Faculdade de Engenharia da Universidade do Porto

Lab 2 - Aplicação de Download e Configuração e Estudo de uma Rede de Computadores

Redes de Computadores L.EIC025-2024/2025-1S:

Regente: Manuel Alberto Pereira Ricardo

Regente: Rui Pedro de Magalhães Claro Prior

Turma 7:

Professor: Filipe Miguel Monteiro Da Silva E Sousa

Autores:

David dos Santos Carvalho Ranito up202206312@fe.up.pt

Pedro Daniel Freitas João up202204962@fe.up.pt

Tiago Freitas Torres up202208938@fe.up.pt

Data de Entrega:

23/12/2024

Index

Sumário

Este projeto foi realizado no âmbito da unidade curricular de Redes de Computadores que tem como objetivo o desenvolvimento de uma aplicação que implementa o protocolo FTP como descrito no RFC959 e a configuração e análise de uma rede de computadores.

Ao longo do desenvolvimento deste projeto, aplicamos e reforçamos os conteúdos aprendidos nas aulas teóricas incluindo protocolos e configurações de rede e a sua importância.

Introdução

O objetivo deste trabalho é dividido em duas partes: Aplicação de download e Configuração e análise da rede.

Na primeira parte é descrita a aplicação de download que implementa o protocolo FTP, os mecanismos implementados, a sua utilização e apresenta os resultados.

A segunda parte, configuração e análise da rede, está divida por experiências. Em cada experiência descrevemos e explicamos detalhadamente o que fizemos e os comandos utilizados, e respondemos às questões propostas no guião usando, como suporte, as capturas dos logs do Wireshark.

Este projeto foi desenvolvido e testado em ambiente de laboratório o que nos deu uma perspetiva mais prática que complementa e ajuda a entender a vertente mais teórica.

Parte 1 – Aplicação de download

Na primeira parte deste projeto foi desenvolvida uma aplicação de *download* de ficheiros, de acordo com o protocolo FTP (*file transfer protocol*) como descrito no RFC959, usando a sintaxe de URL descrita no RFC1738:

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

No início da aplicação é feito o parsing do URL recebido como argumento, o qual permite-nos extrair e armazenar informações como user, password, nome do host e o caminho do ficheiro (por omissão user e password são anonymous). Após o parsing utiliza-se uma função fornecida chamada getIP que converte o nome do host no endereço IP correspondente. Este IP é usado para criar um socket TCP por meio da função createSocket, conectando-se ao servidor FTP na porta 21, como especificado pelo protocolo.

Em seguida, inicia-se o envio de comandos e receção de respostas ao servidor, utilizando um buffer. A receção das respostas por parte do servidor é realizada através da função readResponse, que verifica se o código recebido é o esperado (normalmente iniciados por um 2 ou 3). O servidor pode enviar respostas com múltiplas linhas, sendo necessário verificar se a seguir ao código da resposta, se correto, um espaço é apresentado por exemplo “226 “, significando que a resposta é final. Se o código não for o esperado, o protocolo termina exibindo uma mensagem de erro.

Inicialmente envia-se os comandos USER e PASS, com as credenciais guardadas do URL, fazendo a autenticação com o servidor. Depois é enviado o comando PASV para o servidor entrar em modo passivo, sendo obtido na resposta do servidor o ip e a porta para abrir o socket que vai receber o ficheiro.

Após a criação do segundo socket, pedimos para transferir o ficheiro desejado enviando o comando RETR com o caminho do ficheiro, fazendo então o download. Depois de receber a confirmação de transferência, enviamos o comando QUIT fechando a conexão com o servidor, finalizando o protocolo.

Resultados

A aplicação comporta-se como esperado, quando executada para ficheiros de tamanho diferentes, com erros no URL em diversos campos e no ambiente laboratorial.

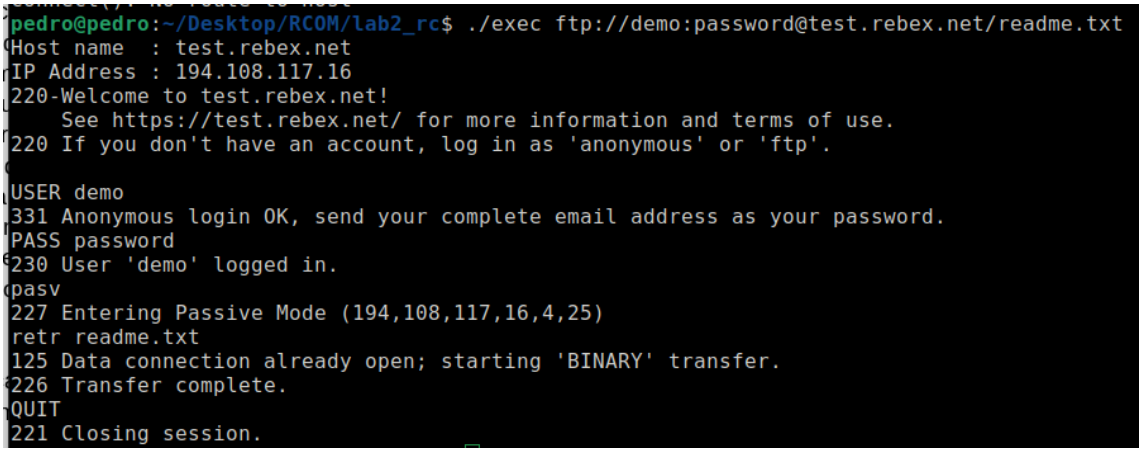


Figura 1: Execução da aplicação

Parte 2 – Configuração e análise da rede

Experiência 1 – Configurar uma rede IP

Conectamos a E1 do tux 73 à porta 3 do switch e a E1 do tux74 à porta 4. Configuramos a interface eth1 de ambos utilizando os comandos:

tux73: ifconfig eth1 up
tux73: ifconfig eth1 172.16.70.1/24
tux74: ifconfig eth1 up
tux74: ifconfig eth1 172.16.70.254/24

tux73(eth1) - IP address - 172.16.70.1 e MAC address - 00:01:02:9f:81:2e
tux74(eth1) - IP address - 172.16.70.254 e MAC address - 00:c0:df:02:55:95

Os packets ARP são utilizados para mapear o IP num MAC, uma vez que é necessário saber qual o endereço correto de hardware a que o pacote tem que ser enviado.

Como vemos nas linhas 10 e 11 do Anexo 3.1.1, dado que limpamos a tabela ARP ao fazermos ping tux74 a partir do tux73, o tux3 não sabe qual o MAC associado ao IP do tux74 e por isso é enviado um pacote ARP request em broadcast de forma a

obter o MAC (src: 172.16.70.1(01:02:9f:81:2e) e dst: 172.16.70.254(ff:ff:ff:ff:ff:ff)), após o tux74 receber este pacote que contém o IP e MAC do remetente, envia um ARP reply diretamente para o tux73 dizendo qual é o seu MAC (src: 172.16.70.254(00:c0:df:02:55:95) dst: 172.16.70.1(00:01:02:9f:81:2e)).

Após ambos saberem os endereços MAC associado a cada IP, começam a ser enviados pacotes ICMP Echo Request e Echo Reply gerados pelo comando ping para testar a conectividade entre os dispositivos. Os pacotes de request (ex. linha 12) têm como origem os endereços MAC e IP do tux73 e destino os endereços MAC e IP do tux74, e os pacotes de reply (ex. linha 13) têm o destino e a origem invertidos.

Para sabermos se um frame Ethernet é ARP ou IP temos que inspecionar o campo EtherType do header, se for 0x0806 é ARP e se for 0x0800 é IP. No caso do ICMP, no header dos pacotes IP o campo Protocol deve ser 1. Para sabermos o tamanho de um frame que recebemos devemos inspecionar o campo Length do header Ethernet

A interface loopback é uma interface virtual no próprio dispositivo que está sempre “up”. É importante porque permite ao host testar o próprio network stack sem necessitar conexões a redes exteriores, diagnosticar problemas, desenvolver e testar serviços locais sem necessidade de configurar uma rede física e garante que alguma funcionalidade está disponível no caso de não ter nenhuma rede externa.

Experiência 2 – Implementar duas bridges num switch

Conectamos a E1 do tux 72 à porta 2 do switch e configuramos a interface eth1 utilizando os comandos:

tux72: ifconfig eth1 up
tux72: ifconfig eth1 172.16.71.1/24

tux72(eth1) - IP address - 172.16.71.1 e MAC address - 00:e0:7d:b5:8c:8f

Para configurarmos as duas bridges conectamos a consola do switch a S0 de um dos tux utilizando o RS232 - Cisco Adapter para converter os sinais da porta série num formato compatível com o switch. Depois no tux escolhido utilizamos o GTKTerm para interagir com o switch.

Começamos por criar duas bridges, a bridge70 e a bridge71:
switch: /interface bridge add name=bridge70
switch: /interface bridge add name=bridge71

As portas do switch usadas pela interface eth1 de cada tux7X são X. Removemos essas portas da bridge default:
switch: /interface bridge port remove [find interface=ether2]
switch: /interface bridge port remove [find interface=ether3]
switch: /interface bridge port remove [find interface=ether4]

E adicionamos as portas 3 e 4 à bridge70 e a 2 à bridge71:

switch: /interface bridge port add bridge=bridge70 interface=ether3

switch: /interface bridge port add bridge=bridge70 interface=ether4

switch: /interface bridge port add bridge=bridge71 interface=ether2

No tux73 quando fazemos ping tux74 verificamos que há conectividade como vemos no Anexo 3.2.1, os ICMP Echo Requests do tux73 para o tux74 são sempre seguidos de ICMP Echo Replies do tux74 para o tux73.

Quando fazemos ping tux72 a partir do tux73 não há resposta. Isto acontece porque o tux73 e o tux72 estão em dois domínios de broadcast diferentes, um para a bridge70 e outro para bridge 71, e o tux73 não possui nenhuma rota configurada para o domínio do tux72.

Podemos observar estes domínios ao fazer ping broadcast (ping -b172.16.70.255) no tux73. Nos Anexo 3.2.2, Anexo 3.2.3, Anexo 3.2.4, respetivamente do tux72, tux73, tux74, conseguimos ver que apenas o tux74 recebeu os pacotes ICMP.

Quando fizemos ping broadcast (ping -b 172.16.71.255) no tux72, nenhum dos outros dois recebeu os pacotes ICMP (Anexo 3.2.5)

Experiência 3 – Configurar um router em Linux

Para transformar o tux74 num router, começamos por configurar a interface eth2 do tux74:

tux74: ifconfig eth2 up

tux74: ifconfig eth2 172.16.71.253/24

tux74(eth2) - IP address - 172.16.41.253 e MAC address 00:01:02:a0:ad:91

De seguida conectamos a E2 do tux74 à porta 8 do switch, removemos a ether8 da bridge default e adicionamos à bridge71:

switch: /interface bridge port remove [find interface=ether8]

switch: /interface bridge port add bridge=bridge71 interface=ether8

No tux74 habilitamos o IP forwarding e desabilitamos o ICMP echo-ignore-broadcast, através dos seguintes comandos, respetivamente:

tux74: sysctl net.ipv4.ip_forward=1

tux74: sysctl net.ipv4.icmp_echo_ignore_broadcasts=0

Agora que o tux74 é um router podemos adicionar rotas para que os tux73 e tux74 possam comunicar entre eles, passando pelo tux74. Adicionamos estas rotas com os comandos:

tux72: route add -net 172.16.70.0/24 gw 172.16.71.253

tux73: route add -net 172.16.71.0/24 gw 172.16.70.254

Rotas do tux72:

Uma rota para rede local 172.16.71.0/24, com o gateway 0.0.0.0 e interface eth1 que significa que a conexão é feita diretamente pela eth1.

Uma rota para a rede da bridge70 172.16.70.0/24 com o gateway 172.16.71.253 e interface eth1. Isto significa que para ir para esta rede envia os pacotes para a interface do router tux74 que está na rede da bridge71

Rotas do tux73:

Uma rota para rede local 172.16.70.0/24, com o gateway 0.0.0.0 e interface eth1 que significa que a conexão é feita diretamente pela eth1.

Uma rota para a rede da bridge71 172.16.71.0/24, com o gateway 172.16.70.254 e interface eth1. Isto significa que para ir para esta rede envia os pacotes para a interface do router tux74 que está na rede da bridge70.

Rotas do tux74:

Rotas para as redes 172.16.70.0/24 e 172.16.71.0/24, com o gateway 0.0.0.0 e interfaces eth1 e eth2 respetivamente. Isto significa que a conexão com as redes das bridges bridge70 e bridge71 são feitas diretamente pelas interfaces eth1 e eth2 respetivamente.

O tux74 tem visibilidade de ambas as redes e por isso pode encaminhar pacotes entre elas.

Uma entrada da tabela de encaminhamento tem o Destino que é a rede ou host ao qual a rota se aplica, a Máscara de sub-rede que define o tamanho da sub-rede, o Gateway que é o endereço IP para o qual pacotes com esse destino devem ser encaminhados e Interface que indica qual a interface a usar para o próximo “hop”.

Ao fazer ping tux72 a partir do tux73 capturamos os pacotes com o Wireshark na interface eth1 (Anexo 3.3.1) e na eth2 (Anexo 3.3.2) do tux74.

Como o tux72 não está na rede local o tux73 vê a próxima rota, que neste caso é compatível com o endereço para o qual o pacote deve ser enviado. Assim o pacote vai ser enviado através do gateway utilizado para a rede 172.16.71.0/24 que é o 172.16.70.254, este IP pertence a eth1 do tux74.

Uma vez que é necessário saber o MAC address associado a este IP o tux73 envia um pacote ARP em broadcast (src: MAC do tux73 e dst: ff:ff:ff:ff:ff:ff), como se vê na linha 17 do log da eth1.

Após receber esse pacote o tux74 responde com outro pacote ARP (src: MAC eth1 do tux74 e dst: MAC do tux73) para o tux73 onde envia qual o próprio MAC, como é possível visualizar na linha 18 do log da eth1.

Depois de saber o endereço MAC para o qual enviar o pacote ICMP Echo Request o tux73 é possível verificar na linha 19 que é enviado esse pacote (src: IP tux73 (MAC tux73) e dst: tux72 (MAC eth1 tux74)). O IP do destino é o tux72 uma vez que este é o destino do ping e o endereço MAC é da interface eth1 do tux74 dado que este é o próximo “hop”.

O tux74 quer saber qual o MAC do 172.16.70.1 e por isso envia um pacote ARP em broadcast (src: MAC eth2 do tux74 e dst: ff:ff:ff:ff:ff:ff), visível na linha 17 do log da

eth2. Este pacote é respondido pelo tux72, como vemos na linha 18 do log da eth2 (src: MAC tux72 e dst: MAC eth2 do tux74).

Após descobrir o endereço MAC do tux72, a linha 19 do log da eth2 mostra o envio pacote ICMP Echo Request para o tux72 (src: IP tux73 (MAC eth2 do tux74) dst: IP tux72 (MAC tux72)). Apesar da origem do pacote ter sido o tux73 ele é enviado a partir da eth2 do tux74 e por isso esse é o MAC na origem do pacote.

Finalmente, nas linhas 20 dos logs vemos os pacotes ICMP Echo Reply, primeiro a chegar a eth2 do tux74 e depois a sair da eth1 do tux74. A lógica destes pacotes é similar à dos Requests, apenas com a origem e destino invertidas.

Experiência 4 – Configurar um router comercial e implementar NAT

Inicialmente ligou-se uma das entradas do router ao switch na interface 7, e adicionamos a interface à bridge 71, com os seguintes comandos:

switch: /interface bridge port remove [find interface=ether7]

switch: /interface bridge port add bridge=bridge71 interface=ether7

Para configurar o router é necessário iniciar a sessão do router no GTKterm, para isso ligamos o cabo antes ligado ao switch à entrada do router. Na consola do router, adicionamos os IPs referentes a cada interface (servidor e switch) e uma rota que permite a ligação ao tux74.

router: /ip address add address=172.16.1.71/24 interface=ether1

router: /ip address add address=172.16.71.254/24 interface=ether2

router: /ip route add dst-address=172.16.70.0/24 gateway=172.16.71.253

Após configurar o router, é preciso adicionar rotas para cada tux conseguir alcançar o servidor, e verificamos que o tux73 consegue dar ping aos outros tuxs e ao router.

tux73: route add -net 172.16.1.0/24 gw 172.16.70.254

tux74: route add -net 172.16.1.0/24 gw 172.16.71.254

tux72: route add -net 172.16.1.0/24 gw 172.16.71.254

Em seguida no tux72 desativamos os redirects:

tux72: sysctl net.ipv4.conf.eth1.accept_redirects=0

tux72: sysctl net.ipv4.conf.all.accept_redirects=0

Ao mudar as routes, para usar o router como gateway para a subnet 172.17.70.0/24 em vez do tux74, podemos verificar que ao fazer ping do tux72 para o tux73, que apesar de existir uma rota melhor para a conexão (pelo tux74), os packets vão sempre pelo router, uma vez que os redirects estão desativados. Com os redirects ativados podemos verificar que os pacotes são automaticamente redirecionados pela rota mais optimal, neste caso pela rota do tux74. Como podemos ver no Anexo 3.4.1, ao dar ping do tux72 para o tux73, o router encaminha o ICMP echo request pelo tux74 e envia um pacote ICMP Redirect para o tux72 a informar o redirecionamento. Os pacotes ICMP Redirects permitem ao sistema informar o remetente sobre uma rota mais eficiente, otimizando o tráfego

NAT é um mecanismo que traduz um endereço de IP privado em um endereço de IP público, e vice-versa, permitindo um grupo de computadores ser representado por um único endereço de IP. NAT no nosso caso é feito pelo router, este mecanismo permite ao dispositivo atuar como intermediário entre a rede local (privada) e a rede pública (internet). Este mecanismo reduz a necessidade da existência de endereços de IP públicos individuais permitindo a uma rede local usar apenas um, também torna possível esconder a estrutura da rede local do resto do mundo, dificultando o ataque aos dispositivos na rede.

O NAT pode ser ativado através do comando /ip firewall nat add chain=srcnat action=masquerade out-interface=ether1.

Ao tentar fazer ping do tux73 para o server com o NAT desativado resulta no ICMP request não ter nenhuma resposta, apesar de o pacote Echo Request conseguir chegar ao servidor o Echo Reply não consegue chegar ao tux73 uma vez que o endereço privado do tux73 não é traduzido na rede pública, deste modo o pacote é solto, uma vez que o endereço de IP privado não é reconhecido na rede pública.

Experiência 5 – DNS

Para configurar o serviço DNS num host é necessário editar o ficheiro /etc/resolv.conf, limpar todo o conteúdo e adicionar o endereço do IP do servidor DNS através da linha “nameserver x” em que x é o endereço. No nosso caso colocamos “nameserver 10.227.20.3” nos três tuxs.

O log desta experiência está no Anexo 3.5.1. As linhas 36 e 37 mostram pacotes de DNS Query de um dos tux para o servidor DNS com IP 10.227.20.3 na tentativa de descobrir o IP correspondente ao nome ftp.netlab.fe.up.pt. O primeiro tem “A” porque é para descobrir o IPv4 e o segundo tem “AAAA” para descobrir o IPv6.

Os pacotes DNS Query Response são visíveis nas duas seguintes linhas, na primeira podemos ver que o servidor de DNS retornou o IP 172.16.1.10 que é o endereço correto.

Experiência 6 – Conexões TCP

Nesta fase, toda a rede vai ser testada. A nossa aplicação vai fazer o download de um ficheiro no host ftp.netlab.fe.up.pt. O primeiro passo é descobrir o IP do host através de uma query de DNS.

São abertas duas conexões. A primeira é usada para enviar comandos ao servidor. O servidor vai abrir uma nova porta e a segunda conexão vai conectar a essa porta para transferir o ficheiro.

A informação de controle de FTP é transportada na primeira conexão do cliente à porta 21 do servidor.

Como é possível ver na imagem do Anexo 3.6.1, o primeiro passo é o Three-Way-Handshake, feito pelos packets SYN, SYN-ACK e ACK. De seguida são transmitidos comandos e o ficheiro em si. Para fechar a conexão é enviado o packet FIN-ACK.

O mecanismo ARQ, através de sequence/acknowledgement numbers, checksum e window size, serve para garantir a transmissão fiável de dados, responsabilizando-se pelos pacotes perdidos.

O window size (quantidade de dados que pode ser enviada) aumenta até ocorrer algum erro. Caso ocorra um timeout ou 3 ACKs repetidos, o window size diminui e volta a aumentar mais lentamente.

No gráfico do Anexo 3.6.2, vê-se que a window (a verde) cresce exponencialmente no início, e quando um pacote é perdido, diminui.

Cada emissor determina a sua capacidade de transmissão O campo CongestionWindow é usado. Quando a rede está descongestionada, a CongestionWIndow aumenta, tal como a capacidade de transmissão. Numa rede congestionada, o oposto acontece.

A throughput começa lento (Slow-Start), mas cresce exponencialmente, desde que não haja perda de pacotes. Quando ocorre perda de pacotes, diminui e o seu crescimento passa a ser linear. É possível ver isso no gráfico do Anexo 3.6.3: crescimento rápido, congestionamento e diminuição do throughput, e novo crescimento.

Quando começamos uma segunda conexão, o throughput na primeira vai diminuir, pois a capacidade tem de ser dividida entre as duas conexões. É possível ver isso no gráfico, na marca de 8 segundos.

Conclusão

O trabalho consolidou conhecimentos teóricos e práticos sobre protocolos e configuração de redes. Na Parte 1, o desenvolvimento da aplicação FTP ajudou na compreensão do protocolo, do serviço DNS e da utilização de sockets e TCP na linguagem C. Na Parte 2, as experiências abordaram configuração de redes, routing, NAT, DNS e análise de tráfego, destacando a importância de ferramentas como Wireshark para entender o funcionamento detalhado da rede.

Referências

- Slides das aulas teóricas
- Lab 2 - Guide 2024-2025
- RFC959 - https://www.rfc-editor.org/rfc/rfc959

Anexos

Anexo 1 – Código da aplicação de download

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

#include <string.h>
```

```
#define BUF_SIZE 512

int getIP(char* hostname, char* ip) {
    struct hostent *h;

    #define h_addr h_addr_list[0]

    if ((h = gethostbyname(hostname)) == NULL) {
        herror("gethostbyname()");
        exit(-1);
    }
    strncpy(ip, inet_ntoa(*((struct in_addr *) h->h_addr)), 17);
    printf("Host name  : %s\n", h->h_name);
    printf("IP Address : %s\n", ip);
    return 0;
}

int createSocket(char* ip, int port){
    int sockfd;
    struct sockaddr_in server_addr;
    size_t bytes;

    /*server address handling*/
    bzero((char *) &server_addr, sizeof(server_addr));
    server_addr.sin_family = AF_INET;
    server_addr.sin_addr.s_addr = inet_addr(ip);    /*32 bit Internet address network byte
ordered*/
    server_addr.sin_port = htons(port);          /*server TCP port must be network byte
ordered */

    /*open a TCP socket*/
    if ((sockfd = socket(AF_INET, SOCK_STREAM, 0)) < 0) {
        perror("socket()");
        exit(-1);
    }
    /*connect to the server*/
    if (connect(sockfd,
            (struct sockaddr *) &server_addr,
            sizeof(server_addr)) < 0) {
        perror("connect()");
        exit(-1);
    }
    return sockfd;
}

int readResponse(int sockfd, char *buf, int buf_size, const char* expected_codes[], int
num_codes) {
    memset(buf, 0, buf_size);
    int total_bytes = 0, bytes = 0;
    char line[BUF_SIZE] = {0};
    int matched = 0;
    int errorCode = 1;

    while ((bytes = read(sockfd, line, BUF_SIZE - 1)) > 0) {
        line[bytes] = '\0';
```

```
        strcat(buf, line);
        total_bytes += bytes;

        for (int i = 0; i < num_codes; i++) {
            char code_prefix[16];
            snprintf(code_prefix, sizeof(code_prefix), "%s ", expected_codes[i]);
            //printf("CODE PREFIX:%s\n",code_prefix);
            if (strstr(line, code_prefix) != NULL) {
                matched = 1;
                errorCode = 0;
                break;
            }

            snprintf(code_prefix, sizeof(code_prefix), "%s-", expected_codes[i]);
            if (strstr(line, code_prefix) != NULL) {
                matched = 0;
                errorCode = 0;
                break;
            }
        }

        if(errorCode){
          printf("%s",line);
          exit(-1);
        }

        if (matched) {
            break;
        }
        memset(line, 0, BUF_SIZE);
    }
    if (!matched) {
        perror("code()");
        exit(-1);
    }

    if (bytes < 0) {
        perror("read()");
    }

    return total_bytes;
}



int main(int argc, char **argv) {
    char* url = argv[1];
    char user[256] = {0}, pass[256] = {0}, hostname[256] = {0}, path[256] = {0};
    char ip[17] = {0};
    int bytes = 0;
    char buf[BUF_SIZE] = {0};

    if (sscanf(url, "ftp://%255[^:@]:%255[^@]@%255[^/]/%s", user, pass, hostname, path)
== 4) {
    }
```

```
    else {
        if (sscanf(url, "ftp://%255[^/]/%s", hostname, path) == 2) {
            strcpy(user, "anonymous");
            strcpy(pass, "anonymous");
            printf("User: %s\nPass: %s\nHost: %s\nPath: %s\n", user, pass, hostname,
path);
        }
    }

    getIP(hostname, ip);
    int sockfd = createSocket(ip, 21);
    usleep(100000);     //VERIFICAR ESTE SLEEP
    bytes=read(sockfd,buf, BUF_SIZE);
    printf("%s\n",buf);
    memset(buf,0,BUF_SIZE);

    const char *user_expected[] = { "331", "230" };
    sprintf(buf, "USER %s\r\n", user);

    bytes = write(sockfd, buf, strlen(buf));
    if (bytes > 0){
        printf("%s",buf);
        memset(buf, 0, BUF_SIZE);
        readResponse(sockfd, buf, BUF_SIZE, user_expected,2);
        printf("%s",buf);
    }
    else {
        perror("write()");
        exit(-1);
    }

    const char *pass_expected[] = { "230", "202" };
    sprintf(buf, "PASS %s\r\n", pass);
    bytes = write(sockfd, buf, strlen(buf));
    if (bytes > 0){
        printf("%s",buf);
        memset(buf, 0, BUF_SIZE);
        readResponse(sockfd, buf, BUF_SIZE, pass_expected,2);
        printf("%s",buf);
    }
    else {
        perror("write()");
        exit(-1);
    }

    const char *pasv_expected[] = { "227" };
    char pasv[7] = "pasv\r\n";
    int ip1, ip2, ip3, ip4;
    int port1, port2;
    char ipPasv[256];
    int portPasv = 0;

    bytes = write(sockfd, pasv, strlen(pasv));
    if (bytes > 0){
        printf("%s",pasv);
```

```
        memset(buf, 0, BUF_SIZE);
        readResponse(sockfd, buf, BUF_SIZE, pasv_expected,1);
        sscanf(buf, "227 Entering Passive Mode (%d,%d,%d,%d,%d,%d).", &ip1, &ip2,
&ip3, &ip4, &port1, &port2);
        printf("%s",buf);
    }
    else {
        perror("write()");
        exit(-1);
    }

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

    int sockfdpasv = createSocket(ipPasv, portPasv);
    memset(buf, 0, BUF_SIZE);

    const char *retr_expected[] = { "150", "125" };
    sprintf(buf, "retr %s\r\n", path);

    bytes = write(sockfd, buf, strlen(buf));
    int file_sz = 0;
    if (bytes > 0){
        printf("%s",buf);
        memset(buf, 0, BUF_SIZE);
        readResponse(sockfd, buf, BUF_SIZE, retr_expected, 2);
        //sscanf(buf, "150 Opening BINARY mode data connection for %s (%d bytes).",
path ,&file_sz);
        printf("%s",buf);
    }
    else {
        perror("write()");
        exit(-1);
    }
    char file_downloaded[256];
    const char *last_slash = strrchr(path, '/');
    if (last_slash && *(last_slash + 1) != '\0') {
        strcpy(file_downloaded, last_slash + 1);
    } else {
        strcpy(file_downloaded, path);
    }

    FILE *filefd = fopen(file_downloaded,"wb");
    if (!filefd) {
        perror("Error opening file for writing");
        close(sockfdpasv);
        exit(-1);
    }
    int file_bytes =0;
    while ((file_bytes = read(sockfdpasv, buf, BUF_SIZE)) > 0) {
        int written = fwrite(buf, 1, file_bytes, filefd);
        if (written != file_bytes) {
            perror("Error writing to file");
            fclose(filefd);
            close(sockfdpasv);
```

```
            exit(-1);
        }
    }

    if (file_bytes < 0) {
        perror("Error reading from data connection");
    }

    fclose(filefd);

    if (close(sockfdpasv)<0) {
        perror("close()");
        exit(-1);
    }

    const char *transfer_completed[] = { "226"};
    readResponse(sockfd, buf, BUF_SIZE, transfer_completed,1);
    printf("%s",buf);

    const char *quit_expected[] = { "221" };
    sprintf(buf, "QUIT\r\n");
    bytes = write(sockfd, buf, strlen(buf));
    if (bytes > 0){
        printf("%s",buf);
        memset(buf, 0, BUF_SIZE);
        readResponse(sockfd, buf, BUF_SIZE, quit_expected,1);
        printf("%s",buf);
    }

    if (close(sockfd)<0) {
        perror("close()");
        exit(-1);
    }
    return 0;
}
```

Anexo 2 - Comandos de configuração

- Exp1

tux73: ifconfig eth1 up
tux73: ifconfig eth1 172.16.70.1/24

tux74: ifconfig eth1 up
tux74: ifconfig eth1 172.16.70.254/24

- Exp2

tux72: ifconfig eth1 up
tux72: ifconfig eth1 172.16.71.1/24

switch: /interface bridge port remove [find interface=ether2]
switch: /interface bridge port remove [find interface=ether3]
switch: /interface bridge port remove [find interface=ether4]

switch: /interface bridge add name=bridge70
switch: /interface bridge add name=bridge71

switch: /interface bridge port add bridge=bridge70 interface=ether3
switch: /interface bridge port add bridge=bridge70 interface=ether4
switch: /interface bridge port add bridge=bridge71 interface=ether2

- Exp3

tux74: ifconfig eth2 up
tux74: ifconfig eth2 172.16.71.253/24

switch: /interface bridge port remove [find interface=ether8]
switch: /interface bridge port add bridge=bridge71 interface=ether8

tux74: sysctl net.ipv4.ip_forward=1
tux74: sysctl net.ipv4.icmp_echo_ignore_broadcasts=0

tux72: route add -net 172.16.70.0/24 gw 172.16.71.253
tux73: route add -net 172.16.71.0/24 gw 172.16.70.254

- Exp4

switch: /interface bridge port remove [find interface=ether7]
switch: /interface bridge port add bridge=bridge71 interface=ether7

router: /ip address add address=172.16.1.71/24 interface=ether1
router: /ip address add address=172.16.71.254/24 interface=ether2
router: /ip route add dst-address=172.16.70.0/24 gateway=172.16.71.253

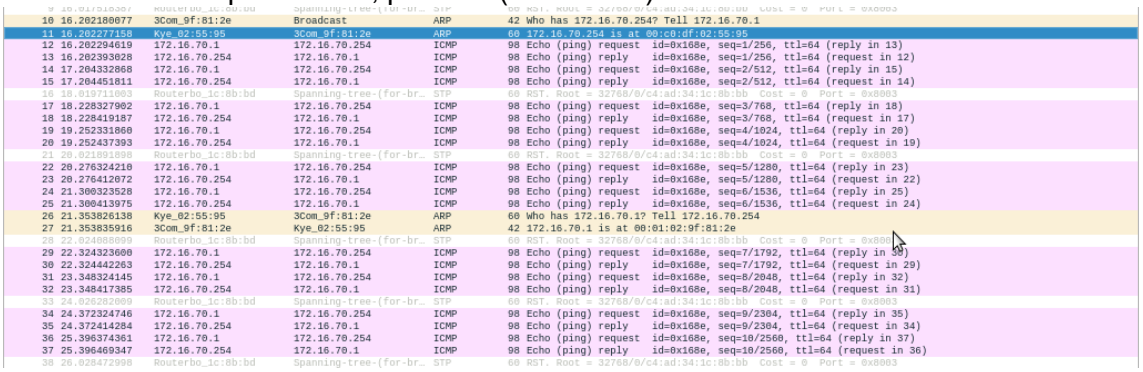
tux73: route add -net 172.16.1.0/24 gw 172.16.70.254
tux74: route add -net 172.16.1.0/24 gw 172.16.71.254
tux72: route add -net 172.16.1.0/24 gw 172.16.71.254

- Exp 6

tux73: gcc clientTCP.c -o download
tux73: ./download ftp://rcom:rcom@ftp.netlab.fe.up.pt/README

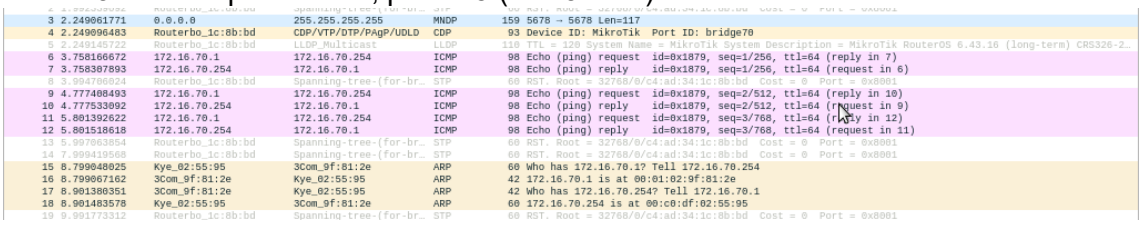
Anexo 3 - Logs capturados

Anexo 3.1.1 – Experiência 1, passo 8 (tux73.eth1)



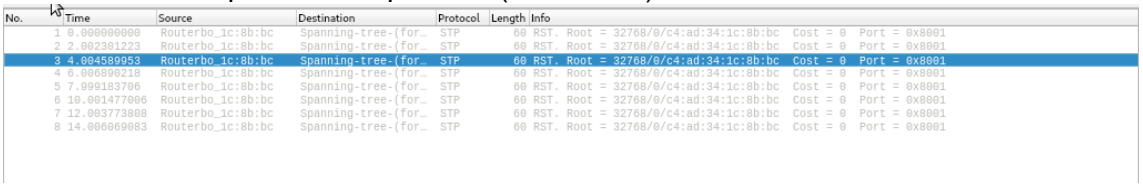
10	16.202180077	3Com_9f:81:2e	Broadcast	ARP	42 Who has 172.16.70.254? Tell 172.16.70.1
11	16.202277158	Kye_02:55:95	3Com_9f:81:2e	ARP	60 172.16.70.254 is at 00:c0:df:02:55:95
12	16.202294619	172.16.70.1	172.16.70.254	ICMP	98 Echo (ping) request id=0x168e, seq=1/256, ttl=64 (reply in 13)
13	16.202393028	172.16.70.254	172.16.70.1	ICMP	98 Echo (ping) reply id=0x168e, seq=1/256, ttl=64 (request in 12)
14	17.204332868	172.16.70.1	172.16.70.254	ICMP	98 Echo (ping) request id=0x168e, seq=2/512, ttl=64 (reply in 15)
15	17.204451811	172.16.70.254	172.16.70.1	ICMP	98 Echo (ping) reply id=0x168e, seq=2/512, ttl=64 (request in 14)
16	18.019711003	Routerbo_1c:8b:bd	Spanning-tree-(for-br…	STP	60 RST. Root = 32768/0/c4:ad:34:1c:8b:bb Cost = 0 Port = 0x8003
17	18.228327902	172.16.70.1	172.16.70.254	ICMP	98 Echo (ping) request id=0x168e, seq=3/768, ttl=64 (reply in 18)
18	18.228419187	172.16.70.254	172.16.70.1	ICMP	98 Echo (ping) reply id=0x168e, seq=3/768, ttl=64 (request in 17)
19	19.252331860	172.16.70.1	172.16.70.254	ICMP	98 Echo (ping) request id=0x168e, seq=4/1024, ttl=64 (reply in 20)
20	19.252437393	172.16.70.254	172.16.70.1	ICMP	98 Echo (ping) reply id=0x168e, seq=4/1024, ttl=64 (request in 19)
21	20.021891098	Routerbo_1c:8b:bd	Spanning-tree-(for-br…	STP	60 RST. Root = 32768/0/c4:ad:34:1c:8b:bb Cost = 0 Port = 0x8003
22	20.276324210	172.16.70.1	172.16.70.254	ICMP	98 Echo (ping) request id=0x168e, seq=5/1280, ttl=64 (reply in 23)
23	20.276412072	172.16.70.254	172.16.70.1	ICMP	98 Echo (ping) reply id=0x168e, seq=5/1280, ttl=64 (request in 22)
24	21.300323528	172.16.70.1	172.16.70.254	ICMP	98 Echo (ping) request id=0x168e, seq=6/1536, ttl=64 (reply in 25)
25	21.300413975	172.16.70.254	172.16.70.1	ICMP	98 Echo (ping) reply id=0x168e, seq=6/1536, ttl=64 (request in 24)
26	21.353826138	Kye_02:55:95	3Com_9f:81:2e	ARP	60 Who has 172.16.70.1? Tell 172.16.70.254
27	21.353835916	3Com_9f:81:2e	Kye_02:55:95	ARP	42 172.16.70.1 is at 00:01:02:9f:81:2e
28	22.024888069	Routerbo_1c:8b:bd	Spanning-tree-(for-br…	STP	60 RST. Root = 32768/0/c4:ad:34:1c:8b:bb Cost = 0 Port = 0x8003
29	22.324323600	172.16.70.1	172.16.70.254	ICMP	98 Echo (ping) request id=0x168e, seq=7/1792, ttl=64 (reply in 30)
30	22.324442263	172.16.70.254	172.16.70.1	ICMP	98 Echo (ping) reply id=0x168e, seq=7/1792, ttl=64 (request in 29)
31	23.348324145	172.16.70.1	172.16.70.254	ICMP	98 Echo (ping) request id=0x168e, seq=8/2048, ttl=64 (reply in 32)
32	23.348417385	172.16.70.254	172.16.70.1	ICMP	98 Echo (ping) reply id=0x168e, seq=8/2048, ttl=64 (request in 31)
33	24.026282099	Routerbo_1c:8b:bd	Spanning-tree-(for-br…	STP	60 RST. Root = 32768/0/c4:ad:34:1c:8b:bb Cost = 0 Port = 0x8003
34	24.372324746	172.16.70.1	172.16.70.254	ICMP	98 Echo (ping) request id=0x168e, seq=9/2304, ttl=64 (reply in 35)
35	24.372414284	172.16.70.254	172.16.70.1	ICMP	98 Echo (ping) reply id=0x168e, seq=9/2304, ttl=64 (request in 34)
36	25.396374361	172.16.70.1	172.16.70.254	ICMP	98 Echo (ping) request id=0x168e, seq=10/2560, ttl=64 (reply in 37)
37	25.396469347	172.16.70.254	172.16.70.1	ICMP	98 Echo (ping) reply id=0x168e, seq=10/2560, ttl=64 (request in 36)
38	26.028472998	Routerbo_1c:8b:bd	Spanning-tree-(for-br…	STP	60 RST. Root = 32768/0/c4:ad:34:1c:8b:bb Cost = 0 Port = 0x8003

Anexo 3.2.1 – Experiência 2, passo 5 (tux73.eth1)



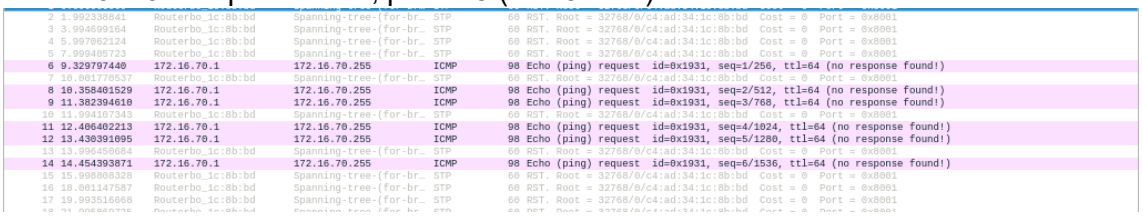
3	2.249061771	0.0.0.0	255.255.255.255	MNDP	159 5678 → 5678 Len=117
4	2.249096483	Routerbo_1c:8b:bd	CDP/VTP/DTP/PAgP/UDLD	CDP	93 Device ID: MikroTik Port ID: bridge70
5	2.249145722	Routerbo_1c:8b:bd	LLDP_Multicast	LLDP	110 TTL = 120 System Name = MikroTik System Description = MikroTik RouterOS 6.43.16 (long-term) CRS326-2…
6	3.758166672	172.16.70.1	172.16.70.254	ICMP	98 Echo (ping) request id=0x1879, seq=1/256, ttl=64 (reply in 7)
7	3.758307893	172.16.70.254	172.16.70.1	ICMP	98 Echo (ping) reply id=0x1879, seq=1/256, ttl=64 (request in 6)
8	3.904786024	Routerbo_1c:8b:bd	Spanning-tree-(for-br…	STP	60 RST. Root = 32768/0/c4:ad:34:1c:8b:bd Cost = 0 Port = 0x8001
9	4.777408493	172.16.70.1	172.16.70.254	ICMP	98 Echo (ping) request id=0x1879, seq=2/512, ttl=64 (reply in 10)
10	4.777533092	172.16.70.254	172.16.70.1	ICMP	98 Echo (ping) reply id=0x1879, seq=2/512, ttl=64 (request in 9)
11	5.801392622	172.16.70.1	172.16.70.254	ICMP	98 Echo (ping) request id=0x1879, seq=3/768, ttl=64 (reply in 12)
12	5.801518618	172.16.70.254	172.16.70.1	ICMP	98 Echo (ping) reply id=0x1879, seq=3/768, ttl=64 (request in 11)
13	5.997063854	Routerbo_1c:8b:bd	Spanning-tree-(for-br…	STP	60 RST. Root = 32768/0/c4:ad:34:1c:8b:bd Cost = 0 Port = 0x8001
14	7.999410568	Routerbo_1c:8b:bd	Spanning-tree-(for-br…	STP	60 RST. Root = 32768/0/c4:ad:34:1c:8b:bd Cost = 0 Port = 0x8001
15	8.799048025	Kye_02:55:95	3Com_9f:81:2e	ARP	60 Who has 172.16.70.1? Tell 172.16.70.254
16	8.799067162	3Com_9f:81:2e	Kye_02:55:95	ARP	42 172.16.70.1 is at 00:01:02:9f:81:2e
17	8.901380351	3Com_9f:81:2e	Kye_02:55:95	ARP	42 Who has 172.16.70.254? Tell 172.16.70.1
18	8.901483578	Kye_02:55:95	3Com_9f:81:2e	ARP	60 172.16.70.254 is at 00:c0:df:02:55:95
19	9.991773312	Routerbo_1c:8b:bd	Spanning-tree-(for-br…	STP	60 RST. Root = 32768/0/c4:ad:34:1c:8b:bd Cost = 0 Port = 0x8001

Anexo 3.2.2 – Experiência 2, passo 8 (tux72.eth1)



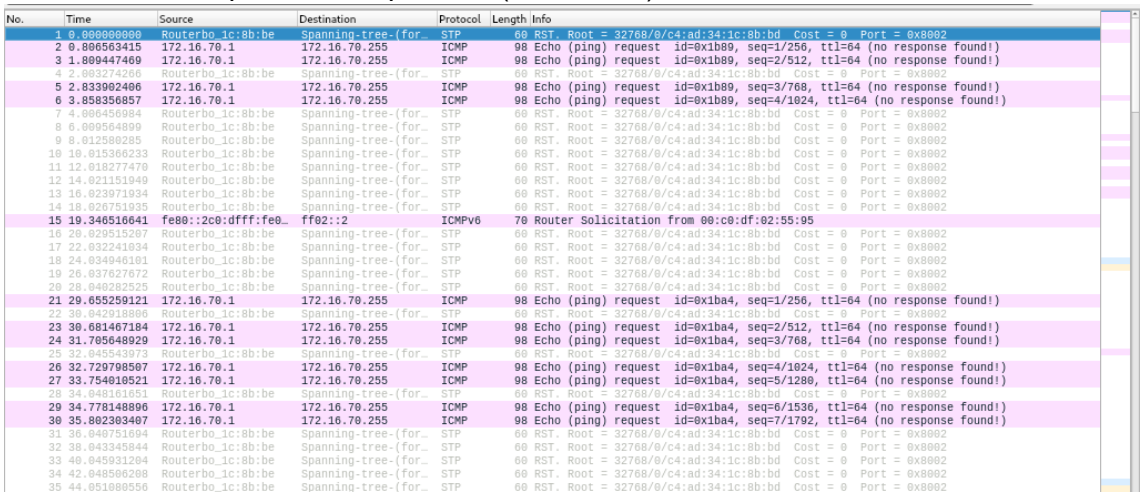
No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	Routerbo_1c:8b:bc	Spanning-tree-(for…	STP	60	RST. Root = 32768/0/c4:ad:34:1c:8b:bc Cost = 0 Port = 0x8001
2	2.002301223	Routerbo_1c:8b:bc	Spanning-tree-(for…	STP	60	RST. Root = 32768/0/c4:ad:34:1c:8b:bc Cost = 0 Port = 0x8001
3	4.004589953	Routerbo_1c:8b:bc	Spanning-tree-(for…	STP	60	RST. Root = 32768/0/c4:ad:34:1c:8b:bc Cost = 0 Port = 0x8001
4	6.006890218	Routerbo_1c:8b:bc	Spanning-tree-(for…	STP	60	RST. Root = 32768/0/c4:ad:34:1c:8b:bc Cost = 0 Port = 0x8001
5	7.999183706	Routerbo_1c:8b:bc	Spanning-tree-(for…	STP	60	RST. Root = 32768/0/c4:ad:34:1c:8b:bc Cost = 0 Port = 0x8001
6	10.001477006	Routerbo_1c:8b:bc	Spanning-tree-(for…	STP	60	RST. Root = 32768/0/c4:ad:34:1c:8b:bc Cost = 0 Port = 0x8001
7	12.003773808	Routerbo_1c:8b:bc	Spanning-tree-(for…	STP	60	RST. Root = 32768/0/c4:ad:34:1c:8b:bc Cost = 0 Port = 0x8001
8	14.006069083	Routerbo_1c:8b:bc	Spanning-tree-(for…	STP	60	RST. Root = 32768/0/c4:ad:34:1c:8b:bc Cost = 0 Port = 0x8001

Anexo 3.2.3 – Experiência 2, passo 8 (tux73.eth1)



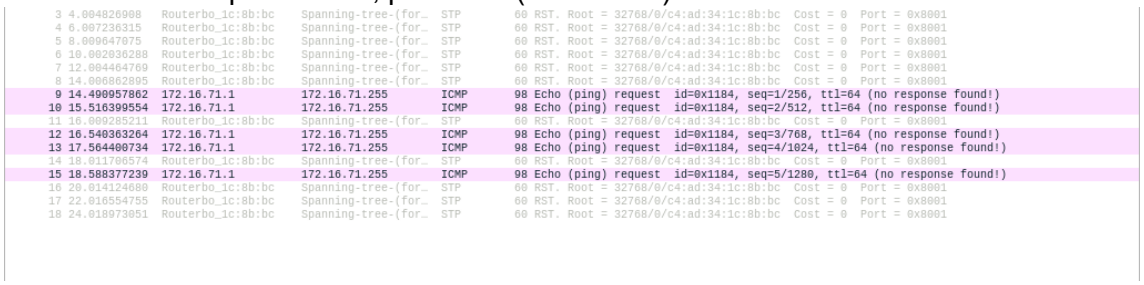
2	1.992338841	Routerbo_1c:8b:bd	Spanning-tree-(for-br…	STP	60 RST. Root = 32768/0/c4:ad:34:1c:8b:bd Cost = 0 Port = 0x8001
3	3.994699164	Routerbo_1c:8b:bd	Spanning-tree-(for-br…	STP	60 RST. Root = 32768/0/c4:ad:34:1c:8b:bd Cost = 0 Port = 0x8001
4	5.997062124	Routerbo_1c:8b:bd	Spanning-tree-(for-br…	STP	60 RST. Root = 32768/0/c4:ad:34:1c:8b:bd Cost = 0 Port = 0x8001
5	7.999405723	Routerbo_1c:8b:bd	Spanning-tree-(for-br…	STP	60 RST. Root = 32768/0/c4:ad:34:1c:8b:bd Cost = 0 Port = 0x8001
6	9.329797440	172.16.70.1	172.16.70.255	ICMP	98 Echo (ping) request id=0x1931, seq=1/256, ttl=64 (no response found!)
7	10.001770537	Routerbo_1c:8b:bd	Spanning-tree-(for-br…	STP	60 RST. Root = 32768/0/c4:ad:34:1c:8b:bd Cost = 0 Port = 0x8001
8	10.358401529	172.16.70.1	172.16.70.255	ICMP	98 Echo (ping) request id=0x1931, seq=2/512, ttl=64 (no response found!)
9	11.382394610	172.16.70.1	172.16.70.255	ICMP	98 Echo (ping) request id=0x1931, seq=3/768, ttl=64 (no response found!)
10	11.994107343	Routerbo_1c:8b:bd	Spanning-tree-(for-br…	STP	60 RST. Root = 32768/0/c4:ad:34:1c:8b:bd Cost = 0 Port = 0x8001
11	12.406402213	172.16.70.1	172.16.70.255	ICMP	98 Echo (ping) request id=0x1931, seq=4/1024, ttl=64 (no response found!)
12	13.430391095	172.16.70.1	172.16.70.255	ICMP	98 Echo (ping) request id=0x1931, seq=5/1280, ttl=64 (no response found!)
13	13.996450684	Routerbo_1c:8b:bd	Spanning-tree-(for-br…	STP	60 RST. Root = 32768/0/c4:ad:34:1c:8b:bd Cost = 0 Port = 0x8001
14	14.454393871	172.16.70.1	172.16.70.255	ICMP	98 Echo (ping) request id=0x1931, seq=6/1536, ttl=64 (no response found!)
15	15.998808328	Routerbo_1c:8b:bd	Spanning-tree-(for-br…	STP	60 RST. Root = 32768/0/c4:ad:34:1c:8b:bd Cost = 0 Port = 0x8001
16	18.001147587	Routerbo_1c:8b:bd	Spanning-tree-(for-br…	STP	60 RST. Root = 32768/0/c4:ad:34:1c:8b:bd Cost = 0 Port = 0x8001
17	19.993516668	Routerbo_1c:8b:bd	Spanning-tree-(for-br…	STP	60 RST. Root = 32768/0/c4:ad:34:1c:8b:bd Cost = 0 Port = 0x8001

Anexo 3.2.4 – Experiência 2, passo 8 (tux74.eth1)



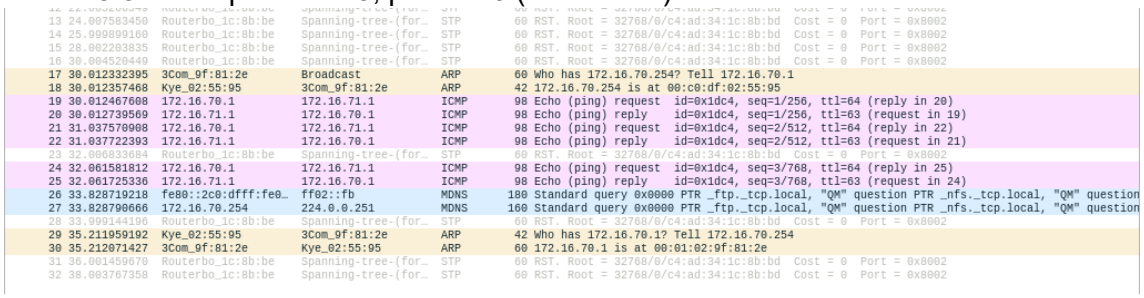
No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	Routerbo_1c:8b:be	Spanning-tree-(for…	STP	60	RST. Root = 32768/0/c4:ad:34:1c:8b:bd Cost = 0 Port = 0x8002
2	0.806563415	172.16.70.1	172.16.70.255	ICMP	98	Echo (ping) request id=0x1b89, seq=1/256, ttl=64 (no response found!)
3	1.809447469	172.16.70.1	172.16.70.255	ICMP	98	Echo (ping) request id=0x1b89, seq=2/512, ttl=64 (no response found!)
4	2.833274206	Routerbo_1c:8b:be	Spanning-tree-(for…	STP	60	RST. Root = 32768/0/c4:ad:34:1c:8b:bd Cost = 0 Port = 0x8002
5	2.833902406	172.16.70.1	172.16.70.255	ICMP	98	Echo (ping) request id=0x1b89, seq=3/768, ttl=64 (no response found!)
6	3.858356857	172.16.70.1	172.16.70.255	ICMP	98	Echo (ping) request id=0x1b89, seq=4/1024, ttl=64 (no response found!)
7	4.006456984	Routerbo_1c:8b:be	Spanning-tree-(for…	STP	60	RST. Root = 32768/0/c4:ad:34:1c:8b:bd Cost = 0 Port = 0x8002
8	6.009564899	Routerbo_1c:8b:be	Spanning-tree-(for…	STP	60	RST. Root = 32768/0/c4:ad:34:1c:8b:bd Cost = 0 Port = 0x8002
9	8.012580285	Routerbo_1c:8b:be	Spanning-tree-(for…	STP	60	RST. Root = 32768/0/c4:ad:34:1c:8b:bd Cost = 0 Port = 0x8002
10	10.015366233	Routerbo_1c:8b:be	Spanning-tree-(for…	STP	60	RST. Root = 32768/0/c4:ad:34:1c:8b:bd Cost = 0 Port = 0x8002
11	12.018277470	Routerbo_1c:8b:be	Spanning-tree-(for…	STP	60	RST. Root = 32768/0/c4:ad:34:1c:8b:bd Cost = 0 Port = 0x8002
12	14.021151949	Routerbo_1c:8b:be	Spanning-tree-(for…	STP	60	RST. Root = 32768/0/c4:ad:34:1c:8b:bd Cost = 0 Port = 0x8002
13	16.023971934	Routerbo_1c:8b:be	Spanning-tree-(for…	STP	60	RST. Root = 32768/0/c4:ad:34:1c:8b:bd Cost = 0 Port = 0x8002
14	18.026751935	Routerbo_1c:8b:be	Spanning-tree-(for…	STP	60	RST. Root = 32768/0/c4:ad:34:1c:8b:bd Cost = 0 Port = 0x8002
15	19.346516641	fe80::2c0:dfff:fe0…	ff02::2	ICMPv6	70	Router Solicitation from 00:c0:df:02:55:95
16	20.029515207	Routerbo_1c:8b:be	Spanning-tree-(for…	STP	60	RST. Root = 32768/0/c4:ad:34:1c:8b:bd Cost = 0 Port = 0x8002
17	22.032241034	Routerbo_1c:8b:be	Spanning-tree-(for…	STP	60	RST. Root = 32768/0/c4:ad:34:1c:8b:bd Cost = 0 Port = 0x8002
18	24.034946101	Routerbo_1c:8b:be	Spanning-tree-(for…	STP	60	RST. Root = 32768/0/c4:ad:34:1c:8b:bd Cost = 0 Port = 0x8002
19	26.037627672	Routerbo_1c:8b:be	Spanning-tree-(for…	STP	60	RST. Root = 32768/0/c4:ad:34:1c:8b:bd Cost = 0 Port = 0x8002
20	28.040282525	Routerbo_1c:8b:be	Spanning-tree-(for…	STP	60	RST. Root = 32768/0/c4:ad:34:1c:8b:bd Cost = 0 Port = 0x8002
21	29.655259121	172.16.70.1	172.16.70.255	ICMP	98	Echo (ping) request id=0x1ba4, seq=1/256, ttl=64 (no response found!)
22	30.042918806	Routerbo_1c:8b:be	Spanning-tree-(for…	STP	60	RST. Root = 32768/0/c4:ad:34:1c:8b:bd Cost = 0 Port = 0x8002
23	30.681467184	172.16.70.1	172.16.70.255	ICMP	98	Echo (ping) request id=0x1ba4, seq=2/512, ttl=64 (no response found!)
24	31.705648929	172.16.70.1	172.16.70.255	ICMP	98	Echo (ping) request id=0x1ba4, seq=3/768, ttl=64 (no response found!)
25	32.045543073	Routerbo_1c:8b:be	Spanning-tree-(for…	STP	60	RST. Root = 32768/0/c4:ad:34:1c:8b:bd Cost = 0 Port = 0x8002
26	32.729798507	172.16.70.1	172.16.70.255	ICMP	98	Echo (ping) request id=0x1ba4, seq=4/1024, ttl=64 (no response found!)
27	33.754010521	172.16.70.1	172.16.70.255	ICMP	98	Echo (ping) request id=0x1ba4, seq=5/1280, ttl=64 (no response found!)
28	34.048161051	Routerbo_1c:8b:be	Spanning-tree-(for…	STP	60	RST. Root = 32768/0/c4:ad:34:1c:8b:bd Cost = 0 Port = 0x8002
29	34.778148896	172.16.70.1	172.16.70.255	ICMP	98	Echo (ping) request id=0x1ba4, seq=6/1536, ttl=64 (no response found!)
30	35.802303407	172.16.70.1	172.16.70.255	ICMP	98	Echo (ping) request id=0x1ba4, seq=7/1792, ttl=64 (no response found!)
31	36.040751694	Routerbo_1c:8b:be	Spanning-tree-(for…	STP	60	RST. Root = 32768/0/c4:ad:34:1c:8b:bd Cost = 0 Port = 0x8002
32	38.043345844	Routerbo_1c:8b:be	Spanning-tree-(for…	STP	60	RST. Root = 32768/0/c4:ad:34:1c:8b:bd Cost = 0 Port = 0x8002
33	40.045931204	Routerbo_1c:8b:be	Spanning-tree-(for…	STP	60	RST. Root = 32768/0/c4:ad:34:1c:8b:bd Cost = 0 Port = 0x8002
34	42.048506208	Routerbo_1c:8b:be	Spanning-tree-(for…	STP	60	RST. Root = 32768/0/c4:ad:34:1c:8b:bd Cost = 0 Port = 0x8002
35	44.051080556	Routerbo_1c:8b:be	Spanning-tree-(for…	STP	60	RST. Root = 32768/0/c4:ad:34:1c:8b:bd Cost = 0 Port = 0x8002

Anexo 3.2.5 – Experiência 2, passo 10 (tux72.eth1)



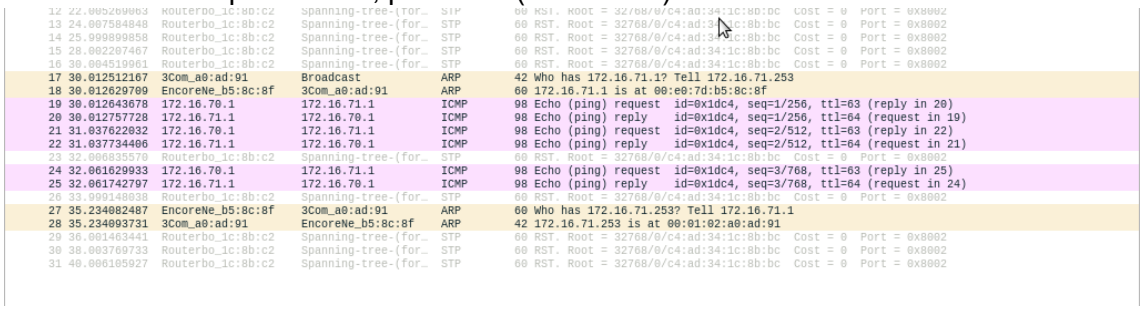
3	4.004826908	Routerbo_1c:8b:bc	Spanning-tree-(for…	STP	60	RST. Root = 32768/0/c4:ad:34:1c:8b:bc Cost = 0 Port = 0x8001
4	6.007236315	Routerbo_1c:8b:bc	Spanning-tree-(for…	STP	60	RST. Root = 32768/0/c4:ad:34:1c:8b:bc Cost = 0 Port = 0x8001
5	8.009647075	Routerbo_1c:8b:bc	Spanning-tree-(for…	STP	60	RST. Root = 32768/0/c4:ad:34:1c:8b:bc Cost = 0 Port = 0x8001
6	10.002036288	Routerbo_1c:8b:bc	Spanning-tree-(for…	STP	60	RST. Root = 32768/0/c4:ad:34:1c:8b:bc Cost = 0 Port = 0x8001
7	12.004464769	Routerbo_1c:8b:bc	Spanning-tree-(for…	STP	60	RST. Root = 32768/0/c4:ad:34:1c:8b:bc Cost = 0 Port = 0x8001
8	14.006862895	Routerbo_1c:8b:bc	Spanning-tree-(for…	STP	60	RST. Root = 32768/0/c4:ad:34:1c:8b:bc Cost = 0 Port = 0x8001
9	14.490957862	172.16.71.1	172.16.71.255	ICMP	98	Echo (ping) request id=0x1184, seq=1/256, ttl=64 (no response found!)
10	15.516399554	172.16.71.1	172.16.71.255	ICMP	98	Echo (ping) request id=0x1184, seq=2/512, ttl=64 (no response found!)
11	16.009285211	Routerbo_1c:8b:bc	Spanning-tree-(for…	STP	60	RST. Root = 32768/0/c4:ad:34:1c:8b:bc Cost = 0 Port = 0x8001
12	16.540363264	172.16.71.1	172.16.71.255	ICMP	98	Echo (ping) request id=0x1184, seq=3/768, ttl=64 (no response found!)
13	17.564400734	172.16.71.1	172.16.71.255	ICMP	98	Echo (ping) request id=0x1184, seq=4/1024, ttl=64 (no response found!)
14	18.011706574	Routerbo_1c:8b:bc	Spanning-tree-(for…	STP	60	RST. Root = 32768/0/c4:ad:34:1c:8b:bc Cost = 0 Port = 0x8001
15	18.588377239	172.16.71.1	172.16.71.255	ICMP	98	Echo (ping) request id=0x1184, seq=5/1280, ttl=64 (no response found!)
16	20.014124680	Routerbo_1c:8b:bc	Spanning-tree-(for…	STP	60	RST. Root = 32768/0/c4:ad:34:1c:8b:bc Cost = 0 Port = 0x8001
17	22.016554755	Routerbo_1c:8b:bc	Spanning-tree-(for…	STP	60	RST. Root = 32768/0/c4:ad:34:1c:8b:bc Cost = 0 Port = 0x8001
18	24.018973051	Routerbo_1c:8b:bc	Spanning-tree-(for…	STP	60	RST. Root = 32768/0/c4:ad:34:1c:8b:bc Cost = 0 Port = 0x8001

Anexo 3.3.1 – Experiência 3, passo 10 (tux74.eth1)



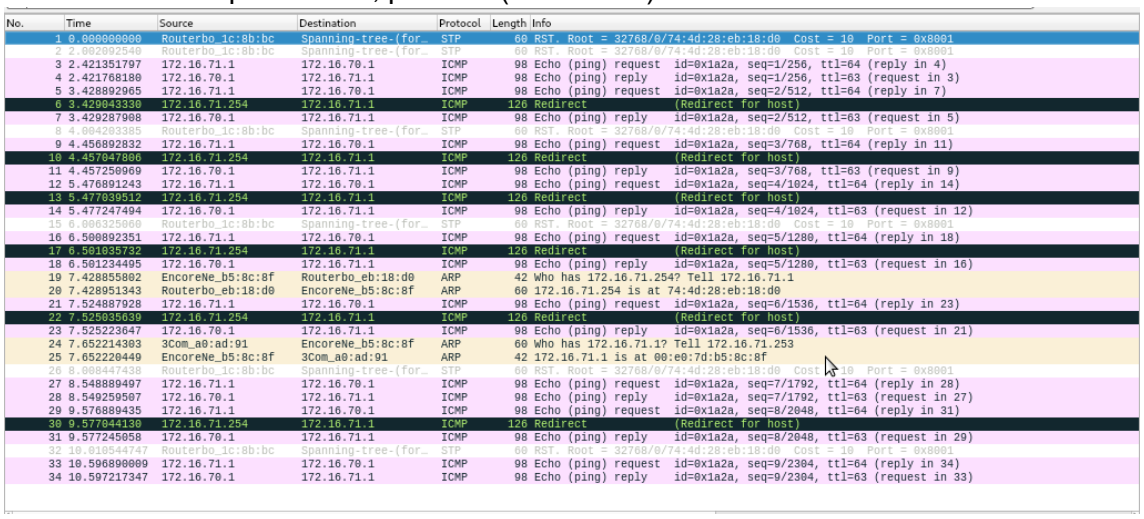
13	24.007583450	Routerbo_1c:8b:be	Spanning-tree-(for…	STP	60	RST. Root = 32768/0/c4:ad:34:1c:8b:bd Cost = 0 Port = 0x8002
14	25.999899160	Routerbo_1c:8b:be	Spanning-tree-(for…	STP	60	RST. Root = 32768/0/c4:ad:34:1c:8b:bd Cost = 0 Port = 0x8002
15	28.002203835	Routerbo_1c:8b:be	Spanning-tree-(for…	STP	60	RST. Root = 32768/0/c4:ad:34:1c:8b:bd Cost = 0 Port = 0x8002
16	30.004520449	Routerbo_1c:8b:be	Spanning-tree-(for…	STP	60	RST. Root = 32768/0/c4:ad:34:1c:8b:bd Cost = 0 Port = 0x8002
17	30.012332395	3Com_9f:81:2e	Broadcast	ARP	60	Who has 172.16.70.254? Tell 172.16.70.1
18	30.012357468	Kye_02:55:95	3Com_9f:81:2e	ARP	42	172.16.70.254 is at 00:c0:df:02:55:95
19	30.012467608	172.16.70.1	172.16.71.1	ICMP	98	Echo (ping) request id=0x1dc4, seq=1/256, ttl=64 (reply in 20)
20	30.012739569	172.16.71.1	172.16.70.1	ICMP	98	Echo (ping) reply id=0x1dc4, seq=1/256, ttl=63 (request in 19)
21	31.037570908	172.16.70.1	172.16.71.1	ICMP	98	Echo (ping) request id=0x1dc4, seq=2/512, ttl=64 (reply in 22)
22	31.037722393	172.16.71.1	172.16.70.1	ICMP	98	Echo (ping) reply id=0x1dc4, seq=2/512, ttl=63 (request in 21)
23	32.006833684	Routerbo_1c:8b:be	Spanning-tree-(for…	STP	60	RST. Root = 32768/0/c4:ad:34:1c:8b:bd Cost = 0 Port = 0x8002
24	32.061581812	172.16.70.1	172.16.71.1	ICMP	98	Echo (ping) request id=0x1dc4, seq=3/768, ttl=64 (reply in 25)
25	32.061725336	172.16.71.1	172.16.70.1	ICMP	98	Echo (ping) reply id=0x1dc4, seq=3/768, ttl=63 (request in 24)
26	33.828719218	fe80::2c0:dfff:fe0…	ff02::fb	MDNS	180	Standard query 0x0000 PTR _ftp._tcp.local, "QM" question PTR _nfs._tcp.local, "QM" question
27	33.828790666	172.16.70.254	224.0.0.251	MDNS	160	Standard query 0x0000 PTR _ftp._tcp.local, "QM" question PTR _nfs._tcp.local, "QM" question
28	33.999144196	Routerbo_1c:8b:be	Spanning-tree-(for…	STP	60	RST. Root = 32768/0/c4:ad:34:1c:8b:bd Cost = 0 Port = 0x8002
29	35.211959192	Kye_02:55:95	3Com_9f:81:2e	ARP	42	Who has 172.16.70.1? Tell 172.16.70.254
30	35.212071427	3Com_9f:81:2e	Kye_02:55:95	ARP	60	172.16.70.1 is at 00:01:02:9f:81:2e
31	36.001459070	Routerbo_1c:8b:be	Spanning-tree-(for…	STP	60	RST. Root = 32768/0/c4:ad:34:1c:8b:bd Cost = 0 Port = 0x8002
32	38.003767358	Routerbo_1c:8b:be	Spanning-tree-(for…	STP	60	RST. Root = 32768/0/c4:ad:34:1c:8b:bd Cost = 0 Port = 0x8002

Anexo 3.3.2 – Experiência 3, passo 10 (tux74.eth2)



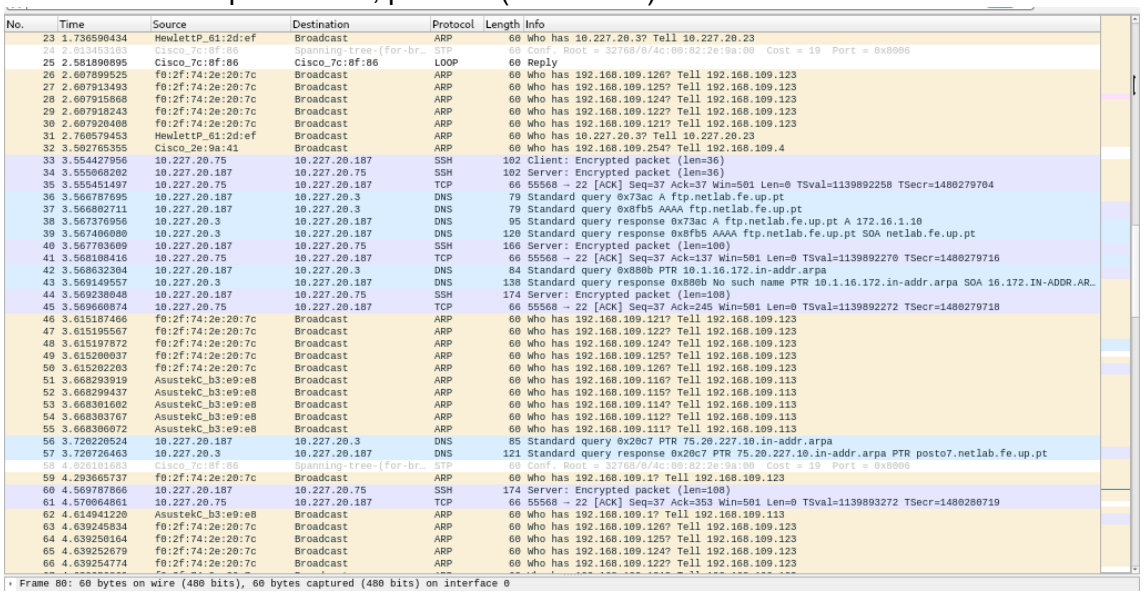
12	22.005269005	Routerbo_1c:8b:c2	Spanning-tree-(for…	STP	60	RST. Root = 32768/0/c4:ad:34:1c:8b:bc Cost = 0 Port = 0x8002
13	24.007584848	Routerbo_1c:8b:c2	Spanning-tree-(for…	STP	60	RST. Root = 32768/0/c4:ad:34:1c:8b:bc Cost = 0 Port = 0x8002
14	25.999899858	Routerbo_1c:8b:c2	Spanning-tree-(for…	STP	60	RST. Root = 32768/0/c4:ad:34:1c:8b:bc Cost = 0 Port = 0x8002
15	28.002207467	Routerbo_1c:8b:c2	Spanning-tree-(for…	STP	60	RST. Root = 32768/0/c4:ad:34:1c:8b:bc Cost = 0 Port = 0x8002
16	30.004519961	Routerbo_1c:8b:c2	Spanning-tree-(for…	STP	60	RST. Root = 32768/0/c4:ad:34:1c:8b:bc Cost = 0 Port = 0x8002
17	30.012512167	3Com_a0:ad:91	Broadcast	ARP	42	Who has 172.16.71.1? Tell 172.16.71.253
18	30.012629709	EncoreNe_b5:8c:8f	3Com_a0:ad:91	ARP	60	172.16.71.1 is at 00:e0:7d:b5:8c:8f
19	30.012643678	172.16.70.1	172.16.71.1	ICMP	98	Echo (ping) request id=0x1dc4, seq=1/256, ttl=63 (reply in 20)
20	30.012757728	172.16.71.1	172.16.70.1	ICMP	98	Echo (ping) reply id=0x1dc4, seq=1/256, ttl=64 (request in 19)
21	31.037622032	172.16.70.1	172.16.71.1	ICMP	98	Echo (ping) request id=0x1dc4, seq=2/512, ttl=63 (reply in 22)
22	31.037734406	172.16.71.1	172.16.70.1	ICMP	98	Echo (ping) reply id=0x1dc4, seq=2/512, ttl=64 (request in 21)
23	32.006835570	Routerbo_1c:8b:c2	Spanning-tree-(for…	STP	60	RST. Root = 32768/0/c4:ad:34:1c:8b:bc Cost = 0 Port = 0x8002
24	32.061629933	172.16.70.1	172.16.71.1	ICMP	98	Echo (ping) request id=0x1dc4, seq=3/768, ttl=63 (reply in 25)
25	32.061742797	172.16.71.1	172.16.70.1	ICMP	98	Echo (ping) reply id=0x1dc4, seq=3/768, ttl=64 (request in 24)
26	33.999148038	Routerbo_1c:8b:c2	Spanning-tree-(for…	STP	60	RST. Root = 32768/0/c4:ad:34:1c:8b:bc Cost = 0 Port = 0x8002
27	35.234082487	EncoreNe_b5:8c:8f	3Com_a0:ad:91	ARP	60	Who has 172.16.71.253? Tell 172.16.71.1
28	35.234093731	3Com_a0:ad:91	EncoreNe_b5:8c:8f	ARP	42	172.16.71.253 is at 00:01:02:a0:ad:91
29	36.001463441	Routerbo_1c:8b:c2	Spanning-tree-(for…	STP	60	RST. Root = 32768/0/c4:ad:34:1c:8b:bc Cost = 0 Port = 0x8002
30	38.003769733	Routerbo_1c:8b:c2	Spanning-tree-(for…	STP	60	RST. Root = 32768/0/c4:ad:34:1c:8b:bc Cost = 0 Port = 0x8002
31	40.006105927	Routerbo_1c:8b:c2	Spanning-tree-(for…	STP	60	RST. Root = 32768/0/c4:ad:34:1c:8b:bc Cost = 0 Port = 0x8002

Anexo 3.4.1 – Experiência 4, passo 4 (tux72.eth1)



No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	Routerbo_1c:8b:bc	Spanning-tree-(for…	STP	60	RST. Root = 32768/0/74:4d:28:eb:18:d0 Cost = 10 Port = 0x8001
2	2.002092540	Routerbo_1c:8b:bc	Spanning-tree-(for…	STP	60	RST. Root = 32768/0/74:4d:28:eb:18:d0 Cost = 10 Port = 0x8001
3	2.421351797	172.16.71.1	172.16.70.1	ICMP	98	Echo (ping) request id=0x1a2a, seq=1/256, ttl=64 (reply in 4)
4	2.421768180	172.16.70.1	172.16.71.1	ICMP	98	Echo (ping) reply id=0x1a2a, seq=1/256, ttl=63 (request in 3)
5	3.428892965	172.16.71.1	172.16.70.1	ICMP	98	Echo (ping) request id=0x1a2a, seq=2/512, ttl=64 (reply in 7)
6	3.429043330	172.16.71.254	172.16.71.1	ICMP	126	Redirect (Redirect for host)
7	3.429287908	172.16.70.1	172.16.71.1	ICMP	98	Echo (ping) reply id=0x1a2a, seq=2/512, ttl=63 (request in 5)
8	4.004263385	Routerbo_1c:8b:bc	Spanning-tree-(for…	STP	60	RST. Root = 32768/0/74:4d:28:eb:18:d0 Cost = 10 Port = 0x8001
9	4.456892832	172.16.71.1	172.16.70.1	ICMP	98	Echo (ping) request id=0x1a2a, seq=3/768, ttl=64 (reply in 11)
10	4.457047806	172.16.71.254	172.16.71.1	ICMP	126	Redirect (Redirect for host)
11	4.457250969	172.16.70.1	172.16.71.1	ICMP	98	Echo (ping) reply id=0x1a2a, seq=3/768, ttl=63 (request in 9)
12	5.476891243	172.16.71.1	172.16.70.1	ICMP	98	Echo (ping) request id=0x1a2a, seq=4/1024, ttl=64 (reply in 14)
13	5.477039512	172.16.71.254	172.16.71.1	ICMP	126	Redirect (Redirect for host)
14	5.477247494	172.16.70.1	172.16.71.1	ICMP	98	Echo (ping) reply id=0x1a2a, seq=4/1024, ttl=63 (request in 12)
15	6.006325060	Routerbo_1c:8b:bc	Spanning-tree-(for…	STP	60	RST. Root = 32768/0/74:4d:28:eb:18:d0 Cost = 10 Port = 0x8001
16	6.500892351	172.16.71.1	172.16.70.1	ICMP	98	Echo (ping) request id=0x1a2a, seq=5/1280, ttl=64 (reply in 18)
17	6.501035732	172.16.71.254	172.16.71.1	ICMP	126	Redirect (Redirect for host)
18	6.501234495	172.16.70.1	172.16.71.1	ICMP	98	Echo (ping) reply id=0x1a2a, seq=5/1280, ttl=63 (request in 16)
19	7.428855802	EncoreNe_b5:8c:8f	Routerbo_eb:18:d0	ARP	42	Who has 172.16.71.254? Tell 172.16.71.1
20	7.428951343	Routerbo_eb:18:d0	EncoreNe_b5:8c:8f	ARP	60	172.16.71.254 is at 74:4d:28:eb:18:d0
21	7.524887928	172.16.71.1	172.16.70.1	ICMP	98	Echo (ping) request id=0x1a2a, seq=6/1536, ttl=64 (reply in 23)
22	7.525035639	172.16.71.254	172.16.71.1	ICMP	126	Redirect (Redirect for host)
23	7.525223647	172.16.70.1	172.16.71.1	ICMP	98	Echo (ping) reply id=0x1a2a, seq=6/1536, ttl=63 (request in 21)
24	7.652214303	3Com_a0:ad:91	EncoreNe_b5:8c:8f	ARP	60	Who has 172.16.71.1? Tell 172.16.71.253
25	7.652220449	EncoreNe_b5:8c:8f	3Com_a0:ad:91	ARP	42	172.16.71.1 is at 00:e0:7d:b5:8c:8f
26	8.008447438	Routerbo_1c:8b:bc	Spanning-tree-(for…	STP	60	RST. Root = 32768/0/74:4d:28:eb:18:d0 Cost = 10 Port = 0x8001
27	8.548889497	172.16.71.1	172.16.70.1	ICMP	98	Echo (ping) request id=0x1a2a, seq=7/1792, ttl=64 (reply in 28)
28	8.549259507	172.16.70.1	172.16.71.1	ICMP	98	Echo (ping) reply id=0x1a2a, seq=7/1792, ttl=63 (request in 27)
29	9.576889435	172.16.71.1	172.16.70.1	ICMP	98	Echo (ping) request id=0x1a2a, seq=8/2048, ttl=64 (reply in 31)
30	9.577044130	172.16.71.254	172.16.71.1	ICMP	126	Redirect (Redirect for host)
31	9.577245058	172.16.70.1	172.16.71.1	ICMP	98	Echo (ping) reply id=0x1a2a, seq=8/2048, ttl=63 (request in 29)
32	10.010544747	Routerbo_1c:8b:bc	Spanning-tree-(for…	STP	60	RST. Root = 32768/0/74:4d:28:eb:18:d0 Cost = 10 Port = 0x8001
33	10.596890009	172.16.71.1	172.16.70.1	ICMP	98	Echo (ping) request id=0x1a2a, seq=9/2304, ttl=64 (reply in 34)
34	10.597217347	172.16.70.1	172.16.71.1	ICMP	98	Echo (ping) reply id=0x1a2a, seq=9/2304, ttl=63 (request in 33)

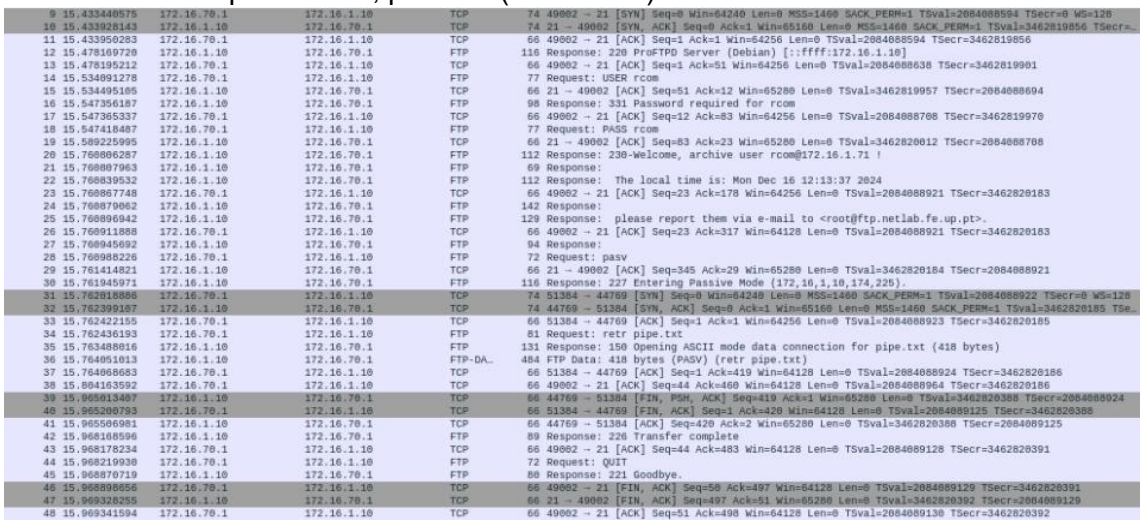
Anexo 3.5.1 – Experiência 5, passo 3 (tux73.eth1)



No.	Time	Source	Destination	Protocol	Length	Info
23	1.736590434	HewlettP_61:2d:ef	Broadcast	ARP	60	Who has 10.227.20.3? Tell 10.227.20.23
24	2.013453103	Cisco_7c:8f:86	Spanning-tree-(for-br…	STP	60	Conf. Root = 32768/0/4c:00:82:2e:9a:00 Cost = 19 Port = 0x8006
25	2.581890895	Cisco_7c:8f:86	Cisco_7c:8f:86	LOOP	60	Reply
26	2.607899525	f0:2f:74:2e:20:7c	Broadcast	ARP	60	Who has 192.168.109.126? Tell 192.168.109.123
27	2.607913493	f0:2f:74:2e:20:7c	Broadcast	ARP	60	Who has 192.168.109.125? Tell 192.168.109.123
28	2.607915868	f0:2f:74:2e:20:7c	Broadcast	ARP	60	Who has 192.168.109.124? Tell 192.168.109.123
29	2.607918243	f0:2f:74:2e:20:7c	Broadcast	ARP	60	Who has 192.168.109.122? Tell 192.168.109.123
30	2.607920408	f0:2f:74:2e:20:7c	Broadcast	ARP	60	Who has 192.168.109.121? Tell 192.168.109.123
31	2.760579453	HewlettP_61:2d:ef	Broadcast	ARP	60	Who has 10.227.20.3? Tell 10.227.20.23
32	3.502765355	Cisco_2e:9a:41	Broadcast	ARP	60	Who has 192.168.109.254? Tell 192.168.109.4
33	3.554427956	10.227.20.75	10.227.20.187	SSH	102	Client: Encrypted packet (len=36)
34	3.555068202	10.227.20.187	10.227.20.75	SSH	102	Server: Encrypted packet (len=36)
35	3.555451497	10.227.20.75	10.227.20.187	TCP	66	55568 → 22 [ACK] Seq=37 Ack=37 Win=501 Len=0 TSval=1139892258 TSecr=1480279704
36	3.566787695	10.227.20.187	10.227.20.3	DNS	79	Standard query 0x73ac A ftp.netlab.fe.up.pt
37	3.566882711	10.227.20.187	10.227.20.3	DNS	79	Standard query 0x8fb5 AAAA ftp.netlab.fe.up.pt
38	3.567376956	10.227.20.3	10.227.20.187	DNS	95	Standard query response 0x73ac A ftp.netlab.fe.up.pt A 172.16.1.10
39	3.567406080	10.227.20.3	10.227.20.187	DNS	120	Standard query response 0x8fb5 AAAA ftp.netlab.fe.up.pt SOA netlab.fe.up.pt
40	3.567703609	10.227.20.187	10.227.20.75	SSH	166	Server: Encrypted packet (len=100)
41	3.568108416	10.227.20.75	10.227.20.187	TCP	66	55568 → 22 [ACK] Seq=37 Ack=137 Win=501 Len=0 TSval=1139892270 TSecr=1480279716
42	3.568632304	10.227.20.187	10.227.20.3	DNS	84	Standard query 0x880b PTR 10.1.16.172.in-addr.arpa
43	3.569149557	10.227.20.3	10.227.20.187	DNS	138	Standard query response 0x880b No such name PTR 10.1.16.172.in-addr.arpa SOA 16.172.IN-ADDR.AR…
44	3.569238048	10.227.20.187	10.227.20.75	SSH	174	Server: Encrypted packet (len=108)
45	3.569660874	10.227.20.75	10.227.20.187	TCP	66	55568 → 22 [ACK] Seq=37 Ack=245 Win=501 Len=0 TSval=1139892272 TSecr=1480279718
46	3.615187466	f0:2f:74:2e:20:7c	Broadcast	ARP	60	Who has 192.168.109.121? Tell 192.168.109.123
47	3.615195567	f0:2f:74:2e:20:7c	Broadcast	ARP	60	Who has 192.168.109.122? Tell 192.168.109.123
48	3.615197872	f0:2f:74:2e:20:7c	Broadcast	ARP	60	Who has 192.168.109.124? Tell 192.168.109.123
49	3.615200037	f0:2f:74:2e:20:7c	Broadcast	ARP	60	Who has 192.168.109.125? Tell 192.168.109.123
50	3.615202203	f0:2f:74:2e:20:7c	Broadcast	ARP	60	Who has 192.168.109.126? Tell 192.168.109.123
51	3.668293919	AsustekC_b3:e9:e8	Broadcast	ARP	60	Who has 192.168.109.116? Tell 192.168.109.113
52	3.668299437	AsustekC_b3:e9:e8	Broadcast	ARP	60	Who has 192.168.109.115? Tell 192.168.109.113
53	3.668301602	AsustekC_b3:e9:e8	Broadcast	ARP	60	Who has 192.168.109.114? Tell 192.168.109.113
54	3.668303767	AsustekC_b3:e9:e8	Broadcast	ARP	60	Who has 192.168.109.112? Tell 192.168.109.113
55	3.668306072	AsustekC_b3:e9:e8	Broadcast	ARP	60	Who has 192.168.109.111? Tell 192.168.109.113
56	3.720220524	10.227.20.187	10.227.20.3	DNS	85	Standard query 0x20c7 PTR 75.20.227.10.in-addr.arpa
57	3.720726463	10.227.20.3	10.227.20.187	DNS	121	Standard query response 0x20c7 PTR 75.20.227.10.in-addr.arpa PTR posto7.netlab.fe.up.pt
58	4.026101683	Cisco_7c:8f:86	Spanning-tree-(for-br…	STP	60	Conf. Root = 32768/0/4c:00:82:2e:9a:00 Cost = 19 Port = 0x8006
59	4.293665737	f0:2f:74:2e:20:7c	Broadcast	ARP	60	Who has 192.168.109.1? Tell 192.168.109.123
60	4.569787866	10.227.20.187	10.227.20.75	SSH	174	Server: Encrypted packet (len=108)
61	4.570064861	10.227.20.75	10.227.20.187	TCP	66	55568 → 22 [ACK] Seq=37 Ack=353 Win=501 Len=0 TSval=1139893272 TSecr=1480280719
62	4.614941220	AsustekC_b3:e9:e8	Broadcast	ARP	60	Who has 192.168.109.1? Tell 192.168.109.113
63	4.639245834	f0:2f:74:2e:20:7c	Broadcast	ARP	60	Who has 192.168.109.126? Tell 192.168.109.123
64	4.639250164	f0:2f:74:2e:20:7c	Broadcast	ARP	60	Who has 192.168.109.125? Tell 192.168.109.123
65	4.639252679	f0:2f:74:2e:20:7c	Broadcast	ARP	60	Who has 192.168.109.124? Tell 192.168.109.123
66	4.639254774	f0:2f:74:2e:20:7c	Broadcast	ARP	60	Who has 192.168.109.122? Tell 192.168.109.123

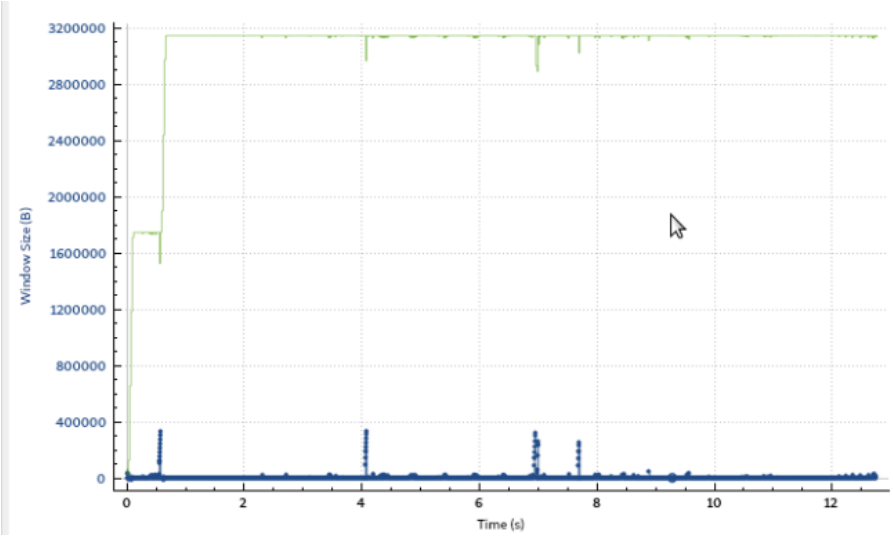
› Frame 80: 60 bytes on wire (480 bits), 60 bytes captured (480 bits) on interface 0

Anexo 3.6.1 – Experiência 6, passo 2 (tux73.eth1)



9	15.433440575	172.16.70.1	172.16.1.10	TCP	74	49002 → 21 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=2084088594 TSecr=0 WS=128
10	15.433928143	172.16.1.10	172.16.70.1	TCP	74	21 → 49002 [SYN, ACK] Seq=0 Ack=1 Win=65160 Len=0 MSS=1460 SACK_PERM=1 TSval=3462819856 TSecr=…
11	15.433950283	172.16.70.1	172.16.1.10	TCP	66	49002 → 21 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=2084088594 TSecr=3462819856
12	15.478169720	172.16.1.10	172.16.70.1	FTP	116	Response: 220 ProFTPD Server (Debian) [::ffff:172.16.1.10]
13	15.478195212	172.16.70.1	172.16.1.10	TCP	66	49002 → 21 [ACK] Seq=1 Ack=51 Win=64256 Len=0 TSval=2084088638 TSecr=3462819901
14	15.534091278	172.16.70.1	172.16.1.10	FTP	77	Request: USER rcom
15	15.534495105	172.16.1.10	172.16.70.1	TCP	66	21 → 49002 [ACK] Seq=51 Ack=12 Win=65280 Len=0 TSval=3462819957 TSecr=2084088694
16	15.547356187	172.16.1.10	172.16.70.1	FTP	98	Response: 331 Password required for rcom
17	15.547365337	172.16.70.1	172.16.1.10	TCP	66	49002 → 21 [ACK] Seq=12 Ack=83 Win=64256 Len=0 TSval=2084088708 TSecr=3462819970
18	15.547418487	172.16.70.1	172.16.1.10	FTP	77	Request: PASS rcom
19	15.589225995	172.16.1.10	172.16.70.1	TCP	66	21 → 49002 [ACK] Seq=83 Ack=23 Win=65280 Len=0 TSval=3462820012 TSecr=2084088708
20	15.760806287	172.16.1.10	172.16.70.1	FTP	112	Response: 230-Welcome, archive user rcom@172.16.1.71 !
21	15.760807963	172.16.1.10	172.16.70.1	FTP	69	Response:
22	15.760839532	172.16.1.10	172.16.70.1	FTP	112	Response: The local time is: Mon Dec 16 12:13:37 2024
23	15.760867748	172.16.70.1	172.16.1.10	TCP	66	49002 → 21 [ACK] Seq=23 Ack=178 Win=64256 Len=0 TSval=2084088921 TSecr=3462820183
24	15.760879062	172.16.1.10	172.16.70.1	FTP	142	Response:
25	15.760896942	172.16.1.10	172.16.70.1	FTP	129	Response: please report them via e-mail to <root@ftp.netlab.fe.up.pt>.
26	15.760911888	172.16.70.1	172.16.1.10	TCP	66	49002 → 21 [ACK] Seq=23 Ack=317 Win=64128 Len=0 TSval=2084088921 TSecr=3462820183
27	15.760945692	172.16.1.10	172.16.70.1	FTP	94	Response:
28	15.760988226	172.16.70.1	172.16.1.10	FTP	72	Request: pasv
29	15.761414821	172.16.1.10	172.16.70.1	TCP	66	21 → 49002 [ACK] Seq=345 Ack=29 Win=65280 Len=0 TSval=3462820184 TSecr=2084088921
30	15.761945971	172.16.1.10	172.16.70.1	FTP	116	Response: 227 Entering Passive Mode (172,16,1,10,174,225).
31	15.762018886	172.16.70.1	172.16.1.10	TCP	74	51384 → 44769 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=2084088922 TSecr=0 WS=128
32	15.762399107	172.16.1.10	172.16.70.1	TCP	74	44769 → 51384 [SYN, ACK] Seq=0 Ack=1 Win=65160 Len=0 MSS=1460 SACK_PERM=1 TSval=3462820185 TSe…
33	15.762422155	172.16.70.1	172.16.1.10	TCP	66	51384 → 44769 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=2084088923 TSecr=3462820185
34	15.762436193	172.16.70.1	172.16.1.10	FTP	81	Request: retr pipe.txt
35	15.763488816	172.16.1.10	172.16.70.1	FTP	131	Response: 150 Opening ASCII mode data connection for pipe.txt (418 bytes)
36	15.764051013	172.16.1.10	172.16.70.1	FTP-DA…	484	FTP Data: 418 bytes (PASV) (retr pipe.txt)
37	15.764068683	172.16.70.1	172.16.1.10	TCP	66	51384 → 44769 [ACK] Seq=1 Ack=419 Win=64128 Len=0 TSval=2084088924 TSecr=3462820186
38	15.804163592	172.16.70.1	172.16.1.10	TCP	66	49002 → 21 [ACK] Seq=44 Ack=460 Win=64128 Len=0 TSval=2084088964 TSecr=3462820186
39	15.965013487	172.16.1.10	172.16.70.1	TCP	66	44769 → 51384 [FIN, PSH, ACK] Seq=419 Ack=1 Win=65280 Len=0 TSval=3462820388 TSecr=2084088924
40	15.965200793	172.16.70.1	172.16.1.10	TCP	66	51384 → 44769 [FIN, ACK] Seq=1 Ack=420 Win=64128 Len=0 TSval=2084089125 TSecr=3462820388
41	15.965506981	172.16.1.10	172.16.70.1	TCP	66	44769 → 51384 [ACK] Seq=420 Ack=2 Win=65280 Len=0 TSval=3462820388 TSecr=2084089125
42	15.968168596	172.16.1.10	172.16.70.1	FTP	89	Response: 226 Transfer complete
43	15.968178234	172.16.70.1	172.16.1.10	TCP	66	49002 → 21 [ACK] Seq=44 Ack=483 Win=64128 Len=0 TSval=2084089128 TSecr=3462820391
44	15.968219930	172.16.70.1	172.16.1.10	FTP	72	Request: QUIT
45	15.968870719	172.16.1.10	172.16.70.1	FTP	80	Response: 221 Goodbye.
46	15.968898656	172.16.70.1	172.16.1.10	TCP	66	49002 → 21 [FIN, ACK] Seq=50 Ack=497 Win=64128 Len=0 TSval=2084089129 TSecr=3462820391
47	15.969328255	172.16.1.10	172.16.70.1	TCP	66	21 → 49002 [FIN, ACK] Seq=497 Ack=51 Win=65280 Len=0 TSval=3462820392 TSecr=2084089129
48	15.969341594	172.16.70.1	172.16.1.10	TCP	66	49002 → 21 [ACK] Seq=51 Ack=498 Win=64128 Len=0 TSval=2084089130 TSecr=3462820392

Anexo 3.6.2 – Experiência 6, passo 4 (tux73.eth1)



Anexo 3.6.3 – Experiência 6, passo 5 (tux73.eth1)