



**FEUP** **FACULDADE DE ENGENHARIA**  
**UNIVERSIDADE DO PORTO**

## **Redes de computadores**

**2º trabalho prático**

**Licenciatura em Engenharia Informática e Computação**

**Diogo Fernandes (202108752)**  
**José Sereno (202108729)**

Dezembro 2023

# Contents

<b>1</b>	<b>Introdução</b>	<b>2</b>
<b>2</b>	<b>Desenvolvimento da aplicação</b>	<b>2</b>
2.1	Estrutura do código	2
2.2	Fluxo do programa	3
<b>3</b>	<b>Configurar e analisar o funcionamento de uma rede</b>	<b>3</b>
3.1	Experiência 1	3
3.1.1	Arquitetura da rede	3
3.1.2	Objetivo	3
3.1.3	Quais são os comandos necessários para esta experiência?	3
3.1.4	O que são pacotes ARP e para que são usados?	4
3.1.5	O que são os endereços MAC e IP dos pacotes ARP e porquê?	4
3.1.6	Que pacotes é que o comando ping gera?	4
3.1.7	Quais são os MAC e IP dos pacotes ping?	4
3.1.8	Como determinar se um pacote é ARP, IP ou ICMP?	4
3.1.9	Como determinar o tamanho de um pacote recebido?	4
3.1.10	O que é a interface loopback e porque é que é importante?	4
3.1.11	Análise dos logs	4
3.2	Experiência 2	5
3.2.1	Arquitetura da rede	5
3.2.2	Objetivo	5
3.2.3	Quais são os comandos necessários para esta experiência?	5
3.2.4	Quantos domínios de broadcast existem? O que podemos concluir a partir dos logs? / Análise dos logs	5
3.3	Experiência 3	5
3.3.1	Arquitetura da rede	5
3.3.2	Objetivo	5
3.3.3	Quais são os comandos necessários para esta experiência?	5
3.3.4	Que rotas existem nos computadores? Qual o seu significado?	6
3.3.5	Que informação contém uma entrada da tabela de encaminhamento?	6
3.3.6	Que mensagens ARP, e os endereços MAC, são observadas e porquê? / Análise dos logs	6
3.4	Experiência 4	6
3.4.1	Configuração de um Router Comercial e Implementação de NAT	6
3.4.2	Quais são os comandos necessários para esta experiência?	6
3.4.3	Como configurar uma rota estática num router comercial?	6
3.4.4	Quais são os caminhos seguidos pelos pacotes nas experiências, porquê?	7
3.4.5	Como configurar o NAT num router comercial?	7
3.4.6	O que faz o NAT?	7
3.4.7	Análise dos logs	7
3.5	Experiência 5	7
3.5.1	Arquitetura da rede	7
3.5.2	Objetivo	7
3.5.3	Quais são os comandos necessários para esta experiência?	7
3.5.4	Como configurar o DNS num host?	7
3.5.5	Que pacotes são trocados pelo DNS e que informação é transportada?	7
3.5.6	Análise dos logs	7
3.6	Experiência 6	8
3.6.1	Objective	8
3.6.2	Quantas conexões TCP são abertas pela aplicação FTP?	8
3.6.3	Em que conexão é transportada a informação de controlo do FTP?	8

3.6.4	Quais são as fases de uma conexão TCP? . . . . .	8
3.6.5	Como funciona o mecanismo ARQ TCP? Quais são os campos de TCP relevantes? . . . . .	8
3.6.6	Como é que o mecanismo de controlo de congestão do TCP funciona? Como é que a taxa de transferência de dados evolui com o passar do tempo? . . . . .	8
3.6.7	A taxa de transferência em ligações de dados TCP é perturbada pelo aparecimento de uma segunda ligação TCP? Como? . . . . .	8
4	Conclusão . . . . .	8
5	Anexos . . . . .	9
5.1	Código . . . . .	9
5.1.1	download.h . . . . .	9
5.1.2	server.h . . . . .	9
5.1.3	url.h . . . . .	9
5.1.4	download.c . . . . .	10
5.1.5	server.c . . . . .	11
5.1.6	url.c . . . . .	14
5.1.7	main.c . . . . .	16
5.2	Wireshark . . . . .	16
5.2.1	Experiência 1 . . . . .	16
5.2.2	Experiência 2 . . . . .	17
5.2.3	Experiência 3 . . . . .	19
5.2.4	Experiência 4 . . . . .	20
5.2.5	Experiência 5 . . . . .	24
5.2.6	Experiência 6 . . . . .	24

## 1 Introdução

**Objetivos** Este projeto teve dois objetivos:

1. Desenvolver uma aplicação em **C** que implemente o protocolo **FTP** - descrito no [RFC959](#) - para fazer o download de um ficheiro através de um **URL** - a sintaxe deste URL deveria seguir o [RFC1738](#).
2. Configurar e analisar o funcionamento de uma rede de computadores.

o [RFC1738](#), para além de outras documentações, para entender o funcionamento do protocolo **FTP** e da sintaxe do **URL**.

**Sockets** Aprendemos a usar **sockets** em **C** como forma de comunicação entre o cliente e o servidor.

**DNS (Domain Name System)** Procurámos entender o funcionamento do **DNS** e como é que este é usado para traduzir um **URL** num endereço **IP**.

## 2 Desenvolvimento da aplicação

O programa **download** foi desenvolvido em **C** e tem como objetivo fazer o download de um ficheiro através de um **URL** que segue a sintaxe do [RFC1738](#). Através da realização deste programa aprofundamos os nossos conhecimentos sobre os seguintes temas:

**Client-Server & TCP/IP** Aprendemos como se dá a comunicação entre um cliente e um servidor através do protocolo **TCP/IP**.

**RFCs** RFCs são documentos que descrevem os padrões da internet. Consultámos, como referido anteriormente, o [RFC959](#) e

**UNIX** Aprendemos alguns comandos de **UNIX** que usamos no nosso programa (ex.: [getad-  
drinfo](#), [socket](#), [connect](#), [recv](#), [send](#))

### 2.1 Estrutura do código

O código do programa **download** está dividido em 3 ficheiros: **download.c**, **server.c** e **url.h**. É no ficheiro **download.c** onde se encontra a função **main** - onde se percebe o fluxo do programa. O ficheiro **server.c** contém as funções que implementam o protocolo **FTP** e o ficheiro **url.h** contém as funções que permitem fazer o **parse** do **URL**. O programa **download** é compilado através do comando **make** e é executado da seguinte forma:

## 2.2 Fluxo do programa

O programa é executado do seguinte comando:

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

O fluxo do programa será o seguinte:

1. **Parse** do **URL** para obter os seguintes campos: **user** (opcional), **password** (opcional), **host**, **port** (opcional, é usado o valor 21 por default no protocolo FTP) e **url-path**.
2. Criação de uma **socket** que, através de uma ligação TCP/IP, inicia uma conexão com o **host** na porta **port**.
3. Envio dos comandos **USER** e **PASS** para o servidor juntos dos valores **user** e **password**, respetivamente. Caso estes valores não tenham sido especificados, usa-se o valor **anonymous**.
4. Envio do comando **PASV** para o servidor para que este abra uma porta à qual nos conectaremos para futuramente recebermos o ficheiro.
5. Criação de uma nova **socket** que, através de uma ligação TCP/IP, inicia uma conexão com a nova porta aberta pelo servidor.
6. Envio do comando **RETR** através da porta inicial para o servidor junto do valor **url-path** para que o servidor nos envie o recurso especificado.
7. Leitura do ficheiro através da **socket** criada no ponto 5 e escrita do mesmo para um ficheiro local.
8. Fecho das **sockets** criadas.

O **parse** do **URL** é feito a partir da função **parse\_url** que recebe uma **string** e devolve uma **struct URL** com os campos especificados no ponto 1. Esta função usa uma máquina de estados e expressões regulares.

Todo o processo de comunicação entre o cliente e o servidor é feito através de **sockets** e o protocolo usado é o **TCP/IP**. A criação das **sockets** é feita através da função **getaddrinfo** que recebe o **host-name** e a **porta** e devolve uma **struct addrinfo** com os campos necessários para a criação da **socket**.

As **sockets** são criadas através da função **socket** e são feitas as respetivas ligações através da função **connect**.

O envio de comandos pelas **sockets** é feito através

da função **send** e a leitura das respostas do servidor é feita através da função **recv**.

Antes de ser enviado qualquer comando para o servidor, primeiro é feita a leitura do código de status do servidor através da função **recv**, de modo a termos uma noção do estado do servidor.

A escrita do ficheiro no disco é feita através da função **write**.

## 3 Configurar e analisar o funcionamento de uma rede

O objetivo deste conjunto de experiências é configurar uma rede de computadores de modo a que estes tenham acesso à internet para instalar ficheiros a partir de um servidor remoto usando o protocolo FTP desenvolvido ou seja, a aplicação.

### 3.1 Experiência 1

#### 3.1.1 Arquitetura da rede

No fim desta experiência, a configuração da rede deverá consistir em 2 computadores (TUX63 e TUX64) conectados pelo Switch.

#### 3.1.2 Objetivo

O propósito desta experiência foi a configuração de dois computadores na mesma rede de modo a permiti-los comunicar.

#### 3.1.3 Quais são os comandos necessários para esta experiência?

```
# TUX63:  
ifconfig eth0 up  
ifconfig eth0 172.16.60.1/24  
  
# TUX64:  
ifconfig eth0 up  
ifconfig eth0 172.16.60.254/24
```

Estando conectados ao mesmo **Switch**, os computadores comunicam entre si usando a bridge default. Podemos testar a comunicação entre os dois computadores usando o comando **ping** passando o endereço IP do outro computador como argumento:

```
ping 172.16.60.254 # No TUX63
```

### 3.1.4 O que são pacotes ARP e para que são usados?

ARP (**A**ddress **R**esolution **P**rotocol) é um protocolo que traduz endereços IPv4 em endereços MAC numa LAN (Local Area Network). Esta tradução é importante porque apesar de os IPs serem usados para identificar os computadores numa rede, podem mudar ao longo do tempo e devido ao ambiente. Já os MACs são usados para identificar o hardware de um computador e são únicos e imutáveis.

Numa situação em que um **computador 1** quer enviar uma mensagem para um **computador 2**, este começa por verificar se o endereço IP do computador 2 está na sua cache de vizinhos de rede. Caso contrário, o **computador 1** terá de fazer uma tradução do endereço IP do **computador 2** para o seu endereço MAC. Para fazer esta tradução, o ARP fará um broadcast - envia para todos os computadores da rede - de um pedido ARP que apenas será respondido pelo **computador 2**. O **computador 2** responde ao pedido ARP com o seu endereço MAC e o **computador 1** guarda-o na sua cache de vizinhos de rede.

O comando de **UNIX**: **arp**; é usado para manipular ou exibir a cache de vizinhos de rede IPv4 do kernel.

### 3.1.5 O que são os os endereços MAC e IP dos pacotes ARP e porquê?

O MAC **Media Access Control** é um endereço físico que identifica um dispositivo numa rede. É usado na data link layer para assegurar o endereço físico do computador - isto significa que está relacionado com o hardware. Os endereços MAC são únicos e não podem ser alterados. É composto por 6 bytes (48 bits) e é representado em hexadecimal. O IP **Internet Protocol** é um endereço lógico que identifica uma conexão entre um computador e uma rede. Pode mudar ao longo do tempo e devido ao ambiente. É usado na network layer para assegurar o endereço lógico do computador - isto significa que está relacionado com o software. Os endereços IP podem ser facilmente encontrados por terceiros, pois são transmitidos pela internet. É composto por 4 bytes (32 bits) e é representado em decimal.

### 3.1.6 Que pacotes é que o comando ping gera?

O comando **ping** gera pacotes **ARP** e pacotes **ICMP**. ICMP (**I**nternet **C**ontrol **M**essage **P**rotocol)

é um protocolo da camada de rede que reporta erros e fornece outras informações relevantes para o processamento de pacotes IP. Neste contexto, o ICMP é usado pelo comando ping para testar uma conexão de rede IP.

### 3.1.7 Quais são os MAC e IP dos pacotes ping?

Ver o ponto 3.1.11.

### 3.1.8 Como determinar se um pacote é ARP, IP ou ICMP?

É possível determinar se um frame Ethernet recebido é **ARP**, **IP**, **ICMP** verificando a captura do **WireShark**, na coluna Protocol. O **WireShark** faz esta distinção através do campo Type do cabeçalho Ethernet. O valor 0x0800 indica que o pacote é IP ou ICMP já que este se encontra guardado no pacote do IPv4, o valor 0x0806 indica que o pacote é ARP.

### 3.1.9 Como determinar o tamanho de um pacote recebido?

O tamanho de um pacote recebido pode ser determinado através da captura do **WireShark**, na coluna Length. Além disso, o tamanho dos pacotes IPv4 pode ser determinado a partir de 2 bytes que se encontram no pacote. Quanto aos pacotes ARP, estes possuem um tamanho fixo de 28 bytes.

### 3.1.10 O que é a interface loopback e porque é que é importante?

A interface **loopback** é uma interface de rede virtual que permite que um computador comunique consigo mesmo. É importante pois permite testar a stack de protocolos **TCP/IP** sem a necessidade de uma rede física. Essencialmente, a interface loopback, muitas vezes identificada pelo endereço **IP 127.0.0.1**, cria um ambiente isolado no próprio dispositivo, onde os dados enviados são retornados para a si.

### 3.1.11 Análise dos logs

Inicialmente, o TUX63 não sabe o endereço MAC do TUX64 e vice-versa. Assim, o TUX63 envia um pacote ARP para o **broadcast** (MAC FF:FF:FF:FF:FF:FF) com o endereço IP do TUX64. O TUX64 recebe o pacote ARP vindo do **broadcast**, que contém o IP do TUX63, e responde ao TUX63 de modo a indicar que guardou o seu endereço IP no pedido ARP, enviando o seu

endereço MAC. O TUX63 ao receber este pacote, guarda o endereço MAC do TUX64 na sua tabela ARP. Permitindo assim obter um PING request e PING response. Os endereços IP e Mac obtidos são os seguintes:

PC	IP	MAC
TUX63	172.16.60.1	00:21:5a:5a:75:bb
TUX64	172.16.60.254	00:21:5a:61:2d:df

## 3.2 Experiência 2

### 3.2.1 Arquitetura da rede

No final da experiência, a configuração da rede deverá consistir em 2 computadores (**TUX63** e **TUX64**) conectados à bridge (**bridge0**) e um computador (**TUX62**) conectado à bridge (**bridge1**).

### 3.2.2 Objetivo

Esta experiência teve como objetivo ensinar-nos a configurar 2 domínios de rede diferentes no mesmo switch e verificar que por defeito não ocorre comunicação entre estes domínios, sendo necessária a sua configuração.

### 3.2.3 Quais são os comandos necessários para esta experiência?

```
# Continuando a partir da experiencia
anterior:

# No tux2
ifconfig eth0 up
ifconfig eth0 172.16.61.1/24

# Switch
/system reset-configuration

/interface bridge add name=bridge60
/interface bridge add name=bridge61

# No tux3, para a sua porta eth0 X, no
tux4 para a sua porta eth0 Y e no
tux2 para a sua porta eth0 Z:
/interface bridge port remove [find
interface=etherX]
/interface bridge port remove [find
interface=etherY]
/interface bridge port remove [find
interface=etherZ]
/interface bridge port add bridge=
bridge60 interface=etherX
/interface bridge port add bridge=
bridge60 interface=etherY
/interface bridge port add bridge=
bridge61 interface=etherZ
```

### 3.2.4 Quantos domínios de broadcast existem? O que podemos concluir a partir dos logs? / Análise dos logs

Como configuramos 2 bridges, podemos concluir que existem 2 domínios de broadcast. Isto porque cada bridge é um domínio de broadcast. Podemos concluir isto a partir dos logs pois o TUX3 obteve uma resposta do TUX4, mas não do TUX2. Isto significa que o TUX3 está no mesmo domínio de broadcast que o TUX4, mas não no mesmo domínio de broadcast que o TUX2. Além disso, é possível concluir que passaram a existir 2 sub-redes, uma para cada bridge. Isto porque, apesar de ser possível fazer ping do TUX3 para o TUX4, não é possível fazer ping do TUX3 para o TUX2 já que, nos logs, não existem pacotes ICMP.

## 3.3 Experiência 3

### 3.3.1 Arquitetura da rede

No final desta experiência, é esperado que tenhamos uma arquitetura semelhante à da experiência anterior, com uma nova da conexão TUX64 à bridge (**bridge61**) que mediará a comunicação entre as duas bridges.

### 3.3.2 Objetivo

O objetivo desta experiência foi ensinar-nos a transformar o TUX64 num router e a configurar o mesmo para que este possa comunicar com os restantes computadores e permitir que estes comuniquem entre si.

### 3.3.3 Quais são os comandos necessários para esta experiência?

```
# Continuando a partir da
experiencia anterior:

# Switch - seja agora a conexao do
eth1 do tux4 a bridge61 pela
porta W

/interface bridge port remove [
find interface=etherW]
/interface bridge port add bridge=
bridge61 interface=etherW

# No 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
```



```
# No tux3
route add -net 172.16.61.0/24 gw
172.16.60.254

# No tux2
route add -net 172.16.60.0/24 gw
172.16.61.253
```

Nota: Devido a não conseguirmos obter os logs na nossa bancada, foi necessário utilizar outros ips, mas o procedimento é o mesmo.

### 3.3.4 Que rotas existem nos computadores? Qual o seu significado?

Existem 2 rotas, no TUX62 e no TUX63. Como a rota 64 é um gateway de ambos, a rota 62 é para chegar ao 63 e a rota 63 é para chegar ao 62, passando pelo 64.

### 3.3.5 Que informação contém uma entrada da tabela de encaminhamento?

Uma entrada da tabela de encaminhamento contém o endereço de destino/origem, o endereço de gateway e a máscara de rede.

### 3.3.6 Que mensagens ARP, e os endereços MAC, são observadas e porquê? / Análise dos logs

No caso do ping do TUX63 para o TUX62. As mensagens ARP trocadas contém apenas os endereços MAC do TUX63 e do TUX64 e não do destino final (TUX62). Isto ocorre devido à existência da rota. O TUX63 não conhece o endereço do TUX62, apenas conhece o endereço do gateway (TUX64) que leva ao TUX62.

Quando se apaga as tabelas ARP no TUX64 e se corre o mesmo ping novamente, os 3 computadores não se conhecem, pois não sabem os endereços MAC uns dos outros. Ao realizar o ping, é lançado um pedido ARP para o **broadcast** e para a sub-net da bridge60 a pedir o endereço MAC do TUX64, default gateway do TUX62. É gerada a resposta ARP e esta é enviada de volta para o TUX63, que guarda o endereço MAC do TUX64 na sua tabela ARP e vice-versa. De seguida, o ping passa pelo TUX64 e alcança o TUX62, sendo realizado o mesmo processo de troca de mensagens ARP entre o TUX64 e o TUX62.

É importante notar que o TUX63 não tem informação sobre o endereço MAC do TUX62 e vice-versa. Cada um destes computadores apenas conhece o endereço MAC do seu gateway, que é o TUX64.

## 3.4 Experiência 4

### 3.4.1 Configuração de um Router Comercial e Implementação de NAT

Nesta experiência foi-nos pedido que configurássemos um router comercial na nossa bridge (**bridge61**).

### 3.4.2 Quais são os comandos necessários para esta experiência?

```
# Router Serial Console
/interface bridge port remove [find
interface=ether5]
/interface bridge port add bridge=
bridge61 interface=ether5
/ip address add address=172.16.2.69/24
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=172.16.2.254
/ip firewall nat disable 0
/ip firewall nat enable 0

# Terminais dos TUXs
route add default gw 172.16.61.254 #
No tux62
route add default gw 172.16.60.254 #
No tux63
route add default gw 172.16.61.254 #
No tux64
sysctl net.ipv4.conf.eth0.
accept_redirects=0
sysctl net.ipv4.conf.all.
accept_redirects=0
route del -net 172.16.60.0 gw
172.16.61.253 netmask
255.255.255.0
traceroute -n
route add -net 172.16.60.0/24 gw
172.16.61.253
sysctl net.ipv4.conf.eth0.
accept_redirects=0
sysctl net.ipv4.conf.all.
accept_redirects=0
```

### 3.4.3 Como configurar uma rota estática num router comercial?

Para configurarmos uma rota estática no router comercial começamos por ligar o router ao **TUX63** através de um cabo de série. De seguida, acedemos à consola de comandos do router através do **GTKTerm** de modo a podermos configurar o router. Adicioná-mo-lo à bridge pretendida (**bridge61**) e atribuímos-lhe um IP na LAN. Temos também de lhe atribuir um IP para a rede exterior. Definimos também as rotas necessárias para chegar às redes

da bridge ([bridge60](#)) e para chegar à internet, indicando os [gateways](#) necessários.

### 3.4.4 Quais são os caminhos seguidos pelos pacotes nas experiências, porquê?

Com os redirects desativados, os pacotes seguem o caminho TUX62 → TUX64 → TUX63. Isto acontece porque o TUX62 não sabe que o TUX64 é um router e, portanto, envia os pacotes para o TUX64. O TUX64, por sua vez, envia os pacotes para o TUX63, que é o destino final. Com os redirects ativados, os pacotes seguem o caminho TUX62 → TUX63. Isto acontece porque o TUX62 sabe que o TUX64 é um router e, portanto, envia os pacotes diretamente para o TUX63.

### 3.4.5 Como configurar o NAT num router comercial?

Para configurar o NAT num router comercial, temos de aceder à consola de comandos do router através do [GTKTerm](#) e executar os seguintes comandos:

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

### 3.4.6 O que faz o NAT?

O NAT ([Network Address Translation](#)) é um processo que permite que vários computadores partilhem um único endereço IP que é usado na comunicação com o exterior. O NAT é usado para traduzir endereços IP de uma [LAN](#) (i.e. privados) para endereços IP públicos. Este processo é muito importante uma vez que o número de endereços IPv4 públicos é limitado e, portanto, não é possível atribuir um endereço IPv4 público a cada dispositivo que se liga à internet.

### 3.4.7 Análise dos logs

Desenvolvendo o tópico abordado em [3.4.4](#), ao enivarmos um [ping](#) do [TUX62](#) para o [TUX63](#) enquanto temos os redirects desativados, e após ter sido apagada a rota do [TUX62](#) para a rede [172.16.60.0/24](#) através do [TUX64](#), verificamos que o [TUX62](#) envia o [ping](#) para o [router](#) pela default route que definimos até conseguir chegar ao [TUX63](#). Já quando ativamos os redirects, o [TUX62](#) envia o [ping](#) para o [TUX64](#) e este reencaminha-o para o [TUX63](#). Isto acontece pois a ligação mais direta da rede passou a ser através do [TUX64](#) e não o router. O [TUX63](#) responde

ao [TUX64](#) e este reencaminha a resposta para o [TUX62](#).

## 3.5 Experiência 5

### 3.5.1 Arquitetura da rede

A arquitetura desta experiência é a mesma da experiência anterior.

### 3.5.2 Objetivo

O objetivo desta experiência foi ensinar-nos a dar ping a hosts com a utilização do DNS.

### 3.5.3 Quais são os comandos necessários para esta experiência?

```
# Continuando a experiencia anterior:

# No tux2
echo 'nameserver 172.16.2.1' > /etc/
    resolv.conf

# No tux3
echo 'nameserver 172.16.2.1' > /etc/
    resolv.conf
```

### 3.5.4 Como configurar o DNS num host?

No terminal do TUX2 e do TUX3, precisamos de correr o comando `hlsudo nano /etc/resolv.conf` e adicionar a seguinte linha:

```
nameserver <DNS IP address>
```

O [DNS](#) ([Domain Name System](#)) mapeia um nome de um host/domínio para endereços de IP. Portanto, ao utilizar este comando, estamos a permitir dar pings a hosts e domínios.

### 3.5.5 Que pacotes são trocados pelo DNS e que informação é transportada?

Os pacotes DNS trocados são o DNS query e o DNS response. O DNS query contém o nome de domínio e o DNS response contém o endereço IP do nome de domínio. Tornando possível traduzir o nome de domínio num endereço IP no router.

### 3.5.6 Análise dos logs

Nesta experiência, a partir do TUX62, damos ping ao google.com. Como google.com não é um IP, é necessário utilizar o [DNS](#). Inicialmente, o DNS procura os nameservers que estão definidos no ficheiro `/etc/resolv.conf`. De seguida, o DNS envia um pacote [DNS query](#) para o nameserver.



O nameserver responde com um pacote **DNS response** que contém o endereço IP do google.com, permitindo com que o ping seja bem sucedido.

## 3.6 Experiência 6

### 3.6.1 Objective

Nesta experiência devemos testar a aplicação que desenvolvemos na primeira parte do trabalho prático dentro da rede que configuramos nas experiências anteriores.

### 3.6.2 Quantas conexões TCP são abertas pela aplicação FTP?

A aplicação FTP abre duas conexões TCP. Uma para o controlo da aplicação e outra para a transferência de dados.

### 3.6.3 Em que conexão é transportada a informação de controlo do FTP?

A informação de controlo do FTP é transportada na primeira conexão TCP que é aberta pela aplicação.

### 3.6.4 Quais são as fases de uma conexão TCP?

As fases de uma conexão TCP são as seguintes:

1. **Estabelecimento de conexão** - O cliente envia um pacote **SYN** para o servidor para iniciar uma conexão TCP. O servidor responde com um pacote **SYN-ACK** para o cliente. O cliente responde com um pacote **ACK** para o servidor.
2. **Transferência de dados** - O cliente e o servidor trocam dados.
3. **Encerramento da ligação** - O cliente envia um pacote **FIN** para o servidor para terminar a conexão TCP. O servidor responde com um pacote **ACK** para o cliente. Opcionalmente, o servidor também pode enviar um pacote **FIN** para o cliente para terminar a conexão TCP ao qual o cliente responderá com um pacote **ACK** para o servidor.

### 3.6.5 Como funciona o mecanismo ARQ TCP? Quais são os campos de TCP relevantes?

O mecanismo ARQ (**Automatic Repeat request**) TCP é um mecanismo que permite que o

protocolo TCP recupere pacotes perdidos. Este protocolo assegura que os pacotes são entregues ao destino sem erros e na ordem correta. Este mecanismo opera através de ACKs (**Acknowledgement**) e timeouts. Quando um pacote é enviado, o emissor espera por um ACK. Se o ACK não chegar dentro de um determinado período de tempo, o emissor reenvia o pacote. O campo de TCP que reflete este protocolo é o campo "Sequence Number" que indica o número de sequência do pacote. Este campo é usado para ordenar os pacotes e para verificar se algum pacote foi perdido.

### 3.6.6 Como é que o mecanismo de controlo de congestão do TCP funciona? Como é que a taxa de transferência de dados evolui com o passar do tempo?

O protocolo de controlo de congestão do TCP visa otimizar o desempenho da transferência de dados ao mesmo tempo que procura evitar a congestão da rede. A taxa de transferência de dados começa lenta e aumenta exponencialmente até que ocorra uma perda de pacotes. Quando uma perda de pacotes ocorre, a taxa de transferência de dados diminui e aumenta gradualmente até que ocorra uma nova perda de pacotes.

### 3.6.7 A taxa de transferência em ligações de dados TCP é perturbada pelo aparecimento de uma segunda ligação TCP? Como?

A taxa de transferência numa ligação de dados TCP é perturbada pelo aparecimento de uma segunda ligação uma vez que o protocolo TCP divide a sua largura de banda igualmente entre as ligações.

## 4 Conclusão

Com a realização completa e correta das experiências, como também da aplicação de download. Foi possível aprender mais sobre o funcionamento e configuração de uma rede como também e sobre os protocolos envolvidos na transferência dos dados, tanto ao longo da propagação pela network layer como na link layer.

## 5 Anexos

### 5.1 Código

#### 5.1.1 download.h

```
#ifndef DOWNLOAD_H
#define DOWNLOAD_H

#include <stdlib.h>
#include <stdio.h>

#include "url.h"
#include "server.h"

int download(const char *arg);

#endif
```

#### 5.1.2 server.h

```
#ifndef UTIL_H
#define UTIL_H

#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <unistd.h>
#include <netdb.h>
#include <sys/types.h>
#include <sys/socket.h>

/**
 * https://man7.org/linux/man-pages/man3/getaddrinfo.3.html
 */

// Server Status Codes
#define SERVER_LOGIN_READY 220
#define SERVER_PASSWORD_REQUIRED 331
#define SERVER_LOGIN_SUCCESS 230
#define SERVER_PASSIVE_READY 227
#define FILE_STATUS_OKAY 150
#define CLOSING_DATA_CONNECTION 226

int get_status(int sfd);

int get_connection(const char *hostname, const char *port);

void auth(int sfd, const char *username, const char *password);

void get_passive(int sfd, char *host, char *port);

void request_file(int sfd, const char *filename);

void get_file(int psfd, const char *filename);

#endif
```

#### 5.1.3 url.h

```
#ifndef URL_H
#define URL_H

#include <stdio.h>
#include <stdlib.h>
```

```

#include <string.h>

enum url_read_state
{
    PROTOCOL,
    USER,
    HOST,
    PORT,
    PATH,
    END,
};

struct URL
{
    char protocol[16];
    char username[256];
    char password[256];
    char hostname[256];
    char ip[256];
    char port[16];
    char path[256];
};

struct URL parse_url(const char *arg);

void print_url(struct URL url);

const char *get_filename(const char *path);

#endif

```

#### 5.1.4 download.c

```

#include "download.h"

int download(const char *arg)
{
    printf("-----\n"); // PARSE URL
    struct URL url = parse_url(arg);
    print_url(url);

    printf("-----\n"); // CONNECT TO SERVER
    int sfd = get_connection(url.hostname, url.port);
    printf("Connected to server (socket fd: %d)\n", sfd);

    printf("-----\n"); // AUTHENTICATE
    auth(sfd, url.username, url.password);
    printf("Successfully authenticated as %s\n", url.username);

    printf("-----\n"); // OPEN PASSIVE MODE
    char psv_hostname[INET6_ADDRSTRLEN], psv_port[6];
    get_passive(sfd, psv_hostname, psv_port);
    printf("Passive mode address: %s:%s\n", psv_hostname, psv_port);

    printf("-----\n"); // CONNECT TO PASSIVE
    int psfd = get_connection(psv_hostname, psv_port);
    printf("Connected to passive (socket fd: %d)\n", psfd);

    printf("-----\n"); // DOWNLOAD FILE
    const char *filename = get_filename(url.path);
    printf("Requesting file %s\n", url.path);
    request_file(sfd, url.path);
    printf("Starting download of %s\n", filename);
    get_file(psfd, filename);

    printf("-----\n"); // CLOSE CONNECTIONS
}

```

```

        close(sfd);
        printf("Closed connection to server\n");
        close(psfd);
        printf("Closed connection to passive\n");

        return 0;
    }

```

### 5.1.5 server.c

```

#include "server.h"

// #include <arpa/inet.h>
// #include <netinet/in.h>

int get_status(int sfd)
{
    char buffer[1024];
    ssize_t total_bytes_read = 0;

    while (1)
    {
        usleep(100000);
        ssize_t bytes_read = recv(sfd, buffer, sizeof buffer - total_bytes_read,
            MSG_DONTWAIT);
        if (bytes_read < 0)
        {
            buffer[total_bytes_read] = '\0';
            break;
        }
        total_bytes_read += bytes_read;
    }

    int status;
    if (sscanf(buffer, "%d", &status) != 1)
    {
        fprintf(stderr, "Invalid status line: %s\n", buffer);
        exit(EXIT_FAILURE);
    }

    // printf("Full message:\n\"%s\"\n", buffer); // debug2
    // printf("Status: %d\n\n", status);           // debug2

    return status;
}

int get_connection(const char *hostname, const char *port)
{
    int sfd, s;
    struct addrinfo hints;
    struct addrinfo *result, *rp;
    // char ipstr[INET6_ADDRSTRLEN];

    memset(&hints, 0, sizeof hints);
    hints.ai_family = AF_INET;
    hints.ai_socktype = SOCK_STREAM;
    hints.ai_protocol = IPPROTO_TCP;

    s = getaddrinfo(hostname, port, &hints, &result);
    if (s != 0)
    {
        fprintf(stderr, "getaddrinfo: %s\n", gai_strerror(s));
        exit(EXIT_FAILURE);
    }

    for (rp = result; rp != NULL; rp = rp->ai_next)

```

```

{
    sfd = socket(rp->ai_family, rp->ai_socktype, rp->ai_protocol);

    if (sfd == -1)
        continue;

    if (connect(sfd, rp->ai_addr, rp->ai_addrlen) != -1)
        break;

    close(sfd);
}

// struct sockaddr_in *ipv4 = (struct sockaddr_in *)rp->ai_addr;
// void *addr = &(ipv4->sin_addr);
// inet_ntop(rp->ai_family, addr, ipstr, sizeof ipstr);
// printf("Connected to %s:%s (%s)\n", hostname, port, ipstr);

freeaddrinfo(result);

if (rp == NULL)
{
    fprintf(stderr, "Could not connect\n");
    exit(EXIT_FAILURE);
}

return sfd;
}

void auth(int sfd, const char *username, const char *password)
{
    char buf[266];
    int command_length;

    // printf("1 Step ----- (check if service ready for new user) | =220\n");
    // debug1
    if (get_status(sfd) != SERVER_LOGIN_READY)
    {
        fprintf(stderr, "Service not ready for new user.\n");
        exit(EXIT_FAILURE);
    }

    // printf("2 Step ----- (sending user) | \n"); //
    // debug1
    command_length = snprintf(buf, 266, "user %s\n", username);
    if (send(sfd, buf, command_length, 0) < 0)
    {
        fprintf(stderr, "Error sending username (%s).\n", username);
        exit(EXIT_FAILURE);
    }

    // printf("3 Step ----- (check if service needs password) | =331\n");
    // debug1
    if (get_status(sfd) != SERVER_PASSWORD_REQUIRED)
    {
        fprintf(stderr, "Error, server should expect password\n");
        exit(EXIT_FAILURE);
    }

    // printf("4 Step ----- (send password) |\n"); //
    // debug1
    command_length = snprintf(buf, 266, "pass %s\n", password);
    if (send(sfd, buf, command_length, 0) < 0)
    {
        fprintf(stderr, "Error sending password (%s).\n", password);
        exit(EXIT_FAILURE);
    }
}

```

```

    // printf("5 Step ----- (check if login was successful) | =230\n");
    // debug1
    if (get_status(sfd) != SERVER_LOGIN_SUCCESS)
    {
        fprintf(stderr, "Login failed.\n");
        exit(EXIT_FAILURE);
    }
}

void get_passive(int sfd, char *host, char *port)
{
    char *in_buf = "pasv\n", out_buf[256];

    if (send(sfd, in_buf, strlen(in_buf), 0) < 0)
    {
        fprintf(stderr, "Error sending passive command.\n");
        exit(EXIT_FAILURE);
    }

    ssize_t bytes = recv(sfd, out_buf, sizeof out_buf - 1, 0);
    if (bytes < 0)
    {
        perror("recv");
        exit(EXIT_FAILURE);
    }
    out_buf[bytes] = '\0';

    int code;
    uint8_t h1, h2, h3, h4, p1, p2;
    sscanf(out_buf, "%d %*[^()(%hhu, %hhu, %hhu, %hhu, %hhu, %hhu)\n", &code, &h1, &h2,
        &h3, &h4, &p1, &p2);
    if (code != SERVER_PASSIVE_READY)
    {
        fprintf(stderr, "Error entering passive mode.\n");
        exit(EXIT_FAILURE);
    }

    sprintf(host, "%hhu.%hhu.%hhu.%hhu", h1, h2, h3, h4);
    sprintf(port, "%hu", p1 * 256 + p2);
}

void request_file(int sfd, const char *path)
{
    char buf[266];
    int command_length;

    command_length = snprintf(buf, 266, "retr %s\n", path);
    if (send(sfd, buf, command_length, 0) < 0)
    {
        fprintf(stderr, "Error sending file request (%s).\n", path);
        exit(EXIT_FAILURE);
    }

    int status = get_status(sfd);
    if (status == FILE_STATUS_OKAY || status == CLOSING_DATA_CONNECTION)
    {
        printf("File ok.\n");
        return;
    }
    else
    {
        fprintf(stderr, "File unavailable.\n");
        exit(EXIT_FAILURE);
    }
}

```



```

void get_file(int psfd, const char *filename)
{
    char output_path[256];
    strcpy(output_path, "output/");
    strcat(output_path, filename);

    FILE *fp = fopen(output_path, "w");
    if (fp == NULL)
    {
        fprintf(stderr, "Error opening file (%s).\n", filename);
        exit(EXIT_FAILURE);
    }

    char buf[256];
    ssize_t bytes;
    while ((bytes = recv(psfd, buf, sizeof buf - 1, 0)) > 0)
    {
        fwrite(buf, 1, bytes, fp);
    }

    fclose(fp);
}

```

### 5.1.6 url.c

```

#include "url.h"

struct URL parse_url(const char *arg)
{
    int state = PROTOCOL;
    struct URL url = {"", "", "", "", "", "", ""};

    while (state != END)
    {
        int n = 0;
        switch (state)
        {
            case PROTOCOL:
                // printf("PROTOCOL: %s\n", arg);
                if (sscanf(arg, "%15[^:@/]://%n", url.protocol, &n) && n > 0)
                {
                    state = USER;
                    arg += n;
                }
                else
                {
                    // fprintf(stderr, "error: protocol not found for url: %s\n", arg);
                    exit(EXIT_FAILURE);
                }
                break;
            case USER:
                // printf("USER: %s\n", arg);
                if (sscanf(arg, "%255[^:@/]:%255[^:@/]%n", url.username, url.password, &n)
                    && n > 0)
                {
                    arg += n;
                }
                else if (sscanf(arg, "%255[^:@/]%n", url.username, &n) && n > 0)
                {
                    url.password[0] = '\0';
                    arg += n;
                }
                else
                {
                    url.username[0] = '\0';

```

```

        url.password[0] = '\0';
    }
    state = HOST;
    break;
case HOST:
    // printf("HOST: %s\n", arg);
    if (sscanf(arg, "%255[^:/%n", url.hostname, &n) && n > 0)
    {
        state = PORT;
    }
    else if (sscanf(arg, "%255[^:/]%n", url.hostname, &n) && n > 0)
    {
        state = PATH;
    }
    else if (sscanf(arg, "%255[^:/%n", url.hostname, &n) && n > 0)
    {
        state = END;
    }
    else
    {
        // fprintf(stderr, "error: host not found for url: %s\n", arg);
        exit(EXIT_FAILURE);
    }
    arg += n;
    break;
case PORT:
    // printf("PORT: %s\n", arg);
    if (sscanf(arg, "%5[0123456789]%n", url.port, &n) && n > 0)
    {
        state = PATH;
    }
    else if (sscanf(arg, "%5[0123456789]%n", url.port, &n) && n > 0)
    {
        state = END;
    }
    else
    {
        // fprintf(stderr, "error: port not found for url: %s\n", arg);
        exit(EXIT_FAILURE);
    }
    arg += n;
    break;
case PATH:
    // printf("PATH: %s\n", arg);
    if (sscanf(arg, "%255[^:@]%n", url.path, &n) == -1)
    {
        exit(EXIT_FAILURE);
    }
    arg += n;
    state = END;
    break;
}
}
if (strlen(arg) > 0) // If there is something left in arg, it is an error
{
    exit(EXIT_FAILURE);
}
if (strlen(url.username) == 0) // If username is empty, set it to anonymous
{
    strcpy(url.username, "anonymous");
}
if (strlen(url.port) == 0) // If port is empty, set it to 21
{
    strcpy(url.port, "21");
}
return url;

```

```

}

void print_url(struct URL url)
{
    printf("protocol: %s\n", url.protocol);
    printf("username: %s\n", url.username);
    printf("password: %s\n", url.password);
    printf("host: %s\n", url.hostname);
    printf("port: %s\n", url.port);
    printf("path: %s\n", url.path);
}

const char *get_filename(const char *path)
{
    const char *filename = strrchr(path, '/');
    if (filename == NULL)
    {
        return path;
    }
    else
    {
        return filename + 1;
    }
}

```

### 5.1.7 main.c

```

#include <stdio.h>
#include <stdlib.h>

#include "download.h"

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

    return download(argv[1]);
}

```

## 5.2 Wireshark

### 5.2.1 Experiência 1

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	Routerboardc_1c:8d:...	Spanning-tree-(for-...	STP	60	RST, Root = 32768/0/c4:ad:34:1c:8d:2b Cost = 0 Port = 0x8018
2	2.002152079	Routerboardc_1c:8d:...	Spanning-tree-(for-...	STP	60	RST, Root = 32768/0/c4:ad:34:1c:8d:2b Cost = 0 Port = 0x8018
3	4.004311212	Routerboardc_1c:8d:...	Spanning-tree-(for-...	STP	60	RST, Root = 32768/0/c4:ad:34:1c:8d:2b Cost = 0 Port = 0x8018
4	6.006463989	Routerboardc_1c:8d:...	Spanning-tree-(for-...	STP	60	RST, Root = 32768/0/c4:ad:34:1c:8d:2b Cost = 0 Port = 0x8018
5	8.008646379	Routerboardc_1c:8d:...	Spanning-tree-(for-...	STP	60	RST, Root = 32768/0/c4:ad:34:1c:8d:2b Cost = 0 Port = 0x8018
6	10.010808174	Routerboardc_1c:8d:...	Spanning-tree-(for-...	STP	60	RST, Root = 32768/0/c4:ad:34:1c:8d:2b Cost = 0 Port = 0x8018
7	10.423904605	HewlettPacka_5a:75:...	Broadcast	ARP	42	Who has 172.16.60.1? Tell 172.16.60.254
8	10.424002682	HewlettPacka_a7:32:...	HewlettPacka_5a:75:...	ARP	60	172.16.60.1 is at 00:22:64:a7:32:ab
9	10.424017489	172.16.60.254	172.16.60.1	ICMP	98	Echo (ping) request id=0x44f8, seq=1/256, ttl=64 (reply in 10)
10	10.424108561	172.16.60.1	172.16.60.254	ICMP	98	Echo (ping) reply id=0x44f8, seq=1/256, ttl=64 (request in 9)
11	11.433163299	172.16.60.254	172.16.60.1	ICMP	98	Echo (ping) request id=0x44f8, seq=2/512, ttl=64 (reply in 12)
12	11.433299061	172.16.60.1	172.16.60.254	ICMP	98	Echo (ping) reply id=0x44f8, seq=2/512, ttl=64 (request in 11)
13	12.0457156169	Routerboardc_1c:8d:...	Spanning-tree-(for-...	STP	60	RST, Root = 32768/0/c4:ad:34:1c:8d:2b Cost = 0 Port = 0x8018
14	12.457156169	172.16.60.254	172.16.60.1	ICMP	98	Echo (ping) request id=0x44f8, seq=3/768, ttl=64 (reply in 15)
15	12.457286911	172.16.60.1	172.16.60.254	ICMP	98	Echo (ping) reply id=0x44f8, seq=3/768, ttl=64 (request in 14)
16	13.481135978	172.16.60.254	172.16.60.1	ICMP	98	Echo (ping) request id=0x44f8, seq=4/1024, ttl=64 (reply in 17)
17	13.481259178	172.16.60.1	172.16.60.254	ICMP	98	Echo (ping) reply id=0x44f8, seq=4/1024, ttl=64 (request in 16)
18	14.015134673	Routerboardc_1c:8d:...	Spanning-tree-(for-...	STP	60	RST, Root = 32768/0/c4:ad:34:1c:8d:2b Cost = 0 Port = 0x8018
19	14.505159569	172.16.60.254	172.16.60.1	ICMP	98	Echo (ping) request id=0x44f8, seq=5/1280, ttl=64 (reply in 20)
20	14.505282708	172.16.60.1	172.16.60.254	ICMP	98	Echo (ping) reply id=0x44f8, seq=5/1280, ttl=64 (request in 19)
21	15.490927290	HewlettPacka_a7:32:...	HewlettPacka_5a:75:...	ARP	60	Who has 172.16.60.254? Tell 172.16.60.1
22	15.490914650	HewlettPacka_5a:75:...	HewlettPacka_a7:32:...	ARP	42	172.16.60.254 is at 00:21:5a:5a:75:bb
23	15.529144556	172.16.60.254	172.16.60.1	ICMP	98	Echo (ping) request id=0x44f8, seq=6/1536, ttl=64 (reply in 24)
24	15.529235210	172.16.60.1	172.16.60.254	ICMP	98	Echo (ping) reply id=0x44f8, seq=6/1536, ttl=64 (request in 23)
25	16.017297437	Routerboardc_1c:8d:...	Spanning-tree-(for-...	STP	60	RST, Root = 32768/0/c4:ad:34:1c:8d:2b Cost = 0 Port = 0x8018
26	16.553154607	172.16.60.254	172.16.60.1	ICMP	98	Echo (ping) request id=0x44f8, seq=7/1792, ttl=64 (reply in 27)
27	16.553282277	172.16.60.1	172.16.60.254	ICMP	98	Echo (ping) reply id=0x44f8, seq=7/1792, ttl=64 (request in 26)
28	17.577147128	172.16.60.254	172.16.60.1	ICMP	98	Echo (ping) request id=0x44f8, seq=8/2048, ttl=64 (reply in 29)
29	17.577277032	172.16.60.1	172.16.60.254	ICMP	98	Echo (ping) reply id=0x44f8, seq=8/2048, ttl=64 (request in 28)

Figure 1: Experiência 1 - TUX64 j- TUX 63

## 5.2.2 Experiência 2

1	0.00000000	Routerboardc_1c:8d:...	Spanning-tree-(for-...	STP	60	RST. Root = 32768/0/c4:ad:34:1c:8d:33	Cost = 0	Port
2	2.002252220	Routerboardc_1c:8d:...	Spanning-tree-(for-...	STP	60	RST. Root = 32768/0/c4:ad:34:1c:8d:33	Cost = 0	Port
3	4.004517151	Routerboardc_1c:8d:...	Spanning-tree-(for-...	STP	60	RST. Root = 32768/0/c4:ad:34:1c:8d:33	Cost = 0	Port
4	6.006785923	Routerboardc_1c:8d:...	Spanning-tree-(for-...	STP	60	RST. Root = 32768/0/c4:ad:34:1c:8d:33	Cost = 0	Port
5	8.009039400	Routerboardc_1c:8d:...	Spanning-tree-(for-...	STP	60	RST. Root = 32768/0/c4:ad:34:1c:8d:33	Cost = 0	Port
6	8.180329493	0.0.0.0	255.255.255.255	MNDP	159	5678 → 5678 Len=117		
7	8.180362109	Routerboardc_1c:8d:...	CDP/VTP/DTP/PagP/UD...	CDP	93	Device ID: MikroTik Port ID: bridge60		
8	8.180408973	Routerboardc_1c:8d:...	LLDP_Multicast	LLDP	110	MA/c4:ad:34:1c:8d:33 IN/bridge60 120 SysN=MikroTik Sys		
9	10.011304191	Routerboardc_1c:8d:...	Spanning-tree-(for-...	STP	60	RST. Root = 32768/0/c4:ad:34:1c:8d:33	Cost = 0	Port
10	12.013578132	Routerboardc_1c:8d:...	Spanning-tree-(for-...	STP	60	RST. Root = 32768/0/c4:ad:34:1c:8d:33	Cost = 0	Port

Figure 2: Experiência 2 - TUX63 -j TUX62

1	0.00000000	Routerboardc_1c:8d:...	Spanning-tree-(for-...	STP	60	RST. Root = 32768/0/c4:ad:34:1c:8d:33	Cost = 0	
2	2.002194531	Routerboardc_1c:8d:...	Spanning-tree-(for-...	STP	60	RST. Root = 32768/0/c4:ad:34:1c:8d:33	Cost = 0	
3	3.536051241	0.0.0.0	255.255.255.255	MNDP	159	5678 → 5678 Len=117		
4	3.536073031	Routerboardc_1c:8d:...	CDP/VTP/DTP/PagP/UD...	CDP	93	Device ID: MikroTik Port ID: bridge60		
5	3.536121292	Routerboardc_1c:8d:...	LLDP_Multicast	LLDP	110	MA/c4:ad:34:1c:8d:33 IN/bridge60 120 SysN=MikroTik		
6	4.004376699	Routerboardc_1c:8d:...	Spanning-tree-(for-...	STP	60	RST. Root = 32768/0/c4:ad:34:1c:8d:33	Cost = 0	
7	6.006585477	Routerboardc_1c:8d:...	Spanning-tree-(for-...	STP	60	RST. Root = 32768/0/c4:ad:34:1c:8d:33	Cost = 0	
8	8.008775747	Routerboardc_1c:8d:...	Spanning-tree-(for-...	STP	60	RST. Root = 32768/0/c4:ad:34:1c:8d:33	Cost = 0	
9	10.010969369	Routerboardc_1c:8d:...	Spanning-tree-(for-...	STP	60	RST. Root = 32768/0/c4:ad:34:1c:8d:33	Cost = 0	
10	12.013168509	Routerboardc_1c:8d:...	Spanning-tree-(for-...	STP	60	RST. Root = 32768/0/c4:ad:34:1c:8d:33	Cost = 0	
11	14.015359338	Routerboardc_1c:8d:...	Spanning-tree-(for-...	STP	60	RST. Root = 32768/0/c4:ad:34:1c:8d:33	Cost = 0	
12	16.017552471	Routerboardc_1c:8d:...	Spanning-tree-(for-...	STP	60	RST. Root = 32768/0/c4:ad:34:1c:8d:33	Cost = 0	
13	18.019746932	Routerboardc_1c:8d:...	Spanning-tree-(for-...	STP	60	RST. Root = 32768/0/c4:ad:34:1c:8d:33	Cost = 0	
14	20.021949703	Routerboardc_1c:8d:...	Spanning-tree-(for-...	STP	60	RST. Root = 32768/0/c4:ad:34:1c:8d:33	Cost = 0	
15	22.024153592	Routerboardc_1c:8d:...	Spanning-tree-(for-...	STP	60	RST. Root = 32768/0/c4:ad:34:1c:8d:33	Cost = 0	
16	24.026345050	Routerboardc_1c:8d:...	Spanning-tree-(for-...	STP	60	RST. Root = 32768/0/c4:ad:34:1c:8d:33	Cost = 0	
17	26.028546704	Routerboardc_1c:8d:...	Spanning-tree-(for-...	STP	60	RST. Root = 32768/0/c4:ad:34:1c:8d:33	Cost = 0	

Figure 3: Experiência 2 - TUX63 -j TUX62

13	19.811426/52	Routerboardc_1c:8d:...	LLDP_Multicast	LLDP	110	MA/c4:ad:34:1c:8d:33 IN/bridge60 120 SysN=MikroTik		
14	20.020899420	Routerboardc_1c:8d:...	Spanning-tree-(for-...	STP	60	RST. Root = 32768/0/c4:ad:34:1c:8d:33	Cost = 0	
15	20.381163440	172.16.60.1	172.16.60.255	ICMP	98	Echo (ping) request id=0x4a91, seq=1/256, ttl=64		
16	20.381339093	172.16.60.254	172.16.60.1	ICMP	98	Echo (ping) reply id=0x4a91, seq=1/256, ttl=64		
17	21.412221910	172.16.60.1	172.16.60.255	ICMP	98	Echo (ping) request id=0x4a91, seq=2/512, ttl=64		
18	21.412410483	172.16.60.254	172.16.60.1	ICMP	98	Echo (ping) reply id=0x4a91, seq=2/512, ttl=64		
19	22.023101284	Routerboardc_1c:8d:...	Spanning-tree-(for-...	STP	60	RST. Root = 32768/0/c4:ad:34:1c:8d:33	Cost = 0	
20	22.436244016	172.16.60.1	172.16.60.255	ICMP	98	Echo (ping) request id=0x4a91, seq=3/768, ttl=64		
21	22.436426862	172.16.60.254	172.16.60.1	ICMP	98	Echo (ping) reply id=0x4a91, seq=3/768, ttl=64		
22	23.460234693	172.16.60.1	172.16.60.255	ICMP	98	Echo (ping) request id=0x4a91, seq=4/1024, ttl=64		
23	23.460417190	172.16.60.254	172.16.60.1	ICMP	98	Echo (ping) reply id=0x4a91, seq=4/1024, ttl=64		
24	24.025294976	Routerboardc_1c:8d:...	Spanning-tree-(for-...	STP	60	RST. Root = 32768/0/c4:ad:34:1c:8d:33	Cost = 0	
25	24.484253237	172.16.60.1	172.16.60.255	ICMP	98	Echo (ping) request id=0x4a91, seq=5/1280, ttl=64		
26	24.484437130	172.16.60.254	172.16.60.1	ICMP	98	Echo (ping) reply id=0x4a91, seq=5/1280, ttl=64		
27	25.436365337	HewlettPacka_5a:75:...	HewlettPacka_a7:32:...	ARP	60	Who has 172.16.60.1? Tell 172.16.60.254		
28	25.436382030	HewlettPacka_a7:32:...	HewlettPacka_5a:75:...	ARP	42	172.16.60.1 is at 00:22:64:a7:32:ab		
29	25.508227990	172.16.60.1	172.16.60.255	ICMP	98	Echo (ping) request id=0x4a91, seq=6/1536, ttl=64		

Figure 4: Experiência 2 - TUX63 -j TUX62

19	18.431958655	172.16.60.254	172.16.60.1	ICMP	98	Echo (ping) reply id=0x4a91, seq=3/768, ttl=64		
20	19.455913375	172.16.60.1	172.16.60.255	ICMP	98	Echo (ping) request id=0x4a91, seq=4/1024, ttl=64 (no response found!)		
21	19.455949622	172.16.60.254	172.16.60.1	ICMP	98	Echo (ping) reply id=0x4a91, seq=4/1024, ttl=64		
22	20.020900222	Routerboardc_1c:8d:...	Spanning-tree-(for-...	STP	60	RST. Root = 32768/0/c4:ad:34:1c:8d:33 Cost = 0 Port = 0x8002		
23	20.479934513	172.16.60.1	172.16.60.255	ICMP	98	Echo (ping) request id=0x4a91, seq=5/1280, ttl=64 (no response found!)		
24	20.479969364	172.16.60.254	172.16.60.1	ICMP	98	Echo (ping) reply id=0x4a91, seq=5/1280, ttl=64		
25	21.431878604	HewlettPacka_5a:75:...	HewlettPacka_a7:32:...	ARP	42	Who has 172.16.60.1? Tell 172.16.60.254		
26	21.432025200	HewlettPacka_a7:32:...	HewlettPacka_5a:75:...	ARP	60	172.16.60.1 is at 00:22:64:a7:32:ab		
27	21.503902363	172.16.60.1	172.16.60.255	ICMP	98	Echo (ping) request id=0x4a91, seq=6/1536, ttl=64 (no response found!)		
28	21.503926309	172.16.60.254	172.16.60.1	ICMP	98	Echo (ping) reply id=0x4a91, seq=6/1536, ttl=64		
29	22.023060340	Routerboardc_1c:8d:...	Spanning-tree-(for-...	STP	60	RST. Root = 32768/0/c4:ad:34:1c:8d:33 Cost = 0 Port = 0x8002		
30	22.527938238	172.16.60.1	172.16.60.255	ICMP	98	Echo (ping) request id=0x4a91, seq=7/1792, ttl=64 (no response found!)		
31	22.527972670	172.16.60.254	172.16.60.1	ICMP	98	Echo (ping) reply id=0x4a91, seq=7/1792, ttl=64		
32	23.551923968	172.16.60.1	172.16.60.255	ICMP	98	Echo (ping) request id=0x4a91, seq=8/2048, ttl=64 (no response found!)		
33	23.551958958	172.16.60.254	172.16.60.1	ICMP	98	Echo (ping) reply id=0x4a91, seq=8/2048, ttl=64		
34	24.025285907	Routerboardc_1c:8d:...	Spanning-tree-(for-...	STP	60	RST. Root = 32768/0/c4:ad:34:1c:8d:33 Cost = 0 Port = 0x8002		
35	24.575944128	172.16.60.1	172.16.60.255	ICMP	98	Echo (ping) request id=0x4a91, seq=9/2304, ttl=64 (no response found!)		
36	24.575978351	172.16.60.254	172.16.60.1	ICMP	98	Echo (ping) reply id=0x4a91, seq=9/2304, ttl=64		
37	26.027482590	Routerboardc_1c:8d:...	Spanning-tree-(for-...	STP	60	RST. Root = 32768/0/c4:ad:34:1c:8d:33 Cost = 0 Port = 0x8002		
38	28.029606537	Routerboardc_1c:8d:...	Spanning-tree-(for-...	STP	60	RST. Root = 32768/0/c4:ad:34:1c:8d:33 Cost = 0 Port = 0x8002		
39	30.031872605	Routerboardc_1c:8d:...	Spanning-tree-(for-...	STP	60	RST. Root = 32768/0/c4:ad:34:1c:8d:33 Cost = 0 Port = 0x8002		

Figure 5: Experiência 2 - TUX63 -j TUX64

no.	time	source	destination	protocol	length	info
1	0.000000000	Routerboard_ic18d...	Spanning-tree-(for...	STP	60	RST, Root = 32768/0/c4:ad34:1c:8d:33 Cost = 0 Port = 0x8001
2	2.002194531	Routerboard_ic18d...	Spanning-tree-(for...	STP	60	RST, Root = 32768/0/c4:ad34:1c:8d:33 Cost = 0 Port = 0x8001
3	3.536051241	0.0.0.0	255.255.255.255	RNDP	159	5678 + 5678 Len=117
4	5.536073831	Routerboard_ic18d...	CDP/VTP/DTP/PagP/UD...	CDP	93	Device ID: MikroTik Port ID: bridge60
5	5.556112152	Routerboard_ic18d...	LDP, Multicast	LDP	110	PA/c4:ad34:1c:8d:33 10/bridge60 120 SysD=MikroTik SysD=MikroTik RouterOS 6.43.16 (long-term) CRS326-24G-25+
6	4.004376699	Routerboard_ic18d...	Spanning-tree-(for...	STP	60	RST, Root = 32768/0/c4:ad34:1c:8d:33 Cost = 0 Port = 0x8001
7	6.006585477	Routerboard_ic18d...	Spanning-tree-(for...	STP	60	RST, Root = 32768/0/c4:ad34:1c:8d:33 Cost = 0 Port = 0x8001
8	8.008775747	Routerboard_ic18d...	Spanning-tree-(for...	STP	60	RST, Root = 32768/0/c4:ad34:1c:8d:33 Cost = 0 Port = 0x8001
9	10.010969309	Routerboard_ic18d...	Spanning-tree-(for...	STP	60	RST, Root = 32768/0/c4:ad34:1c:8d:33 Cost = 0 Port = 0x8001
10	12.013168309	Routerboard_ic18d...	Spanning-tree-(for...	STP	60	RST, Root = 32768/0/c4:ad34:1c:8d:33 Cost = 0 Port = 0x8001
11	14.015359338	Routerboard_ic18d...	Spanning-tree-(for...	STP	60	RST, Root = 32768/0/c4:ad34:1c:8d:33 Cost = 0 Port = 0x8001
12	16.017552471	Routerboard_ic18d...	Spanning-tree-(for...	STP	60	RST, Root = 32768/0/c4:ad34:1c:8d:33 Cost = 0 Port = 0x8001
13	18.019746932	Routerboard_ic18d...	Spanning-tree-(for...	STP	60	RST, Root = 32768/0/c4:ad34:1c:8d:33 Cost = 0 Port = 0x8001
14	20.021949793	Routerboard_ic18d...	Spanning-tree-(for...	STP	60	RST, Root = 32768/0/c4:ad34:1c:8d:33 Cost = 0 Port = 0x8001
15	22.024153592	Routerboard_ic18d...	Spanning-tree-(for...	STP	60	RST, Root = 32768/0/c4:ad34:1c:8d:33 Cost = 0 Port = 0x8001
16	24.026345050	Routerboard_ic18d...	Spanning-tree-(for...	STP	60	RST, Root = 32768/0/c4:ad34:1c:8d:33 Cost = 0 Port = 0x8001
17	26.028546704	Routerboard_ic18d...	Spanning-tree-(for...	STP	60	RST, Root = 32768/0/c4:ad34:1c:8d:33 Cost = 0 Port = 0x8001
18	28.030748444	Routerboard_ic18d...	Spanning-tree-(for...	STP	60	RST, Root = 32768/0/c4:ad34:1c:8d:33 Cost = 0 Port = 0x8001
19	30.032945473	Routerboard_ic18d...	Spanning-tree-(for...	STP	60	RST, Root = 32768/0/c4:ad34:1c:8d:33 Cost = 0 Port = 0x8001
20	32.035145171	Routerboard_ic18d...	Spanning-tree-(for...	STP	60	RST, Root = 32768/0/c4:ad34:1c:8d:33 Cost = 0 Port = 0x8001
21	34.037336978	Routerboard_ic18d...	Spanning-tree-(for...	STP	60	RST, Root = 32768/0/c4:ad34:1c:8d:33 Cost = 0 Port = 0x8001
22	36.039538363	Routerboard_ic18d...	Spanning-tree-(for...	STP	60	RST, Root = 32768/0/c4:ad34:1c:8d:33 Cost = 0 Port = 0x8001

Figure 6: Experiência 2 - TUX62 -¿ TUX63

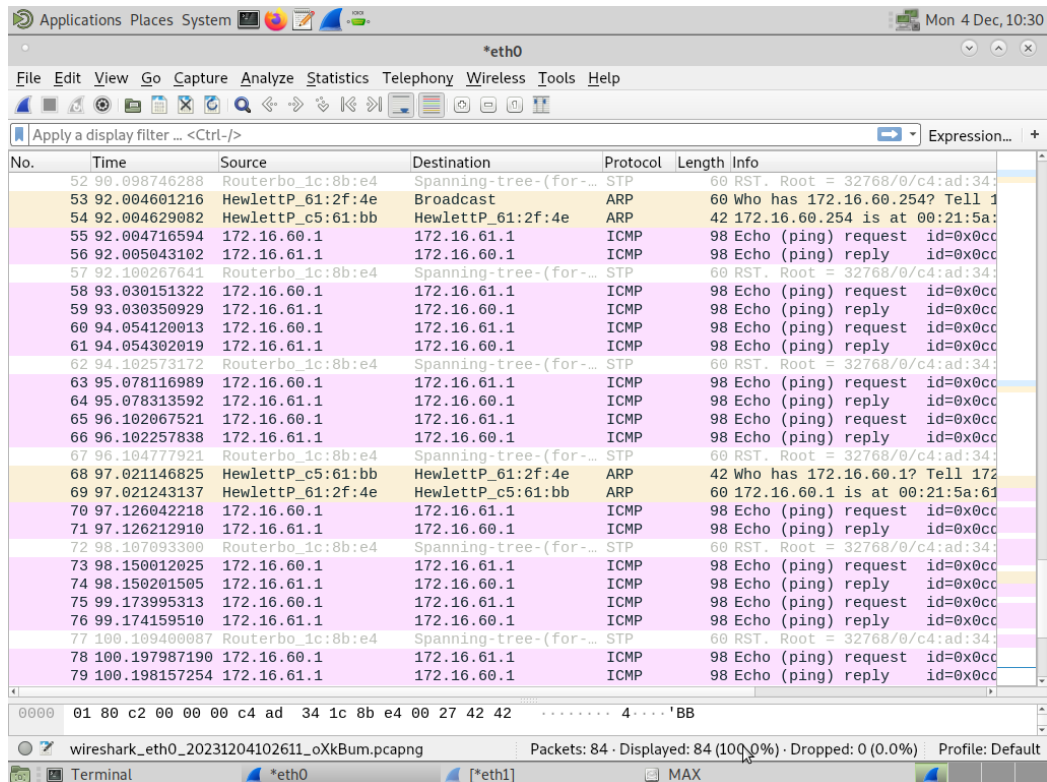
2	2.002184182	Routerboard_ic18d...	Spanning-tree-(for...	STP	60	RST, Root = 32768/0/c4:ad34:1c:8d:33 Cost = 0 Port = 0x8002
3	4.004383031	Routerboard_ic18d...	Spanning-tree-(for...	STP	60	RST, Root = 32768/0/c4:ad34:1c:8d:33 Cost = 0 Port = 0x8002
4	6.006577842	Routerboard_ic18d...	Spanning-tree-(for...	STP	60	RST, Root = 32768/0/c4:ad34:1c:8d:33 Cost = 0 Port = 0x8002
5	8.008774037	Routerboard_ic18d...	Spanning-tree-(for...	STP	60	RST, Root = 32768/0/c4:ad34:1c:8d:33 Cost = 0 Port = 0x8002
6	10.009974911	Routerboard_ic18d...	Spanning-tree-(for...	STP	60	RST, Root = 32768/0/c4:ad34:1c:8d:33 Cost = 0 Port = 0x8002
7	11.543794503	0.0.0.0	255.255.255.255	RNDP	159	5678 + 5678 Len=117
8	11.543825373	Routerboard_ic18d...	CDP/VTP/DTP/PagP/UD...	CDP	93	Device ID: MikroTik Port ID: bridge60
9	11.543872167	Routerboard_ic18d...	LDP, Multicast	LDP	110	PA/c4:ad34:1c:8d:33 10/bridge60 120 SysD=MikroTik SysD=MikroTik RouterOS 6.43.16 (long-term) CRS326-24G-25+
10	12.012158674	Routerboard_ic18d...	Spanning-tree-(for...	STP	60	RST, Root = 32768/0/c4:ad34:1c:8d:33 Cost = 0 Port = 0x8002
11	14.014367650	Routerboard_ic18d...	Spanning-tree-(for...	STP	60	RST, Root = 32768/0/c4:ad34:1c:8d:33 Cost = 0 Port = 0x8002
12	16.016560353	Routerboard_ic18d...	Spanning-tree-(for...	STP	60	RST, Root = 32768/0/c4:ad34:1c:8d:33 Cost = 0 Port = 0x8002
13	18.018754103	Routerboard_ic18d...	Spanning-tree-(for...	STP	60	RST, Root = 32768/0/c4:ad34:1c:8d:33 Cost = 0 Port = 0x8002
14	20.020951695	Routerboard_ic18d...	Spanning-tree-(for...	STP	60	RST, Root = 32768/0/c4:ad34:1c:8d:33 Cost = 0 Port = 0x8002
15	22.023145096	Routerboard_ic18d...	Spanning-tree-(for...	STP	60	RST, Root = 32768/0/c4:ad34:1c:8d:33 Cost = 0 Port = 0x8002

Figure 7: Experiência 2 - TUX62 -¿ TUX64

13	16.675972895	172.16.61.1	172.16.61.255	ICMP	98	Echo (ping) request id=0x41ea, seq=1/256, ttl=64 (no response found!)
14	17.693559045	172.16.61.1	172.16.61.255	ICMP	98	Echo (ping) request id=0x41ea, seq=2/512, ttl=64 (no response found!)
15	18.488228647	Routerboard_ic18d...	Spanning-tree-(for...	STP	60	RST, Root = 32768/0/c4:ad34:1c:8d:2b Cost = 0 Port = 0x8001
16	18.717541072	172.16.61.1	172.16.61.255	ICMP	98	Echo (ping) request id=0x41ea, seq=3/768, ttl=64 (no response found!)
17	19.741571918	172.16.61.1	172.16.61.255	ICMP	98	Echo (ping) request id=0x41ea, seq=4/1024, ttl=64 (no response found!)
18	20.490421093	Routerboard_ic18d...	Spanning-tree-(for...	STP	60	RST, Root = 32768/0/c4:ad34:1c:8d:2b Cost = 0 Port = 0x8001
19	20.765542072	172.16.61.1	172.16.61.255	ICMP	98	Echo (ping) request id=0x41ea, seq=5/1280, ttl=64 (no response found!)
20	21.789543236	172.16.61.1	172.16.61.255	ICMP	98	Echo (ping) request id=0x41ea, seq=6/1536, ttl=64 (no response found!)
21	22.492619685	Routerboard_ic18d...	Spanning-tree-(for...	STP	60	RST, Root = 32768/0/c4:ad34:1c:8d:2b Cost = 0 Port = 0x8001
22	22.813540558	172.16.61.1	172.16.61.255	ICMP	98	Echo (ping) request id=0x41ea, seq=7/1792, ttl=64 (no response found!)
23	23.837551988	172.16.61.1	172.16.61.255	ICMP	98	Echo (ping) request id=0x41ea, seq=8/2048, ttl=64 (no response found!)
24	24.494822677	Routerboard_ic18d...	Spanning-tree-(for...	STP	60	RST, Root = 32768/0/c4:ad34:1c:8d:2b Cost = 0 Port = 0x8001
25	24.861577177	172.16.61.1	172.16.61.255	ICMP	98	Echo (ping) request id=0x41ea, seq=9/2304, ttl=64 (no response found!)
26	25.885550544	172.16.61.1	172.16.61.255	ICMP	98	Echo (ping) request id=0x41ea, seq=10/2560, ttl=64 (no response found!)
27	26.4597013243	Routerboard_ic18d...	Spanning-tree-(for...	STP	60	RST, Root = 32768/0/c4:ad34:1c:8d:2b Cost = 0 Port = 0x8001
28	26.909551568	172.16.61.1	172.16.61.255	ICMP	98	Echo (ping) request id=0x41ea, seq=11/2816, ttl=64 (no response found!)

Figure 8: Experiência 2 - TUX62 -¿ TUX62

### 5.2.3 Experiência 3

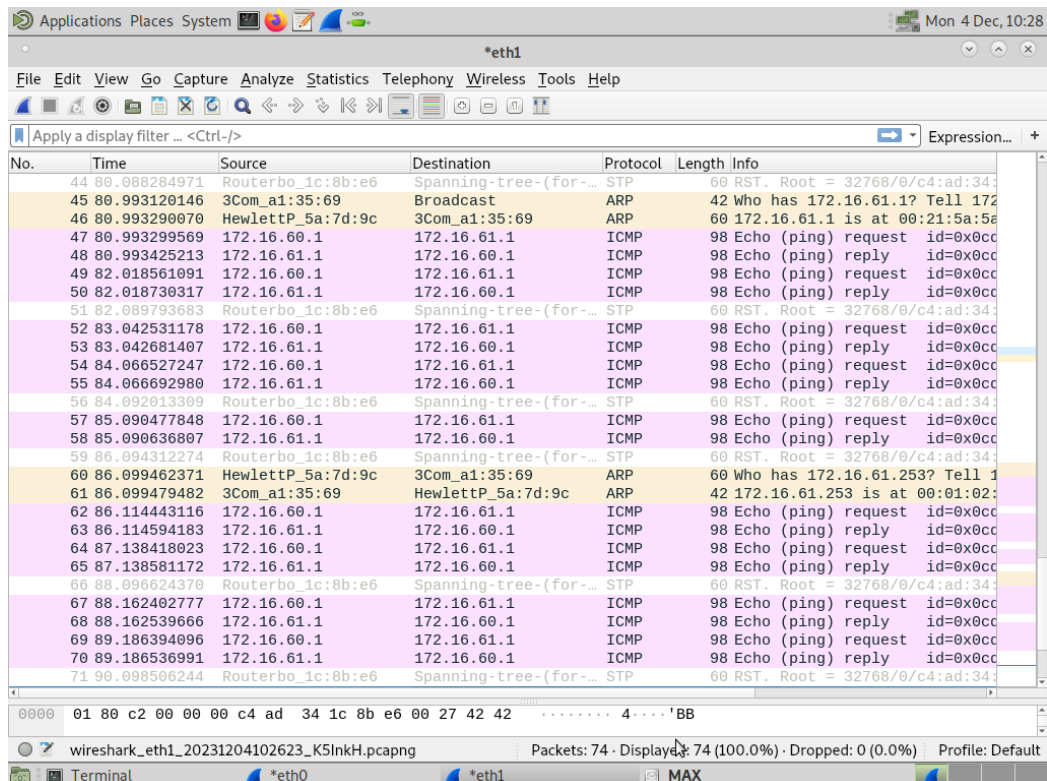


No.	Time	Source	Destination	Protocol	Length	Info
52	90.098746288	Routerbo_1c:8b:e4	Spanning-tree-(for...	STP	60	RST. Root = 32768/0/c4:ad:34:
53	92.004601216	HewlettP_61:2f:4e	Broadcast	ARP	60	Who has 172.16.60.254? Tell 1
54	92.004629082	HewlettP_c5:61:bb	HewlettP_61:2f:4e	ARP	42	172.16.60.254 is at 00:21:5a:
55	92.004716594	172.16.60.1	172.16.61.1	ICMP	98	Echo (ping) request id=0x0cc
56	92.005043102	172.16.61.1	172.16.60.1	ICMP	98	Echo (ping) reply id=0x0cc
57	92.100267641	Routerbo_1c:8b:e4	Spanning-tree-(for...	STP	60	RST. Root = 32768/0/c4:ad:34:
58	93.030151322	172.16.60.1	172.16.61.1	ICMP	98	Echo (ping) request id=0x0cc
59	93.030350929	172.16.61.1	172.16.60.1	ICMP	98	Echo (ping) reply id=0x0cc
60	94.054120013	172.16.60.1	172.16.61.1	ICMP	98	Echo (ping) request id=0x0cc
61	94.054302019	172.16.61.1	172.16.60.1	ICMP	98	Echo (ping) reply id=0x0cc
62	94.102573172	Routerbo_1c:8b:e4	Spanning-tree-(for...	STP	60	RST. Root = 32768/0/c4:ad:34:
63	95.078116989	172.16.60.1	172.16.61.1	ICMP	98	Echo (ping) request id=0x0cc
64	95.078313592	172.16.61.1	172.16.60.1	ICMP	98	Echo (ping) reply id=0x0cc
65	96.102067521	172.16.60.1	172.16.61.1	ICMP	98	Echo (ping) request id=0x0cc
66	96.102257838	172.16.61.1	172.16.60.1	ICMP	98	Echo (ping) reply id=0x0cc
67	96.104777921	Routerbo_1c:8b:e4	Spanning-tree-(for...	STP	60	RST. Root = 32768/0/c4:ad:34:
68	97.021146825	HewlettP_c5:61:bb	HewlettP_61:2f:4e	ARP	42	Who has 172.16.60.1? Tell 172
69	97.021243137	HewlettP_61:2f:4e	HewlettP_c5:61:bb	ARP	60	172.16.60.1 is at 00:21:5a:61
70	97.126042218	172.16.60.1	172.16.61.1	ICMP	98	Echo (ping) request id=0x0cc
71	97.126212910	172.16.61.1	172.16.60.1	ICMP	98	Echo (ping) reply id=0x0cc
72	98.107093300	Routerbo_1c:8b:e4	Spanning-tree-(for...	STP	60	RST. Root = 32768/0/c4:ad:34:
73	98.150012025	172.16.60.1	172.16.61.1	ICMP	98	Echo (ping) request id=0x0cc
74	98.150201505	172.16.61.1	172.16.60.1	ICMP	98	Echo (ping) reply id=0x0cc
75	99.173995313	172.16.60.1	172.16.61.1	ICMP	98	Echo (ping) request id=0x0cc
76	99.174159510	172.16.61.1	172.16.60.1	ICMP	98	Echo (ping) reply id=0x0cc
77	100.109400087	Routerbo_1c:8b:e4	Spanning-tree-(for...	STP	60	RST. Root = 32768/0/c4:ad:34:
78	100.197987190	172.16.60.1	172.16.61.1	ICMP	98	Echo (ping) request id=0x0cc
79	100.198157254	172.16.61.1	172.16.60.1	ICMP	98	Echo (ping) reply id=0x0cc

0000 01 80 c2 00 00 00 c4 ad 34 1c 8b e4 00 27 42 42 ..... 4....'BB

wireshark\_eth0\_20231204102611\_oXkBum.pcapng Packets: 84 · Displayed: 84 (100.0%) · Dropped: 0 (0.0%) Profile: Default

Figure 9: Experiência 2 - TUX62 -> TUX62



No.	Time	Source	Destination	Protocol	Length	Info
44	80.088284971	Routerbo_1c:8b:e6	Spanning-tree-(for...	STP	60	RST. Root = 32768/0/c4:ad:34:
45	80.993120146	3Com_a1:35:69	Broadcast	ARP	42	Who has 172.16.61.1? Tell 172
46	80.993290070	HewlettP_5a:7d:9c	3Com_a1:35:69	ARP	60	172.16.61.1 is at 00:21:5a:5a
47	80.993299569	172.16.60.1	172.16.61.1	ICMP	98	Echo (ping) request id=0x0cc
48	80.993425213	172.16.61.1	172.16.60.1	ICMP	98	Echo (ping) reply id=0x0cc
49	82.018561091	172.16.60.1	172.16.61.1	ICMP	98	Echo (ping) request id=0x0cc
50	82.018730317	172.16.61.1	172.16.60.1	ICMP	98	Echo (ping) reply id=0x0cc
51	82.089793683	Routerbo_1c:8b:e6	Spanning-tree-(for...	STP	60	RST. Root = 32768/0/c4:ad:34:
52	83.042531178	172.16.60.1	172.16.61.1	ICMP	98	Echo (ping) request id=0x0cc
53	83.042681407	172.16.61.1	172.16.60.1	ICMP	98	Echo (ping) reply id=0x0cc
54	84.066527247	172.16.60.1	172.16.61.1	ICMP	98	Echo (ping) request id=0x0cc
55	84.066692980	172.16.61.1	172.16.60.1	ICMP	98	Echo (ping) reply id=0x0cc
56	84.092013309	Routerbo_1c:8b:e6	Spanning-tree-(for...	STP	60	RST. Root = 32768/0/c4:ad:34:
57	85.090477848	172.16.60.1	172.16.61.1	ICMP	98	Echo (ping) request id=0x0cc
58	85.090636807	172.16.61.1	172.16.60.1	ICMP	98	Echo (ping) reply id=0x0cc
59	86.094312274	Routerbo_1c:8b:e6	Spanning-tree-(for...	STP	60	RST. Root = 32768/0/c4:ad:34:
60	86.099462371	HewlettP_5a:7d:9c	3Com_a1:35:69	ARP	60	Who has 172.16.61.253? Tell 1
61	86.099479482	3Com_a1:35:69	HewlettP_5a:7d:9c	ARP	42	172.16.61.253 is at 00:01:02:
62	86.114443116	172.16.60.1	172.16.61.1	ICMP	98	Echo (ping) request id=0x0cc
63	86.114594183	172.16.61.1	172.16.60.1	ICMP	98	Echo (ping) reply id=0x0cc
64	87.138418023	172.16.60.1	172.16.61.1	ICMP	98	Echo (ping) request id=0x0cc
65	87.138581172	172.16.61.1	172.16.60.1	ICMP	98	Echo (ping) reply id=0x0cc
66	88.096624370	Routerbo_1c:8b:e6	Spanning-tree-(for...	STP	60	RST. Root = 32768/0/c4:ad:34:
67	88.162402777	172.16.60.1	172.16.61.1	ICMP	98	Echo (ping) request id=0x0cc
68	88.162539666	172.16.61.1	172.16.60.1	ICMP	98	Echo (ping) reply id=0x0cc
69	89.186394096	172.16.60.1	172.16.61.1	ICMP	98	Echo (ping) request id=0x0cc
70	89.186536991	172.16.61.1	172.16.60.1	ICMP	98	Echo (ping) reply id=0x0cc
71	90.098596244	Routerbo_1c:8b:e6	Spanning-tree-(for...	STP	60	RST. Root = 32768/0/c4:ad:34:

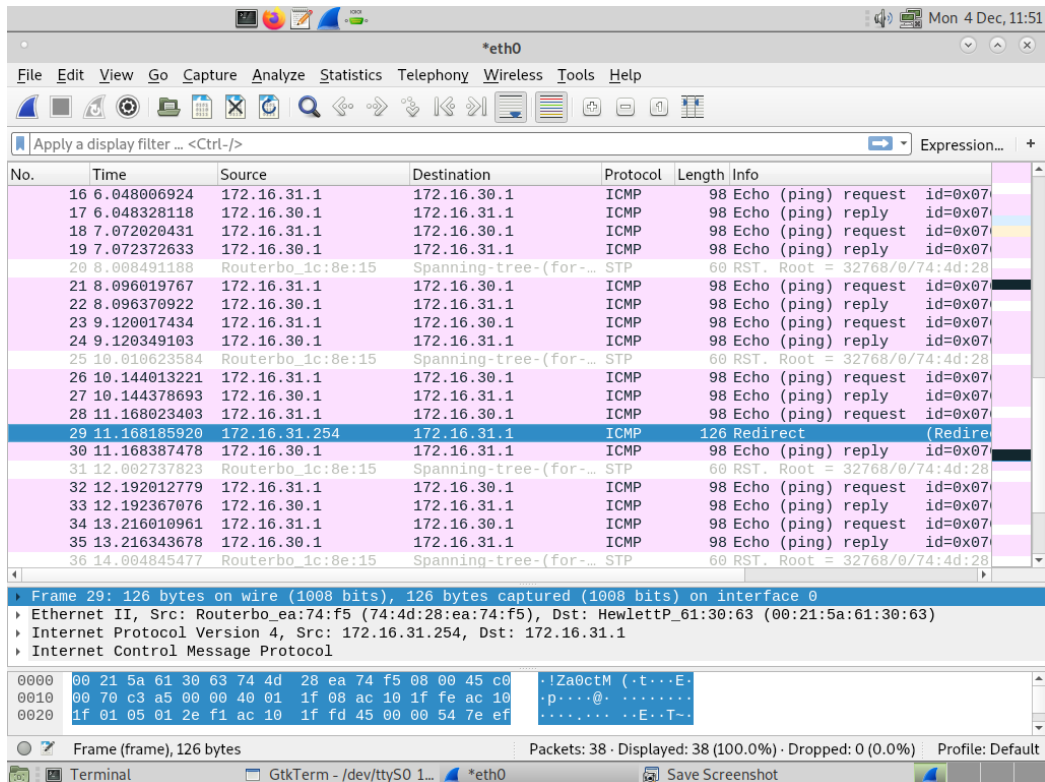
0000 01 80 c2 00 00 00 c4 ad 34 1c 8b e6 00 27 42 42 ..... 4....'BB

wireshark\_eth1\_20231204102623\_K5InkH.pcapng Packets: 74 · Displayed: 74 (100.0%) · Dropped: 0 (0.0%) Profile: Default

Figure 10: Experiência 2 - TUX62 -> TUX62



## 5.2.4 Experiência 4



No.	Time	Source	Destination	Protocol	Length	Info
16	6.048006924	172.16.31.1	172.16.31.1	ICMP	98	Echo (ping) request id=0x07
17	6.048328118	172.16.31.1	172.16.31.1	ICMP	98	Echo (ping) reply id=0x07
18	7.072020431	172.16.31.1	172.16.31.1	ICMP	98	Echo (ping) request id=0x07
19	7.072372633	172.16.31.1	172.16.31.1	ICMP	98	Echo (ping) reply id=0x07
20	8.008491188	Routerbo_1c:8e:15	Spanning-tree-(for...	STP	60	RST. Root = 32768/0/74:4d:28
21	8.096019767	172.16.31.1	172.16.31.1	ICMP	98	Echo (ping) request id=0x07
22	8.096370922	172.16.31.1	172.16.31.1	ICMP	98	Echo (ping) reply id=0x07
23	9.120017434	172.16.31.1	172.16.31.1	ICMP	98	Echo (ping) request id=0x07
24	9.120349103	172.16.31.1	172.16.31.1	ICMP	98	Echo (ping) reply id=0x07
25	10.010623584	Routerbo_1c:8e:15	Spanning-tree-(for...	STP	60	RST. Root = 32768/0/74:4d:28
26	10.144013221	172.16.31.1	172.16.31.1	ICMP	98	Echo (ping) request id=0x07
27	10.144378693	172.16.31.1	172.16.31.1	ICMP	98	Echo (ping) reply id=0x07
28	11.168023403	172.16.31.1	172.16.31.1	ICMP	98	Echo (ping) request id=0x07
29	11.168185920	172.16.31.254	172.16.31.1	ICMP	126	Redirect (Redirect to 172.16.31.1)
30	11.168387478	172.16.31.1	172.16.31.1	ICMP	98	Echo (ping) reply id=0x07
31	12.002737823	Routerbo_1c:8e:15	Spanning-tree-(for...	STP	60	RST. Root = 32768/0/74:4d:28
32	12.192012779	172.16.31.1	172.16.31.1	ICMP	98	Echo (ping) request id=0x07
33	12.192367076	172.16.31.1	172.16.31.1	ICMP	98	Echo (ping) reply id=0x07
34	13.216010961	172.16.31.1	172.16.31.1	ICMP	98	Echo (ping) request id=0x07
35	13.216343678	172.16.31.1	172.16.31.1	ICMP	98	Echo (ping) reply id=0x07
36	14.004845477	Routerbo_1c:8e:15	Spanning-tree-(for...	STP	60	RST. Root = 32768/0/74:4d:28

Frame 29: 126 bytes on wire (1008 bits), 126 bytes captured (1008 bits) on interface 0

Ethernet II, Src: Routerbo\_ea:74:f5 (74:4d:28:ea:74:f5), Dst: HewlettP\_61:30:63 (00:21:5a:61:30:63)

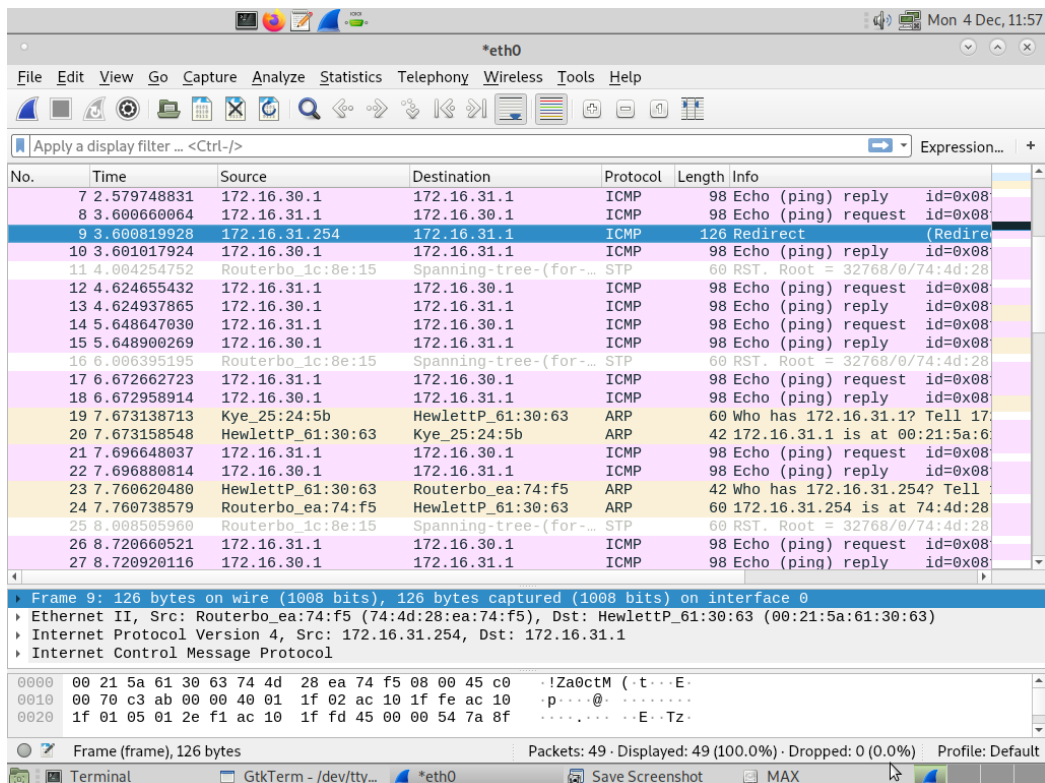
Internet Protocol Version 4, Src: 172.16.31.254, Dst: 172.16.31.1

Internet Control Message Protocol

0000 00 21 5a 61 30 63 74 4d 28 ea 74 f5 08 00 45 c0 !ZaOctM (.t...E-  
0010 00 70 c3 a5 00 00 40 01 1f 08 ac 10 1f fe ac 10 p...@.....  
0020 1f 01 05 01 2e f1 ac 10 1f fd 45 00 00 54 7e ef .....-E..Tz..

Frame (frame), 126 bytes Packets: 38 · Displayed: 38 (100.0%) · Dropped: 0 (0.0%) Profile: Default

Figure 11: Experiência 4 - TUX62 -> TUX63 (redirects desativados)



No.	Time	Source	Destination	Protocol	Length	Info
7	2.579748831	172.16.31.1	172.16.31.1	ICMP	98	Echo (ping) reply id=0x08
8	3.600660064	172.16.31.1	172.16.31.1	ICMP	98	Echo (ping) request id=0x08
9	3.600819928	172.16.31.254	172.16.31.1	ICMP	126	Redirect (Redirect to 172.16.31.1)
10	3.601017924	172.16.31.1	172.16.31.1	ICMP	98	Echo (ping) reply id=0x08
11	4.004254752	Routerbo_1c:8e:15	Spanning-tree-(for...	STP	60	RST. Root = 32768/0/74:4d:28
12	4.624655432	172.16.31.1	172.16.31.1	ICMP	98	Echo (ping) request id=0x08
13	4.624937865	172.16.31.1	172.16.31.1	ICMP	98	Echo (ping) reply id=0x08
14	5.648647030	172.16.31.1	172.16.31.1	ICMP	98	Echo (ping) request id=0x08
15	5.648900269	172.16.31.1	172.16.31.1	ICMP	98	Echo (ping) reply id=0x08
16	6.006395195	Routerbo_1c:8e:15	Spanning-tree-(for...	STP	60	RST. Root = 32768/0/74:4d:28
17	6.672662723	172.16.31.1	172.16.31.1	ICMP	98	Echo (ping) request id=0x08
18	6.672958914	172.16.31.1	172.16.31.1	ICMP	98	Echo (ping) reply id=0x08
19	7.673138713	Kye_25:24:5b	HewlettP_61:30:63	ARP	60	Who has 172.16.31.1? Tell 172.16.31.1
20	7.673158548	HewlettP_61:30:63	Kye_25:24:5b	ARP	42	172.16.31.1 is at 00:21:5a:61:30:63
21	7.696648037	172.16.31.1	172.16.31.1	ICMP	98	Echo (ping) request id=0x08
22	7.696880814	172.16.31.1	172.16.31.1	ICMP	98	Echo (ping) reply id=0x08
23	7.760620480	HewlettP_61:30:63	Routerbo_ea:74:f5	ARP	42	Who has 172.16.31.254? Tell 172.16.31.1
24	7.760738579	Routerbo_ea:74:f5	HewlettP_61:30:63	ARP	60	172.16.31.254 is at 74:4d:28:ea:74:f5
25	8.008505960	Routerbo_1c:8e:15	Spanning-tree-(for...	STP	60	RST. Root = 32768/0/74:4d:28
26	8.720660521	172.16.31.1	172.16.31.1	ICMP	98	Echo (ping) request id=0x08
27	8.720920116	172.16.31.1	172.16.31.1	ICMP	98	Echo (ping) reply id=0x08

Frame 9: 126 bytes on wire (1008 bits), 126 bytes captured (1008 bits) on interface 0

Ethernet II, Src: Routerbo\_ea:74:f5 (74:4d:28:ea:74:f5), Dst: HewlettP\_61:30:63 (00:21:5a:61:30:63)

Internet Protocol Version 4, Src: 172.16.31.254, Dst: 172.16.31.1

Internet Control Message Protocol

0000 00 21 5a 61 30 63 74 4d 28 ea 74 f5 08 00 45 c0 !ZaOctM (.t...E-  
0010 00 70 c3 ab 00 00 40 01 1f 02 ac 10 1f fe ac 10 p...@.....  
0020 1f 01 05 01 2e f1 ac 10 1f fd 45 00 00 54 7a 8f .....-E..Tz..

Frame (frame), 126 bytes Packets: 49 · Displayed: 49 (100.0%) · Dropped: 0 (0.0%) Profile: Default

Figure 12: Experiência 4 - TUX62 -> TUX63 (redirects ativados)

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	Routerbo 1c:8e:13	Spanning-tree-(for-bri...	STP	60	RST. Root = 32768/0/c4:ad:34:1c:8e:13
2	2.002146432	Routerbo 1c:8e:13	Spanning-tree-(for-bri...	STP	60	RST. Root = 32768/0/c4:ad:34:1c:8e:13
3	4.004314794	Routerbo 1c:8e:13	Spanning-tree-(for-bri...	STP	60	RST. Root = 32768/0/c4:ad:34:1c:8e:13
4	6.006466115	Routerbo 1c:8e:13	Spanning-tree-(for-bri...	STP	60	RST. Root = 32768/0/c4:ad:34:1c:8e:13
5	8.008622395	Routerbo 1c:8e:13	Spanning-tree-(for-bri...	STP	60	RST. Root = 32768/0/c4:ad:34:1c:8e:13
6	10.010794320	Routerbo 1c:8e:13	Spanning-tree-(for-bri...	STP	60	RST. Root = 32768/0/c4:ad:34:1c:8e:13
7	12.012916098	Routerbo 1c:8e:13	Spanning-tree-(for-bri...	STP	60	RST. Root = 32768/0/c4:ad:34:1c:8e:13
8	12.974552141	172.16.30.1	172.16.31.1	ICMP	98	Echo (ping) request id=0xe22, seq=1/25
9	12.974874878	172.16.31.1	172.16.30.1	ICMP	98	Echo (ping) reply id=0xe22, seq=1/25
10	14.001671372	172.16.30.1	172.16.31.1	ICMP	98	Echo (ping) request id=0xe22, seq=2/51
11	14.001960515	172.16.31.1	172.16.30.1	ICMP	98	Echo (ping) reply id=0xe22, seq=2/51
12	14.015067279	Routerbo 1c:8e:13	Spanning-tree-(for-bri...	STP	60	RST. Root = 32768/0/c4:ad:34:1c:8e:13
13	15.025671491	172.16.30.1	172.16.31.1	ICMP	98	Echo (ping) request id=0xe22, seq=3/76
14	15.025934094	172.16.31.1	172.16.30.1	ICMP	98	Echo (ping) reply id=0xe22, seq=3/76
15	16.017208419	Routerbo 1c:8e:13	Spanning-tree-(for-bri...	STP	60	RST. Root = 32768/0/c4:ad:34:1c:8e:13
16	16.049686677	172.16.30.1	172.16.31.1	ICMP	98	Echo (ping) request id=0xe22, seq=4/10
17	16.049948531	172.16.31.1	172.16.30.1	ICMP	98	Echo (ping) reply id=0xe22, seq=4/10
18	17.073672148	172.16.30.1	172.16.31.1	ICMP	98	Echo (ping) request id=0xe22, seq=5/12
19	17.073938802	172.16.31.1	172.16.30.1	ICMP	98	Echo (ping) reply id=0xe22, seq=5/12
20	18.019439274	Routerbo 1c:8e:13	Spanning-tree-(for-bri...	STP	60	RST. Root = 32768/0/c4:ad:34:1c:8e:13
21	18.036815715	HewlettP_5a:7d:74	HewlettP_61:24:92	ARP	42	Who has 172.16.30.1? Tell 172.16.30.254
22	18.036823957	HewlettP_61:24:92	HewlettP_5a:7d:74	ARP	42	172.16.30.1 is at 00:21:5a:61:24:92
23	18.097634902	HewlettP_61:24:92	HewlettP_5a:7d:74	ARP	42	Who has 172.16.30.254? Tell 172.16.30.1
24	18.097653619	172.16.30.1	172.16.31.1	ICMP	98	Echo (ping) request id=0xe22, seq=6/15
25	18.097775492	HewlettP_5a:7d:74	HewlettP_61:24:92	ARP	60	172.16.30.254 is at 00:21:5a:7d:74
26	18.097891918	172.16.31.1	172.16.30.1	ICMP	98	Echo (ping) reply id=0xe22, seq=6/15

Frame 1: 60 bytes on wire (480 bits), 60 bytes captured (480 bits) on interface 0  
 IEEE 802.3 Ethernet  
 Logical-Link Control  
 Spanning Tree Protocol

0000 01 80 c2 00 00 00 c4 ad 34 1c 8e 13 00 27 42 42 ..... 4.....BB  
 0010 03 00 00 02 02 3c 80 00 c4 ad 34 1c 8e 13 00 00 .....<...4.....

Ethernet (eth), 21 bytes      Packets: 28 · Displayed: 28 (100.0%)      Profile: Default

Figure 13: Experiência 4 - TUX63 -> TUX62 (redirects desativados)


No.	Time	Source	Destination	Protocol	Length	Info
17	30.677934324	172.16.30.1	172.16.30.254	ICMP	98	Echo (ping) request id=0xc7f, s
18	30.678106064	172.16.30.254	172.16.30.1	ICMP	98	Echo (ping) reply id=0xc7f, s
19	31.681127817	172.16.30.1	172.16.30.254	ICMP	98	Echo (ping) request id=0xc7f, s
20	31.681310801	172.16.30.254	172.16.30.1	ICMP	98	Echo (ping) reply id=0xc7f, s
21	32.024095243	Routerbo 1c:8e:13	Spanning-tree-(for-bri...	STP	60	RST. Root = 32768/0/c4:ad:34:1c:8
22	32.705127447	172.16.30.1	172.16.30.254	ICMP	98	Echo (ping) request id=0xc7f, s
23	32.705275650	172.16.30.254	172.16.30.1	ICMP	98	Echo (ping) reply id=0xc7f, s
24	33.729141395	172.16.30.1	172.16.30.254	ICMP	98	Echo (ping) request id=0xc7f, s
25	33.729292881	172.16.30.254	172.16.30.1	ICMP	98	Echo (ping) reply id=0xc7f, s
26	34.026246914	Routerbo 1c:8e:13	Spanning-tree-(for-bri...	STP	60	RST. Root = 32768/0/c4:ad:34:1c:8
27	34.753143400	172.16.30.1	172.16.30.254	ICMP	98	Echo (ping) request id=0xc7f, s
28	34.753291323	172.16.30.254	172.16.30.1	ICMP	98	Echo (ping) reply id=0xc7f, s
29	35.777141982	172.16.30.1	172.16.30.254	ICMP	98	Echo (ping) request id=0xc7f, s
30	35.777291163	172.16.30.254	172.16.30.1	ICMP	98	Echo (ping) reply id=0xc7f, s
31	35.819441844	HewlettP_5a:7d:74	HewlettP_61:24:92	ARP	60	Who has 172.16.30.1? Tell 172.16.
32	35.819448898	HewlettP_61:24:92	HewlettP_5a:7d:74	ARP	42	172.16.30.1 is at 00:21:5a:61:24:
33	35.873095874	HewlettP_61:24:92	HewlettP_5a:7d:74	ARP	42	Who has 172.16.30.254? Tell 172.1
34	35.873220680	HewlettP_5a:7d:74	HewlettP_61:24:92	ARP	60	172.16.30.254 is at 00:21:5a:7d:7
35	36.029412301	Routerbo 1c:8e:13	Spanning-tree-(for-bri...	STP	60	RST. Root = 32768/0/c4:ad:34:1c:8
36	36.801134488	172.16.30.1	172.16.30.254	ICMP	98	Echo (ping) request id=0xc7f, s
37	36.801299454	172.16.30.254	172.16.30.1	ICMP	98	Echo (ping) reply id=0xc7f, s
38	37.825133141	172.16.30.1	172.16.30.254	ICMP	98	Echo (ping) request id=0xc7f, s
39	37.825282462	172.16.30.254	172.16.30.1	ICMP	98	Echo (ping) reply id=0xc7f, s
40	38.030564502	Routerbo 1c:8e:13	Spanning-tree-(for-bri...	STP	60	RST. Root = 32768/0/c4:ad:34:1c:8
41	38.849131793	172.16.30.1	172.16.30.254	ICMP	98	Echo (ping) request id=0xc7f, s
42	38.849282022	172.16.30.254	172.16.30.1	ICMP	98	Echo (ping) reply id=0xc7f, s
43	39.873132052	172.16.30.1	172.16.30.254	ICMP	98	Echo (ping) request id=0xc7f, s
44	39.873275366	172.16.30.254	172.16.30.1	ICMP	98	Echo (ping) reply id=0xc7f, s
45	40.032729442	Routerbo 1c:8e:13	Spanning-tree-(for-bri...	STP	60	RST. Root = 32768/0/c4:ad:34:1c:8

Frame 45: 60 bytes on wire (480 bits), 60 bytes captured (480 bits) on interface 0  
 IEEE 802.3 Ethernet  
 Logical-Link Control  
 Spanning Tree Protocol

0000 01 80 c2 00 00 00 c4 ad 34 1c 8e 13 00 27 42 42 ..... 4.....BB  
 0010 03 00 00 02 02 3c 80 00 c4 ad 34 1c 8e 13 00 00 .....<...4.....

Ethernet (eth), 21 bytes      Packets: 65 · Displayed: 65 (100.0%) · Dropped: 0 (0.0%)      Profile: Default

Figure 14: Experiência 4 - TUX63 -> TUX64 (redirects desativados)

Applications Places System  Mon 4 Dec, 11:29

Capturing from eth0

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

Apply a display filter ... <Ctrl-/> Expression...


No.	Time	Source	Destination	Protocol	Length	Info
96	69.075597604	Routerbo_1c:8e:13	Spanning-tree-(for-bri...	STP	60	RST. Root = 32768/0/c4:ad:34:1c:8e:13
97	71.077714564	Routerbo_1c:8e:13	Spanning-tree-(for-bri...	STP	60	RST. Root = 32768/0/c4:ad:34:1c:8e:13
98	72.380872179	172.16.30.1	172.16.31.254	ICMP	98	Echo (ping) request id=0x0e95, seq=1/
99	72.381236122	172.16.31.254	172.16.30.1	ICMP	98	Echo (ping) reply id=0x0e95, seq=1/
100	73.069791489	Routerbo_1c:8e:13	Spanning-tree-(for-bri...	STP	60	RST. Root = 32768/0/c4:ad:34:1c:8e:13
101	73.412056242	172.16.30.1	172.16.31.254	ICMP	98	Echo (ping) request id=0x0e95, seq=2/
102	73.412350833	172.16.31.254	172.16.30.1	ICMP	98	Echo (ping) reply id=0x0e95, seq=2/
103	74.432017719	172.16.30.1	172.16.31.254	ICMP	98	Echo (ping) request id=0x0e95, seq=3/
104	74.432318037	172.16.31.254	172.16.30.1	ICMP	98	Echo (ping) reply id=0x0e95, seq=3/
105	75.071961528	Routerbo_1c:8e:13	Spanning-tree-(for-bri...	STP	60	RST. Root = 32768/0/c4:ad:34:1c:8e:13
106	75.456000448	172.16.30.1	172.16.31.254	ICMP	98	Echo (ping) request id=0x0e95, seq=4/
107	75.456279743	172.16.31.254	172.16.30.1	ICMP	98	Echo (ping) reply id=0x0e95, seq=4/
108	76.480000706	172.16.30.1	172.16.31.254	ICMP	98	Echo (ping) request id=0x0e95, seq=5/
109	76.480271970	172.16.31.254	172.16.30.1	ICMP	98	Echo (ping) reply id=0x0e95, seq=5/
110	77.074086239	Routerbo_1c:8e:13	Spanning-tree-(for-bri...	STP	60	RST. Root = 32768/0/c4:ad:34:1c:8e:13
111	77.442651474	HewlettP_5a:7d:74	HewlettP_61:24:92	ARP	60	Who has 172.16.30.1? Tell 172.16.30.25
112	77.442672775	HewlettP_61:24:92	HewlettP_5a:7d:74	ARP	42	172.16.30.1 is at 00:21:5a:61:24:92
113	77.503956336	HewlettP_61:24:92	HewlettP_5a:7d:74	ARP	42	Who has 172.16.30.254? Tell 172.16.30.
114	77.504001803	172.16.30.1	172.16.31.254	ICMP	98	Echo (ping) request id=0x0e95, seq=6/
115	77.504085333	HewlettP_5a:7d:74	HewlettP_61:24:92	ARP	60	172.16.30.254 is at 00:21:5a:5a:7d:74
116	77.504245549	172.16.31.254	172.16.30.1	ICMP	98	Echo (ping) reply id=0x0e95, seq=6/
117	78.527997662	172.16.30.1	172.16.31.254	ICMP	98	Echo (ping) request id=0x0e95, seq=7/
118	78.528293859	172.16.31.254	172.16.30.1	ICMP	98	Echo (ping) reply id=0x0e95, seq=7/
119	79.076254462	Routerbo_1c:8e:13	Spanning-tree-(for-bri...	STP	60	RST. Root = 32768/0/c4:ad:34:1c:8e:13
120	79.551998270	172.16.30.1	172.16.31.254	ICMP	98	Echo (ping) request id=0x0e95, seq=8/
121	79.552276587	172.16.31.254	172.16.30.1	ICMP	98	Echo (ping) reply id=0x0e95, seq=8/
122	80.575995944	172.16.30.1	172.16.31.254	ICMP	98	Echo (ping) request id=0x0e95, seq=9/
123	80.576273563	172.16.31.254	172.16.30.1	ICMP	98	Echo (ping) reply id=0x0e95, seq=9/
124	81.078400405	Routerbo_1c:8e:13	Spanning-tree-(for-bri...	STP	60	RST. Root = 32768/0/c4:ad:34:1c:8e:13

Frame 99: 98 bytes on wire (784 bits), 98 bytes captured (784 bits) on interface 0  
 Ethernet II, Src: HewlettP\_5a:7d:74 (00:21:5a:5a:7d:74), Dst: HewlettP\_61:24:92 (00:21:5a:61:24:92)  
 Internet Protocol Version 4, Src: 172.16.31.254, Dst: 172.16.30.1  
 Internet Control Message Protocol

0000 00 21 5a 61 24 92 00 21 5a 5a 7d 74 08 00 45 00 ..!Za\$...! ZZ)t...E..  
 0010 00 54 fe 1e 00 00 3f 01 e7 6a ac 10 1f fe ac 10 .T....?..j.....

Ethernet (eth), 14 bytes Packets: 124 · Displayed: 124 (100.0%) Profile: Default

Figure 15: Experiência 4 - TUX63 -> TUX64 (redirects ativados)

Applications Places System  Mon 4 Dec, 12:01

Capturing from eth0

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

Apply a display filter ... <Ctrl-/> Expression...

No.	Time	Source	Destination	Protocol	Length	Info
5	3.994337019	Routerbo_1c:8e:13	Spanning-tree-(for-bri...	STP	60	RST. Root = 32768/0/c4:ad:34:1c:8e:13
6	4.021724599	172.16.30.1	172.16.1.254	ICMP	98	Echo (ping) request id=0x1389, seq=2/
7	4.022178849	172.16.1.254	172.16.30.1	ICMP	98	Echo (ping) reply id=0x1389, seq=2/
8	5.045733937	172.16.30.1	172.16.1.254	ICMP	98	Echo (ping) request id=0x1389, seq=3/
9	5.046190003	172.16.1.254	172.16.30.1	ICMP	98	Echo (ping) reply id=0x1389, seq=3/
10	5.996516360	Routerbo_1c:8e:13	Spanning-tree-(for-bri...	STP	60	RST. Root = 32768/0/c4:ad:34:1c:8e:13
11	6.069721904	172.16.30.1	172.16.1.254	ICMP	98	Echo (ping) request id=0x1389, seq=4/
12	6.070155411	172.16.1.254	172.16.30.1	ICMP	98	Echo (ping) reply id=0x1389, seq=4/
13	7.093728031	172.16.30.1	172.16.1.254	ICMP	98	Echo (ping) request id=0x1389, seq=5/
14	7.094153227	172.16.1.254	172.16.30.1	ICMP	98	Echo (ping) reply id=0x1389, seq=5/
15	7.998187048	Routerbo_1c:8e:13	Spanning-tree-(for-bri...	STP	60	RST. Root = 32768/0/c4:ad:34:1c:8e:13
16	8.121272468	172.16.30.1	172.16.1.254	ICMP	98	Echo (ping) request id=0x1389, seq=6/
17	8.122140890	172.16.1.254	172.16.30.1	ICMP	98	Echo (ping) reply id=0x1389, seq=6/
18	9.141725481	172.16.30.1	172.16.1.254	ICMP	98	Echo (ping) request id=0x1389, seq=7/
19	9.142136499	172.16.1.254	172.16.30.1	ICMP	98	Echo (ping) reply id=0x1389, seq=7/
20	10.000355151	Routerbo_1c:8e:13	Spanning-tree-(for-bri...	STP	60	RST. Root = 32768/0/c4:ad:34:1c:8e:13
21	10.165721273	172.16.30.1	172.16.1.254	ICMP	98	Echo (ping) request id=0x1389, seq=8/
22	10.166174616	172.16.1.254	172.16.30.1	ICMP	98	Echo (ping) reply id=0x1389, seq=8/
23	11.189723912	172.16.30.1	172.16.1.254	ICMP	98	Echo (ping) request id=0x1389, seq=9/
24	11.190166610	172.16.1.254	172.16.30.1	ICMP	98	Echo (ping) reply id=0x1389, seq=9/
25	12.002508522	Routerbo_1c:8e:13	Spanning-tree-(for-bri...	STP	60	RST. Root = 32768/0/c4:ad:34:1c:8e:13
26	12.213721313	172.16.30.1	172.16.1.254	ICMP	98	Echo (ping) request id=0x1389, seq=10/
27	12.214154471	172.16.1.254	172.16.30.1	ICMP	98	Echo (ping) reply id=0x1389, seq=10/
28	13.241723577	172.16.30.1	172.16.1.254	ICMP	98	Echo (ping) request id=0x1389, seq=11/
29	13.242156176	172.16.1.254	172.16.30.1	ICMP	98	Echo (ping) reply id=0x1389, seq=11/
30	13.994538203	Routerbo_1c:8e:13	Spanning-tree-(for-bri...	STP	60	RST. Root = 32768/0/c4:ad:34:1c:8e:13
31	14.261723381	172.16.30.1	172.16.1.254	ICMP	98	Echo (ping) request id=0x1389, seq=12/
32	14.262168691	172.16.1.254	172.16.30.1	ICMP	98	Echo (ping) reply id=0x1389, seq=12/

Frame 4: 98 bytes on wire (784 bits), 98 bytes captured (784 bits) on interface 0  
 Ethernet II, Src: HewlettP\_5a:7d:74 (00:21:5a:5a:7d:74), Dst: HewlettP\_61:24:92 (00:21:5a:61:24:92)  
 Internet Protocol Version 4, Src: 172.16.1.254, Dst: 172.16.30.1  
 Internet Control Message Protocol

0000 00 21 5a 61 24 92 00 21 5a 5a 7d 74 08 00 45 00 ..!Za\$...! ZZ)t...E..  
 0010 00 54 90 6a 00 00 3e 01 74 1f ac 10 01 fe ac 10 .T.j...: t.....

eth0: <live capture in progress> Packets: 32 · Displayed: 32 (100.0%) Profile: Default

Figure 16: Experiência 4 - TUX63 -> TUX64 (redirects ativados)

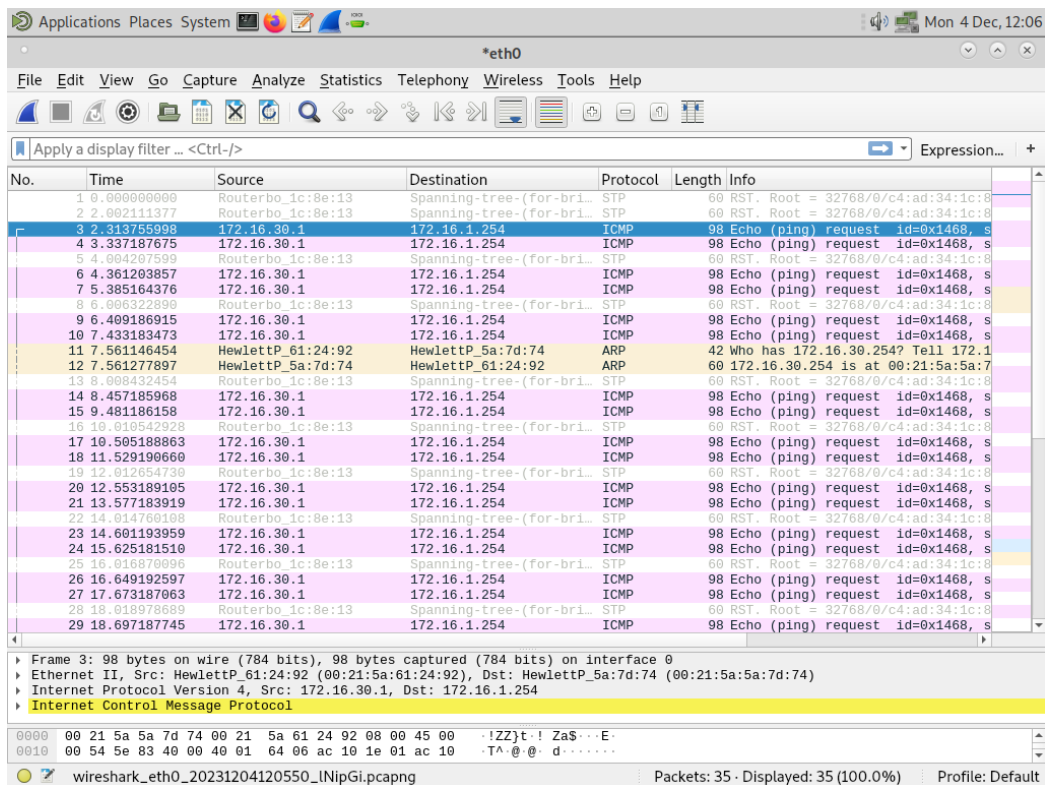


Figure 17: Experiência 4 - TUX63 -> TUX64 (redirects ativados)

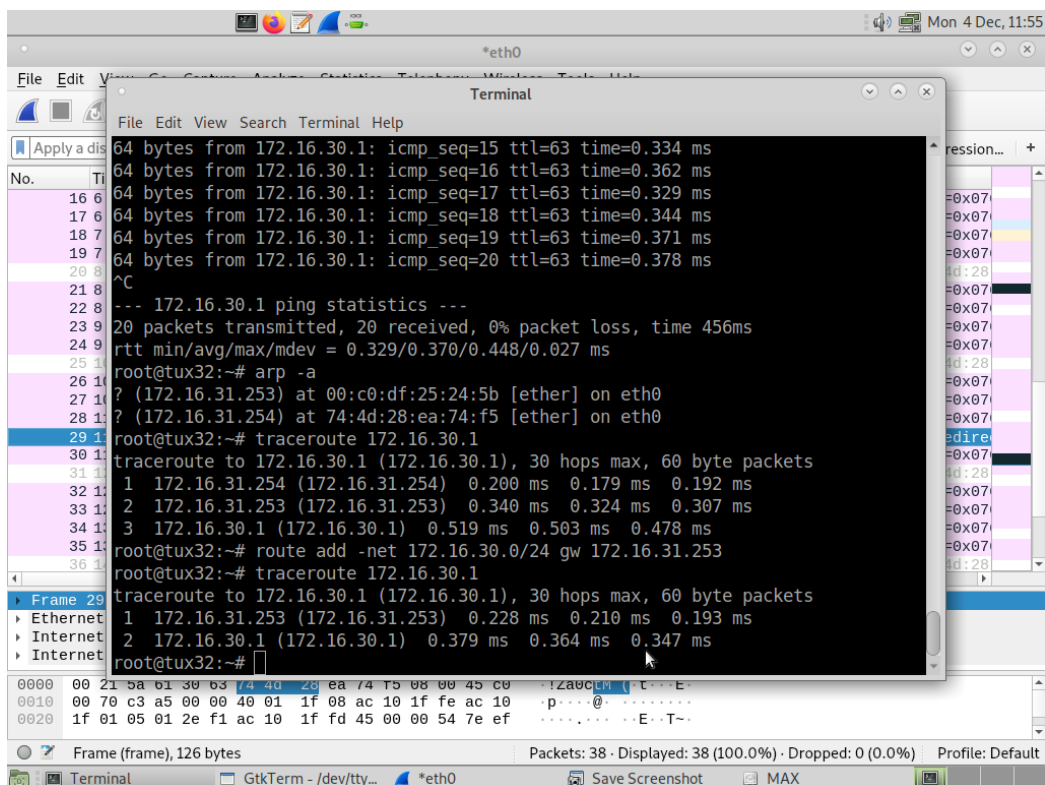
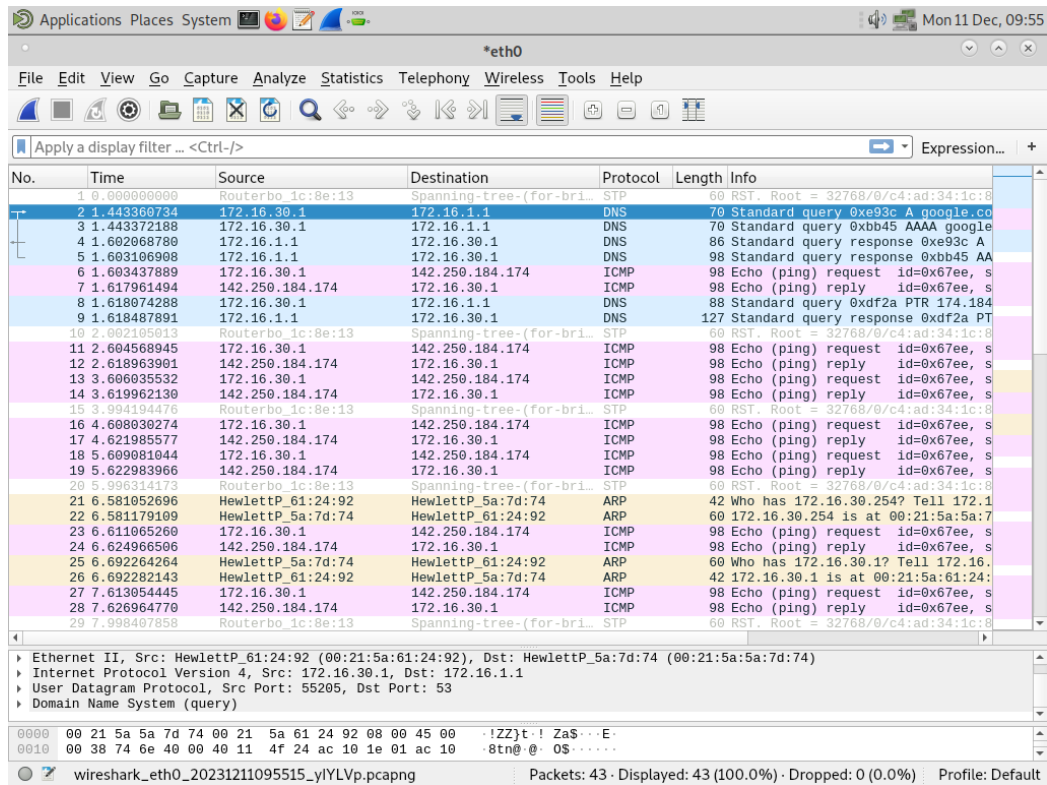


Figure 18: Experiência 4 - TUX63 -> TUX64 (redirects ativados)



## 5.2.5 Experiência 5



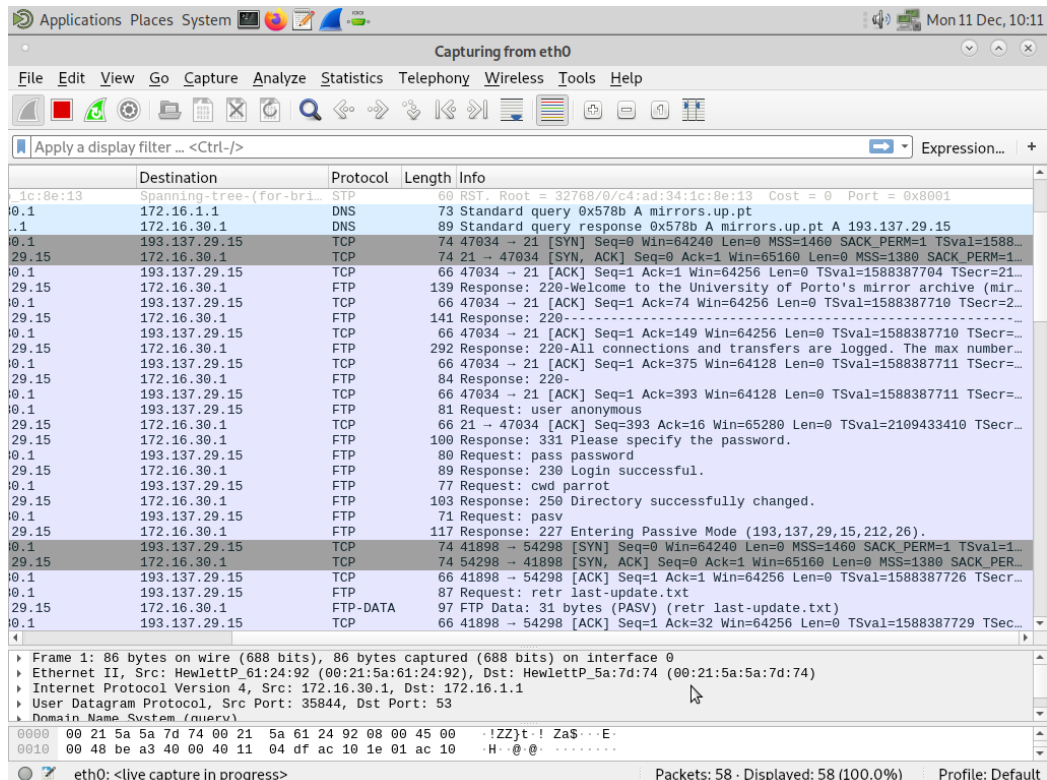
No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	Routerbo_1c:8e:13	Spanning-tree-(for-bri...	STP	60	RST. Root = 32768/0/c4:ad:34:1c:8
2	1.443360734	172.16.30.1	172.16.1.1	DNS	70	Standard query 0xe93c A google.co
3	1.443372188	172.16.30.1	172.16.1.1	DNS	70	Standard query 0xbb45 AAAA google
4	1.602968780	172.16.1.1	172.16.30.1	DNS	86	Standard query response 0xe93c A
5	1.603106908	172.16.1.1	172.16.30.1	DNS	98	Standard query response 0xbb45 AA
6	1.603437809	172.16.30.1	142.250.184.174	ICMP	98	Echo (ping) request id=0x67ee, s
7	1.617961494	142.250.184.174	172.16.30.1	ICMP	98	Echo (ping) reply id=0x67ee, s
8	1.618074288	172.16.30.1	172.16.1.1	DNS	88	Standard query 0xdf2a PTR 174.184
9	1.618487891	172.16.1.1	172.16.30.1	DNS	127	Standard query response 0xdf2a PT
10	2.002105013	Routerbo_1c:8e:13	Spanning-tree-(for-bri...	STP	60	RST. Root = 32768/0/c4:ad:34:1c:8
11	2.604568945	172.16.30.1	142.250.184.174	ICMP	98	Echo (ping) request id=0x67ee, s
12	2.618963901	142.250.184.174	172.16.30.1	ICMP	98	Echo (ping) reply id=0x67ee, s
13	3.606035532	172.16.30.1	142.250.184.174	ICMP	98	Echo (ping) request id=0x67ee, s
14	3.619962130	142.250.184.174	172.16.30.1	ICMP	98	Echo (ping) reply id=0x67ee, s
15	3.994194476	Routerbo_1c:8e:13	Spanning-tree-(for-bri...	STP	60	RST. Root = 32768/0/c4:ad:34:1c:8
16	4.608030274	172.16.30.1	142.250.184.174	ICMP	98	Echo (ping) request id=0x67ee, s
17	4.621985577	142.250.184.174	172.16.30.1	ICMP	98	Echo (ping) reply id=0x67ee, s
18	5.609081044	172.16.30.1	142.250.184.174	ICMP	98	Echo (ping) request id=0x67ee, s
19	5.622983966	142.250.184.174	172.16.30.1	ICMP	98	Echo (ping) reply id=0x67ee, s
20	5.996314173	Routerbo_1c:8e:13	Spanning-tree-(for-bri...	STP	60	RST. Root = 32768/0/c4:ad:34:1c:8
21	6.581052696	HewlettP_61:24:92	HewlettP_5a:7d:74	ARP	42	Who has 172.16.30.254? Tell 172.1
22	6.581179109	HewlettP_5a:7d:74	HewlettP_61:24:92	ARP	60	172.16.30.254 is at 00:21:5a:5a:7
23	6.611065260	172.16.30.1	142.250.184.174	ICMP	98	Echo (ping) request id=0x67ee, s
24	6.624966506	142.250.184.174	172.16.30.1	ICMP	98	Echo (ping) reply id=0x67ee, s
25	6.692264264	HewlettP_5a:7d:74	HewlettP_61:24:92	ARP	60	Who has 172.16.30.1? Tell 172.16
26	6.692282143	HewlettP_61:24:92	HewlettP_5a:7d:74	ARP	42	172.16.30.1 is at 00:21:5a:61:24
27	7.613054445	172.16.30.1	142.250.184.174	ICMP	98	Echo (ping) request id=0x67ee, s
28	7.626964770	142.250.184.174	172.16.30.1	ICMP	98	Echo (ping) reply id=0x67ee, s
29	7.998407858	Routerbo_1c:8e:13	Spanning-tree-(for-bri...	STP	60	RST. Root = 32768/0/c4:ad:34:1c:8

Ethernet II, Src: HewlettP\_61:24:92 (00:21:5a:61:24:92), Dst: HewlettP\_5a:7d:74 (00:21:5a:5a:7d:74)  
Internet Protocol Version 4, Src: 172.16.30.1, Dst: 172.16.1.1  
User Datagram Protocol, Src Port: 55205, Dst Port: 53  
Domain Name System (query)

0000 00 21 5a 5a 7d 74 00 21 5a 61 24 92 08 00 45 00 :!ZZ}t! Za\$...E.  
0010 00 38 74 6e 40 00 40 11 4f 24 ac 10 1e 01 ac 10 :8tn@ @ OS .....  
wireshark\_eth0\_20231211095515\_yiYLVp.pcapng Packets: 43 · Displayed: 43 (100.0%) · Dropped: 0 (0.0%) Profile: Default

Figure 19: Experiência 5 - TUX62 -i TUX63

## 5.2.6 Experiência 6



No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	Routerbo_1c:8e:13	Spanning-tree-(for-bri...	STP	60	RST. Root = 32768/0/c4:ad:34:1c:8e:13 Cost = 0 Port = 0x8001
2	1.443360734	172.16.1.1	172.16.30.1	DNS	73	Standard query 0x578b A mirrors.up.pt
3	1.443372188	172.16.30.1	172.16.1.1	DNS	89	Standard query response 0x578b A mirrors.up.pt A 193.137.29.15
4	1.602968780	172.16.1.1	172.16.30.1	TCP	74	47034 → 21 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=1588
5	1.603106908	172.16.30.1	172.16.1.1	TCP	74	21 → 47034 [SYN, ACK] Seq=0 Ack=1 Win=65160 Len=0 MSS=1380 SACK_PERM=1
6	1.603437809	172.16.30.1	172.16.30.1	TCP	66	47034 → 21 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=1588387704 TSecr=21
7	1.617961494	172.16.30.1	172.16.30.1	FTP	139	Response: 220-Welcome to the University of Porto's mirror archive (mirr
8	1.618074288	172.16.30.1	172.16.30.1	TCP	66	47034 → 21 [ACK] Seq=1 Ack=74 Win=64256 Len=0 TSval=1588387710 TSecr=2
9	1.618487891	172.16.30.1	172.16.30.1	FTP	141	Response: 220-.....
10	1.618963901	172.16.30.1	172.16.30.1	TCP	66	47034 → 21 [ACK] Seq=1 Ack=149 Win=64256 Len=0 TSval=1588387710 TSecr=
11	1.619962130	172.16.30.1	172.16.30.1	FTP	292	Response: 220-All connections and transfers are logged. The max number-
12	1.621985577	172.16.30.1	172.16.30.1	TCP	66	47034 → 21 [ACK] Seq=1 Ack=375 Win=64128 Len=0 TSval=1588387711 TSecr=
13	1.622983966	172.16.30.1	172.16.30.1	FTP	84	Response: 220-
14	1.624966506	172.16.30.1	172.16.30.1	TCP	66	47034 → 21 [ACK] Seq=1 Ack=393 Win=64128 Len=0 TSval=1588387711 TSecr=
15	1.626964770	172.16.30.1	172.16.30.1	FTP	81	Request: user anonymous
16	1.628966996	172.16.30.1	172.16.30.1	TCP	66	21 → 47034 [ACK] Seq=393 Ack=16 Win=65280 Len=0 TSval=2109433410 TSecr=
17	1.630969226	172.16.30.1	172.16.30.1	FTP	100	Response: 331 Please specify the password.
18	1.632971452	172.16.30.1	172.16.30.1	FTP	80	Request: pass password
19	1.634973678	172.16.30.1	172.16.30.1	FTP	89	Request: 230 Login successful.
20	1.636975904	172.16.30.1	172.16.30.1	FTP	77	Request: cwd parrot
21	1.638978130	172.16.30.1	172.16.30.1	FTP	103	Response: 250 Directory successfully changed.
22	1.640980356	172.16.30.1	172.16.30.1	FTP	71	Request: pasv
23	1.642982582	172.16.30.1	172.16.30.1	FTP	117	Response: 227 Entering Passive Mode (193,137,29,15,212,26).
24	1.644984808	172.16.30.1	172.16.30.1	TCP	74	41898 → 54298 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=1
25	1.646987034	172.16.30.1	172.16.30.1	TCP	74	54298 → 41898 [SYN, ACK] Seq=0 Ack=1 Win=65160 Len=0 MSS=1380 SACK_PER
26	1.648989260	172.16.30.1	172.16.30.1	TCP	66	41898 → 54298 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=1588387726 TSecr=
27	1.650991486	172.16.30.1	172.16.30.1	FTP	87	Request: retr last-update.txt
28	1.652993712	172.16.30.1	172.16.30.1	FTP-DATA	97	FTP Data: 31 bytes (PASV) (retr last-update.txt)
29	1.654995938	172.16.30.1	172.16.30.1	TCP	66	41898 → 54298 [ACK] Seq=1 Ack=32 Win=64256 Len=0 TSval=1588387729 TSecr=

Frame 1: 86 bytes on wire (688 bits), 86 bytes captured (688 bits) on interface 0  
Ethernet II, Src: HewlettP\_61:24:92 (00:21:5a:61:24:92), Dst: HewlettP\_5a:7d:74 (00:21:5a:5a:7d:74)  
Internet Protocol Version 4, Src: 172.16.30.1, Dst: 172.16.1.1  
User Datagram Protocol, Src Port: 35844, Dst Port: 53  
Domain Name System (query)

0000 00 21 5a 5a 7d 74 00 21 5a 61 24 92 08 00 45 00 :!ZZ}t! Za\$...E.  
0010 00 48 be a3 40 00 40 11 04 df ac 10 1e 01 ac 10 :H...@ @ .....  
eth0: <live capture in progress> Packets: 58 · Displayed: 58 (100.0%) Profile: Default

Figure 20: Experiência 5 - TUX62 -i TUX63

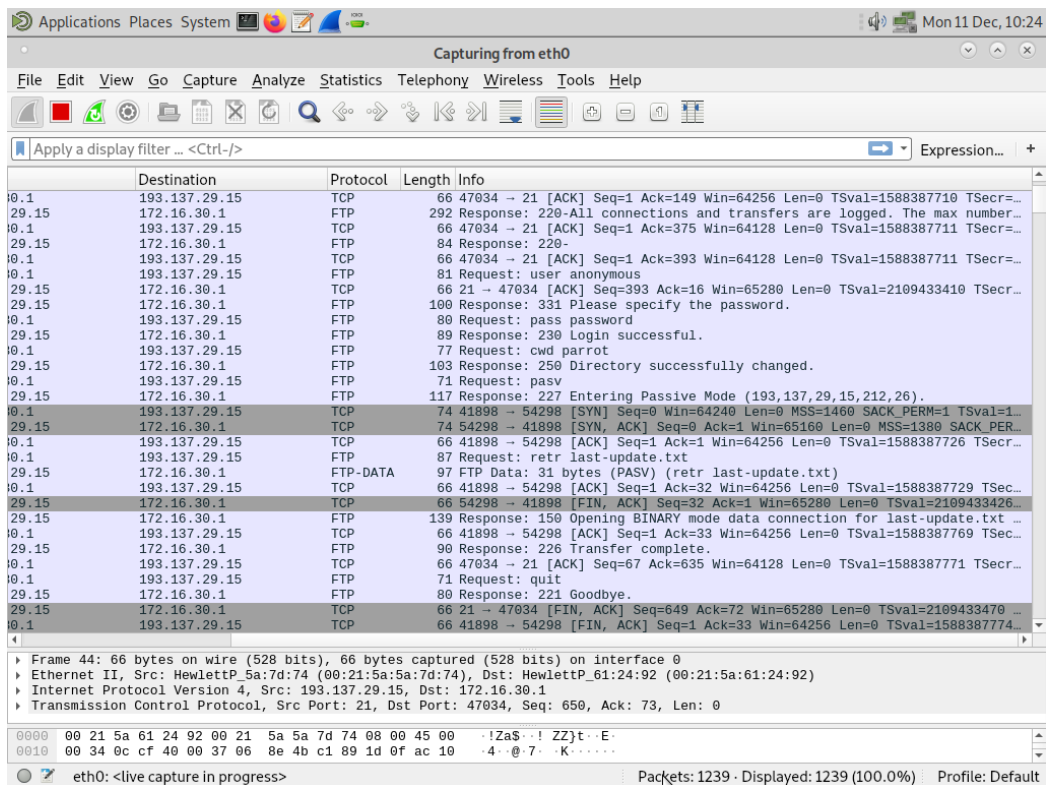


Figure 21: Experiência 5 - TUX62 -> TUX63