Rede de Computadores

Relatório
Trabalho2

Universidade do Porto

Faculdade de Engenharia

FEUP

Mestrado Integrado em Engenharia Informática e Computação

Redes de Computadores

Hugo Ari Rodrigues Drumond — 201102900 — hugo.drumond@fe.up.pt
José Pedro Pereira Amorim — 201206111 — ei12190@fe.up.pt
João Ricardo Pintas Soares — 201200740 — ei12039@fe.up.pt

Faculdade de Engenharia da Universidade do Porto
Rua Roberto Frias, 4200–65 Porto, Portugal

28 de Dezembro de 2014

1 Introdução

Este trabalho laboratorial, desenvolvido no âmbito da Unidade Curricular de Redes de Computadores (RCOM), teve como objetivo desenvolver uma aplicação ftp e compreender/implementar/configurar uma rede de computadores. Ao longo deste relatório, serão descritos os aspetos fundamentais do referido trabalho, permitindo obter um conhecimento detalhado deste. Os ficheiros de análise usados para chegar às conclusões presentes ao longo do relatório encontram-se na pasta parte2 e o código da aplicação ftp na pasta parte1. Só não foram colocados os ficheiros da última experiência devido ao tamanho dos mesmos.

2 Parte 1 - Desenvolvimento de uma aplicação de download

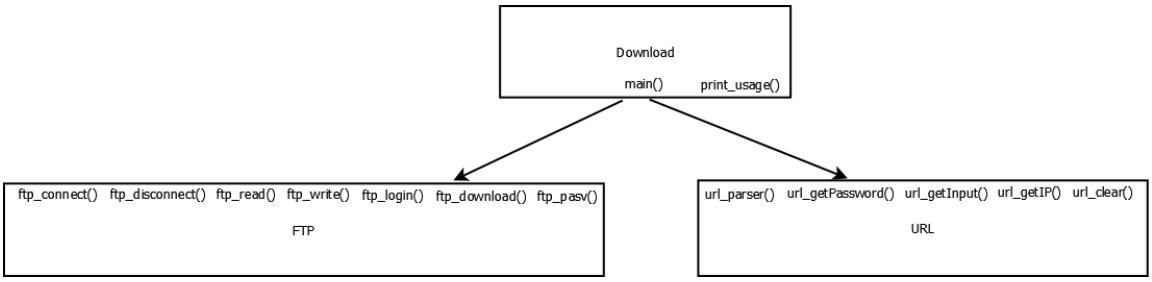

Figura 1: Arquitetura da aplicação download

A aplicação consiste em três componentes: FTP, URL e download. O componente FTP é responsável pela comunicação com o servidor FTP usando Berkeley Sockets. As suas funções fornecem os meios para o estabelecimento das conexões TCP (controlo e dados), a terminação dessas conexões, e a transferência de dados. Este componente não tem visibilidade para nenhum dos outros componentes da aplicação. O componente URL é responsável por fazer parser do URL FTP passado para a aplicação como argumento na linha de comandos, retirando o host, path, user e password. Caso um, ou vários, destes parâmetros estejam em falta, URL também fornece os meios para os obter perguntando ao utilizador para os introduzir. Este componente também não tem visibilidade para nenhum dos outros componentes da aplicação. O componente download utilza os outros componentes para fazer o download do ficheiro.

2.0.1 Componente FTP

O componete FTP estabelece a ligação FTP na função ftp_connect(), utilizando Berkeley Sockets para estabelecer as conexões TCP. O file descriptor data_socket_fd corresponde à conexão de dados e control_socket_fd à conexão de controlo.

2.0.2 Componente URL

Cada um das strings da estrutura URL correspondem a parâmetros necessários no estabelecimento da conexão FTP e no download do ficheiro e são retirados do URL recebido como argumento da linha de comandos ou inserida pelo utilizador. Caso o utilizador não tenha especificado a password no argumento da linha de comandos, então é pedida sem haver echo, não sendo mostrada no monitor e sendo uma alternativa mais segura para a introdução da password.

2.0.3 Successful Download

Foi testado a aplicação de download usando vários servidores FTP da FEUP, incluindo ftp.up.pt, pinguim.fe.up.pt e mirrors.fe.up.pt. Todos os download foram efetuados com sucesso, incluindo os testados na experiência 6 e na demostração da aplicação.

3 Parte 2 - Configuração e Estudo de uma Rede

3.1 Experiência 1 - Configurar um rede IP

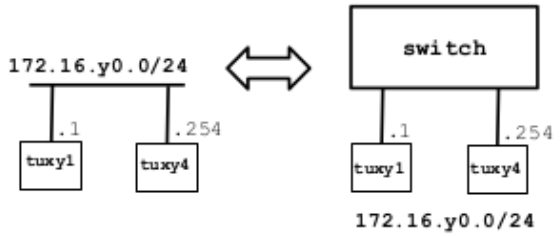

Figura 2: Arquitetura da experiência 1

3.1.1 Objetivos

Tentar interligar dois computadores através de um switch. E entender o funcionamento do protocolo ARP e do comando ping.

3.1.2 Comandos principais de configuração

Foi dado reset aos equipamentos:

```
Computadores,
    updateimage
Switch,
    no vlan 1-4094
    copy flash:tux5-clean startup-config
    reload
```

E ligados os cabos do tux1(eth0) e tux4(eth0) ao switch. E configurados os ips de cada uma dessas interfaces:

```
Tux1,
    ifconfig eth0 172.16.50.1/24
Tux4,
    ifconfig eth0 172.16.50.254/24
```

E apagadas cada uma das linhas da arp table para cada um dos pcs:

```
arp -d ipaddress
```

3.1.3 Análise das capturas

Uma vez que foram apagadas as linhas da arp table torna-se crucial investigar que máquina tem um dado ip. Tal é visível nos logs do wireshark, devido ao ping que fizemos para o tux4(ping 172.16.50.254). Do tux1 é feito um broadcast de uma trama do tipo ARP (Opcode) Request, que contém: o endereço IP do alvo; e a sua informação, ip e mac. Ficando à espera da resposta do alvo. Quando o tux4 capta esta mensagem, aceita-a porque tem o mesmo ip que o ip destino, e envia em unicast uma trama do tipo ARP Reply contendo o seu endereço MAC, juntamento com todos os outros dados.

```
31 43.06416800( G-ProCom_8b:e4:a7    Broadcast            ARP          60 Who has 172.16.1.254?  Tell 172.16.1.1
32 43.06418000( Hewlett- c3:78:70    G-ProCom 8b:e4:a7    ARP          42 172.16.1.254 is at 00:21:5a:c3:78:70
```

Figura 3: Trama EthernetII do tipo ARP

```
▾ Address Resolution Protocol (request)
    Hardware type: Ethernet (1)
    Protocol type: IP (0x0800)
    Hardware size: 6
    Protocol size: 4
    Opcode: request (1)
    Sender MAC address: G-ProCom_8b:e4:a7 (00:0f:fe:8b:e4:a7)
    Sender IP address: 172.16.1.1 (172.16.1.1)
    Target MAC address: 00:00:00_00:00:00 (00:00:00:00:00:00)
    Target IP address: 172.16.1.254 (172.16.1.254)
```

Figura 4: Payload da trama EthernetII do tipo ARP, mais concretamente Request

```
▾ Address Resolution Protocol (reply)
    Hardware type: Ethernet (1)
    Protocol type: IP (0x0800)
    Hardware size: 6
    Protocol size: 4
    Opcode: reply (2)
    Sender MAC address: Hewlett-_c3:78:70 (00:21:5a:c3:78:70)
    Sender IP address: 172.16.1.254 (172.16.1.254)
    Target MAC address: G-ProCom_8b:e4:a7 (00:0f:fe:8b:e4:a7)
    Target IP address: 172.16.1.1 (172.16.1.1)
```

Figura 5: Payload da trama EthernetII do tipo ARP, mais concretamente Reply

Quando o tux1 recebe esta mensagem é inserida uma entrada na sua ARP table, com o IP (network layer) e mac address (link layer) do tux4. O mesmo acontece ao tux4 se tiver a ARP table sem uma linha para 172.16.50.1. Para além disto, observámos que o comando ping gera um pacote Echo Request e espera por um Echo Reply do host alvo, sendo que os endereços do network layer e da link layer estão bem definidos e encontram-se respetivamente: na header do pacote, e no header do EthernetII Frame(depende das máquinas pode onde a trama passa). Verificou-se também que há um campo na header da EthernetII frame que indica o tipo de protocolo, por exemplo se é ARP, IP ou ICMP. No entanto, não existe nenhum campo em que seja especificado o tamanho da trama (ao contrário do que acontece no pacote IP). Isto, deve-se ao facto do Physical Layer ter mecanismos de reconhecimento do início e fim de uma trama, dependentes da codificação e do meio de transmissão. Concluímos também que não é necessário haver um switch para uma máquina falar com ela própria, uma vez que existe uma rede de interfaces virtuais(loopback) que permitem isso mesmo. Necessário para: testar configurações; correr servidores localmente; fazer simulações de redes; etc. De modo rápido.

3.2 Experiência 2 - Implementar de duas LANs virtuais num switch

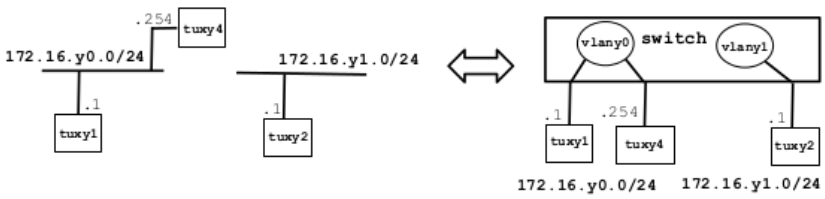

Figura 6: Arquitetura da experiência 2

3.2.1 Objetivos

Criação de duas LANs virtuais no switch, uma constituída pelo tuxy1 e tuxy4 e outro pelo tuxy2. Visto que se encontram em sub-redes diferentes, tuxy1 e tuxy4 não conseguem comunicar com tuxy2.

3.2.2 Comandos principais de configuração

Para configurar o switch conectou-se a porta série de uma das máquinas com o switch e usou-se o programa gtkterm para inserir os comandos necessários. Para adicionar cada uma das vlans ao switch, entrou-se na consola de configuração e criou-se a vlan com o identificador x0 usando os comandos:

```
configure terminal
vlan x0
end
show vlan id x0
```

Cada máquina foi ligada a uma porta do switch, configurou-se uma Fast Ethernet Interface para cada porta, e ligou-se as portas cujas máquinas tuxy1 e tuxy4 se conectaram à vlany0 e a porta da máquina tuxy2 à vlany1, usandos os comandos:

```
configure terminal
interface fastEthernet 0/porta
switchport mode access
switchport access vlan x0
end
```

Verificando as configurações:

```
show running-config interface fastEthernet 0/porta
show interfaces fastEthernet 0/porta switchport
```

Por exemplo:

```
50 VLAN0050 active Fa0/1, Fa0/2
51  VLAN0051 active Fa0/3
```

De modo a não serem ignorados broadcasts de icmp echo:

```
echo 0 > /proc/sys/net/ipv4/icmp_echo_ignore_broadcasts
```

3.2.3 Análise das capturas

Foi executando um ping da máquina tuxy1 para a máquina tuxy4 o que foi possível visto que se encontram na mesma vlan, e o ping entre tux1 e tuxy2 não foi executado com sucesso, pois pertencem a vlans diferentes. Existem dois domínios de broadcast, devido a existir duas vlans (sem interligação entre elas). Através dos logs verificamos que quando fazemos broadcast no tuxy1 só o tuxy4 responde. E quando fazemos broadcast no tuxy2 nenhuma máquina responde, porque não chega nenhum pedido às outras máquinas.

3.3 Experiência 3 - Configurar um Router em Linux

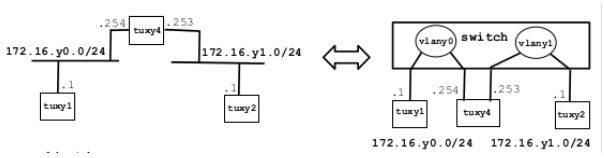

Figura 7: Arquitetura da experiência 3

3.3.1 Objetivos

Transformar a máquina tuxy4 num router, permitindo a comunicação entre as duas sub-redes criadas na experiência anterior.

3.3.2 Comandos principais de configuração

Foi configurada a interface eth1 da máquina tuxy4 com IP 172.16.y1.253 e adicionada à vlan do tuxy2 (vlany1).
Para que possa haver routing da parte do tuxy4 e para haver comunicação entre as sub-redes ativou-se o IP fowarding no tuxy4(router):

```
echo 1 > /proc/sys/net/ipv4/ip_foward
```

No tuxy1 foi adicionada uma rota para a vlany1:

```
route add -net 172.16.y1.0/24 gw 172.16.y0.254
route -n
```

Destination	Gateway	Genmask	Flags	Metric	Ref	Use	Iface
172.16.50.0	0.0.0.0	255.255.255.0	U	0	0	0	eth0
172.16.51.0	172.16.50.254	255.255.255.0	UG	0	0	0	eth0

Esta entrada na routing table faz com que haja um redirecionamento dos ips to tipo 172.16.y1.0/24 para o gateway router 172.16.y0.254(eth0 tuxy4). A primeira entrada da routing table tem o significado de aceitar o tráfego proveniente de 172.16.50.0/24.
Faz-se o mesmo para o tuxy2, adicionando uma rota para a vlany0:

```
route add -net 172.16.y0.0/24 gw 172.16.y1.253
route -n
```

Destination	Gateway	Genmask	Flags	Metric	Ref	Use	Iface
172.16.50.0	172.16.51.253	255.255.255.0	UG	0	0	0	eth0
172.16.51.0	0.0.0.0	255.255.255.0	Ui	0	0	0	eth0

Faz forward dos ips 172.16.y0.0/24 para gateway router 172.16.y1.253(eth1 tuxy4). Aceita os ips provenientes da rede 172.16.51.0/24.

Tux4 funciona como um gateway router, pois redireciona de uma interface (eth0) para outra (eth1).

Destination	Gateway	Genmask	Flags	Metric	Ref	Use	Iface
172.16.50.0	0.0.0.0	255.255.255.0	U	0	0	0	eth0
172.16.51.0	0.0.0.0	255.255.255.0	U	0	0	0	eth1

Aceita o tráfego proveniente destas redes e não envia para nenhum gateway visto que os clientes estão ligados diretamente, isto é, não há um decremento do ttl.
Para que os pacotes echos ICMP enviados em broadcasts não sejam automaticamente ignorados, é usado o seguinte comando:

```
echo 0 > /proc/sys/net/ipv4/icmp_echo_ignore_broadcasts
```

MAC addresses observadas:

```
HWaddr 00:21:5a:c3:78:70, é o mac address da interface eth0(172.16.50.254) do tux4.
HWaddr 00:c0:df:08:d5:b0, é o mac address da interface eth1(172.16.51.253) do tux4.
HWaddr 00:0f:fe:8b:e4:a7, é o mac address da interface eth0(172.16.50.1) do tux1.
HWaddr 00:21:5a:61:2f:d6, é o mac address da interface eth0(172.16.51.1) do tux2.
```

3.3.3 Análise das capturas

Pode-se observar nos logs que naturalmente depois da configurações referidas é possível fazer ping do tuxy1 para todas as outras interfaces. Após limpar as ARP tables das três máquinas fez-se ping do tuxy1 para o tuxy2, e é possível observar nos logs que uma vez que a ethernet usa MAC addresses e a ARP table foi limpa, o tuxy1 precisa de saber qual o mac address da interface eth0 do tuxy4, o tuxy4 precisa de saber o mac do tuxy2, o tuxy2 o mac do eth1 tuxy4, e o tuxy4 o mac do tuxy1. Tuxy1 envia um ARP request para obter o MAC address de eth0 tuxy4 e tuxy2 envia um ARP request a eth1 tuxy4. É possível observar que os pacotes ICMP echo, request(tipo 8) e reply(tipo 0) presentes nos logs, são reencaminhados para o tuxy4. Estes são visíveis pelo tuxy4, em ambas as interfaces. Acontece uma coisa um pouco estranha que é o facto do tux3 parecer estar a fazer reply sem antes saber o mac do tux4 eth1 (através dos logs do wireshark). Só depois de algum tempo é que se vê o broadcast ARP a partir do tux3. Pensamos que isto é devido ao tux3 ter respondido ao broadcast do tux4, e de ter ficado de modo implícito com o mac do tux4 eth1 em cache, sendo só depois feito o pedido formal. O mesmo acontece do tux4 para o tux1.

52	80.42657100(	G-ProCom_8b:e4:a7	Broadcast	ARP	60 Who has 172.16.50.254? Tell 172.16.50.1
53	80.42659600(	Hewlett-_c3:78:70	G-ProCom_8b:e4:a7	ARP	42 172.16.50.254 is at 00:21:5a:c3:78:70
54	80.42693800(	172.16.50.1	172.16.51.1	ICMP	98 Echo (ping) request id=0x1134, seq=1/256, ttl=64 (reply in 55)
55	80.42725600(	172.16.51.1	172.16.50.1	ICMP	98 Echo (ping) reply id=0x1134, seq=1/256, ttl=63 (request in 54)
56	81.42785700(	172.16.50.1	172.16.51.1	ICMP	98 Echo (ping) request id=0x1134, seq=2/512, ttl=64 (reply in 57)
57	81.42802900(	172.16.51.1	172.16.50.1	ICMP	98 Echo (ping) reply id=0x1134, seq=2/512, ttl=63 (request in 56)
58	82.19351700(	Cisco_3a:f6:04	Spanning-tree-(for-br:	STP	60 Conf. Root = 32768/50/fc:fb:fb:3a:f6:00 Cost = 0 Port = 0x8004
59	82.42840100(	172.16.50.1	172.16.51.1	ICMP	98 Echo (ping) request id=0x1134, seq=3/768, ttl=64 (reply in 60)
60	82.42855800(	172.16.51.1	172.16.50.1	ICMP	98 Echo (ping) reply id=0x1134, seq=3/768, ttl=63 (request in 59)
61	83.42838400(	172.16.50.1	172.16.51.1	ICMP	98 Echo (ping) request id=0x1134, seq=4/1024, ttl=64 (reply in 62)
62	83.42852400(	172.16.51.1	172.16.50.1	ICMP	98 Echo (ping) reply id=0x1134, seq=4/1024, ttl=63 (request in 61)
63	84.19837300(	Cisco_3a:f6:04	Spanning-tree-(for-br:	STP	60 Conf. Root = 32768/50/fc:fb:fb:3a:f6:00 Cost = 0 Port = 0x8004
64	84.42837800(	172.16.50.1	172.16.51.1	ICMP	98 Echo (ping) request id=0x1134, seq=5/1280, ttl=64 (reply in 65)
65	84.42851800(	172.16.51.1	172.16.50.1	ICMP	98 Echo (ping) reply id=0x1134, seq=5/1280, ttl=63 (request in 64)
66	85.32250200(	Cisco_3a:f6:04	Cisco_3a:f6:04	LOOP	60 Reply
67	85.42839100(	172.16.50.1	172.16.51.1	ICMP	98 Echo (ping) request id=0x1134, seq=6/1536, ttl=64 (reply in 68)
68	85.42853200(	172.16.51.1	172.16.50.1	ICMP	98 Echo (ping) reply id=0x1134, seq=6/1536, ttl=63 (request in 67)
69	85.43481200(	Hewlett-_c3:78:70	G-ProCom_8b:e4:a7	ARP	42 Who has 172.16.50.1? Tell 172.16.50.254
70	85.43514700(	G-ProCom_8b:e4:a7	Hewlett-_c3:78:70	ARP	60 172.16.50.1 is at 00:0f:fe:8b:e4:a7
71	86.20321500(	Cisco_3a:f6:04	Spanning-tree-(for-br:	STP	60 Conf. Root = 32768/50/fc:fb:fb:3a:f6:00 Cost = 0 Port = 0x8004
72	86.42837000(	172.16.50.1	172.16.51.1	ICMP	98 Echo (ping) request id=0x1134, seq=7/1792, ttl=64 (reply in 73)
73	86.42852800(	172.16.51.1	172.16.50.1	ICMP	98 Echo (ping) reply id=0x1134, seq=7/1792, ttl=63 (request in 72)

Figura 8: Comunicações na interface eth0 do Tux4

3.4 Experiência 4 - Configurar um Router Comercial e Implementar o NAT

3.4.1 Objetivos

Configurar um router comercial configurado com NAT de modo a termos accesso à internet através da topologia da rede criada.

3.4.2 Comandos principais de configuração

Dar reset aos computadores e ao switch através dos comandos já apresentados.

Dar reset ao router através do seguinte comando:

```
copy flash:tux5-clean startup-config
reload
```

Configurar cada pc com o ip correcto:

```
tux1,
    ifconfig eth0 172.16.50.1/24
tux2,
    ifconfig eth0 172.16.51.1/24
tux4,
    ifconfig eth0 172.16.50.254/24
    ifconfig eth1 172.16.51.253/24
```

Configurar as routes para os pcs:

```
tux1,
    route add default gw 172.16.50.254
tux2,
    route add default gw 172.16.51.254
    route add -net 172.16.50.0/24 gw 172.16.51.253
tux4,
    route add default gw 172.16.51.254
```

Criar as vlans como anteriormente demonstrado. E permitir o redirecionamento de pacotes no tux4:

```
echo 1 > /proc/sys/net/ipv4/ip_forward
```

Configurar o router cisco com nat:

```
conf t
interface gigabitethernet 0/0
ip address 172.16.51.254 255.255.255.0
no shutdown
ip nat inside # Interface do lado lan
exit
interface gigabitethernet 0/1
ip address 172.16.1.59 255.255.255.0
no shutdown
ip nat outside # Interface do lado ‘‘wan’’
exit
ip nat pool ovrld 172.16.1.59 172.16.1.59 prefix 24
```

```
ip nat inside source list 1 pool ovrld overload
# Esta permissão não deixa que o tux4 tenha acesso à net
# 0.0.0.255 para as permissões serem iguais para os hosts
access-list 1 permit 172.16.50.0 0.0.0.7
access-list 1 permit 172.16.51.0 0.0.0.7
# Default gateway do router com nat
ip route 0.0.0.0 0.0.0.0 172.16.1.254
# Para falar com os hosts do outro lado do router
ip route 172.16.50.0 255.255.255.0 172.16.51.253
end
```

3.4.3 Análise das capturas

Na primeira captura do wireshark observamos que são descobertos os mac adresses através do protocolo layer 2, ARP. Ao remover a route 172.16.y0.0/24 via tux4 do tux2, observámos que o pacote Echo Reply passa pelo router cisco, que por sua vez redireciona para o router linux(tux4), chegando por fim ao tux1 por forwarding do tux4. A introdução dessa route previne passar pelo router cisco, aumentando assim a velocidade da conexão. Esta diferença deve-se ao algoritmo das forwarding tables e das entradas presentes em cada máquina. Verificou-se também que, quando o nat não está implementado é possível fazer pedidos para fora da nossa rede, só que a resposta nunca chega.

```
4 2.690312000 172.16.50.1          172.16.1.254          ICMP         98 Echo (ping) request  id=0x15bc, seq=1/256, ttl=64 (no response found!)
5 3.689672000 172.16.50.1          172.16.1.254          ICMP         98 Echo (ping) request  id=0x15bc, seq=2/512, ttl=64 (no response found!)
```

Figura 9: Não há resposta sem NAT

Isto, deve-se ao facto do router cisco não saber para que host enviar uma dada trama. Daí ser necessário a funcionalidade NAT. Como constatámos, esta técnica permite utilizar os mesmos ips em diferentes redes, chamadas redes privadas(10.0.0.0/8, 172.16.0.0/12, 192.168.0.0/16). Isto é possível porque os ips privados são traduzidos para um público. Este mapeamento é feito através de uma tabela chamada tabela de tradução de endereços de rede, que possibilita que o router saiba para que hosts são destinados certos pacotes vindos da internet. Cada entrada da tabela tem uma parte para o lado WAN e outra para o lado LAN, cada um deles tem dois campos: um para o ip e a outra para a porta. A porta é o que permite fazer a distinção entre destinos de pacotes. É possível reencaminhar todos os pacotes wan com destino a uma dada porta (por exemplo, porta default para o server http, 80) para uma dada porta de um host, que terá um processo em modo listening associado a essa porta.

```
 9 8.074267000 172.16.50.1          172.16.1.254          ICMP         98 Echo (ping) request  id=0x1796, seq=1/256, ttl=64 (reply in 10)
10 8.075766000 172.16.1.254         172.16.50.1           ICMP         98 Echo (ping) reply    id=0x1796, seq=1/256, ttl=62 (request in 9)
```

Figura 10: Há resposta visto NAT estar configurado

3.5 Experiência 5 - DNS

3.5.1 Objetivos

Configurar o DNS de modo a conseguir acessar a redes externas, acedendo à Internet através da rede previamente criada. A arquitetura da rede continuou igual à da experiência anterior.

3.5.2 Comandos principais de configuração

Para configurar o serviço DNS alterou-se o ficheiro /etc/resolv.conf, em cada uma das máquinas. Nesta experiência o conteúdo deste ficheiro foi alterado para:

```
search lixa.netlab.fe.up.pt
nameserver 172.16.1.1
```

172.16.1.1, é o endereço do servidor de nomes para onde se fazem os pedidos. lixa.netlab.fe.up.pt, é o único elemento da lista de procura de hostnames.

3.5.3 Análise das capturas

É possível verificar a troca de pacotes DNS no início da ligação à Internet: Através dos logs, verificámos que existem dois tipos de pacotes DNS: Queries, Responses. Os pacotes query e reponses(replies) têm um formato comum. Para além dos 12 bytes do cabeçalho: identificação; flags; número de perguntas; número de respostas de RRs; número de registos de servidores authoritative; e, número adicional de RRs. São enviadas: perguntas com os campos nome e tipo(A, NS, CNAME e MX) selecionados; respostas a uma pergunta, RRs completos; registo de servidores authoritative; e, informação adicional útil. Não encontrámos nenhum pacote DNS do tipo update, no entanto a estrutura do pacote é idêntica.

```
2 1.233423000  172.16.40.1          8.8.8.8              DNS          69 Standard query 0x59fa  A ftp.up.pt
3 1.243386000  8.8.8.8              172.16.40.1          DNS          85 Standard query response 0x59fa  A 193.136.37.8
```

Figura 11: Pacotes DNS - Query e Response

Query:

```
ftp.up.pt: type A, class IN
```

Response:

```
ftp.up.pt: type A, class IN, addr 193.136.37.8, Time to live: 3616, Data length: 4
```

3.6 Experiência 6 - Ligações TCP

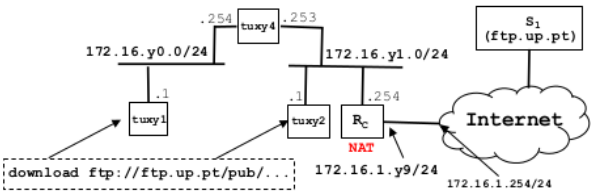

Figura 12: Arquitetura da experiência 6

3.6.1 Objetivos

Estudo das ligações TCP de dados e controlo e as suas fases; estudo das ligações FTP; Estudo do mecanismo TCP ARQ e mecanismo de controlo de congestionamento TCP.

3.6.2 Comandos principais de configuração

Foi compilada a aplicação de download desenvolvida e descrita na parte 1 do presente relatório e usada para o download de um ficheiro presente num servidor FTP. O download foi concluído com sucesso, e os logs foram analisados.
São abertas duas conexões TCP pela aplicação FTP, uma de controlo e outra de dados. Primeiramente a aplicação cria uma conexão TCP de controlo numa porta sem privilégios (porta $>$ 1023)

que conecta à porta FTP do servidor (porta 21) que vai continuar aberta durante a transferência. Foi utilizado nesta aplicação o modo passivo por oposição ao modo activo. No modo activo o cliente envia o numero da porta(de dados) sem privilégios para que o servidor(porta 20) se ligue. Por outro lado, no modo passivo, é o servidor que envia o número de uma das suas portas sem privilégios(> 1023) para que o cliente se conecte(porta cmd cliente + 1 = porta dados).

3.6.3 Análise das capturas

Verificámos que a informação de controlo FTP é transportada na conexão de controlo que liga uma porta aleatória(> 1023) do cliente à porta 21 do servidor.

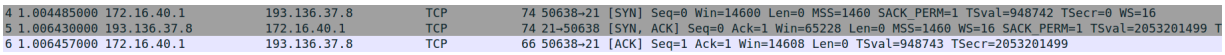

Figura 13: Exemplo de informação de controlo FTP

No nosso caso o cliente tem a porta de controlo 50638. E o servidor ftp a porta 21.

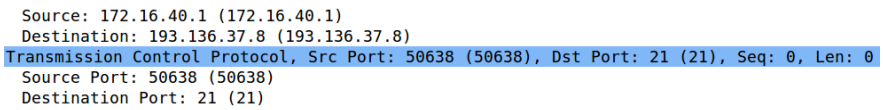

Figura 14: Portas e IPs de controlo cliente -> servidor modo passivo

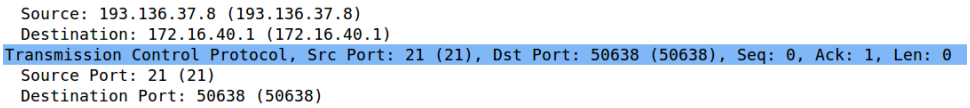

Figura 15: Portas e IPs de controlo servidor -> cliente modo passivo

Concluímos também que as três fases de conexão TCP são: connection estabelishment, data transfer e connection termination. Na parte da connection establishment o cliente manda um SYN para o servidor; recebe uma reposta SYN-ACK; e, finalmente o cliente manda um ACK. Como se pode verificar na primeira imagem desta secção. Para além disso o receptor não faz request de retransmissão de pacotes perdidos. O transmissor espera pelo ACK e quando não o recebe, reenvia os pacotes após um certo intervalo. Pode ser implementado de várias formas (Stop-and-wait, Go-Back-N, Selective Repeat). O TCP header contém campos como a porta fonte e a porta do destinatário, o número de sequência, o número de acknowledgement e o tamanho da window. É atribuído um número de sequência a cada byte de dados, sendo que o número de sequência do TCP header é o do primeiro byte de dados desse pacote. O número de acknowledge indica o número de sequência que o receptor espera do emissor e o tamanho da window indica o número de bytes que o receptor está disposto a receber. Existem vários mecanismos de controlo de congestionamento TCP. Cada um deles possui algoritmos de controlo de congestionamento juntamento com outras abordagens. Por exemplo: Slow start; Congestion avoidance; Fast retransmit; Fast Recovery; etc. Exemplos de mecanismos que usam alguns destes algoritmos: TCP Tahoe; TCP Newreno; TCP Vegas; etc. Este mecanismo pode ser observado em /proc/sys/net/ipv4/tcp_congestion_control. Embora não tenhamos visto qual deles está a ser usado nos pcs das labs sabe-se que este implementa o algoritmo Slow start visto que o throughput evolui até um determinado ponto e diminui quando há congestionamento. Evidenciado através da existência de outra conexão TCP.

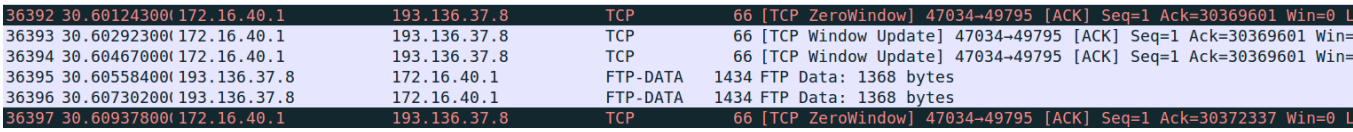

Figura 16: Congestionamento TCP

4 Conclusões

A elaboração deste trabalho contribuiu para a aprendizagem dos conhecimentos básicos do protocolo ftp, tal como o modo de estruturação e configuração de componentes de uma rede de computadores. O projecto desenvolvido resultou numa aplicação capaz de fazer o download de ficheiros via ftp em modo passivo, com ou sem autenticação. E numa rede atual funcional. O trabalho foi realizado com sucesso e com grande entusiasmo, visto que, compreendemos a maneira como o protocolo ftp funciona, como implementá-lo, e como criar a nossa própria rede de computadores. A partir dos logs guardados durante as experiências podemos inspecionar os pacotes transferidos, e adquirir um conhecimento mais aprofundado e mais prático dos protocolos, FTP, TCP, ICMP, e LOOP. Este trabalho permitiu consolidar a matéria dada durante o semestre na cadeira de Redes de Computadores. Goodbye MEO router, welcome openwrt :).