

Trabalho 2

Configuração de uma rede e desenvolvimento de
uma aplicação de download

Relatório Final



Mestrado Integrado em Engenharia Informática e
Computação

Redes de Computadores

Professor:
Manuel Ricardo

Turma 4:
Henrique Manuel Martins Ferrolho - ei12079
João Filipe Figueiredo Pereira - ei12023
José Pedro Vieira de Carvalho Pinto - ei12164
Miguel Ângelo Jesus Vidal Ribeiro - ei11144

Faculdade de Engenharia da Universidade do Porto
Rua Roberto Frias, sn, 4200-465 Porto, Portugal

24 de Dezembro de 2014

Resumo

Este relatório complementa o segundo projecto da Unidade Curricular de Redes de Computadores, do Mestrado Integrado em Engenharia Informática e de Computação. O projecto consiste na configuração de uma rede de computadores e no desenvolvimento de uma aplicação de download de um ficheiro. Este documento subdivide-se em diversas secções destacando-se duas delas:

- A secção da aplicação de download onde é descrita a sua arquitectura e apresentados os resultados da sua execução, assim como a sua análise;
- A secção de configuração da rede onde foi proposto ao grupo a realização de seis experiências tendo cada uma delas objectivos delineados e independentes.

As experiências acima referidas basearam-se na configuração de um **IP de rede**, de um **router em Linux**, de um **router comercial** e do **DNS** (*Domain Name System*), e na implementação de duas **LAN's** (*Local Area Network*) **virtuais no switch** e do **NAT** (*Network Address Translation*) e num teste com a aplicação de download desenvolvida para a verificação de um bom **funcionamento nas ligações TCP** (*Transmission Control Protocol*). Estes conceitos e funções de protocolos, sistemas e redes, referidos anteriormente, serão explicados mais à frente no relatório.

Conteúdo

1	Introdução	4
2	Parte 1 - Aplicação de download	5
2.1	Arquitetura	5
2.2	Resultados de download	7
3	Parte 2 - Configuração da rede e análise	7
3.1	Experiência 1 - Configurar um IP de rede	7
3.2	Experiência 2 - Implementar duas LAN's virtuais no switch . . .	8
3.3	Experiência 3 - Configurar um router em Linux	8
3.4	Experiência 4 - Configurar um router comercial e implementar o NAT	9
3.5	Experiência 5 - DNS	10
3.6	Experiência 6 - Ligações TCP	10
4	Conclusões	10
	Referências	12
A	Anexos	12
A.1	Código da aplicação	12
A.2	Comandos de configuração	12
A.3	Logs gravados	12

1 Introdução

O segundo projecto de Redes de Computadores desenvolveu-se ao longo de diversas aulas laboratoriais, sendo que a primeira aula serviu para uma maior interiorização acerca de protocolos de aplicação IETF (*Internet Engineering Task Force*). Esta comunidade tem como objectivo proporcionar soluções a problemas relacionados com ligações à *Internet* e para tal são recomendados os documentos RFC (*Request for Comments*) que descrevem padrões de protocolos da mesma. O protocolo usado no trabalho foi o FTP com auxílio de um servidor da faculdade, a exemplo *ftp.fe.up.pt*, *ftp.up.pt*, entre outros. Este trabalho visou o estudo de uma rede de computadores, da sua configuração e posterior ligação a uma aplicação desenvolvida pelo grupo. Para tal, além de seguir as recomendações e instruções fornecidas no guião, o grupo teve de fazer pesquisas acerca do funcionamento do protocolo em questão e respectiva ligação ao servidor em uso.

O projecto divide-se em duas grandes componentes: a configuração de uma rede e o desenvolvimento de uma aplicação de download.

O principal objectivo da configuração de rede é permitir a execução de uma aplicação, a partir de duas *VLANs* dentro de um *switch*. Numa das *VLAN* foi implementado o NAT, estando este activo, e na outra não, tendo esta última que conseguir ter ligação à *Internet* para a aplicação de download funcionar correctamente.

Quanto aos objectivos da aplicação de download era essencial o grupo entender o que é um cliente, um servidor e as suas especificidades em TCP/IP, saber como se caracterizam protocolos em aplicações no geral, como definir um *URL* e descrever o comportamento de um servidor FTP. Com estes objectivos concluídos, o grupo poderia avançar para o desenvolvimento da aplicação implementando um cliente FTP e uma ligação TCP a partir de *sockets*. Só então poderíamos concluir a importância do DNS na conversão de um *URL* para um IP, permitindo a sua localização num *host* com domínio determinado.

Este relatório divide-se em:

- **Introdução**, onde são descritos os objectivos do trabalho;
- **Parte 1 - Aplicação de Download**, onde é descrita a sua arquitectura, apresentados resultados e a sua análise e quais foram os documentos que o grupo utilizou em auxílio na sua implementação;
- **Parte 2 - Configuração da rede e análise**, onde é descrita a sua arquitectura, objectivos de cada experiência, comandos de configuração e análise dos *logs* gravados durante a sua realização;
- **Conclusões**, onde são redigidas as últimas análises e opinião final do grupo ao projecto;
- **Bibliografia**, onde são colocados todos os documentos/*sites* de consulta efectuados pelo grupo;
- **Anexos**, onde será colocado o código relativo à aplicação, comandos de configuração e *logs* gravados.

Antes de prosseguir é de referir que o grupo desenvolveu este projecto em ambiente LINUX, com a linguagem de programação C.

2 Parte 1 - Aplicação de download

Uma das componentes do segundo projecto de Redes de Computadores era o desenvolvimento de uma aplicação de download na linguagem de programação C. Para a sua implementação o grupo teve de estudar vários documentos, nomeadamente o RFC959 que aborda o protocolo de transferência de ficheiros (FTP) e o RFC1738 que informa sobre o uso de *URL's* e o seu devido tratamento.

De seguida iremos descrever resumidamente o plano de implementação do programa e quais as suas funcionalidades, assim como a apresentação de resultados e a sua análise.

2.1 Arquitetura

Para implementar a aplicação o grupo decidiu criar duas camadas: a de processamento do URL e a do cliente FTP. Em cada camada, existe uma estrutura que contém as propriedades necessárias às funções que estas desempenham. A aplicação aceita um *link* como argumento, que deve ser especificado através da linha de comandos. O *link* pode conter um *username* e *password*, ou então nenhum caso se pretenda usar o modo *anonymous*.

```
joao_pereira@JoaoPereira:~/git/feup-rcom/practical-work-2/bin$ ./ftpdnloader
WARNING: Wrong number of arguments.

Usage1 Normal: ./ftpdnloader ftp://[<user>:<password>@]<host>/<url-path>
Usage2 Anonymous: ./ftpdnloader ftp://<host>/<url-path>

joao_pereira@JoaoPereira:~/git/feup-rcom/practical-work-2/bin$
```

Figura 1: Application usage.

A estrutura URL é responsável pelo processamento do argumento especificado na linha de comandos. Esta estrutura contém diversas *strings* que são preenchidas com os diferentes dados presentes no link: *user*, *password*, *hostname*, *path* e *filename*. Após o processamento do URL, o atributo *ip* é preenchido. O atributo *port* é sempre 21 (número da porta de controlo do protocolo FTP).

```
typedef char url_content[256];

typedef struct URL {
    url_content user; // string to user
    url_content password; // string to password
    url_content host; // string to host
    url_content ip; // string to IP
    url_content path; // string to path
    url_content filename; // string to filename
    int port; // integer to port
} url;
```

Figura 2: URL struct.

As funções características desta camada são apresentadas de seguida.

```

void initURL(url* url);
int parseURL(url* url, const char* str); // Parse a string with the url to create the URL structure
int getIpByHost(url* url); // gets an IP by host name

char* processElementUntilChar(char* str, char chr);

```

Figura 3: URL functions.

Breve descrição das funções que constituem esta estrutura:

- **initURL**, instancia o objecto e aloca memória para os seus atributos;
- **deleteURL**, liberta a memória alocada anteriormente;
- **parseURL**, processa o *link* enviado como argumento ao programa e guarda a informação nos respectivos atributos de *url*;
- **getIpByHost**, obtém o IP a partir de um hostname passado como argumento. Este processo deve-se à função *gethostbyname* que retorna uma estrutura do tipo *hostent*, que é usada na função *inet_ntoa* através de um cast para uma estrutura do tipo *in_addr* e é devolvido um *char** no formato de números e pontos representando o IP.

A função **processElementUntilChar** processa uma sub-string até um determinado carácter passado como argumento.

No que diz respeito à estrutura do cliente FTP, apenas são necessários dois atributos: um descritor de ficheiro para o socket de controlo e outro para o socket de dados.

```

typedef struct FTP
{
    int control_socket_fd; // file descriptor to control socket
    int data_socket_fd; // file descriptor to data socket
} ftp;

```

Figura 4: FTP Client struct.

Após o processamento do *URL* estar concluído, é necessário ligar o cliente FTP através de um socket TCP ao servidor em questão, neste caso FTP. Para isso utiliza-se a função **ftpConnect**. Seguindo o protocolo FTP, e a primeira aula laboratorial, o grupo estabeleceu uma ordem de comandos a enviar que será analisada na apresentação de resultados. A ordem pela qual a comunicação foi feita foi:

- USER user**, o nome do utilizador é enviado;
- PASS pass**, onde o utilizador envia a password para o servidor;
- CWD path**, permite ao servidor alterar o diretório em que se encontra, indo para aquele onde se encontra o ficheiro;
- PASV**, entrada em modo passivo, permitindo uma mútua comunicação entre o servidor e o cliente FTP. É também feita nova conexão do socket mas, desta vez, a uma porta processada com informação recebida do servidor, sendo guardada no descritor de dados do cliente FTP;

-**RETR filename**, é pedido ao servidor o envio do ficheiro para download.

Ao fim de realizados estes passos, inicia-se a transferência do ficheiro especificado pelo utilizador. As funções responsáveis pela comunicação entre o cliente FTP e o servidor apresentam-se de seguida.

Um ponto importante a frisar na comunicação entre as duas camadas é que o cliente FTP recorre aos atributos do URL previamente formado para

```

int ftpConnect(ftp* ftp, const char* ip, int port);
int ftpLogin(ftp* ftp, const char* user, const char* password);
int ftpCWD(ftp* ftp, const char* path);
int ftpPasv(ftp* ftp);
int ftpRetr(ftp* ftp, const char* filename);
int ftpDownload(ftp* ftp, const char* filename);
int ftpDisconnect(ftp* ftp);

int ftpSend(ftp* ftp, const char* str, size_t size);
int ftpRead(ftp* ftp, char* str, size_t size);

```

Figura 5: FTP Client functions.

executar todas as acções de comunicação e posterior transferência do ficheiro, não existindo mais nenhuma relação entre estas.

2.2 Resultados de download

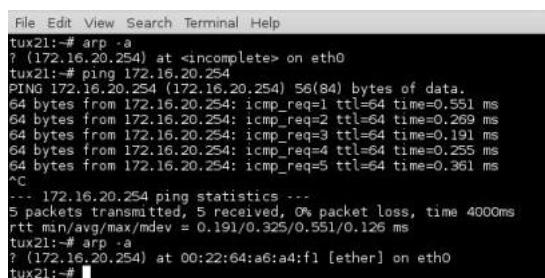
TODO

3 Parte 2 - Configuração da rede e análise

3.1 Experiência 1 - Configurar um IP de rede

A finalidade desta experiência foi a compreensão da configuração de IP's em máquinas diferentes, de modo a que estas consigam comunicar entre si. Assim, após a configuração dos IP's das portas eth0 de dois computadores e a adição das rotas necessárias à tabela de reencaminhamento, foi enviado o sinal “ping” de um computador para o outro, para verificar que, de facto, as máquinas tinham uma ligação entre si.

Para a configuração dos computadores foi utilizado o comando `<ifconfig [ip]>`, que atribui ao IP da interface o IP passado como argumento. Após esta configuração, executamos um ping de uma máquina para a outra com os IP's definidos, operação essa que foi executada com sucesso. Foi também possível ver os pedidos ARP, com os pings definidos e a resposta da máquina correspondente com o seu endereço MAC.



```

File Edit View Search Terminal Help
tux21:~# arp -a
? (172.16.20.254) at <incomplete> on eth0
tux21:~# ping 172.16.20.254
PING 172.16.20.254 (172.16.20.254) 56(84) bytes of data.
64 bytes from 172.16.20.254: icmp_req=1 ttl=64 time=0.551 ms
64 bytes from 172.16.20.254: icmp_req=2 ttl=64 time=0.269 ms
64 bytes from 172.16.20.254: icmp_req=3 ttl=64 time=0.191 ms
64 bytes from 172.16.20.254: icmp_req=4 ttl=64 time=0.255 ms
64 bytes from 172.16.20.254: icmp_req=5 ttl=64 time=0.361 ms
^C
--- 172.16.20.254 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4000ms
rtt min/avg/max/mdev = 0.191/0.325/0.551/0.126 ms
tux21:~# arp -a
? (172.16.20.254) at 00:22:64:a6:a4:f1 [ether] on eth0
tux21:~#

```

Figura 6: Exemplo do output de um *ping*.

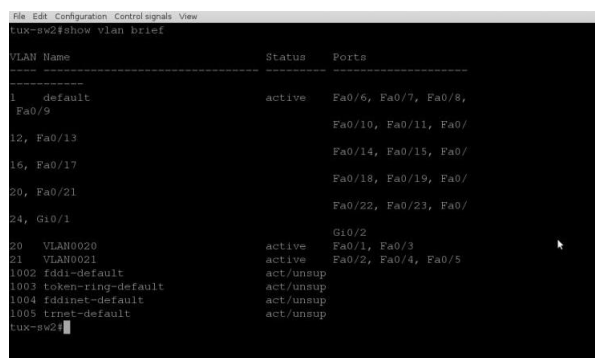
O ping, após obter o endereço MAC através dos pacotes ARP, gera pacotes do protocolo ICMP. Em frames do tipo Ethernet, os bits vinte e um e vinte e dois do frame identificam o protocolo para o qual deve ser enviado o payload.

A interface loopback é uma interface de rede virtual que o computador utiliza para comunicar com ele próprio, com o objectivo de realizar testes de diagnóstico, ou aceder a servidores na própria máquina, como se fosse um cliente. Assim, uma interface loopback, permite a existência de um endereço IP no router, que está sempre activo, em vez de ser dependente de uma interface física.

3.2 Experiência 2 - Implementar duas LAN's virtuais no switch

Nesta experiência foram criadas duas LANs virtuais no switch: a primeira constituída pelas máquinas 1 e 4, e a segunda pela máquina 2. Com esta configuração, a máquina 2 deixaria de ter acesso às máquinas 1 e 4, uma vez que se encontrariam em sub-redes diferentes.

Para a configuração do switch foi necessário entrar na sua consola de configuração e executar o comando `<vlan [n]>`, onde *n* é o número identificador da VLAN. Após a configuração das VLANs foi necessário adicionar as portas do switch às respectivas VLANs, para criar duas sub-redes individuais. Para isso usaram-se os comandos `<interface fastethernet 0/[i]>`, onde *i* é o identificador da porta do switch, seguido de `<switchport mode access>` e `<switchport access VLAN [n]>` onde *n* é o identificador da VLAN criada.



VLAN Name	Status	Ports
1 default	active	Fa0/6, Fa0/7, Fa0/8, Fa0/9, Fa0/10, Fa0/11, Fa0/12, Fa0/13
16, Fa0/17		Fa0/14, Fa0/15, Fa0/16, Fa0/17
20, Fa0/21		Fa0/18, Fa0/19, Fa0/20, Fa0/21
24, Gi0/1		Fa0/22, Fa0/23, Fa0/24, Gi0/2
20 VLAN0020	active	Fa0/1, Fa0/3
21 VLAN0021	active	Fa0/2, Fa0/4, Fa0/5
1002 fddi-default	act/unsup	
1003 token-ring-default	act/unsup	
1004 fddinet-default	act/unsup	
1005 trnet-default	act/unsup	

Figura 7: Output do briefing da VLAN.

Após estas configurações foi executado um ping para a máquina 2 que, tal como esperado, falhou, uma vez que se encontrava numa sub-rede diferente, inacessível tanto pela máquina 1, como pela máquina 4.

3.3 Experiência 3 - Configurar um router em Linux

O objectivo desta experiência era configurar a máquina 4 como router entre as duas sub-redes criadas na experiência dois.

Para realizar esta tarefa, foi necessário ligar a interface ethernet 1 da máquina 4 e configurá-la com um IP dentro da mesma gama que a máquina 2, e adicionar esta interface à sub-rede da máquina 2.

Após esta configuração, adicionou-se uma rota à máquina 1 utilizando o comando `<route add -net 172.16.y1.0/24 gw 172.16.y0.254>`. O primeiro endereço identifica a gama de endereços para a qual se quer adicionar a rota; o segundo endereço identifica o IP para o qual se deve reencaminhar o pacote (neste caso o IP da máquina 4). Posteriormente, repetiu-se o mesmo procedimento para a máquina 2, mas utilizando os seguintes endereços `<route add`

`-net 172.16.y0.0/24 gw 172.16.y1.253>`. Novamente, o IP 172.16.y1.253, é o IP da máquina 4 nesta sub-rede.

Finalmente, foi possível *pingar* a máquina 2 a partir da máquina 1. O pedido para o IP da máquina 2 - 172.16.y1.1 -, é reencaminhado para a máquina 4 - 172.16.y0.254; como a máquina 4 está ligada à sub-rede de ambas as máquinas, consegue aceder à máquina 2 - 172.16.y1.1 -, através da sua interface eth1, que está nessa sub-rede, e assim reencaminha o pacote para a máquina 2. Na resposta, o processo é idêntico, sendo o pacote reencaminhado da máquina 2, para a máquina 1.

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

Nesta experiência pretendia-se que fosse configurado um router comercial com NAT devidamente implementado. A implementação do NAT (Network Address Translation) teve como objectivo possibilitar a comunicação entre os computadores da rede criada com redes externas. Por se tratar de uma rede privada, os ip's nunca seriam reconhecidos fora da rede. Por isso, criou-se uma técnica que permite reescrever os IP's de origem de uma rede interna, para que possam aceder a uma rede externa. Este procedimento gera um número de 16 bits, utilizando esse valor numa hash table, e escrevendo-o no campo da porta de origem. Na resposta, o processo é revertido, e o router sabe para qual computador da rede interna deve enviar a resposta.

Para configurar o router, foi necessário configurar a interface interna no processo de NAT. Para isso, entrou-se na consola de configuração da interface fastethernet 0/0 do router, com o comando `<interface fastethernet 0/0>`. Em adição, teve de ser especificado qual o IP para essa interface, introduzindo o comando `<ip address [ip] [mask]>`, onde, neste caso, o *ip* correspondeu ao 172.16.21.254, e a *mask* a 255.255.255.0.

Posteriormente, foi configurada a interface externa, atribuindo um IP à interface 1, que estava ligada ao router da sala. Para isso, introduziram-se os seguintes comandos: `<interface fastethernet 0/1>`, `<ip address 172.16.1.29 255.255.255.0>`. Para ambos os casos foi necessário introduzir o comando `<no shutdown>`, para que estas configurações se mantivessem caso o router fosse desligado.

De seguida, para que fosse garantida a gama de endereços, introduziram-se os comandos: `<ip nat pool ovrld 172.16.1.29 172.16.1.29 prefix 24>` e `<ip nat inside source list 1 pool ovrld overload>`.

Posteriormente, foi criada uma lista de acessos e permissões de pacotes, para cada uma das sub-redes, com o comando `<accesslist 1 permit ip [máximo]>`. Neste caso, o IP utilizado foi 172.16.20.0 e 172.16.21.0, que poderia ir até 172.16.2X.255, colocando 0.0.0.255 no campo *máximo*.

Finalmente, foram definidas as rotas internas e externas, aplicando `<ip route 0.0.0.0 0.0.0.0 172.16.1.254>` e `<ip route 172.16.20.0 255.255.255.0 172.16.21.253>`, este comando cria uma rota, quando o IP de destino for 172.16.20.0-255 deve redireccionar os pacotes para o IP 172.16.21.253. Para testar, foi executado na máquina 1 um ping ao router da sala e verificou-se que os pacotes enviados para a máquina 1, passavam pela máquina 2, onde eram reencaminhados para o router no IP 172.16.21.254.

3.5 Experiência 5 - DNS

O objectivo desta experiência era conseguir aceder a redes externas, conseguindo desta forma aceder à Internet, através da rede interna criada. Para isto, foi necessário configurar o DNS.

Esta configuração passa por, em todos os hosts da rede criada, aceder e editar o ficheiro `resolv.conf`. Este ficheiro é lido cada vez que são invocadas rotinas que fornecem acesso à Internet. Neste caso, o ficheiro foi editado colocando **“nameserver 172.16.1.1”**, que se trata do endereço IP do servidor que deve ser acedido.



```
File Edit View Search Terminal Help
GNU nano 2.2.6 File: /etc/resolv.conf
#tux resolv.conf
domain netlab.fe.up.pt
search netlab.fe.up.pt fe.up.pt
nameserver 172.16.1.1
#nameserver 193.136.28.10
```

Para testar esta experiência, foi feito o teste de ping usando *www.google.com*. Nos logs, consequentemente, verificou-se que o DNS pergunta a informação contida num dado domain name, e este responde com o tempo de vida e o tamanho do pacote de dados.

Exemplo:

Query-> www.google.com: Type A, class IN.

Answer-> Name: www.google.com, Type: A, Class: IN, Time to live: 39. seconds, Data Length: 4, Addr: 173.194.41.206

3.6 Experiência 6 - Ligações TCP

Por fim, na experiência 6, compilou-se e executou-se a aplicação desenvolvida e descrita na primeira parte do relatório.

Para testar a aplicação, foi usado um servidor ftp e efectuado o download de um ficheiro. O download efectuou-se correctamente, o que demonstrou que a rede estava bem configurada, não trazendo qualquer problema no acesso por protocolo ftp, assim como à utilização de um servidor exterior à rede.

TCP utiliza *Selective Repeat ARQ*, que é semelhante ao *GO-BACK-N ARQ*, com a diferença que o receptor não deixa de processar os frames recebidos quando detecta um erro. Quando falha de um frame é detectada, o receptor continua um *acknowledgement* com o número da frame que falhou. O receptor continua a receber e a processar as frames seguintes, enviando sempre, no *ack*, o número da frame que falhou primeiro. No final do envio, o emissor verifica os *ack* e reenvia os frame perdidos.

4 Conclusões

Após a realização deste trabalho, concluímos com sucesso os principais objectivos do projecto: configurar uma rede, e elaborar uma aplicação de download que fosse capaz de executar utilizando a rede criada.

Desta forma, adquirimos competências e conhecimentos importantes de como configurar uma rede de comunicação. Além disso, ficámos com um conhecimento mais alargado sobre diferentes tipos de protocolos utilizados para a comunicação, e troca de informação numa rede de computadores.

Em suma, podemos afirmar que os objectivos a que nos propusemos na elaboração deste trabalho foram atingidos, e que, sem dúvida, a elaboração deste projecto ajudou a assimilar e fortalecer alguns conhecimentos de redes de computadores, e de protocolos da ligação utilizados.

A Anexos

A.1 Código da aplicação

Código da aplicação.

A.2 Comandos de configuração

Comandos de configuração.

A.3 Logs gravados

Logs gravados.

Figura 8: Experiência 1 - WireShark

