Relatório do 2º Trabalho Laboratorial

Universidade do Porto

Faculdade de Engenharia

FEUP

Mestrado Integrado em Engenharia Informática e Computação

Redes de Computadores

Membros do Grupo:

Daniel Teixeira - 100509067

Luis Guilherme Martins – 100509105

Victor Cerqueira - 100509055

19 de Dezembro de 2012

1. Resumo

Este trabalho laboratorial tem como objectivos principais desenvolver uma aplicação de *download* que utilize o protocolo FTP para servir o seu propósito e a configuração de uma rede de computadores onde se abordaram vários conceitos tais como VLAN, IP Forwarding, MAC, DNS, entre outros.

2. Introdução

Neste relatório vamos abordar alguns aspectos da aplicação de *download,* do seu desenvolvimento, estrutura e funcionamento e posteriormente, vamos analisar a configuração implementada na rede de computadores, dividindo-a por partes que correspondem às experiências que estão referidas no enunciado. Ainda relativamente às experiências, foram obtidos *logs* para cada uma, para se proceder à análise da informação obtida.

3. Aplicação de Download

A aplicação de *download* requerida para este trabalho laboratorial foi desenvolvida em C e o objectivo principal foi que fosse bastante simples, prática e legível. Usou-se o protocolo FTP e a API de *sockets* do sistema operativo Linux para comunicar com o servidor e proceder ao download do ficheiro indicado no URL.

Estrutura

Como foi referido anteriormente, um dos objectivos definido para o cliente de download era que fosse simples. Por isso, toda a aplicação foi desenvolvida num só ficheiro (que seguirá em anexo com o nome *ftp.c*) e com funções que sejam o mais auto-explicativas possível.

A lista de funções é esta:

- separateURL

Esta função tem como objectivo separar para arrays a o *username*, a *password*, o *link* do servidor onde está alojado o ficheiro, a localização e o nome do ficheiro.

- parseDatafd

Esta função separa o endereço recebido pelo cliente quando entra em modo passivo para obter o IP e a porta para descarregar o ficheiro.

- getHost

Esta função devolve a *struct* correspondente ao *host* do servidor quando recebe como *input* o seu URL. Converte, por exemplo, o URL do servidor no seu IP.

- login

Esta função recorre à API de *sockets* do Linux e comunica com o servidor através destes. É responsável pelo envio de informação de *login* para além de verificar os códigos de resposta do servidor, que podem ser se sucesso (código 2xx) ou de falha (códigos 4xx ou 5xx).

- enterPassiveMode

Nesta função o cliente entra em modo passivo e usa a função *parseDatafd* para obter o IP e a porta para se ligar.

- getFileSize

Esta função simplesmente devolve o tamanho do ficheiro a descarregar para efeitos de estatística aquando a descarga.

- download

Esta função procura o ficheiro no servidor e efectua a descarga do mesmo.

Execução

De forma a tornar o programa o mais prático possível, fizemos com que ele apenas devolve a informação mais importante. Desta maneira, códigos de resposta com mensagens de erro não são exibidos, sendo substituídos por mensagens mais esclarecedoras para o utilizador.

Fica aqui um exemplo de uma execução decorrida com sucesso:

```
linus60:~/Desktop> ./ftp ftp://anonymous:pass@ftp.up.pt/pub/robots.txt
Nome: ftp.up.pt
IP: 193.136.37.8
Successfuly logged!
227 Entering Passive Mode (193,136,37,8,212,255)

Entered Passive Mode!
150 Opening BINARY mode data connection for pub/robots.txt (23 bytes).

Starting to download file...
Writing 23 bytes of 23...(100.000000 %)
Writing 23 bytes of 23...(100.000000 %)
Download completed!
```

4. Configuração da rede e respectiva análise

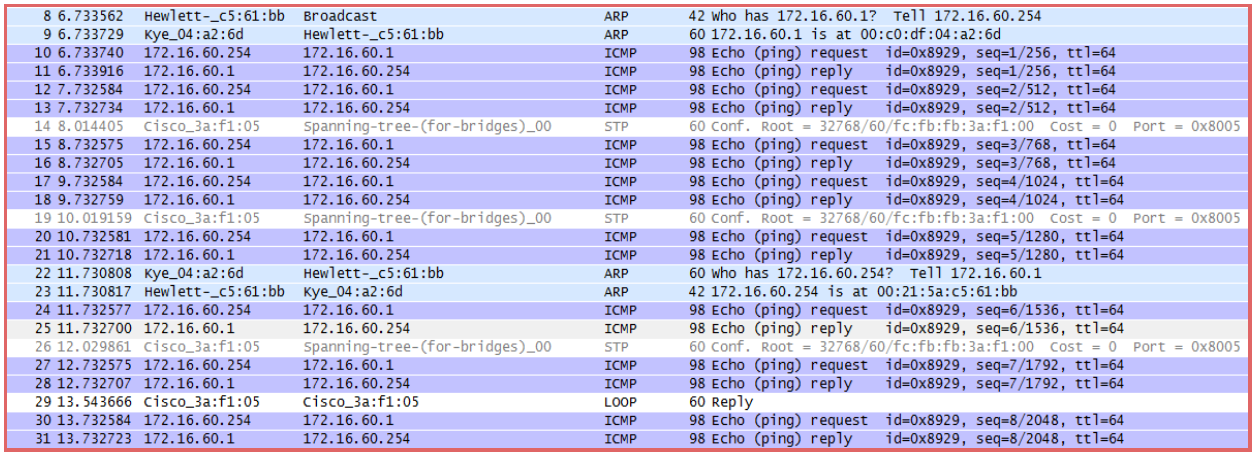
○ Experiência 1:

A primeira experiência consiste na configuração de uma rede IP, onde se pretende observar o funcionamento do ARP e das tabelas de encaminhamento. Os principais comandos utilizados foram os seguintes:

- *ifconfig*
- *route*
- *arp*

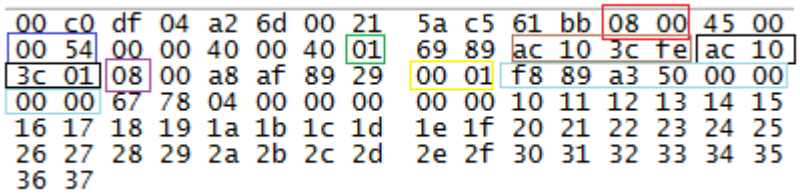
Para verificar o bom funcionamentos da rede configurada nesta experiência fez-se

ping ao tux61 a partir do tux64 registando-se os seguintes pacotes de dados no primeiro:



8	6.733562	Hewlett-_c5:61:bb	Broadcast	ARP	42 Who has 172.16.60.1? Tell 172.16.60.254
9	6.733729	Kye_04:a2:6d	Hewlett-_c5:61:bb	ARP	60 172.16.60.1 is at 00:c0:df:04:a2:6d
10	6.733740	172.16.60.254	172.16.60.1	ICMP	98 Echo (ping) request id=0x8929, seq=1/256, ttl=64
11	6.733916	172.16.60.1	172.16.60.254	ICMP	98 Echo (ping) reply id=0x8929, seq=1/256, ttl=64
12	7.732584	172.16.60.254	172.16.60.1	ICMP	98 Echo (ping) request id=0x8929, seq=2/512, ttl=64
13	7.732734	172.16.60.1	172.16.60.254	ICMP	98 Echo (ping) reply id=0x8929, seq=2/512, ttl=64
14	8.014405	Cisco_3a:f1:05	Spanning-tree-(for-bridges)_00	STP	60 Conf. Root = 32768/60/fc:fb:fb:3a:f1:00 Cost = 0 Port = 0x8005
15	8.732575	172.16.60.254	172.16.60.1	ICMP	98 Echo (ping) request id=0x8929, seq=3/768, ttl=64
16	8.732705	172.16.60.1	172.16.60.254	ICMP	98 Echo (ping) reply id=0x8929, seq=3/768, ttl=64
17	9.732584	172.16.60.254	172.16.60.1	ICMP	98 Echo (ping) request id=0x8929, seq=4/1024, ttl=64
18	9.732759	172.16.60.1	172.16.60.254	ICMP	98 Echo (ping) reply id=0x8929, seq=4/1024, ttl=64
19	10.019159	Cisco_3a:f1:05	Spanning-tree-(for-bridges)_00	STP	60 Conf. Root = 32768/60/fc:fb:fb:3a:f1:00 Cost = 0 Port = 0x8005
20	10.732581	172.16.60.254	172.16.60.1	ICMP	98 Echo (ping) request id=0x8929, seq=5/1280, ttl=64
21	10.732718	172.16.60.1	172.16.60.254	ICMP	98 Echo (ping) reply id=0x8929, seq=5/1280, ttl=64
22	11.730808	Kye_04:a2:6d	Hewlett-_c5:61:bb	ARP	60 Who has 172.16.60.254? Tell 172.16.60.1
23	11.730817	Hewlett-_c5:61:bb	Kye_04:a2:6d	ARP	42 172.16.60.254 is at 00:21:5a:c5:61:bb
24	11.732577	172.16.60.254	172.16.60.1	ICMP	98 Echo (ping) request id=0x8929, seq=6/1536, ttl=64
25	11.732700	172.16.60.1	172.16.60.254	ICMP	98 Echo (ping) reply id=0x8929, seq=6/1536, ttl=64
26	12.029861	Cisco_3a:f1:05	Spanning-tree-(for-bridges)_00	STP	60 Conf. Root = 32768/60/fc:fb:fb:3a:f1:00 Cost = 0 Port = 0x8005
27	12.732575	172.16.60.254	172.16.60.1	ICMP	98 Echo (ping) request id=0x8929, seq=7/1792, ttl=64
28	12.732707	172.16.60.1	172.16.60.254	ICMP	98 Echo (ping) reply id=0x8929, seq=7/1792, ttl=64
29	13.543666	Cisco_3a:f1:05	Cisco_3a:f1:05	LOOP	60 Reply
30	13.732584	172.16.60.254	172.16.60.1	ICMP	98 Echo (ping) request id=0x8929, seq=8/2048, ttl=64
31	13.732723	172.16.60.1	172.16.60.254	ICMP	98 Echo (ping) reply id=0x8929, seq=8/2048, ttl=64

Examinando o que contêm os pacotes enviados pelo tux64 podemos ver:



```
00 c0 df 04 a2 6d 00 21  5a c5 61 bb 08 00 45 00
00 54 00 00 40 00 40 01  69 89 ac 10 3c fe ac 10
3c 01 08 00 a8 af 89 29  00 01 f8 89 a3 50 00 00
00 00 67 78 04 00 00 00  00 00 10 11 12 13 14 15
16 17 18 19 1a 1b 1c 1d  1e 1f 20 21 22 23 24 25
26 27 28 29 2a 2b 2c 2d  2e 2f 30 31 32 33 34 35
36 37
```

Flag que identifica um pacote IP (0x0800)
Tamanho do pacote (84 bits)
Flag que identifica o protocolo IMCP (0x01)
Emissor (172.16.60.254)
Destino (172.16.60.1)
Flag que identifica um *ping request*
Número do pacote enviado (neste caso foi o primeiro pedido de ping)
Informação da data e hora em que foi enviado

Por sua vez, as respostas do tux61 são em tudo semelhantes salvo numa flag, que podemos ver de seguida:

```
00 21 5a c5 61 bb 00 c0  df 04 a2 6d 08 00 45 00
00 54 91 e4 00 00 40 01  17 a5 ac 10 3c 01 ac 10
3c fe 00 00 b0 af 89 29  00 01 f8 89 a3 50 00 00
00 00 67 78 04 00 00 00  00 00 10 11 12 13 14 15
16 17 18 19 1a 1b 1c 1d  1e 1f 20 21 22 23 24 25
26 27 28 29 2a 2b 2c 2d  2e 2f 30 31 32 33 34 35
36 37
```

Flag que identifica uma *ping response*

Note-se que em ambos os casos o endereço MAC da origem e do destino estão definidos, sendo isto apenas possível devido ao protocolo ARP. Como se pode ver no log, o tux64 recorreu a este para identificar qual o endereço MAC do tux61 antes de poder realizar ping.

Address Resolution Protocol ou ARP refere-se a um protocolo de comunicação cujo objectivo é permitir encontrar um endereço da *link layer* a partir de um endereço da *network layer*. No caso deste projecto é utilizado para obter o endereço MAC a partir do endereço IP. O protocolo ARP funciona da seguinte forma:

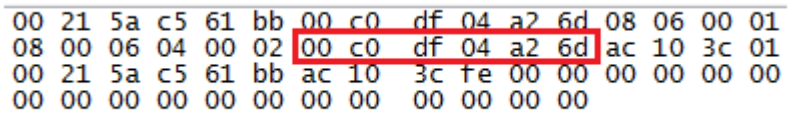
- O emissor envia para toda a rede um pacote ARP com o seu próprio endereço MAC e o endereço IP para o qual pretende descobrir a correspondência.
- Se algum receptor na rede identificar o IP enviado como sendo o seu responde enviando o seu endereço MAC.

Assim, se no tux64 pretendessemos descobrir qual o endereço MAC do tux61 enviaríamos o seguinte pacote de dados:

```
ff ff ff ff ff ff 00 21  5a c5 61 bb 08 06 00 01
08 00 06 04 00 01 00 21  5a c5 61 bb ac 10 3c fe
00 00 00 00 00 00 ac 10  3c 01
```

Flag que identifica um pacote ARP (0x0806)
Endereço MAC do emissor (00:21:5a:c5:61:bb)
Endereço IP do emissor (172.16.60.254)
Endereço IP que se pretende converter para MAC (172.16.60.1)

E, por sua vez o tux61 responderia:

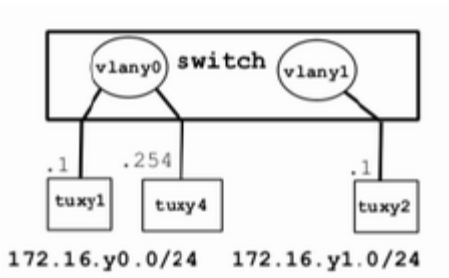


Endereço MAC do receptor procurado (00:c0:df:04:a2:6d)

Note-se que o primeiro pacote é enviado do tux64 em modo *broadcast*, ou seja, usando o endereço MAC ff:ff:ff:ff:ff:ff uma vez que este pretende descobrir qual o endereço MAC que corresponde a um determinado IP, logo não pode direccionar o pacote, tendo que o enviar para toda a rede.
Por sua vez a resposta do tux61 é direccionada apenas a quem solicitou a informação do seu endereço MAC (neste caso o tux64, com endereço MAC 00:21:5a:c5:61:bb).
Como nota final apenas é relevante referir a importância da interface de *loopback* (representados no log como usando o protocolo LOOP). Estes pacotes são como uma espécie de ping enviados para o próprio emissor e permitem verificar a operacionalidade dos protocolos, garantindo assim que não há problema nenhum com o emissor.

- **Experiência 2:**

Esta experiência gira à volta dum conceito já referido anteriormente, as VLANs (Redes Locais Virtuais) e para a realizarmos, foi preciso um dispositivo físico, um switch da Cisco.
As VLANs configuradas seguiram o seguinte esquema que está representado a seguir:



Foi no switch que tivemos que configurar as VLANs, através duma consola própria

para o efeito, utilizando os comandos que se seguem e que estão disponíveis no enunciado do trabalho prático.

```
configure terminal
vlan 60
end
show vlan id 60

configure terminal
vlan 61
end
show vlan id 61
```

Uma vez criadas as VLANs, era preciso adicionar as portas e estava concluída a experiência.
Para testar fizeram-se alguns *pings* para verificar se a configuração estava bem feita.

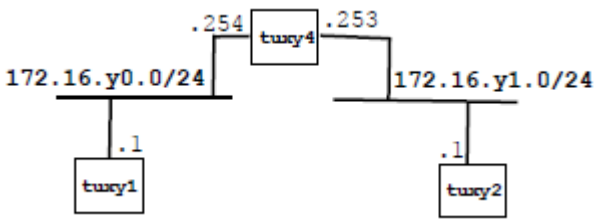
```
19 16.615162  172.16.60.1     172.16.60.254    ICMP     98 Echo (ping) request  id=0x0663, seq=4/1024, ttl=64
20 16.615168  172.16.60.254   172.16.60.1      ICMP     98 Echo (ping) reply    id=0x0663, seq=4/1024, ttl=64

30 23.277743  172.16.61.1     172.16.61.253    ICMP     98 Echo (ping) request  id=0x941c, seq=2/512, ttl=64
31 23.277758  172.16.61.253   172.16.61.1      ICMP     98 Echo (ping) reply    id=0x941c, seq=2/512, ttl=64
```

Nestas imagens vê-se que os *pings* feitos nas VLANs criadas seguem a configuração representada no esquema anterior uma vez que só respondem aos *requests* os computadores que estão ligados na sua VLAN. Desta forma pode-se concluir que existem dois domínios de *broadcast*, que corresponde cada um à sua VLAN, isolando-as uma da outra.

- **Experiência 3:**

Na terceira experiência, o objectivo seria transformar o tux4 num router, usando as vlans previamente criadas e configurando outra ligação eth1, permitindo assim transmitir informação de um tux para uma qualquer porta representada na imagem.



```
configure terminal
interface fastethernet 0/1
switchport mode acess
switchport access vlan 60
end
```

```
configure terminal
interface fastethernet 0/2
switchport mode acess
switchport access vlan 60
end
```

```
configure terminal
interface fastethernet 0/3
switchport mode acess
switchport access vlan 61
end
```

```
configure terminal
interface fastethernet 0/4
switchport mode acess
switchport access vlan 61
end
```

Com os comandos aqui apresentados, é possivel configurar as ligações às 4 portas representadas na figura.

```
echo 1 > /proc/sys/net/ipv4/ip_forward
echo 0 > /proc/sys/net/ipv4/icmp_echo_
_ignore_broadcasts
```

Depois de permitir o IP forwarding e desactivar o ICMP echo_ignore_broadcasts, com os comandos anteriores.
A seguir estão representados os logs do eth0:

```
17 14.630731  172.16.20.1   172.16.21.1   ICMP   98 Echo (ping) request  id=0xeb18, seq=4/1024, ttl=63
18 14.630822  172.16.21.1   172.16.20.1   ICMP   98 Echo (ping) reply    id=0xeb18, seq=4/1024, ttl=64
19 15.630792  172.16.20.1   172.16.21.1   ICMP   98 Echo (ping) request  id=0xeb18, seq=5/1280, ttl=63
20 15.630914  172.16.21.1   172.16.20.1   ICMP   98 Echo (ping) reply    id=0xeb18, seq=5/1280, ttl=64
```

Agora os logs do eth1:

```
18 27.668062  172.16.20.1   172.16.21.1   ICMP   98 Echo (ping) request  id=0xeb18, seq=1/256, ttl=64
19 27.668209  172.16.21.1   172.16.20.1   ICMP   98 Echo (ping) reply    id=0xeb18, seq=1/256, ttl=63
20 28.669968  172.16.20.1   172.16.21.1   ICMP   98 Echo (ping) request  id=0xeb18, seq=2/512, ttl=64
21 28.670074  172.16.21.1   172.16.20.1   ICMP   98 Echo (ping) reply    id=0xeb18, seq=2/512, ttl=63
```

Experiência 4:

A quarta experiência tem como objectivo configurar um router comercial e implementar o *Network Address Translation* ou NAT para permitir acesso à internet na nossa rede.

Afim de configurar o router comercial usaram-se os seguintes comandos:

```
configure terminal
```

```
interface gigabitethernet 0/0
ip address 172.16.61.254 255.255.255.0
no shutdown
exit

configure terminal
interface gigabitethernet 0/1
ip address 172.16.1.69 255.255.255.0
no shutdown
exit
configure terminal
ip route 172.16.60.0 255.255.255.0 172.16.61.253
exit
```

Para configurar o NAT introduzimos os seguintes comandos:

```
configure terminal
interface gigabitethernet 0/0
ip address 172.16.61.254 255.255.255.0
no shutdown
ip nat inside
exit

configure terminal
interface gigabitethernet 0/1
ip address 172.16.1.69 255.255.255.0
no shutdown
ip nat outside
exit

configure terminal
access-list 1 permit 172.16.60.0 0.0.0.255
access-list 1 permit 172.16.61.0 0.0.0.255
ip route 0.0.0.0 0.0.0.0 172.16.1.254
ip route 172.16.60.0 255.255.255.0 172.16.61.253
ip nat pool ovrld 172.16.1.69 172.16.1.69 prefix 24
```

```
ip nat inside source list 1 pool ovrld overload
exit
```

A partir deste momento o router irá reescrever os endereços IP da rede interna de modo a poder haver transferência de pacotes entre esta rede e a rede global. Esta "tradução" de endereços globais para endereços locais permite que se alojem muitos mais domínios na internet do que os que se poderiam alojar de outra forma, pois estariam limitados a todas as combinações possíveis de IPs.

Depois de configurado o NAT, fazendo ping a um pc externo à rede local obtemos efectivamente uma resposta, como se pode facilmente constatar no log abaixo.

5.090847	172.16.60.1	172.16.1.254	ICMP	98 Echo (ping) request id=0x5e24, seq=1/256, ttl=63
5.091492	172.16.1.254	172.16.60.1	ICMP	98 Echo (ping) reply id=0x5e24, seq=1/256, ttl=63
5.817173	Cisco_3a:f1:05	Cisco_3a:f1:05	LOOP	60 Reply
6.014835	Cisco_3a:f1:05	Spanning-tree-(for-bridges)_00	STP	60 Conf. Root = 32768/61/fc:fb:fb:3a:f1:00 Cost = 0
6.092831	172.16.60.1	172.16.1.254	ICMP	98 Echo (ping) request id=0x5e24, seq=2/512, ttl=63
6.093234	172.16.1.254	172.16.60.1	ICMP	98 Echo (ping) reply id=0x5e24, seq=2/512, ttl=63
7.092870	172.16.60.1	172.16.1.254	ICMP	98 Echo (ping) request id=0x5e24, seq=3/768, ttl=63
7.093285	172.16.1.254	172.16.60.1	ICMP	98 Echo (ping) reply id=0x5e24, seq=3/768, ttl=63

- **Experiência 5:**

Esta experiência tem como objectivo principal o uso do *Domain Name System* (vulgarmente DNS). Este consiste num sistema de "tradução" de nomes de domínio em endereços IP baseado em servidores espalhados por todo o mundo que contêm uma base de dados com todos os domínios e seus respectivos endereços IP.

O processo de configuração é bastante simples e consiste apenas em introduzir dois comandos no ficheiro de configuração como exemplificado abaixo.

```
add to /etc/resolv.conf
search lixa.netlab.fe.up.pt
nameserver 172.16.1.2
```

Assim, depois desta configuração efectuada, ao fazer ping a um nome de domínio

(suponhamos google.com), o primeiro pacote enviado seguirá o protocolo UDP (0x11) para o endereço de um servidor DNS que responderá fornecendo o IP correspondente ao domínio enviado (ou com uma mensagem de erro, caso tal não exista). Estando na posse do IP a instrução ping funciona exactamente como se fosse chamada com o IP ao qual se pretende fazer ping.

- **Experiência 6:**

Esta experiência visava testar a aplicação desenvolvida de modo a tirar algumas conclusões acerca do funcionamento do protocolo FTP. Lamentavelmente não conseguimos testar a nossa aplicação como era pedido, conseguindo apenas que ela funcionasse em ambiente Linux, sendo esses logs que iremos mostrar uma vez que nos permitem tirar grande parte das conclusões pedidas. Retiramos então o seguinte log:

```
22 16.6532000 192.168.0.102   208.67.222.222   DNS       69 Standard query 0xc68b  A ftp.up.pt
23 16.7125240 208.67.222.222  192.168.0.102    DNS       85 Standard query response 0xc68b  A 193.136.37.8
24 16.7128530 192.168.0.102   193.136.37.8     TCP       74 54704 > ftp [SYN] Seq=0 Win=14600 Len=0 MSS=1460 SACK_PERM=1 TSval=771296 TSecr=0 WS=128
25 16.7334870 193.136.37.8    192.168.0.102    TCP       70 ftp > 54704 [SYN, ACK] Seq=0 Ack=1 Win=5792 Len=0 MSS=1380 TSval=544143128 TSecr=771296
26 16.7335220 192.168.0.102   193.136.37.8     TCP       66 54704 > ftp [ACK] Seq=1 Ack=1 Win=14600 Len=0 TSval=771302 TSecr=544143128
27 16.7580850 193.136.37.8    192.168.0.102    FTP      106 Response: 220 Bem-vindo \303\240 Universidade do Porto
28 16.7581450 192.168.0.102   193.136.37.8     TCP       66 54704 > ftp [ACK] Seq=1 Ack=41 Win=14600 Len=0 TSval=771309 TSecr=544143136
29 17.7337860 192.168.0.102   193.136.37.8     FTP       82 Request: USER anonymous
30 17.7568120 193.136.37.8    192.168.0.102    TCP       66 ftp > 54704 [ACK] Seq=41 Ack=17 Win=5792 Len=0 TSval=544143385 TSecr=771602
31 17.7568660 193.136.37.8    192.168.0.102    FTP      100 Response: 331 Please specify the password.
32 17.7568880 192.168.0.102   193.136.37.8     TCP       66 54704 > ftp [ACK] Seq=17 Ack=75 Win=14600 Len=0 TSval=771609 TSecr=544143385
33 18.7340390 192.168.0.102   193.136.37.8     FTP       77 Request: PASS pass
34 18.7579760 193.136.37.8    192.168.0.102    FTP       89 Response: 230 Login successful.
35 18.7580390 192.168.0.102   193.136.37.8     TCP       66 54704 > ftp [ACK] Seq=28 Ack=98 Win=14600 Len=0 TSval=771909 TSecr=544143636
36 19.7342810 192.168.0.102   193.136.37.8     FTP       72 Request: pasv
37 19.7559970 193.136.37.8    192.168.0.102    FTP      116 Response: 227 Entering Passive Mode (193,136,37,8,104,180)
38 19.7561310 192.168.0.102   193.136.37.8     TCP       66 54704 > ftp [ACK] Seq=34 Ack=148 Win=14600 Len=0 TSval=772209 TSecr=544143885
39 19.7562780 192.168.0.102   193.136.37.8     TCP       74 60781 > 26804 [SYN] Seq=0 Win=14600 Len=0 MSS=1460 SACK_PERM=1 TSval=772209 TSecr=0 WS=128
40 19.7774380 193.136.37.8    192.168.0.102    TCP       70 26804 > 60781 [SYN, ACK] Seq=0 Ack=1 Win=5792 Len=0 MSS=1380 SACK_PERM=1 TSval=544143891 TSecr=772209
41 19.7775190 192.168.0.102   193.136.37.8     TCP       66 60781 > 26804 [ACK] Seq=1 Ack=1 Win=14600 Len=0 TSval=772215 TSecr=544143891
42 19.7776360 192.168.0.102   193.136.37.8     FTP       88 Request: RETR /pub/robots.txt
43 19.7998280 193.136.37.8    192.168.0.102    FTP-DAT   89 FTP Data: 23 bytes
44 19.7998500 192.168.0.102   193.136.37.8     TCP       66 60781 > 26804 [ACK] Seq=1 Ack=24 Win=14600 Len=0 TSval=772222 TSecr=544143895
45 19.7998650 193.136.37.8    192.168.0.102    TCP       66 26804 > 60781 [FIN, ACK] Seq=24 Ack=1 Win=5792 Len=0 TSval=544143895 TSecr=772215
46 19.7998710 193.136.37.8    192.168.0.102    FTP      139 Response: 150 Opening BINARY mode data connection for /pub/robots.txt (23 bytes).
```

Assim, como podemos ver no *log* acima, os primeiros pacotes enviados pela aplicação são um pedido DNS (explicação detalhada na experiência 5). Descobrindo qual o IP a aplicação esta estabelece uma conexão FTP e começa a enviar e receber pacotes FTP (que usam o protocolo TCP como "meio de transporte", daí aparecerem sempre duas entradas, uma TCP e outra FTP por comunicação no *Wireshark*).

Repare-se que a partir de certo ponto é aberta outra conexão. Esta é criada a partir dos dados devolvidos pelo servidor ftp quando entramos em modo passivo. Esta nova

ligação serve exclusivamente para transferir os dados, sendo totalmente independente da primeira que é usada para enviar os comandos.

5. Conclusões

Findo este trabalho laboratorial, é possível concluir que foi desenvolvido com sucesso, tendo contribuído para um assimilar de conhecimentos notável por parte de todos os elementos do grupo. Foi um trabalho bastante completo, permitindo aplicar vários conceitos que eram necessários para concluir as experiências que constavam no enunciado.

A outro nível, foi uma soberba oportunidade para que os elementos do grupo entrassem em contacto com equipamentos não muito vulgares cujo manuseamento corrente é ainda mais raro. Para além do material utilizado, que exigiu um período de habituação razoável, usou-se também a ferramenta *Wireshark* que demonstrou um enorme potencial para analisar os pacotes de rede, conceito que usamos diariamente praticamente sem reparar.

Para finalizar, gostávamos apenas de ressalvar que este trabalho pode vir a ser muito importante pois é o primeiro na área das Redes de Computadores e pode vir a perspectivar aos elementos do grupo um futuro académico ou profissional relacionado com esse mesmo tema.

6. Bibliografia

[1] Vários autores, *File Transfer Protocol - Wikipedia, the free encyclopedia*, http://en.wikipedia.org/wiki/File_Transfer_Protocol (Consultado em 18 de Dezembro de 2012);

[2] Microsoft, *SOCKADDR_IN Structure (MFC)*, http://msdn.microsoft.com/en-us/library/zx63b042(v=vs.80).aspx (Consultado em 18 de Dezembro de 2012);

[3] Tenouk, *An introduction to Linux socket and network Application Programming Interface (API) functions*, http://www.tenouk.com/Module39b.html (Consultado em 18 de Dezembro de 2012);

[4] Vários autores, *Address resolution protocol - Wikipedia, the free encyclopedia*, http://en.wikipedia.org/wiki/Address_Resolution_Protocol (Consultado em 18 de Dezembro de 2012);

[5] Vários autores, *Loopback - Wikipedia, the free encyclopedia*, http://en.wikipedia.org/wiki/Loopback (Consultado em 18 de Dezembro de 2012);

[6] Kozierok, Charles M. , *The TCP/IP Guide - IP Network Address Translation (NAT) Protocol*, http://www.tcpipguide.com/free/t_IPNetworkAddressTranslationNATProtocol.htm (Consultado em 18 de Dezembro de 2012);

[7] Vários autores, *Domain Name System - Wikipedia, the free encyclopedia*, http://en.wikipedia.org/wiki/Domain_Name_System (Consultado em 18 de Dezembro de 2012);

7. Anexos

Em anexo segue o código fonte da aplicação desenvolvida no âmbito deste trabalho laboratorial e de nome *ftp.c*.