2º Trabalho Laboratorial

Redes de Computadores

Licenciatura em Engenharia Informática e Computação
1º Semestre - 2023/2024

Redes de Computadores - Turma 7

João Filipe de Menezes Falcão e Sousa Guedes (up202108711@up.pt)

Eduardo Afonso Soares Ferreira Machado (up202105337@up.pt)

Porto, 15 de Dezembro de 2023

Indíce

Sumário

Este projeto foi realizado no âmbito da unidade curricular de Redes de Computadores com o objetivo de desenvolvimento de uma aplicação de *download* de ficheiros usando o protocolo FTP (File Transfer Protocol). Permitindo também a aprendizagem sobre conceitos como: TCP/IP, RFCs, servidores DNS, entre outros. Graças a este projeto conseguimos aplicar a matéria dada em relação ao protocolo FTP e a utilização de uma rede de computadores.

Introdução

O projeto foi desenvolvido com a realização de seis experiências e a aplicação de download que usa o protocolo FTP e configuração de uma rede de computadores. O relatório está dividido nas seguintes partes:

- Parte 1 - Aplicação de Download
    - Arquitetura
    - Resultados
- Parte 2 - Configuração e Análise da Rede
    - Para cada experiência (1 a 6)
        - Arquitetura de rede, objetivos da experiência, principais comandos de configuração, logs relevantes;
        - Análise dos logs capturados que são relevantes para os objetivos de aprendizagem;
- Conclusões

Parte 1 - Aplicação de Download

Arquitetura

Esta aplicação foi desenhada com o objetivo de realizar um download de um ficheiro alocado num servidor utilizando um protocolo de transferencia de ficheiros (FTP). O programa está dividido em várias funções, cada uma responsável por uma parte específica do protocolo FTP ou da operação do programa.

A função *main()* é o ponto de entrada do programa. Esta função analisa os argumentos introduzidos pelo utilizador na linha de comando e inicializa a estrutura de URL. Para além dos elementos préviamente referidos o papel principal desta função é assegurar o bom funcionamento do programa em geral, uma vez que se encarrega de chamar as outras funções na ordem correta para estabelecer uma conexão com o servidor FTP, autenticar, mudar para o modo passivo, solicitar/descarregar o arquivo e, finalmente, encerrar a conexão.

A função *parseURL()* é responsável por analisar o URL. Para tal, a função utiliza expressões regulares para verificar o formato do URL e, em seguida, extrai diversos dados sendo eles: o host (nome do servidor), user, password, caminho do recurso do URL, e o ficheiro (nome e extensão). Par além da extração de dados, a função *parseURL()* também encarrega-se da tradução do host para um endereço IP—através do serviço DNS.

A função *establishSocketConnection()* cria um socket TCP, conecta-o ao endereço IP e a porta. Utiliza as chamadas de sistema *socket()* e *connect()* para o fazer. Se ocorrer um erro durante esse processo, imprime uma mensagem de erro e encerra o programa. A função *authenticateConnection()* envia os comandos *user* e *pass* para autenticar-se no servidor. Escreve esses comandos no socket da conexão ao servidor e, em seguida, retorna a resposta do mesmo.

A função *passiveMode()* envia o comando FTP PASV para mudar para o modo passivo. No modo passivo, o cliente inicia todas as conexões, o que pode ajudar a evitar problemas com firewalls e NAT. A resposta do servidor ao comando PASV inclui o endereço IP e a porta aos quais o cliente deve conectar-se para a transferência do ficheiro.

A função *readSocketResponse()* foi criada para ler a resposta do servidor. Ela lê a resposta byte a byte e guarda-a em um buffer. A função *sendFileRequest()* envia o comando FTP RETR para solicitar o ficheiro. A resposta do servidor a este comando inclui o tamanho e o início da transferência do arquivo. A função *transferFileFromServer()* lê os dados do ficheiro no servidor e escreve-os em um ficheiro local. Esta função só termina uma vez que todos os dados do servidor tenham sido transferidos.

Finalmente, a função *closeSocketConnection()* envia o comando FTP QUIT para fechar a conexão e, em seguida, fecha o socket.

Resultados

Testamos o funcionamento da nossa aplicação *download* ao realizar o download de vários ficheiros de diferentes tamanhos. Testamos o input de URLs errados, URLs com user e password ou anónimos. Em caso de erro, o programa lança na consola uma mensagem ligada ao erro ocorrido tal como termina o programa para ser realizado outra vez de forma correta.

Parte 2 - Configuração e Análise da Rede

Exp 1 - Configuração de uma Rede IP

O objetivo desta experiência foi a configuração de dois endereços IP de dois computadores distintos, tux3 e tux4, ligados a um switch permitindo que eles comunicassem entre eles. Para esta experiência foram utilizados os comandos *ifconfig* para configurar o IP em cada computador, tux53 (IP - 172.16.50.1/24, endereço MAC - 00:22:64:19:09:5c) e tux54 (IP - 172.16.50.254/24, endereço MAC - 00:21:5a:61:2c:54).

Ao executar o comando *ping* em cada um dos computadores, podemos observar criação de pacotes ARP (Address Resolution Protocol), que estabelecem a ligação entre o IP e o endereço fisíco, o endereço MAC (Media Access Control), do computador. Os pings enviam o IP do computador destino e do computador origem no mesmo pacote, e, como resposta, o computador destino envia um pacote ARP com o seu endereço MAC, permitindo assim a comunicação entres os dois; esta conexão entre IP e MAC é guardada na tabela ARP. O processo também gera pacotes ICMP (Internet Control Message Protocol), permitindo o envio de mensagems de controlo e erro entre os computadores.

Podemos distinguir que tipo de pacote é enviado sobre a trama receptora Ethernet nas capturas no Wireshark na coluna Protocol. Esta distinção sobre pacotes também está presente nos dois primeiros bytes do cabeçalho da frame Ethernet (0x0800 para IPs/ICMPs e 0x0806 para ARPs). Também é possível observar o tamanho da trama na coluna Length.

A interface de “loopback” é uma interface numa rede de computadores que permite a comunicação interna (ou seja transimissão e receção de dados) sem que seja necessário o envolvimento de qualquer tipo de hardware fisico. O “loopback” facilita o processo de teste e debugging uma vez que permite verificar se as ligacoes de rede estão devidamente configuradas.

A captura do log e configuração da experiência estão disponíveis nos anexos.

Exp 2 - Implementar duas pontes num switch

O objetivo desta experiência foi a implementação de duas pontes ligadas ao switch, uma com tux3 e tux4, e outra com apenas o tux2.

Para esta experiência, repetimos o processo de configurar os IPs do tux3 e 4. No tux2, configuramos um novo IP (172.16.51.1). Com o comando */interface bridge port remove [find interface=etherX]* removemos as portas as quais os tux2/3/4 estão ligadas por default. A seguir, adicionamos novas portas, */interface bridge port add bridge=bridgeX0/1 interface=etherY.* Verificamos as portas criadas com */interface bridge port print*. No tux3, damos ping ao tux4 e 2. No tux2, damos ping ao tux3 e 4.

Concluímos que existem dois domínios de broadcast correspondentes as bridges criadas, uma vez que ao correr o comando ping -b (broadcast) a partir do tux3, o tux 4 era alcançado, no entanto o tux2 é unreachable. Ao realizar a experiencia no tux2, nem o tux3 nem o tux4 são alcancaveis. Só existe conexão entre tux3 e 4, o tux 2 está numa ponte diferente.

A captura do log e configuração da experiência estão disponíveis nos anexos.

Exp 3 - Configurar um Router no Linux

O objetivo desta experiência é a configuração de um router que permite a comunicação entre dois computadores em redes distintas. O tux4 estará ligado a duas redes, eth0 com o tux3 e em eth1 com o tux2, permitindo então a comunicação previamente não disponível entre tux2 e 3.

Com o tux4 conectado à eth1, configuramos o seu IP. Eliminamos as portas às quais o tux2 estava ligado por default e adicionamos uma nova porta, */interface bridge port remove [find interface=etherY]* e */interface bridge port add bridge=bridgeX1 interface=etherY.* Ativamos o IP forwarding e desativamos o ICMP (*echo 1 > /proc/sys/net/ipv4/ip_forward* e *echo 0 > /proc/sys/net/ipv4/icmp_echo_ignore_broadcasts*). Adicionamos as seguintes rotas: no tux2, *route add -net 172.16.Y0.0/24 gw 172.16.Y1.253*; no tux3, *route add -net 172.16.Y1.0/24 gw 172.16.Y0.253*, agora estes conseguem alcançar um ao outro através do tux4. Finalizamos a dar os pings necessário no tux3, limpamos as tabelas ARPs em todos os tuxs e damos ping ao tux2 no tux3.

Durante a experiência verificamos a existencia de duas routes, uma na tux2 e outra na tux3, ambas têm o tux4 como gateway pois é comum às duas bridges criadas. Na tabela de forwarding também encontramos informação sobre a *destination* (IP destino), *genmask* (netmask da rede destino), *gateway* (IP para qual a mensagem vai ser enviada), *interface*

(rede, eth0/1), *metric* (custo da rota), *flags* (informação sobre a rota), *ref* (numero de referencia para a rota) e *use* (contador do numero de consultas à rota).

Como visto anteriormente, a ARP liga o MAC address ao IP, mas ao dar ping ao tux2 no tux3, ele não recebe o endereço MAC do tux2, o endereço observado é do tux4 (o gateway) e não do computador destino, pois é com ele que tem ligação. O tux3 não consegue comunicar diretamente com o tux2.

Observamos pacotes ICMP Request e Reply, que contém o IP origem (tux3) e o IP destino (tux2) — o que demonstra que a rede foi bem configurada. No entanto, o tux2 e 3 estão em redes diferentes, portanto são enviados dois pacotes, um com o IP e endereço MAC do tux3 e 4 e outro com o IP e endereço MAC do tux4 e 2, ligando as duas bridges.

A captura dos logs e configuração da experiência estão disponíveis nos anexos.

Exp 4 - Configurar um Router Comercial e Implementar NAT

O objetivo desta experiência foi configurar um router comercial com NAT implementado, de modo a permitir acesso à rede externa.

Para esta experiência mantemos a rede configurada na experiência 3. Temos duas bridges : numa temos tux3 e 4; noutra temos tux4 e 2, com o tux4 a conectar os tux3 e 2. Começamos por conectar o eth1 do router à porta PY.1 e o eth2 do router à porta da ponte Y1 no Switch. Eliminamos as portas default do etherY do switch e ligamos o etherY à ponteY1: */interface bridge port remove [find interface=etherY]* e */interface bridge port add bridge=bridgeY1 interface=etherY*. Para as configurações do router, configuramos os IP pela consola do router no GTKterm no tux2: */ip address add address=172.16.1.Y9/25 interface=ether1* e */ip address add address=172.16.Y1.254/24 interface=ether2.* Adicionamos o router como gateway default nos tuxs: *route add default gw 172.16.XY.254*. Por fim, adicionamos rotas default no router: */ip route add dst_address=172.16.Y0.0/24 gateway=172.16.Y1.253* e */ip route add dst_address=0.0.0.0/0 gateway=172.16.1.254*.

Para podemos configurar uma rota estática num router comercial, como na experiência, temos de dar reset as suas configurações, adicioná-lo à ponte correspondente e atribuir um IP interno e externo.

Os caminhos seguidos pelos pacotes na experiência foram reencaminhados através do router implementado até ao endereço IP de destino, isto acontece porque não temos ligação do tux2 ao tux4. Mas depois não houve qualquer reencaminhamento porque a ligação mais curta da rede estava disponível.

Para configurar a NAT num router comercial, corremos o comando /ip firewall nat enable 0 na consola do router. O NAT (Network Address Translation) permite que vários dispositivos em uma rede local compartilhem um único endereço IP público para aceder à Internet. Ele traduz os endereços IP privados internos para um único endereço IP público, e vice-versa, otimizando o uso de endereços IP e oferecendo uma camada de segurança adicional ao ocultar os endereços internos da Internet. Permite reduzir o número de endereços publicos utilizados.

A captura dos logs e configuração da experiência estão disponíveis nos anexos.

Exp 5 - DNS

O objetivo desta experiência é configurar o DNS (Domain Name Server) para a tradução do nome dos domínios dos sites para endereços IP.

Para esta experiência mantivemos a configuração da rede efetuada ao longo das quatro experiências anteriores. Começamos por introduzir em todos os tuxs a linha '*nameserver 172.16.1.1*' ao ficheiro */etc/resolv.conf.* E, em todos os tuxs, demos ping ao google.com. Desta forma, configuramos os serviços DNS no host: adicionando o "namerserver <IP>" no ficheiro /etc/resolv.conf. E esperamos que o hostname nos pacotes enviados pelo DNS seja identificado e traduzido para um IP.

A captura dos logs e configuração da experiência estão disponíveis nos anexos.

Exp 6 - Conexões TCP

O objetivo desta experiência foi testar o funcionamento da rede configurada ao longo destas experiências, e testar o funcionamento da nossa aplicação de protocolo FTP.

Para avaliar a nossa aplicação, no tux3, fazemos download de um ficheiro do servidor FTP. E depois, no tux2 e 3, fazemos download de dois ficheiros do servidor FTP nos respetivos tuxs; um ficheiro de maior tamanho no tux3, e um ficheiro de tamanho inferior no tux2 para que seja feito download a meio do download do ficheiro no tux3.

Na nossa aplicação FTP, foram efetuadas duas conexões TCP (Transmission Control Protocol), uma para enviar comandos de controlo e recepção de mensagens do servidor, onde é transportada a informação de controlo FTP, e outra para receber o ficheiro.

A conexão TCP tem três fases: conexão, transeferência de dados e fim da conexão.

O ARQ (Automatic Repeat reQuest) é um método de controlo de erros para transmissão de dados que usa ACK (acknowledgements) e timeout para obter uma transmissão de dados confiável através de um canal de comunicação não confiável. Se não for recebido um ACK antes do timeout, ele reenviará o pacote até receber uma confirmação ou exceder um número predefinido de retransmissões. Os campos importantes no TCP são: o sequence number, ACK numbers, window size e flags.

O TCP tem um mecanismo de controlo de congestão que controla o fluxo de dados, que usa um algoritmo que inclui vários aspectos tais como um esquema de additive increase, na transferência seguinte envia mais um pacote que na anterior; juntamente com outros esquemas, como o slow start, que começa com um tamanho de janela de congestão de 1, 2, 4 ou 10 MSS (maximum segment size), depois o valor do tamanho da janela pode ser aumentado em 1 MSS com cada ACK recebida, efetivamente duplicando o tamanho da janela a cada round-trip, para evitar redes congestionadas (perda de pacotes).

Depois de efetuar o download, podemos observar o impacto do TCP no gráfico obtido. Observamos que após a taxa de tranferência aumentar ao longo dum período de tempo, segue-se uma descida no número de transferências que remonta a uma taxa de transferência inferior ao número original no intervalo de tempo anterior.

Na segunda parte da experiência, durante o download de um ficheiro de maior tamanho no tux3, realizou-se outro download de um ficheiro mais pequeno no tux2. Observa-se que quando este último foi inicializado, houve uma queda de velocidade no download no tux3. Isto deve-se ao TCP.

A captura dos logs da experiência estão disponíveis nos anexos.

Conclusões

Ao concluir este projeto, o desenvolvimento de uma aplicação de transferência de ficheiros utilizando o protocolo FTP, foi-nos possível entender os conceitos necessários, e que nos eram pedidos no guião, para compreender como funcionam e como são postos em prática os protocolos que se encontram numa rede de computadores para a transferência de dados entre cada máquina, tal como conhecer como se configura uma rede de computadores.

Referências

1. https://en.wikipedia.org/wiki/TCP_congestion_control#Slow_start
2. https://en.wikipedia.org/wiki/Automatic_repeat_request
3. https://beej.us/guide/bgnet/pdf/bgnet_a4_c_2.pdf

Anexos

Guião

Experiência 1

Tux 3/4 :
ifconfig eth0 up
ifconfig eth0 <IP>
- 172.16.Y0.1/24 para o Tux3
- 172.16.Y0.254/24 para o Tux4

Tux3:
ping 172.16.Y0.254

Tux4:
ping 172.16.Y0.1

Tux 3:
arp -a
arp -d 172.16.Y0.254/24
arp -a

Experiência 2

Switch console:
/system reset configuration

tux2:
ifconfig eth0 up
ifconfig eth0 172.16.Y1.1/24

Switch console:
> /interface bridge add name=bridgeY0
> /interface bridge add name=bridgeY1
> /interface bridge port remove [find interface=ether1]
> /interface bridge port remove [find interface=ether2]
> /interface bridge port remove [find interface=ether3]
> /interface bridge port add bridge=bridgeY0 interface=ether1
> /interface bridge port add bridge=bridgeY0 interface=ether2
> /interface bridge port add bridge=bridgeY1 interface=ether3
>/interface bridge port print

tux3:
ping 172.16.Y0.254
ping 172.16.Y1.1
ping -b 172.16.Y0.255

tux2:
ping -b 172.16.Y1.255

Experiência 3

tux4:
ifconfig eht1 up
ifconfig eht1 172.16.Y1.253/24

Switch console:
/interface bridge port remove [find interface=ether4]
/interface bridge port add bridge=bridgeY1 interface=ether4

tux4:
echo 1 > /proc/sys/net/ipv4/ip_forward
echo 0 > /proc/sys/net/ipv4/icmp_echo_ignore_broadcasts

tux2:
route add -net 172.16.Y0.0/24 gw 172.16.Y1.253

tux3:
route add -net 172.16.Y1.0/24 gw 172.16.Y0.254

tux3:
ping 172.16.Y0.254
ping 172.16.Y1.253

ping 172.16.Y1.1

tux2:
arp -d 172.16.Y1.253

tux3:
arp-d 172.16.T0.254

tux4:
arp -d 172.16.Y0.1
arp -d 172.16.Y1.1

Experiencia 4

Switch console:
/interface bridge port remove [find interface=ether5]
/interface bridge port add bridge=bridgeY1 interface=ether5

Router console:
/system reset-configuration
/ip address add address=172.16.1.Y9/24 interface=ether1
/ip address add address=172.16.Y1.254/24 interface=ether2
/ip route add dst-address=172.16.Y0.0/24 gateway=172.16.Y1.253
/ip route add dst-address=0.0.0.0/0 gateway=172.16.1.254

tux2:
route add default gw 172.16.Y1.254

tux3:
route add default gw 172.16.Y0.254

tux4:
route add default gw 172.16.Y1.254

tux3:
ping 172.16.Y0.254
ping 172.16.Y1.1
ping 172.16.Y1.254

tux2:
echo 0 > /proc/sys/net/ipv4/conf/eth0/accept_redirects
echo 0 > /proc/sys/net/ipv4/conf/all/accept_redirects
route del -net 172.16.Y0.0 gw 172.16.Y1.253 netmask 255.255.255.0
ping 172.16.Y0.1
traceroute -n 172.16.Y0.1
echo 1 > /proc/sys/net/ipv4/conf/eth0/accept_redirects

echo 1 > /proc/sys/net/ipv4/conf/all/accept_redirects

tux3:
ping 172.16.1.254

Router console:
/ip firewall nat disable 0

tux3:
ping 172.16.1.254

Router console:
/ip firewall nat disable 0

Experiência 5

tux 2,3,4:
colocar a seguinte linha ao ficheiro /etc/resolv.conf -> nameserver 172.16.1.1
ping google.com

Código

download.c

```
#include "download.h"

int parseURL(char *inputURL, struct URL *url)
{
   struct hostent *h;

   regex_t regex;
   regcomp(&regex, "/", 0);

   if (regexec(&regex, inputURL, 0, NULL, 0))
   {
       return -1;
   }

   regcomp(&regex, "@", 0);

   if (regexec(&regex, inputURL, 0, NULL, 0) != 0)
```

```
    { // ftp://<host>/<url-path>

        sscanf(inputURL, "%*[^/]//%[^/]", url->host);
        strcpy(url->user, "anonymous");
        strcpy(url->password, "password");
    }

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

        sscanf(inputURL, "%*[^/]//%*[^@]@%[^/]", url->host);
        sscanf(inputURL, "%*[^/]//%[^:/]", url->user);
        sscanf(inputURL, "%*[^/]//%*[^:]:%[^@\n$]", url->password);
    }

    sscanf(inputURL, "%*[^/]//%*[^/]/%s", url->resource);
    strcpy(url->file, strrchr(inputURL, '/') + 1);

    if (strlen(url->host) == 0)
    {
        return -1;
    }

    if ((h = gethostbyname(url->host)) == NULL)
    {
        printf("Invalid hostname '%s'\n", url->host);
        exit(-1);
    }

    strcpy(url->ip, inet_ntoa(*((struct in_addr *)h->h_addr)));

    return !(strlen(url->host) && strlen(url->user) &&
strlen(url->password) && strlen(url->resource) && strlen(url->file));
}

int establishSocketConnection(char *ip, int port)
{

    int socket_fd;
```

```
    struct sockaddr_in server_addr;

    bzero((char *)&server_addr, sizeof(server_addr));

    server_addr.sin_family = AF_INET;
    server_addr.sin_addr.s_addr = inet_addr(ip);
    server_addr.sin_port = htons(port);

    if ((socket_fd = socket(AF_INET, SOCK_STREAM, 0)) < 0)
    {
        perror("Error in socket()");
        exit(-1);
    }
    if (connect(socket_fd, (struct sockaddr *)&server_addr,
sizeof(server_addr)) < 0)
    {
        perror("Error in connect()");
        exit(-1);
    }

    return socket_fd;
}

int authenticateConnection(const int fd, const char *user, const char
*pass)
{

    char userCommand[5 + strlen(user) + 1];
    char passCommand[5 + strlen(pass) + 1];
    char answer[500];

    sprintf(userCommand, "user %s\n", user);
    sprintf(passCommand, "pass %s\n", pass);

    write(fd, userCommand, strlen(userCommand));

    if (readSocketResponse(fd, answer) != 331)
    {
        printf("Error in username: '%s'. Operation aborted.\n", user);
```

```
        exit(-1);
    }

    write(fd, passCommand, strlen(passCommand));

    return readSocketResponse(fd, answer);
}

int passiveMode(const int fd, char *ip, int *port)
{
    char answer[500];
    int ip1;
    int ip2;
    int ip3;
    int ip4;
    int port1;
    int port2;

    write(fd, "pasv\n", 5);

    if (readSocketResponse(fd, answer) != 227)
    {
        return -1;
    }

    sscanf(answer, "%*[^(](%d,%d,%d,%d,%d,%d)%*[^\n$)]", &ip1, &ip2,
&ip3, &ip4, &port1, &port2);

    *port = port1 * 256 + port2;

    sprintf(ip, "%d.%d.%d.%d", ip1, ip2, ip3, ip4);

    return 227;
}

int readSocketResponse(const int fd, char *buffer)
{
    char byte;
```

```
int index = 0;
int status_code;

responseStatus state = START;
memset(buffer, 0, 500);

while (state != END)
{

    read(fd, &byte, 1);
    switch (state)
    {
    case START:
        if (byte == ' ')
        {
            state = SINGLE;
        }
        else if (byte == '-')
        {
            state = MULTIPLE;
        }
        else if (byte == '\n')
        {
            state = END;
        }
        else
        {
            buffer[index++] = byte;
        }
        break;
    case SINGLE:
        if (byte == '\n')
        {
            state = END;
        }
        else
        {
            buffer[index++] = byte;
        }
```

```
            break;
        case MULTIPLE:
            if (byte == '\n')
            {
                memset(buffer, 0, 500);
                state = START;
                index = 0;
            }
            else
            {
                buffer[index++] = byte;
            }
            break;
        case END:
            break;
        default:
            break;
        }
    }

    sscanf(buffer, "%d", &status_code);

    return status_code;
}

int sendFileRequest(const int fd, char *resource)
{
    char fileCommand[5 + strlen(resource) + 1], answer[500];
    sprintf(fileCommand, "retr %s\n", resource);
    write(fd, fileCommand, sizeof(fileCommand));

    return readSocketResponse(fd, answer);
}

int transferFileFromServer(const int fd1, const int fd2, char
*filename)
{
    FILE *fd = fopen(filename, "wb");
    char buffer[500];
```

```
    int bytes;

    if (fd == NULL)
    {
        printf("Error opening or creating file '%s'\n", filename);
        exit(-1);
    }

    do
    {
        bytes = read(fd2, buffer, 500);
        if (fwrite(buffer, bytes, 1, fd) < 0)
        {
            return -1;
        }
    } while (bytes);

    fclose(fd);

    return readSocketResponse(fd1, buffer);
}

int closeSocketConnection(const int fd1, const int fd2)
{
    char answer[500];
    write(fd1, "quit\n", 5);

    if (readSocketResponse(fd1, answer) != 221)
    {
        return -1;
    }

    return close(fd1) || close(fd2);
}

int main(int argc, char *argv[])
{
    struct URL url;
    char answer[500];
```

```
    int fd1;
    int fd2;
    int port;
    char ip[500];

    if (argc != 2)
    {
        printf("INVALID! Use the following format: ./download
ftp://[<user>:<password>@]<host>/<url-path>\n");
        exit(-1);
    }

    memset(&url, 0, sizeof(url));
    if (parseURL(argv[1], &url) != 0)
    {
        printf("Parse error. INVALID! Use the following format:
./download ftp://[<user>:<password>@]<host>/<url-path>\n");
        exit(-1);
    }

    printf("Host: %s\nResource: %s\nFile: %s\nUser: %s\nPassword: %s\nIP
Address: %s\n", url.host, url.resource, url.file, url.user,
url.password, url.ip);

    fd1 = establishSocketConnection(url.ip, 21);
    if (fd1 < 0 || readSocketResponse(fd1, answer) != 220)
    {
        printf("Failed to create socket for '%s' on port %d\n", url.ip,
21);
        exit(-1);
    }

    if (authenticateConnection(fd1, url.user, url.password) != 230)
    {
        printf("Incorrect username: '%s'\nIncorrect password: '%s'.\n",
url.user, url.password);
        exit(-1);
    }
```

```
    if (passiveMode(fd1, ip, &port) != 227)
    {
        printf("Failed to switch to passive mode\n");
        exit(-1);
    }

    fd2 = establishSocketConnection(ip, port);
    if (fd2 < 0)
    {
        printf("Failed to create socket to '%s:%d'\n", ip, port);
        exit(-1);
    }

    if (sendFileRequest(fd1, url.resource) != 150)
    {
        printf("Unidentified resource '%s' at '%s:%d'\n", url.resource,
ip, port);
        exit(-1);
    }

    if (transferFileFromServer(fd1, fd2, url.file) != 226)
    {
        printf("Failed to transfer file '%s' from '%s:%d'\n", url.file,
ip, port);
        exit(-1);
    }

    if (closeSocketConnection(fd1, fd2) != 0)
    {
        printf("Error closing sockets\n");
        exit(-1);
    }

    return 0;
}
```

download.h

```c
#include <stdio.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <stdlib.h>
#include <netdb.h>
#include <unistd.h>
#include <string.h>
#include <regex.h>
#include <termios.h>

struct URL
{
    char host[500];
    char resource[500];
    char file[500];
    char user[500];
    char password[500];
    char ip[500];
};

typedef enum
{
    START,
    SINGLE,
    MULTIPLE,
    END
} responseStatus;

int parseURL(char *inputURL, struct URL *url);
int establishSocketConnection(char *ip, int port);
int authenticateConnection(const int fd, const char *user, const char *pass);
int readSocketResponse(const int fd, char *buffer);
int passiveMode(const int fd, char *ip, int *port);
int sendFileRequest(const int fd, char *resource);
int transferFileFromServer(const int fd1, const int fd2, char *filename);
int closeSocketConnection(const int socketA, const int socketB);

```

Logs (Wireshark)

Exp 1 - Ping do tux4 no tux3

No.	Time	Source	Destination	Protocol	Length	Info
6	9.262986864	HewlettPacka_19:09…	Broadcast	ARP	42	Who has 172.16.50.1? Tell 172.16.50.254
7	9.263095398	HewlettPacka_61:2c…	HewlettPacka_19:09…	ARP	60	172.16.50.1 is at 00:21:5a:61:2c:54
8	9.263117887	172.16.50.254	172.16.50.1	ICMP	98	Echo (ping) request id=0x1394, seq=1/256, ttl=64 (reply in 9)
9	9.263207772	172.16.50.1	172.16.50.254	ICMP	98	Echo (ping) reply id=0x1394, seq=1/256, ttl=64 (request in 8)
10	10.010702758	Routerboardc_1c:95…	Spanning-tree-(for…	STP	60	RST. Root = 32768/0/c4:ad:34:1c:95:c8 Cost = 0 Port = 0x8004
11	10.292155658	172.16.50.254	172.16.50.1	ICMP	98	Echo (ping) request id=0x1394, seq=2/512, ttl=64 (reply in 12)
12	10.292284445	172.16.50.1	172.16.50.254	ICMP	98	Echo (ping) reply id=0x1394, seq=2/512, ttl=64 (request in 11)
13	11.312158481	172.16.50.254	172.16.50.1	ICMP	98	Echo (ping) request id=0x1394, seq=3/768, ttl=64 (reply in 14)
14	11.312287617	172.16.50.1	172.16.50.254	ICMP	98	Echo (ping) reply id=0x1394, seq=3/768, ttl=64 (request in 13)

Exp 2 - Ping Broadcast no Tux3

39	58.063520179	Routerboardc_1c:95…	Spanning-tree-(for…	STP	60	RST. Root = 32768/0/c4:ad:34:1c:95:d1 Cost = 0 Port = 0x8002
40	58.903328038	172.16.50.1	172.16.50.254	ICMP	98	Echo (ping) request id=0x3966, seq=3/768, ttl=64 (reply in 41)
41	58.903457104	172.16.50.254	172.16.50.1	ICMP	98	Echo (ping) reply id=0x3966, seq=3/768, ttl=64 (request in 40)
42	59.927322396	172.16.50.1	172.16.50.254	ICMP	98	Echo (ping) request id=0x3966, seq=4/1024, ttl=64 (reply in 43)
43	59.927482750	172.16.50.254	172.16.50.1	ICMP	98	Echo (ping) reply id=0x3966, seq=4/1024, ttl=64 (request in 42)

Exp 2 - Ping Broadcast no Tux2

17	22.441105168	172.16.51.1	172.16.51.255	ICMP	98	Echo (ping) request id=0x4e0a, seq=2/512, ttl=64 (no response found!)
18	23.465108894	172.16.51.1	172.16.51.255	ICMP	98	Echo (ping) request id=0x4e0a, seq=3/768, ttl=64 (no response found!)
19	24.026188811	Routerboardc_1c:95…	Spanning-tree-(for…	STP	60	RST. Root = 32768/0/c4:ad:34:1c:95:d2 Cost = 0 Port = 0x8001
20	24.489106474	172.16.51.1	172.16.51.255	ICMP	98	Echo (ping) request id=0x4e0a, seq=4/1024, ttl=64 (no response found!)
21	25.513109222	172.16.51.1	172.16.51.255	ICMP	98	Echo (ping) request id=0x4e0a, seq=5/1280, ttl=64 (no response found!)

Exp 3 - Capturado no tux4, ping do tux2 no tux3

310	166.8723596…	HewlettPacka_5a:78…	HewlettPacka_a7:26…	ARP	60	Who has 172.16.20.254? Tell 172.16.20.1
311	166.8723695…	HewlettPacka_a7:26…	HewlettPacka_5a:78…	ARP	42	172.16.20.254 is at 00:22:64:a7:26:a2
312	167.8644089…	172.16.20.1	172.16.21.1	ICMP	98	Echo (ping) request id=0x0f62, seq=4/1024, ttl=64 (reply in 313)
313	167.8645682…	172.16.21.1	172.16.20.1	ICMP	98	Echo (ping) reply id=0x0f62, seq=4/1024, ttl=63 (request in 312)
314	168.1826723…	Routerboardc_2b:fa…	Spanning-tree-(for…	STP	60	RST. Root = 32768/0/c4:ad:34:2b:fa:15 Cost = 0 Port = 0x8002
315	168.8883882…	172.16.20.1	172.16.21.1	ICMP	98	Echo (ping) request id=0x0f62, seq=5/1280, ttl=64 (reply in 316)
316	168.8885463…	172.16.21.1	172.16.20.1	ICMP	98	Echo (ping) reply id=0x0f62, seq=5/1280, ttl=63 (request in 315)
317	169.9123938…	172.16.20.1	172.16.21.1	ICMP	98	Echo (ping) request id=0x0f62, seq=6/1536, ttl=64 (reply in 318)
318	169.9125527…	172.16.21.1	172.16.20.1	ICMP	98	Echo (ping) reply id=0x0f62, seq=6/1536, ttl=63 (request in 317)

Exp 4 - Tux3 ping ao tux4

12	8.039238532	172.16.50.1	172.16.50.254	ICMP	98	Echo (ping) request id=0x66ad, seq=4/1024, ttl=64 (reply in 13)
13	8.039361316	172.16.50.254	172.16.50.1	ICMP	98	Echo (ping) reply id=0x66ad, seq=4/1024, ttl=64 (request in 12)
14	9.063239243	172.16.50.1	172.16.50.254	ICMP	98	Echo (ping) request id=0x66ad, seq=5/1280, ttl=64 (reply in 15)
15	9.063394644	172.16.50.254	172.16.50.1	ICMP	98	Echo (ping) reply id=0x66ad, seq=5/1280, ttl=64 (request in 14)

Exp 4 - Tux3 ping ao tux2

301	23.584891058	172.16.50.1	172.16.51.1	ICMP	98	Echo (ping) request id=0x66be, seq=1/256, ttl=64 (reply in 302)
302	23.585323035	172.16.51.1	172.16.50.1	ICMP	98	Echo (ping) reply id=0x66be, seq=1/256, ttl=63 (request in 301)

Exp 4 - Tux3 ping ao Router

392	43.911210951	HewlettPacka_61:2d…	HewlettPacka_c3:78…	ARP	42	Who has 172.16.50.254? Tell 172.16.50.1
393	43.911349938	HewlettPacka_c3:78…	HewlettPacka_61:2d…	ARP	60	172.16.50.254 is at 00:21:5a:c3:78:70
394	44.021548293	Routerboardc_1c:8b…	Spanning-tree-(for…	STP	60	RST. Root = 32768/0/c4:ad:34:1c:8b:c6 Cost = 0 Port = 0x8001
395	44.295244368	172.16.50.1	172.16.51.254	ICMP	98	Echo (ping) request id=0x66cb, seq=5/1280, ttl=64 (reply in 396)
396	44.295523809	172.16.51.254	172.16.50.1	ICMP	98	Echo (ping) reply id=0x66cb, seq=5/1280, ttl=63 (request in 395)
397	45.319244664	172.16.50.1	172.16.51.254	ICMP	98	Echo (ping) request id=0x66cb, seq=6/1536, ttl=64 (reply in 398)
398	45.319516841	172.16.51.254	172.16.50.1	ICMP	98	Echo (ping) reply id=0x66cb, seq=6/1536, ttl=63 (request in 397)

Exp 4 - Ping do router no tux3 (pré-disable do NAT)

22	7.165080819	HewlettPacka_c3:78…	HewlettPacka_61:2d…	ARP	60	Who has 172.16.50.1? Tell 172.16.50.254
23	7.165101772	HewlettPacka_61:2d…	HewlettPacka_c3:78…	ARP	42	172.16.50.1 is at 00:21:5a:61:2d:72
24	7.998661706	Routerboardc_1c:8b…	Spanning-tree-(for…	STP	60	RST. Root = 32768/0/c4:ad:34:1c:8b:c6 Cost = 0 Port = 0x8001
25	8.136569752	172.16.50.1	172.16.1.254	ICMP	98	Echo (ping) request id=0x687b, seq=7/1792, ttl=64 (reply in 26)
26	8.136965411	172.16.1.254	172.16.50.1	ICMP	98	Echo (ping) reply id=0x687b, seq=7/1792, ttl=62 (request in 25)
27	9.160565547	172.16.50.1	172.16.1.254	ICMP	98	Echo (ping) request id=0x687b, seq=8/2048, ttl=64 (reply in 28)
28	9.160980972	172.16.1.254	172.16.50.1	ICMP	98	Echo (ping) reply id=0x687b, seq=8/2048, ttl=62 (request in 27)

Exp 4 - Ping do router no tux3 (pós-disable do NAT)

15	10.071262823	HewlettPacka_61:2d…	HewlettPacka_c3:78…	ARP	42	Who has 172.16.50.254? Tell 172.16.50.1
16	10.071391822	HewlettPacka_c3:78…	HewlettPacka_61:2d…	ARP	60	172.16.50.254 is at 00:21:5a:c3:78:70
17	10.103278244	172.16.50.1	172.16.1.254	ICMP	98	Echo (ping) request id=0x68e6, seq=6/1536, ttl=64 (no response found!)
18	11.127279419	172.16.50.1	172.16.1.254	ICMP	98	Echo (ping) request id=0x68e6, seq=7/1792, ttl=64 (no response found!)
19	12.013006362	Routerboardc_1c:8b…	Spanning-tree-(for…	STP	60	RST. Root = 32768/0/c4:ad:34:1c:8b:c6 Cost = 0 Port = 0x8001
20	12.151302105	172.16.50.1	172.16.1.254	ICMP	98	Echo (ping) request id=0x68e6, seq=8/2048, ttl=64 (no response found!)
21	13.175278765	172.16.50.1	172.16.1.254	ICMP	98	Echo (ping) request id=0x68e6, seq=9/2304, ttl=64 (no response found!)

Exp 5 - Ping google.com

11	11.047663401	172.16.51.1	193.136.28.10	DNS	70	Standard query 0xabe7 A google.com
12	11.047673946	172.16.51.1	193.136.28.10	DNS	70	Standard query 0x40f0 AAAA google.com
13	11.048294617	193.136.28.10	172.16.51.1	DNS	334	Standard query response 0xabe7 A google.com A 142.250.184.174 NS ns3.
14	11.048353353	193.136.28.10	172.16.51.1	DNS	346	Standard query response 0x40f0 AAAA google.com AAAA 2a00:1450:4003:80
15	11.048641444	172.16.51.1	142.250.184.174	ICMP	98	Echo (ping) request id=0x5fa1, seq=1/256, ttl=64 (reply in 16)
16	11.063983747	142.250.184.174	172.16.51.1	ICMP	98	Echo (ping) reply id=0x5fa1, seq=1/256, ttl=108 (request in 15)
17	11.064076145	172.16.51.1	193.136.28.10	DNS	88	Standard query 0x223e PTR 174.184.250.142.in-addr.arpa
18	11.064638150	193.136.28.10	172.16.51.1	DNS	385	Standard query response 0x223e PTR 174.184.250.142.in-addr.arpa PTR m

Exp 6 - Primeira transferência (um ficheiro - tux3)

8	12.506785407	172.16.50.1	193.136.28.10	DNS	76	Standard query 0x7a44 A netlab1.fe.up.pt
9	12.507687260	193.136.28.10	172.16.50.1	DNS	286	Standard query response 0x7a44 A netlab1.fe.up.pt A 192.168.109.136 NS ns1.fe.up.pt NS cns1.fe.up.pt NS cns2.fe.u
10	12.507848382	172.16.50.1	192.168.109.136	TCP	74	51278 → 21 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM TSval=2964992497 TSecr=0 WS=128
11	12.508449850	192.168.109.136	172.16.50.1	TCP	74	21 → 51278 [SYN, ACK] Seq=0 Ack=1 Win=65160 Len=0 MSS=1460 SACK_PERM TSval=1539790828 TSecr=2964992497 WS=128
12	12.508467170	172.16.50.1	192.168.109.136	TCP	66	51278 → 21 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=2964992498 TSecr=1539790828
13	12.510153753	192.168.109.136	172.16.50.1	FTP	100	Response: 220 Welcome to netlab-FTP server
14	12.510163740	172.16.50.1	192.168.109.136	TCP	66	51278 → 21 [ACK] Seq=1 Ack=35 Win=64256 Len=0 TSval=2964992500 TSecr=1539790830
15	12.510229740	172.16.50.1	192.168.109.136	FTP	81	Request: user anonymous
16	12.510694110	192.168.109.136	172.16.50.1	TCP	66	21 → 51278 [ACK] Seq=35 Ack=16 Win=65280 Len=0 TSval=1539790830 TSecr=2964992500
17	12.510748167	192.168.109.136	172.16.50.1	FTP	100	Response: 331 Please specify the password.
18	12.510802014	172.16.50.1	192.168.109.136	FTP	80	Request: pass password
19	12.511358714	192.168.109.136	172.16.50.1	TCP	66	21 → 51278 [ACK] Seq=69 Ack=30 Win=65280 Len=0 TSval=1539790831 TSecr=2964992500
20	12.512690855	192.168.109.136	172.16.50.1	FTP	89	Response: 230 Login successful.
21	12.512730455	172.16.50.1	192.168.109.136	FTP	71	Request: pasv
22	12.513246228	192.168.109.136	172.16.50.1	TCP	66	21 → 51278 [ACK] Seq=92 Ack=35 Win=65280 Len=0 TSval=1539790833 TSecr=2964992502
23	12.513384373	192.168.109.136	172.16.50.1	FTP	119	Response: 227 Entering Passive Mode (192,168,109,136,169,79).
24	12.513472442	172.16.50.1	192.168.109.136	TCP	74	41362 → 43343 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM TSval=2964992503 TSecr=0 WS=128
25	12.514095282	192.168.109.136	172.16.50.1	TCP	74	43343 → 41362 [SYN, ACK] Seq=0 Ack=1 Win=65160 Len=0 MSS=1460 SACK_PERM TSval=1539790834 TSecr=2964992503 WS=128
26	12.514106526	172.16.50.1	192.168.109.136	TCP	66	41362 → 43343 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=2964992503 TSecr=1539790834
27	12.514121681	172.16.50.1	192.168.109.136	FTP	79	Request: retr pub.txt
28	12.514683550	192.168.109.136	172.16.50.1	TCP	66	21 → 51278 [ACK] Seq=145 Ack=48 Win=65280 Len=0 TSval=1539790834 TSecr=2964992504
29	12.514826933	192.168.109.136	172.16.50.1	FTP	132	Response: 150 Opening BINARY mode data connection for pub.txt (672 bytes).
30	12.515015363	192.168.109.136	172.16.50.1	FTP-D…	738	FTP Data: 672 bytes (PASV) (retr pub.txt)
31	12.515024372	172.16.50.1	192.168.109.136	TCP	66	41362 → 43343 [ACK] Seq=1 Ack=673 Win=64128 Len=0 TSval=2964992504 TSecr=1539790834
32	12.515027655	192.168.109.136	172.16.50.1	TCP	66	43343 → 41362 [FIN, ACK] Seq=673 Ack=1 Win=65280 Len=0 TSval=1539790835 TSecr=2964992503
33	12.557585631	172.16.50.1	192.168.109.136	TCP	66	41362 → 43343 [ACK] Seq=1 Ack=674 Win=64128 Len=0 TSval=2964992547 TSecr=1539790835
34	12.557595549	172.16.50.1	192.168.109.136	TCP	66	51278 → 21 [ACK] Seq=48 Ack=211 Win=64256 Len=0 TSval=2964992547 TSecr=1539790834
35	12.558262877	192.168.109.136	172.16.50.1	FTP	90	Response: 226 Transfer complete.

Exp 6 - Segunda transferência (dois ficheiros)

tux3 - (ínicio e fim)

4	2.305794431	193.136.28.10	172.16.50.1	DNS	286	Standard query response 0xfaa6 A netlab1.fe.up.pt A 192.168.109.136 NS cns1.fe.up.pt NS ns1.fe.up.pt NS ns2.fe.up.p
5	2.305966937	172.16.50.1	192.168.109.136	TCP	74	54834 → 21 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM TSval=2968263047 TSecr=0 WS=128
6	2.306794757	192.168.109.136	172.16.50.1	TCP	74	21 → 54834 [SYN, ACK] Seq=0 Ack=1 Win=65160 Len=0 MSS=1460 SACK_PERM TSval=1543061377 TSecr=2968263047 WS=128
7	2.306814662	172.16.50.1	192.168.109.136	TCP	66	54834 → 21 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=2968263048 TSecr=1543061377
8	2.308914837	192.168.109.136	172.16.50.1	FTP	100	Response: 220 Welcome to netlab-FTP server
9	2.308927478	172.16.50.1	192.168.109.136	TCP	66	54834 → 21 [ACK] Seq=1 Ack=35 Win=64256 Len=0 TSval=2968263050 TSecr=1543061379
10	2.308995643	172.16.50.1	192.168.109.136	FTP	76	Request: user rcom
11	2.309726594	192.168.109.136	172.16.50.1	TCP	66	21 → 54834 [ACK] Seq=35 Ack=11 Win=65280 Len=0 TSval=1543061380 TSecr=2968263050
12	2.309798320	192.168.109.136	172.16.50.1	FTP	100	Response: 331 Please specify the password.
13	2.309852796	172.16.50.1	192.168.109.136	FTP	76	Request: pass rcom
14	2.310316398	192.168.109.136	172.16.50.1	TCP	66	21 → 54834 [ACK] Seq=69 Ack=21 Win=65280 Len=0 TSval=1543061381 TSecr=2968263051
15	2.319677617	192.168.109.136	172.16.50.1	FTP	89	Response: 230 Login successful.
16	2.319724759	172.16.50.1	192.168.109.136	FTP	71	Request: pasv
17	2.320159517	192.168.109.136	172.16.50.1	TCP	66	21 → 54834 [ACK] Seq=92 Ack=26 Win=65280 Len=0 TSval=1543061390 TSecr=2968263061
18	2.320331185	192.168.109.136	172.16.50.1	FTP	120	Response: 227 Entering Passive Mode (192,168,109,136,187,143).
19	2.320430568	172.16.50.1	192.168.109.136	TCP	74	37630 → 48015 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM TSval=2968263061 TSecr=0 WS=128
20	2.320953744	192.168.109.136	172.16.50.1	TCP	74	48015 → 37630 [SYN, ACK] Seq=0 Ack=1 Win=65160 Len=0 MSS=1460 SACK_PERM TSval=1543061391 TSecr=2968263061 WS=128
21	2.320971204	172.16.50.1	192.168.109.136	TCP	66	37630 → 48015 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=2968263062 TSecr=1543061391
22	2.320985382	172.16.50.1	192.168.109.136	FTP	109	Request: retr files/archlinux-2023.03.01-x86_64.iso
23	2.321506811	192.168.109.136	172.16.50.1	TCP	66	21 → 54834 [ACK] Seq=146 Ack=69 Win=65280 Len=0 TSval=1543061392 TSecr=2968263062
24	2.321659273	192.168.109.136	172.16.50.1	FTP	168	Response: 150 Opening BINARY mode data connection for files/archlinux-2023.03.01-x86_64.iso (849686528 bytes).
25	2.321946946	192.168.109.136	172.16.50.1	FTP-D…	1514	FTP Data: 1448 bytes (PASV) (retr files/archlinux-2023.03.01-x86_64.iso)

400…	74.910242050	172.16.50.1	192.168.109.136	TCP	66	37630 → 48015 [ACK] Seq=1 Ack=849683505 Win=2856064 Len=0 TSval=2968335651 TSecr=1543133973
400…	74.910357845	192.168.109.136	172.16.50.1	FTP-D…	1514	FTP Data: 1448 bytes (PASV) (retr files/archlinux-2023.03.01-x86_64.iso)
400…	74.910482581	192.168.109.136	172.16.50.1	FTP-D…	1642	FTP Data: 1576 bytes (PASV) (retr files/archlinux-2023.03.01-x86_64.iso)
400…	74.910501158	172.16.50.1	192.168.109.136	TCP	66	37630 → 48015 [ACK] Seq=1 Ack=849686530 Win=2855424 Len=0 TSval=2968335651 TSecr=1543133973
400…	74.911351886	192.168.109.136	172.16.50.1	FTP	90	Response: 226 Transfer complete.
400…	74.911362013	172.16.50.1	192.168.109.136	TCP	66	54834 → 21 [ACK] Seq=69 Ack=272 Win=64256 Len=0 TSval=2968335652 TSecr=1543133982
400…	75.535512844	172.16.50.1	192.168.109.136	FTP	71	Request: quit
400…	75.536272570	192.168.109.136	172.16.50.1	TCP	66	21 → 54834 [ACK] Seq=272 Ack=74 Win=65280 Len=0 TSval=1543134607 TSecr=2968336276
400…	75.536313496	192.168.109.136	172.16.50.1	FTP	80	Response: 221 Goodbye.
400…	75.536325579	172.16.50.1	192.168.109.136	TCP	66	54834 → 21 [ACK] Seq=74 Ack=286 Win=64256 Len=0 TSval=2968336277 TSecr=1543134607
400…	75.536344715	192.168.109.136	172.16.50.1	TCP	66	21 → 54834 [FIN, ACK] Seq=286 Ack=74 Win=65280 Len=0 TSval=1543134607 TSecr=2968336276
400…	75.536376912	172.16.50.1	192.168.109.136	TCP	66	54834 → 21 [FIN, ACK] Seq=74 Ack=287 Win=64256 Len=0 TSval=2968336277 TSecr=1543134607
400…	75.536392486	172.16.50.1	192.168.109.136	TCP	66	37630 → 48015 [FIN, ACK] Seq=1 Ack=849686530 Win=2857216 Len=0 TSval=2968336277 TSecr=1543133973
400…	75.536791834	192.168.109.136	172.16.50.1	TCP	66	21 → 54834 [ACK] Seq=287 Ack=75 Win=65280 Len=0 TSval=1543134607 TSecr=2968336277
400…	75.536833180	192.168.109.136	172.16.50.1	TCP	66	48015 → 37630 [ACK] Seq=849686530 Ack=2 Win=65280 Len=0 TSval=1543134607 TSecr=2968336277

tux2 -

12	20.572062325	172.16.51.1	193.136.28.10	DNS	76	Standard query 0x14a4 A netlab1.fe.up.pt
13	20.577354635	193.136.28.10	172.16.51.1	DNS	286	Standard query response 0x14a4 A netlab1.fe.up.pt A 192.168.109.136 NS cns2.fe.up.pt NS ns1.fe.up.pt NS ns2.fe.
14	20.577486497	172.16.51.1	192.168.109.136	TCP	74	48848 → 21 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM TSval=3024883620 TSecr=0 WS=128
15	20.579522614	192.168.109.136	172.16.51.1	TCP	74	21 → 48848 [SYN, ACK] Seq=0 Ack=1 Win=65160 Len=0 MSS=1460 SACK_PERM TSval=1543068637 TSecr=3024883620 WS=128
16	20.579541681	172.16.51.1	192.168.109.136	TCP	66	48848 → 21 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=3024883623 TSecr=1543068637
17	20.588103314	192.168.109.136	172.16.51.1	FTP	100	Response: 220 Welcome to netlab-FTP server
18	20.588117073	172.16.51.1	192.168.109.136	TCP	66	48848 → 21 [ACK] Seq=1 Ack=35 Win=64256 Len=0 TSval=3024883631 TSecr=1543068647
19	20.588184820	172.16.51.1	192.168.109.136	FTP	76	Request: user rcom
20	20.589980120	192.168.109.136	172.16.51.1	TCP	66	21 → 48848 [ACK] Seq=35 Ack=11 Win=65280 Len=0 TSval=1543068647 TSecr=3024883631
21	20.589986126	192.168.109.136	172.16.51.1	FTP	100	Response: 331 Please specify the password.
22	20.590038927	172.16.51.1	192.168.109.136	FTP	76	Request: pass rcom
23	20.599309531	192.168.109.136	172.16.51.1	TCP	66	21 → 48848 [ACK] Seq=69 Ack=21 Win=65280 Len=0 TSval=1543068658 TSecr=3024883633
24	20.610419296	192.168.109.136	172.16.51.1	FTP	89	Response: 230 Login successful.
25	20.610465601	172.16.51.1	192.168.109.136	FTP	71	Request: pasv
26	20.612360007	192.168.109.136	172.16.51.1	TCP	66	21 → 48848 [ACK] Seq=92 Ack=26 Win=65280 Len=0 TSval=1543068670 TSecr=3024883653
27	20.612365385	192.168.109.136	172.16.51.1	FTP	120	Response: 227 Entering Passive Mode (192,168,109,136,191,179).
28	20.612459952	172.16.51.1	192.168.109.136	TCP	74	40050 → 49075 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM TSval=3024883655 TSecr=0 WS=128
29	20.620951184	192.168.109.136	172.16.51.1	TCP	74	49075 → 40050 [SYN, ACK] Seq=0 Ack=1 Win=65160 Len=0 MSS=1460 SACK_PERM TSval=1543068676 TSecr=3024883655 WS=12
30	20.620964803	172.16.51.1	192.168.109.136	TCP	66	40050 → 49075 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=3024883664 TSecr=1543068676
31	20.620985546	172.16.51.1	192.168.109.136	FTP	80	Request: retr pipe.txt
32	20.623433663	192.168.109.136	172.16.51.1	TCP	66	21 → 48848 [ACK] Seq=146 Ack=40 Win=65280 Len=0 TSval=1543068681 TSecr=3024883664
33	20.623439041	192.168.109.136	172.16.51.1	FTP	134	Response: 150 Opening BINARY mode data connection for pipe.txt (1863 bytes).
34	20.623590459	192.168.109.136	172.16.51.1	FTP-D…	1929	FTP Data: 1863 bytes (PASV) (retr pipe.txt)
35	20.623598910	172.16.51.1	192.168.109.136	TCP	66	40050 → 49075 [ACK] Seq=1 Ack=1864 Win=63488 Len=0 TSval=3024883667 TSecr=1543068681
36	20.623602612	192.168.109.136	172.16.51.1	TCP	66	49075 → 40050 [FIN, ACK] Seq=1864 Ack=1 Win=65280 Len=0 TSval=1543068681 TSecr=3024883664
37	20.666921747	172.16.51.1	192.168.109.136	TCP	66	40050 → 49075 [ACK] Seq=1 Ack=1865 Win=64128 Len=0 TSval=3024883710 TSecr=1543068681
38	20.666926007	172.16.51.1	192.168.109.136	TCP	66	48848 → 21 [ACK] Seq=40 Ack=214 Win=64256 Len=0 TSval=3024883710 TSecr=1543068681
39	20.675971648	192.168.109.136	172.16.51.1	FTP	90	Response: 226 Transfer complete.
40	20.675977376	172.16.51.1	192.168.109.136	TCP	66	48848 → 21 [ACK] Seq=40 Ack=238 Win=64256 Len=0 TSval=3024883719 TSecr=1543068728
41	20.676022564	172.16.51.1	192.168.109.136	FTP	71	Request: quit
42	20.677669727	192.168.109.136	172.16.51.1	FTP	80	Response: 221 Goodbye.
43	20.677709677	192.168.109.136	172.16.51.1	TCP	66	21 → 48848 [FIN, ACK] Seq=252 Ack=45 Win=65280 Len=0 TSval=1543068736 TSecr=3024883719
44	20.677719804	172.16.51.1	192.168.109.136	TCP	66	48848 → 21 [FIN, ACK] Seq=45 Ack=253 Win=64256 Len=0 TSval=3024883721 TSecr=1543068736
45	20.677730560	172.16.51.1	192.168.109.136	TCP	66	40050 → 49075 [FIN, ACK] Seq=1 Ack=1865 Win=64128 Len=0 TSval=3024883721 TSecr=1543068681
46	20.679443725	192.168.109.136	172.16.51.1	TCP	66	21 → 48848 [ACK] Seq=253 Ack=46 Win=65280 Len=0 TSval=1543068737 TSecr=3024883721
47	20.679455249	192.168.109.136	172.16.51.1	TCP	66	49075 → 40050 [ACK] Seq=1865 Ack=2 Win=65280 Len=0 TSval=1543068737 TSecr=3024883721