Licenciatura em Engenharia Informática e Computação

Projeto 2 - Redes de Computadores

10 de dezembro de 2022

João Duarte up201707984@fc.up.pt

Diogo Nunes up202007895@fc.up.pt

Nuno Penafort up202008405@fc.up.pt

Índice

1 - Introdução

No âmbito da unidade curricular de Redes de Computador, foi-nos pedido o desenvolvimento de um trabalho laboratorial com o propósito de configurar uma rede, e uma aplicação de download de ficheiros.

Relativamente à configuração de uma rede, o seu objetivo é permitir a execução de uma aplicação, a partir de duas bridges dentro de um switch. De seguida, foi desenvolvida uma aplicação download de acordo com o protocolo FTP e com a ajuda de ligações TCP (Transmission Control Protocol) a partir de sockets.

Este relatório está dividido em duas partes, na primeira é demonstrada toda a arquitetura e o funcionamento da aplicação de download, e, numa segunda parte são analisadas as várias experiências feitas nas aulas teórico-práticas, com um anexo final do código utilizado para este trabalho.

2 Parte 1 - Aplicação de download

O nosso programa foi efetuado em apenas um documento denominado de “proj2.c”, nele foram implementadas todas as funções e processos necessários para a configuração de rede, e download dos ficheiros indicados no servidor FEUP.

2.1 Arquitetura

A parte 1 do projeto consistem em desenvolver um simples programa de download de ficheiros, implementando o protocolo FTP.
Executar o programa é simples basta compilar e correr o programa passando-lhe o link alvo como argumento.
Esse link deve seguir a sintaxe descrita em RFC173:

- ftp://[<user>:<password>]@<host>/<url-path>

O programa começa por dar *parse* ao url, dividindo cada uma das suas componentes numa variável para depois usar o valor quando necessário.

Tal como esta indicado no RFC173, os campos de *user* e *password* são opcionais, e caso sejam omitidos a aplicação fara download em modo *anonymous.*
Se o programa detetar que todos os parâmetros necessários foram inseridos, ele tenta estabelecer uma conexão TCP ao <host>, com as credencias escolhidas (ou *anonymous*).
Depois de efetuado o login o programa envia um pedido de modo passivo (comando PASV) e o servidor, se aceitar envia o número da porta onde o programa deve abrir uma segunda conexão TCP. A segunda conexão é utilizada para a transferência de dados.
Quando o download acabar, o programa fecha as 2 conexões TCP e guarda os dados recebidos num ficheiro local, dando-lhe o mesmo nome que este tinha no servidor.

2.2 Relatório de um download com sucesso

Em baixo encontra-se um exemplo completo de um caso de sucesso de download usando o nosso programa, neste caso sem autenticação no servidor

```
diogo@ubuntu:~/Documents/RCOM/proj2$ gcc proj2.c -o ftpClient
diogo@ubuntu:~/Documents/RCOM/proj2$ ./ftpClient ftp://ftp.up.pt/pub/kodi/timestamp.txt
user:anonymous password:password host:ftp.up.pt path:pub/kodi/timestamp.txt|
Host name  : mirrors.up.pt
IP Address : 193.137.29.15
220-Welcome to the University of Porto's mirror archive (mirrors.up.pt)
220----------------------------------------------------------------
220-
220-All connections and transfers are logged. The max number of connections is 200.
220-
220-For more information please visit our website: http://mirrors.up.pt/
220-Questions and comments can be sent to mirrors@uporto.pt
220-
220-
220

Bytes escritos 16
331 Please specify the password.

230 Login successful.
227 Entering Passive Mode (193,137,29,15,209,147).
Bytes escritos 29
Receiving file timestamp.txt...
226
: Success
```

Figura 2.a: Download de timestamp.txt em modo *anonymous*

3 Parte 2 - Configuração de rede e análise

3.1 - Experiência 1 - Configurar um IP de rede

A objetivo desta experiência foi configurar uma ligação entre o tuxY3 e otuxY4, recorrendo ao switch. Para tal, foram utilizados comandos como **ifconfig** e **route** para ligar as portas eth0 dos dois computadores. Após a configuração foi enviado o sinal "ping" de um computador para o outro para verificar que, de facto, as ligações foram bem realizadas e as máquinas possuíam uma conexão.

Ao capturar pacotes usando o wireshark, enquanto deixamos correr o comando ping podemos observar que os primeiros pacotes enviados são do tipo ARP.

O ARP (Address Resolution Protocol) é um protocolo de comunicação que serve para descobrir o endereço da camada de ligação associado ao endereço IPv4. Serve para mapear o endereço de rede a um endereço físico como o endereço MAC.

Estes pacotes são emitidos para toda a LAN, que neste caso é apenas 2 computadores e 1 bridge. Assim sendo, quando, por exemplo o tux X dá ping ao tux Y, pergunta a todos os tuxs ligados à LAN qual é que possui o IP do tux Y. A pergunta em si é um pacote ARP, que possui o endereço MAC X bem como o endereço IP do tux X e pergunta pelo IP Y.
A resposta contém o endereço IP e MAC do tux Y.

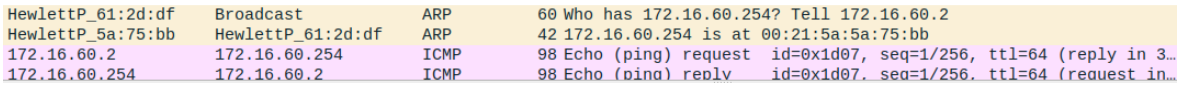

Figura 3.1.a: Pacotes ARP e ICMP

Depois dos pacotes ARP podemos ver que os pacotes seguintes são do tipo ICMP estes endereços contêm os endereços IP e MAC de origem e destino, como pode ser visto na figura anterior.

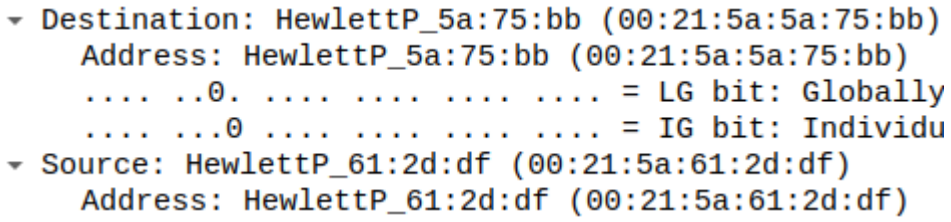

Figura 3.1.b: Endereços MAC contidos no pacote ICMP

Se inspecionarmos os pacotes com mais detalhe reparamos que cada tipo de trama tem cabeçalho diferente. Por exemplo, no caso das tramas do tipo ARP, no cabeçalho o campo type terá o valor 0x0806 e se o valor for 0x0800 é uma trama do tipo IP.

No entanto o datagram do endereço IP engloba o protocolo ICMP, ou seja, se o campo type for preenchido com 1 no header o pacote IP, então é do protocolo ICMP.

Também aqui podemos determinar o comprimento da trama. Como podemos verificar em baixo a trama recebida tem um comprimento de 98 bytes.

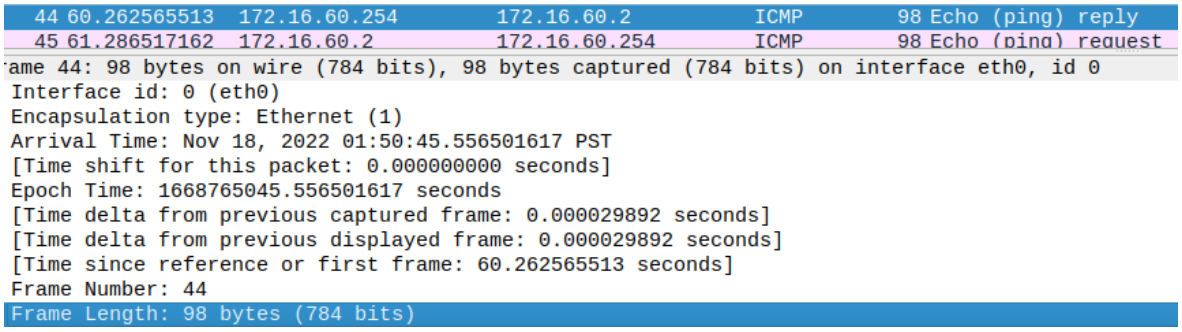

Figura 3.1.c: Detalhes do pacote

Cada tux tem também ao ser dispor uma interface loopback.
A interface loopback é uma interface virtual da rede que está sempre disponível como resultado, isto permite ao computador enviar e receber respostas de si mesmo. Esta interface é útil para testes e configurações locais.

3.2 Experiência 2 - Implementar duas bridges no switch

Para a realização desta experiência tivemos de configurar duas bridges no switch, sendo que na bridgeY0 foram conectados o tuxY3 e o tuxY4 e na bridgeY1 foi conectado o tuxY2. O objetivo desta experiência era verificar que com esta configuração o tuxY2 deixaria de ter acesso aos tuxY3 e tuxY4, uma vez que estas máquinas se encontrariam em sub-redes diferentes.

Começamos por fazer as seguintes conexões com cabos ethernet.

- tuxY3E0 -> porta 1 da switch
- tuxY2E0 -> porta 2 da switch
- tuxY4E0 -> porta 3 da switch
- tuxY4S0 -> T3

- T4 -> console da switch

Abrir GTKTerm em tuxY4 e mudar baudrate para 115200.
Efetuar login no switch e executar os seguintes comandos:

- /system reset-configuration
- /interface bridge add name="bridge0"
- /interface bridge add name="bridge1"

- /interface bridge port remove [find interface=ether1]
- /interface bridge port remove [find interface=ether2]
- /interface bridge port remove [find interface=ether3]

- /interface bridge port add bridge=bridge0 interface=ether1
- /interface bridge port add bridge=bridge1 interface=ether2
- /interface bridge port add bridge=bridge0 interface=ether3

Com esta configuração existem dois domínios de broadcast, bridge 0, que contem tuxY3 e tuxY4 e bridge1 que contem tuxY2.

Podemos concluir isto porque tuxY2 está localizado numa gama de IPs diferente e numa bridge diferente E porque este não consegue comunicar com os outros 2 tuxs, que conseguem comunicar entre si.

3.3 Experiência 3 - Configurar um router em Linux

Esta experiência teve como objetivo configurar o tuxY4 como um router entre as duas sub-redes previamente criadas. Para este efeito, foi necessário ativar *IP forwading* e desativar *ICMP echo-ignore-broadcast*, com os seguintes comandos:

- echo 1 > /proc/sys/net/ipv4/ip_forward
- echo 0 > /proc/sys/net/ipv4/icmp_echo_ignore_broadcasts

De seguida ligamos a interface *eth1* do tuxY4 ao switch(porta 4) e configurá-la com um IP dentro da mesma gama do tuxY2; adicionando, de seguida, esta interface à mesma bridge do tuxY2.
Após este passo, adicionou-se uma rota ao tuxY3 utilizando o comando <***route add –net 172.16.y1.0/24 gw 172.16.y0.254***>.

Repetiu-se o procedimento para o tuxY2, mas utilizando os seguintes endereços:
<***route add –net 172.16.y0.0/24 gw 172.16.y1.253***>

Depois de corridos estes comandos, o tuxY4 encontra-se a servir de router, possibilitando comunicação entre si mesmo e tuxY3 a tuxY2

Em cada um dos tuxs podemos encontrar uma route que indica o IP por onde as mensagens de uma das sub-redes podem ir para a outra sub-rede (E.g: mensagens de tuxY3(172.16.50.1) com destino a tuxY2(172.16.51.1) devem primeiro ir para tuxY4(172.16.30.254) para este a encaminhar para o seu destino.

Em cada um dos tuxs podemos encontrar uma forwarding table. Esta guarda informação acerca da destination, netmask (estes 2 definem a network id), gateway (indica por onde a rede pode ser acedida), interface (indica a interface responsável por atingir o gateway) e, por fim, a métrica (indica o custo associado à rota).

Novamente no início da comunicação podemos novamente observar pacotes ARP

```
Broadcast           ARP         42 Who has 172.16.50.1? Tell 172.16.50.254
HewlettP_c3:78:70   ARP         60 172.16.50.1 is at 00:21:5a:61:2d:72
```

No início da comunicação, quer o tuxY3, quer o tuxY2 não sabem qual o endereço MAC do respetivo da respetiva do tuxY4 que se encontra na sua sub-rede, portanto, também emitem um pacote ARP para descobrir o endereço MAC.

Na tentativa do tux3 dar ping ao tuxY2, é possível verificar a mensagem ARP em que o tuxY4 pede o endereço MAC relativo ao IP do tuxY2. O tuxY2 envia a informação para o tuxY3, sendo esta redirecionada para o tuxY4 através da tabela de routing. Visto o tuxY4 não possuir na sua tabela ARP o endereço MAC do tuxY2 este necessita de transmitir (broadcast) um pacote ARP para descobrir esse endereço MAC.

```
Kye_08:d5:b0        Broadcast           ARP         42 Who has 172.16.51.1? Tell 172.16.51.253
HewlettP_61:2f:d6   Kye_08:d5:b0        ARP         60 172.16.51.1 is at 00:21:5a:61:2f:d6
```

Na sequência da execução do comando ping no tuxY3, com as interfaces (172.16.50.254, 172.16.51.253, 172.16.51.1) como destino é possível verificar os diferentes pacotes ICMP associados a cada IP.

```
172.16.50.1         172.16.51.1         ICMP        98 Echo (ping) request
172.16.51.1         172.16.50.1         ICMP        98 Echo (ping) reply
172.16.50.1         172.16.51.1         ICMP        98 Echo (ping) request
172.16.51.1         172.16.50.1         ICMP        98 Echo (ping) reply
```

```
172.16.50.1         172.16.50.254
172.16.50.254       172.16.50.1
172.16.50.1         172.16.50.254
172.16.50.254       172.16.50.1
```

```
172.16.50.1         172.16.51.253
172.16.51.253       172.16.50.1
172.16.50.1         172.16.51.253
172.16.51.253       172.16.50.1
```

Os endereços MAC associados aos pacotes observados são os das respetivas

máquinas de destino e origem. No entanto, devido ao *forwarding* de pacotes pelo tuxY4, os pacotes podem ter endereço IP de destino correspondente ao tuxY4 (terminado em 50.254 ou 51.253).

No final desta experiência foi possível enviar um comando *ping* do tuxY3 para o tuxY2 com sucesso, usando o tuxY4 como router.

3.4 - Experiência 4 - Configurar um router comercial e implementar NAT

Nesta experiência pretendia-se fazer a configuração de um router comercial com NAT devidamente implementado, sendo necessária a configuração de duas interfaces deste mesmo router (ether1 ligada à bridge1 e ether2 ligada à rede exterior da sala).

Para esta experiência foi necessário configurar os endereços das portas ethernet do router, bem como adicionar uma route estática e uma default gateway.

Fizemos as seguintes ligações:

- ether1 -> lab network (PY.1)
- ether2 -> port 5(switch)

Os comandos usados foram os seguintes:

- /system reset-configuration

- /ip address add address=172.16.1.59/24 interface=ether1
- /ip address add address=172.16.51.254/24 interface=ether2
- /ip address print

- /ip route add address=0.0.0.0/0 gateway=172.16.2.254
- /ip route add dst-address=172.16.50.0/24 gateway=172.16.51.253

O último comando adiciona a route estática para que os pacotes exteriores consigam chegar

ao tuxY3

Os caminhos seguidos nesta experiência são similares aos da experiência anterior com a adição dos novos destinos configurados no router.

Nesta experiência desativamos a funcionalidade NAT do router, usando o comando:

- /ip firewall nat disable 0

O NAT (Network Address Translation) possibilita a comunicação entre os computadores da rede criada interna e redes externas. Como estamos a trabalhar numa rede privada os IP's definidos nunca seriam reconhecidos externamente e por isso é necessário reescrevê-los.
Portanto ao desativamos o NAT, perdemos a capacidade de comunicar com o exterior

3.5 - Experiência 5 - DNS

Nesta experiência pretendia-se configurar o servidor de DNS para permitir ligação à Internet através da procura de nomes de domínios. Domain Name System (DNS) é responsável por associar e traduzir esses nomes em endereços IP, para possibilitar comunicação através desses nomes, invés de decorar IPs.

Para configurar um serviço DNS é necessário aceder e editar o ficheiro resolv.conf. Este ficheiro é lido sempre que são invocadas rotinas que fornecem acesso à Internet. Neste caso adicionamos uma entrada ao ficheiro encontrado em etc/resolv.conf: *"nameserver 172.16.1.1"* que indica que um servidor DNS encontra-se no IP indicado.

Para testar funcionalidade DNS foi feito um ping ao domínio ftp.up.pt .Através da análise dos resultados do *Wireshark* podemos observar que no início da comunicação, antes do envio/receção de pacotes ICMP (usando o comando **ping ftp.up.pt**), foram enviados pacotes DNS para identificar o endereço IP do destino (neste caso ftp.up.pt). Foi pedido ao DNS que enviasse os atributos do domínio. Como resposta o servidor enviou informações o endereço IP do destino. Posteriormente, foi também feito um pedido *reverse* DNS (rDNS) que processa o pedido contrário, ou seja, obter o nome domínio a partir do endereço IP.

```
172.16.50.1     172.16.1.1      DNS      86 Standard query 0x899e PTR 233.90.160.34.in-addr.arpa
172.16.1.1      172.16.50.1     DNS     138 Standard query response 0x899e PTR 233.90.160.34.in-addr.arpa…
172.16.50.1     172.16.1.1      DNS      87 Standard query 0xfe56 PTR 201.181.244.35.in-addr.arpa
172.16.50.1     172.16.1.1      DNS      69 Standard query 0x9819 A ftp.up.pt
172.16.1.1      172.16.50.1     DNS     107 Standard query response 0x9819 A ftp.up.pt CNAME mirrors.up.p…
```

Figura 3.5.a: Pacotes DNS

3.6 - Experiência 6 - Ligações TCP

Nesta experiência executou-se a aplicação desenvolvida na primeira partedo trabalho, analisando as suas características de comunicação.

A aplicação abriu 2 ligações TCP, a primeira recebia e enviava os comandos para o servidor e a outra recebia os dados enviados pelo servidor.

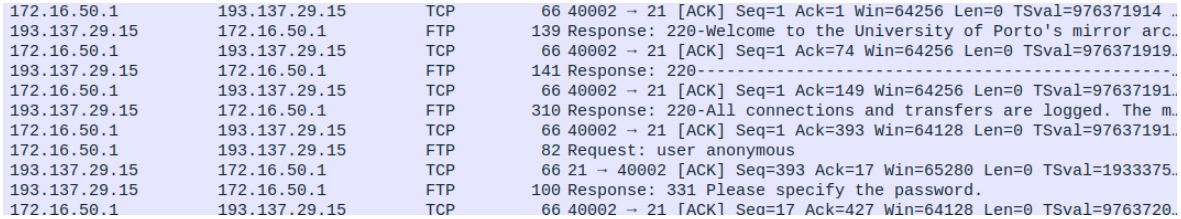

Figura 3.6.a: Pacotes TCP

Cada conexão TCP apresenta 3 fases: a conexão ao servidor, a troca de dados/informação e a terminação da conexão.

O protocolo TCP apresenta o mecanismo ARQ (Automatic Repeat Request), uma variação de Go-Back-N com a diferença que o recetor não deixa de processar os frames recebidos quando deteta um erro. O recetor continua a receber as frames seguintes enviando no ACK o número da frame que falhou até a conseguir receber. O emissor verifica os ACK e reenvia a frames perdidas.

Os campos mais importantes para este mecanismo estão contidos no header dos pacotes TCP, sendo eles:

- A flag ACK, que indica que os dados foram enviados corretamente;

- O campo checksum, que server para averiguar a integridade dos dados;

- Os campos de sequência e acknowledge que servem para garantir que os dados são entregues na ordem correta.

O protocolo TCP apresenta também um mecanismo de controlo de congestionamento, ou seja, ele limita ou aumenta a taxa de envio de dados em função do congestionamento da rede.

Durante o início de uma conexão TCP temos a fase de partida lenta que aumenta sua velocidade exponencialmente. Seguidamente sofre uma descida quando chega perto do seu máximo e acaba por estabilizar, mas ainda sofrendo algumas alterações na taxa de transferência.

Ao iniciar uma segunda conexão TCP o fluxo de dados é afetado. Apesar da média de

transferência de pacotes se manter ao iniciar uma segunda conexão verifica-se também um aumento de decréscimos, levando, assim, a que o download do ficheiro do servidor TCP demora-se mais tempo.

4 - Conclusão

No final, foi-nos possível implementar os objetivos propostos inicialmente no guião, e obter uma aplicação robusta que teve como objetivo o download de ficheiros sem erros. Consideramos que o trabalho realizado ao longo das aulas práticas com as experiências propostas foram de facto bastante úteis no que diz respeito à aprendizagem da matéria e revelaram-se assim uma boa fonte de conhecimento. Foi necessário dominar os conceitos de cliente-servidor, assim como o protocolo de comunicação TCP/IP, e também o protocolo de comunicação FTP (File Transfer Protocol) assim como o serviço DNS. No geral achamos que foi uma boa aprendizagem.

5 - Anexos

Todos os ficheiros usados na elaboração deste projeto foram entregues em conjunto com este documento num ficheiro zip.

Alternativamente também estaram disponíveis no seguinte link

https://github.com/ICWeiner/FEUP-RCOM