Relatório - Trabalho Laboratorial Nº2

Licenciatura em Engenharia Electrotécnica e de Computadores

Redes de Computadores

Duarte Ribeiro Afonso Branco 201905327
Pedro Afonso da Silva Correia de Castro Lopes 201907097

Junho 2022

Índice

Sumário

Este trabalho, realizado no âmbito da unidade curricular de Redes de Computadores, teve como objetivo o desenvolvimento de uma aplicação de *download* através do protocolo *File Transfer Protocol* (*FTP)*, assim como o desenvolvimento de uma rede de computadores. Após configurada a rede a aplicação de *download* foi testada dentro da mesma.

Os objetivos deste trabalho foram alcançados, visto que foram alcançados os objetivos definidos, nomeadamente, verificou-se o correto funcionamento da aplicação de *download* quando inserida na rede de computadores desenvolvida, sendo esta capaz de aceder à *internet* através do *router.*

Introdução

Este trabalho teve como objetivos a configuração de uma rede de computadores, composta por duas sub-redes, três *tuxs* e um *router*, capaz de garantir o acesso à *internet*. Foi ainda desenvolvida uma aplicação de *download* que, através do protocolo *FTP*, como o nome indica, realiza o *download* de qualquer ficheiro presente num servidor *FTP*. Esta aplicação foi introduzida na rede de computadores desenvolvida verificando-se a transmissão dos dados através da rede criada.

O presente relatório encontra-se dividido em diversas secções. Em Arquitetura e Estrutura do código é descrita a organização do código de um modo geral. Em Casos de uso principais é descrito como executar o projeto e as sequências de chamadas de funções. Em Protocolo de ligação lógica é descrito em detalhe todas as funções do código implementado e em Protocolo de aplicação é descrito o funcionamento do código fornecido. Em Validação e Elementos de valorização são descritos os testes realizados e descritos os elementos de valorização implementados, respectivamente. Por fim, na Conclusão é feita uma síntese de todos os tópicos e reflexão do trabalho realizado.

Parte 1 - Aplicação de download

Arquitetura

O projeto desenvolvido, correspondente ao ficheiro *LAB2_FTP.c*, recebe como argumento o *URL* do ficheiro a transferir no seguinte formato:

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

Primeiramente é verificado se, de facto, é inserido o número correto de argumentos, procedendo-se depois ao processamento do *URL*. Para tal foi implementada a função *processURL* que separa e guarda os segmentos do *URL* no *array* correspondente na *struct URL*. Caso não seja indicado *user* e *password*, ou seja, ocorra uma ligação em modo anónimo a função *processURL* define os valores de *user* e *password* como *anonymous*, funcionando normalmente para os restantes segmentos.

Seguidamente, é utilizada a função *gethostname* de modo a determinar o *IP* correspondente ao *host* indicado. É iniciada a ligação ao servidor através da porta *default* de um servidor *FTP*, nomeadamente a porta 21, através da função *socket*, sendo o descritor de

ficheiro correspondente à *socket* indicado na função *connect* que assegura a ligação ao *IP* do servidor através da *socket* criada.

A função *read_reply* é utilizada para escrever no terminal a mensagem de resposta e identificar o código de resposta devolvido pelo servidor *FTP*. Se o código de resposta não for o esperado é indicado erro e terminado o programa.

A função *send_msg* é utilizada para transmitir informação do cliente para o servidor, sendo indicado o tipo de mensagem, por exemplo *"user ",* e o seu conteúdo.

Assim sendo, através da utilização destas funções foi continuado o processo de ligação ao servidor. São transmitidos os dados de utilizador e palavra-passe e depois enviado o comando *"pasv "* de modo a iniciar-se uma ligação passiva com o servidor para a transferência do ficheiro.

Tendo em conta que na resposta após enviado o comando para ligação passiva é necessário guardar os valores das constantes utilizadas para o cálculo da nova porta passiva, foi desenvolvida a função *read_reply_pasv* que para além de guardar os números indicados comporta-se como a *read_reply*.

É iniciada uma ligação com o servidor através da porta passiva e enviado o comando *"retr "* com o *path* do ficheiro a transferir. A função *get_file_name* permite distinguir o nome do ficheiro do seu *path*. Por fim, é criado um ficheiro com o nome determinado e escrito neste os dados transmitidos através do *socket* associado à ligação passiva, indicado o tamanho do ficheiro para verificação e fechadas as ligações e ficheiro.

Em caso de erro este é indicado e o programa termina, fechando as ligações existentes. Sempre que relevante a memória alocada em excesso é livre.

Download bem sucedido

A aplicação desenvolvida foi testada com diversos tipos de ficheiro e de diversos tamanhos, incluindo o ficheiro *crab.mp4*, em modo anónimo e não anónimo e inserida na rede de computadores desenvolvida. Verificou-se em todos os casos a total e correta transmissão dos ficheiros desejados.

Parte 2 - Configuração e Análise da Rede

Experiência 1

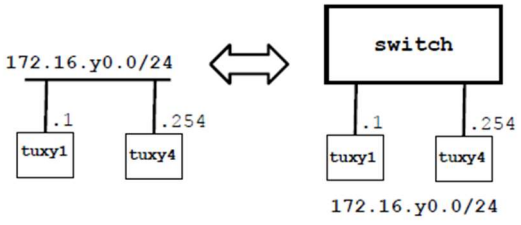

Figura 1: Representação gráfica da experiência 1

A primeira experiência teve como objetivo a configuração dos endereços *IP* das *tux4* e *tux3,* correspondente em todas as experiências à *tux1* indicada no Guião, a verificação dos endereços *MAC* correspondentes e a interpretação do protocolo *ARP.* Todos os comandos utilizados encontram-se indicados no Anexo 2, Experiência 1 e a respetiva captura no Anexo 3, Experiência 1 através da qual se verificam os comportamentos descritos seguidamente.

O protocolo *Address Resolution Protocol* (*ARP*) associa os endereços *IP* aos endereços *MAC* correspondentes. Assim sendo, ao existir um *IP* não associado a um *MAC* são transmitidos pelo emissor para todos os computadores presentes na rede, através de um *broadcast*, pacotes *ARP* que indicam o *IP* do destinatário. Por sua vez, este responde com o seu *MAC* que fica então associado ao seu *IP* numa tabela.

Os endereços *MAC* e *IP* dos pacotes *ARP* correspondem aos únicos computadores na rede, ou seja, os *tux3* e *tux4*:

tux3:

*MAC:*00:21:5a:5a:78:c7

IP: 172.16.20.1

tux4:

MAC: 00.22:64:a7:26:a2

IP: 172.16.20.254

O comando *ping* gera pacotes do tipo *ICMP* que envia para o *IP* do destinatário, recebendo como resposta outro pacote *ICMP* do destinatário para o computador que invocou o comando, correspondendo os *IP*s e *MACs* ao par *tux3* e *tux4*.

É possível distinguir os diferentes tipos de pacotes, *ARP*, *IP* e *ICMP*, através dos seus diferentes cabeçalhos, sendo que o *Wireshark* identifica o tipo de pacote e facilita a sua visualização com cores correspondentes. O tamanho do *frame* a receber encontra-se no campo *Frame lenght* do *Wireshark*.

A interferência de *loopback* corresponde a uma mensagem enviada e recebida a cada dez segundos com o objetivo de verificar a ligação à rede.

Experiência 2

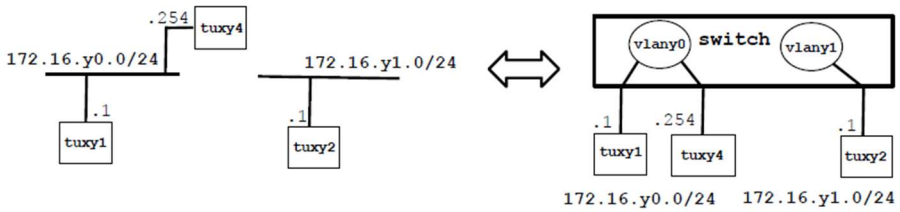

Figura 2: Representação gráfica da experiência 2

Nesta experiência foi definido o endereço *IP* do *tux2* e configuradas duas sub-redes do modo indicado pela Figura 2. É possível verificar como configurar o *vlan20* no Anexo2, Experência 2.

Através do *ping* verificou-se a impossibilidade de contacto entre os computadores, como verificável no Anexo3, Experiência 2. Assim sendo, existem dois domínios de *broadcast,* uma vez que, existem duas sub-redes distintas, nomeadamente, *vlan20* e *vlan21*, sem qualquer conexão.

Experiência 3

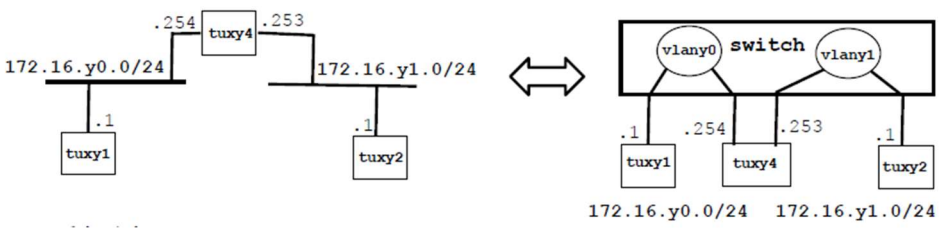

Figura 3: Representação gráfica da experiência 3

Nesta experiência foi adicionado um endereço *IP* à *tux4* associado à porta *eth1* na sub-rede *vlan21*, comportando-se este como um *router* e tornando possível a comunicação entre as duas sub-redes. Os comandos utilizados para a configuração do *tux4* e as tabelas rotas são indicados no Anexo 2, Experiência 3.

As tabelas de *forwarding* indicam qual a *gateway* a utilizar para um endereço *IP* destino definido, ou seja, para onde enviar o pacote tendo em conta que o seu destino final é noutra sub-rede. Por exemplo, para enviar um pacote do *tux2* para *tux3* é necessário configurar a *forwarding table* associando o *IP* da *tux3* ao *gateway* de *IP* 172.16.21.253

Assim foi possível realizar *ping* entre os *tux3* e *tux4*, correspondendo as mensagens *ARP* à associação do *IP* do *tux2* ao *MAC* do *tux4*. Os pacotes *ICMP* correspondem à verificação de ligação do *ping* entre o par de *tuxs* respetivo. O *IP* de destino é o do *tux2*, no entanto, o *MAC* de destino é o da *tux4*, associado na *ARP*, uma vez que este é responsável pela comunicação entre as duas sub-redes. É possível visualizar o descrito no Anexo 3, Experiência 3.

Experiência 4

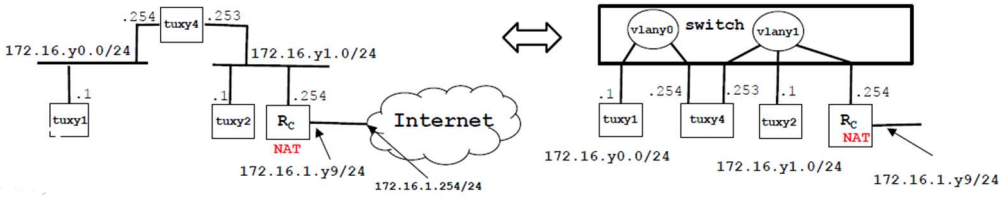

Figura 4: Representação gráfica da experiência 4

Nesta experiência configurou-se o *router* na *vlan21* e implementou-se *Network Address Translation* (*NAT*) permitindo o acesso à *internet*. É possível verificar como configurar o *router* e a *NAT* no Anexo 2, Experiência 4.

O protocolo *NAT* é utilizado para associar apenas um endereço *IP* público a uma rede privada com qualquer número de utilizadores, sendo assim possível economizar *IP*s.Este protocolo traduz os *IPs* privados no *IP* público e, posteriormente, identifica a qual *IP* privado enviar a resposta.

Através da análise do Anexo 3, Experiência 3 é possível verificar o caminho dos pacotes. Por exemplo, no *ping* do *tux3* ao *router*, os pacotes originários do *tux3* atingem o *IP* 172.16.20.254 da *tux4*, sendo encaminhados para o *IP* 172.16.20.253 da *tux4. Depois atingem o IP 172.16.21.254* do *router* e só, por fim, chegam ao *IP* 172.16.2.29 do *router.* Na resposta ao *ping* é realizado o caminho inverso.

Experiência 5

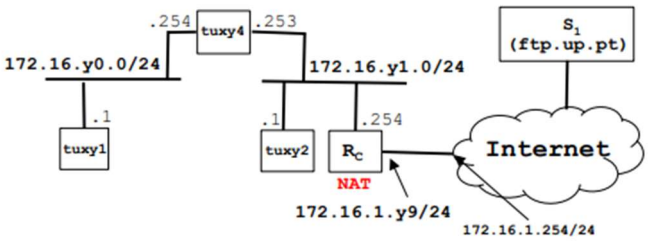

Figura 5: Representação gráfica da experiência 5

Nesta experiência configurou-se o serviço de *Domain Name Service (DNS)* que associa ao *link* do *site* o endereço IP.

A configuração do serviço *DNS*, encontra-se no Anexo 2, Experiência 5 onde é possível verificar os comandos utilizados para a edição do ficheiro *resolv.conf*, onde é associado o endereço IP do servidor *DNS*.

Foi realizado um *ping* para *ftp.up.pt* e *www.google.pt* e utilizando o *browser* acedeu-se ao *Youtube* verificando assim o correto funcionamento do *DNS*.

Experiência 6

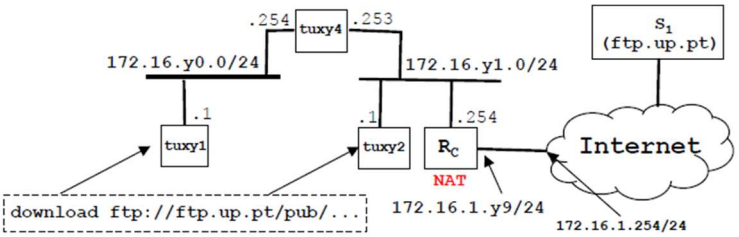

Figura 6: Representação gráfica da experiência 6

Nesta experiência foi introduzida a aplicação de *download* na rede desenvolvida. Para tal foi verificado o seu correto funcionamento e da rede, transferindo-se vários ficheiros dos servidores *FTP ftp.up.pt* e *netlab1.fe.up.pt* em modo anónimo e não anónimo.

Através das capturas no Anexo 3, Experiência 6, verifica-se que são estabelecidas duas conexões *FTP*, tal como definido no código: uma para a porta 21, onde é transportada a informação de controlo, e outra para a porta passiva onde é transmitido o ficheiro a transferir.

A ligação *TCP* tem três fase, nomeadamente, o *three-way handshake*, onde é garantida a ligação através do envio de um pacote *SYN,* da recepção de uma *SYN,ACK* e envio de *ACK*, a transferência de dados e fecho de ligação com o envio de *FIN* e recepção de *FIN, ACK*.

O *ARQ TCP* é utilizado na verificação de erros através do envio de *ACK* e de números de sequência, destacando-se o *sequence number field* e *ACK number field*, correspondendo ao número de sequência do pacote atual e do seguinte, respetivamente.

Verificou-se o controlo de congestionamento, através da transferência de dois ficheiros de elevado tamanho, no Anexo 3, Experiência 6, na medida em que, dependendo da quantidade de *ACKs* enviados, regula a janela de transmissão. Neste casos, ao fazer duas transferências simultâneas o congestionamento na rede aumenta pelo que a janela de transmissão diminui, sendo que quando é atingido o número máximo de pacotes a transmissão é interrompida, De modo inverso, se o congestionamento diminui a janela aumenta.

O *troughput* da ligação diminui com a introdução de uma segunda transmissão uma vez, o *troughput* total vai deixar de corresponder ao da ligação e vai ser dividido pelas duas ligações.

Conclusões

Através da realização deste trabalho foi possível o desenvolvimento de uma rede local composta por múltiplos clientes, com a capacidade de conectar-se à *internet* e de uma aplicação de *download* com protocolo *FTP*.

O projeto foi realizado com sucesso, destacando-se o aumento nas competências associadas, nomeadamente, à programação, configuração de redes, *tuxes* e *routers*, abordagem de problemas, *debugging* e implementação de protocolos associados ao *FTP*.

Em suma, compreendemos agora seguramente melhor como é realizada a transferência de informação através da camada de transporte e quais os algoritmos associados à transferência de dados e ligações *TCP*.

Anexo 1 - Código fonte

```
#include <stdio.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <stdlib.h>
#include <unistd.h>
#include <signal.h>
#include <netdb.h>
#include <string.h>

#define MAX_STRING_SIZE 1024
#define DEFAULT_FTP_PORT 21


struct URL_struct
{
    char* user;
    char* pass;
    char* host;
    char* path;
};

int processURL(char* url, struct URL_struct* URL)
{
    printf("----------------processURL\n");

    //ftp://<user>:<password>@<host>/<url-path>
    puts(url);
    if(url[0]!='f' || url[1]!='t' || url[2]!='p' || url[3]!=':' ||
url[4]!='/' || url[5]!='/') return -1;
    //printf("1\n");

    int i=6, state=0, pos=0, url_lenght=strlen(url);
    printf("url_lenght=%d\n", url_lenght);

    //if no user/pass set as anonymous
    const char ch1='@';
    if(strchr(url, ch1)==NULL)
    {
        state=2;
        strcpy(URL->user, "anonymous");
```

```
    strcpy(URL->pass, "anonymous");
}

while(i<url_lenght)
{
    //printf("url[%d]=%c\n", i, url[i]);
    switch (state)
    {
        //user until :
        case 0:
            if(pos>MAX_STRING_SIZE-1) return -1;
            else if(url[i]==':')
            {
                URL->user[pos]='\0';
                pos=0;
                state=1;
            }
            else
            {
                URL->user[pos]=url[i];
                pos++;
            }
            break;

        //pass until @
        case 1:
            if(pos>MAX_STRING_SIZE-1) return -1;
            else if(url[i]=='@')
            {
                URL->pass[pos]='\0';
                pos=0;
                state=2;
            }
            else
            {
                URL->pass[pos]=url[i];
                pos++;
            }
            break;

        //host until /
        case 2:
            if(pos>MAX_STRING_SIZE-1) return -1;
```

```
                else if(url[i]=='/')
                {
                    URL->host[pos]='\0';
                    pos=0;
                    state=3;
                }
                else
                {
                    URL->host[pos]=url[i];
                    pos++;
                }
                break;

            //path until not i<url_lenght
            case 3:
                if(pos>MAX_STRING_SIZE-1) return -1;
                else
                {
                    URL->path[pos]=url[i];
                    pos++;
                }
                break;

            default:
                return -1;
                break;
        }
        i++;
    }

    /*printf("user:%s\n", URL->user);
    printf("pass:%s\n", URL->pass);
    printf("host:%s\n", URL->host);
    printf("path:%s\n", URL->path);*/
    URL->path[pos]='\0';
    if(state!=3 || pos==0) return -1;

    return 0;
}

int get_file_name(char* path, char* file_name)
{
    printf("----------------get_file_name\n");
```

```
    const char ch1='/';
    if(strrchr(path, ch1)==NULL) strcpy(file_name, path);
    else strcpy(file_name, strrchr(path, ch1)+1);


    printf("strlen(file_name)=%ld\n", strlen(file_name));
    if(strlen(file_name)==0) return -1;
    else return 0;
}

int read_reply(int sockfd, char* reply_code)
{
    printf("----------------read_reply\n");

    int i=0, res=0, state=0, done=0;
    char c;

    while(done!=1)
    {
        res+=read(sockfd, &c, 1);
        //else if()
        printf("%c", c);
        switch (state)
        {
            case 0:
                if(c>=48 && c<=57)
                {
                    reply_code[i]=c;
                    i++;
                    state=1;
                }
                break;

            case 1:
                if(c>=48 && c<=57)
                {
                    reply_code[i]=c;
                    i++;
                    state=2;
                }
                else
                {
                    i=0;
```

```
                    state=0;
                }
                break;

            case 2:
                if(c>=48 && c<=57)
                {
                    reply_code[i]=c;
                    i++;
                    state=3;
                }
                else
                {
                    i=0;
                    state=0;
                }
                break;
            case 3: //if read 3 digits, if - go to state 4 and ignore
everything until new line, else done in \n
                if(c=='-') state=4;
                else state=5;
                break;
            case 4:
                if(c=='\n')
                {
                    i=0;
                    state=0;
                }
                break;
            case 5:
                if(c=='\n') done=1;
                break;

            default:
                return -1;
                break;

        }
    }
    //printf("strlen(reply_code):%ld\n", strlen(reply_code));
    //printf("reply_code:%s\n", reply_code);

    return res;
```

```
}

int read_reply_pasv(int sockfd, char* reply_code, char* num1, char*
num2)
{
    printf("----------------read_reply\n");

    int i=0, j=0, k=0, res=0, state=0, done=0;
    char c;

    while(done!=1)
    {
        res+=read(sockfd, &c, 1);
        //else if()
        printf("%c", c);
        switch (state)
        {
            case 0:
                if(c>=48 && c<=57)
                {
                    reply_code[i]=c;
                    i++;
                    state=1;
                }
                break;

            case 1:
                if(c>=48 && c<=57)
                {
                    reply_code[i]=c;
                    i++;
                    state=2;
                }
                else
                {
                    i=0;
                    state=0;
                }
                break;

            case 2:
                if(c>=48 && c<=57)
                {
```

```
            reply_code[i]=c;
            i++;
            state=3;
        }
        else
        {
            i=0;
            state=0;
        }
        break;
    case 3: //if read 3 digits, if 4x ',' go to state 4
        if(c==',') j++;

        if(j==4) state=4;
        break;
    case 4:
        if(c>=48 && c<=57)
        {
            num1[k]=c;
            k++;
        }
        else if(c==',')
        {
            num1[k]='\0';
            k=0;
            state=5;
        }

        break;
    case 5:
        if(c>=48 && c<=57)
        {
            num2[k]=c;
            k++;
        }
        else if(c=='\n')
        {
            num2[k]='\0';
            done=1;
        }
        break;

    default:
```

```
                return -1;
                break;

        }
    }
    //printf("strlen(reply_code):%ld\n", strlen(reply_code));
    //printf("reply_code:%s\n", reply_code);
    //puts(num1);
    //puts(num2);
    return res;
}

int send_msg(int sockfd, char* type, char* content)
{
    printf("----------------send_msg\n");
    int res=0, size=0;

    printf("type: %s | content: %s\n", type, content);

    if(content!=NULL) size=strlen(type)+strlen(content)+2+1;
    else size=strlen(type)+2+1;

    printf("size:%d\n", size);

    //size: +2 for \r\n, +1 for \0. \0 done by strcat.
    char* msg=(char*)calloc(size, sizeof(char)); if(msg==NULL)
{{printf("----------------ERROR: Malloc user\n"); return -1;}}


    strcpy(msg, type);
    if(content!=NULL) strcat(msg, content);
    strcat(msg, "\r\n");

    printf("msg: %s | strlen(msg)=%ld\n", msg, strlen(msg));

    res=write(sockfd, msg, strlen(msg));

    return res;
}

int main(int argc, char* argv[])
{
    if(argc!=2)
```

```
    {
        printf("\n----------------ERROR: Please pass URL in agurment in
format ftp://<user>:<password>@<host>/<url-path> \n");
        return -1;
    }

    struct URL_struct URL;
    URL.user=(char*)malloc(MAX_STRING_SIZE*sizeof(char));
if(URL.user==NULL) {printf("----------------ERROR: Malloc user\n");
return -1;}
    URL.pass=(char*)malloc(MAX_STRING_SIZE*sizeof(char));
if(URL.pass==NULL) {printf("----------------ERROR: Malloc pass\n");
return -1;}
    URL.host=(char*)malloc(MAX_STRING_SIZE*sizeof(char));
if(URL.host==NULL) {printf("----------------ERROR: Malloc host\n");
return -1;}
    URL.path=(char*)malloc(MAX_STRING_SIZE*sizeof(char));
if(URL.path==NULL) {printf("----------------ERROR: Malloc path\n");
return -1;}

    puts(argv[1]);

    if(processURL(argv[1], &URL)!=0)
    {
        printf("\n----------------ERROR: Please pass valid URL in
agurment in format ftp://<user>:<password>@<host>/<url-path> \n");
        return -1;
    }

                                                    //+1 for \0
    URL.user=(char*)realloc(URL.user, strlen(URL.user)+1);
if(URL.user==NULL) {printf("----------------ERROR: Realloc user\n");
return -1;}
    URL.pass=(char*)realloc(URL.pass, strlen(URL.pass)+1);
if(URL.pass==NULL) {printf("----------------ERROR: Realloc pass\n");
return -1;}
    URL.host=(char*)realloc(URL.host, strlen(URL.host)+1);
if(URL.host==NULL) {printf("----------------ERROR: Realloc host\n");
return -1;}
    URL.path=(char*)realloc(URL.path, strlen(URL.path)+1);
if(URL.path==NULL) {printf("----------------ERROR: Realloc path\n");
return -1;}
```

```
    printf("URL.user:%s | strlen(URL.user)=%ld\n", URL.user,
strlen(URL.user));
    printf("URL.pass:%s | strlen(URL.pass)=%ld\n", URL.pass,
strlen(URL.user));
    printf("URL.host:%s | strlen(URL.host)=%ld\n", URL.host,
strlen(URL.user));
    printf("URL.path:%s | strlen(URL.path)=%ld\n", URL.path,
strlen(URL.user));

    //get ip
    struct hostent *h;
    /*
        struct hostent {
            char    *h_name;    Official name of the host.
                char    **h_aliases;    A NULL-terminated array of
alternate names for the host.
            int     h_addrtype; The type of address being returned;
usually AF_INET.
                int     h_length;   The length of the address in bytes.
            char    **h_addr_list;  A zero-terminated array of network
addresses for the host.
                        Host addresses are in Network Byte Order.
        };

        #define h_addr h_addr_list[0]   The first address in
h_addr_list.
    */

    h=gethostbyname(URL.host);
    if(h==NULL)
    {
        printf("\n----------------ERROR: gethostname\n");
        return -1;
    }

    printf("h->h_name: %s\n", h->h_name);
    printf("h->h_addr: %s\n", inet_ntoa(*((struct in_addr *)h-
>h_addr))); //inet_ntoa() function converts the Internet host address
in, given in network byte order, to a string in IPv4 dotted-decimal
notation

    //open connection
```

```
    int sockfd;
    struct  sockaddr_in server_addr;
    /*
        struct sockaddr_in {
            sa_family_t    sin_family; address family: always AF_INET
            in_port_t      sin_port;   port in network byte order
            struct in_addr sin_addr;    internet address
        };

           /Internet address
           struct in_addr {
               uint32_t       s_addr;     address in network byte order
            };
    */

    //server address handling
    //AF_INET is IPv4 protocols
    //htonl() function converts the unsigned integer hostlong from host
byte order to network byte order.

    bzero((char*)&server_addr, sizeof(server_addr)); // delete
sizeof(server_adrr) bytes in pointer &server_addr
    server_addr.sin_family = AF_INET;
    server_addr.sin_addr.s_addr = inet_addr(inet_ntoa(*((struct in_addr
*)h->h_addr))); //32 bit Internet address network byte ordered
    server_addr.sin_port = htons(DEFAULT_FTP_PORT);     //server TCP
port must be network byte ordered.

    //open an TCP socket
    //socket() creates an endpoint for communication and returns a file
descriptor
    //int socket(int domain, int type, int protocol);
    //The domain argument specifies a communication domain; this
selects the protocol family. AF_INET is IPv4 protocols
    //SOCK_STREAM provides sequenced, reliable, two-way, connection-
based byte streams
    //The protocol specifies a particular protocol to be used with the
socket.  Normally only a single protocol exists to support a particular
socket type within a given protocol family, in which case protocol can
be specified as 0.

    if ((sockfd = socket(AF_INET, SOCK_STREAM, 0)) < 0) {printf("------
----------ERROR: socket\n"); close(sockfd); return -1;}
```

```
    //connect to the server
    //int connect(int sockfd, const struct sockaddr *addr, socklen_t
addrlen);
    //The connect() system call connects the socket referred to by the
file descriptor sockfd to the address specified by addr.
    /*
        struct sockaddr {
            sa_family_t sa_family;
            char        sa_data[14];
        }~
    */

    if(connect(sockfd, (struct sockaddr *)&server_addr,
sizeof(server_addr)) < 0) {printf("----------------ERROR: connect\n");
close(sockfd); return -1;}


    //https://en.wikipedia.org/wiki/List_of_FTP_server_return_codes

    int res=0;
    char reply_code1[4]=""; //4 for \0, compiler does it,
https://stackoverflow.com/questions/8202897/null-terminated-string-in-c

    res=read_reply(sockfd, reply_code1);
    if(res<0) {printf("----------------ERROR: read_reply\n");
close(sockfd); return -1;}
    printf("reply_code1:%s\n", reply_code1);
    if(strcmp(reply_code1, "220\0")!=0) {printf("----------------ERROR:
220 not recieved\n"); close(sockfd); return -1;}



    //ALL SENT AND RECIEVED END IN \r\n
    //send user

    res=send_msg(sockfd, "user ", URL.user);
    if(res<0) {printf("----------------ERROR: send_msg\n");
close(sockfd); return -1;}
    printf("Written bytes: %d\n", res);

    char reply_code2[4]="";
    res=read_reply(sockfd, reply_code2);
```

```
    if(res<0) {printf("----------------ERROR: read_reply\n");
close(sockfd); return -1;}

    printf("reply_code2:%s\n", reply_code2);
    if(strcmp(reply_code2, "331\0")!=0) {printf("----------------ERROR:
331 not recieved\n"); close(sockfd); return -1;}

    //send pass

    res=send_msg(sockfd, "pass ", URL.pass);
    if(res<0) {printf("----------------ERROR: send_msg\n");
close(sockfd); return -1;}
    printf("Written bytes: %d\n", res);

    char reply_code3[4]="";
    res=read_reply(sockfd, reply_code3);
    if(res<0) {printf("----------------ERROR: read_reply\n");
close(sockfd); return -1;}

    printf("reply_code3:%s\n", reply_code3);
    if(strcmp(reply_code3, "230\0")!=0) {printf("----------------ERROR:
230 not recieved\n"); close(sockfd); return -1;}

    //send pasv

    res=send_msg(sockfd, "pasv ", NULL);
    if(res<0) {printf("----------------ERROR: send_msg\n");
close(sockfd); return -1;}
    printf("Written bytes: %d\n", res);

    char reply_code4[4]="";
    char* num1, *num2;
    num1=(char*)malloc(4*sizeof(char)); if(num1==NULL) {printf("-------
---------ERROR: Malloc num1\n"); close(sockfd); return -1;}
    num2=(char*)malloc(4*sizeof(char)); if(num2==NULL) {printf("-------
---------ERROR: Malloc num2\n"); close(sockfd); return -1;}

    res=read_reply_pasv(sockfd, reply_code4, num1, num2);

    num1=(char*)realloc(num1, strlen(num1)+1); if(num1==NULL)
{printf("----------------ERROR: Realloc num1\n"); close(sockfd); return
-1;}
```

```
    num2=(char*)realloc(num2, strlen(num2)+1); if(num2==NULL)
{printf("----------------ERROR: Realloc num2\n"); close(sockfd); return
-1;}

    if(res<0) {printf("----------------ERROR: read_reply\n"); return -
1;}

    printf("reply_code4:%s\n", reply_code4);
    if(strcmp(reply_code4, "227\0")!=0) {printf("----------------ERROR:
227 not recieved\n"); return -1;}

    printf("num1:%s | strlen(num1)=%ld | atoi(num1)=%d\n", num1,
strlen(num1), atoi(num1));
    printf("num2:%s | strlen(num2)=%ld | atoi(num2)=%d\n", num2,
strlen(num2), atoi(num2));

    if(strcmp(reply_code4, "227\0")!=0) {printf("----------------ERROR:
227 not recieved\n"); return -1;}

    int new_port=atoi(num1)*256+atoi(num2);
    printf("port_client:%d\n", new_port);

    //connect client to new_port

    int sockfd_new;
    struct  sockaddr_in new_addr;

    bzero((char*)&new_addr, sizeof(new_addr)); // delete
sizeof(server_adrr) bytes in pointer &server_addr
    new_addr.sin_family = AF_INET;
    new_addr.sin_addr.s_addr = inet_addr(inet_ntoa(*((struct in_addr
*)h->h_addr)));    //32 bit Internet address network byte ordered
    new_addr.sin_port = htons(new_port);        //server TCP port must
be network byte ordered.

    if((sockfd_new = socket(AF_INET, SOCK_STREAM, 0)) < 0) {printf("---
-------------ERROR: socket\n"); close(sockfd); close(sockfd_new);
return -1;}
    if(connect(sockfd_new, (struct sockaddr *)&new_addr,
sizeof(new_addr)) < 0) {printf("----------------ERROR: connect\n");
close(sockfd); close(sockfd_new); return -1;}

    //send retr
```

```
    res=send_msg(sockfd, "retr ", URL.path);
    if(res<0) {printf("----------------ERROR: send_msg\n"); return -1;}
    printf("Written bytes: %d\n", res);

    char reply_code5[4]="";
    res=read_reply(sockfd, reply_code5);
    if(res<0) {printf("----------------ERROR: read_reply\n"); return -
1;}

    printf("reply_code5:%s\n", reply_code5);
    if(strcmp(reply_code5, "150\0")!=0) {printf("----------------ERROR:
150 not recieved\n"); return -1;}

    //filename

    char* file_name;
    file_name=(char*) malloc(MAX_STRING_SIZE*sizeof(char));
if(file_name==NULL) {printf("----------------ERROR: Malloc
file_name\n"); close(sockfd); close(sockfd_new); return -1;}

    if(get_file_name(URL.path, file_name)<0) {printf("----------------
ERROR: get_file_name. Please enter valid path with file_name in format
path/file_name\n"); return -1;}
    file_name=(char*)realloc(file_name, strlen(file_name)+1);
if(file_name==NULL) {printf("----------------ERROR: Realloc
file_name.\n"); close(sockfd); close(sockfd_new); return -1;}

    printf("file_name:%s\n", file_name);

    //download

    FILE* fd;
    fd=fopen(file_name, "w");
    if(fd==NULL)  {printf("----------------ERROR: fopen\n");
close(sockfd); close(sockfd_new); return -1;}

    char* packet;
    packet=(char*) malloc(MAX_STRING_SIZE*sizeof(char));
if(packet==NULL) {printf("----------------ERROR: Malloc packet\n");
close(sockfd); close(sockfd_new); return -1;}

    int fd_size=0;
```

```
    res=0;
    while(res=read(sockfd_new, packet, MAX_STRING_SIZE))
fd_size+=fwrite(packet, sizeof(char), res, fd);

    printf("fd_size:%d\n", fd_size);

    close(sockfd);
    close(sockfd_new);
    fclose(fd);

    return 0;
}
```

Anexo 2 - Comandos de configuração de Rede

Experiência 1:

No *tux3*:
> ifconfig eth0 up
> ifconfig eth0 172.16.20.1/24
> ifconfig eth0

No *tux4*:
> ifconfig eth0 up
> ifconfig eth0 172.16.20.254/24
>ifconfig eth0

No *tux3*:
> arp -d 172.16.20.254
> ping 172.16.20.254

Experiência 2:

No *tux2*:
> ifconfig eth0 up
>ifconfig eth0 172.16.21.0/24
>ifconfig eth0

GTK Term - configuração de vlans no switch:
> vlan 20
> end
> show vlan id 20
> configure terminal
> interface fastethernet 0/3
> switchport mode access

> switchport access vlan 20
> end
> show running-config interface fastethernet 0/3
> show interfaces fastethernet 0/3 switchport
> configure terminal
> interface fastethernet 0/4
> switchport mode access
> switchport access vlan 20
> end
> show running-config interface fastethernet 0/4
> show interfaces fastethernet 0/4 switchport
> show vlan id 20
> configure terminal
> vlan 21
> end
> show vlan id 21
> configure terminal
> interface fastethernet 0/2
> switchport mode access
> switchport access vlan 21
> end
> show running-config interface fastethernet 0/2
> show interfaces fastethernet 0/2 switchport

No *tux3*:
> ping 172.16.20.254
> ping 172.16.21.1
> ping -b 172.16.20.255

No *tux2*:
> ping -b 172.16.21.255

Experiência 3:

No *tux4*:
> ifconfig eth1 up
> ifconfig eth1 172.16.21.253/24
> ifconfig eth1
> echo 1 > /proc/sys/net/ipv4/ip_forward
> echo 0 > /proc/sys/net/ipv4/icmp_echo_ignore_broadcasts

GTK Term - adicionar *tux4* à vlan 21:
> configure terminal
> interface fastethernet 0/5
> switchport mode access
> switchport access vlan 21
> end
> show running-config interface fastethernet 0/5

> show interfaces fastethernet 0/5

No *tux3*:
> route add -net 172.16.21.0/24 gw 172.16.20.254
> route -n

No *tux2*:
> route add -net 172.16.20.0/24 gw 172.16.21.253
> route -n

No *tux4*:
> route -n

No *tux3*:
> ping 172.16.20.254
> ping 172.16.21.253
> ping 172.16.21.1

```
root@gnu22:~# route -n
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
172.16.20.0     172.16.21.253   255.255.255.0   UG    0      0        0 eth0
172.16.21.0     0.0.0.0         255.255.255.0   U     0      0        0 eth0
```

```
root@gnu23:~# route -n
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
172.16.20.0     0.0.0.0         255.255.255.0   U     0      0        0 eth0
172.16.21.0     172.16.20.254   255.255.255.0   UG    0      0        0 eth0
```

```
root@gnu24:~# route -n
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
172.16.20.0     0.0.0.0         255.255.255.0   U     0      0        0 eth0
172.16.21.0     0.0.0.0         255.255.255.0   U     0      0        0 eth1
```

Experiência 4:

GTK Term - Configuração Router:
> interface fastethernet 0/0
> ip address 172.16.21.254 255.255.255.0
> no shutdown
> exit
> show interface fastethernet 0/0
> interface fastethernet 0/1
> ip address 172.16.1.29 255.255.255.0
> no shutdown
> exit
> show interface fastethernet 0/1
> ip route 0.0.0.0 0.0.0.0 172.16.1.254
> ip route 172.16.20.0 255.255.255.0 172.16.21.253

GTK Term - Configuração do switch:
> configure terminal
> interface fastethernet 0/10
> switchport mode access
> switchport access vlan 21
> end
> show vlan

No *tux2*:
> route add -net 172.16.1.0/24 gw 172.16.21.254
> route add default gw 172.16.21.254

No *tux3*:
> route add default gw 172.16.20.254

No *tux4*:
> route add -net 172.16.1.0/24 gw 172.16.21.254
> route add default gw 172.16.21.254

No *tux2*:
> route -n

No *tux3*:
> route -n

No *tux4*:
> route -n

No *tux3*:
> ping 172.16.20.254
> ping 172.16.21.253
> ping 172.16.21.1
> ping 172.16.21.254
> ping 172.16.1.29

No *tux2*:
> echo 0 > /proc/sys/net/ipv4/conf/eth0/accept_redirects
> echo 0 > /proc/sys/net/ipv4/conf/all/accept_redirects
> route -n
> route del -net 172.16.20.0/24 gw 172.16.21.253
> route -n
> ping 172.16.20.1
> traceroute 172.16.20.1
> route add -net 172.16.20.0/24 gw 172.16.21.253
> route -n
> traceroute 172.16.20.1
> route -n

> route del -net 172.16.20.0/24 gw 172.16.21.253
> route -n
> echo 1 > /proc/sys/net/ipv4/conf/eth0/accept_redirects
> echo 1 > /proc/sys/net/ipv4/conf/all/accept_redirects
> ping 172.16.20.1

No *tux3*:
> ping 172.16.1.254

GTK Term - configurar Router (adicionar NAT):
> conf t
> interface fastethernet 0/0
> ip address 172.16.21.254 255.255.255.0
> no shutdown
> ip nat inside
> exit
> interface fastethernet 0/1
> ip address 172.16.1.29 255.255.255.0
> no shutdown
> ip nat outside
> exit
> ip nat pool ovrld 172.16.1.29 172.16.1.29 prefix 24
> ip nat inside source list 1 pool ovrld overload
> access-list 1 permit 172.16.20.0 0.0.0.7
> access-list 1 permit 172.16.21.0 0.0.0.7
> ip route 0.0.0.0 0.0.0.0 172.16.1.254
> ip route 172.16.20.0 255.255.255.0 172.16.21.253
> end

No *tux3*:
> ping 172.16.1.254

Experiência 5:

No *tux2*:
> vi /etc/resolv.conf
> nameserver 172.16.2.1

No *tux*3:
> vi /etc/resolv.conf
> nameserver 172.16.2.1
> ping ftp.up.pt
> ping google.com

No *tux3* (no *Browser*):
> www.youtube.com

Anexo 3 - Capturas de Wireshark

Experiência 1

Ping do *tux3* para o *tux4*:

No.	Time	Source	Destination	Protocol	Length	Src Port	Dest Port	Info
16	6.774320674	HewlettP_a7:26:a2	HewlettP_5a:78:c7	ARP	60			Who has 172.16.20.1? Tell 172.16.20.254
17	6.774338832	HewlettP_5a:78:c7	HewlettP_a7:26:a2	ARP	42			172.16.20.1 is at 00:21:5a:5a:78:c7
20	7.027303542	HewlettP_5a:78:c7	HewlettP_a7:26:a2	ARP	42			Who has 172.16.20.254? Tell 172.16.20.1
21	7.027401390	HewlettP_a7:26:a2	HewlettP_5a:78:c7	ARP	60			172.16.20.254 is at 00:22:64:a7:26:a2

Ping do *tux3* para o *tux4*:

No.	Time	Source	Destination	Protocol	Length	Src Port	Dest Port	Info
3	1.771985949	172.16.20.1	172.16.20.254	ICMP	98			Echo (ping) request id=0x0475, seq=1/256, ttl=64 (reply in 4)
4	1.772148889	172.16.20.254	172.16.20.1	ICMP	98			Echo (ping) reply id=0x0475, seq=1/256, ttl=64 (request in 3)
6	2.803336933	172.16.20.1	172.16.20.254	ICMP	98			Echo (ping) request id=0x0475, seq=2/512, ttl=64 (reply in 7)
7	2.803468724	172.16.20.254	172.16.20.1	ICMP	98			Echo (ping) reply id=0x0475, seq=2/512, ttl=64 (request in 6)
8	3.827334398	172.16.20.1	172.16.20.254	ICMP	98			Echo (ping) request id=0x0475, seq=3/768, ttl=64 (reply in 9)
9	3.827470030	172.16.20.254	172.16.20.1	ICMP	98			Echo (ping) reply id=0x0475, seq=3/768, ttl=64 (request in 8)
11	4.851336892	172.16.20.1	172.16.20.254	ICMP	98			Echo (ping) request id=0x0475, seq=4/1024, ttl=64 (reply in 12)
12	4.851491381	172.16.20.254	172.16.20.1	ICMP	98			Echo (ping) reply id=0x0475, seq=4/1024, ttl=64 (request in 11)
13	5.875336312	172.16.20.1	172.16.20.254	ICMP	98			Echo (ping) request id=0x0475, seq=5/1280, ttl=64 (reply in 14)
14	5.875466916	172.16.20.254	172.16.20.1	ICMP	98			Echo (ping) reply id=0x0475, seq=5/1280, ttl=64 (request in 13)
18	6.899329657	172.16.20.1	172.16.20.254	ICMP	98			Echo (ping) request id=0x0475, seq=6/1536, ttl=64 (reply in 19)
19	6.899453905	172.16.20.254	172.16.20.1	ICMP	98			Echo (ping) reply id=0x0475, seq=6/1536, ttl=64 (request in 18)
22	7.923336341	172.16.20.1	172.16.20.254	ICMP	98			Echo (ping) request id=0x0475, seq=7/1792, ttl=64 (reply in 23)
23	7.923466805	172.16.20.254	172.16.20.1	ICMP	98			Echo (ping) reply id=0x0475, seq=7/1792, ttl=64 (request in 22)
25	8.947335762	172.16.20.1	172.16.20.254	ICMP	98			Echo (ping) request id=0x0475, seq=8/2048, ttl=64 (reply in 26)
26	8.947465806	172.16.20.254	172.16.20.1	ICMP	98			Echo (ping) reply id=0x0475, seq=8/2048, ttl=64 (request in 25)
27	9.971330433	172.16.20.1	172.16.20.254	ICMP	98			Echo (ping) request id=0x0475, seq=9/2304, ttl=64 (reply in 28)
28	9.971484293	172.16.20.254	172.16.20.1	ICMP	98			Echo (ping) reply id=0x0475, seq=9/2304, ttl=64 (request in 27)
31	10.995333…	172.16.20.1	172.16.20.254	ICMP	98			Echo (ping) request id=0x0475, seq=10/2560, ttl=64 (reply in 32)
32	10.995465…	172.16.20.254	172.16.20.1	ICMP	98			Echo (ping) reply id=0x0475, seq=10/2560, ttl=64 (request in 31)
33	12.019333…	172.16.20.1	172.16.20.254	ICMP	98			Echo (ping) request id=0x0475, seq=11/2816, ttl=64 (reply in 34)
34	12.019464…	172.16.20.254	172.16.20.1	ICMP	98			Echo (ping) reply id=0x0475, seq=11/2816, ttl=64 (request in 33)
36	13.043335…	172.16.20.1	172.16.20.254	ICMP	98			Echo (ping) request id=0x0475, seq=12/3072, ttl=64 (reply in 37)
37	13.043466…	172.16.20.254	172.16.20.1	ICMP	98			Echo (ping) reply id=0x0475, seq=12/3072, ttl=64 (request in 36)

Experiência 2

2.5 - *Ping* do *tux3* para o *tux4* e *tux2*:

No.	Time	Source	Destination	Protocol	Length	Src Port	Dest Port	Info
2	1.121599122	172.16.20.1	172.16.20.254	ICMP	98			Echo (ping) request id=0x09c7, seq=1/256, ttl=64 (reply in 3)
3	1.121767370	172.16.20.254	172.16.20.1	ICMP	98			Echo (ping) reply id=0x09c7, seq=1/256, ttl=64 (request in 2)
6	2.147727724	172.16.20.1	172.16.20.254	ICMP	98			Echo (ping) request id=0x09c7, seq=2/512, ttl=64 (reply in 7)
7	2.147857908	172.16.20.254	172.16.20.1	ICMP	98			Echo (ping) reply id=0x09c7, seq=2/512, ttl=64 (request in 6)
8	3.171729380	172.16.20.1	172.16.20.254	ICMP	98			Echo (ping) request id=0x09c7, seq=3/768, ttl=64 (reply in 9)
9	3.171858167	172.16.20.254	172.16.20.1	ICMP	98			Echo (ping) reply id=0x09c7, seq=3/768, ttl=64 (request in 8)
11	4.195718115	172.16.20.1	172.16.20.254	ICMP	98			Echo (ping) request id=0x09c7, seq=4/1024, ttl=64 (reply in 12)
12	4.195843410	172.16.20.254	172.16.20.1	ICMP	98			Echo (ping) reply id=0x09c7, seq=4/1024, ttl=64 (request in 11)
13	5.219701472	172.16.20.1	172.16.20.254	ICMP	98			Echo (ping) request id=0x09c7, seq=5/1280, ttl=64 (reply in 14)
14	5.219826348	172.16.20.254	172.16.20.1	ICMP	98			Echo (ping) reply id=0x09c7, seq=5/1280, ttl=64 (request in 13)
16	6.243728130	172.16.20.1	172.16.20.254	ICMP	98			Echo (ping) request id=0x09c7, seq=6/1536, ttl=64 (reply in 17)
17	6.243880315	172.16.20.254	172.16.20.1	ICMP	98			Echo (ping) reply id=0x09c7, seq=6/1536, ttl=64 (request in 16)
22	7.267769107	172.16.20.1	172.16.20.254	ICMP	98			Echo (ping) request id=0x09c7, seq=7/1792, ttl=64 (reply in 23)
23	7.267901805	172.16.20.254	172.16.20.1	ICMP	98			Echo (ping) reply id=0x09c7, seq=7/1792, ttl=64 (request in 22)
25	8.291758749	172.16.20.1	172.16.20.254	ICMP	98			Echo (ping) request id=0x09c7, seq=8/2048, ttl=64 (reply in 26)
26	8.291886908	172.16.20.254	172.16.20.1	ICMP	98			Echo (ping) reply id=0x09c7, seq=8/2048, ttl=64 (request in 25)
27	9.315726183	172.16.20.1	172.16.20.254	ICMP	98			Echo (ping) request id=0x09c7, seq=9/2304, ttl=64 (reply in 28)
28	9.315860697	172.16.20.254	172.16.20.1	ICMP	98			Echo (ping) reply id=0x09c7, seq=9/2304, ttl=64 (request in 27)
30	10.339714…	172.16.20.1	172.16.20.254	ICMP	98			Echo (ping) request id=0x09c7, seq=10/2560, ttl=64 (reply in 31)
31	10.339840…	172.16.20.254	172.16.20.1	ICMP	98			Echo (ping) reply id=0x09c7, seq=10/2560, ttl=64 (request in 30)
32	11.363726…	172.16.20.1	172.16.20.254	ICMP	98			Echo (ping) request id=0x09c7, seq=11/2816, ttl=64 (reply in 33)
33	11.363884…	172.16.20.254	172.16.20.1	ICMP	98			Echo (ping) reply id=0x09c7, seq=11/2816, ttl=64 (request in 32)

2.8 - *Ping broadcast* no *tux3*:

No.	Time	Source	Destination	Protocol	Length	Src Port	Dest Port	Info
12	14.684498…	172.16.20.1	172.16.20.255	ICMP	98			Echo (ping) request id=0x0b2c, seq=1/256, ttl=64 (no response found!)
13	15.697861…	172.16.20.1	172.16.20.255	ICMP	98			Echo (ping) request id=0x0b2c, seq=2/512, ttl=64 (no response found!)
15	16.721860…	172.16.20.1	172.16.20.255	ICMP	98			Echo (ping) request id=0x0b2c, seq=3/768, ttl=64 (no response found!)
16	17.745860…	172.16.20.1	172.16.20.255	ICMP	98			Echo (ping) request id=0x0b2c, seq=4/1024, ttl=64 (no response found!)
18	18.769868…	172.16.20.1	172.16.20.255	ICMP	98			Echo (ping) request id=0x0b2c, seq=5/1280, ttl=64 (no response found!)
19	19.793860…	172.16.20.1	172.16.20.255	ICMP	98			Echo (ping) request id=0x0b2c, seq=6/1536, ttl=64 (no response found!)
22	20.817866…	172.16.20.1	172.16.20.255	ICMP	98			Echo (ping) request id=0x0b2c, seq=7/1792, ttl=64 (no response found!)
23	21.841857…	172.16.20.1	172.16.20.255	ICMP	98			Echo (ping) request id=0x0b2c, seq=8/2048, ttl=64 (no response found!)
25	22.865865…	172.16.20.1	172.16.20.255	ICMP	98			Echo (ping) request id=0x0b2c, seq=9/2304, ttl=64 (no response found!)
26	23.889865…	172.16.20.1	172.16.20.255	ICMP	98			Echo (ping) request id=0x0b2c, seq=10/2560, ttl=64 (no response found!)
28	24.917857…	172.16.20.1	172.16.20.255	ICMP	98			Echo (ping) request id=0x0b2c, seq=11/2816, ttl=64 (no response found!)
29	25.937858…	172.16.20.1	172.16.20.255	ICMP	98			Echo (ping) request id=0x0b2c, seq=12/3072, ttl=64 (no response found!)
31	26.961861…	172.16.20.1	172.16.20.255	ICMP	98			Echo (ping) request id=0x0b2c, seq=13/3328, ttl=64 (no response found!)

2.8 - Captura no *tux4*:

No.	Time	Source	Destination	Protocol	Length	Src Port	Dest Port	Info
24	36.166977…	172.16.20.1	172.16.20.255	ICMP	98			Echo (ping) request id=0x0b2c, seq=1/256, ttl=64 (no response found!)
25	37.180343…	172.16.20.1	172.16.20.255	ICMP	98			Echo (ping) request id=0x0b2c, seq=2/512, ttl=64 (no response found!)
27	38.204339…	172.16.20.1	172.16.20.255	ICMP	98			Echo (ping) request id=0x0b2c, seq=3/768, ttl=64 (no response found!)
28	39.228348…	172.16.20.1	172.16.20.255	ICMP	98			Echo (ping) request id=0x0b2c, seq=4/1024, ttl=64 (no response found!)
30	40.252352…	172.16.20.1	172.16.20.255	ICMP	98			Echo (ping) request id=0x0b2c, seq=5/1280, ttl=64 (no response found!)
31	41.276352…	172.16.20.1	172.16.20.255	ICMP	98			Echo (ping) request id=0x0b2c, seq=6/1536, ttl=64 (no response found!)
34	42.300359…	172.16.20.1	172.16.20.255	ICMP	98			Echo (ping) request id=0x0b2c, seq=7/1792, ttl=64 (no response found!)
35	43.324349…	172.16.20.1	172.16.20.255	ICMP	98			Echo (ping) request id=0x0b2c, seq=8/2048, ttl=64 (no response found!)
37	44.348352…	172.16.20.1	172.16.20.255	ICMP	98			Echo (ping) request id=0x0b2c, seq=9/2304, ttl=64 (no response found!)
38	45.372359…	172.16.20.1	172.16.20.255	ICMP	98			Echo (ping) request id=0x0b2c, seq=10/2560, ttl=64 (no response found!)
40	46.400345…	172.16.20.1	172.16.20.255	ICMP	98			Echo (ping) request id=0x0b2c, seq=11/2816, ttl=64 (no response found!)
41	47.420354…	172.16.20.1	172.16.20.255	ICMP	98			Echo (ping) request id=0x0b2c, seq=12/3072, ttl=64 (no response found!)
43	48.444357…	172.16.20.1	172.16.20.255	ICMP	98			Echo (ping) request id=0x0b2c, seq=13/3328, ttl=64 (no response found!)

2.10 - *Ping broadcast* no *tux2*:

No.	Time	Source	Destination	Protocol	Length	Src Port	Dest Port	Info
13	19.581135…	172.16.21.1	172.16.21.255	ICMP	98			Echo (ping) request id=0x0f9d, seq=1/256, ttl=64 (no response found!)
15	20.586090…	172.16.21.1	172.16.21.255	ICMP	98			Echo (ping) request id=0x0f9d, seq=2/512, ttl=64 (no response found!)
16	21.610085…	172.16.21.1	172.16.21.255	ICMP	98			Echo (ping) request id=0x0f9d, seq=3/768, ttl=64 (no response found!)
18	22.634097…	172.16.21.1	172.16.21.255	ICMP	98			Echo (ping) request id=0x0f9d, seq=4/1024, ttl=64 (no response found!)
19	23.658087…	172.16.21.1	172.16.21.255	ICMP	98			Echo (ping) request id=0x0f9d, seq=5/1280, ttl=64 (no response found!)
21	24.682093…	172.16.21.1	172.16.21.255	ICMP	98			Echo (ping) request id=0x0f9d, seq=6/1536, ttl=64 (no response found!)
23	25.706096…	172.16.21.1	172.16.21.255	ICMP	98			Echo (ping) request id=0x0f9d, seq=7/1792, ttl=64 (no response found!)
26	26.730103…	172.16.21.1	172.16.21.255	ICMP	98			Echo (ping) request id=0x0f9d, seq=8/2048, ttl=64 (no response found!)

Experiência 3

3.6 - *Ping* do *tux3* para o *tux4* (na *vlan* 20):

No.	Time	Source	Destination	Protocol	Length	Src Port	Dest Port	Info
19	27.776803…	172.16.20.1	172.16.20.254	ICMP	98			Echo (ping) request id=0x144a, seq=1/256, ttl=64 (reply in 20)
20	27.776970…	172.16.20.254	172.16.20.1	ICMP	98			Echo (ping) reply id=0x144a, seq=1/256, ttl=64 (request in 19)
22	28.794166…	172.16.20.1	172.16.20.254	ICMP	98			Echo (ping) request id=0x144a, seq=2/512, ttl=64 (reply in 23)
23	28.794301…	172.16.20.254	172.16.20.1	ICMP	98			Echo (ping) reply id=0x144a, seq=2/512, ttl=64 (request in 22)
24	29.818162…	172.16.20.1	172.16.20.254	ICMP	98			Echo (ping) request id=0x144a, seq=3/768, ttl=64 (reply in 25)
25	29.818296…	172.16.20.254	172.16.20.1	ICMP	98			Echo (ping) reply id=0x144a, seq=3/768, ttl=64 (request in 24)
27	30.842164…	172.16.20.1	172.16.20.254	ICMP	98			Echo (ping) request id=0x144a, seq=4/1024, ttl=64 (reply in 28)
28	30.842304…	172.16.20.254	172.16.20.1	ICMP	98			Echo (ping) reply id=0x144a, seq=4/1024, ttl=64 (request in 27)
29	31.866167…	172.16.20.1	172.16.20.254	ICMP	98			Echo (ping) request id=0x144a, seq=5/1280, ttl=64 (reply in 30)
30	31.866300…	172.16.20.254	172.16.20.1	ICMP	98			Echo (ping) reply id=0x144a, seq=5/1280, ttl=64 (request in 29)
32	32.890171…	172.16.20.1	172.16.20.254	ICMP	98			Echo (ping) request id=0x144a, seq=6/1536, ttl=64 (reply in 33)
33	32.890332…	172.16.20.254	172.16.20.1	ICMP	98			Echo (ping) reply id=0x144a, seq=6/1536, ttl=64 (request in 32)
38	33.914173…	172.16.20.1	172.16.20.254	ICMP	98			Echo (ping) request id=0x144a, seq=7/1792, ttl=64 (reply in 39)
39	33.914308…	172.16.20.254	172.16.20.1	ICMP	98			Echo (ping) reply id=0x144a, seq=7/1792, ttl=64 (request in 38)
41	34.938166…	172.16.20.1	172.16.20.254	ICMP	98			Echo (ping) request id=0x144a, seq=8/2048, ttl=64 (reply in 42)
42	34.938301…	172.16.20.254	172.16.20.1	ICMP	98			Echo (ping) reply id=0x144a, seq=8/2048, ttl=64 (request in 41)

3.6 - *Ping* do *tux3* para o *tux4* (na *vlan* 21):

No.	Time	Source	Destination	Protocol	Length	Src Port	Dest Port	Info
47	40.856808…	172.16.20.1	172.16.21.253	ICMP	98			Echo (ping) request id=0x1458, seq=1/256, ttl=64 (reply in 48)
48	40.856970…	172.16.21.253	172.16.20.1	ICMP	98			Echo (ping) reply id=0x1458, seq=1/256, ttl=64 (request in 47)
49	41.882186…	172.16.20.1	172.16.21.253	ICMP	98			Echo (ping) request id=0x1458, seq=2/512, ttl=64 (reply in 50)
50	41.882320…	172.16.21.253	172.16.20.1	ICMP	98			Echo (ping) reply id=0x1458, seq=2/512, ttl=64 (request in 49)
52	42.906169…	172.16.20.1	172.16.21.253	ICMP	98			Echo (ping) request id=0x1458, seq=3/768, ttl=64 (reply in 53)
53	42.906330…	172.16.21.253	172.16.20.1	ICMP	98			Echo (ping) reply id=0x1458, seq=3/768, ttl=64 (request in 52)
54	43.930173…	172.16.20.1	172.16.21.253	ICMP	98			Echo (ping) request id=0x1458, seq=4/1024, ttl=64 (reply in 55)
55	43.930306…	172.16.21.253	172.16.20.1	ICMP	98			Echo (ping) reply id=0x1458, seq=4/1024, ttl=64 (request in 54)
57	44.954164…	172.16.20.1	172.16.21.253	ICMP	98			Echo (ping) request id=0x1458, seq=5/1280, ttl=64 (reply in 58)
58	44.954296…	172.16.21.253	172.16.20.1	ICMP	98			Echo (ping) reply id=0x1458, seq=5/1280, ttl=64 (request in 57)
60	45.978163…	172.16.20.1	172.16.21.253	ICMP	98			Echo (ping) request id=0x1458, seq=6/1536, ttl=64 (reply in 61)
61	45.978296…	172.16.21.253	172.16.20.1	ICMP	98			Echo (ping) reply id=0x1458, seq=6/1536, ttl=64 (request in 60)
63	47.002171…	172.16.20.1	172.16.21.253	ICMP	98			Echo (ping) request id=0x1458, seq=7/1792, ttl=64 (reply in 64)
64	47.002304…	172.16.21.253	172.16.20.1	ICMP	98			Echo (ping) reply id=0x1458, seq=7/1792, ttl=64 (request in 63)

3.6 - *Ping* do *tux3* para o *tux2*:

No.	Time	Source	Destination	Protocol	Length	Src Port	Dest Port	Info
71	56.448644…	172.16.20.1	172.16.21.1	ICMP	98			Echo (ping) request id=0x1462, seq=1/256, ttl=64 (reply in 72)
72	56.448945…	172.16.21.1	172.16.20.1	ICMP	98			Echo (ping) reply id=0x1462, seq=1/256, ttl=63 (request in 71)
73	57.466166…	172.16.20.1	172.16.21.1	ICMP	98			Echo (ping) request id=0x1462, seq=2/512, ttl=64 (reply in 74)
74	57.466432…	172.16.21.1	172.16.20.1	ICMP	98			Echo (ping) reply id=0x1462, seq=2/512, ttl=63 (request in 73)
76	58.490162…	172.16.20.1	172.16.21.1	ICMP	98			Echo (ping) request id=0x1462, seq=3/768, ttl=64 (reply in 77)
77	58.490397…	172.16.21.1	172.16.20.1	ICMP	98			Echo (ping) reply id=0x1462, seq=3/768, ttl=63 (request in 76)
78	59.514163…	172.16.20.1	172.16.21.1	ICMP	98			Echo (ping) request id=0x1462, seq=4/1024, ttl=64 (reply in 79)
79	59.514400…	172.16.21.1	172.16.20.1	ICMP	98			Echo (ping) reply id=0x1462, seq=4/1024, ttl=63 (request in 78)
81	60.538161…	172.16.20.1	172.16.21.1	ICMP	98			Echo (ping) request id=0x1462, seq=5/1280, ttl=64 (reply in 82)
82	60.538410…	172.16.21.1	172.16.20.1	ICMP	98			Echo (ping) reply id=0x1462, seq=5/1280, ttl=63 (request in 81)
83	61.562161…	172.16.20.1	172.16.21.1	ICMP	98			Echo (ping) request id=0x1462, seq=6/1536, ttl=64 (reply in 84)
84	61.562396…	172.16.21.1	172.16.20.1	ICMP	98			Echo (ping) reply id=0x1462, seq=6/1536, ttl=63 (request in 83)

3.10 - *Ping* do *tux3* para o *tux2* - Captura na *network interface eth0*:

No.	Time	Source	Destination	Protocol	Length	Src Port	Dest Port	Info
107	170.28603…	172.16.20.1	172.16.21.1	ICMP	98			Echo (ping) request id=0x168b, seq=1/256, ttl=64 (reply in 108)
108	170.28631…	172.16.21.1	172.16.20.1	ICMP	98			Echo (ping) reply id=0x168b, seq=1/256, ttl=63 (request in 107)
111	171.31015…	172.16.20.1	172.16.21.1	ICMP	98			Echo (ping) request id=0x168b, seq=2/512, ttl=64 (reply in 112)
112	171.31030…	172.16.21.1	172.16.20.1	ICMP	98			Echo (ping) reply id=0x168b, seq=2/512, ttl=63 (request in 111)
113	172.33415…	172.16.20.1	172.16.21.1	ICMP	98			Echo (ping) request id=0x168b, seq=3/768, ttl=64 (reply in 114)
114	172.33433…	172.16.21.1	172.16.20.1	ICMP	98			Echo (ping) reply id=0x168b, seq=3/768, ttl=63 (request in 113)
117	173.35815…	172.16.20.1	172.16.21.1	ICMP	98			Echo (ping) request id=0x168b, seq=4/1024, ttl=64 (reply in 118)
118	173.35830…	172.16.21.1	172.16.20.1	ICMP	98			Echo (ping) reply id=0x168b, seq=4/1024, ttl=63 (request in 117)
119	174.38216…	172.16.20.1	172.16.21.1	ICMP	98			Echo (ping) request id=0x168b, seq=5/1280, ttl=64 (reply in 120)
120	174.38231…	172.16.21.1	172.16.20.1	ICMP	98			Echo (ping) reply id=0x168b, seq=5/1280, ttl=63 (request in 119)

3.10 - *Ping* do *tux3* para o *tux2* - Captura na *network interface eth1* do *tux4*:

No.	Time	Source	Destination	Protocol	Length	Src Port	Dest Port	Info
102	163.6424335…	EncoreNe_c8:7c:55	Broadcast	ARP	42			Who has 172.16.21.1? Tell 172.16.21.253
103	163.6425470…	HewlettP_5a:76:a8	EncoreNe_c8:7c:55	ARP	60			172.16.21.1 is at 00:21:5a:5a:76:a8
118	168.7217491…	HewlettP_5a:76:a8	EncoreNe_c8:7c:55	ARP	60			Who has 172.16.21.253? Tell 172.16.21.1
119	168.7217673…	EncoreNe_c8:7c:55	HewlettP_5a:76:a8	ARP	42			172.16.21.253 is at 00:e0:7d:c8:7c:55

3.10 - *Ping* do *tux3* para o *tux2* - Captura na *network interface eth1* do *tux4*:

No.	Time	Source	Destination	Protocol	Length	Src Port	Dest Port	Info
104	163.64256…	172.16.20.1	172.16.21.1	ICMP	98			Echo (ping) request id=0x168b, seq=1/256, ttl=63 (reply in 105)
105	163.64268…	172.16.21.1	172.16.20.1	ICMP	98			Echo (ping) reply id=0x168b, seq=1/256, ttl=64 (request in 104)
107	164.66655…	172.16.20.1	172.16.21.1	ICMP	98			Echo (ping) request id=0x168b, seq=2/512, ttl=63 (reply in 108)
108	164.66666…	172.16.21.1	172.16.20.1	ICMP	98			Echo (ping) reply id=0x168b, seq=2/512, ttl=64 (request in 107)
109	165.69055…	172.16.20.1	172.16.21.1	ICMP	98			Echo (ping) request id=0x168b, seq=3/768, ttl=63 (reply in 110)
110	165.69069…	172.16.21.1	172.16.20.1	ICMP	98			Echo (ping) reply id=0x168b, seq=3/768, ttl=64 (request in 109)
113	166.71455…	172.16.20.1	172.16.21.1	ICMP	98			Echo (ping) request id=0x168b, seq=4/1024, ttl=63 (reply in 114)
114	166.71466…	172.16.21.1	172.16.20.1	ICMP	98			Echo (ping) reply id=0x168b, seq=4/1024, ttl=64 (request in 113)
115	167.73856…	172.16.20.1	172.16.21.1	ICMP	98			Echo (ping) request id=0x168b, seq=5/1280, ttl=63 (reply in 116)
116	167.73867…	172.16.21.1	172.16.20.1	ICMP	98			Echo (ping) reply id=0x168b, seq=5/1280, ttl=64 (request in 115)

Experiência 4

4.2 - *Ping* do *tux3* para o *tux4* (na *vlan* 20):

No.	Time	Source	Destination	Protocol	Length	Src Port	Dest Port	Info
10	15.212623…	172.16.20.1	172.16.20.254	ICMP	98			Echo (ping) request id=0x26af, seq=1/256, ttl=64 (reply in 11)
11	15.212775…	172.16.20.254	172.16.20.1	ICMP	98			Echo (ping) reply id=0x26af, seq=1/256, ttl=64 (request in 10)
13	16.235222…	172.16.20.1	172.16.20.254	ICMP	98			Echo (ping) request id=0x26af, seq=2/512, ttl=64 (reply in 14)
14	16.235374…	172.16.20.254	172.16.20.1	ICMP	98			Echo (ping) reply id=0x26af, seq=2/512, ttl=64 (request in 13)
15	17.259242…	172.16.20.1	172.16.20.254	ICMP	98			Echo (ping) request id=0x26af, seq=3/768, ttl=64 (reply in 16)
16	17.259377…	172.16.20.254	172.16.20.1	ICMP	98			Echo (ping) reply id=0x26af, seq=3/768, ttl=64 (request in 15)
18	18.283210…	172.16.20.1	172.16.20.254	ICMP	98			Echo (ping) request id=0x26af, seq=4/1024, ttl=64 (reply in 19)
19	18.283338…	172.16.20.254	172.16.20.1	ICMP	98			Echo (ping) reply id=0x26af, seq=4/1024, ttl=64 (request in 18)

4.2 - *Ping* do *tux3* para o *tux4* (na *vlan* 21):

No.	Time	Source	Destination	Protocol	Length	Src Port	Dest Port	Info
38	31.428958…	172.16.20.1	172.16.21.253	ICMP	98			Echo (ping) request id=0x26ce, seq=1/256, ttl=64 (reply in 39)
39	31.429120…	172.16.21.253	172.16.20.1	ICMP	98			Echo (ping) reply id=0x26ce, seq=1/256, ttl=64 (request in 38)
41	32.459248…	172.16.20.1	172.16.21.253	ICMP	98			Echo (ping) request id=0x26ce, seq=2/512, ttl=64 (reply in 42)
42	32.459382…	172.16.21.253	172.16.20.1	ICMP	98			Echo (ping) reply id=0x26ce, seq=2/512, ttl=64 (request in 41)
43	33.483222…	172.16.20.1	172.16.21.253	ICMP	98			Echo (ping) request id=0x26ce, seq=3/768, ttl=64 (reply in 44)
44	33.483350…	172.16.21.253	172.16.20.1	ICMP	98			Echo (ping) reply id=0x26ce, seq=3/768, ttl=64 (request in 43)
46	34.507215…	172.16.20.1	172.16.21.253	ICMP	98			Echo (ping) request id=0x26ce, seq=4/1024, ttl=64 (reply in 47)
47	34.507341…	172.16.21.253	172.16.20.1	ICMP	98			Echo (ping) reply id=0x26ce, seq=4/1024, ttl=64 (request in 46)

4.2 - *Ping* do *tux3* para o *tux2*:

No.	Time	Source	Destination	Protocol	Length	Src Port	Dest Port	Info
55	40.629165…	172.16.20.1	172.16.21.1	ICMP	98			Echo (ping) request id=0x26d5, seq=1/256, ttl=64 (reply in 56)
56	40.629555…	172.16.21.1	172.16.20.1	ICMP	98			Echo (ping) reply id=0x26d5, seq=1/256, ttl=63 (request in 55)
57	41.643227…	172.16.20.1	172.16.21.1	ICMP	98			Echo (ping) request id=0x26d5, seq=2/512, ttl=64 (reply in 58)
58	41.643459…	172.16.21.1	172.16.20.1	ICMP	98			Echo (ping) reply id=0x26d5, seq=2/512, ttl=63 (request in 57)
60	42.667248…	172.16.20.1	172.16.21.1	ICMP	98			Echo (ping) request id=0x26d5, seq=3/768, ttl=64 (reply in 61)
61	42.667486…	172.16.21.1	172.16.20.1	ICMP	98			Echo (ping) reply id=0x26d5, seq=3/768, ttl=63 (request in 60)
62	43.691207…	172.16.20.1	172.16.21.1	ICMP	98			Echo (ping) request id=0x26d5, seq=4/1024, ttl=64 (reply in 63)
63	43.691439…	172.16.21.1	172.16.20.1	ICMP	98			Echo (ping) reply id=0x26d5, seq=4/1024, ttl=63 (request in 62)
65	44.715212…	172.16.20.1	172.16.21.1	ICMP	98			Echo (ping) request id=0x26d5, seq=5/1280, ttl=64 (reply in 66)
66	44.715472…	172.16.21.1	172.16.20.1	ICMP	98			Echo (ping) reply id=0x26d5, seq=5/1280, ttl=63 (request in 65)

4.2 - *Ping* do *tux3* para o *Router* (na *vlan* 21):

No.	Time	Source	Destination	Protocol	Length	Src Port	Dest Port	Info
73	52.037487…	172.16.20.1	172.16.21.254	ICMP	98			Echo (ping) request id=0x26dc, seq=1/256, ttl=64 (reply in 74)
74	52.038296…	172.16.21.254	172.16.20.1	ICMP	98			Echo (ping) reply id=0x26dc, seq=1/256, ttl=254 (request in 73)
76	53.067248…	172.16.20.1	172.16.21.254	ICMP	98			Echo (ping) request id=0x26dc, seq=2/512, ttl=64 (reply in 77)
77	53.068020…	172.16.21.254	172.16.20.1	ICMP	98			Echo (ping) reply id=0x26dc, seq=2/512, ttl=254 (request in 76)
78	54.091207…	172.16.20.1	172.16.21.254	ICMP	98			Echo (ping) request id=0x26dc, seq=3/768, ttl=64 (reply in 79)
79	54.091973…	172.16.21.254	172.16.20.1	ICMP	98			Echo (ping) reply id=0x26dc, seq=3/768, ttl=254 (request in 78)
81	55.115211…	172.16.20.1	172.16.21.254	ICMP	98			Echo (ping) request id=0x26dc, seq=4/1024, ttl=64 (reply in 82)
82	55.115992…	172.16.21.254	172.16.20.1	ICMP	98			Echo (ping) reply id=0x26dc, seq=4/1024, ttl=254 (request in 81)

4.2 - *Ping* do *tux3* para o Router (na parte da *internet*):

No.	Time	Source	Destination	Protocol	Length	Src Port	Dest Port	Info
87	61.981742…	172.16.20.1	172.16.2.29	ICMP	98			Echo (ping) request id=0x26e3, seq=1/256, ttl=64 (reply in 88)
88	61.982574…	172.16.2.29	172.16.20.1	ICMP	98			Echo (ping) reply id=0x26e3, seq=1/256, ttl=254 (request in 87)
90	62.987237…	172.16.20.1	172.16.2.29	ICMP	98			Echo (ping) request id=0x26e3, seq=2/512, ttl=64 (reply in 91)
91	62.988000…	172.16.2.29	172.16.20.1	ICMP	98			Echo (ping) reply id=0x26e3, seq=2/512, ttl=254 (request in 90)
92	64.011233…	172.16.20.1	172.16.2.29	ICMP	98			Echo (ping) request id=0x26e3, seq=3/768, ttl=64 (reply in 93)
93	64.012024…	172.16.2.29	172.16.20.1	ICMP	98			Echo (ping) reply id=0x26e3, seq=3/768, ttl=254 (request in 92)
95	65.035237…	172.16.20.1	172.16.2.29	ICMP	98			Echo (ping) request id=0x26e3, seq=4/1024, ttl=64 (reply in 96)
96	65.036043…	172.16.2.29	172.16.20.1	ICMP	98			Echo (ping) reply id=0x26e3, seq=4/1024, ttl=254 (request in 95)

4.4 - *Traceroute* do *tux3* no *tux2*:

No.	Time	Source	Destination	Protocol	Length	Src Port	Dest Port	Info
11	5.080747958	172.16.21.253	172.16.21.1	ICMP	102	555…	334…	Time-to-live exceeded (Time to live exceeded in transit)
12	5.080760739	172.16.21.253	172.16.21.1	ICMP	102	345…	334…	Time-to-live exceeded (Time to live exceeded in transit)
13	5.080763882	172.16.21.253	172.16.21.1	ICMP	102	423…	334…	Time-to-live exceeded (Time to live exceeded in transit)
24	5.080995968	172.16.20.1	172.16.21.1	ICMP	102	413…	334…	Destination unreachable (Port unreachable)
25	5.081000856	172.16.20.1	172.16.21.1	ICMP	102	539…	334…	Destination unreachable (Port unreachable)
26	5.081003371	172.16.20.1	172.16.21.1	ICMP	102	388…	334…	Destination unreachable (Port unreachable)
27	5.081006095	172.16.20.1	172.16.21.1	ICMP	102	597…	334…	Destination unreachable (Port unreachable)
28	5.081011542	172.16.20.1	172.16.21.1	ICMP	102	471…	334…	Destination unreachable (Port unreachable)
29	5.081032076	172.16.20.1	172.16.21.1	ICMP	102	458…	334…	Destination unreachable (Port unreachable)

4.4 - *Ping* do *tux2* para o *tux3*:

No.	Time	Source	Destination	Protocol	Length	Src Port	Dest Port	Info
6	7.798933709	172.16.21.1	172.16.20.1	ICMP	98			Echo (ping) request id=0x7418, seq=1/256, ttl=64 (reply in 8)
8	7.799969817	172.16.20.1	172.16.21.1	ICMP	98			Echo (ping) reply id=0x7418, seq=1/256, ttl=63 (request in 6)
10	8.800054700	172.16.21.1	172.16.20.1	ICMP	98			Echo (ping) request id=0x7418, seq=2/512, ttl=64 (reply in 11)
11	8.800289929	172.16.20.1	172.16.21.1	ICMP	98			Echo (ping) reply id=0x7418, seq=2/512, ttl=63 (request in 10)
12	9.815381512	172.16.21.1	172.16.20.1	ICMP	98			Echo (ping) request id=0x7418, seq=3/768, ttl=64 (reply in 13)
13	9.815644328	172.16.20.1	172.16.21.1	ICMP	98			Echo (ping) reply id=0x7418, seq=3/768, ttl=63 (request in 12)
15	10.839382298	172.16.21.1	172.16.20.1	ICMP	98			Echo (ping) request id=0x7418, seq=4/1024, ttl=64 (reply in 16)
16	10.839621089	172.16.20.1	172.16.21.1	ICMP	98			Echo (ping) reply id=0x7418, seq=4/1024, ttl=63 (request in 15)

4.5 - *Ping* do *tux3* para o *Router* do laboratório:

No.	Time	Source	Destination	Protocol	Length	Src Port	Dest Port	Info
13	9.892167124	172.16.20.1	172.16.1.254	ICMP	98			Echo (ping) request id=0x28fa, seq=1/256, ttl=64 (no response found!)
17	10.912085350	172.16.20.1	172.16.1.254	ICMP	98			Echo (ping) request id=0x28fa, seq=2/512, ttl=64 (no response found!)
19	11.936081086	172.16.20.1	172.16.1.254	ICMP	98			Echo (ping) request id=0x28fa, seq=3/768, ttl=64 (no response found!)
21	12.960074936	172.16.20.1	172.16.1.254	ICMP	98			Echo (ping) request id=0x28fa, seq=4/1024, ttl=64 (no response found!)
23	13.984078563	172.16.20.1	172.16.1.254	ICMP	98			Echo (ping) request id=0x28fa, seq=5/1280, ttl=64 (no response found!)
25	15.008077163	172.16.20.1	172.16.1.254	ICMP	98			Echo (ping) request id=0x28fa, seq=6/1536, ttl=64 (no response found!)
26	16.032073038	172.16.20.1	172.16.1.254	ICMP	98			Echo (ping) request id=0x28fa, seq=7/1792, ttl=64 (no response found!)
28	17.056075758	172.16.20.1	172.16.1.254	ICMP	98			Echo (ping) request id=0x28fa, seq=8/2048, ttl=64 (no response found!)
30	18.080071424	172.16.20.1	172.16.1.254	ICMP	98			Echo (ping) request id=0x28fa, seq=9/2304, ttl=64 (no response found!)
31	19.104073305	172.16.20.1	172.16.1.254	ICMP	98			Echo (ping) request id=0x28fa, seq=10/2560, ttl=64 (no response found!)
33	20.128072184	172.16.20.1	172.16.1.254	ICMP	98			Echo (ping) request id=0x28fa, seq=11/2816, ttl=64 (no response found!)
34	21.152072599	172.16.20.1	172.16.1.254	ICMP	98			Echo (ping) request id=0x28fa, seq=12/3072, ttl=64 (no response found!)

4.7 - *Ping* do *tux3* para o *Router* (na *vlan* 21):

No.	Time	Source	Destination	Protocol	Length	Src Port	Dest Port	Info
24	37.691425219	172.16.20.1	172.16.2.254	ICMP	98			Echo (ping) request id=0x2b3a, seq=1/256, ttl=64 (reply in 25)
25	37.692527203	172.16.2.254	172.16.20.1	ICMP	98			Echo (ping) reply id=0x2b3a, seq=1/256, ttl=62 (request in 24)
75	38.692631783	172.16.20.1	172.16.2.254	ICMP	98			Echo (ping) request id=0x2b3a, seq=2/512, ttl=64 (reply in 76)
76	38.693276714	172.16.2.254	172.16.20.1	ICMP	98			Echo (ping) reply id=0x2b3a, seq=2/512, ttl=62 (request in 75)
77	39.716531205	172.16.20.1	172.16.2.254	ICMP	98			Echo (ping) request id=0x2b3a, seq=3/768, ttl=64 (reply in 78)
78	39.717168244	172.16.2.254	172.16.20.1	ICMP	98			Echo (ping) reply id=0x2b3a, seq=3/768, ttl=62 (request in 77)
80	40.740527918	172.16.20.1	172.16.2.254	ICMP	98			Echo (ping) request id=0x2b3a, seq=4/1024, ttl=64 (reply in 81)
81	40.741185421	172.16.2.254	172.16.20.1	ICMP	98			Echo (ping) reply id=0x2b3a, seq=4/1024, ttl=62 (request in 80)
82	41.764534200	172.16.20.1	172.16.2.254	ICMP	98			Echo (ping) request id=0x2b3a, seq=5/1280, ttl=64 (reply in 83)
83	41.765148121	172.16.2.254	172.16.20.1	ICMP	98			Echo (ping) reply id=0x2b3a, seq=5/1280, ttl=62 (request in 82)

Experiência 5

Ping do *tux3* para “ftp.up.pt”:

No.	Time	Source	Destination	Protocol	Length	Src Port	Dest Port	Info
44	19.539511733	172.16.20.1	172.16.2.1	DNS	69	545…	53	Standard query 0xb36e A ftp.up.pt
45	19.539522000	172.16.20.1	172.16.2.1	DNS	69	545…	53	Standard query 0x4177 AAAA ftp.up.pt
46	19.544634782	172.16.2.1	172.16.20.1	DNS	554	53	545…	Standard query response 0xb36e A ftp.up.pt CNAME mirrors.up.pt A 193.137.29.15
47	19.545032398	172.16.2.1	172.16.20.1	DNS	550	53	545…	Standard query response 0x4177 AAAA ftp.up.pt CNAME mirrors.up.pt AAAA 2001:690
48	19.545321268	172.16.20.1	193.137.29.15	ICMP	98			Echo (ping) request id=0x2c97, seq=1/256, ttl=64 (reply in 49)
49	19.549778156	193.137.29.15	172.16.20.1	ICMP	98			Echo (ping) reply id=0x2c97, seq=1/256, ttl=57 (request in 48)
50	19.549869789	172.16.20.1	172.16.2.1	DNS	86	475…	53	Standard query 0xe2c0 PTR 15.29.137.193.in-addr.arpa
51	19.553187957	172.16.2.1	172.16.20.1	DNS	544	53	475…	Standard query response 0xe2c0 PTR 15.29.137.193.in-addr.arpa PTR mirrors.up.pt
53	20.547268030	172.16.20.1	193.137.29.15	ICMP	98			Echo (ping) request id=0x2c97, seq=2/512, ttl=64 (reply in 54)
54	20.549723987	193.137.29.15	172.16.20.1	ICMP	98			Echo (ping) reply id=0x2c97, seq=2/512, ttl=57 (request in 53)
55	21.548790797	172.16.20.1	193.137.29.15	ICMP	98			Echo (ping) request id=0x2c97, seq=3/768, ttl=64 (reply in 56)
56	21.550522273	193.137.29.15	172.16.20.1	ICMP	98			Echo (ping) reply id=0x2c97, seq=3/768, ttl=57 (request in 55)
58	22.550602034	172.16.20.1	193.137.29.15	ICMP	98			Echo (ping) request id=0x2c97, seq=4/1024, ttl=64 (reply in 59)
59	22.552316957	193.137.29.15	172.16.20.1	ICMP	98			Echo (ping) reply id=0x2c97, seq=4/1024, ttl=57 (request in 58)

Ping do *tux3* para “*google.com*”:

```
64 29.267701179 172.16.20.1   172.16.2.1    DNS   70 501… 53   Standard query 0x8560 A google.com
65 29.267711446 172.16.20.1   172.16.2.1    DNS   70 501… 53   Standard query 0x5d6a AAAA google.com
66 29.270540015 172.16.2.1    172.16.20.1   DNS  334 53   501… Standard query response 0x8560 A google.com A 172.217.17.14 NS ns4.google.com N
67 29.270893839 172.16.2.1    172.16.20.1   DNS  346 53   501… Standard query response 0x5d6a AAAA google.com AAAA 2a00:1450:4003:802::200e NS
68 29.271163014 172.16.20.1   172.217.17.14 ICMP  98          Echo (ping) request  id=0x2c9e, seq=1/256, ttl=64 (reply in 69)
69 29.286195381 172.217.17.14 172.16.20.1   ICMP  98          Echo (ping) reply    id=0x2c9e, seq=1/256, ttl=112 (request in 68)
70 29.286286596 172.16.20.1   172.16.2.1    DNS   86 558… 53   Standard query 0x67b5 PTR 14.17.217.172.in-addr.arpa
71 29.288971429 172.16.2.1    172.16.20.1   DNS  553 53   558… Standard query response 0x67b5 PTR 14.17.217.172.in-addr.arpa PTR mad07s09-in-f
73 30.273045486 172.16.20.1   172.217.17.14 ICMP  98          Echo (ping) request  id=0x2c9e, seq=2/512, ttl=64 (reply in 74)
74 30.287221718 172.217.17.14 172.16.20.1   ICMP  98          Echo (ping) reply    id=0x2c9e, seq=2/512, ttl=112 (request in 73)
75 31.274275640 172.16.20.1   172.217.17.14 ICMP  98          Echo (ping) request  id=0x2c9e, seq=3/768, ttl=64 (reply in 76)
76 31.289447135 172.217.17.14 172.16.20.1   ICMP  98          Echo (ping) reply    id=0x2c9e, seq=3/768, ttl=112 (request in 75)
78 32.275513352 172.16.20.1   172.217.17.14 ICMP  98          Echo (ping) request  id=0x2c9e, seq=4/1024, ttl=64 (reply in 79)
79 32.289578883 172.217.17.14 172.16.20.1   ICMP  98          Echo (ping) reply    id=0x2c9e, seq=4/1024, ttl=112 (request in 78)
```

Acesso do *tux3* ao *YouTube*:

Experiência 6

Captura do *download* do ficheiro "*pic1.jpg*":

No.	Time	Source	Destination	Protocol	Length	Src Port	Dest Port	Info
376	2.505475011	172.16.20.1	192.168.109.136	TCP	66	469…	446…	46984 → 44606 [ACK] Seq=1 Ack=282361 Win=524672 L
377	2.505592695	192.168.109.136	172.16.20.1	FTP…	15…	446…	469…	FTP Data: 1448 bytes (PASV) (retr files/pic1.jpg)
378	2.505598702	172.16.20.1	192.168.109.136	TCP	66	469…	446…	46984 → 44606 [ACK] Seq=1 Ack=283809 Win=527488 L
379	2.505715897	192.168.109.136	172.16.20.1	FTP…	15…	446…	469…	FTP Data: 1448 bytes (PASV) (retr files/pic1.jpg)
380	2.505722951	172.16.20.1	192.168.109.136	TCP	66	469…	446…	46984 → 44606 [ACK] Seq=1 Ack=285257 Win=530432 L
381	2.506216736	192.168.109.136	172.16.20.1	FTP…	15…	446…	469…	FTP Data: 1448 bytes (PASV) (retr files/pic1.jpg)
382	2.506222673	172.16.20.1	192.168.109.136	TCP	66	469…	446…	46984 → 44606 [ACK] Seq=1 Ack=286705 Win=533248 L
383	2.506340497	192.168.109.136	172.16.20.1	FTP…	15…	446…	469…	FTP Data: 1448 bytes (PASV) (retr files/pic1.jpg)
384	2.506346643	172.16.20.1	192.168.109.136	TCP	66	469…	446…	46984 → 44606 [ACK] Seq=1 Ack=288153 Win=536192 L
385	2.506464327	192.168.109.136	172.16.20.1	FTP…	15…	446…	469…	FTP Data: 1448 bytes (PASV) (retr files/pic1.jpg)
386	2.506470613	172.16.20.1	192.168.109.136	TCP	66	469…	446…	46984 → 44606 [ACK] Seq=1 Ack=289601 Win=539136 L
387	2.506590043	192.168.109.136	172.16.20.1	FTP…	15…	446…	469…	FTP Data: 1448 bytes (PASV) (retr files/pic1.jpg)
388	2.506596189	172.16.20.1	192.168.109.136	TCP	66	469…	446…	46984 → 44606 [ACK] Seq=1 Ack=291049 Win=541952 L

Captura do *download* do ficheiro "*crab.mp4*":

No.	Time	Source	Destination	Protocol	Length	Src Port	Dest Port	Info
115	22.326070751	172.16.20.1	192.168.109.136	TCP	66	484…	408…	48466 → 40898 [ACK] Seq=1 Ack=68057 Win=93056 Len=
116	22.326356058	192.168.109.136	172.16.20.1	FTP…	15…	408…	484…	FTP Data: 1448 bytes (PASV) (retr files/crab.mp4)
117	22.326362134	172.16.20.1	192.168.109.136	TCP	66	484…	408…	48466 → 40898 [ACK] Seq=1 Ack=69505 Win=96000 Len=
118	22.326480028	192.168.109.136	172.16.20.1	FTP…	15…	408…	484…	FTP Data: 1448 bytes (PASV) (retr files/crab.mp4)
119	22.326486244	172.16.20.1	192.168.109.136	TCP	66	484…	408…	48466 → 40898 [ACK] Seq=1 Ack=70953 Win=98944 Len=
120	22.326602811	192.168.109.136	172.16.20.1	FTP…	15…	408…	484…	FTP Data: 1448 bytes (PASV) (retr files/crab.mp4)
121	22.326609167	172.16.20.1	192.168.109.136	TCP	66	484…	408…	48466 → 40898 [ACK] Seq=1 Ack=72401 Win=101760 Len
122	22.326730484	192.168.109.136	172.16.20.1	FTP…	15…	408…	484…	FTP Data: 1448 bytes (PASV) (retr files/crab.mp4)
123	22.326736700	172.16.20.1	192.168.109.136	TCP	66	484…	408…	48466 → 40898 [ACK] Seq=1 Ack=73849 Win=104704 Len
124	22.326853476	192.168.109.136	172.16.20.1	FTP…	15…	408…	484…	FTP Data: 1448 bytes (PASV) (retr files/crab.mp4)
125	22.326859902	172.16.20.1	192.168.109.136	TCP	66	484…	408…	48466 → 40898 [ACK] Seq=1 Ack=75297 Win=107648 Len
126	22.327002032	192.168.109.136	172.16.20.1	FTP…	15…	408…	484…	FTP Data: 1448 bytes (PASV) (retr files/crab.mp4)
127	22.327008108	172.16.20.1	192.168.109.136	TCP	66	484…	408…	48466 → 40898 [ACK] Seq=1 Ack=76745 Win=110464 Len
128	22.327125583	192.168.109.136	172.16.20.1	FTP…	15…	408…	484…	FTP Data: 1448 bytes (PASV) (retr files/crab.mp4)
129	22.327131869	172.16.20.1	192.168.109.136	TCP	66	484…	408…	48466 → 40898 [ACK] Seq=1 Ack=78193 Win=113408 Len
130	22.327249763	192.168.109.136	172.16.20.1	FTP…	15…	408…	484…	FTP Data: 1448 bytes (PASV) (retr files/crab.mp4)
131	22.327256119	172.16.20.1	192.168.109.136	TCP	66	484…	408…	48466 → 40898 [ACK] Seq=1 Ack=79641 Win=116224 Len

Captura do *download* de dois ficheiros de tamanho elevado (*"Crypto101.pdf" e "crab.mp4"*):

Time	Source	Destination	Protocol	Length	Src Port	Dest Port	Info
119.4895570…	193.137.29.15	172.16.20.1	FTP…	14…	549…	435…	FTP Data: 1368 bytes (PASV) (retr pub/parrot/misc/openbooks/crypto/Crypto101.pdf)
119.4894467…	172.16.20.1	193.137.29.15	TCP	66	435…	549…	43552 → 54945 [ACK] Seq=1 Ack=331057 Win=518656 Len=0 TSval=821246477 TSecr=75854
119.4894404…	193.137.29.15	172.16.20.1	FTP…	14…	549…	435…	FTP Data: 1368 bytes (PASV) (retr pub/parrot/misc/openbooks/crypto/Crypto101.pdf)
119.4893308…	172.16.20.1	193.137.29.15	TCP	66	435…	549…	43552 → 54945 [ACK] Seq=1 Ack=329689 Win=515712 Len=0 TSval=821246477 TSecr=75854
119.4893241…	193.137.29.15	172.16.20.1	FTP…	14…	549…	435…	FTP Data: 1368 bytes (PASV) (retr pub/parrot/misc/openbooks/crypto/Crypto101.pdf)
119.4892141…	172.16.20.1	192.168.109.136	TCP	66	505…	486…	50588 → 48625 [ACK] Seq=1 Ack=16384121 Win=1344768 Len=0 TSval=4294374928 TSecr=2
119.4892074…	192.168.109.136	172.16.20.1	FTP…	15…	486…	505…	FTP Data: 1448 bytes (PASV) (retr files/crab.mp4)
119.4890912…	172.16.20.1	193.137.29.15	TCP	66	435…	549…	43552 → 54945 [ACK] Seq=1 Ack=328321 Win=512896 Len=0 TSval=821246476 TSecr=75854
119.4890844…	193.137.29.15	172.16.20.1	FTP…	14…	549…	435…	FTP Data: 1368 bytes (PASV) (retr pub/parrot/misc/openbooks/crypto/Crypto101.pdf)
119.4889688…	192.168.109.136	172.16.20.1	FTP…	15…	486…	505…	FTP Data: 1448 bytes (PASV) (retr files/crab.mp4)
119.4888514…	172.16.20.1	192.168.109.136	TCP	66	505…	486…	50588 → 48625 [ACK] Seq=1 Ack=16381225 Win=1344768 Len=0 TSval=4294374928 TSecr=2
119.4888452…	192.168.109.136	172.16.20.1	FTP…	15…	486…	505…	FTP Data: 1448 bytes (PASV) (retr files/crab.mp4)
119.4887221…	192.168.109.136	172.16.20.1	FTP…	15…	486…	505…	FTP Data: 1448 bytes (PASV) (retr files/crab.mp4)
119.4886069…	172.16.20.1	192.168.109.136	TCP	66	505…	486…	50588 → 48625 [ACK] Seq=1 Ack=16378329 Win=1344768 Len=0 TSval=4294374928 TSecr=2
119.4886003…	192.168.109.136	172.16.20.1	FTP…	15…	486…	505…	FTP Data: 1448 bytes (PASV) (retr files/crab.mp4)
119.4884762…	192.168.109.136	172.16.20.1	FTP…	15…	486…	505…	FTP Data: 1448 bytes (PASV) (retr files/crab.mp4)

Captura do *download* do ficheiro "*pic1.jpg*" - Início:

No.	Time	Source	Destination	Protocol	Length	Src Port	Dest Port	Info
4	2.446302546	172.16.20.1	172.16.2.1	DNS	76	523…	53	Standard query 0xc4a5 A netlab1
5	2.449723141	172.16.2.1	172.16.20.1	DNS	322	53	523…	Standard query response 0xc4a5
6	2.449803669	172.16.20.1	192.168.109.136	TCP	74	454…	21	45482 → 21 [SYN] Seq=0 Win=6424
7	2.452135353	192.168.109.136	172.16.20.1	TCP	74	21	454…	21 → 45482 [SYN, ACK] Seq=0 Ack
8	2.452154280	172.16.20.1	192.168.109.136	TCP	66	454…	21	45482 → 21 [ACK] Seq=1 Ack=1 Wi
9	2.455564399	192.168.109.136	172.16.20.1	FTP	100	21	454…	Response: 220 Welcome to netlab
10	2.455577460	172.16.20.1	192.168.109.136	TCP	66	454…	21	45482 → 21 [ACK] Seq=1 Ack=35 W
11	2.455681105	172.16.20.1	192.168.109.136	FTP	77	454…	21	Request: user rcom
12	2.457997983	192.168.109.136	172.16.20.1	TCP	66	21	454…	21 → 45482 [ACK] Seq=35 Ack=12
13	2.458307035	192.168.109.136	172.16.20.1	FTP	100	21	454…	Response: 331 Please specify th
14	2.458400344	172.16.20.1	192.168.109.136	FTP	77	454…	21	Request: pass rcom
15	2.460158414	192.168.109.136	172.16.20.1	TCP	66	21	454…	21 → 45482 [ACK] Seq=69 Ack=23
16	2.469737770	192.168.109.136	172.16.20.1	FTP	89	21	454…	Response: 230 Login successful.
17	2.469821371	172.16.20.1	192.168.109.136	FTP	73	454…	21	Request: pasv
18	2.471727367	192.168.109.136	172.16.20.1	TCP	66	21	454…	21 → 45482 [ACK] Seq=92 Ack=30
19	2.472111639	192.168.109.136	172.16.20.1	FTP	119	21	454…	Response: 227 Entering Passive
20	2.472226948	172.16.20.1	192.168.109.136	TCP	74	469…	446…	46984 → 44606 [SYN] Seq=0 Win=6
21	2.473217102	192.168.109.136	172.16.20.1	TCP	74	446…	469…	44606 → 46984 [SYN, ACK] Seq=0
22	2.473228417	172.16.20.1	192.168.109.136	TCP	66	469…	446…	46984 → 44606 [ACK] Seq=1 Ack=1
23	2.473278983	172.16.20.1	192.168.109.136	FTP	87	454…	21	Request: retr files/pic1.jpg

Captura do *download* do ficheiro "*pic1.jpg*" - Fim:

451	2.511007497	192.168.109.136	172.16.20.1	FTP…	15…	446…	469…	FTP Data: 1448 bytes (PASV) (retr files/pic1.jpg)
452	2.511012805	172.16.20.1	192.168.109.136	TCP	66	469…	446…	46984 → 44606 [ACK] Seq=1 Ack=337385 Win=634624 Len
453	2.511131816	192.168.109.136	172.16.20.1	FTP…	15…	446…	469…	FTP Data: 1448 bytes (PASV) (retr files/pic1.jpg)
454	2.511137124	172.16.20.1	192.168.109.136	TCP	66	469…	446…	46984 → 44606 [ACK] Seq=1 Ack=338833 Win=637568 Len
455	2.511254110	192.168.109.136	172.16.20.1	FTP…	15…	446…	469…	FTP Data: 1448 bytes (PASV) (retr files/pic1.jpg)
456	2.511259488	172.16.20.1	192.168.109.136	TCP	66	469…	446…	46984 → 44606 [ACK] Seq=1 Ack=340281 Win=640512 Len
457	2.511283024	192.168.109.136	172.16.20.1	FTP…	389	446…	469…	FTP Data: 323 bytes (PASV) (retr files/pic1.jpg)
458	2.511345533	172.16.20.1	192.168.109.136	TCP	66	454…	21	45482 → 21 [FIN, ACK] Seq=51 Ack=221 Win=64256 Len=
459	2.511369280	172.16.20.1	192.168.109.136	TCP	66	469…	446…	46984 → 44606 [FIN, ACK] Seq=1 Ack=340605 Win=64332
460	2.512340506	192.168.109.136	172.16.20.1	TCP	66	446…	469…	44606 → 46984 [ACK] Seq=340605 Ack=2 Win=65280 Len=
461	2.513010294	192.168.109.136	172.16.20.1	FTP	90	21	454…	Response: 226 Transfer complete.
462	2.513033272	172.16.20.1	192.168.109.136	TCP	54	454…	21	45482 → 21 [RST] Seq=51 Win=0 Len=0
463	2.513896382	192.168.109.136	172.16.20.1	TCP	66	21	454…	21 → 45482 [FIN, ACK] Seq=245 Ack=52 Win=65280 Len=
464	2.513905671	172.16.20.1	192.168.109.136	TCP	54	454…	21	45482 → 21 [RST] Seq=52 Win=0 Len=0