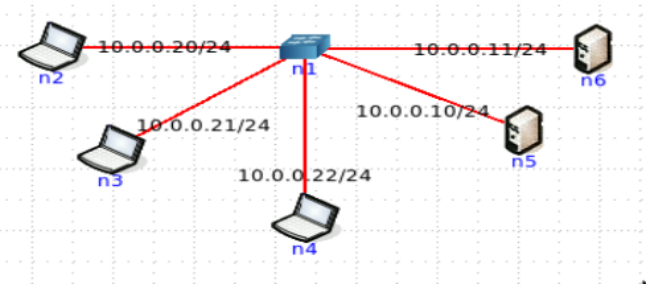


Figura 2 - Topologia em estrela com um SWITCH a ligar os vários terminais

A última rede implementada (Figura 3) segue uma topologia em árvore, constituída por 3 *HUB*'s ligando os vários terminais (4 *computadores* e 2 *hosts*).

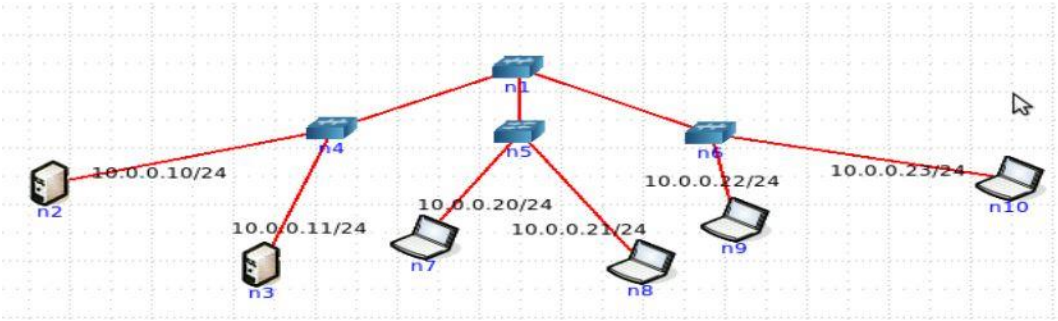


Figura 3 - Topologia em árvore com 1 *SWITCH* e 3 *HUBs*

As Figuras seguintes apresentam toda a análise feita através do comando *ping* às diversas topologias implementadas. Este comando permite-nos verificar se os terminais estão a comunicar bem entre eles, ou seja, a rede está a funcionar de acordo com o que era previsto, também nos me permite analisar o *round-trip-time* seguindo vários parâmetros e calcula a percencentagem de pacotes perdidos.

Temos na Figura 4 a representação da execução do comando *ping* na primeira rede através do terminal do nodo 2 (10.0.0.20/24) para o resto dos terminais que estão ligados nesta rede.

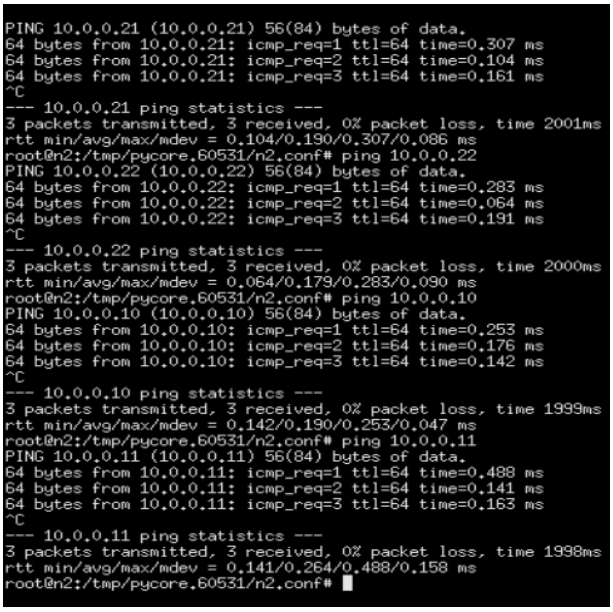


Figura 4 - Resultado do comando *ping* no terminal n2 para os outros terminais da primeira rede

Na Figura 5 podemos ver o resultado das mesmas operações para a segunda rede desenhada.

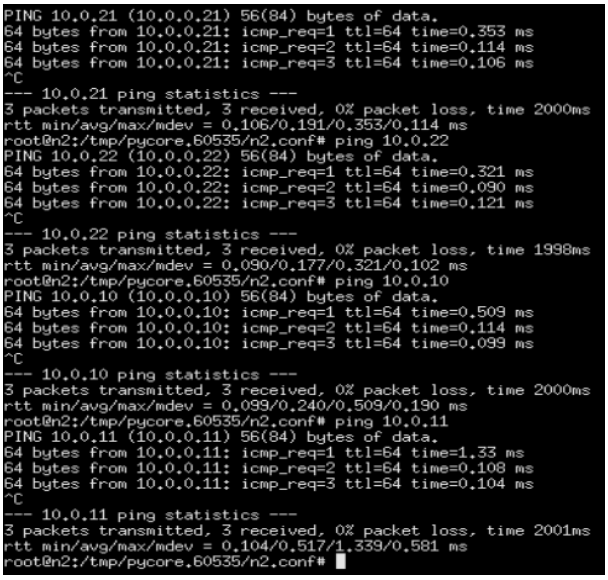


Figura 5 - Resultado do comando *ping* no terminal n2 para os outros terminais da segunda rede

Na terceira rede (topologia em árvore) o grupo executou o comando ping através do nodo 2 para todos os endereços dos terminais envolvidos na rede, podemos verificar o seu resultado na Figura 6. Concluímos que a rede está a funcionar devidamente.

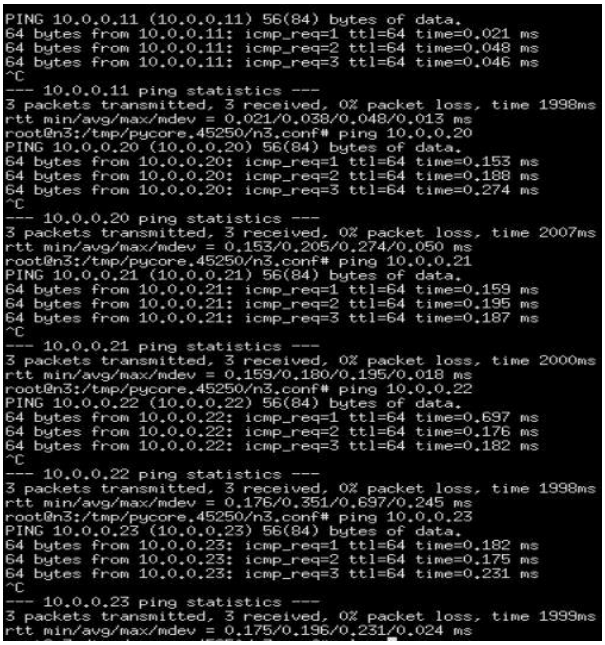


Figura 6 - Resultado do comando ping no terminal n2 para os outros terminais da terceira rede

Para finalizar este exercício analisamos a captura de tráfego por parte do Wireshark para os vários comandos que executamos anteriormente. Esta captura de tráfego permite-nos analisar os pacotes correspondentes aos protocolos ARP (*Address Resolution Protocol*) e ICMP (*Internet Control Message Protocol*) e entender o funcionamento destes dois protocolos e também verificar as diferenças comportamentais entre um *SWITCH* e um *HUB*.

As Figuras 7, 8 e 9 apresentam as capturas de tráfego por parte de alguns terminais envolvidos nas topologias anteriormente desenhadas.

Nas figuras 7 e 8 podemos verificar as diferenças entre a captura de tráfego no nodo 3 (10.0.0.21/24) após a execução do comando *ping* entre a primeira e a segunda rede. Concluímos que se o equipamento for um *SWITCH* a comunicação será mais rápida e eficiente, pois este ao contrário do *HUB* apenas envia pacotes do protocolo ICMP para o terminal pretendido, já o *HUB* repete para todos os terminais que estão ligados a eles.

	Time	Source	Destination	Protocol	Length	Info
1	0.000000	10.0.0.20	10.0.0.21	ICMP	98	Echo (ping) request id=0x001e, seq=1/256, ttl=64
2	0.000041	10.0.0.21	10.0.0.20	ICMP	98	Echo (ping) reply id=0x001e, seq=1/256, ttl=64
3	0.998996	10.0.0.20	10.0.0.21	ICMP	98	Echo (ping) request id=0x001e, seq=2/512, ttl=64
4	0.999039	10.0.0.21	10.0.0.20	ICMP	98	Echo (ping) reply id=0x001e, seq=2/512, ttl=64
5	1.998986	10.0.0.20	10.0.0.21	ICMP	98	Echo (ping) request id=0x001e, seq=3/768, ttl=64
6	1.999039	10.0.0.21	10.0.0.20	ICMP	98	Echo (ping) reply id=0x001e, seq=3/768, ttl=64
7	5.002744	00:00:00_aa:00:01	00:00:00_aa:00:00	ARP	42	Who has 10.0.0.20? Tell 10.0.0.21
8	5.002804	00:00:00_aa:00:00	00:00:00_aa:00:01	ARP	42	10.0.0.20 is at 00:00:00:aa:00:00
9	28.569760	10.0.0.20	10.0.0.22	ICMP	98	Echo (ping) request id=0x001f, seq=1/256, ttl=64
10	28.569839	10.0.0.22	10.0.0.20	ICMP	98	Echo (ping) reply id=0x001f, seq=1/256, ttl=64
11	29.571528	10.0.0.20	10.0.0.22	ICMP	98	Echo (ping) request id=0x001f, seq=2/512, ttl=64
12	29.571644	10.0.0.22	10.0.0.20	ICMP	98	Echo (ping) reply id=0x001f, seq=2/512, ttl=64
13	30.570886	10.0.0.20	10.0.0.22	ICMP	98	Echo (ping) request id=0x001f, seq=3/768, ttl=64
14	30.570956	10.0.0.22	10.0.0.20	ICMP	98	Echo (ping) reply id=0x001f, seq=3/768, ttl=64
15	33.578807	00:00:00_aa:00:02	00:00:00_aa:00:00	ARP	42	Who has 10.0.0.20? Tell 10.0.0.22
16	33.578855	00:00:00_aa:00:00	00:00:00_aa:00:02	ARP	42	10.0.0.20 is at 00:00:00:aa:00:00

Figura 7 - Capturas de tráfego no nodo 3 na primeira rede

	Time	Source	Destination	Protocol	Length	Info
1	0.000000	10.0.0.20	10.0.0.21	ICMP	98	Echo (ping) request id=0x001e, seq=1/256, ttl=64
2	0.000104	10.0.0.21	10.0.0.20	ICMP	98	Echo (ping) reply id=0x001e, seq=1/256, ttl=64
3	1.003533	10.0.0.20	10.0.0.21	ICMP	98	Echo (ping) request id=0x001e, seq=2/512, ttl=64
4	1.003577	10.0.0.21	10.0.0.20	ICMP	98	Echo (ping) reply id=0x001e, seq=2/512, ttl=64
5	2.002537	10.0.0.20	10.0.0.21	ICMP	98	Echo (ping) request id=0x001e, seq=3/768, ttl=64
6	2.002595	10.0.0.21	10.0.0.20	ICMP	98	Echo (ping) reply id=0x001e, seq=3/768, ttl=64
7	5.004705	00:00:00_aa:00:01	00:00:00_aa:00:00	ARP	42	Who has 10.0.0.20? Tell 10.0.0.21
8	5.004754	00:00:00_aa:00:00	00:00:00_aa:00:01	ARP	42	10.0.0.20 is at 00:00:00:aa:00:00
9	12.265213	10.0.0.20	10.0.0.22	ICMP	98	Echo (ping) request id=0x001f, seq=1/256, ttl=64

Figura 8 - Capturas de tráfego no nodo 3 na segunda rede

Na Figura 9 podemos verificar as capturas de tráfego por parte do nodo 7 (10.0.0.20/24), pertencente à topologia C. Foi executado o comando *ping* no nodo 2 (10.0.0.10/24) para o nodo 7 e para o nodo 9 (10.0.0.22/24). Podemos analisar que o nodo 7 apenas capta pacotes do protocolo ARP e ICMP quando o ping é feito somente para este nodo enquanto que quando é feito para outro nodo que não o 7 este apenas irá capturar um pacote do protocolo ARP, sendo que este pacote tem como destino todos os terminais com a intenção de descobrir onde se encontra a máquina para o qual pretende enviar as tramas do protocolo ICMP. Já quando fazemos a mesma análise que fizemos para o nodo 7 no nodo 9 (Figura 10), como este é ligado através de um *HUB* irá captar sempre todas os pacotes que sejam transmitidos na rede, tanto do protocolo ARP como do ICMP.

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	00:00:00_aa:00:00	Broadcast	ARP	42	Who has 10.0.0.20? Tell 10.0.0.10
2	0.000094	00:00:00_aa:00:02	00:00:00_aa:00:00	ARP	42	10.0.0.20 is at 00:00:00:aa:00:02
3	0.000127	10.0.0.10	10.0.0.20	ICMP	98	Echo (ping) request id=0x002a, seq=1/256, ttl=64
4	0.000159	10.0.0.20	10.0.0.10	ICMP	98	Echo (ping) reply id=0x002a, seq=1/256, ttl=64
5	1.000208	10.0.0.10	10.0.0.20	ICMP	98	Echo (ping) request id=0x002a, seq=2/512, ttl=64
6	1.000423	10.0.0.20	10.0.0.10	ICMP	98	Echo (ping) reply id=0x002a, seq=2/512, ttl=64
7	1.999306	10.0.0.10	10.0.0.20	ICMP	98	Echo (ping) request id=0x002a, seq=3/768, ttl=64
8	1.999473	10.0.0.20	10.0.0.10	ICMP	98	Echo (ping) reply id=0x002a, seq=3/768, ttl=64
9	5.007896	00:00:00_aa:00:02	00:00:00_aa:00:00	ARP	42	Who has 10.0.0.10? Tell 10.0.0.20
10	5.008002	00:00:00_aa:00:00	00:00:00_aa:00:02	ARP	42	10.0.0.10 is at 00:00:00:aa:00:00
11	9.013301	00:00:00_aa:00:00	Broadcast	ARP	42	Who has 10.0.0.22? Tell 10.0.0.10
12	9.013363	00:00:00_aa:00:04	00:00:00_aa:00:00	ARP	42	10.0.0.22 is at 00:00:00:aa:00:04
13	9.013413	10.0.0.10	10.0.0.22	ICMP	98	Echo (ping) request id=0x002b, seq=1/256, ttl=64
14	9.013445	10.0.0.22	10.0.0.10	ICMP	98	Echo (ping) reply id=0x002b, seq=1/256, ttl=64
15	10.012394	10.0.0.10	10.0.0.22	ICMP	98	Echo (ping) request id=0x002b, seq=2/512, ttl=64
16	10.012439	10.0.0.22	10.0.0.10	ICMP	98	Echo (ping) reply id=0x002b, seq=2/512, ttl=64
17	11.011409	10.0.0.10	10.0.0.22	ICMP	98	Echo (ping) request id=0x002b, seq=3/768, ttl=64
18	11.011451	10.0.0.22	10.0.0.10	ICMP	98	Echo (ping) reply id=0x002b, seq=3/768, ttl=64
19	14.015519	00:00:00_aa:00:04	00:00:00_aa:00:00	ARP	42	Who has 10.0.0.10? Tell 10.0.0.22
20	14.015572	00:00:00_aa:00:00	00:00:00_aa:00:04	ARP	42	10.0.0.10 is at 00:00:00:aa:00:00

Figura 9 - Capturas de tráfego no nodo 7 na terceira rede num nodo ligado por um *HUB*

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	00:00:00_aa:00:00	Broadcast	ARP	42	Who has 10.0.0.20? Tell 10.0.0.10
2	0.000058	00:00:00_aa:00:02	00:00:00_aa:00:00	ARP	42	10.0.0.20 is at 00:00:00:aa:00:02
3	0.000113	10.0.0.10	10.0.0.20	ICMP	98	Echo (ping) request id=0x002a, seq=1/256, ttl=64
4	0.000123	10.0.0.20	10.0.0.10	ICMP	98	Echo (ping) reply id=0x002a, seq=1/256, ttl=64
5	1.000289	10.0.0.10	10.0.0.20	ICMP	98	Echo (ping) request id=0x002a, seq=2/512, ttl=64
6	1.000338	10.0.0.20	10.0.0.10	ICMP	98	Echo (ping) reply id=0x002a, seq=2/512, ttl=64
7	1.999370	10.0.0.10	10.0.0.20	ICMP	98	Echo (ping) request id=0x002a, seq=3/768, ttl=64
8	1.999407	10.0.0.20	10.0.0.10	ICMP	98	Echo (ping) reply id=0x002a, seq=3/768, ttl=64
9	5.007691	00:00:00_aa:00:02	00:00:00_aa:00:00	ARP	42	Who has 10.0.0.10? Tell 10.0.0.20
10	5.007998	00:00:00_aa:00:00	00:00:00_aa:00:02	ARP	42	10.0.0.10 is at 00:00:00:aa:00:00
11	9.013314	00:00:00_aa:00:00	Broadcast	ARP	42	Who has 10.0.0.22? Tell 10.0.0.10

Figura 10 - Capturas de tráfego no nodo 7 na terceira rede num nodo ligado por um *SWITCH*

3. DHCP

Contrariamente ao que realizamos no primeiro exercício, neste caso vamos utilizar o protocolo DHCP (*Dynamic Host Configuration Protocol*), este protocolo facilita imenso o trabalho de quem configura uma rede, ou seja, este permite às máquinas obterem um endereço IP (*Internet Protocol*) automaticamente.

Imaginemos que éramos um admnistrador de uma rede em que estivessem ligados centenas de computadores, certamente não seria fácil atribuir um endereço IP a cada um deles. Para isto desenvolveram o protocolo DHCP, que por via de um servidor é capaz de distribuir automaticamente endereços IP às máquinas que solicitam conexão à rede em que se insere o servidor. Esta atribuição de endereços IP é feita através do estabelecimento de um tempo pré-definido para cada máquina que se liga à rede. Sempre que uma máquina se desconecte da rede o seu IP ficará livre para outras máquinas que se pretendam ligar à rede.

O protocolo DHCP utiliza o modelo cliente-servidor para isso tivemos que configurar os equipamentos associados à nossa topologia desenhada para este exercício, presente na Figura 11. O servidor DHCP encontra-se no nodo 1, ao qual lhe atribuímos o endereço IP 196.128.0.2/24. A endereço IP escolhido para endereçar os elementos desta rede foi o 196.128.0.0/24.

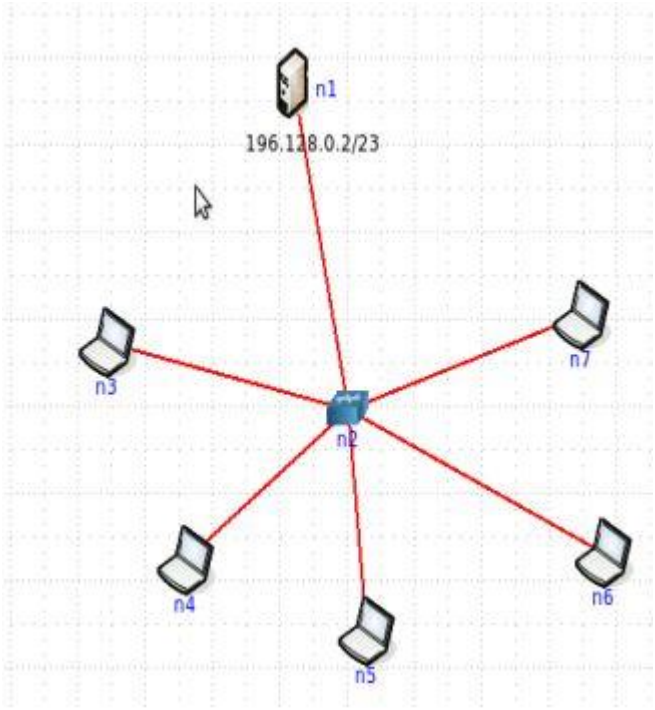


Figura 11 - Topologia em estrela seguindo o protocolo DHCP

Os restantes nodos são computadores (neste caso clientes DHCP) e estão ligados ao servidor através de um *HUB*. Em relação aos clientes deixamos permanecer as

configurações que estavam pré-definidas, podemos ver a sua configuração na Figura 12. Já o servidor tivemos que o configurar de forma a que fosse possível a atribuição automática de endereços IP ligados a esta rede. Na Figura 13 podemos encontrar as configurações que foram feitas dentro do servidor (nodo 1), os comentários explicam o que foi configurado.

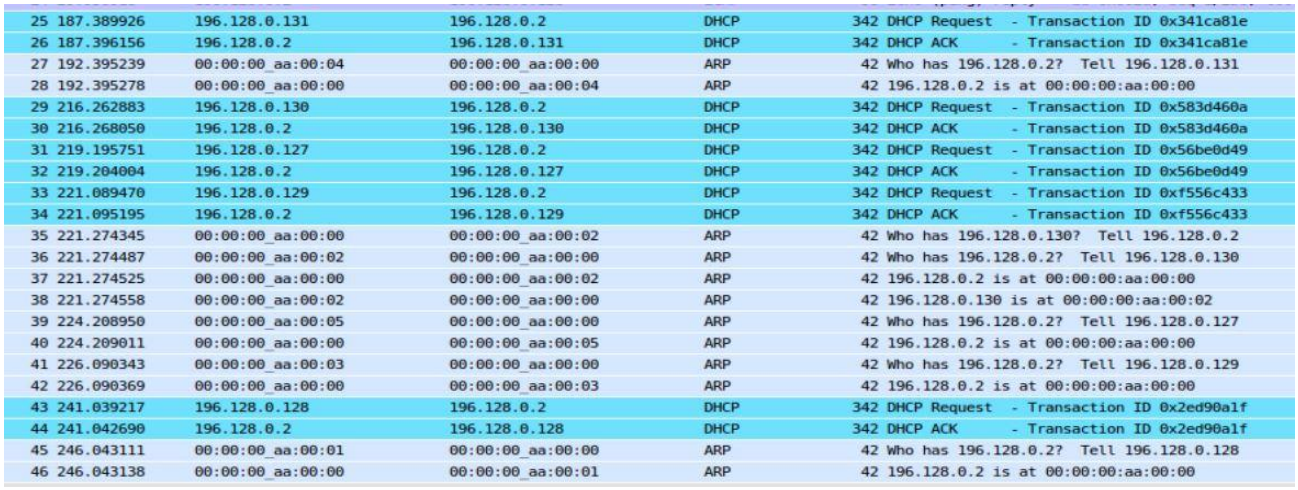
```
#!/bin/sh
# auto-generated by DHCPClient service (utility.py)
# uncomment this mkdir line and symlink line to enable client-side DNS
# resolution based on the DHCP server response.

/sbin/dhclient -nw -pf /var/run/dhclient-eth0.pid -lf /var/run/dhclient-eth0.lea
se eth0
```

Figura 12 - Configuração do cliente DHCP

```
log-facility local6;
#tempo pré-definido para ser atribuído um IP a uma maquina - 600 seg
default-lease-time 600;

#tempo máximo atribuído a uma máquina caso esta pretenda utilizar mais tempo do
#que lhe foi atribuído previamente
max-lease-time 7200;

#desativar o DDNS
ddns-update-style none;

#lista a subrede a qual o servidor pertence e a máscara de rede utilizada
subnet 196.128.0.0 netmask 255.255.255.0 {
  pool {
    #gama válida de endereços IP que se pode atribuir aos clientes(196.128.0.127
    #-254)
    range 196.128.0.127 196.128.0.254;

    default-lease-time 600;

    #definir o IP do servidor que gere a resolução de atribuição de IPs
    option domain-name-servers 196.128.0.2;

    #informar o cliente o IP do equipamento que é o gateway da sua rede
    option routers 196.128.0.1;
  }
}
```

Figura 13 - Configuração do servidor DHCP

De seguida efetuamos a captura de tráfego no servidor DHCP (nodo 1), e através dos diversos computadores ligados à rede executamos o comando *ping* para o endereço IP do servidor (196.128.0.2/24). Verificamos que a rede ficou bem configurada e que são atribuídos endereços IP às máquinas ligadas á rede a partir do endereço 196.128.0.127 até ao 196.128.0.131. A Figura 14 permite-nos visualizar a captura de tráfego realizado após uns momentos de termos executado o comando *ping* nos computadores. No nosso caso configuramos o servidor num modo de funcionamento dinâmico em que sempre que um cliente pedir uma ligação à rede é atribuído um endereço IP a esse cliente, porém a sua ligação á rede será limitada por um período de tempo definido por nós (neste caso 600 seg), caso o cliente pretenda utilizar mais tempo de ligação este prazo será extendido para 7200 seg.



25 187.389926	196.128.0.131	196.128.0.2	DHCP	342	DHCP Request - Transaction ID 0x341ca81e
26 187.396156	196.128.0.2	196.128.0.131	DHCP	342	DHCP ACK - Transaction ID 0x341ca81e
27 192.395239	00:00:00_aa:00:04	00:00:00_aa:00:00	ARP	42	Who has 196.128.0.2? Tell 196.128.0.131
28 192.395278	00:00:00_aa:00:00	00:00:00_aa:00:04	ARP	42	196.128.0.2 is at 00:00:00:aa:00:00
29 216.262883	196.128.0.130	196.128.0.2	DHCP	342	DHCP Request - Transaction ID 0x583d460a
30 216.268050	196.128.0.2	196.128.0.130	DHCP	342	DHCP ACK - Transaction ID 0x583d460a
31 219.195751	196.128.0.127	196.128.0.2	DHCP	342	DHCP Request - Transaction ID 0x56be0d49
32 219.204004	196.128.0.2	196.128.0.127	DHCP	342	DHCP ACK - Transaction ID 0x56be0d49
33 221.089470	196.128.0.129	196.128.0.2	DHCP	342	DHCP Request - Transaction ID 0xf556c433
34 221.095195	196.128.0.2	196.128.0.129	DHCP	342	DHCP ACK - Transaction ID 0xf556c433
35 221.274345	00:00:00_aa:00:00	00:00:00_aa:00:02	ARP	42	Who has 196.128.0.130? Tell 196.128.0.2
36 221.274487	00:00:00_aa:00:02	00:00:00_aa:00:00	ARP	42	Who has 196.128.0.2? Tell 196.128.0.130
37 221.274525	00:00:00_aa:00:00	00:00:00_aa:00:02	ARP	42	196.128.0.2 is at 00:00:00:aa:00:00
38 221.274558	00:00:00_aa:00:02	00:00:00_aa:00:00	ARP	42	196.128.0.130 is at 00:00:00:aa:00:02
39 224.208950	00:00:00_aa:00:05	00:00:00_aa:00:00	ARP	42	Who has 196.128.0.2? Tell 196.128.0.127
40 224.209011	00:00:00_aa:00:00	00:00:00_aa:00:05	ARP	42	196.128.0.2 is at 00:00:00:aa:00:00
41 226.090343	00:00:00_aa:00:03	00:00:00_aa:00:00	ARP	42	Who has 196.128.0.2? Tell 196.128.0.129
42 226.090369	00:00:00_aa:00:00	00:00:00_aa:00:03	ARP	42	196.128.0.2 is at 00:00:00:aa:00:00
43 241.039217	196.128.0.128	196.128.0.2	DHCP	342	DHCP Request - Transaction ID 0x2ed90a1f
44 241.042690	196.128.0.2	196.128.0.128	DHCP	342	DHCP ACK - Transaction ID 0x2ed90a1f
45 246.043111	00:00:00_aa:00:01	00:00:00_aa:00:00	ARP	42	Who has 196.128.0.2? Tell 196.128.0.128
46 246.043138	00:00:00_aa:00:00	00:00:00_aa:00:01	ARP	42	196.128.0.2 is at 00:00:00:aa:00:00

Figura 14 - Captura de tráfego a partir do servidor DHCP

Com este exercício conseguimos compreender como é que funciona o protocolo DHCP, resumidamente, este utiliza um modelo cliente-servidor e como tal desempenha as seguintes tarefas:

1. Quando um cliente (terminal) se liga à rede envia um pacote com um pedido de configurações DHCP em *broadcast*, ou seja, para todas as máquinas ligadas à rede – DHCP *Discover* ;
2. O servidor gere uma gama disponível de endereços IP que tem para atribuir e as suas respetivas configurações (gateway padrão, DNS – *Domain Name System*, tempo limite de ligação à rede, etc.). Este envia para o cliente as suas propostas de configuração de rede – DHCP *Offer* ;
3. O cliente comunica novamente em *broadcast* a informar o servidor que aceita a proposta feita anteriormente – DHCP *Request* ;
4. Quando o servidor recebe um pedido de configurações DHCP este responde com a confirmação da reserva de um dos endereços disponíveis e as suas respetivas configurações – DHCP *ACK*.

A utilização de um servidor DHCP traz vantagens tais como:

- Fiabilidade na configuração de endereços IP – o DHCP minimiza erros de configuração causados pela configuração manual de IP's das máquinas envolvidas na rede;
- Reduz a carga de trabalho ao admnistrador da rede – permite que a configuração TCP/IP seja toda automática e centralizada toda num ou mais servidores, a possibilidade de definir um intervalo de endereços IP a atribuir às várias máquinas que se liguem à rede, etc.

4. Interligação de Redes

Neste exercício o grupo interligou as redes anteriormente desenhadas através de *routers*. Foi estabelecido um esquema de endereçamento como podemos ver na Figura 15. As redes A, B e C são as topologias desenvolvidas para o primeiro exercício e a rede D criada no segundo exercício.

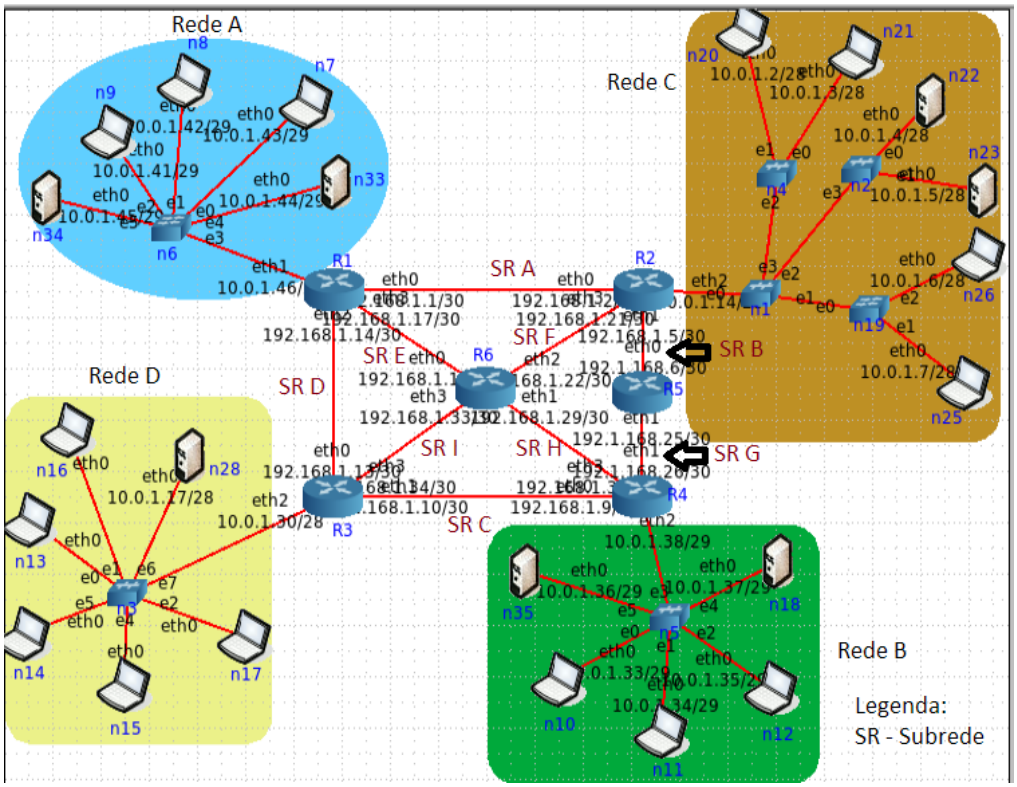


Figura 15 - Interligação das várias topologias de redes locais

Como nos foi solicitado no enunciado os routers estão interligados por uma sub-rede com uma máscara de 30 bits da rede 192.168.1.0/30. As redes locais foram endereçadas a partir do endereço 192.168.1.0/28, sendo que a rede A encontra-se no endereço 10.0.1.40/29, a rede B em 10.0.1.32/29, a rede C no endereço 10.0.1.0/28 e a rede D no endereço 10.0.1.16/28. Para entender melhor o esquema de endereçamento desenhado pelo grupo as tabelas seguintes descrevem todo a atribuição de endereços IP às várias redes locais e para as subredes que interligam os routers.

Tabelas de Endereçamento

Redes Locais

Tabela 1 - Tabela de Encaminhamento para as Redes Locais

Rede Local	End. Rede	End. Difusão	Máscara	End. Válidos (Desde – Até)
A	10.0.1.40	10.0.1.47	255.255.255.248	10.0.1.41 – 46
B	10.0.1.32	10.0.1.39	255.255.255.248	10.0.1.33 – 38
C	10.0.1.0	10.0.1.15	255.255.255.240	10.0.1.1 – 14
D	10.0.1.16	10.0.1.31	255.255.255.240	10.0.1.17-30

Sub-redes entre Routers

Tabela 2 - Tabela de Encaminhamento para as Subredes que interligam os routers

Sub-rede	End. Rede	End. Difusão	Máscara	End. Válidos (Desde – Até)
A	192.168.1.0	192.168.1.3	255.255.255.192	192.168.1.1 - 2
B	192.168.1.4	192.168.1.7	255.255.255.192	192.168.1.5 – 6
C	192.168.1.8	192.168.1.11	255.255.255.192	192.168.1.9 – 10
D	192.168.1.12	192.168.1.15	255.255.255.192	192.168.1.13 – 14
E	192.168.1.16	192.168.1.19	255.255.255.192	192.168.1.17 – 18
F	192.168.1.20	192.168.1.23	255.255.255.192	192.168.1.21 – 22
G	192.168.1.24	192.168.1.27	255.255.255.192	192.168.1.25 – 26
H	192.168.1.28	192.168.1.31	255.255.255.192	192.168.1.29 – 30
I	192.168.1.32	192.168.1.35	255.255.255.192	192.168.1.33 - 34

As próximas tabelas servem para indicar o encaminhamento atribuído aos vários routers, estes é que definem as rotas entre as várias topologias.

Tabela de Encaminhamento

Tabela 3 - Tabela de encaminhamento para os vários *routers* da topologia

Router	Rede Destino	Máscara	Interface de Saída	Próximo nó
R1	10.0.1.0	255.255.255.240	192.168.1.2	192.168.1.1
R1	10.0.1.16	255.255.255.240	192.168.1.13	192.168.1.14
R1	10.0.1.32	255.255.255.248	192.168.1.18	192.168.1.19
R2	10.0.1.16	255.255.255.240	192.168.1.22	192.168.1.21
R2	10.0.1.32	255.255.255.248	192.168.1.6	192.168.1.5
R2	10.0.1.40	255.255.255.248	192.168.1.1	192.168.1.2
R3	10.0.1.0	255.255.255.240	192.168.1.33	192.168.1.34
R3	10.0.1.32	255.255.255.248	192.168.1.9	192.168.1.10
R3	10.0.1.40	255.255.255.248	192.168.1.14	192.168.1.13
R4	10.0.1.0	255.255.255.240	192.168.1.25	192.168.1.26
R4	10.0.1.16	255.255.255.240	192.168.1.10	192.168.1.9
R4	10.0.1.40	255.255.255.248	192.168.1.29	192.168.1.30
R5	10.0.1.0	255.255.255.240	192.168.1.5	192.168.1.6
R5	10.0.1.16	255.255.255.240	192.168.1.26	192.168.1.25
R5	10.0.1.32	255.255.255.248	192.168.1.26	192.168.1.25

R5	10.0.1.40	255.255.255.248	192.168.1.5	192.168.1.6
R6	10.0.1.0	255.255.255.240	192.168.1.21	192.168.1.22
R6	10.0.1.16	255.255.255.240	192.168.1.34	192.168.1.33
R6	10.0.1.32	255.255.255.248	192.168.1.30	192.168.1.29
R6	10.0.1.40	255.255.255.248	192.168.1.17	192.168.1.18

Na Figura 16 podemos verificar as configurações que fizemos no *router* 3 (R3), em que configuramos as rotas que os pacotes que tenham como destino as redes A,B e C deverão seguir, utilizando o comando *ip route*. Ignoramos no caso de R3 a rede D pois o *router* está ligado a esta rede. Fizemos semelhante para os restantes routers, apenas não os apresentamos neste relatório para não o tornar demasiado extenso.

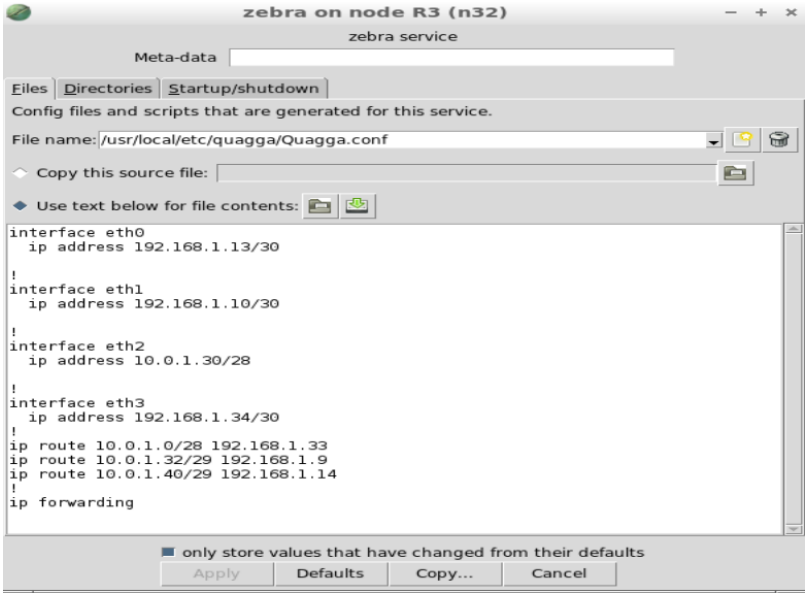


Figura 16 - Configuração do Router 3

Também o poderíamos fazer manualmente com o programa Quagga. Para verificar as configurações de um Router, corremos em modo de execução o CORE e abrimos um terminal *vtysh* para executar o Quagga (exemplo na Figura 17 com R1) e de seguida executamos o comando *show running-config* e o terminal lista-nos todas as configurações do router. Para podermos definir a rota de um pacote que passe por este router temos que configurar o router, por exemplo vamos definir que os pacotes provenientes da rede 10.0.1.0/28 deverão ser encaminhados para o endereço 192.168.1.2 que é o endereço da interface *ethernet0* de R2 então executamos os seguintes comandos:

#configure terminal

#ip route 10.0.1.0/28 192.168.1.2

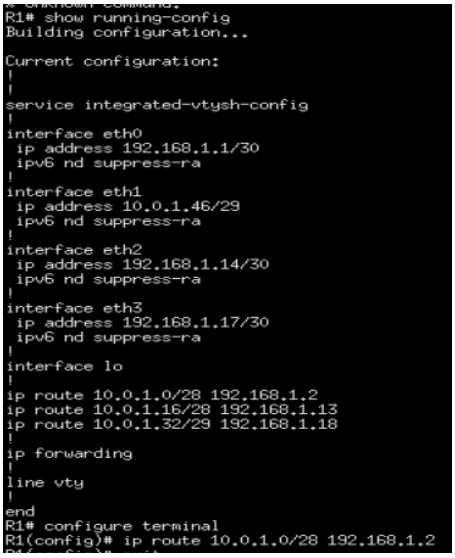


Figura 17 - Configuração manual das rotas em R1

Podemos verificar como configurar um terminal de uma rede local em relação ao endereço atribuído à interface *ethernet* que liga a rede local ao router, neste caso apresentamos as configurações para um terminal presente na rede D (10.0.1.17/28) em que configuramos com o endereço da interface ethernet (10.0.1.30/28), tal podemos visualizar na Figura 18.

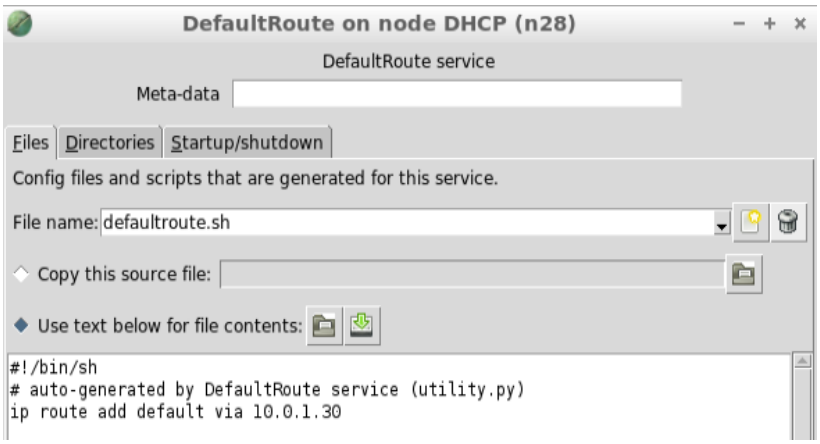


Figura 18 - Configuração do terminal n28

Após termos configurado todos os elementos da topologia que interliga as várias redes locais efetuamos testes através dos comandos *ping* e *traceroute* em vários terminais das várias redes locais para analisar se as configurações feitas anteriormente permitem a boa ligação entre as redes locais. Na Figura 19 podemos ver o resultado dos testes realizados.

Rede A (n8) -> Rede B (n12)

```
root@n8:/tmp/pycore.47275/n8.conf# ping 10.0.1.35
PING 10.0.1.35 (10.0.1.35) 56(84) bytes of data.
64 bytes from 10.0.1.35: icmp_req=1 ttl=61 time=0.299 ms
64 bytes from 10.0.1.35: icmp_req=2 ttl=61 time=0.150 ms
64 bytes from 10.0.1.35: icmp_req=3 ttl=61 time=0.136 ms
^C
--- 10.0.1.35 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 1999ms
rtt min/avg/max/mdev = 0.136/0.195/0.299/0.073 ms
root@n8:/tmp/pycore.47275/n8.conf# traceroute 10.0.1.35
traceroute to 10.0.1.35 (10.0.1.35), 30 hops max, 60 byte packets
 1  10.0.1.46 (10.0.1.46)  0.051 ms  0.017 ms  0.016 ms
 2  192.168.1.18 (192.168.1.18)  0.042 ms  0.026 ms  0.025 ms
 3  * * *
 4  * * *
 5  * * *
 6  * * *
 7  * * *
 8  * 10.0.1.35 (10.0.1.35)  0.243 ms  0.091 ms
root@n8:/tmp/pycore.47275/n8.conf#
```

Rede A (n9) -> Rede D (DHCP Server)

```
root@n9:/tmp/pycore.47275/n9.conf# ping 10.0.1.17
PING 10.0.1.17 (10.0.1.17) 56(84) bytes of data.
64 bytes from 10.0.1.17: icmp_req=1 ttl=62 time=0.313 ms
64 bytes from 10.0.1.17: icmp_req=2 ttl=62 time=0.244 ms
64 bytes from 10.0.1.17: icmp_req=3 ttl=62 time=0.181 ms
^C
--- 10.0.1.17 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 1999ms
rtt min/avg/max/mdev = 0.181/0.246/0.313/0.053 ms
root@n9:/tmp/pycore.47275/n9.conf# traceroute 10.0.1.17
traceroute to 10.0.1.17 (10.0.1.17), 30 hops max, 60 byte packets
 1  10.0.1.46 (10.0.1.46)  0.053 ms  0.016 ms  0.016 ms
 2  192.168.1.13 (192.168.1.13)  0.041 ms  0.026 ms  0.025 ms
 3  * * *
 4  * * *
 5  * * *
 6  * * *
 7  * * *
 8  * 10.0.1.17 (10.0.1.17)  0.119 ms  0.067 ms
root@n9:/tmp/pycore.47275/n9.conf#
```

Rede B (n10) -> Rede C (n21)

```
root@n10:/tmp/pycore.47275/n10.conf# ping 10.0.1.3
PING 10.0.1.3 (10.0.1.3) 56(84) bytes of data.
64 bytes from 10.0.1.3: icmp_req=1 ttl=61 time=0.438 ms
64 bytes from 10.0.1.3: icmp_req=2 ttl=61 time=0.220 ms
64 bytes from 10.0.1.3: icmp_req=3 ttl=61 time=0.342 ms
64 bytes from 10.0.1.3: icmp_req=4 ttl=61 time=0.180 ms
^C
--- 10.0.1.3 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3000ms
rtt min/avg/max/mdev = 0.180/0.295/0.438/0.101 ms
root@n10:/tmp/pycore.47275/n10.conf# traceroute 10.0.1.3
traceroute to 10.0.1.3 (10.0.1.3), 30 hops max, 60 byte packets
 1  10.0.1.38 (10.0.1.38)  0.083 ms  0.041 ms  0.039 ms
 2  192.168.1.25 (192.168.1.25)  0.065 ms  0.051 ms  0.050 ms
 3  * * *
 4  * * *
 5  * * *
 6  * * *
 7  * * *
 8  * 10.0.1.3 (10.0.1.3)  0.289 ms  0.213 ms
root@n10:/tmp/pycore.47275/n10.conf#
```

Rede C (n25) -> Rede D (DHCP Server)

```
root@n25:/tmp/pycore.47275/n25.conf# ping 10.0.1.17
PING 10.0.1.17 (10.0.1.17) 56(84) bytes of data.
64 bytes from 10.0.1.17: icmp_req=1 ttl=61 time=0.547 ms
64 bytes from 10.0.1.17: icmp_req=2 ttl=61 time=6.97 ms
64 bytes from 10.0.1.17: icmp_req=3 ttl=61 time=0.314 ms
^C
--- 10.0.1.17 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2003ms
rtt min/avg/max/mdev = 0.314/2.613/6.979/3.088 ms
root@n25:/tmp/pycore.47275/n25.conf# traceroute 10.0.1.17
traceroute to 10.0.1.17 (10.0.1.17), 30 hops max, 60 byte packets
 1  10.0.1.14 (10.0.1.14)  0.112 ms  0.083 ms  0.066 ms
 2  192.168.1.22 (192.168.1.22)  0.083 ms  0.073 ms  0.067 ms
 3  * * *
 4  * * *
 5  * * *
 6  * * *
 7  * * *
 8  * 10.0.1.17 (10.0.1.17)  0.469 ms  0.264 ms
root@n25:/tmp/pycore.47275/n25.conf#
```

Figura 19 -Testes de conectividade entre as várias redes locais com os comandos ping e traceroute

Com estes testes de conectividade podemos reter que a rede foi bem configurada e que está a funcionar devidamente.

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

Para este exercício o grupo decidiu implementar um servidor FTP (*File Transfer Protocol*) num nodo da rede B com o endereço 10.0.1.36/29 e um servidor HTTP (Hypertext Transfer Protocol) num nodo da rede A com o endereço 10.0.1.44/29. As suas configurações podem ser vistas na Figura 20.

Servidor FTP

Config files and scripts that are generated for this service.

File name: vsftpd.conf

Copy this source file:

Use text below for file contents:

```
# vsftpd.conf auto-generated by FTP service (utility.py)
listen=YES
anonymous_enable=YES
local_enable=YES
write_enable=YES
chroot_local_user=NO
dirmessage_enable=YES
use_localtime=YES
xferlog_enable=YES
connect_from_port_20=YES
xferlog_file=/var/log/vsftpd.log
ftpd_banner=Servidor MIETI Redes Computadores 1
secure_chroot_dir=/var/run/vsftpd/empty
anon_root=/var/ftp
```

Servidor HTTP

Config files and scripts that are generated for this service.

File name: /etc/apache2/apache2.conf

Copy this source file:

Use text below for file contents:

```
# apache2.conf generated by utility.py:HttpService
LockFile ${APACHE_LOCK_DIR}/accept.lock
PidFile ${APACHE_PID_FILE}
Timeout 300
KeepAlive On
MaxKeepAliveRequests 100
KeepAliveTimeout 5

<IfModule mpm_prefork_module>
    StartServers          5
    MinSpareServers       5
    MaxSpareServers      10
    MaxClients          150
    MaxRequestsPerChild   0
</IfModule>

<IfModule mpm_worker_module>
    StartServers          2
    MinSpareThreads      25
    MaxSpareThreads      75
    ThreadLimit          64
    ThreadsPerChild      25
```

Figura 20 - Configurações nos servidores FTP e HTTP

De forma a que os servidores funcionassem da maneira correta foi necessário configurar quais os comandos de arranque e término para cada um deles. Na Figura 21 encontramos os comandos que definimos para cada um dos servidores.

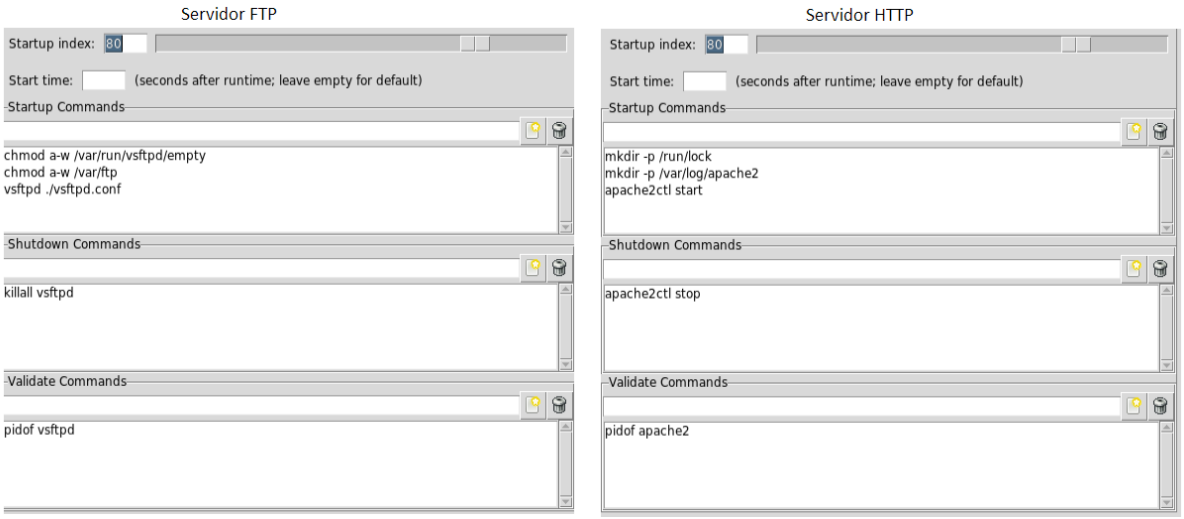


Figura 21 -Configuração dos comandos de arranque e término nos servidores FTP e HTTP

De seguida analisamos se o servidor FTP estava a funcionar devidamente, através de um terminal de outra rede local (n25 da rede C) como podemos ver na Figura 22, depois de inserirmos os dados de *login* efetuamos o comando *ls* para listar quais os ficheiros presentes no servidor.

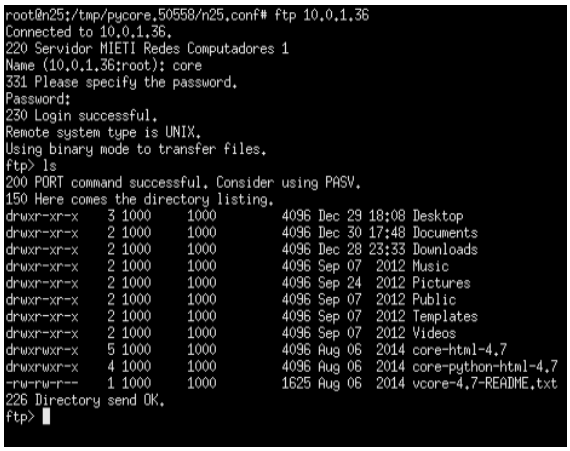


Figura 22 - Conexão ao servidor FTP a partir do nodo 25

Logo após, efetuamos o comando *wget –S 10.0.1.44/29* a partir do nodo 18 da rede B para analisar se o servidor HTTP estava a responder conforme o previsto, tal se confirmou como podemos comprovar na Figura 23.

Figura 23 - Conexão ao servidor HTTP a partir do nodo 18

Simultaneamente aos comandos executados para os dois servidores foi feita uma captura de tráfego destes dois servidores. No caso do servidor FTP, aplicámos um filtro para só capturar os pacotes do protocolo FTP mas analisamos que este funciona em parceria com o protocolo TCP (Transmission Control Protocol). Na Figura 24 encontramos a captura de tráfego após a execução do comando *ls* no servidor FTP a partir do nodo 25 (10.0.1.7/28).

No.	Time	Source	Destination	Protocol	Length	Info
615	136.924312	10.0.1.7	10.0.1.36	FTP	91	[TCP Retransmission] Request: PORT 10,0,1,7,223,134
616	136.924328	10.0.1.7	10.0.1.36	FTP	91	[TCP Retransmission] Request: PORT 10,0,1,7,223,134
617	136.924332	10.0.1.7	10.0.1.36	FTP	91	[TCP Retransmission] Request: PORT 10,0,1,7,223,134
618	136.924317	10.0.1.7	10.0.1.36	FTP	91	[TCP Retransmission] Request: PORT 10,0,1,7,223,134
619	136.924339	10.0.1.7	10.0.1.36	FTP	91	[TCP Retransmission] Request: PORT 10,0,1,7,223,134
620	136.924343	10.0.1.7	10.0.1.36	FTP	91	[TCP Retransmission] Request: PORT 10,0,1,7,223,134
621	136.924352	10.0.1.7	10.0.1.36	FTP	91	[TCP Retransmission] Request: PORT 10,0,1,7,223,134
622	136.924360	10.0.1.7	10.0.1.36	FTP	91	[TCP Retransmission] Request: PORT 10,0,1,7,223,134
623	136.924368	10.0.1.7	10.0.1.36	FTP	91	[TCP Retransmission] Request: PORT 10,0,1,7,223,134
624	136.924374	10.0.1.7	10.0.1.36	FTP	91	[TCP Retransmission] Request: PORT 10,0,1,7,223,134
625	136.924381	10.0.1.7	10.0.1.36	FTP	91	[TCP Retransmission] Request: PORT 10,0,1,7,223,134
626	136.924387	10.0.1.7	10.0.1.36	FTP	91	[TCP Retransmission] Request: PORT 10,0,1,7,223,134
627	136.924391	10.0.1.7	10.0.1.36	FTP	91	[TCP Retransmission] Request: PORT 10,0,1,7,223,134
628	136.924395	10.0.1.7	10.0.1.36	FTP	91	[TCP Retransmission] Request: PORT 10,0,1,7,223,134
629	136.924399	10.0.1.7	10.0.1.36	FTP	91	[TCP Retransmission] Request: PORT 10,0,1,7,223,134
630	136.924402	10.0.1.7	10.0.1.36	FTP	91	[TCP Retransmission] Request: PORT 10,0,1,7,223,134
631	136.924485	10.0.1.36	10.0.1.7	FTP	119	Response: 200 PORT command successful. Consider using PASV.
632	136.924491	10.0.1.36	10.0.1.7	FTP	119	[TCP Retransmission] Response: 200 PORT command successful. Consider using PASV.
633	136.924495	10.0.1.36	10.0.1.7	FTP	119	[TCP Retransmission] Response: 200 PORT command successful. Consider using PASV.
634	136.924520	10.0.1.36	10.0.1.7	FTP	119	[TCP Retransmission] Response: 200 PORT command successful. Consider using PASV.
635	136.924527	10.0.1.36	10.0.1.7	FTP	119	[TCP Retransmission] Response: 200 PORT command successful. Consider using PASV.
636	136.924531	10.0.1.36	10.0.1.7	FTP	119	[TCP Retransmission] Response: 200 PORT command successful. Consider using PASV.
637	136.924539	10.0.1.36	10.0.1.7	FTP	119	[TCP Retransmission] Response: 200 PORT command successful. Consider using PASV.
638	136.924546	10.0.1.36	10.0.1.7	FTP	119	[TCP Retransmission] Response: 200 PORT command successful. Consider using PASV.
639	136.924551	10.0.1.36	10.0.1.7	FTP	119	[TCP Retransmission] Response: 200 PORT command successful. Consider using PASV.

Figura 24 - Captura de tráfego no servidor FTP após execução do comando ls

De seguida analisamos da mesma forma o servidor HTTP, neste caso apresentamos na Figura 25 a captura de tráfego deste após a execução do comando *wget –S 10.0.1.44* no nodo 25 (10.0.1.7/28). Também aplicamos um filtro para só receber pacotes do protocolo HTTP, verificamos que também é utilizado em conjunto o protocolo TCP. O servidor recebe primeiro os pacotes TCP e depois os pacotes HTTP.

No.	Time	Source	Destination	Protocol	Length	Info
63	1.498359	10.0.1.7	10.0.1.44	HTTP	177	GET / HTTP/1.1
64	1.498364	10.0.1.7	10.0.1.44	HTTP	177	[TCP Retransmission] GET / HTTP/1.1
65	1.498365	10.0.1.7	10.0.1.44	HTTP	177	[TCP Retransmission] GET / HTTP/1.1
66	1.498368	10.0.1.7	10.0.1.44	HTTP	177	[TCP Retransmission] GET / HTTP/1.1
67	1.498369	10.0.1.7	10.0.1.44	HTTP	177	[TCP Retransmission] GET / HTTP/1.1
68	1.498371	10.0.1.7	10.0.1.44	HTTP	177	[TCP Retransmission] GET / HTTP/1.1
69	1.498368	10.0.1.7	10.0.1.44	HTTP	177	[TCP Retransmission] GET / HTTP/1.1
70	1.498374	10.0.1.7	10.0.1.44	HTTP	177	[TCP Retransmission] GET / HTTP/1.1
71	1.498376	10.0.1.7	10.0.1.44	HTTP	177	[TCP Retransmission] GET / HTTP/1.1
72	1.498370	10.0.1.7	10.0.1.44	HTTP	177	[TCP Retransmission] GET / HTTP/1.1
73	1.498379	10.0.1.7	10.0.1.44	HTTP	177	[TCP Retransmission] GET / HTTP/1.1
74	1.498380	10.0.1.7	10.0.1.44	HTTP	177	[TCP Retransmission] GET / HTTP/1.1
75	1.498383	10.0.1.7	10.0.1.44	HTTP	177	[TCP Retransmission] GET / HTTP/1.1
76	1.498386	10.0.1.7	10.0.1.44	HTTP	177	[TCP Retransmission] GET / HTTP/1.1
77	1.498389	10.0.1.7	10.0.1.44	HTTP	177	[TCP Retransmission] GET / HTTP/1.1
78	1.498391	10.0.1.7	10.0.1.44	HTTP	177	[TCP Retransmission] GET / HTTP/1.1
94	1.498658	10.0.1.44	10.0.1.7	HTTP	595	HTTP/1.1 200 OK
95	1.498666	10.0.1.44	10.0.1.7	HTTP	595	[TCP Retransmission] HTTP/1.1 200 OK
96	1.498670	10.0.1.44	10.0.1.7	HTTP	595	[TCP Retransmission] HTTP/1.1 200 OK
97	1.498672	10.0.1.44	10.0.1.7	HTTP	595	[TCP Retransmission] HTTP/1.1 200 OK
98	1.498675	10.0.1.44	10.0.1.7	HTTP	595	[TCP Retransmission] HTTP/1.1 200 OK
99	1.498678	10.0.1.44	10.0.1.7	HTTP	595	[TCP Retransmission] HTTP/1.1 200 OK
100	1.498680	10.0.1.44	10.0.1.7	HTTP	595	[TCP Retransmission] HTTP/1.1 200 OK
101	1.498682	10.0.1.44	10.0.1.7	HTTP	595	[TCP Retransmission] HTTP/1.1 200 OK

Figura 25 - Captura de tráfego no servidor HTTP após execução do comando wget

6. Interligação via NAT

Neste último exercício foi-nos pedido que implementássemos uma rede privada (local) associada à topologia desenvolvida no exercício anterior, rede essa que deve usar endereços privados da gama 192.168.2.0/24. Essa rede, para ter conectividade, terá que estar ligada a uma das redes LAN anteriormente utilizadas através de um router NAT (*Network Address Translation*). A rede NAT foi implementada na Rede A (10.0.1.40/29) e o endereço atribuído à interface que liga o *router* NAT à rede local foi o 10.0.1.42/29. A Figura 26 demonstra o que foi incrementado às topologias anteriores.

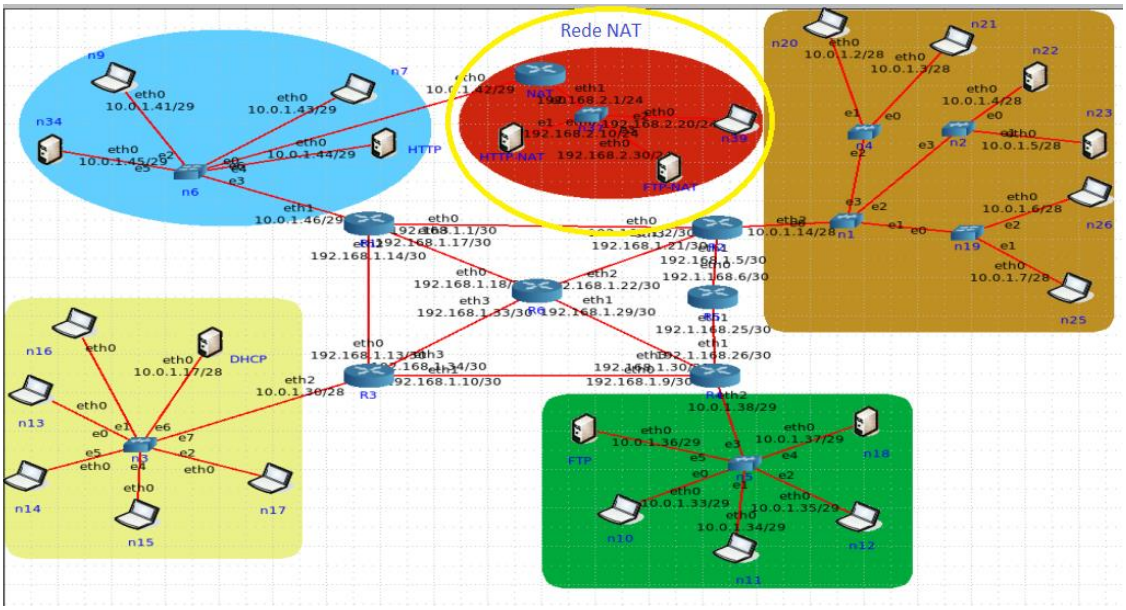


Figura 26 - Rede NAT implementada na Rede Local A

De seguida configuramos os servidores HTTP e FTP da mesma forma que fizemos no exercício anterior. Configuramos ainda o encaminhamento para o *router* NAT como podemos verificar na Figura 27.

```
interface eth0
  ip address 10.0.1.42/29

!
interface eth1
  ip address 192.168.2.1/24

!

ip route 10.0.1.0/28 10.0.1.46
ip route 10.0.1.16/28 10.0.1.46
ip route 10.0.1.32/29 10.0.1.46
!
ip forwarding
```

Figura 27 - Encaminhamento do *Router* NAT

Posto isto é necessário configurar o *Firewall* (Figura 28), recorrendo ao mecanismo *ip tables*, este oferece normas de roteamento e encaminhamento pronta para serem utilizadas, prevenindo o utilização indevidas da rede.

```
iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE
iptables -A FORWARD -i eth1 -j ACCEPT
iptables -A FORWARD -o eth1 -j ACCEPT
iptables -t nat -A PREROUTING -i eth0 -p tcp --dport 80 -j DNAT \
            --to-destination 192.168.2.10:80
iptables -t nat -A PREROUTING -i eth0 -p tcp --dport 21 -j DNAT \
            --to-destination 192.168.2.30:21
```

Figura 28 - Configuração do *Firewall* no *Router* NAT

O primeiro comando (*ip tables –t nat –A POSTROUTING –o eht0 – j MASQUERADE*) é utilizado para que os nodos da rede privada possam comunicar com as outras redes públicas existentes na topologia da Figura 26. O *POSTROUTING* (pós-roteamento) permite que os pacotes que passam pela *router* possam ser alterado mal deixem o dispositivo externo do *Firewall*. Já o *MASQUERADE* é utilizado para mascarar o endereço IP de um nodo da rede privada com o endereço IP externo do *router* NAT (10.0.1.42/29).

Seguidamente aplicamos a norma *FORWARD*, que permite-nos fazer o encaminhamento da rede privada para a LAN à qual está interligada pela interface *eth0*. Os últimos dois comandos aplicados permitem disponibizar os servidores FTP e HTTP externamente, para isso especifcicamos o endereço IP para qual este deve ser encaminhado e qual a porta relacionada com o servidor. Por exemplo, todos as ligações HTTP para a porta 80 serão encaminhadas para o endereço IP da rede privada 192.168.2.10. No caso do servidor FTP a porta utilizada é a 21 e o endereço IP destino é o 192.168.2.30.

No final de configurar tudo, para analisarmos que a rede privada foi bem configurada executamos um *ping* externo para dentro da rede (nodo 26 para o nodo 39), para analisar que a rede não permite a comunicação proveniente de redes externas para os nodos que

não pretende estabelecer comunicação. De seguida executamos os comandos *ping* e *traceroute* a partir do nodo 39 para nodos das várias redes locais pertencentes à topologia projetada e confirmamos que a rede está bem configurada. Estes teste encontram-se na Figura 29.

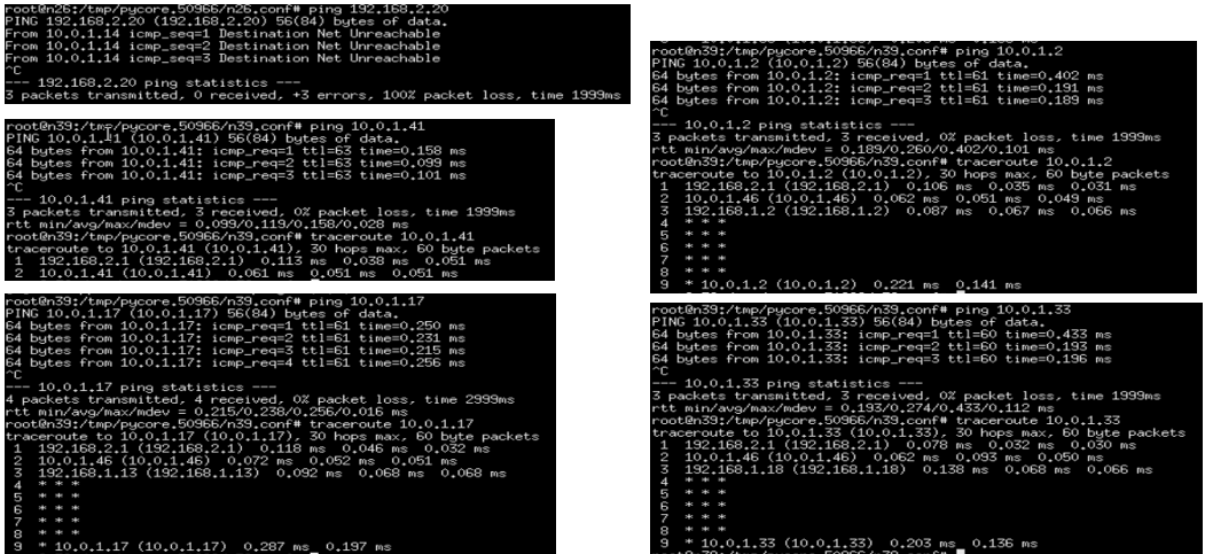


Figura 29 - Verificação do correto funcionamento da rede privada através dos comandos *ping* e *traceroute*

Ainda realizamos a captura do tráfego no *Router* NAT através do *Wireshark* após a execução dos comandos anteriormente utilizados para lidar com os servidores HTTP e FTP. Os resultados destas capturas encontram-se nas Figuras 30 e 31, e assim concluímos o último exercício deste Trabalho Prático.

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	00:00:00 aa:00:2c	Broadcast	ARP	42	Who has 192.168.2.30? Tell 192.168.2.1
2	0.000047	00:00:00_aa:00:2f	00:00:00_aa:00:2c	ARP	42	192.168.2.30 is at 00:00:00:aa:00:2f
3	0.000054	10.0.1.43	192.168.2.30	TCP	74	42423 > ftp [SYN] Seq=0 Win=14600 Len=0 MSS=14
4	0.000100	192.168.2.30	10.0.1.43	TCP	74	ftp > 42423 [SYN, ACK] Seq=0 Ack=1 Win=14480 L
5	0.000147	10.0.1.43	192.168.2.30	TCP	66	42423 > ftp [ACK] Seq=1 Ack=1 Win=14608 Len=0
6	0.005443	192.168.2.30	10.0.1.43	FTP	89	Response: 220 Servidor Rede NAT
7	0.005683	10.0.1.43	192.168.2.30	TCP	66	42423 > ftp [ACK] Seq=1 Ack=24 Win=14608 Len=0
8	2.372522	10.0.1.43	192.168.2.30	FTP	77	Request: USER core
9	2.372675	192.168.2.30	10.0.1.43	TCP	66	ftp > 42423 [ACK] Seq=24 Ack=12 Win=14480 Len=
10	2.372913	192.168.2.30	10.0.1.43	FTP	100	Response: 331 Please specify the password.
11	2.373157	10.0.1.43	192.168.2.30	TCP	66	42423 > ftp [ACK] Seq=12 Ack=58 Win=14608 Len=
12	4.809030	10.0.1.43	192.168.2.30	FTP	77	Request: PASS core
13	4.847451	192.168.2.30	10.0.1.43	TCP	66	ftp > 42423 [ACK] Seq=58 Ack=23 Win=14480 Len=
14	4.915251	192.168.2.30	10.0.1.43	FTP	89	Response: 230 Login successful.
15	4.915705	10.0.1.43	192.168.2.30	TCP	66	42423 > ftp [ACK] Seq=23 Ack=81 Win=14608 Len=
16	4.915858	10.0.1.43	192.168.2.30	FTP	72	Request: SYST
17	4.915893	192.168.2.30	10.0.1.43	TCP	66	ftp > 42423 [ACK] Seq=81 Ack=29 Win=14480 Len=
18	4.916200	192.168.2.30	10.0.1.43	FTP	85	Response: 215 UNIX Type: L8
19	4.955522	10.0.1.43	192.168.2.30	TCP	66	42423 > ftp [ACK] Seq=29 Ack=100 Win=14608 Len
20	8.322284	10.0.1.43	192.168.2.30	FTP	90	Request: PORT 10,0,1,43,222,106
21	8.322900	192.168.2.30	10.0.1.43	FTP	117	Response: 200 PORT command successful. Conside
22	8.323213	10.0.1.43	192.168.2.30	TCP	66	42423 > ftp [ACK] Seq=53 Ack=151 Win=14608 Len
23	8.323529	10.0.1.43	192.168.2.30	FTP	72	Request: LIST
24	8.324665	192.168.2.30	10.0.1.43	TCP	74	ftp-data > 56938 [SYN] Seq=0 Win=14600 Len=0 M
25	8.324850	10.0.1.43	192.168.2.30	TCP	74	56938 > ftp-data [SYN, ACK] Seq=0 Ack=1 Win=14
26	8.325006	192.168.2.30	10.0.1.43	TCP	66	ftp-data > 56938 [ACK] Seq=1 Ack=1 Win=14608 L
27	8.325369	192.168.2.30	10.0.1.43	FTP	105	Response: 150 Here comes the directory listing
28	8.326166	192.168.2.30	10.0.1.43	FTP-DATA	879	FTP Data: 813 bytes
29	8.326304	10.0.1.43	192.168.2.30	TCP	66	56938 > ftp-data [ACK] Seq=1 Ack=814 Win=17376
30	8.326748	192.168.2.30	10.0.1.43	TCP	66	ftp-data > 56938 [FIN, ACK] Seq=814 Ack=1 Win=
31	8.326879	10.0.1.43	192.168.2.30	TCP	66	56938 > ftp-data [FIN, ACK] Seq=1 Ack=815 Win=
32	8.326970	192.168.2.30	10.0.1.43	TCP	66	ftp-data > 56938 [ACK] Seq=815 Ack=2 Win=14608
33	8.327147	192.168.2.30	10.0.1.43	FTP	90	Response: 226 Directory send OK.
34	8.327242	10.0.1.43	192.168.2.30	TCP	66	42423 > ftp [ACK] Seq=59 Ack=214 Win=14608 Len

Figura 30 - Captura de tráfego após execução do comando ftp 10.0.1.42 a partir do nodo 7

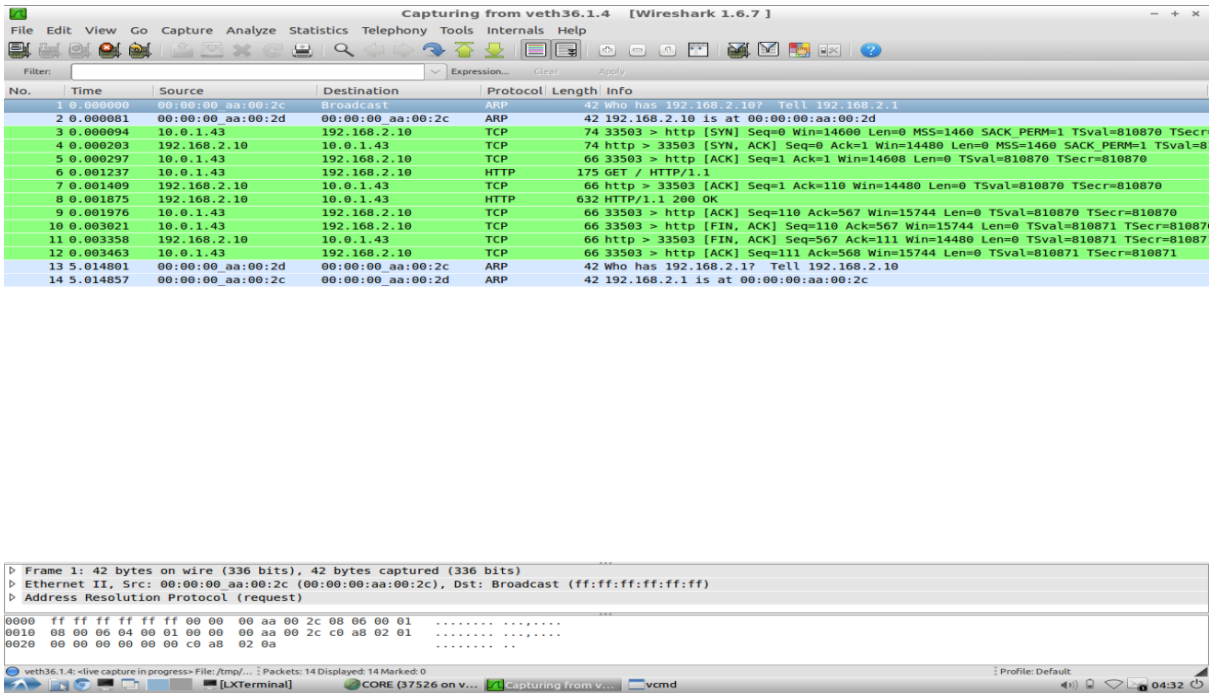


Figura 31 - Captura de tráfego após execução do comando *wget -S 10.0.1.42* a partir do nodo 7

Conclusão

Este trabalho serviu em muito para nos ambientarmos com as ferramentas utilizadas na simulação de redes virtuais e análise de tráfego. Aproximou-nos da realidade aplicando os conceitos retidos nas aulas teóricas.

Trabalhar com novas ferramentas foi um desafio, conseguiu pôr-nos à prova para perceber o porquê de quando uma configuração que tínhamos implementado não estar a funcionar corretamente e tentar resolver o problema.

Em relação às várias configurações de redes foi-nos possível compreender quais os protocolos associados para cada tipo de configuração e como estes funcionam.

Concluindo, o grupo atingiu todos os objetivos propostos pela docente conseguindo pôr a funcionar a topologia que era pedido no enunciado do trabalho prático.

Opinião do grupo para futuros anos: Trabalhos como este são vantajosos para a compreensão da matéria que envolve a UC Redes de Computadores 1, se estes fossem divididos ao longo do semestre acho que seria (Avaliações Periódicas) uma melhor opção e que obrigava também os alunos a investirem um pouco mais de tempo para esta UC.