**Faculdade de Engenharia da Universidade do Porto**



**Segundo Trabalho Laboratorial**

Configuração e Estudo de uma Rede

**RCOM – Redes de Computadores**

**2016/2017, MIEIC**

**Turma 07, Bancada 04:**

* Sara Filipa Couto Fernandes [**up201405955@fe.up.pt**](up201405955@fe.up.pt)
* Vasco Magalhães Pereira [**up201403485@fe.up.pt**](up201403485@fe.up.pt)
* João Pedro Gomes Silva [**up201405490@fe.up.pt**](up201405490@fe.up.pt)
* Diogo Amorim Cepa [**up201403367@fe.up.pt**](up201403367@fe.up.pt)

**Data:** 20 de dezembro de 2016

# Sumário

No âmbito da unidade curricular de Redes de Computadores, **RCOM**, do terceiro ano, primeiro semestre, do Mestrado Integrado em Engenharia Informática e Computação, **MIEIC**, foi proposto o desenvolvimento de um cliente **FTP** (*File Teansfer Protocol*), capaz de executar o *download* de um ficheiro, assim como a configuração e estudo de uma rede de computadores, com auxílio a várias experiências laboratoriais executadas durantes as próprias aulas práticas da unidade curricular.

Ao longo do desenvolvimento deste projeto laboratorial, realizaram-se **seis** experiências laboratoriais. Primeiramente, começou-se por configurar uma **rede IP**. De seguida, implementou-se duas **LANs** (*Local Area Network*) virtuais num *switch*, configurou-se um *router* em Linux e, posteriormente, configurou-se um *router* comercial implementando **NAT** (*Network Address Translation*). Por fim, implementou-se a aplicação de *download* **FTP**, usando como recurso as configurações obtidas pelas experiências laboratoriais.

# Índice

[1. Introdução 4](#_Toc470002176)

[2. Cliente FTP 4](#_Toc470002177)

[3. Experiências Laboratoriais 6](#_Toc470002178)

[3.1 Experiência 1 – Configurar uma rede de IPs 6](#_Toc470002179)

[3.2 Experiência 2 – Implementar duas LANs virtuais num switch 6](#_Toc470002180)

[3.3 Experiência 3 – Configurar um router em Linux 7](#_Toc470002181)

[3.4 Experiência 4 - Configurar um router comercial e implementar NAT 8](#_Toc470002182)

[3.5 Experiência 5 – DNS 8](#_Toc470002183)

[3.6 Experiência 6 – Conexões TCP 9](#_Toc470002184)

[4. Conclusão 11](#_Toc470002185)

[Anexo I 12](#_Toc470002186)

[Anexo II 12](#_Toc470002187)

[Experiência 1 12](#_Toc470002188)

[Experiência 2 12](#_Toc470002189)

[Passo 4 12](#_Toc470002190)

[Passo 7 13](#_Toc470002191)

[Passo 10 14](#_Toc470002192)

[Experiência 3 15](#_Toc470002193)

[Passo 5 15](#_Toc470002194)

[Passo 10 16](#_Toc470002195)

[Experiência 4 17](#_Toc470002196)

[Passo 3 17](#_Toc470002197)

[Passo 4 17](#_Toc470002198)

[Passo 5 18](#_Toc470002199)

[Passo 7 18](#_Toc470002200)

[Experiência 5 18](#_Toc470002201)

[Passo 3 18](#_Toc470002202)

[Experiência 6 19](#_Toc470002203)

[Passo 2 19](#_Toc470002204)

[Passo 5 20](#_Toc470002205)

# Introdução

No âmbito da unidade curricular de Redes de Computadores, **RCOM**, foi proposto ao grupo a implementação de uma aplicação de *download* utilizando o protocolo **FTP**, com recurso a diversas experiências laboratoriais, realizadas no decorrer das aulas práticas.

A aplicação tinha como principal objetivo a implementação de um cliente **FTP**, na linguagem de programação C, usando *sockets* e **TCP** (*Transmission Control Protocol*). Esta aplicação é de fácil utilização, uma vez que o utilizador somente precisa de passar o **URL**, em sintaxe **RFC1738**, de forma correta:

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

As experiências laboratoriais executadas deram a conhecer os comandos necessários para configurar uma rede, aprofundando os conhecimentos teóricos abordados na unidade curricular.

Este relatório contém um capítulo referente a cada parte do trabalho, em que a primeira parte aborda a criação de um *parser* referente ao **URL**, passado como argumento, assim como a criação do cliente **FTP**. A segunda parte destina-se às experiências laboratoriais, onde cada subsecção é uma experiência respetiva.

# Cliente FTP

Foi desenvolvida uma aplicação em linguagem C, com o objetivo de fazer o *download* de um ficheiro de um servidor **FTP**. Após se ter compilado o *makefile* do projeto, deve-se correr o seguinte comando, onde o **URL** segue a sintaxe descrita no **RFC1738**:

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

Inicialmente, a aplicação começa por interpretar o **URL** respetivo, através de **parser.h** e **parser.c**, extraindo o ***username***, ***password***, ***host*** e o **diretório do ficheiro** a descarregar. Caso o utilizador queira fazer o *download* em modo anónimo basta que o **URL** esteja descrito da seguinte forma:

**ftp:// <host>/<url-path>**

Neste último caso, o ***username*** e ***password*** terão valores *default*, sendo definidos com ***“anonymous”*** e ***“mail@domain”***, respetivamente.

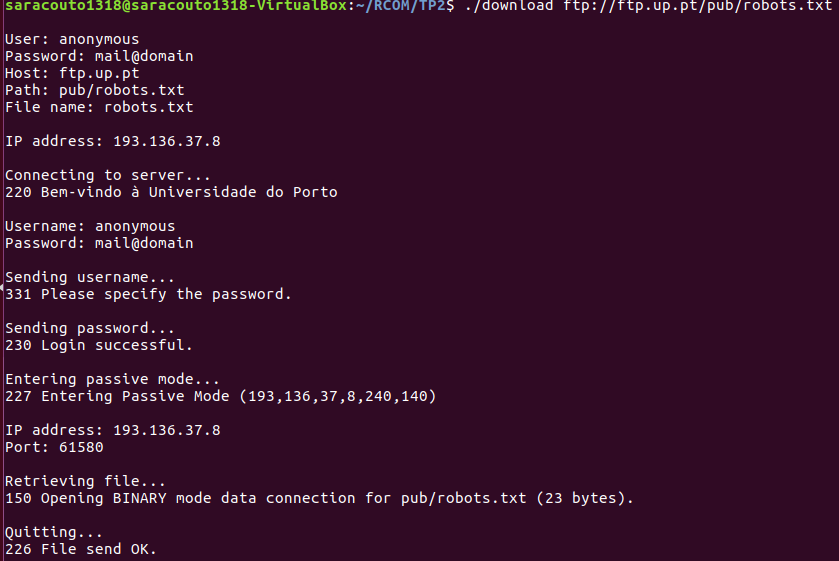
Depois de todas estas informações serem armazenadas, a aplicação tenta converter o ***host*** num endereço **IP**, através da função **getHostIP()**, pertencente a **FTPClient.h** e **FTPClient.c**. Esta função foi fornecida juntamente com o guião referente ao segundo trabalho e tem como objetivo retornar todos os **IPs** possíveis para criar uma conexão ao servidor **FTP**.

De seguida, tenta-se estabelecer a conexão **TCP**, utilizando a **porta 21**, e caso esta seja estabelecida com sucesso, é criado um *socket* para que seja possível comunicar com o servidor, sendo uma mensagem de boas vindas apresentada ao utilizador, através do terminal.

Assim, após todos estes passos é executado o *login*, utilizando o ***username***, através do comando **“USER *username”***, e a sua respetiva ***password***, através do comando **“PASS *password*”**. Se o *login* for executado com sucesso, tenta-se entrar um **modo passivo**, usando o comando **“PASV “**. Caso a entrada em **modo passivo** seja feita com sucesso, o servidor retorna uma mensagem que contém o **IP** e a **porta** por onde irá ser descarregado o ficheiro.

Após a entrada em **modo passivo**, o cliente **FTP** envia o comando **“RETR *url-path*”**, para indicar qual o ficheiro a transferir e por fim, tenta abrir o mesmo, iniciando-se assim o *download* propriamente dito.

Por fim, faz-se o *logout*, utilizando o comando **“QUIT “**, de forma a concluir o *download* e a finalizar a aplicação.



**Figura 1 - Execução da aplicação**

# Experiências Laboratoriais

## Experiência 1 – Configurar uma rede de IPs

Esta primeira experiência teve como objetivo fazer com que soubéssemos distinguir diferentes pacotes de dados e as suas finalidades.

Através da análise dos *logs*, verificámos que um dos pacotes de dados são os pacotes **ARP**. Estes pacotes permitem encontrar um endereço da camada de ligação **(MAC)** através de um **IP**.

O primeiro pacote **ARP**, é enviado para todos os computadores, contendo o **IP** para o qual é pretendido saber o **MAC**. Para que tal seja obtido, deve-se enviar o pacote em *broadcast*, a partir da máquina que deseja receber os dados.

No segundo pacote **ARP**, a máquina cujo **MAC** é o endereço alvo do primeiro pacote, envia o seu endereço **MAC** à máquina que pretendia esta informação.

Outros pacotes de dados verificados foram os pacotes **ICMP**, gerados através do comando *ping*. Estes pacotes têm dois **IPs** associados, um relativo à fonte e outro relativo ao destino.

Para que seja, portanto, possível distinguir as tramas (**ARP, IP** ou **ICMP**), é necessário observar o cabeçalho das mesmas e tentar analisá-lo. É, igualmente, através do cabeçalho que podemos descobrir o tamanho dos pacotes de dados.

Por vezes, no envio das tramas é ativado o mecanismo de *loopback* que permite detetar erros na transmissão de dados, uma vez que o emissor volta a receber o pacote de dados que tentou enviar. Este mecanismo permite, também, saber se os cabos de ligação se encontram em boas condições de uso.

## Experiência 2 – Implementar duas LANs virtuais num *switch*

A segunda experiência tem como objetivo ensinar a configurar o *switch* e, posteriormente, criar duas ***LANs*** virtuais.

Para se conseguir configurar um **VLAN**, primeiro tem que se abrir o modo de configuração, utilizando **“configure terminal”** e depois deve-se executar o comando **“vlan x**”, onde **x** é o número da **VLAN** respetiva, na consola do *switch*.

De seguida, tem que se adicionar as respetivas portas em que os ***tuxs*** estão ligados no *switch*, selecionando a porta na mesma consola através do comando **“interface fastethernet 0/y”**, onde **y** é o número da porta a adicionar. Após o uso do comando referido, é preciso mudar o modo, **“switchport mode access”**, e adicionar a porta a uma **VLAN** com o comando **“switchport access vlan x”**, onde **x** é o número da **VLAN**. Por fim, termina-se a configuração da **VLAN** com o comando “**end**”.

Através da observação e análise dos *logs*, podemos concluir que há um *broadcast domain* para cada **VLAN**. Esta conclusão foi obtida, visto que fazendo *broadcast* no **tux1** apenas os *tuxs* da mesma **VLAN** recebiam os pacotes de dados, verificando-se o mesmo para o **tux2**.

## 3.3 Experiência 3 – Configurar um *router* em Linux

A terceira experiência teve como objetivo converter o **tux4** num *router*.

No **tux1** tínhamos a rota da interface **eth0**, ou seja, a rota que é automaticamente criada quando é atribuído o endereço **IP** a essa mesma interface e uma outra rota para a rede **172.16.41.0**, através da *gateway* **172.16.40.254**.

No **tux2** tínhamos a rota da interface **eth0**, ou seja, a rota que é automaticamente criada quando é atribuído o endereço **IP** a essa mesma interface e uma outra rota para rede **172.16.40.0**, através da *gateway* **172.16.41.253**.

No **tux4** tínhamos as rotas das interfaces **eth0** e **eth1**, sendo que as redes **172.16.40.0** e **172.16.41.0**, são alcançáveis por este mesmo *tux*.

A tabela de *fowarding* (visível através do comando **“route -n”**), contém a informação da rede de destino, a *gateway* por onde o acesso é feito, a máscara de rede e a interface associada.

No **passo 11** desta experiência, pôde-se observar que os pacotes **ARP** têm como objetivo informar o **tux4** sobre quais os endereços **MAC** que estão associados ao **IP** de destino. No início, o **tux1** verifica quem tem o **IP** **172.16.40.254**. De seguida, o **tux4** verifica quem tem o **IP 172.16.41.1** e no caminho oposto, o **tux2** verifica quem tem o **IP 172.16.41.253** e o **tux4**, verifica numa outra vez, quem tem o **IP 172.16.40.1**. Já os pacotes **ICMP** têm sempre o mesmo **IP** de fonte e destino para que ao passarem pelos diferentes *tuxs*saberem sempre para onde têm de reencaminhar os pacotes.

## 3.4 Experiência 4 - Configurar um *router* comercial e implementar NAT

A quarta experiência consistiu em configurar um *route*r comercial e implementar **NAT**, percebendo melhor esta funcionalidade.

Para que se pudesse adicionar uma rota estática num *router* comercial, escreveu-se o seguinte comando **“ip route [rede de destino] [máscara de rede] [gateway IP]”**, na consola do *router*.

Nesta experiência, caso o **tux2** tenha uma rota para a rede **172.16.40.0**, então os pacotes percorrem o seguinte trajeto **tux2 -> tux4 -> tux1**. Se essa rota for removida, os pacotes percorrem o caminho **tux2 -> *router* comercial -> tux4 -> tux1**. No entanto se ativarmos os *redirects* no **tux2**, na primeira vez o trajeto será **tux2 -> *router* comercial -> tux4 -> tux1**, contudo nas restantes vezes será **tux2 -> tux4 -> tux1**.

Aqui é, igualmente, pedido para adicionar a funcionalidade **NAT** ao *router* e para tal, basta selecionar, na consola do *router*, a interface pretendida através do comando **“interface gigabitethernet 0/X”**, onde **X** é o número da interface e posteriormente executar o comando **“ip nat outside”** para que a interface esteja ligada ao *router* da sala. Com **“ip nat inside”** as rotas são primeiro encaminhadas e depois interpretadas, sendo que no caso de **“nat outside”** se verifica o oposto.

O **NAT** é uma funcionalidade que permite reescrever os **IPs** de origem e destino dos pacotes recebidos num *router* de forma a conectar uma rede local a uma rede externa. Assim, quando os pacotes de dados chegam ao *router*, provenientes da rede externa, ele sabe qual é o *tux* de destino, uma vez que este mantém uma *hash table* contendo essa informação.

## 3.5 Experiência 5 – DNS

A quinta experiência teve como objetivo configurar o servidor **DNS**, assim como verificar qual é o seu papel nas redes de computadores.

Para configurar o servidor **DNS** tem-se que editar o ficheiro **/etc/resolv.conf**, especificando o **search (lixa.netlab.fe.up.pt)** e o **nameserver (172.16.1.1)**. Estes valores podem ser alterados utilizando o comando **“echo ”search lixa.fe.up.pt\nnameserver 172.16.1.1“ > /etc/resolv.conf**.

O servidor **DNS** é responsável por converter um endereço web, como por exemplo **www.google.pt**, num endereço **IP** do tipo **8.8.8.8**.

Um primeiro pacote é enviado através do **tux1** ao servidor **DNS**, com o domínio do website e este responde com o respetivo **IP**. Por sua vez, um segundo pacote envia uma lista contendo todos os servidores com aquele nome associado.

## 3.6 Experiência 6 – Conexões TCP

A sexta experiência teve como objetivo testar a aplicação **FTP** desenvolvida pelo grupo.

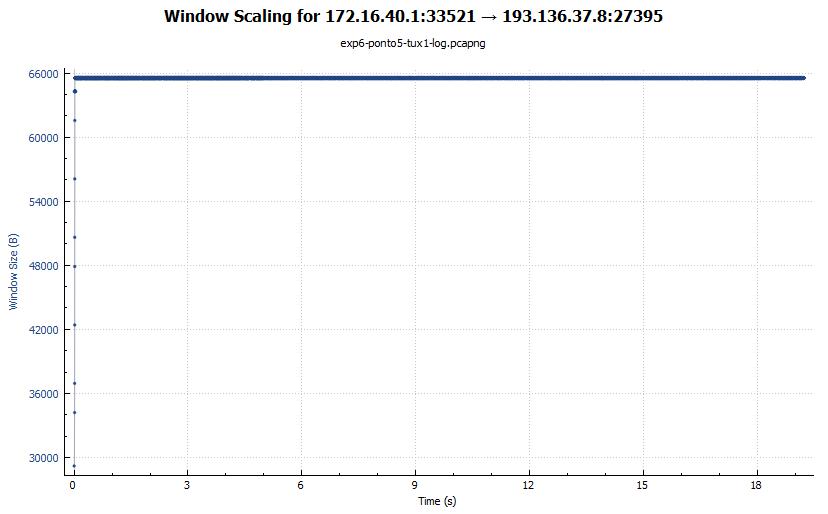
Esta aplicação abre duas ligações **TCP**. A primeira envia a informação do protocolo **FTP**, para se conseguir enviar pedidos ao servidor, como o ***username***, ***password*** e ficheiro que pretendemos receber. A segunda ligação destina-se a transferir a informação do ficheiro que se escolheu para transferir.

A conexão **TCP** tem quatro fases que recaem sobre o estabelecimento da ligação, aceitação e rejeição de pacotes e término da ligação estabelecida.

O protocolo **TCP** utiliza um mecanismo **ARQ** bastante semelhante ao **Go-Back-N**, utilizando uma ***Sliding Window***. Este é um mecanismo que em caso de erro na transmissão respetiva, tentar reenviar um pacote de cada vez, uma vez que assume que só um foi perdido. Os campos relevantes para este método são o **ACK**, tamanho da janela e número de sequência, sendo igualmente importante saber qual foi o último *byte* escrito, para se conseguir atualizar o tamanho da ***Sliding Window***.

Para que o congestionamento de informação seja evitado, o protocolo **TCP** calcula se é possível enviar a informação sem ultrapassar o limite do *buffer*, o que caso aconteça faz com que bloqueie a aplicação até existir mais espaço disponível.

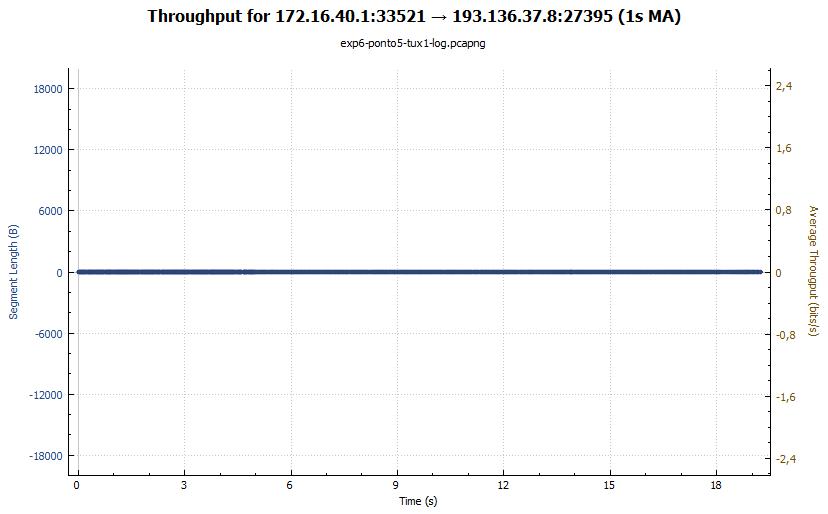
No seguinte gráfico consegue-se perceber a evolução da janela ao longo do tempo:



**Figura 2 - Tamanho de janela**

Assim, podemos verificar o efeito *slow-start* do **TCP**, assim como a consistência do tamanho da janela.

Nesta experiência, foi testado ainda o *download* simultâneo em dois *tuxs*, **tux1** e **tux2**. Apesar *da* ***Sliding Window*** mostrar uma interferência no crescimento, a taxa de transferência mantem-se igual, logo as múltiplas transferências em *tuxs* diferentes não influenciam a taxa de transferência, uma vez que no modo passivo cada transferência ocorre num porto diferente.



**Figura 3 - Taxa de transferência**

# Conclusão

Durante a elaboração da aplicação de *download*, utilizando o protocolo **FTP** e da realização das experiências laboratoriais, a compreensão do guião respetivo tornou-se importante para que todos os objetivos proposto fossem cumpridos. Igualmente, foi fucral o apontamento de diversas informações e comandos para que cada fase das experiências fosse cumprida com precisão de forma a que não se comprometesse as experiências seguintes.

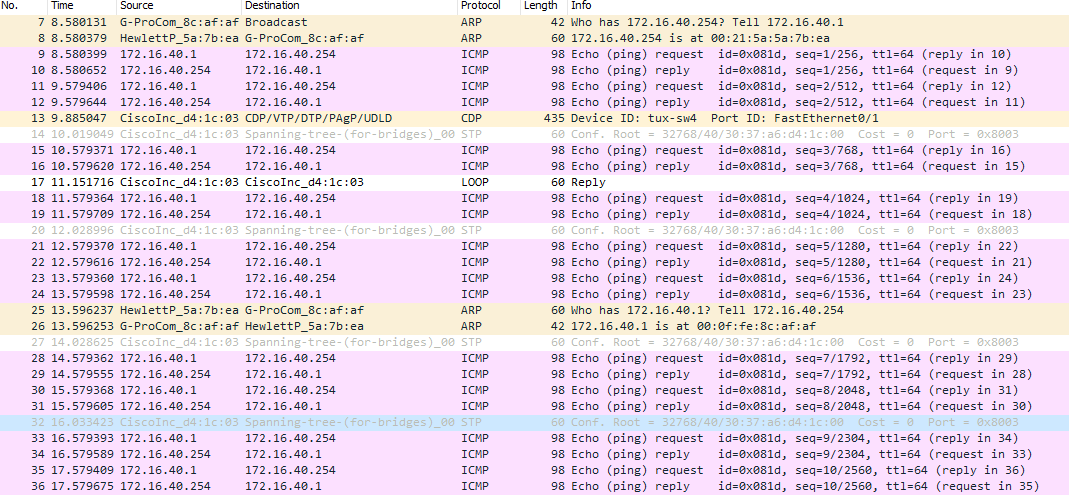
No final da realização das duas partes do guião, o grupo conseguiu perceber como é que funcionava uma rede, neste caso, com um tamanho bastante reduzido, apercebendo-se da complexidade e trabalho que seria necessário para a criação de uma rede de elevada dimensão.

# **Anexo I**

O código fonte do projeto encontra-se anexado juntamente com este relatório, na pasta denominada por “**Código Fonte**”.

# **Anexo II**

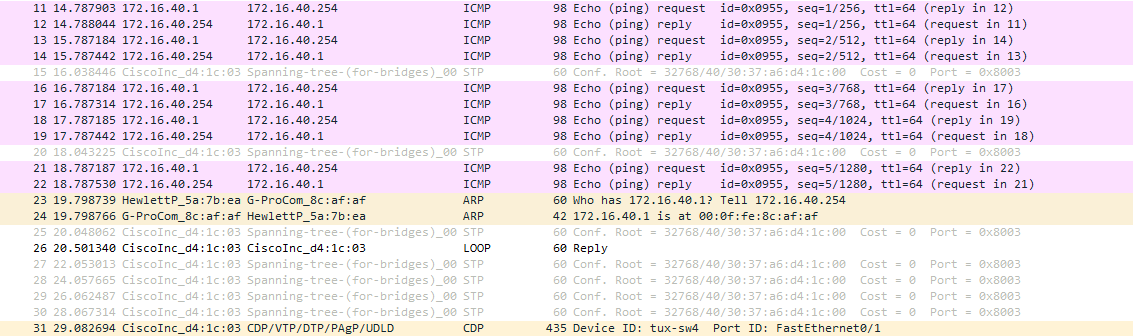
## **Experiência 1**

****

## **Experiência 2**

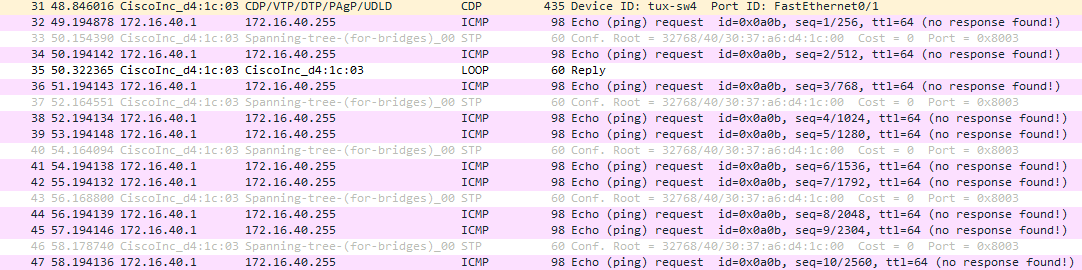
### **Passo 4**

**Tux1**

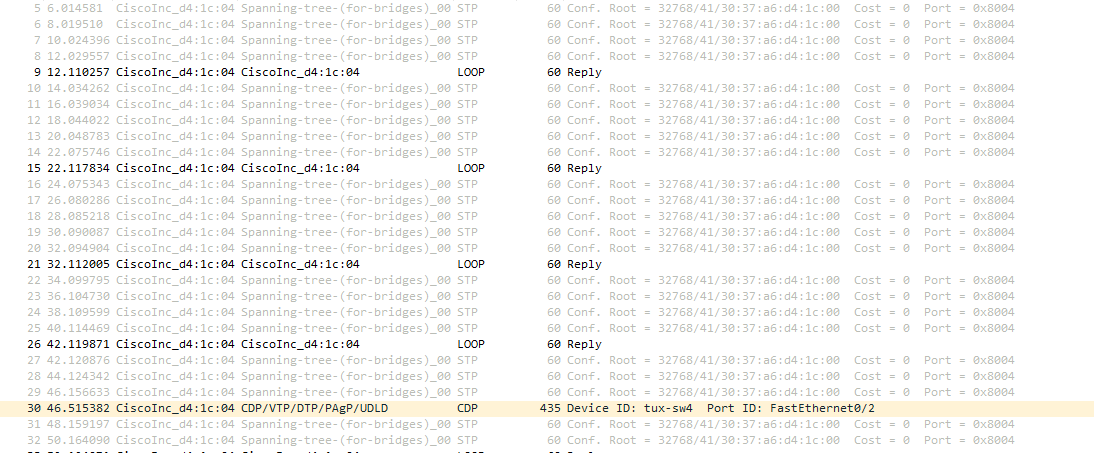
****

### **Passo 7**

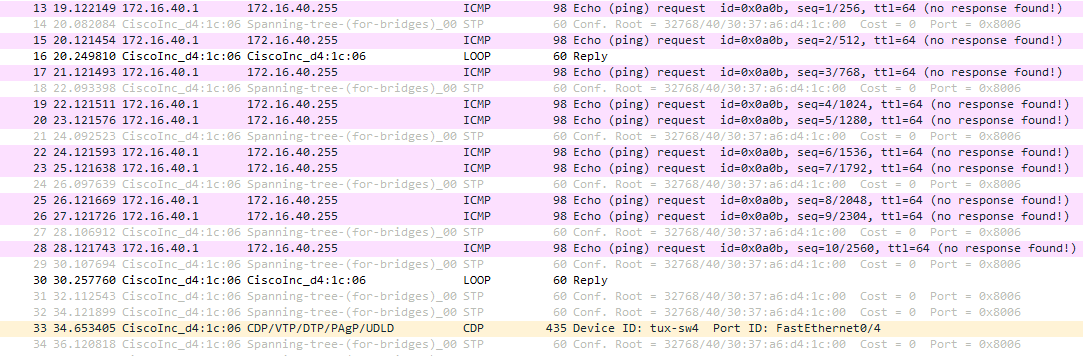
**Tux1**

****

**Tux2**

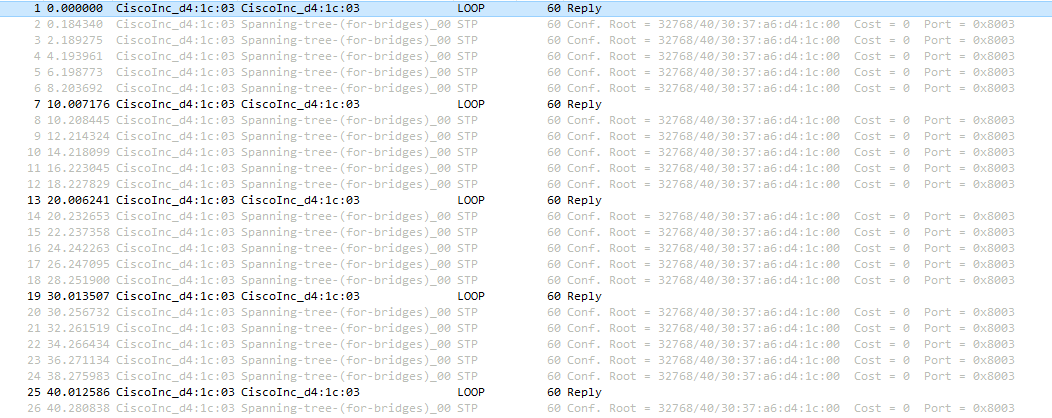
****

**Tux4**

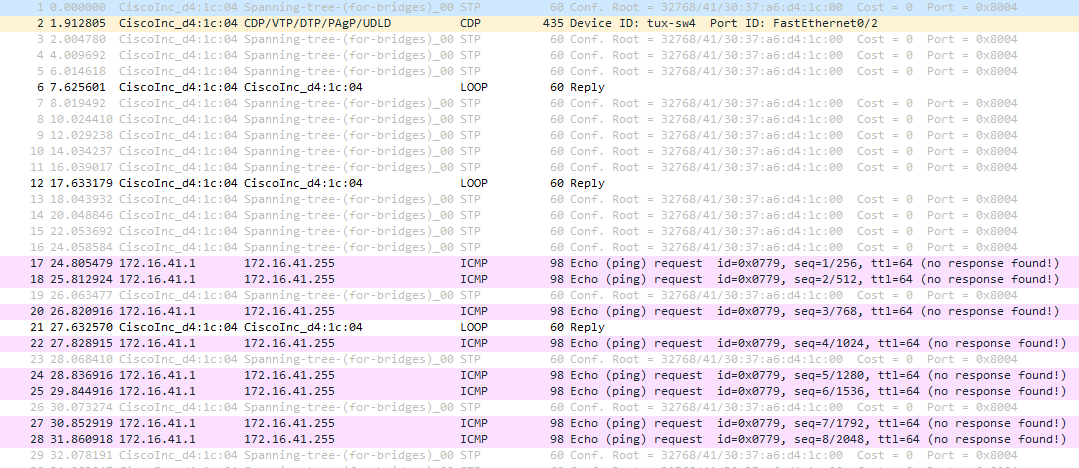
****

### **Passo 10**

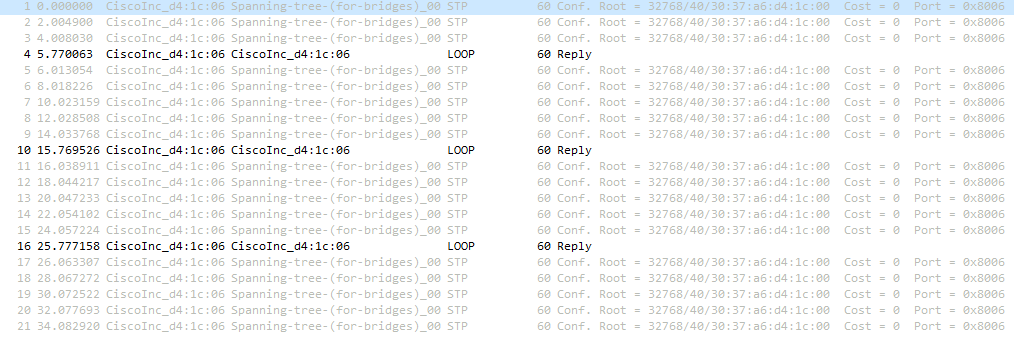
**Tux1**

****

**Tux2**

****

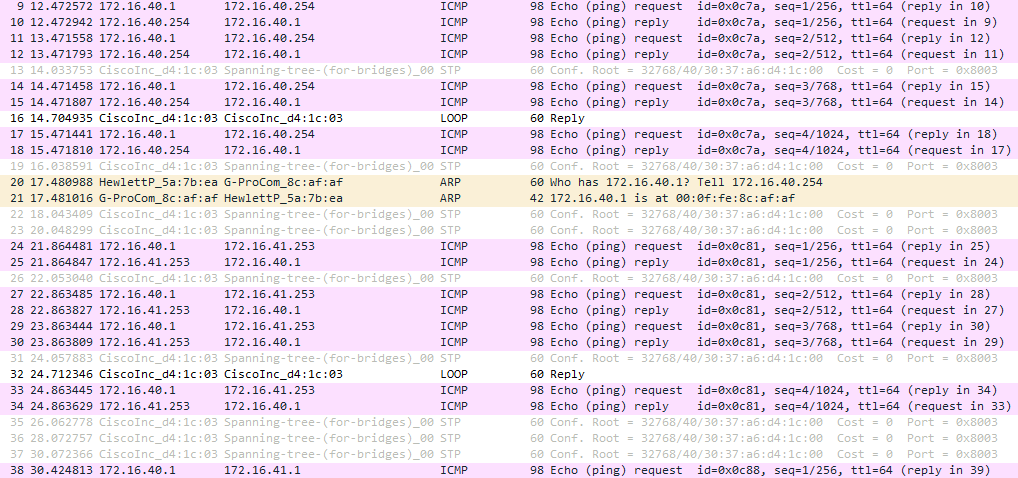
**Tux4**

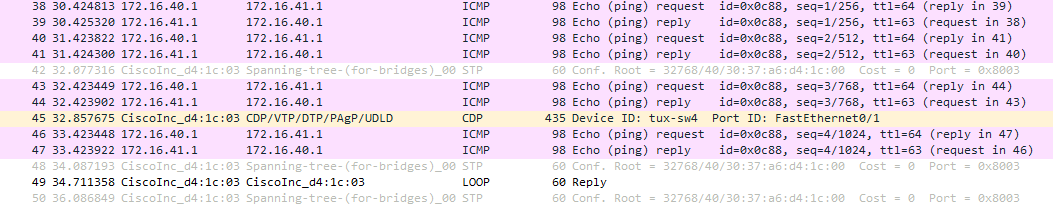
****

## **Experiência 3**

### **Passo 5**

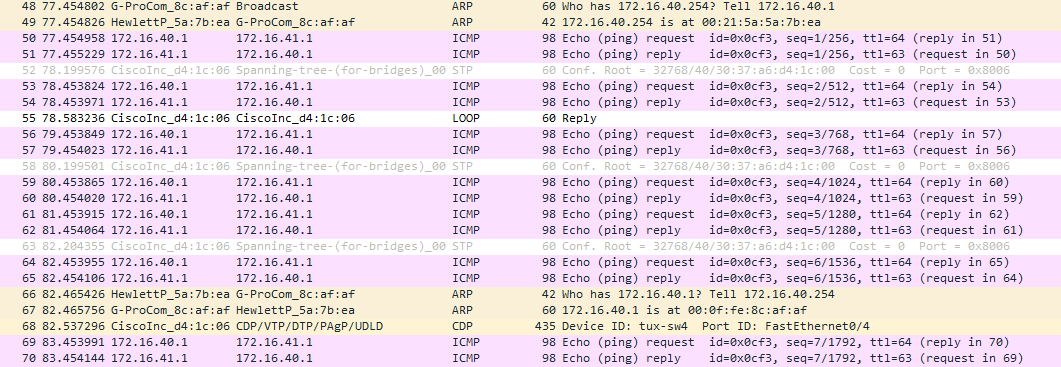
**Tux1**

****

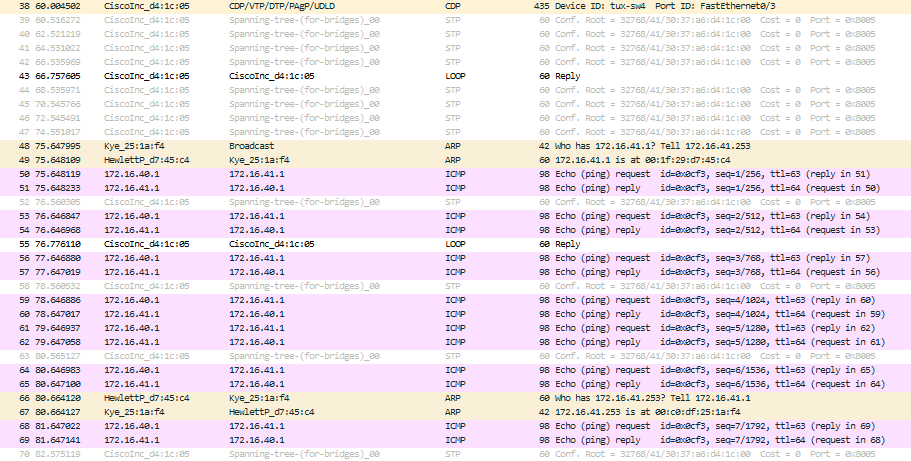
****

### **Passo 10**

**Tux4 eth0**

****

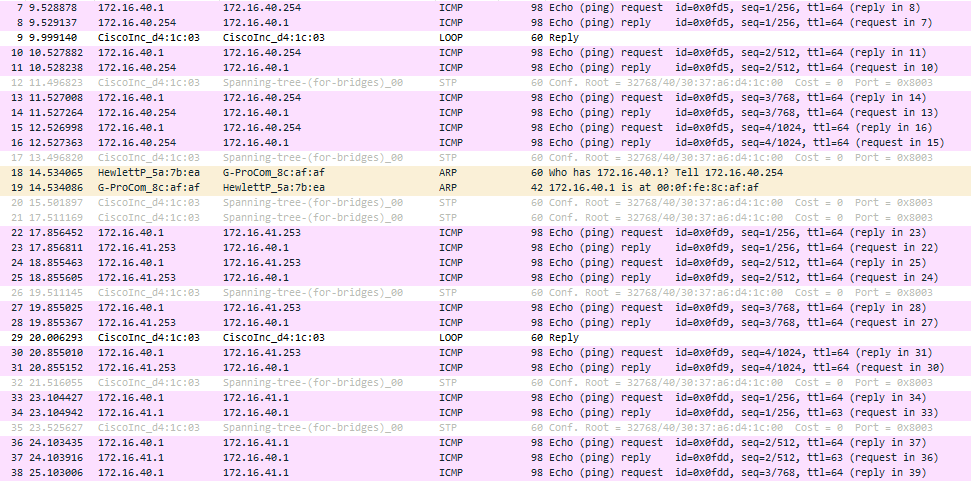
**Tux4 eth1**

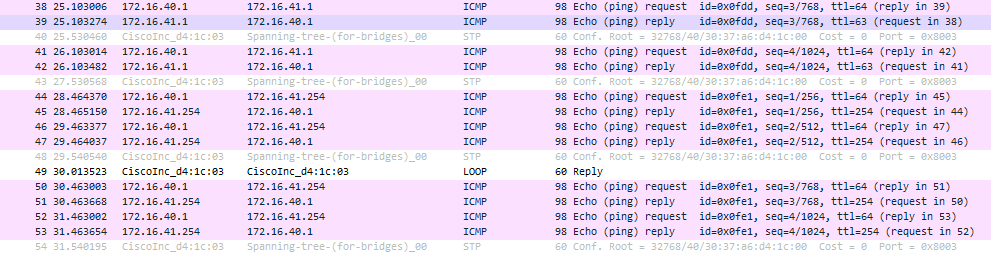
****

## **Experiência 4**

### **Passo 3**

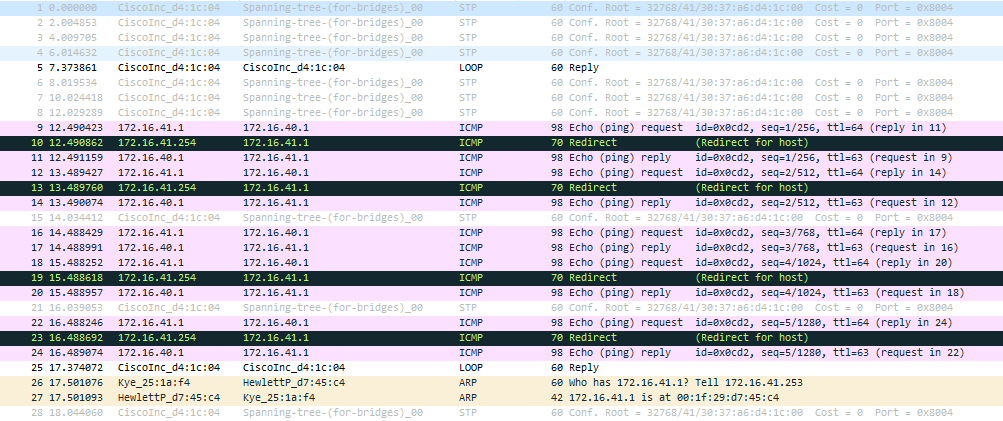
**Tux1**

****

****

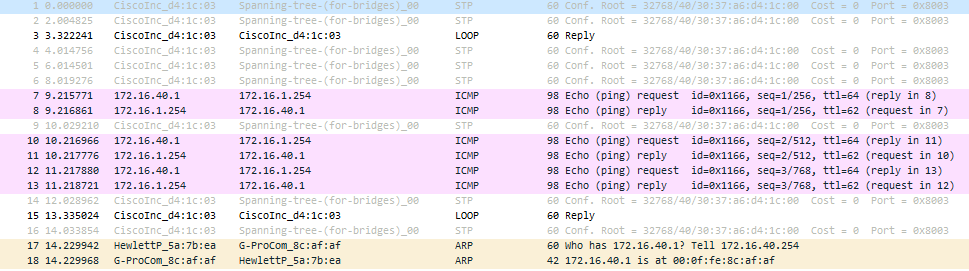
### **Passo 4**

**Tux2**

****

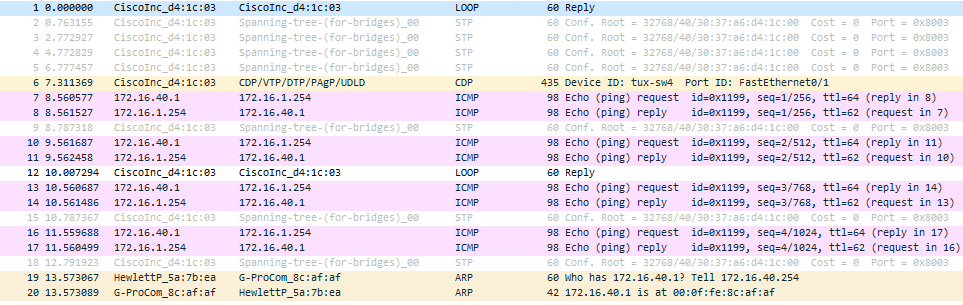
### **Passo 5**

**Tux1**

****

### **Passo 7**

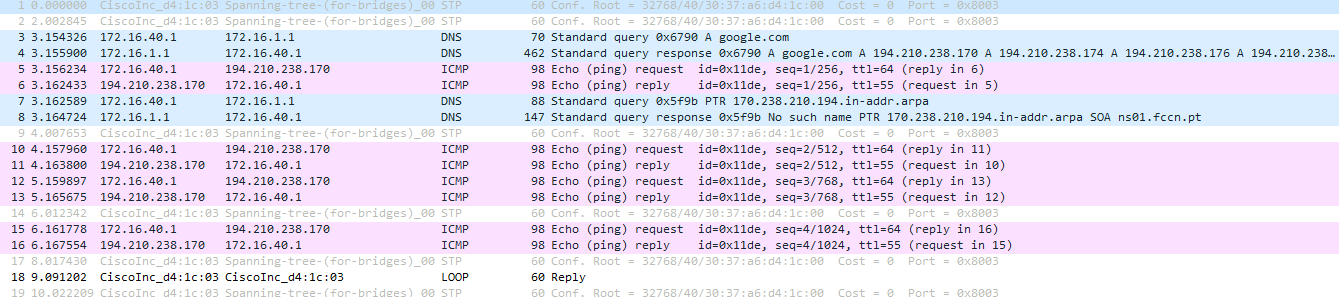
**Tux1**

****

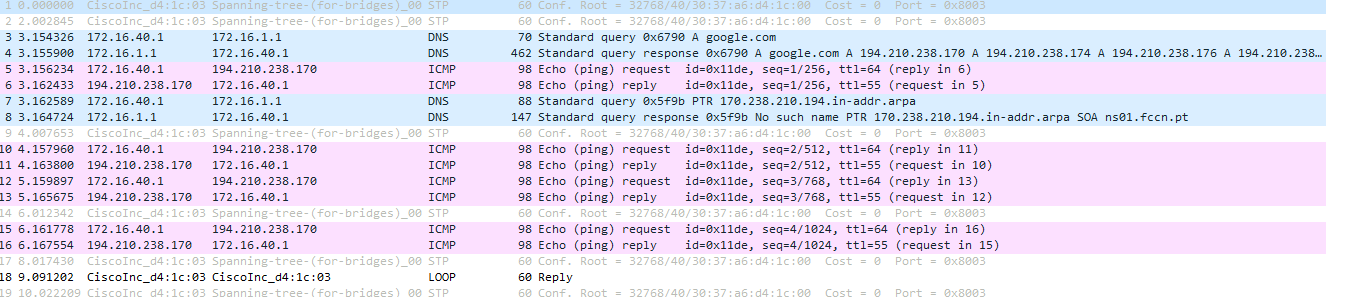
## **Experiência 5**

### **Passo 3**

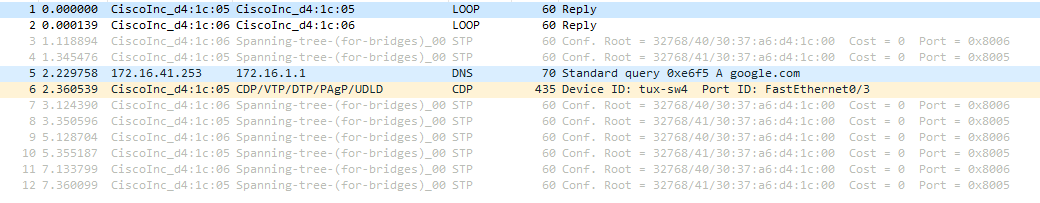
**Tux1**

****

**Tux2**

****

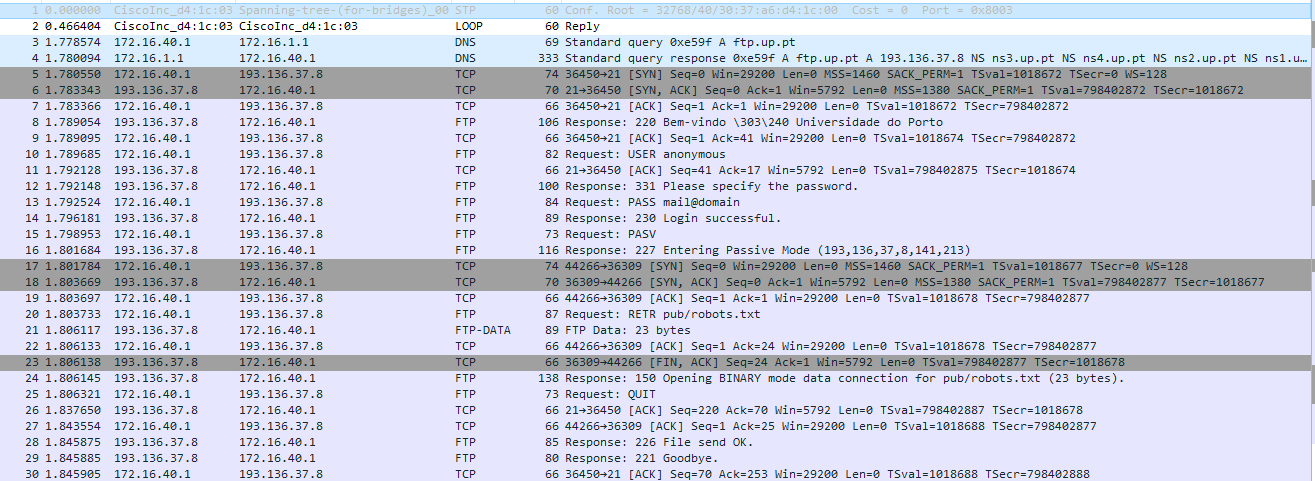
**Tux4**

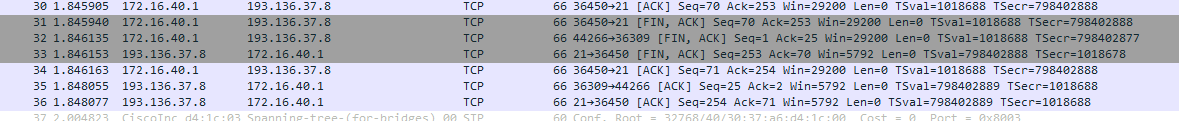
****

## **Experiência 6**

### **Passo 2**

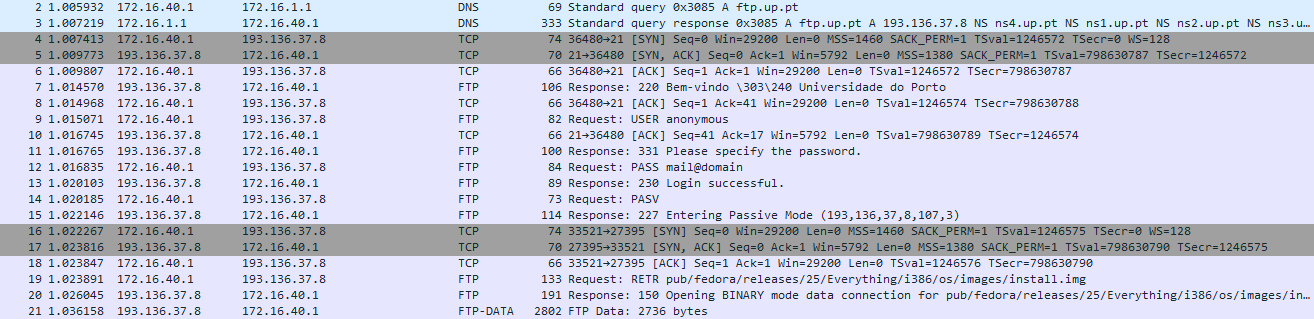
**Tux1**

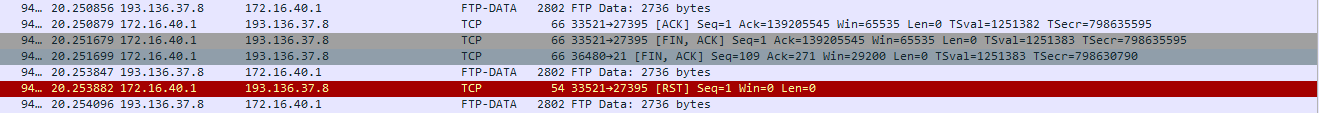
****

****

### **Passo 5**

**Tux1**

****

****