



**DEPARTAMENTO DE CIÊNCIAS E TECNOLOGIAS
LICENCIATURA EM ENGENHARIA INFORMÁTICA**

Projeto de Gestão de Sistemas e Gestão de Redes

Relatório da Unidade Curricular Gestão de Sistemas e Redes

Autores: Bruno Filipe Santos Saraiva; Diogo Filipe Fernandes Palos; Luís Guilherme De Matos Raposo de Ornelas Candelária; Miguel Aires Caldeira Nunes; Ricardo Alexandre Ângelo Moreira Melo;

Orientador: António Caldeira;

Número dos candidatos: 20160782; 30001058; 30001572; 30000814; 30000486;

Julho de 2020

Lisboa

Índice

1. Introdução	3
2. Parte 1 Gestão de Sistemas	4
2.1 Estruturação da Rede e Seus Componentes	4
2.1.1 Configuração do <i>Server1</i>	5
2.1.2 Configuração do <i>Server2</i>	9
2.1.3 Configuração do <i>LoadBalancer</i>	13
2.2 Problemas Encontrados.....	15
2.3 Troubleshooting	15
3. Parte 2 Gestão de Redes.....	16
3.1 Descrição do Layout	16
3.2 LAN Lisboa/ Porto.....	16
3.3 WAN.....	18
3.4 DNS	20
3.5 TESTES	21
4. Conclusão	23



1. Introdução

No âmbito da unidade curricular de Gestão de Sistemas e Redes, foi-nos solicitada a gestão de um sistema. Este sistema consiste na criação de dois servidores, um balanceador de carga (*LoadBalancer*), que neste caso utiliza o algoritmo *roundrobin* para alternar entre o servidor 1 e o servidor 2, e uma máquina física, que se irá conectar à rede 192.168.2.0/24, que é a mesma rede entre todos os componentes deste sistema, para colocar pedidos à *Web App* através do *LoadBalancer*.

Deparamo-nos com imensos problemas na realização da parte 1 deste projeto, iremos abordar todos os problemas que encontramos e consequentemente, a forma como resolvemos estes problemas.

Quanto à segunda parte do projeto foi-nos pedido outro sistema que foi elaborado no simulador *Cisco Packet Tracer*. Este sistema é constituído pelas LAN de Lisboa e Porto, onde ambas incluem um *router*, um *switch*, um servidor *DHCP/WWW* e quatro computadores. De forma a possibilitar a interligação entre a LAN Lisboa e Porto configurámos uma WAN que efetua o encaminhamento dinâmico através de OSPF e RIP. Finalmente, incluímos um servidor *DNS* com o objetivo de fornecer nomes aos diferentes constituintes do nosso sistema.



2. Parte 1 Gestão de Sistemas

2.1 Estruturação da Rede e Seus Componentes

Para começar a realização da parte 1 do trabalho, começamos por criar três *virtual machines* uma para o *Server1*, uma para o *Server2* e outra para o *LoadBalancer*. O primeiro passo que fizemos, foi a instalação e *setup* de todas as bibliotecas nas *virtual machines*, para evitar mexer na interface de cada uma e também para evitar alterar o modo da conexão da VM para *bridged*, constantemente. Segue uma lista de todas as bibliotecas utilizadas:

Server1:

```
sudo apt-get update
sudo apt-get upgrade -y
sudo apt-get install bind9
sudo apt-get install tomcat7
sudo apt-get install default-jdk
sudo apt-get install isc-dhcp-server
```

Server2:

```
sudo apt-get update
sudo apt-get upgrade -y
sudo apt-get install bind9
sudo apt-get install tomcat7
sudo apt-get install default-jdk
sudo apt-get install nfs-server
```

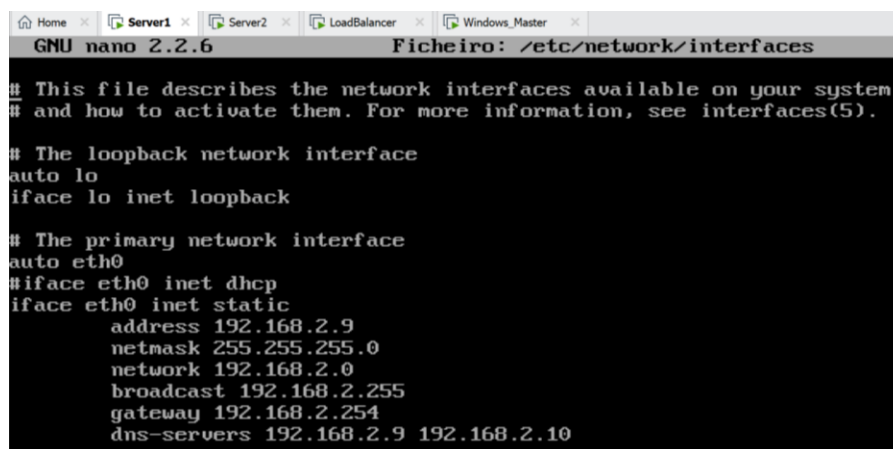
Load Balancer:

```
sudo apt-get update
sudo apt-get upgrade -y
sudo apt-get haproxy
sudo apt-get nfs-client sudo apt-get install default-jdk
```

2.1.1 Configuração do *Server1*

Segue em anexo todas as configurações que tivemos de efetuar no *Server1*.

Configuração da Interface Internet com o comando *sudo nano /etc/network/interfaces*:



```
GNU nano 2.2.6 Ficheiro: /etc/network/interfaces

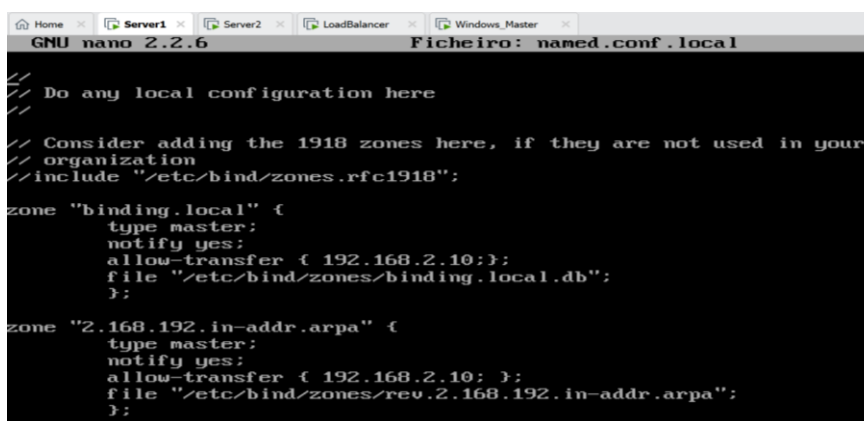
# This file describes the network interfaces available on your system
# and how to activate them. For more information, see interfaces(5).

# The loopback network interface
auto lo
iface lo inet loopback

# The primary network interface
auto eth0
#iface eth0 inet dhcp
iface eth0 inet static
    address 192.168.2.9
    netmask 255.255.255.0
    network 192.168.2.0
    broadcast 192.168.2.255
    gateway 192.168.2.254
    dns-servers 192.168.2.9 192.168.2.10
```

Executar o comando *sudo reboot* para pôr em efeito, as alterações efetuadas. Retiramos o *dhcp* que era o comando *default* para se ligar à rede 192.168.2.0/24, para *static* e definimos como endereço ip o *Server1*: 192.168.2.9. É também necessário definir os dois endereços dos servidores DNS, o endereço do *Server1* e do *Server2*.

De seguida definimos o *Server1* como DNS primário, ou seja, este servidor será o *master*, posteriormente iremos definir o *Server2* como *slave*. Para tal temos de aceder ao diretório *#cd /etc/bind* e editar o ficheiro *named.conf.local*.



```
GNU nano 2.2.6 Ficheiro: named.conf.local

//
// Do any local configuration here
//
// Consider adding the 1918 zones here, if they are not used in your
// organization
//include "/etc/bind/zones.rfc1918";

zone "binding.local" {
    type master;
    notify yes;
    allow-transfer { 192.168.2.10; };
    file "/etc/bind/zones/binding.local.db";
};

zone "2.168.192.in-addr.arpa" {
    type master;
    notify yes;
    allow-transfer { 192.168.2.10; };
    file "/etc/bind/zones/rev.2.168.192.in-addr.arpa";
};
```

Agora que temos configurado o ficheiro responsável por identificar os *PATHS*, vamos criar um diretório *zones* para guardar o ficheiro responsável pelo *bind*.

```
GNU nano 2.2.6 Ficheiro: binding.local.db
binding.local. IN SOA server1.binding.local. postmaster.binding.local. (
                2017030201 ;serial
                3600 ;refresh
                3600 ;retry
                1296000 ;expire
                38400 ) ;negative cache TTL

binding.local. IN NS server1.binding.local.
                IN NS server2.binding.local.
server1 IN A 192.168.2.9
server2 IN A 192.168.2.10
dhcp IN A 192.168.2.12
snmp IN A 192.168.2.13
```

E o ficheiro *rev.2.168.192.in-addr.arpa*.

Agora basta reiniciar o serviço *bind9* responsável pela configuração DNS com o comando *sudo /etc/init.d/bind9 restart*, para reinicar o serviço *bind9* e assim efetuar as alterações.

```
GNU nano 2.2.6 Ficheiro: rev.2.168.192.in-addr.arpa
@ IN SOA server1.binding.local. postmaster.binding.local. (
    2017030201
    3600
    3600
    1296000
    38400 )

    IN NS server1.binding.local.
    IN NS server2.binding.local.

9 IN PTR server1.binding.local.
10 IN PTR server2.binding.local.
12 IN PTR dhcp.binding.local.
13 IN PTR snmp.binding.local.
```

Este servidor, também é o servidor responsável por atribuir endereços IP a qualquer máquina que se ligue a este complexo de rede, pelo que teremos que de configurar este servidor, também, como servidor DHCP.

Primeiramente tivemos de editar o ficheiro *isc-dhcp-server* que se encontra na diretoria */etc/default/*. Onde alterámos a interface para *“eth0”* onde irá servir os leases.

O próximo passo para configurar o DHCP seria entrar na diretoria `/etc/dhcp/` e configurar o ficheiro `dhcpd.conf`, alterando apenas as seguintes definições.

```
# option definitions common to all supported networks...
option domain-name "binding.local";
option domain-name-servers server1, server2;

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

Selecionamos o ficheiro `binding.local` pois é aqui que se encontram as configurações do `bind` do DNS, também informámos quais eram os nossos dois servidores DNS (`Server1` e `Server2`). Definimos também o tempo de empréstimo máximo de um IP para uma máquina, para 7200 segundos.

De seguida só faltava configurar o limite dos servidores que se iram oferecer para `lease` neste caso disponibilizámos IP's desde 192.168.2.64 até 192.168.2.128, mais uma vez, informámos os serviços DHCP, quais são os nossos servidores DNS.

```
subnet 192.168.2.0 netmask 255.255.255.0 {
    range 192.168.2.64 192.168.2.128;
    option subnet-mask 255.255.255.0;
    option domain-name-servers 192.168.2.9, 192.168.2.10;
    option ntp-servers 192.168.2.9;
    option routers 192.168.2.254;
    option broadcast-address 192.168.2.255;
}
```

Posto isto, falta reiniciar o serviço DHCP com o comando `sudo service isc-dhcp-server restart`.

Para verificar os leases já atribuídos basta executar o comando `cat /var/lib/dhcp/dhcpd.leases`.

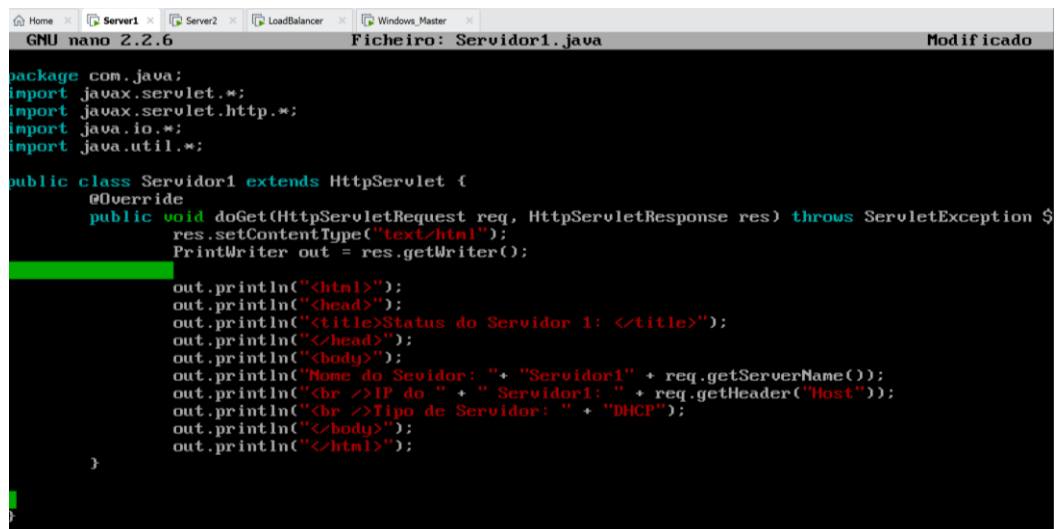
```
lease 192.168.2.65 {
    starts 5 2020/07/03 16:57:07;
    ends 5 2020/07/03 17:07:07;
    cltt 5 2020/07/03 16:57:07;
    binding state active;
    next binding state free;
    rewind binding state free;
    hardware ethernet 00:50:56:c0:00:01;
    uid "\001\000PU\300\000\001";
    client-hostname "DESKTOP-LV495ST";
}
lease 192.168.2.64 {
    starts 5 2020/07/03 16:57:58;
    ends 5 2020/07/03 17:07:58;
    cltt 5 2020/07/03 16:57:58;
    binding state active;
    next binding state free;
    rewind binding state free;
    hardware ethernet 00:0c:29:bc:1a:e4;
    uid "\001\000\014)\274\032\344";
    client-hostname "WIN-IRA2UUOQQGE";
}
amc@ubuntu:~$
```

Feita a configuração do *haproxy* no *LoadBalancer* só falta a criação do *servlet*, a função do *servlet* é disponibilizar a informação do servidor em questão ao utilizador, que se encontra numa máquina física. Este pedido é feito ao *LoadBalancer*, que o transmite para as nossas *backends* o pedido, para o *Server1* e para o *Server2*. Para tal, temos de criar um ficheiro java, compilar esse ficheiro e criar um ficheiro *web.xml*.

Começamos por criar a diretoria *server*, dentro da diretoria */var/lib/tomcat7/webapps/*.

```
amc@ubuntu:~$ cd /var/lib/tomcat7/webapps/
amc@ubuntu:/var/lib/tomcat7/webapps$ ls
ROOT  server
amc@ubuntu:/var/lib/tomcat7/webapps$ _
```

Entramos na diretoria que criamos e voltamos a usar o comando *sudo mkdir WEB-INF* entramos nesta nova diretoria e criamos a diretoria *classes*, criamos dentro desta diretoria, também uma pasta chamada *com* e também ainda dentro desta diretoria a pasta *java* que é onde iremos criar o código java do nosso *servlet*.



```
GNU nano 2.2.6      Ficheiro: Servidor1.java      Modificado
package com.java;
import javax.servlet.*;
import javax.servlet.http.*;
import java.io.*;
import java.util.*;

public class Servidor1 extends HttpServlet {
    @Override
    public void doGet(HttpServletRequest req, HttpServletResponse res) throws ServletException {
        res.setContentType("text/html");
        PrintWriter out = res.getWriter();

        out.println("<html>");
        out.println("<head>");
        out.println("<title>Status do Servidor 1: </title>");
        out.println("</head>");
        out.println("<body>");
        out.println("Nome do Servidor: " + "Servidor1" + req.getServerName());
        out.println("<br />IP do " + " Servidor1: " + req.getHeader("Host"));
        out.println("<br />Tipo de Servidor: " + "DHCP");
        out.println("</body>");
        out.println("</html>");
    }
}
```

Criado o nosso código java basta compilá-lo com o comando *sudo javac -cp /usr/share/tomcat7/lib/servlet-api.jar Servidor1.java*.

```
amc@ubuntu:/var/lib/tomcat7/webapps/server/WEB-INF/classes/com/java$ sudo javac -cp /usr/share/tomcat7/lib/servlet-api.jar Servidor1.java
amc@ubuntu:/var/lib/tomcat7/webapps/server/WEB-INF/classes/com/java$ ls
Servidor1.class  Servidor1.java
amc@ubuntu:/var/lib/tomcat7/webapps/server/WEB-INF/classes/com/java$
```


Tendo o nosso código java compilado só falta criar o ficheiro *web.xml*.

```
GNU nano 2.2.6                               Ficheiro: web.xml
<web-app>
<servlet>
<servlet-name>primeiro</servlet-name>
<servlet-class>com.java.Servidor1</servlet-class>
</servlet>
<servlet-mapping>
<servlet-name>primeiro</servlet-name>
<url-pattern>/status</url-pattern>
</servlet-mapping>
</web-app>
```

Utilizamos o termo *status* para posteriormente colocar no endereço HTML.

Concluimos então a configuração do *Server1*.

2.1.2 Configuração do *Server2*

Segue em anexo todas as configurações que tivemos de efetuar no *Server2*.

Configuração da Interface Internet com o comando *sudo nano /etc/network/interfaces*:

```
GNU nano 2.2.6                               Ficheiro: /etc/network/interfaces
# This file describes the network interfaces available on your system
# and how to activate them. For more information, see interfaces(5).

# The loopback network interface
auto lo
iface lo inet loopback

# The primary network interface
auto eth0
#iface eth0 inet dhcp
iface eth0 inet static
    address 192.168.2.10
    netmask 255.255.255.0
    network 192.168.2.0
    broadcast 192.168.2.255
    gateway 192.168.2.254
    dns-servers 192.168.2.9 192.168.2.10
```

Executar o comando *sudo reboot* para pôr em efeito, as alterações efetuadas. Retiramos o *dhcp* que era o comando *default* para se ligar à rede 192.168.2.0/24, para *static* e definimos como endereço ip o *Server1*: 192.168.2.11. É também necessário definir os dois endereços dos servidores DNS, o endereço do *Server1* e do *Server2*.

De seguida definimos o *Server2* como DNS secundário, ou seja, este servidor será o *slave*, já temos o *Server1* definido como *master*. Para tal temos

de aceder ao diretório `#cd /etc/bind` e editar o ficheiro `named.conf.local` que é igualmente responsável por identificar os *PATHS*. É aqui que também identificamos o endereço do servidor *master*.

```
GNU nano 2.2.6 Ficheiro: named.conf.local

//
// Do any local configuration here
//

// Consider adding the 1918 zones here, if they are not used in your
// organization
//include "/etc/bind/zones.rfc1918";

zone "binding.local" {
    type slave;
    masters { 192.168.2.9; };
    file "/var/lib/bind/binding.local.db";
};

zone "2.168.192.in-addr.arpa" {
    type slave;
    masters { 192.168.2.9; };
    file "/var/lib/bind/rev.2.168.192.in-addr.arpa";
};
```

Agora só temos de fazer o mesmo comando que fizemos, primeiramente no DNS *master*, `sudo /etc/init.d/bind9 restart` para reiniciar o serviço, e assim averiguar se o diretório `/var/lib/bind/` possui os seguintes ficheiros, usando o comando `ls -lisa`.

```
amc@ubuntu:/etc/bind$ cd /var/lib/bind
amc@ubuntu:/var/lib/bind$ ls -lisa
total 20
32672 4 drwxrwxr-x 2 root bind 4096 Jul 2 14:02 .
9762 4 drwxr-xr-x 43 root root 4096 Jul 1 19:13 ..
39891 4 -rw-r--r-- 1 root root 53 Jun 26 23:21 bind9-default.md5sum
40125 4 -rw-r--r-- 1 bind bind 446 Jul 2 22:44 binding.local.db
40099 4 -rw-r--r-- 1 bind bind 514 Jul 3 13:09 rev.2.168.192.in-addr.arpa
amc@ubuntu:/var/lib/bind$
```

Como podemos observar os ficheiros criados no *Server1* encontram-se no *Server2*. Só basta verificar se o *Server2* está a servir como *slave*.

Com o comando `dig @192.168.1.9 -t SOA binding.local +norecurs`.

```
amc@ubuntu:~$ dig @192.168.2.9 -t SOA binding.local +norecurs

; <<> DiG 9.9.5-3ubuntu0.19-Ubuntu <<> @192.168.2.9 -t SOA binding.local +norecurs
; (1 server found)
; global options: +cmd
; Got answer:
; ->HEADER<- opcode: QUERY, status: NOERROR, id: 31720
; flags: qr aa ra; QUERY: 1, ANSWER: 1, AUTHORITY: 2, ADDITIONAL: 3

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags: udp: 4096
;; QUESTION SECTION:
;binding.local.                IN      SOA

;; ANSWER SECTION:
binding.local.                38400   IN      SOA      server1.binding.local. postmaster.binding.local. 2017030201 3600 3600 1296000 38400

;; AUTHORITY SECTION:
binding.local.                38400   IN      NS       server1.binding.local.
binding.local.                38400   IN      NS       server2.binding.local.

;; ADDITIONAL SECTION:
server1.binding.local.        38400   IN      A        192.168.2.9
server2.binding.local.        38400   IN      A        192.168.2.10

;; Query time: 27 msec
;; SERVER: 192.168.2.9#53(192.168.2.9)
;; WHEN: Fri Jul 03 15:40:18 WEST 2020
;; MSG SIZE rcvd: 165
```

Na *ANSWER SECTION* podemos verificar, que de facto, este servidor está a funcionar como servidor de nomes e também temos nas *flags*, a “aa” (*authoritative answers*) o que nos indica que este serviço está a ser executado, logo está a operar como DNS *slave*.

Como sabemos este *Server2*, também é responsável por ser o servidor NFS, para tal temos de o configurar de modo a criar um diretório onde se partilham os ficheiros entre o servidor NFS e os clientes NFS, com o comando *mkdir /mirror*.

Feito este passo, só temos que partilhar esta pasta para estar acessível entre todos os clientes, com o comando */mirror *(rw,no_subtree_check,no_root_squash,sync)*. Para pôr em efeito as alterações, temos de reiniciar o serviço NFS com o comando.

```
amc@ubuntu:~$ sudo service nfs-kernel-server restart
* Stopping NFS kernel daemon                                [ OK ]
* Unexporting directories for NFS kernel daemon...          [ OK ]
* Exporting directories for NFS kernel daemon...             [ OK ]
* Starting NFS kernel daemon                                 [ OK ]
amc@ubuntu:~$ _
```

Agora temos o *Server2* a funcionar como servidor NFS.

Feita a configuração do *haproxy* no *LoadBalancer* só falta a criação do *servlet*, a função do *servlet* é disponibilizar a informação do servidor em questão ao utilizador, que se encontra numa máquina física. Este pedido é feito ao *LoadBalancer*, que o transmite para as nossas *backends* o pedido, para o *Server1* e para o *Server2*. Para tal, temos de criar um ficheiro java, compilar esse ficheiro e criar um ficheiro *web.xml*.

Começamos por criar a diretoria *server*, dentro da diretoria */var/lib/tomcat7/webapps/*.

```
amc@ubuntu:~$ cd /var/lib/tomcat7/webapps/
amc@ubuntu:/var/lib/tomcat7/webapps$ ls
ROOT  server
amc@ubuntu:/var/lib/tomcat7/webapps$ _
```

Entramos na diretoria que criamos e voltamos a usar o comando *sudo mkdir WEB-INF* entramos nesta nova diretoria e criamos a diretoria *classes*, criamos dentro desta diretoria, também uma pasta chamada *com* e também ainda dentro desta diretoria a pasta *java* que é onde iremos criar o código

java do nosso *servlet*.

```
GNU nano 2.2.6 Ficheiro: Servidor2.java Modificado
package com.java;
import javax.servlet.*;
import javax.servlet.http.*;
import java.io.*;
import java.util.*;

public class Servidor2 extends HttpServlet {
    @Override
    public void doGet(HttpServletRequest req, HttpServletResponse res) throws ServletException {
        res.setContentType("text/html");
        PrintWriter out = res.getWriter();

        out.println("<html>");
        out.println("<head>");
        out.println("<title>Status do Servidor 2:</title>");
        out.println("</head>");
        out.println("<body>");
        out.println("Nome do Servidor: " + "Servidor2" + req.getServerName());
        out.println("<br />IP do " + " Servidor2: " + req.getHeader("Host"));
        out.println("<br />Tipo de Servidor: " + "NFS");
        out.println("</body>");
        out.println("</html>");
    }
}
```

Criado o nosso código java basta compilá-lo com o comando *sudo javac -cp /usr/share/tomcat7/lib/servlet-api.jar Servidor2.java*.

```
amc@ubuntu:/var/lib/tomcat7/webapps/server/WEB-INF/classes/com/java$ sudo javac -cp /usr/share/tomcat7/lib/servlet-api.jar Servidor2.java
amc@ubuntu:/var/lib/tomcat7/webapps/server/WEB-INF/classes/com/java$ _
```

Tendo o nosso código java compilado só falta criar o ficheiro *web.xml*.

```
GNU nano 2.2.6 Ficheiro: web.xml
<web-app>
<servlet>
<servlet-name>segundo</servlet-name>
<servlet-class>com.java.Servidor2</servlet-class>
</servlet>
<servlet-mapping>
<servlet-name>segundo</servlet-name>
<url-pattern>/status</url-pattern>
</servlet-mapping>
</web-app>
```

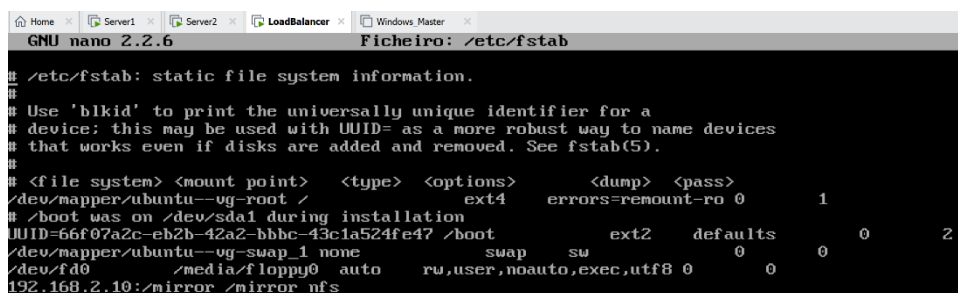
Utilizamos o termo *status* para posteriormente colocar no endereço HTML.

Concluimos então a configuração do *Server2*.

2.1.3 Configuração do *LoadBalancer*

Segue em anexo as configurações que tivemos de efetuar no *LoadBalancer*.

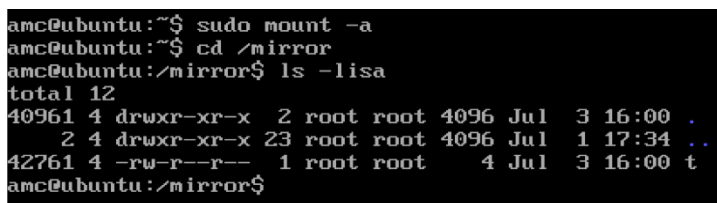
Sabemos que o *LoadBalancer* é cliente NFS do *Server2*, portanto, temos de aceder ao ficheiro *fstab* que se encontra no diretório */etc/*.



```
GNU nano 2.2.6 Ficheiro: /etc/fstab
# /etc/fstab: static file system information.
#
# Use 'blkid' to print the universally unique identifier for a
# device; this may be used with UUID= as a more robust way to name devices
# that works even if disks are added and removed. See fstab(5).
#
# <file system> <mount point> <type> <options> <dump> <pass>
/dev/mapper/ubuntuvg-root / ext4 errors=remount-ro 0 1
# /boot was on /dev/sda1 during installation
UUID=66f07a2c-eb2b-42a2-bbbc-43c1a524fe47 /boot ext2 defaults 0 2
/dev/mapper/ubuntuvg-swap_1 none swap sw 0 0
/dev/fd0 /media/floppy0 auto rw,user,noauto,exec,utf8 0 0
192.168.2.10:/mirror /mirror nfs
```

Só temos de adicionar a linha do endereço IP do cliente que se pretende ligar à pasta do servidor: *192.168.2.10:/mirror /mirror nfs*.

Posto isto, só temos de compilar a alteração nos clientes, com o comando *sudo mount -a*.



```
amc@ubuntu:~$ sudo mount -a
amc@ubuntu:~$ cd /mirror
amc@ubuntu:/mirror$ ls -lisa
total 12
4096 4 drwxr-xr-x 2 root root 4096 Jul 3 16:00 .
      2 4 drwxr-xr-x 23 root root 4096 Jul 1 17:34 ..
42761 4 -rw-r--r-- 1 root root 4 Jul 3 16:00 t
amc@ubuntu:/mirror$
```

Criámos um ficheiro chamado *t* no servidor NFS (*Server2*) e executámos o comando *ls -lisa* dentro do diretório */mirror* no cliente (*LoadBalancer*).

O passo final na configuração deste servidor será configurar a distribuição de carga dos servidores 1 e 2, para este efeito, vamos configurar o script que se encontra em *sudo nano /etc/haproxy/haproxy.cfg*.


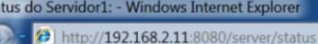

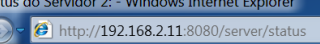








```
GNU nano 2.2.6                               Ficheiro: /etc/haproxy/haproxy.cfg

global
    log /dev/log      local0
    log 127.0.0.1     local1 notice
    maxconn 4000
    chroot /var/lib/haproxy
    user haproxy
    group haproxy
    daemon

defaults
    log          global
    mode         http
    option        httplog
    option        dontlognull
    retries      3
    option        redispatch
    maxconn      2000
    timeout      5000
    clitimeout   50000
    srvtimerout  50000
    errorfile    400 /etc/haproxy/errors/400.http
    errorfile    403 /etc/haproxy/errors/403.http
    errorfile    408 /etc/haproxy/errors/408.http
    errorfile    500 /etc/haproxy/errors/500.http
    errorfile    502 /etc/haproxy/errors/502.http
    errorfile    503 /etc/haproxy/errors/503.http
    errorfile    504 /etc/haproxy/errors/504.http

listen webfarm 0.0.0.0:8080
    mode http
    stats enable
    stats uri /haproxy?stats
    balance roundrobin
    option httpclose
    option forwardfor
    server server1 192.168.2.9:8080 check
    server server2 192.168.2.10:8080 check
```

Informamos então quais os *servers* que o *LoadBalancer* tem de gerir com o algoritmo *roundrobin* que irá alternar a carga entre os *servers*, tanto que quando inserimos o endereço *192.168.2.11:8080/server/status*, numa máquina virtual e pressionamos F5, vamos receber o estado de cada servidor, pois o *roundrobin* irá alterar a carga entre os servidores.

Status do Servidor1: - Windows Internet Explorer	Status do Servidor 2: - Windows Internet Explorer
 	 
  	  
	
Nome do Servidor: Servidor1192.168.2.11 IP do Servidor1: 192.168.2.11:8080 Tipo de Servidor: DHCP	Nome do Servidor: Servidor2192.168.2.11 IP do Servidor2: 192.168.2.11:8080 Tipo de Servidor: NFS



2.2 Problemas Encontrados

Aqui vamos descrever todos os problemas que encontrámos na realização da parte 1 deste projeto.

1. Erro no *hyper v guard*;
2. Era necessária outra VM de auxílio, puramente uma VM DNS;
3. Incoerência na nomenclatura nos nomes dos ficheiros;
4. Problema ao utilizar o serviço de criação de pastas *mkdir*;
5. Tomcat7 não funcionava.
6. Barra a mais no código NFS do cliente o que impedia de encontrar a localização do *rev*;
7. Faltava configurar a network interface do *LoadBalancer*;
8. Erro no código *sudo service isc-dhcp-server restart*;

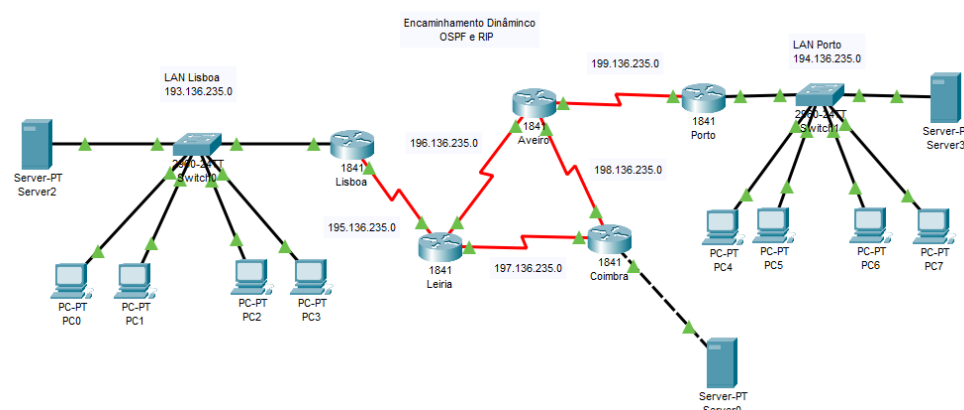
2.3 Troubleshooting

1. Este erro apareceu devido a um *update* do *Windows*, a solução foi desativar este serviço e reinstalar o *VMWare Workstation*, também se teve de alterar as políticas de grupo.
2. Não era necessária nenhuma VM de auxílio, conseguimos configurar tudo corretamente entre o *Server1*, *Server2* e o *LoadBalancer*.
3. Existiam alguns erros que nos tinham escapado, que já foram corrigidos, por exemplo *bindinf*, para *binding*.
4. Tivemos de alterar o PATH deste serviço para o seu correto funcionamento o comando `"export PATH="/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin:/root/bin"`.
5. Tivemos de encontrar onde o tomcat7 se encontrava com o comando, *sudo find / -name "version.sh"*, este encontrava-se no diretório *localgames* e tivemos de efetuar a correção, para o diretório correto.
6. Corrigimos então uma barra em falta no ficheiro *named.conf.local* em file *"/rev"*.
7. Configurámos esta rede e atribuímos o IP 192.168.2.11.
8. Tivemos de reiniciar o *Server1*.

3. Parte 2 Gestão de Redes

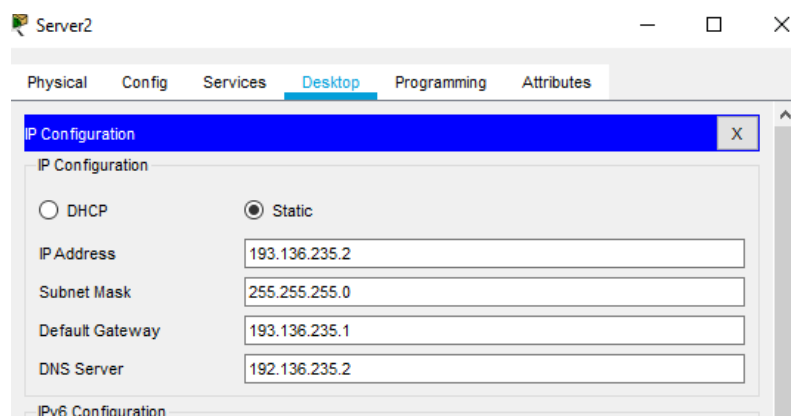
3.1 Descrição do Layout

O desafio imposto na segunda parte do projeto é a elaboração e interligação entre LAN diferentes através de uma WAN que efetua o encaminhamento de forma dinâmica. Neste sistema ainda tivemos que incluir um servidor DNS para a atribuição de nomes aos diferentes constituintes do sistema. Na imagem que se segue podemos observar o layout escolhido para a realização desta simulação.

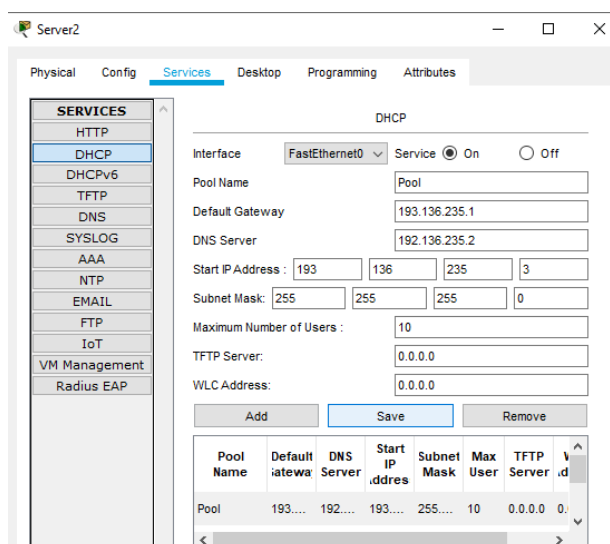


3.2 LAN Lisboa/ Porto

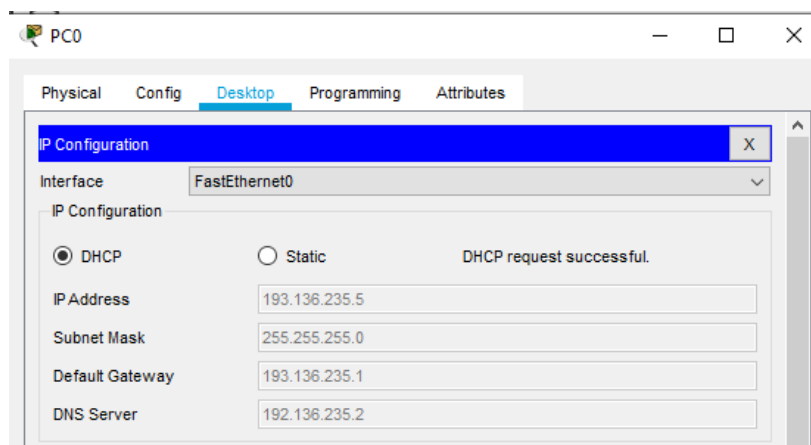
A LAN de Lisboa e Porto incluem um *router*, um *switch*, um servidor *DHCP/WWW* e quatro computadores. O objetivo do servidor é fornecer IP's aos computadores que estão contidos na mesma rede. Em primeiro lugar tivemos que atribuir um IP estático a cada um dos servidores.



A imagem acima concerne à atribuição do IP estático do server localizado em Lisboa. Neste servidor atribuímos-lhe o IP 193.136.235.2, enquanto que no servidor situado no Porto atribuímos-lhe o IP 194.136.235.2. O campo da *Default Gateway* diz respeito à porta de entrada e de saída da rede, por isto, atribuímos o IP 193.136.235.1 no caso da rede de Lisboa, e o IP 194.136.235.1 no caso da rede do Porto. Por último, tivemos que conceder o IP do servidor DNS que está localizado em Coimbra.

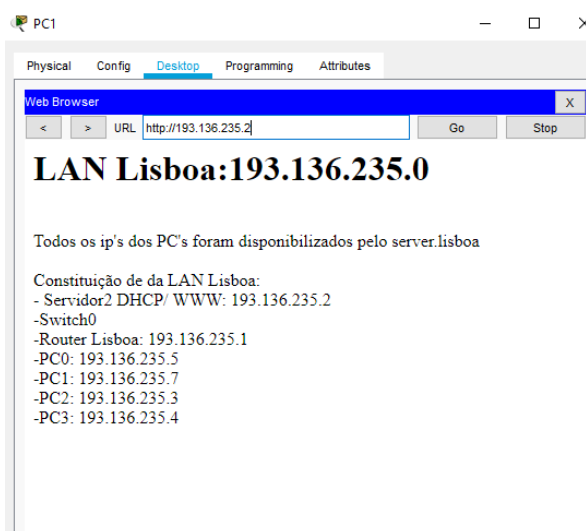


Quanto à configuração do serviço *DHCP* tivemos que, em primeiro lugar ligar a interface fa0. De seguida tivemos que dar um nome à pool e definir o *Start IP Address* para configurar o *range* de endereços. Como a *LAN* de lisboa e do Porto é constituída apenas por quatro *PC's* definimos que o número máximo de distribuições de *IP's* seria dez, no caso de querermos adicionar mais alguns computadores.



Nesta imagem podemos comprovar que o serviço DHCP está a fornecer IP's corretamente. Este computador situa-se na LAN de Lisboa e foi-lhe atribuído o *IP* 193.136.235.5.

Os servidores 2 e 3 também têm a função de disponibilizar páginas estáticas com a informação técnica sobre cada uma das *LAN*. Para que isto fosse possível tivemos que ativar os protocolos *HTTP* e *HTTPS* e editar a página "*index.html*". Posto isto, basta entrar na Web Browser de qualquer computador da *LAN* e digitar o *IP* do servidor no *URL*. Na imagem que se segue podemos observar a página estática disponibilizada pelo servidor de Lisboa.



3.3 WAN

A *WAN* tem o objetivo de interligar a *LAN* de Lisboa e Porto. Esta é constituída por vários routers localizados em diferentes pontos do país, tal como por exemplo em Leiria, Coimbra e Aveiro, que possuem, obviamente, redes diferentes. Na ligação entre os routers utilizamos os cabos *Serial DTE*. Configurámos todas as interfaces seriais da rede. A imagem que se segue mostra as interfaces do router Leiria (através do comando `#show run`).



```
shut down
!
interface FastEthernet0/1
no ip address
duplex auto
speed auto
shut down
!
interface Serial0/0/0
ip address 195.136.235.2 255.255.255.0
clock rate 2000000
!
interface Serial0/0/1
ip address 197.136.235.1 255.255.255.0
!
interface Serial0/1/0
no ip address
clock rate 2000000
shut down
!
interface Serial0/1/1
ip address 196.136.235.1 255.255.255.0
!
```

Foi-nos pedido que a WAN efetuasse o encaminhamento das tramas através dos protocolos OSPF e RIP. Antes demais, tivemos que compreender qual o papel que cada protocolo tinha: o protocolo *RIP* (*Routing Information Protocol*) utiliza o algoritmo vetor-distância, enquanto que o protocolo *OSPF* (*Open Shortest Path First*) utiliza o algoritmo *Link State Routing Protocol*. O algoritmo vetor-distância parte do princípio de que cada router contém uma tabela de informação (ARL) que informa todas as possíveis rotas e, a partir desta tabela o algoritmo escolhe a melhor rota que deve ser utilizada. Por outro lado, o algoritmo *Link State Routing Protocol* procura o menor trajeto possível através dos nós que possuem todos os links da rede que contêm o identificador de interface e a distância. Com esta informação os routers descobrem, por si, a melhor rota possível.

Para incorporar o protocolo *RIP* tivemos que configurar todos os routers que pertencem à WAN com o seguinte código:

```
RouterExemplo# config terminal
RouterExemplo(config)# router rip
RouterExemplo(config-router)#network 193.136.235.0 (exemplo de uma
network)
RouterExemplo(config-router)#CTRL Z
```

Para incorporar o protocolo *OSPF* tivemos que configurar todos os routers que pertencem à *WAN* com o seguinte código:

```
RouterExemplo# config terminal
```

```
RouterExemplo(config)#router ospf 1
```

```
RouterExemplo(config-router)#network 197.136.235.0 0.0.0.0.255 area 0
```

(exemplo de uma *network*)

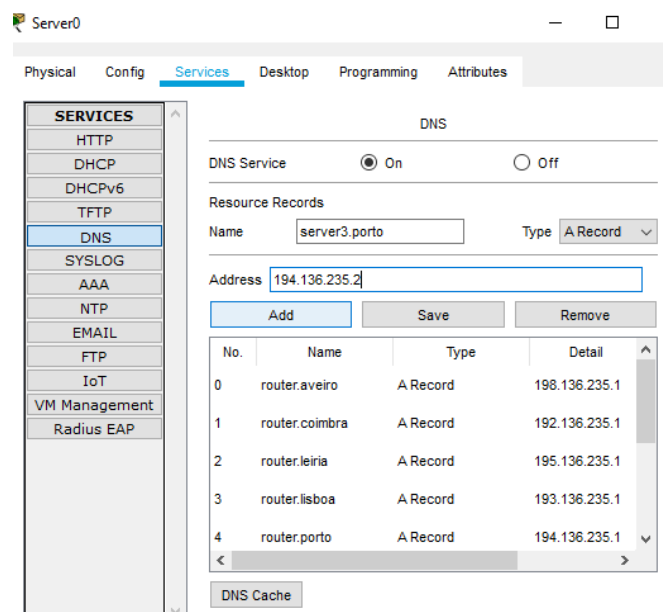
```
RouterExemplo(config-router)#CTRL Z
```

3.4 DNS

No nosso sistema tivemos de incluir um servidor DNS (Domain Name System) que está localizado em Coimbra. Este servidor tem a função de atribuir nomes para todos os routers e servidores que existem no sistema.

Atribuímos ao servidor DNS o IP estático de 192.136.235.2, uma vez que este está na rede de Coimbra. A ligação entre o router Coimbra e o servidor é feita através do cabo *Copper Cross-Over* uma vez que ambos estão situados na mesma localidade.

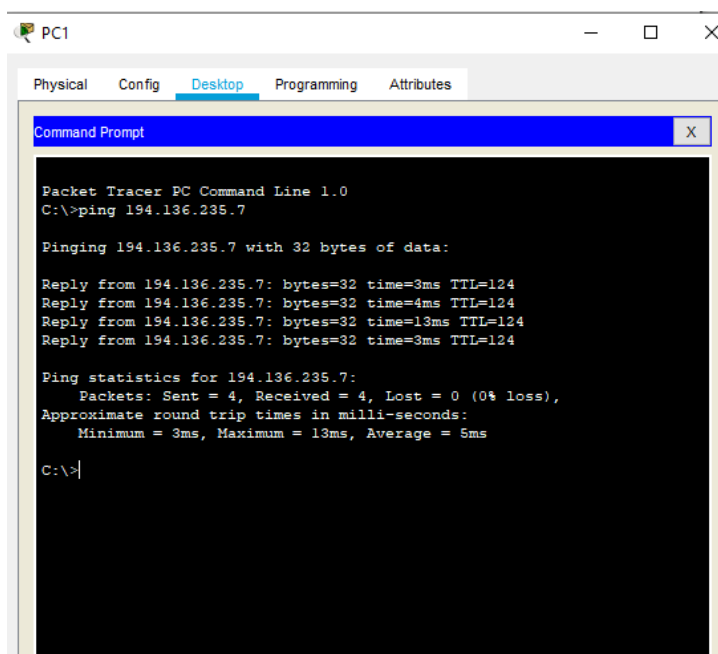
Na imagem que se segue damos o exemplo de atribuir o nome *server3.porto* ao servidor que se encontra no Porto. Tal como podemos observar, temos de atribuir um nome ao IP que queremos traduzir. No final, basta clicar no botão “Add”.



3.5 TESTES

Para testar a interligação da LAN Lisboa e Porto podemos proceder à verificação do *Ping* (*Packet Internet Network Grouper*) entre dois computadores. Se esta verificação se confirmar podemos concluir que estes dois equipamentos estão a enviar dados um ao outro, ou seja, existe conectividade entre ambos.

Na imagem que se segue fizemos Ping do PC1 situado em Lisboa para o PC7 localizado no Porto.



```
PC1
Physical Config Desktop Programming Attributes
Command Prompt
Packet Tracer PC Command Line 1.0
C:\>ping 194.136.235.7

Pinging 194.136.235.7 with 32 bytes of data:

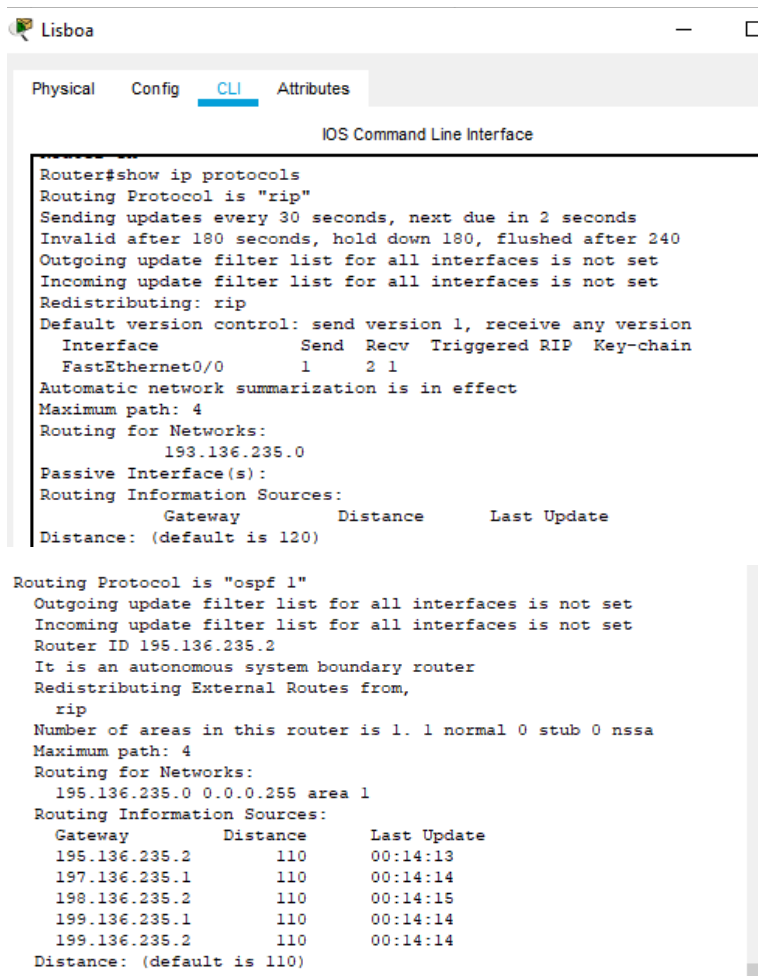
Reply from 194.136.235.7: bytes=32 time=3ms TTL=124
Reply from 194.136.235.7: bytes=32 time=4ms TTL=124
Reply from 194.136.235.7: bytes=32 time=13ms TTL=124
Reply from 194.136.235.7: bytes=32 time=3ms TTL=124

Ping statistics for 194.136.235.7:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 3ms, Maximum = 13ms, Average = 5ms

C:\>
```

Tal como podemos observar os dois computadores estão a conseguir comunicar e, por isto podemos confirmar que a interligação entre a LAN Lisboa e Porto foi bem sucedida.

Também quisemos averiguar se os protocolos RIP e OSPF estão ativos na WAN, para isso, tivemos que utilizar o comando *#show ip protocols*. As imagens que se seguem correspondem à aplicação deste comando no router Lisboa.



```
Router#show ip protocols
Routing Protocol is "rip"
  Sending updates every 30 seconds, next due in 2 seconds
  Invalid after 180 seconds, hold down 180, flushed after 240
  Outgoing update filter list for all interfaces is not set
  Incoming update filter list for all interfaces is not set
  Redistributing: rip
  Default version control: send version 1, receive any version
    Interface          Send Recv Triggered RIP Key-chain
  FastEthernet0/0      1     2 1
  Automatic network summarization is in effect
  Maximum path: 4
  Routing for Networks:
    193.136.235.0
  Passive Interface(s):
  Routing Information Sources:
    Gateway            Distance      Last Update
  Distance: (default is 120)

Routing Protocol is "ospf 1"
  Outgoing update filter list for all interfaces is not set
  Incoming update filter list for all interfaces is not set
  Router ID 195.136.235.2
  It is an autonomous system boundary router
  Redistributing External Routes from,
    rip
  Number of areas in this router is 1. 1 normal 0 stub 0 nssa
  Maximum path: 4
  Routing for Networks:
    195.136.235.0 0.0.0.255 area 1
  Routing Information Sources:
    Gateway            Distance      Last Update
  195.136.235.2         110          00:14:13
  197.136.235.1         110          00:14:14
  198.136.235.2         110          00:14:15
  199.136.235.1         110          00:14:14
  199.136.235.2         110          00:14:14
  Distance: (default is 110)
```

Assim, garantimos que ambos os protocolos estão ativos no Router Lisboa. Os restantes equipamentos da WAN foram também configurados deste modo e, portanto, estão também ativos.



4. Conclusão

A partir da elaboração deste trabalho abordámos todos os conceitos lecionados na unidade curricular de gestão de sistemas e redes.

Cumprimos todos os objetivos propostos, uma vez que, quer na parte de gestão de sistemas, quer no que concerne à gestão de redes, testámos todas as configurações que realizámos e estas foram todas bem sucedidas.

Deste modo, consideramos a elaboração do presente trabalho bastante importante não só para o nosso conhecimento, mas ainda como aprofundamento do tema. De facto, esta tarefa para além de nos ter permitido desenvolver competências de investigação, também nos permitiu aperfeiçoar outras áreas de interesse igualmente associadas aos conteúdos trabalhados.