

# Trabalho 2

Configuração e estudo de uma rede

## Relatório Final



Mestrado Integrado em Engenharia Informática e  
Computação

Redes de Computadores

### **Turma 6:**

António Manuel Vieira Ramadas | nº 201303568  
Rui Filipe Freixo Cardoso Osório | nº 201303843  
Rui Miguel Teixeira Vilares | nº 201207046

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

15 de Dezembro de 2015

## Resumo

Este projeto foi desenvolvido no âmbito da unidade curricular Redes de Computadores do 3º ano do curso Mestrado Integrado em Engenharia Informática e de Computadores da Faculdade de Engenharia da Universidade do Porto. Com a realização deste projeto foi possível perceber melhor o funcionamento de uma rede de computadores e a complexidade das estruturas que a formam. Foi também importante perceber a abstração que é oferecida ao utilizador e que lhe permite utilizar a rede tão facilmente para diversas tarefas, desde acesso à internet até à comunicação entre computadores. Assim, neste relatório é explicado o desenvolvimento de uma aplicação FTP (*File Transfer Protocol*) e a configuração e estudo de uma rede.

Em suma, o trabalho consistiu em desenvolver uma aplicação de *download* nas ligações TCP (*Transmission Control Protocol*) através do FTP, na configuração de uma rede IP (*Internet Protocol*), no uso do DNS (*Domain Name System*), de um router em Linux e de um router comercial com implementação NAT (*Network Address Translation*). Finalmente, também foi abordada a implementação de duas LANs (*Local Area Network*) virtuais num *switch* e o funcionamento das ligações TCP.

# Índice

Introdução.....	4
Parte 1 – Aplicação de <i>Download</i> .....	5
Arquitetura da aplicação de <i>download</i> .....	5
Relatório de um <i>download</i> com sucesso .....	6
Parte 2 – Análise e configuração da rede.....	7
Experiência 1 – Configuração de um IP de rede .....	7
Experiência 2 – Implementação de duas <i>virtual</i> LANs num <i>switch</i> .....	7
Experiência 3 – Configuração de um <i>router</i> em Linux .....	8
Experiência 4 – Configuração de um <i>router</i> comercial e implementação de NAT .....	8
Experiência 5 – DNS .....	9
Experiência 6 – Ligações TCP.....	9
Experiência 7 – Implementação de NAT em Linux.....	10
Conclusões .....	11
Referências.....	12
Anexos.....	13
Anexo A – Código da Aplicação .....	13
Anexo B .....	17
Anexo B1 .....	17
Anexo B2 .....	17
Anexo B3 .....	18
Anexo C .....	19
Experiência 1 – Configuração de um IP de rede.....	19
Experiência 2 – Implementação de duas <i>virtual</i> LANs num <i>switch</i> .....	20
Experiência 3 – Configuração de um <i>router</i> em Linux .....	20
Experiência 4 – Configuração de um <i>router</i> comercial e implementação de NAT .....	20
Experiência 5 – DNS.....	21
Experiência 7 – Implementação de NAT em Linux.....	21
Anexo D - <i>Scripts</i> .....	21
tuxPermission.sh .....	21
tux51.sh.....	21
tux52.sh.....	22
tux54.sh.....	22

## Introdução

Este segundo trabalho, desenvolvido ao longo das aulas práticas da unidade curricular de Redes de Computadores, é facilmente dividido em duas principais secções. Importa também referir que todo ele foi realizado em ambiente Linux, com material da “Cisco” e com recurso à linguagem de programação C.

Numa primeira parte, é abordado o desenvolvimento da aplicação de *download* através do protocolo FTP, descrito no RFC959 (*Request for Comments*) e a sintaxe URL descrita no RFC1738. Este protocolo foi estabelecido para haver uma comunicação universal única entre servidores e clientes para transferência de ficheiros. O código produzido e aqui apresentado, funciona corretamente desde que o URL passado esteja com a sintaxe e a informação correta. De realçar também a fácil implementação no lado do cliente que visa simplificar o trabalho do utilizador.

A segunda parte aborda a configuração e estudo de uma rede em diversos passos até, finalmente ser possível utilizar com sucesso a aplicação de *download* criada e descrita anteriormente. O resultado final é uma rede de vários computadores ligados entre si através de duas LANs virtuais num *switch* com um *router* em Linux, um router comercial e NAT. Será também utilizado o DNS para a conversão do URL para um IP e consequentemente, para o estabelecimento de uma comunicação com o *host* pretendido.

## Parte 1 – Aplicação de *Download*

Para desenvolver a aplicação de *download* através do protocolo de FTP foi, em primeira instância, feita uma pesquisa abrangente. O ambiente de desenvolvimento utilizado é o Linux e a linguagem de programação é o C, tal como é habitual em todas as redes de computadores. Após alguma pesquisa foi encontrado o RFC959, que define o modo como os clientes se devem ligar ao servidor para transferir ficheiros. Foi também essencial entender o RFC1738 para definir a sintaxe URL.

### Arquitetura da aplicação de *download*

A aplicação começa por verificar o número de argumentos, de modo a perceber se esta foi corretamente chamada. O nome da aplicação é *download* e o único argumento deve ser o URL conforme descrito no RFC1738. Veja-se o exemplo de como deve ser executada:

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

Os campos *user* e *password* são facultativos e indicam as credenciais do utilizador naquele servidor, podendo ser omitidas caso o servidor o permita. O campo *host* indica o nome do servidor em que está alojado e que, através do DNS será convertido num endereço IPv4 ou IPv6. Nesta aplicação foi dada especial atenção ao facto de um *host* poder ter vários IPs, assim tem que ser testadas as várias hipóteses até que haja um IP que permita estabelecer uma ligação FTP. Por convenção, a porta para ligações FTP é a 21. Após ser definida uma ligação (abrir um *socket* e ligar ao servidor com sucesso), é então necessário seguir o protocolo de acordo com o RFC959. Neste relatório é descrito que após uma ligação bem estabelecida entre o recetor (cliente) e o emissor (servidor), este deve enviar uma (ou mais) mensagem de sucesso com o código 220.

Caso seja necessário, a aplicação irá fazer o *login* no servidor. Para tal, o utilizador e password especificados no URL são enviados através dos comandos *USER [username]<CR><LF>* (em caso de sucesso, o servidor retorna uma, ou mais, mensagens com o código 331) e *PASS [password]<CR><LF>* (em caso de sucesso, o servidor retorna uma, ou mais, mensagens com o código 230).

Em seguida, é pedido ao servidor para fazer a transferência em modo passivo, ou seja, a transferência será feita noutra porta e possivelmente noutro servidor e este deverá só iniciar a transferência assim que o cliente se conectar. Desta forma, teremos uma ligação para os comandos e outra para o ficheiro. Assim, o comando *PASV<CR><LF>* em caso de sucesso retornará uma mensagem com a seguinte configuração:

```
227 Entering Passive Mode (ip1,ip2,ip3,ip4,p1,p2)
```

Esta resposta deve ser interpretada da seguinte forma:

- IP do servidor (IPv4) onde vai ser feita a transferência: ip1.ip2.ip3.ip4
- Porta da ligação:  $p1 * 256 + p2$

Aqui, a configuração da ligação já está feita e só falta pedir o ficheiro especificado no URL com o comando *RETR [path\_to\_the\_file]<CR><LF>* para iniciar a transferência.

Concluindo, é necessário abrir uma nova ligação no endereço IP e porta especificadas aquando do pedido para transferência em modo passivo. A transferência é iniciada logo que o cliente se conecte.

### **Relatório de um *download* com sucesso**

A aplicação desenvolvida consegue fazer transferências de servidores por FTP. Assim, segue-se um caso de utilização com as diversas mensagens que aparecem no ecrã:

- Execução da aplicação:  
`./download ftp://up201303568:pass@tom.fe.up.pt/public_html/1.jpg`
- Ligação bem sucedida:  
`220 FTP for Alf/Tom/Crazy/Pinguim`
- *Username* aceite:  
`331 Please specify the password.`
- *Password* aceite:  
`230 Login successful.`  
`password.`
- Resposta ao comando do modo passivo:  
`227 Entering Passive Mode (192,168,50,138,85,45).`
- Término da transferência com sucesso:  
`Transfer completed!`

## Parte 2 – Análise e configuração da rede

### Experiência 1 – Configuração de um IP de rede

Nesta experiência foi possível estabelecer uma comunicação entre dois computadores. Como inicialmente estes computadores não tinham ainda qualquer tipo de comunicação (as tabelas ARP foram apagadas em ambos), foi necessário, no momento da comunicação, identificar os computadores de destino. Assim, como se pode ver no anexo B1, linha 33, o computador 1 “pergunta” a todos os computadores qual é o computador com o endereço 172.16.50.254 e na linha seguinte é possível ver a resposta do referido computador, através do endereço MAC. Esta comunicação serviu para identificar “quem” é que está no endereço IP 172.16.50.254 e os pacotes usados para a transmissão são ARP (*Address Resolution Protocol*). O endereço IP distingue o computador na rede e o MAC identifica o computador. Quando é feito um *ping* de um computador de partida para o outro são passadas duas mensagens: a primeira é um *echo request* do computador de partida para o de destino, que responde com um *echo reply*. Para identificar o computador de destino o pacote *ping* contém o endereço MAC e IP. Esta situação é perceptível nas linhas 34 e 35 do anexo supracitado. O *ping* é um descendente do ICMP (*Internet Control Message Protocol*). Para decodificar estes pacotes há que os distinguir através das *frames* conforme descritas no slide 27 do pdf Network das aulas teóricas. Por conseguinte, o *type of service* indica o tipo de serviço, para o qual existem valores universais descritos no slide 42 do mesmo pdf. O tamanho total da *frame* é o campo *length* com 16 bits. A interface *loopback* serve para testar o funcionamento do dispositivo de rede. As mensagens mandadas para o endereço 127.x.x.x são imediatamente recebidas no mesmo dispositivo. Este funcionamento é especialmente importante para o teste de falhas como, por exemplo, para saber se os pinos ou o cabo estão em bom estado.

### Experiência 2 – Implementação de duas *virtual* LANs num *switch*

Nesta experiência foram criadas duas VLAN no switch. Inicialmente, começamos por configurar o *switch*, através da sua consola de configuração. Assim, com o comando ***vlan [n]***, onde ***n*** é o identificador da VLAN, foi criada uma nova sub-rede. Depois, revelou-se necessário adicionar as respetivas portas a cada VLAN. Para tal, introduziram-se os comandos ***interface fastethernet 0/[i]***, onde ***i*** é o número da porta, ***switchport mode access*** e ***switchport access VLAN [n]***, onde ***n*** é o identificador da VLAN. Os comandos estão por extenso no anexo C, experiência 2.

Uma virtual LAN permite simular uma rede diferente e inatingível pelos computadores exteriores a ela, exceto no caso de não terem nenhum elo de ligação (como acontecerá nas próximas experiências). Assim, existem 2 *broadcast domains*, um em cada VLAN. Como é possível ver pelos *logs* do anexo B2, os computadores das figuras 1 e 3 estão na mesma rede. Quando o computador da figura 1 fez um *broadcast*, só os computadores dessa rede é que o receberam. Contudo, através da análise dos *logs* também é possível perceber que o computador da figura 2 não está na mesma rede. Na verdade, está ligado ao mesmo *switch* que os restantes computadores, mas numa VLAN separada. Assim, concluímos que a cada VLAN corresponde um novo *broadcast domain*.

### Experiência 3 – Configuração de um *router* em Linux

Nesta experiência, existem duas VLANs ligadas entre si pelo computador 4. Os computadores da VLAN 50 conseguem comunicar com os computadores da VLAN 51, através das rotas adicionadas aos computadores dentro das VLANs. Assim, no computador 1 foi criada uma rota para quando este pretender comunicar com um dispositivo da outra VLAN reencaminhe o pacote para o computador 4. Para a comunicação da outra VLAN, ligada ao computador 2, foi também introduzida uma rota para reencaminhar os pacotes para o elo de ligação. Quando se pretende adicionar uma rota à tabela de reencaminhamento é necessário introduzir o endereço de destino, a respetiva máscara e o endereço para o qual se pretende reencaminhar. Exemplificando: o comando ***route add -net 172.16.51.0/24 gw 172.16.50.254*** faz com que os endereços que tenham os primeiros 24 bits iguais a 172.16.51 sejam reencaminhados para o endereço 172.16.50.254.

Quando um computador comunica com outro, este guarda na tabela de reencaminhamento o endereço IP e o respetivo endereço MAC do computador de destino. Assim, quando o computador 1 comunica com o computador 2, o IP de destino é o computador 2, mas o MAC é o do computador 4, uma vez que este é o elo de ligação entre os dois computadores. Este exemplo está representado na figura 4 do anexo B3.

No decorrer desta experiência foi feito *ping* do computador 1 para o computador 2 (figura 5 do anexo B3). A porta *ethernet* eth0 do computador 4 estava ligada à VLAN do computador 1 (figura 6 do anexo B3) e porta eth1 na mesma VLAN que a do outro computador (figura 7 do anexo B3). Na análise destes *logs* foi possível concluir que um pacote do computador 1 passa pela porta eth0 para o computador 2 e a comunicação inversa passa pela porta eth1. Nota: nas figuras 5, 6 e 7 do anexo B3, os pacotes podem não corresponder. No entanto, apenas o id da trama é que varia, não alterando assim a conclusão desta experiência.

### Experiência 4 – Configuração de um *router* comercial e implementação de NAT

Nesta experiência foi configurado um *router* comercial com NAT (*Network Address Resolution*) devidamente implementado. Quando se introduziu o *router* à rede foi necessário configurá-lo. Em primeiro lugar, foram configuradas as interfaces *gigabitethernet* do *router*. Em seguida adicionaram-se as rotas estáticas, utilizadas para reencaminhar os pacotes pelo computador 4 (elo de ligação) sempre que seja necessário comunicar com a outra VLAN. Os comandos utilizados estão em detalhe no anexo B4. O modo de funcionamento das máscaras e IP já foi explicado nas experiências anteriores.

Após a configuração do *router*, percebemos que um pacote ao ser transmitido para outra VLAN, por exemplo do computador 2 para o computador 1, segue por um caminho maior (passa pelo *router* comercial) para chegar ao destino. Isto só acontece caso esse computador não tenha adicionado na tabela de reencaminhado o *router* criado (computador 4).

De seguida foi implementado o NAT que permitiu, com sucesso, a comunicação dos computadores da rede criada com redes externas. Esta implementação permite que o endereço do computador local seja traduzido para outro IP na *internet*. A comunicação passa então por uma porta personalizada e o *router* comercial passa os dados recebidos para o computador através do seu endereço local. Sem o NAT implementado, o *router* comercial não é capaz de reencaminhar os pacotes exteriores para os computadores da rede local.



Para implementar o NAT foi necessário garantir a gama de endereços com os comandos ***ip nat pool ovrld 172.16.1.59 172.16.1.59 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. Para finalizar, foram definidas as rotas internas e externas sendo que a comunicação com a outra VLAN deveria passar pelo computador 4.

No decorrer desta experiência, já com o NAT implementado, o computador 1 fez um *ping* ao *router* da sala (IP = 172.16.1.254) e verificou-se o caminho dos pacotes. Assim, esse *ping* passou pelo computador 1 para o computador 4, passando para o *router* comercial e chegando à *internet* (ao *router* da sala). A resposta ao *ping* partiu do *router* da sala para o *router* comercial, seguiu para o computador 4 e chegou com sucesso ao computador 1.

### **Experiência 5 – DNS**

Nesta experiência foi configurado o DNS com o servidor DNS `lixa.netlab.fe.up.pt` (172.16.1.1). O DNS converte uma *string*, que é o nome do *site* a que pretendemos aceder, num endereço IP. Para fazer esta configuração foi necessário alterar o ficheiro “*resolv.conf*”, utilizando os comandos presentes no anexo C, experiência 5. Este ficheiro é lido cada vez que se faz um acesso à Internet.

Para perceber o funcionamento do DNS foi feito *ping* a [www.google.pt](http://www.google.pt). Como já referido anteriormente, esta *string* é traduzida num IP para fazer as futuras comunicações. Assim, como se pode ver nas linhas 10 e 11 do anexo B5, é feita uma *query* ao DNS para saber qual o IP. Em seguida, é obtida uma resposta, neste caso, com o IP 218.58.211.195. A partir deste momento, os pacotes trocados entre estes dois dispositivos é feito com este endereço, deixando a *string* de ser importante. As linhas a vermelho-claro presentes no anexo B5, são uma troca de mensagens, tal como as que foram esclarecidas nas experiências anteriores.

### **Experiência 6 – Ligações TCP**

Nesta experiência foi feito um *download* através do protocolo FTP da *internet*. A aplicação usada para a transferência está descrita na Parte 1 deste relatório. Como referido anteriormente, são abertas duas ligações de TCP para cada *download*, uma vez que este é feito em modo passivo. Alguns detalhes inerentes a esta experiência estão também explicitados anteriormente.

Como é possível observar no anexo B6 figura 9, os comandos do cliente seguem o protocolo TCP e os dados do servidor o protocolo FTP. Contudo, após o envio de todos os comandos e aberta a nova porta para a transferência de dados, os protocolos são invertidos passando o FTP estar a cargo do cliente e o TCP do servidor (ver anexo B6, figura 10). A informação, ou seja, o envio dos comandos, é assim passado na primeira ligação.

O TCP utiliza *Selective Repeat ARQ* para o tratamento de erros, o seu funcionamento encontra-se explicado no slide 43 do ficheiro `data-link-layer.pdf` das apresentações das aulas teóricas. Conforme explicado nos referidos *slides*, apenas as tramas que falharam são reenviadas e o uso da *Slicing Window* limita o número de erros. Por conseguinte, é possível ver os campos enviados do cliente para o servidor no anexo B6, figura 11. Estes são maioritariamente tramas de confirmação positivas (ACK) contendo o número de sequência e o tamanho da janela esperada. Esta informação, tal como uma explicação sobre o funcionamento do TCP ao nível da gestão de erros e confirmações e o transporte que é feito, está melhor e mais detalhadamente explicada no ficheiro `transport.pdf` das aulas teóricas.

Nesta experiência foi também feita a transferência do mesmo ficheiro, do mesmo *host*

através de dois computadores intercalados (primeiro começou um computador e depois, o segundo computador começou também), com o mesmo ponto de acesso à internet: o *router* comercial. Como era expectável, a transferência foi ligeiramente afetada. No entanto, as consequências foram diminutas, uma vez que o tempo de transferência manteve-se relativamente o mesmo de quando as transferências foram feitas isoladamente. Mais evidente foram as alterações das janelas pelos clientes, já que tanto aumentavam como diminuía, sendo este um sinal de que o *buffer* enchia mais rapidamente e de forma inconstante.

### **Experiência 7 – Implementação de NAT em Linux**

Nesta experiência foi implementado NAT em Linux no computador 4 (a ligação entre as duas VLANs). Nas anteriores experiências, este computador apenas reencaminhava os pacotes entre as duas VLANs. Contudo, agora, este computador associa cada IP de cada computador que deseja fazer uma ligação com uma porta e, consequentemente, este IP é traduzido para um IP global diferente. Esta tradução é feita por uma tabela de reencaminhamento. Resumidamente, este funcionamento é o NAT a atuar (para uma explicação detalhada sobre como o NAT funciona sirva-se a Experiência 4 como exemplo). Este modo de funcionamento permitiu o grupo concluir que um computador pode ter o NAT implementado e, assim simular o funcionamento de um *router* doméstico com a *internet*.

## Conclusões

No final deste trabalho, o grupo realizou com sucesso todas as experiências propostas. Com as várias experiências, fizemos uma importante aplicação prática dos conceitos teóricos abordados ao longo do semestre.

A aplicação de *download* implicou um estudo prévio de alguns protocolos, de FTP e de sintaxe do URL. A análise e configuração da rede permitiu interagir diretamente com o material físico necessário, bem como, as configurações a ele associadas.

Numa altura em que cada vez mais os dados estão na *cloud*, torna-se importante perceber como é o funcionamento de uma rede e a sua aplicação em ambiente profissional.

Contudo, concluímos também que faltam alguns conceitos para a criação de uma rede mais completa. Os objetivos desta unidade curricular foram compreendidos, no entanto, criar uma rede a nível empresarial é uma situação mais complexa e exigente, que ficará para uma abordagem futura.

## **Referências**

- Slides da unidade curricular Redes de Computadores
- Brian Hall Beej's Guide to Network Programming - Using Internet Sockets
- J. Postel, J. Reinolds File Transfer Protocol
- T. Berners-Lee Uniform Resource Locators

## Anexos

### Anexo A – Código da Aplicação

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <string.h>
#include <netdb.h>
#include <arpa/inet.h>
#include <fcntl.h>
#define PORT "21" // the port client will be connecting to
#define MAXDATASIZE 10000 // max number of bytes we can get at once

// get sockaddr, IPv4 or IPv6:
void *get_in_addr(struct sockaddr *sa)
{
    if (sa->sa_family == AF_INET) {
        return &((struct sockaddr_in*)sa)->sin_addr;
    }
    return &((struct sockaddr_in6*)sa)->sin6_addr;
}

//abstraction to send user and pass cmd
void sendCmdAndWaitResponse(int fd, char *cmd, int length) {

    char buf[MAXDATASIZE];

    if (write(fd, cmd, length) == -1) {
        perror("send");
        exit(1);
    }
    if (read(fd, buf, MAXDATASIZE-1) > 0) {
        printf("%s\n", buf);
    }
}

//enter passive mode. it returns the port where the data will come out
//server_address will be the ip of the server
int enterPassive(int sockfd, char *server_address) {
    char response[MAXDATASIZE];
    write(sockfd, "pasv\r\n", strlen("pasv\r\n"));
    read(sockfd, response, MAXDATASIZE-1);

    int ipPart1, ipPart2, ipPart3, ipPart4;
    int port1, port2;
    sscanf(response, "227 Entering Passive Mode
(%d,%d,%d,%d,%d,%d)", &ipPart1,
                &ipPart2, &ipPart3, &ipPart4, &port1, &port2);

    sprintf(server_address, "%d.%d.%d.%d", ipPart1, ipPart2,
ipPart3, ipPart4);

    return port1 * 256 + port2;
}

//parse information of the user
void parseUser(char *argv, char *sendUser, char *sendPass, char *host,
char *path, char *userInfo) {

    char *username = strchr(argv+strlen("ftp://"), ':');
```

```

    if (username == NULL || username > userInfo) {
        fprintf(stderr, "Check the user information\n");
        exit(1);
    }

    char *field = argv+strlen("ftp://");
    strcpy(sendUser, "USER ");
    strncat(sendUser, field, abs(username - field));
    strncat(sendUser, "\r\n", 2);

    field = username + 1;
    strcpy(sendPass, "PASS ");
    strncat(sendPass, field, abs(userInfo - field));
    strncat(sendPass, "\r\n", 2);

    userInfo = userInfo + 1;
    field = strchr(userInfo, '/');
    strncpy(host, userInfo, abs(field - userInfo));
    strcat(path, field+1);
    strncat(path, "\r\n", 2);
}

//parse filename
void parseFilename(char *argv, char *filename) {
    strcpy(filename, ".");

    char *name = strrchr(argv, '/');

    strcat(filename, name);
}

//para testar
//ftp://user:pass@ftp.funet.fi/pub/standards/RFC/rfc959.txt
//ftp://ftp.up.pt/pub/CPAN/RECENT-1M.json
int main(int argc, char *argv[])
{
    if (argc != 2) {
        fprintf(stderr, "usage: download
ftp://[<user>:<password>@]<host>/<url-path>\n");
        return 1;
    }
    if (strncmp("ftp://", argv[1], strlen("ftp://")) != 0) {
        fprintf(stderr, "The link must start with ftp://\n");
        return 1;
    }

    char *userInfo = strchr(argv[1], '@');
    char sendUser[1000]; char sendPass[1000]; char host[1000];
    char path[1000]; strcpy(path, "RETR ");
    //verfica se existe um utilizador
    if (userInfo != NULL) {
        parseUser(argv[1], sendUser, sendPass, host, path,
userInfo);
    }
    else {
        char *begin = argv[1]+strlen("ftp://");
        char *field = strchr(begin, '/');
        strncpy(host, begin, abs(field - begin));
        strcat(path, field+1);
        strncat(path, "\r\n", 2);
    }
}

```

```

int sockfd;
struct addrinfo hints, *servinfo, *p;
int rv;
char s[INET6_ADDRSTRLEN];
memset(&hints, 0, sizeof hints);
hints.ai_family = AF_UNSPEC;
hints.ai_socktype = SOCK_STREAM;
if ((rv = getaddrinfo(host, PORT, &hints, &servinfo)) != 0) {
    fprintf(stderr, "getaddrinfo: %s\n", gai_strerror(rv));
    return 1;
}
// loop through all the results and connect to the first we can
//one host can have multiple ip
for(p = servinfo; p != NULL; p = p->ai_next) {
    if ((sockfd = socket(p->ai_family, p->ai_socktype,
        p->ai_protocol)) == -1) {
        perror("client: socket");
        continue;
    }

    if (connect(sockfd, p->ai_addr, p->ai_addrlen) == -1) {
        close(sockfd);
        perror("client: connect");
        continue;
    }
    break;
}

//unsuccessful connection
if (p == NULL) {
    fprintf(stderr, "client: failed to connect\n");
    return 2;
}
inet_ntop(p->ai_family, get_in_addr((struct sockaddr *)p->ai_addr),
    s, sizeof s);

//printf("client: connecting to %s\n", s);
freeaddrinfo(servinfo); // all done with this structure

//read the welcome message
char buf[MAXDATASIZE];
if (read(sockfd, buf, MAXDATASIZE-1) > 0) {
    printf("%s\n", buf);
}

//send user and pass cmd if required
if (userInfo != NULL) {
    sendCmdAndWaitResponse(sockfd, sendUser, strlen(sendUser));
    sendCmdAndWaitResponse(sockfd, sendPass, strlen(sendPass));
} else {
    strcpy(sendUser, "USER ftp");
    strncat(sendUser, "\r\n", 2);
    strcpy(sendPass, "PASS pass");
    strncat(sendPass, "\r\n", 2);
    sendCmdAndWaitResponse(sockfd, sendUser, strlen(sendUser));
    sendCmdAndWaitResponse(sockfd, sendPass, strlen(sendPass));
}

char server_address[20];

```

```

int server_port = enterPassive(sockfd, server_address);

//send the file requested
write(sockfd, path, strlen(path) + 1);

//starts opening the address of the data
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(server_address); //32
bit Internet address network byte ordered
server_addr.sin_port = htons(server_port); //server
TCP port must be network byte ordered

int sockfd_server;
//open an TCP socket
if ((sockfd_server = socket(AF_INET, SOCK_STREAM, 0)) < 0) {
    perror("socket()");
    exit(1);
}

//connect to the server
if(connect(sockfd_server,
           (struct sockaddr *)&server_addr,
           sizeof(server_addr)) < 0){
    perror("connect()");
    exit(1);
}

//open the file
char filename[MAXDATASIZE];
parseFilename(argv[1], filename);
int file = open(filename, O_WRONLY | O_CREAT, 0777);

//read to the file
char print;
while (read(sockfd_server, &print, 1) == 1) {
    write(file, &print, 1);
}

//all done with the transmission
close(file);
close(sockfd_server);
close(sockfd);

printf("Transfer completed!\n");
return 0;
}

```



## Anexo B

### Anexo B1

No.	Time	Source	Destination	Protoc	Length	Info
31 42...	CiscoInc_3a:f6:...	Spanning-tree-(...	STP	60	Conf. Root = 32768/1/fc:fb:fb:3a:f6:00 Cost = 0 Port = 0x8007	
32 44...	CiscoInc_3a:f6:...	Spanning-tree-(...	STP	60	Conf. Root = 32768/1/fc:fb:fb:3a:f6:00 Cost = 0 Port = 0x8007	
33 45...	G-ProCom_8b:e4:...	Broadcast	ARP	42	Who has 172.16.50.254? Tell 172.16.50.1	
34 45...	HewlettP_c3:78:...	G-ProCom_8b:e4:...	ARP	60	172.16.50.254 is at 00:21:5a:c3:78:70	
35 45...	172.16.50.1	172.16.50.254	ICMP	98	Echo (ping) request id=0x4f6d, seq=1/256, ttl=64 (reply in 36)	
36 45...	172.16.50.254	172.16.50.1	ICMP	98	Echo (ping) reply id=0x4f6d, seq=1/256, ttl=64 (request in 35)	
37 46...	CiscoInc_3a:f6:...	Spanning-tree-(...	STP	60	Conf. Root = 32768/1/fc:fb:fb:3a:f6:00 Cost = 0 Port = 0x8007	
38 46...	172.16.50.1	172.16.50.254	ICMP	98	Echo (ping) request id=0x4f6d, seq=2/512, ttl=64 (reply in 39)	
39 46...	172.16.50.254	172.16.50.1	ICMP	98	Echo (ping) reply id=0x4f6d, seq=2/512, ttl=64 (request in 38)	
40 47...	172.16.50.1	172.16.50.254	ICMP	98	Echo (ping) request id=0x4f6d, seq=3/768, ttl=64 (reply in 41)	
41 47...	172.16.50.254	172.16.50.1	ICMP	98	Echo (ping) reply id=0x4f6d, seq=3/768, ttl=64 (request in 40)	
42 48...	CiscoInc_3a:f6:...	Spanning-tree-(...	STP	60	Conf. Root = 32768/1/fc:fb:fb:3a:f6:00 Cost = 0 Port = 0x8007	
43 48...	172.16.50.1	172.16.50.254	ICMP	98	Echo (ping) request id=0x4f6d, seq=4/1024, ttl=64 (reply in 44)	
44 48...	172.16.50.254	172.16.50.1	ICMP	98	Echo (ping) reply id=0x4f6d, seq=4/1024, ttl=64 (request in 43)	
45 48...	CiscoInc_3a:f6:...	CDP/VTP/DTP/Pag...	CDP	453	Device ID: tux-sw5 Port ID: FastEthernet0/5	
46 49...	172.16.50.1	172.16.50.254	ICMP	98	Echo (ping) request id=0x4f6d, seq=5/1280, ttl=64 (reply in 47)	
47 49...	172.16.50.254	172.16.50.1	ICMP	98	Echo (ping) reply id=0x4f6d, seq=5/1280, ttl=64 (request in 46)	
48 50...	CiscoInc_3a:f6:...	Spanning-tree-(...	STP	60	Conf. Root = 32768/1/fc:fb:fb:3a:f6:00 Cost = 0 Port = 0x8007	

### Anexo B2

10 12...	172.16.50.1	172.16.50.255	ICMP	98	Echo (ping) request id=0x3d4b, seq=1/256, ttl=64 (no response found!)
11 13...	172.16.50.1	172.16.50.255	ICMP	98	Echo (ping) request id=0x3d4b, seq=2/512, ttl=64 (no response found!)
12 14...	CiscoInc_3a:f6:...	Spanning-tree-(...	STP	60	Conf. Root = 32768/50/fc:fb:fb:3a:f6:00 Cost = 0 Port = 0x8003
13 14...	172.16.50.1	172.16.50.255	ICMP	98	Echo (ping) request id=0x3d4b, seq=3/768, ttl=64 (no response found!)
14 15...	172.16.50.1	172.16.50.255	ICMP	98	Echo (ping) request id=0x3d4b, seq=4/1024, ttl=64 (no response found!)
15 16...	CiscoInc_3a:f6:...	Spanning-tree-(...	STP	60	Conf. Root = 32768/50/fc:fb:fb:3a:f6:00 Cost = 0 Port = 0x8003
16 16...	172.16.50.1	172.16.50.255	ICMP	98	Echo (ping) request id=0x3d4b, seq=5/1280, ttl=64 (no response found!)
17 17...	172.16.50.1	172.16.50.255	ICMP	98	Echo (ping) request id=0x3d4b, seq=6/1536, ttl=64 (no response found!)
18 18...	CiscoInc_3a:f6:...	Spanning-tree-(...	STP	60	Conf. Root = 32768/50/fc:fb:fb:3a:f6:00 Cost = 0 Port = 0x8003
19 18...	172.16.50.1	172.16.50.255	ICMP	98	Echo (ping) request id=0x3d4b, seq=7/1792, ttl=64 (no response found!)
20 19...	172.16.50.1	172.16.50.255	ICMP	98	Echo (ping) request id=0x3d4b, seq=8/2048, ttl=64 (no response found!)
21 20...	CiscoInc_3a:f6:...	Spanning-tree-(...	STP	60	Conf. Root = 32768/50/fc:fb:fb:3a:f6:00 Cost = 0 Port = 0x8003
22 20...	172.16.50.1	172.16.50.255	ICMP	98	Echo (ping) request id=0x3d4b, seq=9/2304, ttl=64 (no response found!)
23 21...	172.16.50.1	172.16.50.255	ICMP	98	Echo (ping) request id=0x3d4b, seq=10/2560, ttl=64 (no response found!)

Figura 1

1 0...	CiscoInc_3a:f6:...	Spanning-tree-(...	STP	60	Conf. Root = 32768/51/fc:fb:fb:3a:f6:00 Cost = 0 Port = 0x8004
2 2...	CiscoInc_3a:f6:...	Spanning-tree-(...	STP	60	Conf. Root = 32768/51/fc:fb:fb:3a:f6:00 Cost = 0 Port = 0x8004
3 3...	CiscoInc_3a:f6:...	CiscoInc_3a:f6:...	LOOP	60	Reply
4 4...	CiscoInc_3a:f6:...	Spanning-tree-(...	STP	60	Conf. Root = 32768/51/fc:fb:fb:3a:f6:00 Cost = 0 Port = 0x8004
5 6...	CiscoInc_3a:f6:...	Spanning-tree-(...	STP	60	Conf. Root = 32768/51/fc:fb:fb:3a:f6:00 Cost = 0 Port = 0x8004
6 8...	CiscoInc_3a:f6:...	Spanning-tree-(...	STP	60	Conf. Root = 32768/51/fc:fb:fb:3a:f6:00 Cost = 0 Port = 0x8004
7 10...	CiscoInc_3a:f6:...	Spanning-tree-(...	STP	60	Conf. Root = 32768/51/fc:fb:fb:3a:f6:00 Cost = 0 Port = 0x8004
8 12...	CiscoInc_3a:f6:...	Spanning-tree-(...	STP	60	Conf. Root = 32768/51/fc:fb:fb:3a:f6:00 Cost = 0 Port = 0x8004
9 13...	CiscoInc_3a:f6:...	CiscoInc_3a:f6:...	LOOP	60	Reply
10 14...	CiscoInc_3a:f6:...	Spanning-tree-(...	STP	60	Conf. Root = 32768/51/fc:fb:fb:3a:f6:00 Cost = 0 Port = 0x8004
11 16...	CiscoInc_3a:f6:...	Spanning-tree-(...	STP	60	Conf. Root = 32768/51/fc:fb:fb:3a:f6:00 Cost = 0 Port = 0x8004
12 18...	CiscoInc_3a:f6:...	Spanning-tree-(...	STP	60	Conf. Root = 32768/51/fc:fb:fb:3a:f6:00 Cost = 0 Port = 0x8004
13 20...	CiscoInc_3a:f6:...	Spanning-tree-(...	STP	60	Conf. Root = 32768/51/fc:fb:fb:3a:f6:00 Cost = 0 Port = 0x8004
14 22...	CiscoInc_3a:f6:...	Spanning-tree-(...	STP	60	Conf. Root = 32768/51/fc:fb:fb:3a:f6:00 Cost = 0 Port = 0x8004
15 23...	CiscoInc_3a:f6:...	CiscoInc_3a:f6:...	LOOP	60	Reply
16 24...	CiscoInc_3a:f6:...	Spanning-tree-(...	STP	60	Conf. Root = 32768/51/fc:fb:fb:3a:f6:00 Cost = 0 Port = 0x8004
17 26...	CiscoInc_3a:f6:...	Spanning-tree-(...	STP	60	Conf. Root = 32768/51/fc:fb:fb:3a:f6:00 Cost = 0 Port = 0x8004
18 28...	CiscoInc_3a:f6:...	Spanning-tree-(...	STP	60	Conf. Root = 32768/51/fc:fb:fb:3a:f6:00 Cost = 0 Port = 0x8004
19 30...	CiscoInc_3a:f6:...	Spanning-tree-(...	STP	60	Conf. Root = 32768/51/fc:fb:fb:3a:f6:00 Cost = 0 Port = 0x8004
20 32...	CiscoInc_3a:f6:...	Spanning-tree-(...	STP	60	Conf. Root = 32768/51/fc:fb:fb:3a:f6:00 Cost = 0 Port = 0x8004
21 33...	CiscoInc_3a:f6:...	CiscoInc_3a:f6:...	LOOP	60	Reply
22 34...	CiscoInc_3a:f6:...	Spanning-tree-(...	STP	60	Conf. Root = 32768/51/fc:fb:fb:3a:f6:00 Cost = 0 Port = 0x8004
23 36...	CiscoInc_3a:f6:...	Spanning-tree-(...	STP	60	Conf. Root = 32768/51/fc:fb:fb:3a:f6:00 Cost = 0 Port = 0x8004
24 38...	CiscoInc_3a:f6:...	Spanning-tree-(...	STP	60	Conf. Root = 32768/51/fc:fb:fb:3a:f6:00 Cost = 0 Port = 0x8004

Figura 2

22	32...	172.16.50.1	172.16.50.255	ICMP	98 Echo (ping) request	id=0x3d4b, seq=1/256, ttl=64 (no response found!)
23	33...	172.16.50.1	172.16.50.255	ICMP	98 Echo (ping) request	id=0x3d4b, seq=2/512, ttl=64 (no response found!)
24	34...	CiscoInc_3a:f6:...	Spanning-tree-(...	STP	60 Conf. Root = 32768/50/fc:fb:fb:3a:f6:00	Cost = 0 Port = 0x8006
25	34...	172.16.50.1	172.16.50.255	ICMP	98 Echo (ping) request	id=0x3d4b, seq=3/768, ttl=64 (no response found!)
26	35...	172.16.50.1	172.16.50.255	ICMP	98 Echo (ping) request	id=0x3d4b, seq=4/1024, ttl=64 (no response found!)
27	36...	CiscoInc_3a:f6:...	Spanning-tree-(...	STP	60 Conf. Root = 32768/50/fc:fb:fb:3a:f6:00	Cost = 0 Port = 0x8006
28	36...	172.16.50.1	172.16.50.255	ICMP	98 Echo (ping) request	id=0x3d4b, seq=5/1280, ttl=64 (no response found!)
29	37...	172.16.50.1	172.16.50.255	ICMP	98 Echo (ping) request	id=0x3d4b, seq=6/1536, ttl=64 (no response found!)
30	38...	CiscoInc_3a:f6:...	Spanning-tree-(...	STP	60 Conf. Root = 32768/50/fc:fb:fb:3a:f6:00	Cost = 0 Port = 0x8006
31	38...	172.16.50.1	172.16.50.255	ICMP	98 Echo (ping) request	id=0x3d4b, seq=7/1792, ttl=64 (no response found!)
32	39...	172.16.50.1	172.16.50.255	ICMP	98 Echo (ping) request	id=0x3d4b, seq=8/2048, ttl=64 (no response found!)
33	40...	CiscoInc_3a:f6:...	Spanning-tree-(...	STP	60 Conf. Root = 32768/50/fc:fb:fb:3a:f6:00	Cost = 0 Port = 0x8006
34	40...	172.16.50.1	172.16.50.255	ICMP	98 Echo (ping) request	id=0x3d4b, seq=9/2304, ttl=64 (no response found!)
35	41...	172.16.50.1	172.16.50.255	ICMP	98 Echo (ping) request	id=0x3d4b, seq=10/2560, ttl=64 (no response found!)
36	42...	CiscoInc_3a:f6:...	CiscoInc_3a:f6:...	LOOP	60 Reply	
37	42...	CiscoInc_3a:f6:...	Spanning-tree-(...	STP	60 Conf. Root = 32768/50/fc:fb:fb:3a:f6:00	Cost = 0 Port = 0x8006

Figura 3

## Anexo B3

→	114	11...	172.16.50.1	172.16.51.1	ICMP	98 Echo (ping) request	id=0x51ef, seq=1/256, ttl=64 (reply in 115)
←	115	11...	172.16.51.1	172.16.50.1	ICMP	98 Echo (ping) reply	id=0x51ef, seq=1/256, ttl=63 (request in 114)
> Frame 114: 98 bytes on wire (784 bits), 98 bytes captured (784 bits) on interface 0							
> Ethernet II, Src: G-ProCom_8b:e4:a7 (00:0f:fe:8b:e4:a7), Dst: HewlettP_c3:78:70 (00:21:5a:c3:78:70)							
> Internet Protocol Version 4, Src: 172.16.50.1, Dst: 172.16.51.1							
> Internet Control Message Protocol							

Figura 4

112	11...	CiscoInc_3a:f6:...	Spanning-tree-(...	STP	60 Conf. Root = 32768/50/fc:fb:fb:3a:f6:00	Cost = 0 Port = 0x8003
113	11...	CiscoInc_3a:f6:...	CiscoInc_3a:f6:...	LOOP	60 Reply	
114	11...	172.16.50.1	172.16.51.1	ICMP	98 Echo (ping) request	id=0x51ef, seq=1/256, ttl=64 (reply in 115)
115	11...	172.16.51.1	172.16.50.1	ICMP	98 Echo (ping) reply	id=0x51ef, seq=1/256, ttl=63 (request in 114)
116	11...	CiscoInc_3a:f6:...	Spanning-tree-(...	STP	60 Conf. Root = 32768/50/fc:fb:fb:3a:f6:00	Cost = 0 Port = 0x8003
117	11...	172.16.50.1	172.16.51.1	ICMP	98 Echo (ping) request	id=0x51ef, seq=2/512, ttl=64 (reply in 118)
118	11...	172.16.51.1	172.16.50.1	ICMP	98 Echo (ping) reply	id=0x51ef, seq=2/512, ttl=63 (request in 117)
119	11...	172.16.50.1	172.16.51.1	ICMP	98 Echo (ping) request	id=0x51ef, seq=3/768, ttl=64 (reply in 120)
120	11...	172.16.51.1	172.16.50.1	ICMP	98 Echo (ping) reply	id=0x51ef, seq=3/768, ttl=63 (request in 119)
121	11...	CiscoInc_3a:f6:...	Spanning-tree-(...	STP	60 Conf. Root = 32768/50/fc:fb:fb:3a:f6:00	Cost = 0 Port = 0x8003
122	11...	172.16.50.1	172.16.51.1	ICMP	98 Echo (ping) request	id=0x51ef, seq=4/1024, ttl=64 (reply in 123)
123	11...	172.16.51.1	172.16.50.1	ICMP	98 Echo (ping) reply	id=0x51ef, seq=4/1024, ttl=63 (request in 122)
124	11...	172.16.50.1	172.16.51.1	ICMP	98 Echo (ping) request	id=0x51ef, seq=5/1280, ttl=64 (reply in 125)
125	11...	172.16.51.1	172.16.50.1	ICMP	98 Echo (ping) reply	id=0x51ef, seq=5/1280, ttl=63 (request in 124)
126	11...	CiscoInc_3a:f6:...	Spanning-tree-(...	STP	60 Conf. Root = 32768/50/fc:fb:fb:3a:f6:00	Cost = 0 Port = 0x8003

Figura 5

10	14...	CiscoInc_3a:f6:...	Spanning-tree-(...	STP	60 Conf. Root = 32768/50/fc:fb:fb:3a:f6:00	Cost = 0 Port = 0x8006
11	15...	172.16.50.1	172.16.51.1	ICMP	98 Echo (ping) request	id=0x55cf, seq=1/256, ttl=64 (reply in 12)
12	15...	172.16.51.1	172.16.50.1	ICMP	98 Echo (ping) reply	id=0x55cf, seq=1/256, ttl=63 (request in 11)
13	16...	CiscoInc_3a:f6:...	Spanning-tree-(...	STP	60 Conf. Root = 32768/50/fc:fb:fb:3a:f6:00	Cost = 0 Port = 0x8006
14	16...	172.16.50.1	172.16.51.1	ICMP	98 Echo (ping) request	id=0x55cf, seq=2/512, ttl=64 (reply in 15)
15	16...	172.16.51.1	172.16.50.1	ICMP	98 Echo (ping) reply	id=0x55cf, seq=2/512, ttl=63 (request in 14)
16	17...	172.16.50.1	172.16.51.1	ICMP	98 Echo (ping) request	id=0x55cf, seq=3/768, ttl=64 (reply in 17)
17	17...	172.16.51.1	172.16.50.1	ICMP	98 Echo (ping) reply	id=0x55cf, seq=3/768, ttl=63 (request in 16)
18	18...	CiscoInc_3a:f6:...	Spanning-tree-(...	STP	60 Conf. Root = 32768/50/fc:fb:fb:3a:f6:00	Cost = 0 Port = 0x8006
19	18...	172.16.50.1	172.16.51.1	ICMP	98 Echo (ping) request	id=0x55cf, seq=4/1024, ttl=64 (reply in 20)
20	18...	172.16.51.1	172.16.50.1	ICMP	98 Echo (ping) reply	id=0x55cf, seq=4/1024, ttl=63 (request in 19)
21	19...	CiscoInc_3a:f6:...	CDP/VTP/DTP/PAG...	CDP	453 Device ID: tux-sw5 Port ID: FastEthernet0/4	
22	19...	172.16.50.1	172.16.51.1	ICMP	98 Echo (ping) request	id=0x55cf, seq=5/1280, ttl=64 (reply in 23)
23	19...	172.16.51.1	172.16.50.1	ICMP	98 Echo (ping) reply	id=0x55cf, seq=5/1280, ttl=63 (request in 22)

Figura 6

16	14...	CiscoInc_3a:f6:...	Spanning-tree-(...	STP	60 Conf. Root = 32768/51/fc:fb:fb:3a:f6:00	Cost = 0 Port = 0x8008
17	14...	172.16.50.1	172.16.51.1	ICMP	98 Echo (ping) request	id=0x55cf, seq=2/512, ttl=63 (reply in 18)
18	14...	172.16.51.1	172.16.50.1	ICMP	98 Echo (ping) reply	id=0x55cf, seq=2/512, ttl=64 (request in 17)
19	15...	172.16.50.1	172.16.51.1	ICMP	98 Echo (ping) request	id=0x55cf, seq=3/768, ttl=63 (reply in 20)
20	15...	172.16.51.1	172.16.50.1	ICMP	98 Echo (ping) reply	id=0x55cf, seq=3/768, ttl=64 (request in 19)
21	16...	CiscoInc_3a:f6:...	Spanning-tree-(...	STP	60 Conf. Root = 32768/51/fc:fb:fb:3a:f6:00	Cost = 0 Port = 0x8008
22	16...	172.16.50.1	172.16.51.1	ICMP	98 Echo (ping) request	id=0x55cf, seq=4/1024, ttl=63 (reply in 23)
23	16...	172.16.51.1	172.16.50.1	ICMP	98 Echo (ping) reply	id=0x55cf, seq=4/1024, ttl=64 (request in 22)
24	17...	172.16.50.1	172.16.51.1	ICMP	98 Echo (ping) request	id=0x55cf, seq=5/1280, ttl=63 (reply in 25)

Figura 7

## Anexo B5

10 14..	172.16.51.1	172.16.1.1	DNS	73 Standard query 0x2ec1 A www.google.pt
11 14..	172.16.1.1	172.16.51.1	DNS	235 Standard query response 0x2ec1 A www.google.pt A 216.58.211.195 NS ns3.google.com NS ns4.google.com NS ns2.google.com NS ns1.google...
12 14..	172.16.51.1	216.58.211.195	ICMP	98 Echo (ping) request id=0x1db2, seq=1/256, ttl=64 (reply in 13)
13 14..	216.58.211.195	172.16.51.1	ICMP	98 Echo (ping) reply id=0x1db2, seq=1/256, ttl=53 (request in 12)
14 14..	172.16.51.1	172.16.1.1	DNS	87 Standard query 0xcdb4 PTR 195.211.58.216.in-addr.arpa
15 14..	172.16.1.1	172.16.51.1	DNS	382 Standard query response 0xcdb4 PTR 195.211.58.216.in-addr.arpa PTR mad01s25-in-f3.1e100.net PTR mad01s25-in-f195.1e100.net NS ns2.go...
16 15..	172.16.51.1	216.58.211.195	ICMP	98 Echo (ping) request id=0x1db2, seq=2/512, ttl=64 (reply in 17)
17 15..	216.58.211.195	172.16.51.1	ICMP	98 Echo (ping) reply id=0x1db2, seq=2/512, ttl=53 (request in 16)
18 16..	CiscoInc_3a:f6:...	Spanning-tree-...	STP	60 Conf. Root = 32768/51/fc:fb:fb:3a:f6:00 Cost = 0 Port = 0x8004
19 16..	172.16.51.1	216.58.211.195	ICMP	98 Echo (ping) request id=0x1db2, seq=3/768, ttl=64 (reply in 20)
20 16..	216.58.211.195	172.16.51.1	ICMP	98 Echo (ping) reply id=0x1db2, seq=3/768, ttl=53 (request in 19)
21 17..	172.16.51.1	216.58.211.195	ICMP	98 Echo (ping) request id=0x1db2, seq=4/1024, ttl=64 (reply in 22)
22 17..	216.58.211.195	172.16.51.1	ICMP	98 Echo (ping) reply id=0x1db2, seq=4/1024, ttl=53 (request in 21)
23 18..	CiscoInc_3a:f6:...	Spanning-tree-...	STP	60 Conf. Root = 32768/51/fc:fb:fb:3a:f6:00 Cost = 0 Port = 0x8004
24 18..	CiscoInc_3a:f6:...	CiscoInc_3a:f6:...	LOOP	60 Reply
25 18..	172.16.51.1	216.58.211.195	ICMP	98 Echo (ping) request id=0x1db2, seq=5/1280, ttl=64 (reply in 26)
26 18..	216.58.211.195	172.16.51.1	ICMP	98 Echo (ping) reply id=0x1db2, seq=5/1280, ttl=53 (request in 25)
27 19..	172.16.51.1	216.58.211.195	ICMP	98 Echo (ping) request id=0x1db2, seq=6/1536, ttl=64 (reply in 28)
28 19..	216.58.211.195	172.16.51.1	ICMP	98 Echo (ping) reply id=0x1db2, seq=6/1536, ttl=53 (request in 27)
29 20..	CiscoInc_3a:f6:...	Spanning-tree-...	STP	60 Conf. Root = 32768/51/fc:fb:fb:3a:f6:00 Cost = 0 Port = 0x8004

Figura 8

## Anexo B6

5 5...	172.16.50.1	172.16.1.1	DNS	69 Standard query 0xa433 A ftp.up.pt
6 5...	172.16.50.1	172.16.1.1	DNS	69 Standard query 0xc584 AAAA ftp.up.pt
7 5...	172.16.1.1	172.16.50.1	DNS	274 Standard query response 0xa433 A ftp.up.pt A 193.136.37.8 NS dns4.up.pt NS dns3.up.pt NS dns1.up.pt A 193.137.55.20 AAAA 2001:690:122...
8 5...	172.16.1.1	172.16.50.1	DNS	286 Standard query response 0xc584 AAAA ftp.up.pt AAAA 2001:690:2200:910::8 NS dns3.up.pt NS dns1.up.pt NS dns4.up.pt A 193.137.55.20 AA...
9 5...	172.16.50.1	193.136.37.8	TCP	74 53987 → 21 [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TSval=705467 TSecr=0 WS=128
10 5...	193.136.37.8	172.16.50.1	TCP	70 21 → 53987 [SYN, ACK] Seq=0 Ack=1 Win=5792 Len=0 MSS=1380 SACK_PERM=1 TSval=408291507 TSecr=705467
11 5...	172.16.50.1	193.136.37.8	TCP	66 53987 → 21 [ACK] Seq=1 Ack=1 Win=29200 Len=0 TSval=705469 TSecr=408291507
12 5...	193.136.37.8	172.16.50.1	FTP	106 Response: 220 Bem-vindo \303\240 Universidade do Porto
13 5...	172.16.50.1	193.136.37.8	TCP	66 53987 → 21 [ACK] Seq=1 Ack=41 Win=29200 Len=0 TSval=705471 TSecr=408291510
14 5...	172.16.50.1	193.136.37.8	FTP	76 Request: USER ftp
15 5...	193.136.37.8	172.16.50.1	TCP	66 21 → 53987 [ACK] Seq=41 Ack=11 Win=5792 Len=0 TSval=408291511 TSecr=705471
16 5...	193.136.37.8	172.16.50.1	FTP	100 Response: 331 Please specify the password.
17 5...	172.16.50.1	193.136.37.8	FTP	77 Request: PASS pass
18 5...	193.136.37.8	172.16.50.1	FTP	89 Response: 230 Login successful.
19 5...	172.16.50.1	193.136.37.8	FTP	72 Request: passv
20 5...	193.136.37.8	172.16.50.1	FTP	115 Response: 227 Entering Passive Mode (193,136,37,8,160,66)
21 5...	172.16.50.1	193.136.37.8	FTP	97 Request: RETR pub/CPAN/RECENT-1M.json
22 5...	172.16.50.1	193.136.37.8	TCP	74 38422 → 41026 [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TSval=705474 TSecr=0 WS=128
23 5...	193.136.37.8	172.16.50.1	TCP	70 41026 → 38422 [SYN, ACK] Seq=0 Ack=1 Win=5792 Len=0 MSS=1380 SACK_PERM=1 TSval=408291514 TSecr=705474
24 5...	172.16.50.1	193.136.37.8	TCP	66 38422 → 41026 [ACK] Seq=1 Ack=1 Win=29200 Len=0 TSval=705475 TSecr=408291514

Figura 9

22 5...	172.16.50.1	193.136.37.8	TCP	74 38422 → 41026 [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TSval=705474 TSecr=0 WS=128
23 5...	193.136.37.8	172.16.50.1	TCP	70 41026 → 38422 [SYN, ACK] Seq=0 Ack=1 Win=5792 Len=0 MSS=1380 SACK_PERM=1 TSval=408291514 TSecr=705474
24 5...	172.16.50.1	193.136.37.8	TCP	66 38422 → 41026 [ACK] Seq=1 Ack=1 Win=29200 Len=0 TSval=705475 TSecr=408291514
25 5...	193.136.37.8	172.16.50.1	FTP..	14.. FTP Data: 1368 bytes
26 5...	172.16.50.1	193.136.37.8	TCP	66 38422 → 41026 [ACK] Seq=1 Ack=1369 Win=31464 Len=0 TSval=705476 TSecr=408291515
27 5...	193.136.37.8	172.16.50.1	FTP	152 Response: 150 Opening BINARY mode data connection for pub/CPAN/RECENT-1M.json (2180058 bytes).
28 5...	193.136.37.8	172.16.50.1	FTP..	28.. FTP Data: 2736 bytes
29 5...	172.16.50.1	193.136.37.8	TCP	66 38422 → 41026 [ACK] Seq=1 Ack=4105 Win=36936 Len=0 TSval=705476 TSecr=408291515
30 5...	193.136.37.8	172.16.50.1	FTP..	14.. FTP Data: 1368 bytes
31 5...	172.16.50.1	193.136.37.8	TCP	66 38422 → 41026 [ACK] Seq=1 Ack=5473 Win=39672 Len=0 TSval=705476 TSecr=408291515
32 5...	193.136.37.8	172.16.50.1	FTP..	28.. FTP Data: 2736 bytes
33 5...	172.16.50.1	193.136.37.8	TCP	66 38422 → 41026 [ACK] Seq=1 Ack=8209 Win=45144 Len=0 TSval=705477 TSecr=408291515
34 5...	193.136.37.8	172.16.50.1	FTP..	28.. FTP Data: 2736 bytes
35 5...	172.16.50.1	193.136.37.8	TCP	66 38422 → 41026 [ACK] Seq=1 Ack=10945 Win=50616 Len=0 TSval=705477 TSecr=408291515
36 5...	193.136.37.8	172.16.50.1	FTP..	28.. FTP Data: 2736 bytes
37 5...	172.16.50.1	193.136.37.8	TCP	66 38422 → 41026 [ACK] Seq=1 Ack=13681 Win=56088 Len=0 TSval=705477 TSecr=408291516
38 5...	193.136.37.8	172.16.50.1	FTP..	14.. FTP Data: 1368 bytes
39 5...	172.16.50.1	193.136.37.8	TCP	66 38422 → 41026 [ACK] Seq=1 Ack=15049 Win=58824 Len=0 TSval=705478 TSecr=408291516
40 5...	193.136.37.8	172.16.50.1	FTP..	28.. FTP Data: 2736 bytes
41 5...	172.16.50.1	193.136.37.8	TCP	66 38422 → 41026 [ACK] Seq=1 Ack=17785 Win=64296 Len=0 TSval=705478 TSecr=408291516
42 5...	193.136.37.8	172.16.50.1	FTP..	28.. FTP Data: 2736 bytes

Figura 10

39 5...	172.16.50.1	193.136.37.8	TCP	66 38422 → 41026 [ACK] Seq=1 Ack=15049 Win=58824 Len=0 TSval=705478 TSecr=408291516
40 5...	193.136.37.8	172.16.50.1	FTP..	28.. FTP Data: 2736 bytes

Figura 11

## Anexo C

### Experiência 1 – Configuração de um IP de rede

(Switch)

configure terminal

vlan 50

end

configure terminal

interface fastethernet 0/1

switchport mode access

switchport access vlan 50

end

configure terminal

interface fastethernet 0/4

switchport mode access

```
switchport access vlan 50
end
```

```
(Computador 1)
ifconfig eth0 down
ifconfig eth0 up
ifconfig eth0 172.16.50.1/24
```

```
(Computador 4)
ifconfig eth0 up
ifconfig eth0 down
ifconfig eth0 172.16.50.254/24
```

### **Experiência 2 – Implementação de duas virtual LANs num switch**

```
configure terminal
vlan 51
end
```

```
configure terminal
interface fastethernet 0/2
switchport mode access
switchport access vlan 51
end
```

```
(Computador 2)
ifconfig eth0 up
ifconfig eth0 down
ifconfig eth0 172.16.51.1/24
```

### **Experiência 3 – Configuração de um router em Linux**

```
configure terminal
interface fastethernet 0/6
switchport mode access
switchport access vlan 51
end
```

```
Computador 1:
route add -net 172.16.51.0/24 gw 172.16.50.254
route add -net default gw 172.16.50.254
Computador 2:
route add -net 172.16.50.0/24 gw 172.16.51.253
```

### **Experiência 4 – Configuração de um router comercial e implementação de NAT**

```
(Switch)
configure terminal
interface gigabitethernet 0/10
switchport mode access
switchport access vlan 51
end
```

```
(Router)
```

```

conf t
interface gigabitethernet 0/0
ip address 172.16.51.254 255.255.255.0
no shutdown
ip nat inside
exit
interface gigabitethernet 0/1
ip address 172.16.1.59 255.255.255.0
no shutdown
ip nat outside
exit
ip nat pool ovrl 172.16.1.59 172.16.1.59 prefix 24
ip nat inside source list 1 pool ovrl overload
access-list 1 permit 172.16.10.0 0.0.0.255
access-list 1 permit 172.16.11.0 0.0.0.255
ip route 0.0.0.0 0.0.0.0 172.16.1.254
ip route 172.16.50.0 255.255.255.0 172.16.51.253
end

```

Computador 1:

```
route add -net default gw 172.16.50.254
```

Computador 2:

```
route add -net default gw 172.16.51.254
```

Computador 4:

```
route add -net default gw 172.16.51.254
```

### **Experiência 5 – DNS**

(Em cada computador)

```
vi /etc/resolv.conf
```

```
search netlab.fe.up.pt
```

```
nameserver 172.16.1.1
```

### **Experiência 7 – Implementação de NAT em Linux**

(No computador 4)

```
/sbin/iptables -t nat -A POSTROUTING -o eth1 -j MASQUERADE
```

```
/sbin/iptables -A FORWARD -i eth0 -o eth0 -m state
```

```
--state RELATED,ESTABLISHED -j ACCEPT
```

```
/sbin/iptables -A FORWARD -i eth0 -o eth1 -j ACCEPT
```

### **Anexo D - Scripts**

tuxPermission.sh

```
#!/bin/bash
```

```
chmod -x tux51.sh
```

```
chmod -x tux52.sh
```

```
chmod -x tux54.sh
```

tux51.sh

```
#!/bin/bash
```

```
/etc/init.d/networking restart
```

```
ifconfig eth0 up
```

```
ifconfig eth0 172.16.50.1/24
```

```
route add default gw 172.16.50.254
```

```
route -n
```

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

#### tux52.sh

```
#!/bin/bash
/etc/init.d/networking restart
ifconfig eth0 up
ifconfig eth0 172.16.51.1/24
route add -net 172.16.50.0/24 gw 172.16.51.253
route add default gw 172.16.51.254
route -n
echo 1 > /proc/sys/net/ipv4/ip_forward
echo 0 > /proc/sys/net/ipv4/icmp_echo_ignore_broadcasts
```

#### tux54.sh

```
#!/bin/bash
/etc/init.d/networking restart
ifconfig eth0 up
ifconfig eth0 172.16.50.254/24
ifconfig eth1 up
ifconfig eth1 172.16.51.253/24
route add default gw 172.16.51.254
route -n
echo 1 > /proc/sys/net/ipv4/ip_forward
echo 0 > /proc/sys/net/ipv4/icmp_echo_ignore_broadcasts
```