



Faculdade de Engenharia da
Universidade do Porto

Redes de Computadores

2º Trabalho Laboratorial

Turma 12 - Grupo 1

Afonso Neves (up202108884@up.pt)

João Miranda (up202003518@up.pt)

Porto, 22 de dezembro de 2023

Sumário

Este trabalho realizado no âmbito da Unidade Curricular de Redes de Computadores visa a implementação de um programa de download de ficheiros via FTP e a configuração e utilização de uma rede de computadores.

Introdução

O objetivo deste trabalho foi o desenvolvimento de um programa de transferência de ficheiros via FTP e a configuração de uma rede de computadores, de acordo com as indicações presentes no guião fornecido. O presente relatório é composto por três secções:

- **Aplicação FTP**
 - Arquitetura da aplicação
 - Testes realizados e resultados
- **Configuração e análise da rede**
 - Configurar uma rede IP
 - Implementar duas *bridges* num *switch*
 - Configurar um router em Linux
 - Configurar um router comercial com NAT
 - DNS
 - Ligações TCP
- **Conclusões:** Síntese da informação apresentada nas secções anteriores e reflexão sobre os objetivos de aprendizagem alcançados

Aplicação FTP

Arquitetura da aplicação

O programa desenvolvido realiza a transferência de um ficheiro através do protocolo FTP. Para tal, foi consultada a norma RFC959, funcionamento e arquitetura do protocolo FTP, disponibilizada na página da Unidade Curricular.

Primeiramente, o programa usa a função *parseToURL* para realizar um processamento do argumento passado pelo utilizador, utilizando expressões regulares, de modo a criar uma estrutura de dados que contém as informações necessárias para o estabelecimento da ligação: o endereço IP do servidor a conectar; o nome do servidor; o username e password a usar na autenticação; o *path* para chegar ao ficheiro; o nome e extensão do ficheiro a transferir. De seguida, é criado o primeiro socket ligado ao IP do host

através da função *createSocket*. Após o handshake, é feita a autenticação no servidor através da função *authConn* e logo de seguida é enviado o código “PASV” pela função *passiveMode*, para que o servidor entre em modo passivo. Assim que o servidor envie o IP e a porta para uma segunda conexão FTP, o programa utiliza a função *requestResource* para solicitar o envio do ficheiro pretendido. Após a receção do código 150 (Transferência Pronta), a função *getResource* recebe o ficheiro. Finalmente, após a transferência ter terminado, as ligações e os dois sockets são encerrados com comandos “QUIT” através da execução da função *closeConn*. Durante o processo é utilizada a função *readResponse* para interpretar as mensagens enviadas pelo servidor no primeiro socket recorrendo a uma máquina de estados semelhante às implementadas no primeiro trabalho.

Testes realizados e resultados

Ao longo do desenvolvimento, o programa foi testado de várias formas para garantir a sua validade e coerência. Os testes realizados foram os seguintes:

- ✓ Transferência de ficheiros com hosts distintos.
- ✓ Transferência de ficheiros com tamanhos distintos.
- ✓ Transferência de ficheiros com e sem autenticação no servidor.
- ✓ Transferência de ficheiros com pacotes de dados de tamanhos distintos.
- ✓ Transferência de vários ficheiros em simultâneo.

Na seguinte imagem observa-se o resultado de uma transferência bem-sucedida:

1	0.000000000	172.19.238.154	172.19.224.1	DNS	76 Standard query 0x80f4 A netlab1.fe.up.pt
2	0.024931755	172.19.224.1	172.19.238.154	DNS	302 Standard query response 0x80f4 A netlab1.fe.up.pt A 192.168.109.136 A 193.136.28.10 A 193.136.28.9 A 10.227.244.110 A 10.
3	0.025136114	172.19.238.154	192.168.109.136	TCP	74 55048 → 21 [SYN] Seq=0 Win=65500 Len=0 MSS=1310 SACK_PERM TSval=2887865519 TSecr=0 WS=128
4	0.050881880	192.168.109.136	172.19.238.154	TCP	74 21 → 55048 [SYN, ACK] Seq=0 Ack=1 Win=65160 Len=0 MSS=1460 SACK_PERM TSval=608523689 TSecr=2887865519 WS=128
5	0.050984706	172.19.238.154	192.168.109.136	TCP	66 55048 → 21 [ACK] Seq=1 Ack=1 Win=65536 Len=0 TSval=2887865545 TSecr=608523689
6	0.075635159	192.168.109.136	172.19.238.154	FTP	100 Response: 220 Welcome to netlab-FTP server
7	0.075743926	172.19.238.154	192.168.109.136	TCP	66 55048 → 21 [ACK] Seq=1 Ack=35 Win=65536 Len=0 TSval=2887865570 TSecr=608523715
8	0.075824976	172.19.238.154	192.168.109.136	FTP	76 Request: USER rcom
9	0.102588676	192.168.109.136	172.19.238.154	TCP	66 21 → 55048 [ACK] Seq=35 Ack=11 Win=65280 Len=0 TSval=608523740 TSecr=2887865570
10	0.102589127	192.168.109.136	172.19.238.154	FTP	100 Response: 331 Please specify the password.
11	0.102810261	172.19.238.154	192.168.109.136	FTP	76 Request: PASS rcom
12	0.125283162	192.168.109.136	172.19.238.154	TCP	66 21 → 55048 [ACK] Seq=69 Ack=21 Win=65280 Len=0 TSval=608523766 TSecr=2887865597
13	0.151208549	192.168.109.136	172.19.238.154	FTP	89 Response: 230 Login successful.
14	0.151473154	172.19.238.154	192.168.109.136	FTP	71 Request: PASV
15	0.175546574	192.168.109.136	172.19.238.154	TCP	66 21 → 55048 [ACK] Seq=92 Ack=26 Win=65280 Len=0 TSval=608523815 TSecr=2887865645
16	0.175546875	192.168.109.136	172.19.238.154	FTP	120 Response: 227 Entering Passive Mode (192,168,109,136,191,126).
17	0.175734458	172.19.238.154	192.168.109.136	TCP	74 46316 → 49022 [SYN] Seq=0 Win=65500 Len=0 MSS=1310 SACK_PERM TSval=2887865670 TSecr=0 WS=128
18	0.201607931	192.168.109.136	172.19.238.154	TCP	74 49022 → 46316 [SYN, ACK] Seq=0 Ack=1 Win=65160 Len=0 MSS=1460 SACK_PERM TSval=608523839 TSecr=2887865670 WS=128
19	0.201706709	172.19.238.154	192.168.109.136	TCP	66 46316 → 49022 [ACK] Seq=1 Ack=1 Win=65536 Len=0 TSval=2887865696 TSecr=608523839
20	0.201775781	172.19.238.154	192.168.109.136	FTP	80 Request: RETR pipe.txt
21	0.227586961	192.168.109.136	172.19.238.154	FTP	134 Response: 150 Opening BINARY mode data connection for pipe.txt (1863 bytes).
22	0.227587341	192.168.109.136	172.19.238.154	FTP-DL	1929 FTP Data: 1863 bytes (PASV) (RETR pipe.txt)
23	0.227587392	192.168.109.136	172.19.238.154	TCP	66 49022 → 46316 [FIN, ACK] Seq=1864 Ack=1 Win=65280 Len=0 TSval=608523867 TSecr=2887865696
24	0.227686020	172.19.238.154	192.168.109.136	TCP	66 46316 → 49022 [ACK] Seq=1 Ack=1864 Win=63872 Len=0 TSval=2887865722 TSecr=608523866
25	0.277597810	172.19.238.154	192.168.109.136	TCP	66 46316 → 49022 [ACK] Seq=1 Ack=1865 Win=64256 Len=0 TSval=2887865772 TSecr=608523867
26	0.277844870	172.19.238.154	192.168.109.136	TCP	66 55048 → 21 [ACK] Seq=40 Ack=214 Win=65536 Len=0 TSval=2887865772 TSecr=608523866
27	0.302242637	192.168.109.136	172.19.238.154	FTP	90 Response: 226 Transfer complete.
28	0.302274016	172.19.238.154	192.168.109.136	TCP	66 55048 → 21 [ACK] Seq=40 Ack=238 Win=65536 Len=0 TSval=2887865796 TSecr=608523941
29	0.303895058	172.19.238.154	192.168.109.136	FTP	71 Request: QUIT
30	0.327636952	192.168.109.136	172.19.238.154	FTP	80 Response: 221 Goodbye.
31	0.327637352	192.168.109.136	172.19.238.154	TCP	66 21 → 55048 [FIN, ACK] Seq=252 Ack=45 Win=65280 Len=0 TSval=608523967 TSecr=2887865798
32	0.327727925	172.19.238.154	192.168.109.136	TCP	66 55048 → 21 [FIN, ACK] Seq=45 Ack=253 Win=65536 Len=0 TSval=2887865822 TSecr=608523967
33	0.327743615	172.19.238.154	192.168.109.136	TCP	66 46316 → 49022 [FIN, ACK] Seq=1 Ack=1865 Win=64256 Len=0 TSval=2887865822 TSecr=608523867
34	0.351891243	192.168.109.136	172.19.238.154	TCP	66 21 → 55048 [ACK] Seq=253 Ack=46 Win=65280 Len=0 TSval=608523991 TSecr=2887865822
35	0.351891574	192.168.109.136	172.19.238.154	TCP	66 49022 → 46316 [ACK] Seq=1865 Ack=2 Win=65280 Len=0 TSval=608523991 TSecr=2887865822

Figura 1 - Exemplo de transferência bem-sucedida

Configuração e análise da rede

Experiência 1 - Configurar uma rede IP

O objetivo desta experiência era fazer duas máquinas comunicarem através de uma rede configurando os seus endereços IP. Para isso, ligamos as portas E0 do tux3 e do tux4 ao switch e utilizamos o comando `ifconfig` para configurar o IP de cada um. De seguida, utilizamos o comando `ping` para verificar a conexão entre as duas máquinas e observamos os endereços IP e MAC nas tabelas ARP criadas, bem como os pacotes ARP e ICMP trocados.

O protocolo ARP (Address Resolution Protocol) relaciona um IP de uma máquina com o endereço MAC respetivo, fazendo a ligação entre a network layer e a link layer. São enviados, em broadcast, 2 endereços IP num pacote: o da máquina de destino e o da máquina de origem. Por sua vez, o destinatário envia um pacote do mesmo protocolo que contém o seu endereço MAC, guardada esta associação numa das entradas da tabela ARP de ambos os computadores. Quando são eliminadas as entradas das tabelas ARP, existe uma troca de pacotes ARP no início da ligação de modo a repor as associações entre os endereços IP e MAC que foram eliminados.

O protocolo ICMP (Internet Control Message Protocol) é utilizado para trocar mensagens de controlo, que indicam sucesso ou erros durante a comunicação com outro endereço IP.

O log da captura que corrobora o que foi acima descrito pode ser consultado nos anexos, bem como os comandos utilizados na configuração.

Experiência 2 - Implementar duas bridges num switch

O objetivo desta experiência era a implementação de duas redes locais, uma composta pelo tux3 e pelo tux4 e outra composta pelo tux2, utilizando duas bridges no switch. Assim, repetimos a configuração da primeira experiência e ligamos também a porta E0 do tux2 ao switch, configurando também o seu endereço IP. De modo a criar as bridges, necessitamos de configurar o switch, onde criamos as bridges 60 e 61 com o comando `/interface bridge add` e removemos as portas às quais cada tux estava ligado inicialmente com o comando `/interface bridge port remove`, para que fosse possível adicionar as interfaces do tux3 e tux4 à bridge60 e a do tux2 à bridge61 com o comando `/interface bridge port add`.

Nesta experiência, verificamos que ao fazer broadcast a partir do tux3 conseguíamos alcançar o tux4, mas não o tux2. Isto deve-se ao facto de apenas o tux4 se encontrar na mesma LAN do tux3. Fazendo o mesmo a partir do tux2, como previsto, não era possível alcançar nenhuma outra máquina uma vez que se encontrava numa LAN isolada.

Os logs da captura que corroboram o que foi acima descrito podem ser consultados nos anexos, bem como os comandos utilizados na configuração das bridges.

Experiência 3 - Configurar um router em Linux

Nesta experiência, o objetivo era tornar o tux4 num router de forma a ser possível a comunicação entre o tux3 e o tux2. Para isso, conectamos a porta E1 do tux4 ao switch e adicionamos a sua interface à bridge61 seguindo, para isso, os mesmos passos da experiência anterior. De seguida, desativamos o `icmp_echo_ignore_broadcasts` e ativamos o IP forwarding. Por fim, com o comando `route add`, criamos rotas no tux2 e no tux3, tendo como gateway o IP do tux4 que era acessível através de cada um deles.

Com estas rotas, foi possível utilizar o tux4 como router que faz a ligação entre as duas redes locais, sendo que, quando fizemos ping entre o tux2 e o tux3, os pacotes ARP e ICMP chegaram ao destino. Por sua vez, os pacotes ARP e ICMP obtidos no tux4 continham o endereço IP da máquina de destino, mas o endereço MAC do próprio tux4, uma vez que, atuando este como um router, trata do redirecionamento da informação entre as duas redes criadas.

Os logs da captura que corroboram o que foi acima descrito podem ser consultados nos anexos, bem como os comandos utilizados na configuração.

Experiência 4 - Configurar um router comercial com NAT

O objetivo da quarta experiência era adicionar uma conexão de internet à rede desenvolvida através de um router comercial conectado à rede do laboratório. Usando a rede já desenvolvida anteriormente, até à experiência 3, configuramos e adicionamos à bridge61 um router comercial com NAT implementado.

Primeiramente, ligamos uma entrada do router comercial à régua e outra ao switch e adicionamos a interface à bridge61 da mesma forma que já tínhamos feito nas experiências anteriores. De seguida, removemos o cabo que ligava o tux2 ao switch e ligamo-lo ao router comercial para proceder à sua configuração, onde configuramos os IPs correspondentes a cada interface com o comando `/ip address add`. Finalmente, criamos rotas default em cada tux, ligando-os ao router, e uma rota que possibilitava ligar o router a cada uma das redes locais através do tux4.

Numa primeira situação, ao fazer ping desde o tux2 para o tux3 sem ligação do tux2 ao tux4 e os ICMP redirects desativados, os pacotes foram reencaminhados pela default route até ao endereço IP de destino através do router. Por outro lado, ativando de novo as rotas e os ICMP redirects, os pacotes já não chegam a passar pelo router, uma vez que a ligação mais direta estava disponível, atuando o tux4 como intermediário. É possível, com isto,

concluir que os pacotes ICMP permitem um melhor aproveitamento das ligações que existem na rede.

Também nesta experiência, fizemos uso do Network Address Translation (NAT), um método de mapeamento que traduz endereços de uma rede local para um único endereço público (e vice-versa). Uma vez que com o NAT ativado existe ligação à internet, quando este é desativado, o emissor não consegue receber a resposta aos pedidos solicitados, pois o router não é capaz de traduzir o endereço de destino num da rede interna.

Os logs da captura que corroboram o que foi acima descrito podem ser consultados nos anexos, bem como os comandos utilizados na configuração.

Experiência 5 – DNS

O objetivo da penúltima experiência era configurar o DNS para poder aceder a websites através do seu nome de domínio dentro da rede criada até aqui. Foi apenas necessário mudar o conteúdo do ficheiro `/etc/resolv.conf` em cada máquina para nameserver `172.16.1.1`, o endereço IP do router do laboratório.

Após isso, foi possível verificar que a troca de pacotes DNS é feita antes de qualquer outro protocolo, uma vez que aqui será feita a tradução do nome de domínio para o endereço IP correspondente que será usado por todos os protocolos restantes.

Os logs da captura que corroboram o que foi acima descrito podem ser consultados nos anexos, bem como os comandos utilizados na configuração.

Experiência 6 - Ligações TCP

A última experiência tinha como objetivo verificar o funcionamento do protocolo TCP usando a aplicação de download desenvolvida na primeira parte do projeto para descarregar um ficheiro na rede desenvolvida.

Em primeiro lugar, compilamos a aplicação no tux3 e procedemos ao download de um ficheiro. Inicialmente, observamos a transmissão de pacotes DNS que traduziam o nome do servidor no endereço IP, seguido de pacotes FTP que realizavam o handshake entre o cliente e o servidor.

A aplicação FTP abre duas ligações TCP, sendo que uma envia comandos de controlo ao servidor e a outra recebe o ficheiro selecionado. Ambas utilizam o mecanismo ARQ, utilizado para controlo de fluxo, congestionamento e erros.

Num segundo momento desta experiência, fizemos o download de um ficheiro no tux3 e, logo de seguida, no tux2. Devido à ação de controlo de congestionamento, pudemos verificar que a velocidade da transferência no tux3 diminui para cerca de metade quando o download no tux2 é iniciado.

Os logs da captura que corroboram o que foi acima descrito podem ser consultados nos anexos, bem como os comandos utilizados na configuração.

Conclusões

Com o concluir deste projeto, foi nos possível consolidar o conhecimento acerca dos protocolos envolvidos na transferência de dados através de redes de computadores e todos os restantes conceitos associados.

Referências

1. Guião fornecido no [Moodle](#)
2. Code Examples fornecidos no [Moodle](#)
3. Exemplos de ficheiros para transferência fornecidos no [Moodle](#)
4. RFCs fornecidos no [Moodle](#)
5. [Beej's Guide to Network Programming - Using Internet Sockets](#)

Anexo 1 – Código do Programa FTP

1.1 – *client.h*

```
// FTP Program Header

#ifndef _CLIENT_H_
#define _CLIENT_H_

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

// ----- MACROS -----

/* LENGTHS */
#define MAX_LENGTH 500
#define URL_LENGTH 150

/* SERVER RESPONSES */
#define READY_AUTH 220
#define READY_PASS 331
#define LOGIN_SUCCESS 230
#define PASSIVE_MODE 227
#define READY_TRANSFER 150
#define TRANSFER_COMPLETE 226
#define GOODBYE 221

/* REGULAR EXPRESSIONS */
#define DEFAULT_HOST "%^[^/]*//%^[^/]"
#define HOST_REGEX "%^[^/]*//%^[^@]*@%^[^/]"
#define USER_REGEX "%^[^/]*//%^[^:]*/"
#define PASS_REGEX "%^[^/]*//%^[^:]*:%^[^@\\n$]"
#define RESOURCE_REGEX "%^[^/]*//%^[^/]*/%s"
#define PASSIVE_MODE_REGEX "%^[^()](%d,%d,%d,%d,%d,%d,%d)%^[^\\n$)]"

// ----- DATA STRUCTURES -----

typedef struct
{
    char ip[URL_LENGTH];
    char host[URL_LENGTH];
    char user[URL_LENGTH];
    char pass[URL_LENGTH];
    char resource[URL_LENGTH];
    char file[URL_LENGTH];
} URL;
```



```

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

// ----- AUX FUNCTIONS -----

// Parses the given URL into its various components.
int parseToURL(char *input, URL *url);

// Creates a socket and connects to the given IP and port.
int createSocket(char *IP, int port);

// Reads the response from the server, using a state machine.
int readResponse(int socket, char *buffer);

// Authenticates the user with the given credentials.
int authConn(int socket, char *user, char *pass);

// Enters passive mode by sending the PASV command.
int passiveMode(int socket, char* IP, int *port);

// Requests a resource from the server.
int requestResource(int socket, char *resource);

// Downloads a file from the server.
int getResource(int socketA, int socketB, char *filename);

// Closes all connection and sockets.
int closeConn(int socketA, int socketB);

#endif // _CLIENT_H_

```

1.2 – *client.c*

// FTP Program Implementation

```
#include "client.h"
```

```
int parseToURL(char *input, URL *url)
{
    regex_t regex;
    struct hostent *h;
    regcomp(&regex, "/", 0);
    if (regexec(&regex, input, 0, NULL, 0)) return -1;

    regcomp(&regex, "@", 0);
    if (regexec(&regex, input, 0, NULL, 0) == 0)
    {
        sscanf(input, HOST_REGEX, url->host);
        sscanf(input, USER_REGEX, url->user);
        sscanf(input, PASS_REGEX, url->pass);
    }
    else
    {
        sscanf(input, DEFAULT_HOST, url->host);
        strcpy(url->user, "anonymous");
        strcpy(url->pass, "password");
    }

    sscanf(input, RESOURCE_REGEX, url->resource);
    if (strlen(url->resource) == 0) return -1;

    strcpy(url->file, strrchr(input, '/') + 1);
    if (strlen(url->file) == 0) return -1;

    if ((h = gethostbyname(url->host)) == NULL) return -1;
    strcpy(url->ip, inet_ntoa(*(struct in_addr *)h->h_addr_list[0]));

    return 0;
}

int createSocket(char *IP, int port)
{
    int 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 ((fd = socket(AF_INET, SOCK_STREAM, 0)) < 0) return -1;
    if (connect(fd, (struct sockaddr *)&server_addr, sizeof(server_addr)) < 0) return -1;

    return fd;
}
```

```

int readResponse(int socket, char *buffer)
{
    char byte;
    int bufferPos = 0, answerCode;
    ResponseState state = START;

    while (state != END)
    {
        if (read(socket, &byte, 1) > 0)
        {
            switch (state)
            {
                case START:
                    if (byte == ' ')
                        state = SINGLE;
                    else if (byte == '-')
                        state = MULTIPLE;
                    else if (byte == '\n')
                        state = END;
                    else
                        buffer[bufferPos++] = byte;
                    break;

                case SINGLE:
                    if (byte == '\n')
                        state = END;
                    else
                        buffer[bufferPos++] = byte;
                    break;

                case MULTIPLE:
                    if (byte == '\n')
                    {
                        bufferPos = 0;
                        memset(buffer, 0, MAX_LENGTH);
                        state = START;
                    }
                    else
                        buffer[bufferPos++] = byte;
                    break;

                default:
                    break;
            }
        }
    }

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

    return answerCode;
}

```

```

int authConn(int socket, char *user, char *pass)
{
    char answer[MAX_LENGTH];

    char userCommand[6 + strlen(user)];
    sprintf(userCommand, "USER %s\n", user);
    char passCommand[6 + strlen(pass)];
    sprintf(passCommand, "PASS %s\n", pass);

    write(socket, userCommand, strlen(userCommand));
    if (readResponse(socket, answer) != READY_PASS) return -1;

    write(socket, passCommand, strlen(passCommand));
    if (readResponse(socket, answer) != LOGIN_SUCCESS) return -1;

    return 0;
}

int passiveMode(int socket, char *IP, int *port)
{
    char answer[MAX_LENGTH];
    int ip1, ip2, ip3, ip4, port1, port2;

    write(socket, "PASV\n", 5);
    if (readResponse(socket, answer) != PASSIVE_MODE) return -1;

    sscanf(answer, PASSIVE_MODE_REGEX, &ip1, &ip2, &ip3, &ip4, &port1, &port2);

    sprintf(IP, "%d.%d.%d.%d", ip1, ip2, ip3, ip4);
    *port = port1 * 256 + port2;

    return 0;
}

int requestResource(int socket, char *resource)
{
    char answer[MAX_LENGTH];

    char requestCommand[6 + strlen(resource)];
    sprintf(requestCommand, "RETR %s\n", resource);

    write(socket, requestCommand, sizeof(requestCommand));
    if (readResponse(socket, answer) != READY_TRANSFER) return -1;

    return 0;
}

int getResource(int socketA, int socketB, char *filename)
{
    int bytes;
    char answer[MAX_LENGTH], buffer[MAX_LENGTH];

    FILE *fd = fopen(filename, "wb");
    if (fd == NULL) return -1;

```

```

while ((bytes = read(socketB, buffer, MAX_LENGTH)) > 0)
{
    if (fwrite(buffer, bytes, 1, fd) < 0) return -1;
}

if (readResponse(socketA, answer) != TRANSFER_COMPLETE) return -1;

fclose(fd);
return 0;
}

int closeConn(int socketA, int socketB)
{
    char answer[MAX_LENGTH];

    write(socketA, "QUIT\n", 5);
    if (readResponse(socketA, answer) != GOODBYE) return -1;

    return close(socketA) || close(socketB);
}

int main(int argc, char *argv[])
{
    if (argc != 2)
    {
        printf("Usage: %s <URL>\n", argv[0]);
        exit(-1);
    }

    URL url;
    memset(&url, 0, sizeof(url));
    if (parseToURL(argv[1], &url) != 0)
    {
        printf("Parse Error!\n");
        exit(-1);
    }

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

    char answer[MAX_LENGTH];

    int socketA = createSocket(url.ip, 21);
    if (socketA < 0 || readResponse(socketA, answer) != READY_AUTH)
    {
        printf("Socket A Error!\n");
        exit(-1);
    }

    if (authConn(socketA, url.user, url.pass))
    {
        printf("Authentication Error!\n");
        exit(-1);
    }
}

```

```

int port;
char ip[MAX_LENGTH];
if (passiveMode(socketA, ip, &port))
{
    printf("Passive Mode Error!\n");
    exit(-1);
}

int socketB = createSocket(ip, port);
if (socketB < 0)
{
    printf("Socket B Error!\n");
    exit(-1);
}

if (requestResource(socketA, url.resource))
{
    printf("Unknown Resource!\n");
    exit(-1);
}

if (getResource(socketA, socketB, url.file))
{
    printf("Download Error!\n");
    exit(-1);
}

if (closeConn(socketA, socketB) != 0)
{
    printf("Sockets Close Error!\n");
    exit(-1);
}

printf("Download Complete!\n\n");

return 0;
}

```

Anexo 2 – Comandos de Configuração

2.1 – Experiência 1

TUX3:

```
ifconfig eth0 up
ifconfig eth0 172.16.60.1/24
```

TUX4:

```
ifconfig eth0 up
ifconfig eth0 172.16.60.254/24
```

TUX3:

```
ping 172.16.60.254
```

TUX4:

```
ping 172.16.60.1
route -n
arp -a
```

TUX3:

```
route -n
arp -a
arp -d 172.16.60.254/24
```

2.2 – Experiência 2

TUX2:

```
ifconfig eth0 up
ifconfig eth0 172.16.61.1/24
```

Switch Console:

```
/system reset-configuration
/interface bridge add name=bridge60
/interface bridge add name=bridge61
/interface bridge port remove [find interface=ether6]
/interface bridge port remove [find interface=ether8]
/interface bridge port remove [find interface=ether12]
/interface bridge port add bridge=bridge60 interface=ether6
/interface bridge port add bridge=bridge60 interface=ether8
/interface bridge port add bridge=bridge61 interface=ether12
/interface bridge port print
```

TUX3:

```
ping 172.16.60.254
ping 172.16.61.1
ping -b 172.16.60.255
```

TUX2:

```
ping -b 172.16.61.255
```

2.3 – Experiência 3

TUX4:

```
ifconfig eth1 up
ifconfig eth1 172.16.61.253/24
sysctl net.ipv4.ip_forward=1
sysctl net.ipv4.icmp_echo_ignore_broadcasts=0
```

Switch Console:

```
/interface bridge port remove [find interface=ether10]
/interface bridge port add bridge=bridge61 interface=ether10
```

TUX2:

```
route add -net 172.16.60.0/24 gw 172.16.61.253
```

TUX3:

```
route add -net 172.16.61.0/24 gw 172.16.60.254
```

TUX2, TUX3, TUX4:

```
route -n
```

TUX3:

```
ping 172.16.60.254
ping 172.16.61.253
ping 172.16.61.1
```

TUX2:

```
arp -d 172.16.61.253
```

TUX4:

```
arp -d 172.16.60.1
arp -d 172.16.61.1
```

TUX3:

```
arp -d 172.16.60.254
ping 172.16.61.1
```


2.4 – Experiência 4

Switch Console:

```
/interface bridge port remove [find interface=ether14]
/interface bridge port add bridge=bridge61 interface=ether14
```

Router Console:

```
/system reset-configuration
/ip address add address=172.16.1.69/24 interface=ether1
/ip address add address=172.16.61.254/24 interface=ether2
/ip route add dst-address=172.16.60.0/24 gateway=172.16.61.253
/ip route add dst-address=0.0.0.0/0 gateway=172.16.1.254
```

TUX3:

```
route add default gw 172.16.60.254
```

TUX2, TUX4:

```
route add default gw 172.16.61.254
```

TUX3:

```
ping 172.16.60.254
ping 172.16.61.253
ping 172.16.61.1
ping 172.16.61.254
```

TUX2:

```
sysctl net.ipv4.conf.eth0.accept_redirects=0
sysctl net.ipv4.conf.all.accept_redirects=0
route del -net 172.16.60.0/24 gw 172.16.61.253
ping 172.16.60.1
traceroute -n 172.16.60.1
route add -net 172.16.60.0/24 gw 172.16.61.253
traceroute -n 172.16.60.1
sysctl net.ipv4.conf.eth0.accept_redirects=1
sysctl net.ipv4.conf.all.accept_redirects=1
```

TUX3:

```
ping 172.16.1.254
```

Router Console:

```
/ip firewall nat disable 0
```

TUX3:

```
ping 172.16.1.254
```

Router Console:

```
/ip firewall nat enable 0
```

2.5 – Experiência 5

TUX2, TUX3, TUX4:

```
sudo cat /etc/resolv.conf
sudo nano /etc/resolv.conf
->    nameserver 172.16.1.1
ping google.com
```

2.5 – Experiência 6

TUX3:

```
./download ftp://rcom:rcom@netlab1.fe.up.pt/pipe.txt
./download ftp://rcom:rcom@netlab1.fe.up.pt/files/crab.mp4
```

TUX2, TUX3: (simultaneously)

```
./download ftp://rcom:rcom@netlab1.fe.up.pt/pipe.txt
```

Anexo 3 – Logs Capturados

3.1 – Experiência 1

1. Ping Tux3 para Tux4

4	4.259884935	172.16.60.1	172.16.60.254	ICMP	98 Echo (ping) request	id=0x4746, seq=1/256, ttl=64 (reply in 5)
5	4.260055626	172.16.60.254	172.16.60.1	ICMP	98 Echo (ping) reply	id=0x4746, seq=1/256, ttl=64 (request in 4)
6	5.262795926	172.16.60.1	172.16.60.254	ICMP	98 Echo (ping) request	id=0x4746, seq=2/512, ttl=64 (reply in 7)
7	5.262941196	172.16.60.254	172.16.60.1	ICMP	98 Echo (ping) reply	id=0x4746, seq=2/512, ttl=64 (request in 6)
8	6.006972464	Routerboardc_1c:...	Spanning-tree-(f...	STP	60 RST. Root = 32768/0/c4:ad:34:1c:8b:e8	Cost = 0 Port = 0x0001
9	6.286792574	172.16.60.1	172.16.60.254	ICMP	98 Echo (ping) request	id=0x4746, seq=3/768, ttl=64 (reply in 10)
10	6.286932955	172.16.60.254	172.16.60.1	ICMP	98 Echo (ping) reply	id=0x4746, seq=3/768, ttl=64 (request in 9)
11	7.310792226	172.16.60.1	172.16.60.254	ICMP	98 Echo (ping) request	id=0x4746, seq=4/1024, ttl=64 (reply in 12)
12	7.310965781	172.16.60.254	172.16.60.1	ICMP	98 Echo (ping) reply	id=0x4746, seq=4/1024, ttl=64 (request in 11)

3.2 – Experiência 2

1. Ping Tux3 para Tux4

4	4.259884935	172.16.60.1	172.16.60.254	ICMP	98 Echo (ping) request	id=0x4746, seq=1/256, ttl=64 (reply in 5)
5	4.260055626	172.16.60.254	172.16.60.1	ICMP	98 Echo (ping) reply	id=0x4746, seq=1/256, ttl=64 (request in 4)
6	5.262795926	172.16.60.1	172.16.60.254	ICMP	98 Echo (ping) request	id=0x4746, seq=2/512, ttl=64 (reply in 7)
7	5.262941196	172.16.60.254	172.16.60.1	ICMP	98 Echo (ping) reply	id=0x4746, seq=2/512, ttl=64 (request in 6)
8	6.006972464	Routerboardc_1c:...	Spanning-tree-(f...	STP	60 RST. Root = 32768/0/c4:ad:34:1c:8b:e8	Cost = 0 Port = 0x0001
9	6.286792574	172.16.60.1	172.16.60.254	ICMP	98 Echo (ping) request	id=0x4746, seq=3/768, ttl=64 (reply in 10)
10	6.286932955	172.16.60.254	172.16.60.1	ICMP	98 Echo (ping) reply	id=0x4746, seq=3/768, ttl=64 (request in 9)
11	7.310792226	172.16.60.1	172.16.60.254	ICMP	98 Echo (ping) request	id=0x4746, seq=4/1024, ttl=64 (reply in 12)
12	7.310965781	172.16.60.254	172.16.60.1	ICMP	98 Echo (ping) reply	id=0x4746, seq=4/1024, ttl=64 (request in 11)

2. Ping Tux3 para Tux2

51	44.1431921...	172.16.60.1	172.16.61.1	ICMP	98 Echo (ping) request	id=0x4e59, seq=1/256, ttl=64 (reply in 52)
52	44.1435150...	172.16.61.1	172.16.60.1	ICMP	98 Echo (ping) reply	id=0x4e59, seq=1/256, ttl=63 (request in 51)
53	45.1684118...	172.16.60.1	172.16.61.1	ICMP	98 Echo (ping) request	id=0x4e59, seq=2/512, ttl=64 (reply in 54)
54	45.1687146...	172.16.61.1	172.16.60.1	ICMP	98 Echo (ping) reply	id=0x4e59, seq=2/512, ttl=63 (request in 53)
55	46.0366555...	Routerboardc_1c:...	Spanning-tree-(f...	STP	60 RST. Root = 32768/0/c4:ad:34:1c:8b:e8	Cost = 0 Port = 0x0001
56	46.1924121...	172.16.60.1	172.16.61.1	ICMP	98 Echo (ping) request	id=0x4e59, seq=3/768, ttl=64 (reply in 57)
57	46.1926637...	172.16.61.1	172.16.60.1	ICMP	98 Echo (ping) reply	id=0x4e59, seq=3/768, ttl=63 (request in 56)
58	47.2164124...	172.16.60.1	172.16.61.1	ICMP	98 Echo (ping) request	id=0x4e59, seq=4/1024, ttl=64 (reply in 59)
59	47.2166778...	172.16.61.1	172.16.60.1	ICMP	98 Echo (ping) reply	id=0x4e59, seq=4/1024, ttl=63 (request in 58)
60	48.0390440...	Routerboardc_1c:...	Spanning-tree-(f...	STP	60 RST. Root = 32768/0/c4:ad:34:1c:8b:e8	Cost = 0 Port = 0x0001
61	48.2404134...	172.16.60.1	172.16.61.1	ICMP	98 Echo (ping) request	id=0x4e59, seq=5/1280, ttl=64 (reply in 62)
62	48.2406656...	172.16.61.1	172.16.60.1	ICMP	98 Echo (ping) reply	id=0x4e59, seq=5/1280, ttl=63 (request in 61)
63	49.2644157...	172.16.60.1	172.16.61.1	ICMP	98 Echo (ping) request	id=0x4e59, seq=6/1536, ttl=64 (reply in 64)
64	49.2647143...	172.16.61.1	172.16.60.1	ICMP	98 Echo (ping) reply	id=0x4e59, seq=6/1536, ttl=63 (request in 63)

3. Ping Tux3 para Broadcast

9	14.4242992...	172.16.60.1	172.16.60.255	ICMP	98 Echo (ping) request	id=0x4827, seq=1/256, ttl=64 (no response found!)
10	14.4246029...	172.16.60.254	172.16.60.1	ICMP	98 Echo (ping) reply	id=0x4827, seq=1/256, ttl=64
11	15.4516891...	172.16.60.1	172.16.60.255	ICMP	98 Echo (ping) request	id=0x4827, seq=2/512, ttl=64 (no response found!)
12	15.4518930...	172.16.60.254	172.16.60.1	ICMP	98 Echo (ping) reply	id=0x4827, seq=2/512, ttl=64
13	16.0088710...	Routerboardc_1c:...	Spanning-tree-(f...	STP	60 RST. Root = 32768/0/c4:ad:34:1c:8b:e8	Cost = 0 Port = 0x0001
14	16.4756830...	172.16.60.1	172.16.60.255	ICMP	98 Echo (ping) request	id=0x4827, seq=3/768, ttl=64 (no response found!)
15	16.4758829...	172.16.60.254	172.16.60.1	ICMP	98 Echo (ping) reply	id=0x4827, seq=3/768, ttl=64
16	17.4996837...	172.16.60.1	172.16.60.255	ICMP	98 Echo (ping) request	id=0x4827, seq=4/1024, ttl=64 (no response found!)
17	17.4998898...	172.16.60.254	172.16.60.1	ICMP	98 Echo (ping) reply	id=0x4827, seq=4/1024, ttl=64
18	18.0113080...	Routerboardc_1c:...	Spanning-tree-(f...	STP	60 RST. Root = 32768/0/c4:ad:34:1c:8b:e8	Cost = 0 Port = 0x0001
19	18.5236992...	172.16.60.1	172.16.60.255	ICMP	98 Echo (ping) request	id=0x4827, seq=5/1280, ttl=64 (no response found!)
20	18.5239099...	172.16.60.254	172.16.60.1	ICMP	98 Echo (ping) reply	id=0x4827, seq=5/1280, ttl=64

4. Ping Tux2 para Broadcast

10	17.3180034...	172.16.61.1	172.16.61.255	ICMP	98 Echo (ping) request	id=0x06e0, seq=1/256, ttl=64 (no response found!)
11	18.0112926...	Routerboardc_1c:...	Spanning-tree-(f...	STP	60 RST. Root = 32768/0/c4:ad:34:1c:8b:ef	Cost = 0 Port = 0x0001
12	18.3331607...	172.16.61.1	172.16.61.255	ICMP	98 Echo (ping) request	id=0x06e0, seq=2/512, ttl=64 (no response found!)
13	19.3571607...	172.16.61.1	172.16.61.255	ICMP	98 Echo (ping) request	id=0x06e0, seq=3/768, ttl=64 (no response found!)
14	20.0134134...	Routerboardc_1c:...	Spanning-tree-(f...	STP	60 RST. Root = 32768/0/c4:ad:34:1c:8b:ef	Cost = 0 Port = 0x0001
15	20.3811667...	172.16.61.1	172.16.61.255	ICMP	98 Echo (ping) request	id=0x06e0, seq=4/1024, ttl=64 (no response found!)
16	21.4051591...	172.16.61.1	172.16.61.255	ICMP	98 Echo (ping) request	id=0x06e0, seq=5/1280, ttl=64 (no response found!)

3.3 – Experiência 3

1. Ping Tux3 para Tux4.eth0

4	4.259884935	172.16.60.1	172.16.60.254	ICMP	98 Echo (ping) request	id=0x4746, seq=1/256, ttl=64 (reply in 5)
5	4.260055626	172.16.60.254	172.16.60.1	ICMP	98 Echo (ping) reply	id=0x4746, seq=1/256, ttl=64 (request in 4)
6	5.262795926	172.16.60.1	172.16.60.254	ICMP	98 Echo (ping) request	id=0x4746, seq=2/512, ttl=64 (reply in 7)
7	5.262941196	172.16.60.254	172.16.60.1	ICMP	98 Echo (ping) reply	id=0x4746, seq=2/512, ttl=64 (request in 6)
8	6.006972464	Routerboardc_1c:...	Spanning-tree-(f...	STP	60 RST. Root = 32768/0/c4:ad:34:1c:8b:e8	Cost = 0 Port = 0x0001
9	6.286792574	172.16.60.1	172.16.60.254	ICMP	98 Echo (ping) request	id=0x4746, seq=3/768, ttl=64 (reply in 10)
10	6.286932955	172.16.60.254	172.16.60.1	ICMP	98 Echo (ping) reply	id=0x4746, seq=3/768, ttl=64 (request in 9)
11	7.310792226	172.16.60.1	172.16.60.254	ICMP	98 Echo (ping) request	id=0x4746, seq=4/1024, ttl=64 (reply in 12)
12	7.310965781	172.16.60.254	172.16.60.1	ICMP	98 Echo (ping) reply	id=0x4746, seq=4/1024, ttl=64 (request in 11)

2. Ping Tux3 para Tux4.eth1

30	29.4793967...	172.16.60.1	172.16.61.253	ICMP	98 Echo (ping) request	id=0x4e4f, seq=1/256, ttl=64 (reply in 31)
31	29.4795710...	172.16.61.253	172.16.60.1	ICMP	98 Echo (ping) reply	id=0x4e4f, seq=1/256, ttl=64 (request in 30)
32	30.0271364...	Routerboardc_1c:...	Spanning-tree-(f...	STP	60 RST. Root = 32768/0/c4:ad:34:1c:8b:e8	Cost = 0 Port = 0x0001
33	30.2364315...	0.0.0.0	255.255.255.255	MNDP	182 5678 → 5678 Len=140	
34	30.2364877...	Routerboardc_1c:...	CDP/VTP/DTP/PagP...	CDP	116 Device ID: MikroTik	Port ID: bridge60/ether6
35	30.2365845...	Routerboardc_1c:...	LLDP_Multicast	LLDP	133 MA/c4:ad:34:1c:8b:e3	IN/bridge60/ether6 120 SysN=MikroTik SysD=MikroTik Router05
36	30.4804554...	172.16.60.1	172.16.61.253	ICMP	98 Echo (ping) request	id=0x4e4f, seq=2/512, ttl=64 (reply in 37)
37	30.4805881...	172.16.61.253	172.16.60.1	ICMP	98 Echo (ping) reply	id=0x4e4f, seq=2/512, ttl=64 (request in 36)
38	31.5044589...	172.16.60.1	172.16.61.253	ICMP	98 Echo (ping) request	id=0x4e4f, seq=3/768, ttl=64 (reply in 39)
39	31.5046071...	172.16.61.253	172.16.60.1	ICMP	98 Echo (ping) reply	id=0x4e4f, seq=3/768, ttl=64 (request in 38)
40	32.0294494...	Routerboardc_1c:...	Spanning-tree-(f...	STP	60 RST. Root = 32768/0/c4:ad:34:1c:8b:e8	Cost = 0 Port = 0x0001
41	32.5284601...	172.16.60.1	172.16.61.253	ICMP	98 Echo (ping) request	id=0x4e4f, seq=4/1024, ttl=64 (reply in 42)
42	32.5285966...	172.16.61.253	172.16.60.1	ICMP	98 Echo (ping) reply	id=0x4e4f, seq=4/1024, ttl=64 (request in 41)
43	33.5524233...	172.16.60.1	172.16.61.253	ICMP	98 Echo (ping) request	id=0x4e4f, seq=5/1280, ttl=64 (reply in 44)
44	33.5525720...	172.16.61.253	172.16.60.1	ICMP	98 Echo (ping) reply	id=0x4e4f, seq=5/1280, ttl=64 (request in 43)

3. Ping Tux3 para Tux2

51	44.1431921...	172.16.60.1	172.16.61.1	ICMP	98 Echo (ping) request	id=0x4e59, seq=1/256, ttl=64 (reply in 52)
52	44.1435150...	172.16.61.1	172.16.60.1	ICMP	98 Echo (ping) reply	id=0x4e59, seq=1/256, ttl=63 (request in 51)
53	45.1684118...	172.16.60.1	172.16.61.1	ICMP	98 Echo (ping) request	id=0x4e59, seq=2/512, ttl=64 (reply in 54)
54	45.1687146...	172.16.61.1	172.16.60.1	ICMP	98 Echo (ping) reply	id=0x4e59, seq=2/512, ttl=63 (request in 53)
55	46.0366555...	Routerboardc_1c:...	Spanning-tree-(f...	STP	60 RST. Root = 32768/0/c4:ad:34:1c:8b:e8	Cost = 0 Port = 0x0001
56	46.1924121...	172.16.60.1	172.16.61.1	ICMP	98 Echo (ping) request	id=0x4e59, seq=3/768, ttl=64 (reply in 57)
57	46.1926637...	172.16.61.1	172.16.60.1	ICMP	98 Echo (ping) reply	id=0x4e59, seq=3/768, ttl=63 (request in 56)
58	47.2164124...	172.16.60.1	172.16.61.1	ICMP	98 Echo (ping) request	id=0x4e59, seq=4/1024, ttl=64 (reply in 59)
59	47.2166778...	172.16.61.1	172.16.60.1	ICMP	98 Echo (ping) reply	id=0x4e59, seq=4/1024, ttl=63 (request in 58)
60	48.0390440...	Routerboardc_1c:...	Spanning-tree-(f...	STP	60 RST. Root = 32768/0/c4:ad:34:1c:8b:e8	Cost = 0 Port = 0x0001
61	48.2404134...	172.16.61.1	172.16.61.1	ICMP	98 Echo (ping) request	id=0x4e59, seq=5/1280, ttl=64 (reply in 62)
62	48.2406656...	172.16.61.1	172.16.60.1	ICMP	98 Echo (ping) reply	id=0x4e59, seq=5/1280, ttl=63 (request in 61)
63	49.2644157...	172.16.60.1	172.16.61.1	ICMP	98 Echo (ping) request	id=0x4e59, seq=6/1536, ttl=64 (reply in 64)
64	49.2647143...	172.16.61.1	172.16.60.1	ICMP	98 Echo (ping) reply	id=0x4e59, seq=6/1536, ttl=63 (request in 63)

4. Ping Tux3 para Tux2 capturado no Tux4

98	175.349529...	HewlettPacka_61:...	Broadcast	ARP	60 Who has 172.16.60.254? Tell 172.16.60.1	
99	175.349555...	HewlettPacka_c5:...	HewlettPacka_61:...	ARP	42 172.16.60.254 is at 00:21:5a:c5:61:bb	
100	175.349683...	172.16.60.1	172.16.61.1	ICMP	98 Echo (ping) request	id=0x6fcb, seq=1/256, ttl=64 (reply in 101)
101	175.350044...	172.16.61.1	172.16.60.1	ICMP	98 Echo (ping) reply	id=0x6fcb, seq=1/256, ttl=63 (request in 100)
102	176.196372...	Routerboardc_1c:...	Spanning-tree-(f...	STP	60 RST. Root = 32768/0/c4:ad:34:1c:8b:e6	Cost = 0 Port = 0x0002
103	176.375214...	172.16.60.1	172.16.61.1	ICMP	98 Echo (ping) request	id=0x6fcb, seq=2/512, ttl=64 (reply in 104)
104	176.375404...	172.16.61.1	172.16.60.1	ICMP	98 Echo (ping) reply	id=0x6fcb, seq=2/512, ttl=63 (request in 103)
105	177.399205...	172.16.60.1	172.16.61.1	ICMP	98 Echo (ping) request	id=0x6fcb, seq=3/768, ttl=64 (reply in 106)
106	177.399372...	172.16.61.1	172.16.60.1	ICMP	98 Echo (ping) reply	id=0x6fcb, seq=3/768, ttl=63 (request in 105)
107	178.198877...	Routerboardc_1c:...	Spanning-tree-(f...	STP	60 RST. Root = 32768/0/c4:ad:34:1c:8b:e6	Cost = 0 Port = 0x0002
108	178.423166...	172.16.60.1	172.16.61.1	ICMP	98 Echo (ping) request	id=0x6fcb, seq=4/1024, ttl=64 (reply in 109)
109	178.423327...	172.16.61.1	172.16.60.1	ICMP	98 Echo (ping) reply	id=0x6fcb, seq=4/1024, ttl=63 (request in 108)
110	179.447151...	172.16.60.1	172.16.61.1	ICMP	98 Echo (ping) request	id=0x6fcb, seq=5/1280, ttl=64 (reply in 111)
111	179.447317...	172.16.61.1	172.16.60.1	ICMP	98 Echo (ping) reply	id=0x6fcb, seq=5/1280, ttl=63 (request in 110)

3.4 – Experiência 4

1. Ping Tux3 para Tux2

51	44.1431921...	172.16.60.1	172.16.61.1	ICMP	98 Echo (ping) request	id=0x4e59, seq=1/256, ttl=64 (reply in 52)
52	44.1435150...	172.16.61.1	172.16.60.1	ICMP	98 Echo (ping) reply	id=0x4e59, seq=1/256, ttl=63 (request in 51)
53	45.1684118...	172.16.60.1	172.16.61.1	ICMP	98 Echo (ping) request	id=0x4e59, seq=2/512, ttl=64 (reply in 54)
54	45.1687146...	172.16.61.1	172.16.60.1	ICMP	98 Echo (ping) reply	id=0x4e59, seq=2/512, ttl=63 (request in 53)
55	46.0366555...	Routerboardc_1c:...	Spanning-tree-(f...	STP	60 RST. Root = 32768/0/c4:ad:34:1c:8b:e8	Cost = 0 Port = 0x0001
56	46.1924121...	172.16.60.1	172.16.61.1	ICMP	98 Echo (ping) request	id=0x4e59, seq=3/768, ttl=64 (reply in 57)
57	46.1926637...	172.16.61.1	172.16.60.1	ICMP	98 Echo (ping) reply	id=0x4e59, seq=3/768, ttl=63 (request in 56)
58	47.2164124...	172.16.60.1	172.16.61.1	ICMP	98 Echo (ping) request	id=0x4e59, seq=4/1024, ttl=64 (reply in 59)
59	47.2166778...	172.16.61.1	172.16.60.1	ICMP	98 Echo (ping) reply	id=0x4e59, seq=4/1024, ttl=63 (request in 58)
60	48.0390440...	Routerboardc_1c:...	Spanning-tree-(f...	STP	60 RST. Root = 32768/0/c4:ad:34:1c:8b:e8	Cost = 0 Port = 0x0001
61	48.2404134...	172.16.60.1	172.16.61.1	ICMP	98 Echo (ping) request	id=0x4e59, seq=5/1280, ttl=64 (reply in 62)
62	48.2406656...	172.16.61.1	172.16.60.1	ICMP	98 Echo (ping) reply	id=0x4e59, seq=5/1280, ttl=63 (request in 61)
63	49.2644157...	172.16.60.1	172.16.61.1	ICMP	98 Echo (ping) request	id=0x4e59, seq=6/1536, ttl=64 (reply in 64)
64	49.2647143...	172.16.61.1	172.16.60.1	ICMP	98 Echo (ping) reply	id=0x4e59, seq=6/1536, ttl=63 (request in 63)

2. Ping Tux3 para Tux4.eth0

4	4.259884935	172.16.60.1	172.16.60.254	ICMP	98 Echo (ping) request	id=0x4746, seq=1/256, ttl=64 (reply in 5)
5	4.260055626	172.16.60.1	172.16.60.1	ICMP	98 Echo (ping) reply	id=0x4746, seq=1/256, ttl=64 (request in 4)
6	5.262795926	172.16.60.1	172.16.60.254	ICMP	98 Echo (ping) request	id=0x4746, seq=2/512, ttl=64 (reply in 7)
7	5.262941196	172.16.60.254	172.16.60.1	ICMP	98 Echo (ping) reply	id=0x4746, seq=2/512, ttl=64 (request in 6)
8	6.006972464	Routerboardc_1c:...	Spanning-tree-(f...	STP	60 RST. Root = 32768/0/c4:ad:34:1c:8b:e8	Cost = 0 Port = 0x0001
9	6.286792574	172.16.60.1	172.16.60.254	ICMP	98 Echo (ping) request	id=0x4746, seq=3/768, ttl=64 (reply in 10)
10	6.286932955	172.16.60.254	172.16.60.1	ICMP	98 Echo (ping) reply	id=0x4746, seq=3/768, ttl=64 (request in 9)
11	7.310792226	172.16.60.1	172.16.60.254	ICMP	98 Echo (ping) request	id=0x4746, seq=4/1024, ttl=64 (reply in 12)
12	7.310965781	172.16.60.254	172.16.60.1	ICMP	98 Echo (ping) reply	id=0x4746, seq=4/1024, ttl=64 (request in 11)

3. Ping Tux3 para Tux4.eth1

30	29.4793967...	172.16.60.1	172.16.61.253	ICMP	98 Echo (ping) request	id=0x4e4f, seq=1/256, ttl=64 (reply in 31)
31	29.4795710...	172.16.61.253	172.16.60.1	ICMP	98 Echo (ping) reply	id=0x4e4f, seq=1/256, ttl=64 (request in 30)
32	30.0271364...	Routerboardc_1c:...	Spanning-tree-(f...	STP	60 RST. Root = 32768/0/c4:ad:34:1c:8b:e8	Cost = 0 Port = 0x0001
33	30.2364315...	0.0.0.0	255.255.255.255	MNDP	182 5678 → 5678 Len=140	
34	30.2364877...	Routerboardc_1c:...	CDP/VTP/DTP/PagP...	CDP	116 Device ID: MikroTik Port ID: bridge60/ether6	
35	30.2365845...	Routerboardc_1c:...	LLDP_Multicast	LLDP	133 MA/c4:ad:34:1c:8b:e3 IN/bridge60/ether6 120 SysN=MikroTik SysD=MikroTik RouterOS	
36	30.4804554...	172.16.60.1	172.16.61.253	ICMP	98 Echo (ping) request	id=0x4e4f, seq=2/512, ttl=64 (reply in 37)
37	30.4805881...	172.16.61.253	172.16.60.1	ICMP	98 Echo (ping) reply	id=0x4e4f, seq=2/512, ttl=64 (request in 36)
38	31.5044589...	172.16.60.1	172.16.61.253	ICMP	98 Echo (ping) request	id=0x4e4f, seq=3/768, ttl=64 (reply in 39)
39	31.5046071...	172.16.61.253	172.16.60.1	ICMP	98 Echo (ping) reply	id=0x4e4f, seq=3/768, ttl=64 (request in 38)
40	32.0294494...	Routerboardc_1c:...	Spanning-tree-(f...	STP	60 RST. Root = 32768/0/c4:ad:34:1c:8b:e8	Cost = 0 Port = 0x0001
41	32.5284601...	172.16.60.1	172.16.61.253	ICMP	98 Echo (ping) request	id=0x4e4f, seq=4/1024, ttl=64 (reply in 42)
42	32.5285966...	172.16.61.253	172.16.60.1	ICMP	98 Echo (ping) reply	id=0x4e4f, seq=4/1024, ttl=64 (request in 41)
43	33.5524233...	172.16.60.1	172.16.61.253	ICMP	98 Echo (ping) request	id=0x4e4f, seq=5/1280, ttl=64 (reply in 44)
44	33.5525720...	172.16.61.253	172.16.60.1	ICMP	98 Echo (ping) reply	id=0x4e4f, seq=5/1280, ttl=64 (request in 43)

4. Ping Tux3 para RC

63	61.3237240...	172.16.60.1	172.16.61.254	ICMP	98 Echo (ping) request	id=0x73e5, seq=1/256, ttl=64 (reply in 64)
64	61.3240794...	172.16.61.254	172.16.60.1	ICMP	98 Echo (ping) reply	id=0x73e5, seq=1/256, ttl=63 (request in 63)
65	62.0554611...	Routerboardc_1c:...	Spanning-tree-(f...	STP	60 RST. Root = 32768/0/c4:ad:34:1c:8b:e6	Cost = 0 Port = 0x0001
66	62.3538380...	172.16.60.1	172.16.61.254	ICMP	98 Echo (ping) request	id=0x73e5, seq=2/512, ttl=64 (reply in 67)
67	62.3541153...	172.16.61.254	172.16.60.1	ICMP	98 Echo (ping) reply	id=0x73e5, seq=2/512, ttl=63 (request in 66)
68	63.3778380...	172.16.60.1	172.16.61.254	ICMP	98 Echo (ping) request	id=0x73e5, seq=3/768, ttl=64 (reply in 69)
69	63.3781494...	172.16.61.254	172.16.60.1	ICMP	98 Echo (ping) reply	id=0x73e5, seq=3/768, ttl=63 (request in 68)
70	64.0575240...	Routerboardc_1c:...	Spanning-tree-(f...	STP	60 RST. Root = 32768/0/c4:ad:34:1c:8b:e6	Cost = 0 Port = 0x0001
71	64.4018248...	172.16.60.1	172.16.61.254	ICMP	98 Echo (ping) request	id=0x73e5, seq=4/1024, ttl=64 (reply in 72)
72	64.4020963...	172.16.61.254	172.16.60.1	ICMP	98 Echo (ping) reply	id=0x73e5, seq=4/1024, ttl=63 (request in 71)
73	65.4258343...	172.16.60.1	172.16.61.254	ICMP	98 Echo (ping) request	id=0x73e5, seq=5/1280, ttl=64 (reply in 74)
74	65.4261315...	172.16.61.254	172.16.60.1	ICMP	98 Echo (ping) reply	id=0x73e5, seq=5/1280, ttl=63 (request in 73)

6. Ping Tux2 para Tux3 sem redirects

16	23.0892069...	172.16.61.1	172.16.60.1	ICMP	98 Echo (ping) request	id=0x78a4, seq=1/256, ttl=64 (reply in 17)
17	23.0896203...	172.16.60.1	172.16.61.1	ICMP	98 Echo (ping) reply	id=0x78a4, seq=1/256, ttl=63 (request in 16)
18	24.0257609...	Routerboardc_1c:...	Spanning-tree-(f...	STP	60 RST. Root = 32768/0/74:4d:28:eb:23:fc	Cost = 10 Port = 0x0001
19	24.1175561...	172.16.61.1	172.16.60.1	ICMP	98 Echo (ping) request	id=0x78a4, seq=2/512, ttl=64 (reply in 21)
20	24.1177272...	172.16.61.254	172.16.61.1	ICMP	126 Redirect	(Redirect for host)
21	24.1179266...	172.16.60.1	172.16.61.1	ICMP	98 Echo (ping) reply	id=0x78a4, seq=2/512, ttl=63 (request in 19)
22	25.1415592...	172.16.61.1	172.16.60.1	ICMP	98 Echo (ping) request	id=0x78a4, seq=3/768, ttl=64 (reply in 24)
23	25.1417222...	172.16.61.254	172.16.61.1	ICMP	126 Redirect	(Redirect for host)
24	25.1419453...	172.16.60.1	172.16.61.1	ICMP	98 Echo (ping) reply	id=0x78a4, seq=3/768, ttl=63 (request in 22)
25	26.0280359...	Routerboardc_1c:...	Spanning-tree-(f...	STP	60 RST. Root = 32768/0/74:4d:28:eb:23:fc	Cost = 10 Port = 0x0001
26	26.1655540...	172.16.61.1	172.16.60.1	ICMP	98 Echo (ping) request	id=0x78a4, seq=4/1024, ttl=64 (reply in 28)
27	26.1657196...	172.16.61.254	172.16.61.1	ICMP	126 Redirect	(Redirect for host)
28	26.1659238...	172.16.60.1	172.16.61.1	ICMP	98 Echo (ping) reply	id=0x78a4, seq=4/1024, ttl=63 (request in 26)
29	27.1895616...	172.16.61.1	172.16.60.1	ICMP	98 Echo (ping) request	id=0x78a4, seq=5/1280, ttl=64 (reply in 31)
30	27.1897286...	172.16.61.254	172.16.61.1	ICMP	126 Redirect	(Redirect for host)
31	27.1899288...	172.16.60.1	172.16.61.1	ICMP	98 Echo (ping) reply	id=0x78a4, seq=5/1280, ttl=63 (request in 29)

3.5 – Experiência 5

1. DNS Resolve

95	95.1960046...	172.16.60.1	172.16.1.1	DNS	75 Standard query 0xf87f A 172.16.1.69.254	
96	95.1960148...	172.16.60.1	172.16.1.1	DNS	75 Standard query 0xd089 AAAA 172.16.1.69.254	
97	95.2024745...	172.16.1.1	172.16.60.1	DNS	150 Standard query response 0xf87f No such name A 172.16.1.69.254 SOA a.root-servers.net	
98	95.2025411...	172.16.1.1	172.16.60.1	DNS	150 Standard query response 0xd089 No such name AAAA 172.16.1.69.254 SOA a.root-servers.net	
99	95.2026008...	172.16.60.1	172.16.1.1	DNS	91 Standard query 0xa276 A 172.16.1.69.254.netlab.fe.up.pt	
100	95.2026059...	172.16.60.1	172.16.1.1	DNS	91 Standard query 0x837e AAAA 172.16.1.69.254.netlab.fe.up.pt	
101	95.2030681...	172.16.1.1	172.16.60.1	DNS	166 Standard query response 0xa276 No such name A 172.16.1.69.254.netlab.fe.up.pt SOA ns.netlab.fe.up.pt	
102	95.2030742...	172.16.1.1	172.16.60.1	DNS	166 Standard query response 0x837e No such name AAAA 172.16.1.69.254.netlab.fe.up.pt SOA ns.netlab.fe.up.pt	
103	95.2031117...	172.16.60.1	172.16.1.1	DNS	84 Standard query 0xaa07 A 172.16.1.69.254.fe.up.pt	
104	95.2031166...	172.16.60.1	172.16.1.1	DNS	84 Standard query 0x9d0d AAAA 172.16.1.69.254.fe.up.pt	
105	95.2429206...	172.16.1.1	172.16.60.1	DNS	138 Standard query response 0xaa07 No such name A 172.16.1.69.254.fe.up.pt SOA dns1.fe.up.pt	
106	95.2429283...	172.16.1.1	172.16.60.1	DNS	138 Standard query response 0x9d0d No such name AAAA 172.16.1.69.254.fe.up.pt SOA dns1.fe.up.pt	

3.6 – Experiência 6

1	0.000000000	172.19.238.154	172.19.224.1	DNS	76 Standard query 0x80f4 A netlab1.fe.up.pt	
2	0.024931755	172.19.224.1	172.19.238.154	DNS	302 Standard query response 0x80f4 A netlab1.fe.up.pt A 192.168.109.136 A 193.136.28.10 A 193.136.28.9 A 10.227.244.110 A 10.	
3	0.025136114	172.19.238.154	192.168.109.136	TCP	74 55048 → 21 [SYN] Seq=0 Win=65500 Len=0 MSS=1310 SACK_PERM TSval=2887865519 TSecr=0 WS=128	
4	0.050881880	192.168.109.136	172.19.238.154	TCP	74 21 → 55048 [SYN, ACK] Seq=0 Ack=1 Win=65160 Len=0 MSS=1460 SACK_PERM TSval=608523689 TSecr=2887865519 WS=128	
5	0.050984706	172.19.238.154	192.168.109.136	TCP	66 55048 → 21 [ACK] Seq=1 Ack=1 Win=65536 Len=0 TSval=2887865545 TSecr=608523689	
6	0.075635159	192.168.109.136	172.19.238.154	FTP	100 Response: 220 Welcome to netlab-FTP server	
7	0.075743926	172.19.238.154	192.168.109.136	TCP	66 55048 → 21 [ACK] Seq=1 Ack=35 Win=65536 Len=0 TSval=2887865570 TSecr=608523715	
8	0.075824976	172.19.238.154	192.168.109.136	FTP	76 Request: USER rcom	
9	0.102588676	192.168.109.136	172.19.238.154	TCP	66 21 → 55048 [ACK] Seq=35 Ack=11 Win=65280 Len=0 TSval=608523740 TSecr=2887865570	
10	0.102589127	192.168.109.136	172.19.238.154	FTP	100 Response: 331 Please specify the password.	
11	0.102810261	172.19.238.154	192.168.109.136	FTP	76 Request: PASS rcom	
12	0.125283162	192.168.109.136	172.19.238.154	TCP	66 21 → 55048 [ACK] Seq=69 Ack=21 Win=65280 Len=0 TSval=608523766 TSecr=2887865597	
13	0.151208549	192.168.109.136	172.19.238.154	FTP	89 Response: 230 Login successful.	
14	0.151473154	172.19.238.154	192.168.109.136	FTP	71 Request: PASV	
15	0.175546574	192.168.109.136	172.19.238.154	TCP	66 21 → 55048 [ACK] Seq=92 Ack=26 Win=65280 Len=0 TSval=608523815 TSecr=2887865645	
16	0.175546875	192.168.109.136	172.19.238.154	FTP	120 Response: 227 Entering Passive Mode (192,168,109,136,191,126).	
17	0.175734450	172.19.238.154	192.168.109.136	TCP	74 46316 → 49022 [SYN] Seq=0 Win=65500 Len=0 MSS=1310 SACK_PERM TSval=2887865670 TSecr=0 WS=128	
18	0.201607931	192.168.109.136	172.19.238.154	TCP	74 49022 → 46316 [SYN, ACK] Seq=0 Ack=1 Win=65160 Len=0 MSS=1460 SACK_PERM TSval=608523839 TSecr=2887865670 WS=128	
19	0.201706709	172.19.238.154	192.168.109.136	TCP	66 46316 → 49022 [ACK] Seq=1 Ack=1 Win=65536 Len=0 TSval=2887865696 TSecr=608523839	
20	0.201775781	172.19.238.154	192.168.109.136	FTP	80 Request: RETR pipe.txt	
21	0.227586961	192.168.109.136	172.19.238.154	FTP	134 Response: 150 Opening BINARY mode data connection for pipe.txt (1863 bytes).	
22	0.227587341	192.168.109.136	172.19.238.154	FTP-DL	1929 FTP Data: 1863 bytes (PASV) (RETR pipe.txt)	
23	0.227587392	192.168.109.136	172.19.238.154	TCP	66 49022 → 46316 [FIN, ACK] Seq=1864 Ack=1 Win=65280 Len=0 TSval=608523867 TSecr=2887865696	
24	0.227686020	172.19.238.154	192.168.109.136	TCP	66 46316 → 49022 [ACK] Seq=1 Ack=1864 Win=63872 Len=0 TSval=2887865722 TSecr=608523866	
25	0.277597810	172.19.238.154	192.168.109.136	TCP	66 46316 → 49022 [ACK] Seq=1 Ack=1865 Win=64256 Len=0 TSval=2887865772 TSecr=608523867	
26	0.277844870	172.19.238.154	192.168.109.136	TCP	66 55048 → 21 [ACK] Seq=40 Ack=214 Win=65536 Len=0 TSval=2887865772 TSecr=608523866	
27	0.302242637	192.168.109.136	172.19.238.154	FTP	90 Response: 226 Transfer complete.	
28	0.302274016	172.19.238.154	192.168.109.136	TCP	66 55048 → 21 [ACK] Seq=40 Ack=238 Win=65536 Len=0 TSval=2887865796 TSecr=608523941	
29	0.303895058	172.19.238.154	192.168.109.136	FTP	71 Request: QUIT	
30	0.327636952	192.168.109.136	172.19.238.154	FTP	80 Response: 221 Goodbye.	
31	0.327637352	192.168.109.136	172.19.238.154	TCP	66 21 → 55048 [FIN, ACK] Seq=252 Ack=45 Win=65280 Len=0 TSval=608523967 TSecr=2887865798	
32	0.327727925	172.19.238.154	192.168.109.136	TCP	66 55048 → 21 [FIN, ACK] Seq=45 Ack=253 Win=65536 Len=0 TSval=2887865822 TSecr=608523967	
33	0.327743615	172.19.238.154	192.168.109.136	TCP	66 46316 → 49022 [FIN, ACK] Seq=1 Ack=1865 Win=64256 Len=0 TSval=2887865822 TSecr=608523867	
34	0.351891243	192.168.109.136	172.19.238.154	TCP	66 21 → 55048 [ACK] Seq=253 Ack=46 Win=65280 Len=0 TSval=608523991 TSecr=2887865822	
35	0.351891574	192.168.109.136	172.19.238.154	TCP	66 49022 → 46316 [ACK] Seq=1865 Ack=2 Win=65280 Len=0 TSval=608523991 TSecr=2887865822	