

Redes de Computadores

### RCOM – 3MIEIC06

2º Trabalho Laboratorial

João Álvaro Cardoso Soares Ferreira – [up201605592@fe.up.pt](mailto:up201605592@fe.up.pt)

[João Augusto dos Santos Lima](https://sigarra.up.pt/feup/pt/fest_geral.cursos_list?pv_num_unico=201605314) – [up201605314@fe.up.pt](mailto:up201605314@fe.up.pt)

Índice

[RCOM – 3MIEIC06 1](#_Toc532355680)

[Sumário 3](#_Toc532355681)

[Introdução 3](#_Toc532355682)

[Parte 1 – Desenvolvimento da aplicação para download 4](#_Toc532355683)

[Arquitetura da aplicação 4](#_Toc532355684)

[Resultados 5](#_Toc532355685)

[Parte 2 – Configuração de uma rede e experiências 7](#_Toc532355686)

[Experiência 1 - Configuração de uma IP Network 7](#_Toc532355687)

[Experiência 2 - Implementação de duas virtual LANs num switch 10](#_Toc532355688)

[Experiência 3 - Configuração de um router em Linux 12](#_Toc532355689)

[Experiência 4 - Configuração de um commercial router e implementação de NAT 15](#_Toc532355690)

[Experiência 5 - Configuração de DNS 16](#_Toc532355691)

[Experiência 6 - Configuração de ligações TCP 17](#_Toc532355692)

[Conclusão 19](#_Toc532355693)

[Anexos 20](#_Toc532355694)

[Main.c 20](#_Toc532355695)

[Auxiliar.c 22](#_Toc532355696)

[TCP.c 25](#_Toc532355697)

## Sumário

Este relatório pretende demonstrar e explicar o trabalho realizado durante o segundo trabalho laboratorial da disciplina de Redes de Computadores. Neste trabalho desenvolvemos uma aplicação com o propósito de descarregar um ficheiro de um servidor FTP e efetuamos várias experiências que envolvem a configuração de routers e switches. Ao longo do relatório, o trabalho realizado será exposto em pormenor.

Os objetivos propostos no guião de trabalho – tanto os práticos relativamente à realização do projeto, como os relativamente à nossa aprendizagem dos temas em questão, foram completos com sucesso.

## Introdução

O segundo projeto na unidade curricular de Redes de Computadores, tal como o guião do trabalho indica, foi dividido em duas partes com um objetivo final – efetuar o download de um ficheiro com o protocolo FTP.

A primeira parte é o desenvolvimento do código da aplicação responsável pelo download, enquanto que a segunda parte é a configuração da rede e realização das experiências indicadas no guião.

As experiências consistem no seguinte:

1. Configuração de uma IP Network
2. Implementação de duas virtual LANs num switch
3. Configuração de um router em Linux
4. Configuração de um commercial router e implementação de NAT
5. Configuração de DNS
6. Configuração de ligações TCP

## Parte 1 – Desenvolvimento da aplicação para download

A aplicação desenvolvida tem o propósito de descarregar um ficheiro, utilizando protocolo FTP como descrito em RFC959 e sintaxe de URL como descrita em RFC1738. No desenvolvimento desta servimo-nos também de código fornecido pelos professores como base.

### Arquitetura da aplicação

Tendo em conta que nos foi indicado o uso de dois protocolos específicos, dividimos a aplicação em duas camadas, sendo que cada uma é responsável pelo seu respetivo protocolo independentemente. Uma das camadas lida com o URL e a restante informação necessária para o download, enquanto que a outra lida com a ligação ao servidor e download do ficheiro.

É notado ao utilizador que o programa espera um input com o seguinte formato: “./run ftp://[<user>:<password>@]<host>/<url-path> “ de modo a poder mais facilmente extrair a informação necessária.

O programa começa por correr a função decomposeURL,que se encontra num ficheiro auxiliar.c, que se serve do input inicial do utilizador (estando este no formato indicado) para obter toda a informação necessária para o acesso ao site e ao ficheiro, guardando-a numa stuct Info:

Uma imagem com interior

Descrição gerada automaticamente

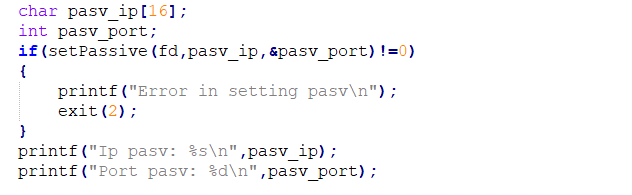
Sendo guardada essa informação, esta é impressa no ecrã de modo ao utilizador a poder verificar. Após isto, o programa estabelece uma ligação com a função makeConnection e de seguida tentará fazer o log in com a informação recebida. Esta função makeConnection abre uma TCP socket e estabelece uma ligação com o servidor fazendo uso desta.

Uma imagem com interior

Descrição gerada automaticamente

Como se pode verificar neste caso, se eventualmente o LogIn não for bem sucedido o programa imprime uma mensagem de erro e faz exit com código 2. Isto ocorre para todas as situações de erro no programa.

Estando efetuado o Log In do utilizador, o servidor FTP é colocado em modo passivo. Para este efeito é chamada a função setPassive que envia para o servidor o comando “pasv /r /n”. Esta função passa também para duas variáveis locais, pasv\_ip e pasv\_port, o IP e o port de onde o ficheiro será descarregado.



Para tal, estas mesmas variáveis são usadas pela função makeConnection novamente, desta vez obtendo o path com a função getPath e, de seguida, ir descarregando a este o ficheiro com a função getFile.

Uma imagem com captura de ecrã

Descrição gerada automaticamente

Da mesma forma que setPassive, getPath envia um comando ao servidor – desta vez o comando “retr /n” para este responder com a localização do ficheiro (ou, se este não existir, terminar o programa).

Por fim, as sockets abertas são fechadas e o programa é terminado.

### Resultados

Este programa foi testado na Experiência 6 da segunda parte deste trabalho, sendo que os resultados estavam de acordo com o esperado – o ficheiro é descarregado do servidor FTP desde que a formatação esteja correta e nenhum dos dados esteja errado. Uma melhor demonstração dos resultados é feita na secção dedicada a essa experiência, na segunda parte do relatório.

No entanto, apresentamos na página seguinte os logs de vários downloads bem sucedidos:

Uma imagem com captura de ecrã

Descrição gerada automaticamenteDownload no computador Tux31:

Download no computador Uma imagem com captura de ecrã, interior

Descrição gerada automaticamenteTux32:

Download no computador Tux31 com interferência devido ao processo ocorrer em simultâneo em Tux32:

Uma imagem com captura de ecrã, computador, interior, monitor

Descrição gerada automaticamente

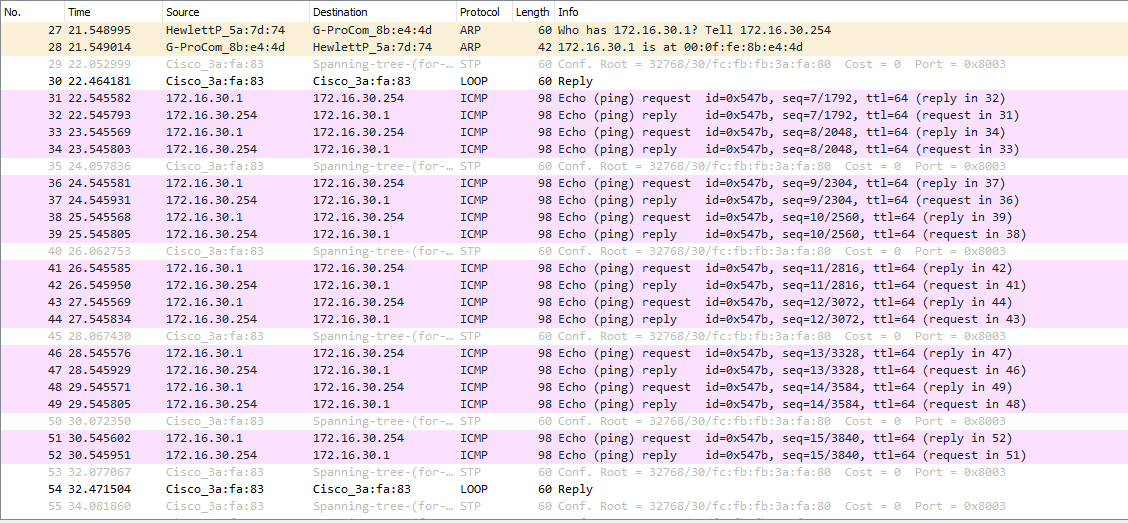
## Parte 2 – Configuração de uma rede e experiências

Antes da realização de qualquer experiência, pusemos em prática as indicações fornecidas no fim do guião para limpar todas as configurações que possivelmente tivessem sido feitas por outros grupos antes de nós. Retiramos também todos os cabos da bancada. Para o efeito deste relatório, todas as experiências foram realizadas na bancada 3 da sala.

### Experiência 1 - Configuração de uma IP Network

O objetivo desta experiência é estabelecer a ligação entre dois computadores (também referidos como tux) numa mesma rede. Os computadores utilizados para isto são o tux31 e o tux34. Inicialmente, atribuímos os IPs 172.16.30.1 à porta eth0 do tux31 e 172.16.30.254 à porta eth0 do tux34. Ligamos também os cabos respetivos ao switch (à porta 1 e 2 do switch, respetivamente) e os cabos necessários para o funcionamento do switch.

A atribuição destes IPs foi feita com o comando ifconfig. O resultado que obtivemos ao fazer ping do tux31 para o tux34 foi o seguinte:



Uma imagem com captura de ecrã

Descrição gerada automaticamenteO endereço IP e o MAC atribuídos a eth0 no tux31 da bancada 3 podem ser vistos aqui:

#### Respostas das questões:

* What are the ARP packets and what are they used for?

Pacotes ARP (Address Resolution Protocol) são pacotes de um protocolo de linha que envolve requisição e resposta. A requisição é realizada por via de Broadcast, solicita o endereço MAC através do endereço IP (como podemos ver nos logs do wireshark) e a reply é fornecida de forma direta.

* What are the MAC and IP addresses of ARP packets and why?

Os endereço IP de pacotes ARP são, como descrito na resposta à questão anterior, os endereços utilizados para requisitar os endereços MAC (endereços físicos) da máquina com o respetivo endereço IP.

* What packets does the ping command generate?

O commando ping gera packets do tipo ICMP.

* What are the MAC and IP addresses of the ping packets?

Os pacotes ping obtêm o endereço MAC através dos pacotes ARP, enviando para o respetivo endereço IP uma mensagem de controlo – estes endereços são, portanto, os do destinatário e da fonte.

* How to determine if a receiving Ethernet frame is ARP, IP, ICMP?

Em frames do tipo Ethernet, os bits 21 e 22 do frame indicam o protocolo utilizado.

* How to determine the length of a receiving frame?

A Physical Coding Sublayer é responsável por delimitar o tamanho de frames e enviá-las para a MAC layer.

* What is the loopback interface and why is it important?

A interface de loopback é responsável por verificar se a ligação e o sistema se mantêm ativos, tendo portanto grande importância, e faz-lo enviando pacotes LOOP de 10 em 10 segundos.

### Experiência 2 - Implementação de duas virtual LANs num switch

Na segunda experiência o objetivo é a criação de duas VLANs num switch, sendo a primeira VLAN constituída pelos tux31 e tux34 e uma segunda VLAN constituída apenas pelo tux32. Isto implica que enquanto o tux31 e o tux34 poderão comunicar livremente, o tux32 estará isolado destes devido a estar numa diferente sub-rede e não entrará em contacto com eles.

Começamos por fazer a configuração do switch, utilizando a respetiva consola. Esta configuração foi feita com o comando “vlan x” (sendo x o número da vlan que queremos configurar, 30 ou 31 neste caso), e adicionar as portas do switch a cada uma das VLANs – as portas 1 e 2 na primeira VLAN (ligando, respetivamente, ao tux31 e 34) e a porta 3 para a segunda VLAN, que liga ao tux32. Para isto usamos os comandos indicados para o efeito no guião: “interface fastethernet”, “switchport mode access” e “switchport access vlan”, com as alterações apropriadas para cada uma das configurações.

A Vlan 30 tem o IP 172.16.30.0 e a Vlan 31 tem o IP de 172.16.31.0, como indicado no guião.

Dando ping ao tux32 a partir do tux31:

Uma imagem com interior, parede

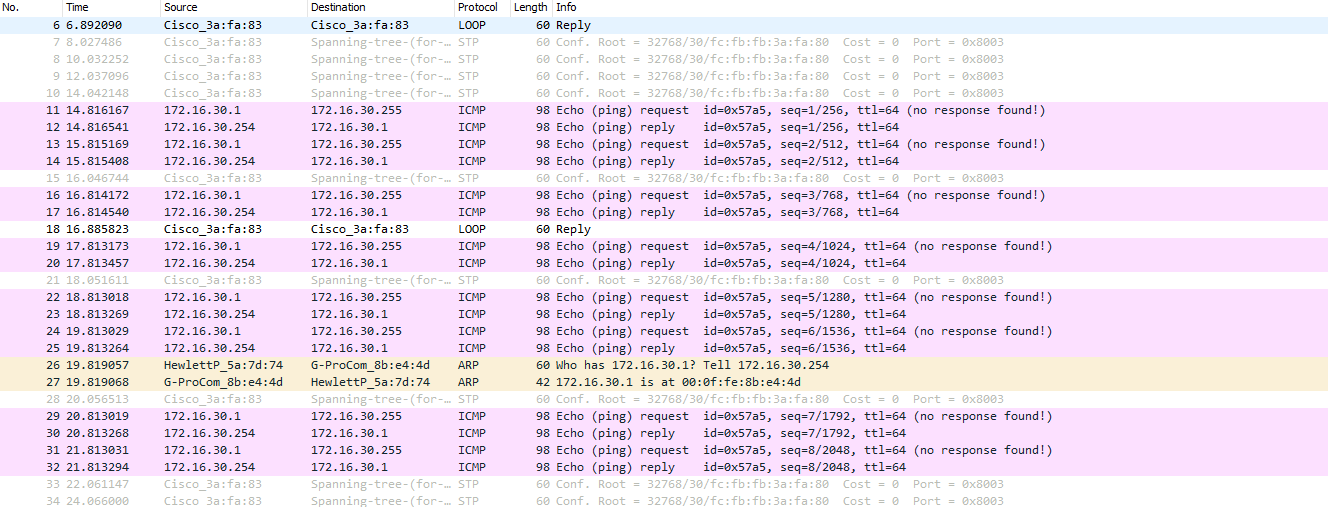
Descrição gerada automaticamente

Dando ping ao tux31 a partir do tux32:

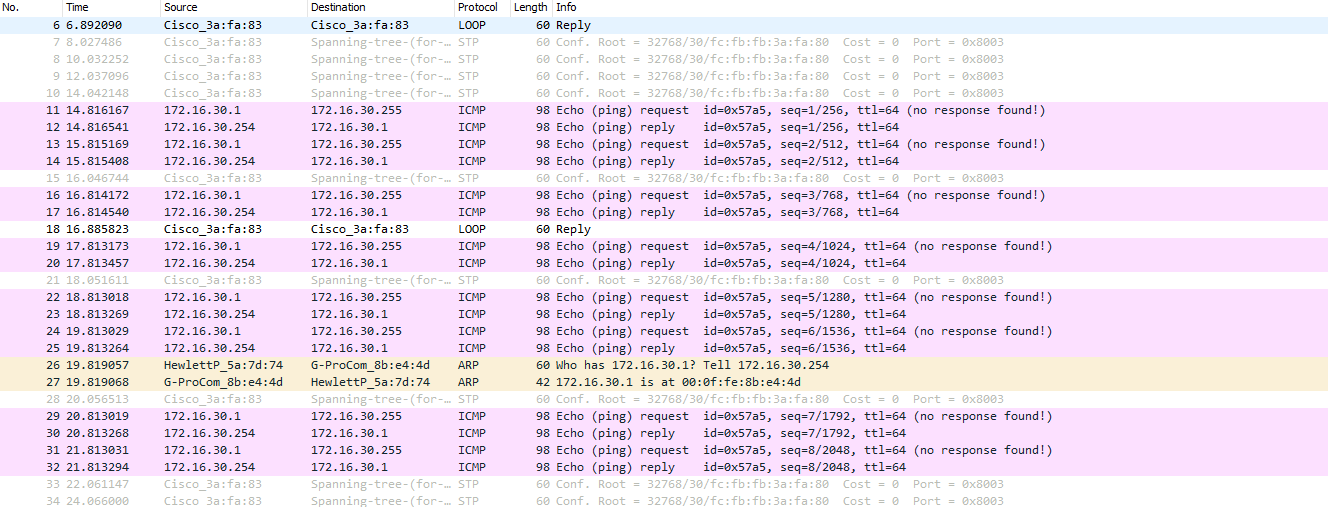
Uma imagem com interior, parede

Descrição gerada automaticamente

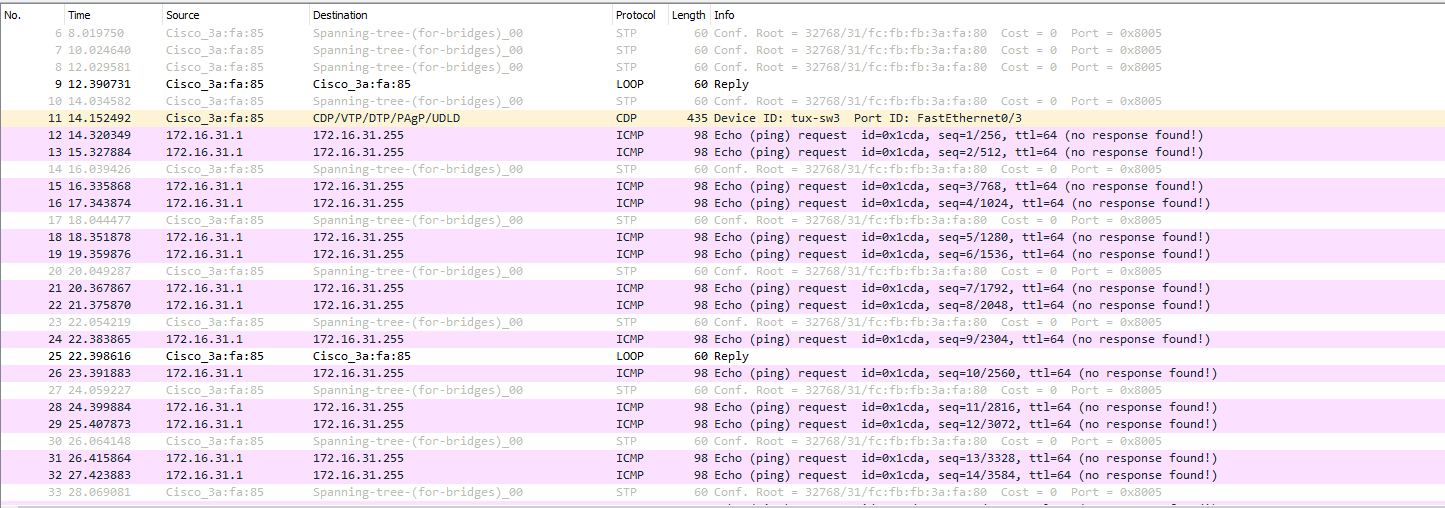
Dando ping ao tux34 a partir do tux31:



Fazendo ping broadcast (ping –b 172.16.y0.255) a partir do tux31:



Fazendo ping broadcast (ping –b 172.16.y0.255) a partir do tux32:



#### Respostas das questões:

* How to configure vlany0?

1. Enable
2. Configure terminal
3. vlan y0 (substituir y pelo numero apropriado)
4. exit
5. interface fastethernet 0/1 (substituir 1 pela entrada do switch utilizada se for diferente)
6. switchport mode access
7. switchport access vlan y0
8. interface fastethernet 0/2
9. switchport mode access
10. switchport access vlan y0
11. end

* How many broadcast domains are there? How can you conclude it from the logs?

Através dos logs do wireshark podemos verificar que existem duas sub-redes diferentes, a que liga tux31 e tux34 e a rede onde tux32 se encontra sozinho. Isto é visível através das respostas a ping broadcast em tux31 e tux32.

### Experiência 3 - Configuração de um router em Linux

Esta experiência consiste na configuração de um router em Linux, configurando o tux34 como router de modo a ligar as duas vlans criadas na experiência anterior.

Para tal, ativamos a porta eth1 do tux34, sendo esta configurada (como feito na experiência 1 para a porta eth0) com o IP 172.16.31.253. De seguida, utilizando o comando route add adicionamos em tux31 e tux32 as rotas necessárias a poder comunicar através de tux34. Nomeadamente:

Para o tux 31:

* route add -net 172.16.31.0/24 gw 172.16.30.254

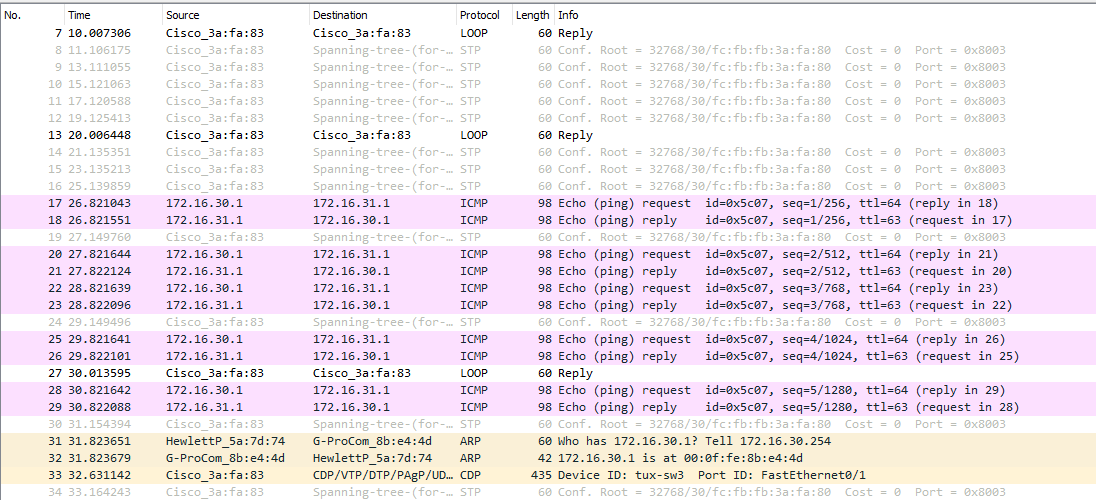
Para o tux 32:

* route add -net 172.16.30.0/24 gw 172.16.31.253

O primeiro endereço identifica a gama de endereços para a qual pretendemos adicionar a rota, enquanto que o segundo identifica o IP para onde o pacote será reencaminhado.

Feito isto, foi possível comunicar entre tux31 e tux32 através do router, como vemos nos seguintes logs (pings da parte de tux31):

Uma imagem com captura de ecrã

Descrição gerada automaticamente

Exemplo de routing table (tux1):

Uma imagem com captura de ecrã, texto

Descrição gerada automaticamente

#### Respostas das questões:

* What routes are there in the tuxes?

As rotas presentes nos tuxes são as que os endereços a que estes conseguem comunicar diretamente.

* What are their meaning?

São os endereços IPs dos restantes tuxes, da vlan ou do router establecido no tux4.

* What information does an entry of the forwarding table contain?

Uma routing table (ou forwarding table) é uma tabela de dados num router ou computador ligado em rede que lista as rotas para destinos em rede particulares.

* What ARP messages, and associated MAC addresses, are observed and why?
* What ICMP packets are observed and why?
* What are the IP and MAC addresses associated to ICMP packets and why?

### Experiência 4 - Configuração de um commercial router e implementação de NAT

Na quarta experiência, o objetivo é a configuração de um router comercial e implementação de NAT neste. Este é um passo crucial para possibilitar a comunicação com redes externas através dos tuxes. Os endereços dos tuxes não seriam reconhecidos por redes externas devido a não serem endereços públicos. A implementação do NAT permite a tradução dos endereços privados dos tuxes para um endereço público.

Alteramos o cabo de switch para router e aplicamos os seguintes comandos no tux34:

Uma imagem com captura de ecrã

Descrição gerada automaticamente

Após isto, verificamos através de pings que todos os tuxes eram capazes de interagir uns com os outros e com o router.

#### Respostas das questões:

* How to configure a static route in a commercial router?

configure terminal

ip route prefix mask {ip-address | interface-type interface-number [ip-address]} What are the paths followed by the packets in the experiments carried out and why?

* How to configure NAT in a commercial router?

Explicado anteriormente.

* What does NAT do?

Permite a redes publicas interagir com IPs privados (traduzidos para um router comercial).

### Experiência 5 - Configuração de DNS

Nesta experiência o objetivo foi conseguir aceder à internet após configurar um DNS em todos os tuxes. Isto foi feito alterando o ficheiro resolv.conf (presente na pasta /etc/ do sistema) para algo semelhante a:

search netlab.fe.up.pt

nameserver 172.16.1.1

Tendo feito estas alterações, obtivemos acesso à internet e pudemos efetuar os testes pedidos no guião, como demonstramos nos logs seguintes:

* Uma imagem com captura de ecrã

  Descrição gerada automaticamenteTux31:
* Tux32:

#### Uma imagem com captura de ecrã Descrição gerada automaticamenteRespostas das questões:

* How to configure the DNS service at an host?

Como demonstrado na experiência.

* What packets are exchanged by DNS and what information is transported

Queries sobre os destinos

### Experiência 6 - Configuração de ligações TCP

A última experiência consiste na execução da aplicação descrita na primeira parte do trabalho de modo a descarregar um ficheiro através de um servidor FTP. O servidor utilizado foi <http://ftp.up.pt/> e o download de vários ficheiros deste servidor foi efetuado com sucesso, demonstrando portanto que tanto as experiências como o desenvolvimento da aplicação foram bem sucedidas.

Para o comprovar, apesar de já o termos feito na secção dos resultados da parte 1, apresentamos de seguida logs dos downloads.

* Uma imagem com captura de ecrã

  Descrição gerada automaticamenteTux31:
* Uma imagem com captura de ecrã, interior

  Descrição gerada automaticamenteTux32:
* Tux31 com interferência devido ao processo ocorrer em simultâneo em Tux32:

Uma imagem com captura de ecrã, computador, interior, monitor

Descrição gerada automaticamente

#### Respostas das questões:

* How many TCP connections are opened by your ftp application?
* In what connection is transported the FTP control information?
* What are the phases of a TCP connection?
* How does the ARQ TCP mechanism work? What are the relevant TCP fields? What relevant information can be observed in the logs?
* How does the TCP congestion control mechanism work? What are the relevant fields. How did the throughput of the data connection evolve along the time? Is it according the TCP congestion control mechanism?
* Is the throughput of a TCP data connections disturbed by the appearance of a second TCP connection? How?

## Conclusão

O desenvolvimento deste trabalho foi extremamente instrutivo relativamente a como uma rede local funciona e interage com outras redes, desde a informação essencial como o funcionamento de um router à complexidade de protocolos como o FTP.

Um dos factos mais cruciais para a interiorização dos conceitos trabalhados foi a repetição. Devido a todas as experiências requererem que as anteriores estivessem completas e funcionais, o nosso grupo acabou por repetir todas as experiências múltiplas vezes (as últimas devido a erros que ocorreram nas suas primeiras iterações que se tiveram de resolver ao longo de vários repetições, de modo a poder registar os logs corretamente). As experiências iniciais foram, portanto, as mais repetidas e consolidadas.

Tendo concluído o segundo projeto que foi proposto ao nosso grupo nesta unidade curricular, podemos afirmar que os objetivos

## Anexos

### Main.c

#include "includes.h"

#include "auxiliar.h"

#include "tcp.h"

int main(int argc, char \*\*argv)

{

if(argc != 2)

{

printf("Numero incorreto de parametro. Formato: ./run ftp://[<user>:<password>@]<host>/<url-path>");

exit(1);

}

Info information;

decomposeURL(argv[1],&information);

printf("User: %s\n",information.user);

printf("Pass: %s\n",information.pass);

printf("Host: %s\n",information.host);

printf("Url-Path: %s\n",information.url);

printf("Filename: %s\n",information.filename);

printf("Port: %d\n",information.port);

printf("Host-name: %s\n",information.host\_name);

printf("IP: %s\n",information.ip);

int fd = makeConnection(information.ip,information.port);

if(makeLogIn(fd,information.user,information.pass)!=0)

{

printf("Error making log in.\n");

exit(2);

}

char pasv\_ip[16];

int pasv\_port;

if(setPassive(fd,pasv\_ip,&pasv\_port)!=0)

{

printf("Error in setting pasv\n");

exit(2);

}

printf("Ip pasv: %s\n",pasv\_ip);

printf("Port pasv: %d\n",pasv\_port);

int pasv\_fd = makeConnection(pasv\_ip,pasv\_port);

if(getPath(fd,information.url)!=0)

{

printf("Error setting the path.\n");

exit(2);

}

if(getFile(pasv\_fd,information.filename)!=0)

{

printf("Error downloading the file\n");

exit(2);

}

close(fd);

close(pasv\_fd);

return 0;

}

### Auxiliar.c

#include "auxiliar.h"

#define DEFAULT\_PORT 21

int decomposeURL(char\* url,Info\* information)

{

char \*text;

text = strtok(url,"[");

if(strcmp(text,"ftp://") !=0)

{

printf("Paramatro dados incorreto( ftp:// ). Formato: ./run ftp://[<user>:<password>@]<host>/<url-path>\n");

exit(1);

}

text = strtok(NULL,":");

strcpy(information->user,text);

text = strtok(NULL,"]");

if(text[strlen(text)-1] != '@')

{

printf("Paramatro dados incorreto( @] ). Formato: ./run ftp://[<user>:<password>@]<host>/<url-path>\n");

exit(1);

}

text[strlen(text)-1] = '\0';

strcpy(information->pass,text);

text = strtok(NULL,"/");

strcpy(information->host,text);

text = strtok(NULL," ");

strcpy(information->url,text);

text = strtok(NULL," ");

if(text != NULL)

{

printf("Paramatro dados incorreto( sem fim ). Formato: ./run ftp://[<user>:<password>@]<host>/<url-path>\n");

exit(1);

}

information->port=DEFAULT\_PORT;

char \*copy=malloc(strlen(information->url));

char last[255];

strcpy(copy,information->url);

text = strtok(copy,"/");

while(text != NULL)

{

strcpy(last,text);

text=strtok(NULL,"/");

}

strcpy(information->filename,last);

free(copy);

struct hostent \*h;

/\*

struct hostent {

char \*h\_name; Official name of the host.

char \*\*h\_aliases; A NULL-terminated array of alternate names for the host.

int h\_addrtype; The type of address being returned; usually AF\_INET.

int h\_length; The length of the address in bytes.

char \*\*h\_addr\_list; A zero-terminated array of network addresses for the host.

Host addresses are in Network Byte Order.

};

#define h\_addr h\_addr\_list[0] The first address in h\_addr\_list.

\*/

if ((h = gethostbyname(information->host)) == NULL)

{

herror("gethostbyname");

exit(1);

}

strcpy(information->host\_name,h->h\_name);

strcpy(information->ip,inet\_ntoa(\*((struct in\_addr \*)h->h\_addr)));

return 0;

}

### TCP.c

#include "tcp.h"

int makeConnection(char\* ip,int port)

{

int sockfd;

struct sockaddr\_in server\_addr;

/\*server address handling\*/

bzero((char \*)&server\_addr, sizeof(server\_addr));

server\_addr.sin\_family = AF\_INET;

server\_addr.sin\_addr.s\_addr = inet\_addr(ip); /\*32 bit Internet address network byte ordered\*/

server\_addr.sin\_port = htons(port); /\*server TCP port must be network byte ordered \*/

/\*open an TCP socket\*/

if ((sockfd = socket(AF\_INET, SOCK\_STREAM, 0)) < 0)

{

perror("socket()");

exit(0);

}

/\*connect to the server\*/

if (connect(sockfd,(struct sockaddr \*)&server\_addr,sizeof(server\_addr)) < 0)

{

perror("connect()");

exit(0);

}

printf("Connection to server made!\n");

if(port==21)

{

sleep(1);

char buf[1000];

int len = read(sockfd, &buf, 1000);

buf[len] = '\0';

printf("%s",buf);

}

return sockfd;

}

int writeMessage(int fd,char\* message)

{

int res = write(fd,message,strlen(message));

if(res != strlen(message))

return 1;

return 0;

}

int readMessage(int fd,char\* message)

{

sleep(1);

int res = read(fd,message,255);

message[res]='\0';

return res;

}

int makeLogIn(int fd,char\* user,char\* pass)

{

char\* mandar=malloc((strlen(user)+6)\*sizeof(char));

sprintf(mandar,"user %s\n",user);

if(writeMessage(fd,mandar))

return 1;

char response[255];

readMessage(fd,response);

if(response[0]!='3' || response[1]!='3' || response[2]!='1')

return 1;

printf("%s",response);

free(mandar);

mandar=malloc((strlen(pass)+6)\*sizeof(char));

sprintf(mandar,"pass %s\n",pass);

if(writeMessage(fd,mandar))

return 1;

readMessage(fd,response);

if(response[0]!='2' || response[1]!='3' || response[2]!='0')

return 1;

printf("%s",response);

free(mandar);

return 0;

}

int setPassive(int fd, char\*ip , int\* port)

{

char response[255];

if(writeMessage(fd,"pasv\n"))

return 1;

readMessage(fd,response);

if(response[0]!='2' || response[1]!='2' || response[2]!='7')

return 1;

printf("%s",response);

int bytes[6];

sscanf(response,"227 Entering Passive Mode (%d,%d,%d,%d,%d,%d).",&bytes[0],&bytes[1],&bytes[2],&bytes[3],&bytes[4],&bytes[5]);

sprintf(ip,"%d.%d.%d.%d",bytes[0],bytes[1],bytes[2],bytes[3]);

\*port=bytes[4]\*256 +bytes[5];

return 0;

}

int getPath(int fd,char\* url)

{

char response[255];

char\* mandar=malloc((strlen(url)+6)\*sizeof(char));

sprintf(mandar,"retr %s\n",url);

if(writeMessage(fd,mandar))

return 1;

readMessage(fd,response);

if(response[0]!='1' || response[1]!='5' || response[2]!='0')

{

printf("%s",response);

return 1;

}

printf("%s",response);

free(mandar);

return 0;

}

int getFile(int fd,char\* filename)

{

sleep(1);

char\* path=malloc(strlen(filename)+6);

sprintf(path,"files/%s",filename);

int fileFd = open(path,O\_WRONLY | O\_TRUNC | O\_CREAT ,0600);

if(fileFd <0)

return 1;

char ler;

int res;

while((res = read(fd,&ler,1)))

{

if(res < 1)

return 1;

if(write(fileFd,&ler,1)!=1)

return 1;

}

close(fileFd);

return 0;

}