

# *Trabalho Laboratorial 2*

## Relatório Laboratorial



Mestrado Integrado em Engenharia Informática e  
Computação

Redes de Computadores

### **Grupo 3:**

Carlos Miguel Lucas - ei11140  
Maria Teresa Chaves - up201306842

Faculdade de Engenharia da Universidade do Porto  
Rua Roberto Frias, sn, 4200-465 Porto, Portugal

23 de Dezembro de 2016

## Resumo

Este projeto, desenvolvido no âmbito da unidade curricular de Redes de Computadores, consiste na elaboração de uma aplicação para o download de um ficheiro (parte 1) e também na configuração e estudo de uma rede de computadores (parte 2). Para a primeira parte deste projeto desenvolveu-se uma aplicação, em linguagem C, para o download de um ficheiro através de um protocolo FTP. Para a segunda parte deste projeto executou-se um conjunto de experiências que permitiram a configuração e estudo da rede. No final da execução deste conjunto de experiências obteve-se uma rede com duas *VLANs* num *switch*. Esta configuração tinha como principal objetivo permitir o uso da aplicação anteriormente desenvolvida, de forma a que convertesse o *URL* do *host* com o qual iria comunicar para um endereço IP, sendo assim possível o download de um ficheiro de uma rede externa.

# Conteúdo

<b>1</b>	<b>Introdução</b>	<b>5</b>
<b>2</b>	<b>Parte 1 - aplicação <i>Download</i></b>	<b>5</b>
2.1	Arquitetura da aplicação <i>Download</i> . . . . .	5
2.2	Relato de um <i>download</i> com sucesso . . . . .	6
<b>3</b>	<b>Parte 2 – Configuração e análise da rede</b>	<b>7</b>
3.1	Experiência 1 - Configuração de uma rede IP . . . . .	7
3.1.1	Objetivos da experiência . . . . .	7
3.1.2	Arquitetura da rede . . . . .	7
3.1.3	Análise dos <i>logs</i> . . . . .	7
3.2	Experiência 2 . . . . .	7
3.2.1	Objetivos da experiência . . . . .	7
3.2.2	Arquitetura da rede . . . . .	8
3.2.3	Análise dos <i>logs</i> . . . . .	8
3.3	Experiência 3 . . . . .	8
3.3.1	Objetivos da experiência . . . . .	8
3.3.2	Arquitetura da rede . . . . .	8
3.3.3	Análise dos <i>logs</i> . . . . .	8
3.4	Experiência 4 . . . . .	9
3.4.1	Objetivos da experiência . . . . .	9
3.4.2	Arquitetura da rede . . . . .	9
3.4.3	Análise dos <i>logs</i> . . . . .	9
3.5	Experiência 5 . . . . .	9
3.5.1	Objetivos da experiência . . . . .	9
3.5.2	Arquitetura da rede . . . . .	10
3.5.3	Análise dos <i>logs</i> . . . . .	10
3.6	Experiência 6 . . . . .	10
3.6.1	Objetivos da experiência . . . . .	10
3.6.2	Arquitetura da rede . . . . .	10
3.6.3	Análise dos <i>logs</i> . . . . .	11
<b>4</b>	<b>Conclusões</b>	<b>11</b>
<b>5</b>	<b>Referências</b>	<b>11</b>
<b>6</b>	<b>Anexos</b>	<b>12</b>
6.1	Logs . . . . .	12
6.1.1	Experiência 1 - Ping para o <b>tux34</b> a partir do <b>tux31</b> .	12
6.1.2	Experiência 2 - Ping broadcast a partir do <b>tux31</b> . . .	12
6.1.3	Experiência 2 - Ping broadcast a partir do <b>tux31</b> , visto do <b>tux32</b> . . . . .	13
6.1.4	Experiência 2 - Ping broadcast a partir do <b>tux31</b> , visto do <b>tux34</b> . . . . .	13

6.1.5	Experiência 2 - Ping para o <b>tux32</b> e de seguida para o <b>tux34</b> a partir do <b>tux31</b> . . . . .	14
6.1.6	Experiência 3 - Ping para o <b>tux32</b> a partir do <b>tux31</b> , visto do <b>tux34</b> . . . . .	14
6.1.7	Experiência 3 - Ping para o <b>tux32</b> , <b>tux34.eth0</b> e <b>tux34.eth1</b> em simultâneo a partir do <b>tux31</b> . . . . .	15
6.1.8	Experiência 4 - Ping para o <b>tux32</b> , <b>tux34</b> e <b>Rc</b> em simultâneo a partir do <b>tux31</b> . . . . .	15
6.1.9	Experiência 5 - Ping para <i>www.google.pt</i> a partir do <b>tux31</b> . . . . .	16
6.1.10	Experiência 5 - Ping para <i>www.google.pt</i> a partir do <b>tux32</b> . . . . .	16
6.1.11	Experiência 6 - Download de um ficheiro a partir do <b>tux31</b> . . . . .	17
6.1.12	Experiência 6 - Gráfico mostrando o número de pacotes por segundo de um download de um ficheiro a partir do <b>tux31</b> . . . . .	17
6.1.13	Experiência 6 - Gráfico mostrando o número de pacotes por segundo de um download de um ficheiro a partir do <b>tux31</b> , em simultâneo com outro download no <b>tux32</b> . . . . .	18
6.2	Código Fonte . . . . .	18
6.3	Scripts . . . . .	30
6.3.1	Configuração do <b>tux31</b> . . . . .	30
6.3.2	Configuração do <b>tux32</b> . . . . .	30
6.3.3	Configuração do <b>tux34</b> . . . . .	30
6.4	Configurações do router e switch . . . . .	31

# 1 Introdução

No âmbito da unidade curricular de Redes de Computadores, desenvolveu-se uma aplicação para o download de um ficheiro através de um protocolo FTP. Para isso inicialmente foi desenvolvida uma aplicação em linguagem C. No entanto, para que fosse possível a transferência do ficheiro usando esta mesma aplicação, foi necessário configurar e estudar uma rede de computadores. Para esta configuração executaram-se várias experiências de forma a que no final se obtivesse uma rede com duas *VLANs* num *switch*. Assim é possível ao utilizador passar à aplicação um *URL* do *host* com o qual quer comunicar e o programa converte-o num IP, fazendo posteriormente o download do ficheiro pretendido. O trabalho dividiu-se assim em duas fases: o desenvolvimento de uma aplicação de download (parte 1) e a realização de várias experiências para a configuração de uma rede de computadores (parte 2).

## 2 Parte 1 - aplicação *Download*

Nesta primeira fase do projeto desenvolveu-se uma aplicação para download de um ficheiro desde uma rede externa através de um protocolo FTP.

O código fonte encontra-se na secção 6.2 dos Anexos.

### 2.1 Arquitetura da aplicação *Download*

Para poder iniciar a aplicação, deverá ser efetuado o seguinte comando, depois de compilado o projeto:

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

Inicialmente, esta aplicação consiste num *parser* para a string que é indicada pelo utilizador como terceiro parâmetro. Esta string é subdividida pelos seguintes campos:

- **<user>** - *username* do utilizador para efetuar o *login* no servidor ao qual se pretende aceder;
- **<password>** - *password* do utilizador para o mesmo *login*;
- **<host>** - endereço do servidor que se pretende aceder, e onde se encontra o ficheiro pretendido para download;
- **<url-path>** - caminho onde se encontra o ficheiro que se pretende transferir.

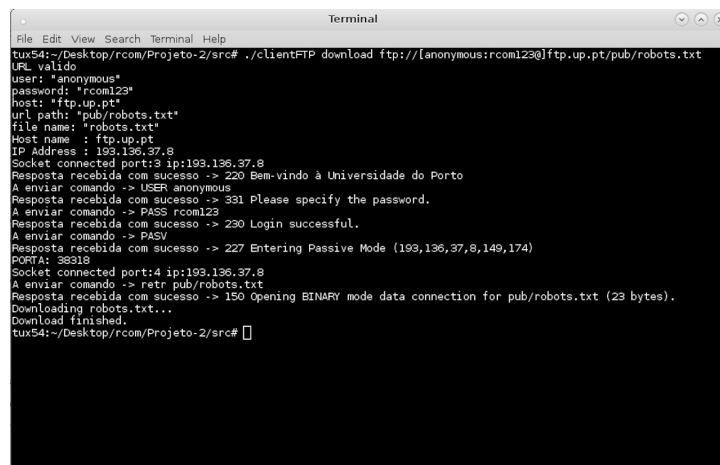
Assim que todos estes campos são corretamente separados, são guardados numa estrutura de dados adequada, para a sua fácil utilização. Caso o *username* e a *password* não sejam passados no *url*, então será utilizado um valor *default* para estes campos. Posteriormente, executam-se os seguintes passos para a conclusão da aplicação:

1. É estabelecida uma ligação com o servidor através da porta 21;
2. É recebida uma mensagem de boas-vindas do servidor, e é enviado o comando "user <user>", de modo a introduzir o **username** do utilizador;
3. É pedida a password do utilizador pelo servidor, e esta é fornecida pelo comando "pass <password>;
4. Após receber uma confirmação de **login** com sucesso por parte do servidor, é enviado o comando "pasv" para entrar em modo passivo;
5. Com base na mensagem enviada pelo servidor, é calculada a nova porta por onde será efetuado o download do ficheiro pretendido, sendo de seguida estabelecida a ligação com o servidor através desta nova porta;
6. Por fim, é enviado o comando "retr <url-path>" para ser iniciado o download do ficheiro indicado.

Concluídos estes passos, o ficheiro indicado pelo parâmetro <url-path> ficará guardado na pasta do projeto.

## 2.2 Relato de um *download* com sucesso

Nesta secção serão mostradas as mensagens que aparecem no ecrã durante o funcionamento da nossa aplicação no caso de o download do ficheiro ter sido finalizado com sucesso.



```

tux54:~/Desktop/rcom/Projeto-2/src# ./clientFTP download ftp://[anonymous:rcml23@]ftp.up.pt/pub/robots.txt
URL valido
user: "anonymous"
password: "rcml23"
host: "ftp.up.pt"
url path: "pub/robots.txt"
file name: "robots.txt"
Host name : ftp.up.pt
IP Address : 193.136.37.8
Socket connected port:3 ip:193.136.37.8
Resposta recebida com sucesso -> 220 Bem-vindo à Universidade do Porto
A enviar comando -> USER anonymous
Resposta recebida com sucesso -> 331 Please specify the password.
A enviar comando -> PASS rcml23
Resposta recebida com sucesso -> 230 Login successful.
A enviar comando -> PASV
Resposta recebida com sucesso -> 227 Entering Passive Mode (193,136,37,8,149,174)
PORT: 38318
Socket connected port:4 ip:193.136.37.8
A enviar comando -> retr pub/robots.txt
Resposta recebida com sucesso -> 150 Opening BINARY mode data connection for pub/robots.txt (23 bytes).
Downloading robots.txt...
Download finished.
tux54:~/Desktop/rcom/Projeto-2/src#

```

Figura 1: Mensagens que são apresentadas no download de um ficheiro com sucesso.

## 3 Parte 2 – Configuração e análise da rede

Nesta segunda parte do projeto executaram-se diversas experiências, enunciadas no guião fornecido pelos professores, que permitiram no final obter uma configuração da rede de forma a ser possível a execução da aplicação. Abaixo serão explicitadas todas as experiências executadas.

As configurações usadas nas experiências estão listadas nas secções 6.3 e 6.4 dos Anexos.

### 3.1 Experiência 1 - Configuração de uma rede IP

#### 3.1.1 Objetivos da experiência

Esta experiência permite que dois computadores ligados através de um *switch* consigam comunicar um com o outro. Assim no final foi possível que o computador *tuxy1* e o computador *tuxy4* comunicassem entre si (*y* é o número da bancada onde está a ser executada a experiência).

#### 3.1.2 Arquitetura da rede

Como é possível observar na imagem da Figura 2 nesta experiência são ligados os *tuxy1* e o *tuxy4* através de um *switch*.

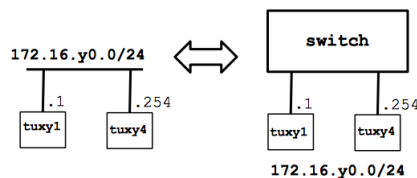


Figura 2: Arquitetura da rede da experiência 1.

#### 3.1.3 Análise dos *logs*

No log da figura 6.1.1 dos Anexos, é possível verificar a transmissão de pacotes entre o *tux31* e o *tux34*, o que significa que a conexão entre os dois foi estabelecida com sucesso.

### 3.2 Experiência 2

#### 3.2.1 Objetivos da experiência

Nesta experiência foram criadas as VLANs *vlan30* e *vlan31* num *switch*. O objetivo de serem criadas estas duas VLANs é permitir que uma rede seja dividida em vários *broadcast domains*. Desta forma é possível separar *hosts* que não devem ter acesso entre si.

### 3.2.2 Arquitetura da rede

Como é possível observar na imagem da Figura 3 nesta experiência são criadas as VLANs `vlan30` e `vlan31` num *switch* (na figura o `y` representa o número da bancada).

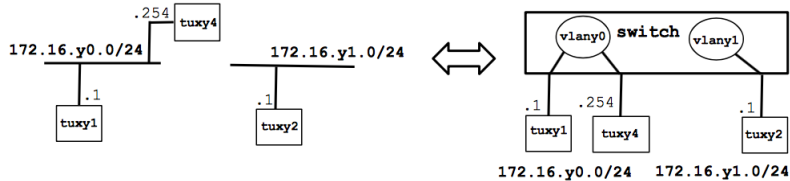


Figura 3: Arquitetura da rede da experiência 2.

### 3.2.3 Análise dos logs

Nos logs das figuras 6.1.2, 6.1.3 e 6.1.4 foi efetuado um ping broadcast a partir do `tux31`, verificando quais os endereços que estavam conectados na rede local no momento desta experiência. Foi efetuado ainda um outro ping broadcast (`ping -b 172.16.30.255`) a partir do mesmo `tux31`, e verificado que não existia conexão com o `tux32`, o que significa que a divisão instaurada pelas VLANs foi corretamente configurada.

## 3.3 Experiência 3

### 3.3.1 Objetivos da experiência

Nesta experiência adicionou-se a `eth1`, nova interface do `tux34`, na `vlan31`. Adicionaram-se também as rotas de comunicação entre o `tux31` e o `tux32`, apesar de este estar numa `vlan` diferente. Portanto o objetivo é permitir a comunicação entre os diferentes computadores independentemente da `vlan` onde estão conectados. Assim o `tux34` passa a funcionar como um *router*.

### 3.3.2 Arquitetura da rede

Como é possível observar na imagem da Figura 4 nesta experiência é adicionada a nova interface ao `tux34` e são adicionadas as rotas de comunicação entre o `tux31` e o `tux32`. Pode também concluir-se com a imagem que o `tux34` funcionará como um *router*.

### 3.3.3 Análise dos logs

Verifica-se no log da figura 6.1.6 a conexão entre o `tux31` e o `tux32`, através do `tux34`. Este último passará a funcionar como *router*, encaminhando os pacotes vindos do `tux31` para o `tux32` e vice-versa. É visível também a possibilidade de contactar o `tux31` com todos os elementos da rede, incluindo as interfaces `eth0` e `eth1` do `tux34` (figura 6.1.7)



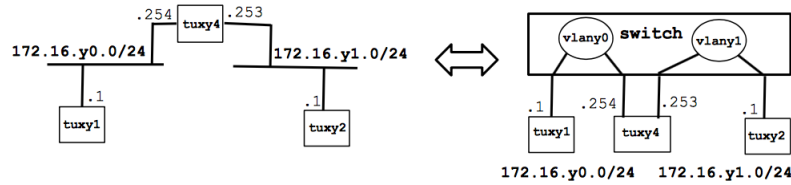


Figura 4: Arquitetura da rede da experiência 3.

### 3.4 Experiência 4

#### 3.4.1 Objetivos da experiência

Nesta experiência, foi adicionado o router Cisco (Rc) à rede das experiências anteriores, e ligado este à rede local do laboratório I321. Assim, este será o router para o tux34 e para o tux32, enquanto que o tux34 irá funcionar como router para o tux31.

#### 3.4.2 Arquitetura da rede

Como pode ser verificado na Figura 9, adicionando-se o router com a funcionalidade NAT foi possível conectar toda a rede à rede local do laboratório. O router foi configurado de modo a que o tráfego de/para os tux31 e tux32 fosse permitido, sendo que foi criada uma rota entre o tux34 e o router, de modo a que todos os computadores tivessem ligação com este.

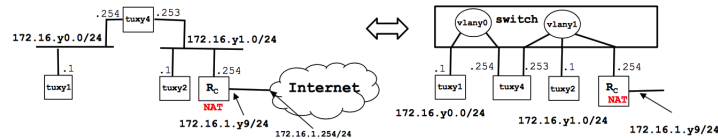


Figura 5: Arquitetura da rede da experiência 4.

#### 3.4.3 Análise dos logs

Foi efetuado, a partir do tux31, um ping em três terminais em simultâneo, um para cada destino pretendido: tux32, tux34 e o próprio router (figura 6.1.8), tendo-se verificado a conexão com sucesso em todos.

### 3.5 Experiência 5

#### 3.5.1 Objetivos da experiência

Na experiência 5, foi necessário configurar o serviço DNS no tux31, tux32 e tux34. O DNS converte uma string (o endereço do site a que se quer aceder) num endereço IP.

### 3.5.2 Arquitetura da rede

Para a experiência, foi usado o servidor *liza.netlab.fe.up.pt* (172.16.1.2), indicado no guião, para a sua correta configuração.

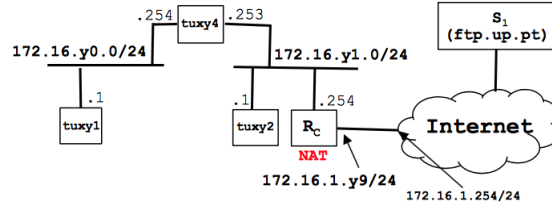


Figura 6: Arquitetura da rede da experiência 5.

### 3.5.3 Análise dos logs

Foi testada a ligação a partir do tux31 (figura 6.1.9) e do tux32 (figura 6.1.10) a um website (por exemplo, *www.google.pt*), tendo sido verificada a conexão com êxito em ambos os computadores.

## 3.6 Experiência 6

### 3.6.1 Objetivos da experiência

Para esta experiência, foi usada a aplicação desenvolvida, indicada na parte 1 deste relatório, assim como em todas as configurações usadas em todas as experiências até este ponto. O objetivo seria fazer o download de um ficheiro usando o protocolo FTP.

### 3.6.2 Arquitetura da rede

Nesta rede, foi usada a aplicação de download no tux31 e no tux32, sendo que estes tinham sido corretamente configurados nas experiências anteriores para ser conseguido o acesso à rede do laboratório e, por consequência à Internet.

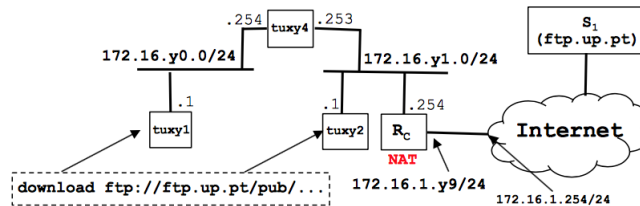


Figura 7: Arquitetura da rede da experiência 6.

### 3.6.3 Análise dos logs

Foram usadas 2 abordagens para esta experiência. Na 1ª, foi usada a aplicação desenvolvida para fazer o download de um ficheiro somente no `tux31` (figuras 6.1.11 e 6.1.12). Na 2ª abordagem, foi iniciado o download do mesmo ficheiro no mesmo `tux` mas, após alguns segundos, foi iniciado o download do mesmo ficheiro mas no `tux32`. Verificou-se que o número de pacotes transmitidos por segundo reduziu consideravelmente a partir do momento em que foi iniciado o download do `tux32`, de modo a permitir que ambos os computadores pudessem efetuar o download com sucesso, sem colocar em causa a integridade da rede. Esta situação é visível no gráfico da imagem 6.1.13 dos Anexos.

## 4 Conclusões

O conjunto de experiências, anteriormente detalhado, feito após a configuração da rede local, permitiu pôr em prática os conceitos teóricos relativos a redes de computadores, nomeadamente ao nível do funcionamento da NAT e do reencaminhamento de pacotes, cumprindo os objetivos previstos para o trabalho.

De entre os diversos protocolos utilizados para estabelecer a comunicação entre redes, foram implementados protocolos FTP, DNS, TCP e ARP, destacando-se o FTP, que foi usado para permitir o download remoto de um ficheiro, tendo sido também desenvolvida uma aplicação em linguagem C para o efeito.

No entanto, tendo em conta que este trabalho consistiu numa rede com uma ligação de 2 VLANs num switch, apesar do interesse académico deste desenvolvimento, existiria ainda interesse numa abordagem mais complexa e exaustiva destes conceitos, quer em termos de número de ligações, quer em termos do tipo e quantidade de dados a transmitir, que permita aplicar tais ferramentas num contexto real.

## 5 Referências

### Referências

- [1] File transfer protocol (ftp). <https://www.ietf.org/rfc/rfc959.txt>.
- [2] Slides da unidade curricular redes de computadores.
- [3] David J. Wetherall Andrew S. Tanenbaum. *Computer Networks*. Pearson Prentice Hall, 2011.
- [4] W. Richard Stevens. *TCP/IP illustrated, Volume 1: The Protocols*. Addison Wesley, 1994.

## 6 Anexos

### 6.1 Logs

#### 6.1.1 Experiência 1 - Ping para o tux34 a partir do tux31

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	Cisco_3a:fa:83	Spanning-tree-(for-br	STP	60	Conf. Root = 32768/30/fc:fb:fb:3a:fa:80 Cos
2	0.855766000	Cisco_3a:fa:83	Cisco_3a:fa:83	LOOP	60	Reply
3	2.004921000	Cisco_3a:fa:83	Spanning-tree-(for-br	STP	60	Conf. Root = 32768/30/fc:fb:fb:3a:fa:80 Cos
4	4.009661000	Cisco_3a:fa:83	Spanning-tree-(for-br	STP	60	Conf. Root = 32768/30/fc:fb:fb:3a:fa:80 Cos
5	4.666125000	172.16.30.1	172.16.30.254	ICMP	98	Echo (ping) request id=0x73bd, seq=1/256, t
6	4.666317000	172.16.30.254	172.16.30.1	ICMP	98	Echo (ping) reply id=0x73bd, seq=1/256, t
7	5.665129000	172.16.30.1	172.16.30.254	ICMP	98	Echo (ping) request id=0x73bd, seq=2/512, t
8	5.665338000	172.16.30.254	172.16.30.1	ICMP	98	Echo (ping) reply id=0x73bd, seq=2/512, t
9	6.014373000	Cisco_3a:fa:83	Spanning-tree-(for-br	STP	60	Conf. Root = 32768/30/fc:fb:fb:3a:fa:80 Cos
10	6.664562000	172.16.30.1	172.16.30.254	ICMP	98	Echo (ping) request id=0x73bd, seq=3/768, t
11	6.664759000	172.16.30.254	172.16.30.1	ICMP	98	Echo (ping) reply id=0x73bd, seq=3/768, t
12	7.664560000	172.16.30.1	172.16.30.254	ICMP	98	Echo (ping) request id=0x73bd, seq=4/1024, t
13	7.664719000	172.16.30.254	172.16.30.1	ICMP	98	Echo (ping) reply id=0x73bd, seq=4/1024, t
14	8.019075000	Cisco_3a:fa:83	Spanning-tree-(for-br	STP	60	Conf. Root = 32768/30/fc:fb:fb:3a:fa:80 Cos
15	8.664566000	172.16.30.1	172.16.30.254	ICMP	98	Echo (ping) request id=0x73bd, seq=5/1280, t
16	8.664746000	172.16.30.254	172.16.30.1	ICMP	98	Echo (ping) reply id=0x73bd, seq=5/1280, t
17	9.664572000	172.16.30.1	172.16.30.254	ICMP	98	Echo (ping) request id=0x73bd, seq=6/1536, t
18	9.664728000	172.16.30.254	172.16.30.1	ICMP	98	Echo (ping) reply id=0x73bd, seq=6/1536, t
19	9.669218000	Hewlett-_5a:7d:74	G-ProCom_8b:e4:4d	ARP	60	who has 172.16.30.1? Tell 172.16.30.254
20	9.669237000	G-ProCom_8b:e4:4d	Hewlett-_5a:7d:74	ARP	42	172.16.30.1 is at 00:0f:fe:8b:e4:4d
21	10.02401200	Cisco_3a:fa:83	Spanning-tree-(for-br	STP	60	Conf. Root = 32768/30/fc:fb:fb:3a:fa:80 Cos
22	10.86795400	Cisco_3a:fa:83	Cisco_3a:fa:83	LOOP	60	Reply

#### 6.1.2 Experiência 2 - Ping broadcast a partir do tux31

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	Cisco_3a:f6:03	Spanning-tree-(for-br	STP	60	Conf. Root = 32768/30/fc:fb:fb:3a:f6:00 Cost
2	2.004901000	Cisco_3a:f6:03	Spanning-tree-(for-br	STP	60	Conf. Root = 32768/30/fc:fb:fb:3a:f6:00 Cost
3	4.009810000	Cisco_3a:f6:03	Spanning-tree-(for-br	STP	60	Conf. Root = 32768/30/fc:fb:fb:3a:f6:00 Cost
4	6.014566000	Cisco_3a:f6:03	Spanning-tree-(for-br	STP	60	Conf. Root = 32768/30/fc:fb:fb:3a:f6:00 Cost
5	7.922898000	Cisco_3a:f6:03	Cisco_3a:f6:03	LOOP	60	Reply
6	8.019524000	Cisco_3a:f6:03	Spanning-tree-(for-br	STP	60	Conf. Root = 32768/30/fc:fb:fb:3a:f6:00 Cost
7	8.209823000	172.16.30.1	172.16.31.255	ICMP	98	Echo (ping) request id=0x0d5f, seq=1/256, tt
8	8.210222000	172.16.30.254	172.16.30.1	ICMP	98	Echo (ping) reply id=0x0d5f, seq=1/256, tt
9	9.209777000	172.16.30.1	172.16.31.255	ICMP	98	Echo (ping) request id=0x0d5f, seq=2/512, tt
10	9.210012000	172.16.30.254	172.16.30.1	ICMP	98	Echo (ping) reply id=0x0d5f, seq=2/512, tt
11	10.02427500	Cisco_3a:f6:03	Spanning-tree-(for-br	STP	60	Conf. Root = 32768/30/fc:fb:fb:3a:f6:00 Cost
12	10.20978600	172.16.30.1	172.16.31.255	ICMP	98	Echo (ping) request id=0x0d5f, seq=3/768, tt
13	10.21019200	172.16.30.254	172.16.30.1	ICMP	98	Echo (ping) reply id=0x0d5f, seq=3/768, tt
14	11.20978000	172.16.30.1	172.16.31.255	ICMP	98	Echo (ping) request id=0x0d5f, seq=4/1024, tt
15	11.21001500	172.16.30.254	172.16.30.1	ICMP	98	Echo (ping) reply id=0x0d5f, seq=4/1024, tt
16	12.03417800	Cisco_3a:f6:03	Spanning-tree-(for-br	STP	60	Conf. Root = 32768/30/fc:fb:fb:3a:f6:00 Cost
17	12.20978200	172.16.30.1	172.16.31.255	ICMP	98	Echo (ping) request id=0x0d5f, seq=5/1280, tt
18	12.21023000	172.16.30.254	172.16.30.1	ICMP	98	Echo (ping) reply id=0x0d5f, seq=5/1280, tt
19	13.20978700	172.16.30.1	172.16.31.255	ICMP	98	Echo (ping) request id=0x0d5f, seq=6/1536, tt
20	13.21002100	172.16.30.254	172.16.30.1	ICMP	98	Echo (ping) reply id=0x0d5f, seq=6/1536, tt
21	13.21092200	Hewlett-_c3:78:70	G-ProCom_8b:e4:a7	ARP	60	who has 172.16.30.1? Tell 172.16.30.254
22	13.21093900	G-ProCom_8b:e4:a7	Hewlett-_c3:78:70	ARP	42	172.16.30.1 is at 00:0f:fe:8b:e4:a7
23	14.03400700	Cisco_3a:f6:03	Spanning-tree-(for-br	STP	60	Conf. Root = 32768/30/fc:fb:fb:3a:f6:00 Cost

### 6.1.3 Experiência 2 - Ping broadcast a partir do tux31, visto do tux32

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	Cisco_3a:f6:04	Spanning-tree-(for-br	STP	60	Conf. Root = 32768/31/fc:fb:fb:3a:f6:00 Cost = 0 Port = 0x8004
2	2.004893000	Cisco_3a:f6:04	Spanning-tree-(for-br	STP	60	Conf. Root = 32768/31/fc:fb:fb:3a:f6:00 Cost = 0 Port = 0x8004
3	3.467913000	172.16.31.1	172.16.1.1	DNS	81	Standard query 0x4090 AAAA tux52.netlab.fe.up.pt
4	3.468932000	172.16.1.1	172.16.31.1	DNS	141	Standard query response 0x4090
5	3.469022000	172.16.31.1	172.16.1.1	DNS	74	Standard query 0x5a41 AAAA tux52.fe.up.pt
6	3.469777000	172.16.1.1	172.16.31.1	DNS	119	Standard query response 0x5a41 No such name
7	3.469818000	172.16.31.1	172.16.1.1	DNS	65	Standard query 0xe9dd AAAA tux52
8	3.471924000	172.16.1.1	172.16.31.1	DNS	140	Standard query response 0xe9dd No such name
9	3.560019000	172.16.31.1	172.16.1.1	DNS	81	Standard query 0x3be7 AAAA tux52.netlab.fe.up.pt
10	3.560863000	172.16.1.1	172.16.31.1	DNS	141	Standard query response 0x3be7
11	3.560927000	172.16.31.1	172.16.1.1	DNS	74	Standard query 0x3265 AAAA tux52.fe.up.pt
12	3.561673000	172.16.1.1	172.16.31.1	DNS	119	Standard query response 0x3265 No such name
13	3.561713000	172.16.31.1	172.16.1.1	DNS	65	Standard query 0xa720 AAAA tux52
14	3.563170000	172.16.1.1	172.16.31.1	DNS	140	Standard query response 0xa720 No such name
15	4.009802000	Cisco_3a:f6:04	Spanning-tree-(for-br	STP	60	Conf. Root = 32768/31/fc:fb:fb:3a:f6:00 Cost = 0 Port = 0x8004
16	4.175074000	Cisco_3a:f6:04	Cisco_3a:f6:04	LOOP	60	Reply
17	6.014628000	Cisco_3a:f6:04	Spanning-tree-(for-br	STP	60	Conf. Root = 32768/31/fc:fb:fb:3a:f6:00 Cost = 0 Port = 0x8004
18	6.627122000	Cisco_3a:f6:04	CDP/VTP/DTP/PAGP/UDLD	CDP	435	Device ID: tux-sw5 Port ID: FastEthernet0/2
19	8.019447000	Cisco_3a:f6:04	Spanning-tree-(for-br	STP	60	Conf. Root = 32768/31/fc:fb:fb:3a:f6:00 Cost = 0 Port = 0x8004
20	8.472830000	Hewlett_61:2f:d6	Kye_08:d5:b0	ARP	42	Who has 172.16.31.253? Tell 172.16.31.1
21	8.472949000	Kye_08:d5:b0	Hewlett_61:2f:d6	ARP	60	172.16.31.253 is at 00:c0:df:08:d5:b0
22	10.024433000	Cisco_3a:f6:04	Spanning-tree-(for-br	STP	60	Conf. Root = 32768/31/fc:fb:fb:3a:f6:00 Cost = 0 Port = 0x8004
23	12.029254000	Cisco_3a:f6:04	Spanning-tree-(for-br	STP	60	Conf. Root = 32768/31/fc:fb:fb:3a:f6:00 Cost = 0 Port = 0x8004
24	14.034358000	Cisco_3a:f6:04	Spanning-tree-(for-br	STP	60	Conf. Root = 32768/31/fc:fb:fb:3a:f6:00 Cost = 0 Port = 0x8004
25	14.172388000	Cisco_3a:f6:04	Cisco_3a:f6:04	LOOP	60	Reply
26	16.039055000	Cisco_3a:f6:04	Spanning-tree-(for-br	STP	60	Conf. Root = 32768/31/fc:fb:fb:3a:f6:00 Cost = 0 Port = 0x8004

### 6.1.4 Experiência 2 - Ping broadcast a partir do tux31, visto do tux34

Interface id	Source	Destination	Protocol	Info
0	Cisco_3a:f6:05	Spanning-tree-(for-br	STP	Conf. Root = 32768/30/fc:fb:fb:3a:f6:00 Cost = 0 Port = 0x8005
0	Cisco_3a:f6:05	Cisco_3a:f6:05	LOOP	Reply
0	Cisco_3a:f6:05	Spanning-tree-(for-br	STP	Conf. Root = 32768/30/fc:fb:fb:3a:f6:00 Cost = 0 Port = 0x8005
0	172.16.30.1	172.16.31.255	ICMP	Echo (ping) request id=0x0d5f, seq=1/256, ttl=64 (no response found!)
0	172.16.30.254	172.16.30.1	ICMP	Echo (ping) reply id=0x0d5f, seq=1/256, ttl=64
0	172.16.30.1	172.16.31.255	ICMP	Echo (ping) request id=0x0d5f, seq=2/512, ttl=64 (no response found!)
0	172.16.30.254	172.16.30.1	ICMP	Echo (ping) reply id=0x0d5f, seq=2/512, ttl=64
0	Cisco_3a:f6:05	Spanning-tree-(for-br	STP	Conf. Root = 32768/30/fc:fb:fb:3a:f6:00 Cost = 0 Port = 0x8005
0	172.16.30.1	172.16.31.255	ICMP	Echo (ping) request id=0x0d5f, seq=3/768, ttl=64 (no response found!)
0	172.16.30.254	172.16.30.1	ICMP	Echo (ping) reply id=0x0d5f, seq=3/768, ttl=64
0	172.16.30.1	172.16.31.255	ICMP	Echo (ping) request id=0x0d5f, seq=4/1024, ttl=64 (no response found!)
0	172.16.30.254	172.16.30.1	ICMP	Echo (ping) reply id=0x0d5f, seq=4/1024, ttl=64
0	Cisco_3a:f6:05	Spanning-tree-(for-br	STP	Conf. Root = 32768/30/fc:fb:fb:3a:f6:00 Cost = 0 Port = 0x8005
0	172.16.30.1	172.16.31.255	ICMP	Echo (ping) request id=0x0d5f, seq=5/1280, ttl=64 (no response found!)
0	172.16.30.254	172.16.30.1	ICMP	Echo (ping) reply id=0x0d5f, seq=5/1280, ttl=64
0	172.16.30.1	172.16.31.255	ICMP	Echo (ping) request id=0x0d5f, seq=6/1536, ttl=64 (no response found!)
0	172.16.30.254	172.16.30.1	ICMP	Echo (ping) reply id=0x0d5f, seq=6/1536, ttl=64
0	Hewlett_c3:78:70	G-ProCom_8b:e4:a7	ARP	Who has 172.16.30.1? Tell 172.16.30.254
0	G-ProCom_8b:e4:a7	Hewlett_c3:78:70	ARP	172.16.30.1 is at 00:0f:fe:8b:e4:a7
0	Cisco_3a:f6:05	Spanning-tree-(for-br	STP	Conf. Root = 32768/30/fc:fb:fb:3a:f6:00 Cost = 0 Port = 0x8005
0	Cisco_3a:f6:05	Spanning-tree-(for-br	STP	Conf. Root = 32768/30/fc:fb:fb:3a:f6:00 Cost = 0 Port = 0x8005
0	Cisco_3a:f6:05	Cisco_3a:f6:05	LOOP	Reply

### 6.1.5 Experiência 2 - Ping para o tux32 e de seguida para o tux34 a partir do tux31

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	Cisco_3a:f6:03	Spanning-tree-(for-br	STP	60	Conf. Root = 32768/30/fc:fb:fb:3a:f6:00 Cos
2	1.481531000	172.16.30.1	172.16.30.254	ICMP	98	Echo (ping) request id=0x0bfd, seq=1/256, t
3	1.481783000	172.16.30.254	172.16.30.1	ICMP	98	Echo (ping) reply id=0x0bfd, seq=1/256, t
4	2.010250000	Cisco_3a:f6:03	Spanning-tree-(for-br	STP	60	Conf. Root = 32768/30/fc:fb:fb:3a:f6:00 Cos
5	2.481288000	172.16.30.1	172.16.30.254	ICMP	98	Echo (ping) request id=0x0bfd, seq=2/512, t
6	2.481536000	172.16.30.254	172.16.30.1	ICMP	98	Echo (ping) reply id=0x0bfd, seq=2/512, t
7	3.481302000	172.16.30.1	172.16.30.254	ICMP	98	Echo (ping) request id=0x0bfd, seq=3/768, t
8	3.481553000	172.16.30.254	172.16.30.1	ICMP	98	Echo (ping) reply id=0x0bfd, seq=3/768, t
9	4.009952000	Cisco_3a:f6:03	Spanning-tree-(for-br	STP	60	Conf. Root = 32768/30/fc:fb:fb:3a:f6:00 Cos
10	4.481292000	172.16.30.1	172.16.30.254	ICMP	98	Echo (ping) request id=0x0bfd, seq=4/1024,
11	4.481539000	172.16.30.254	172.16.30.1	ICMP	98	Echo (ping) reply id=0x0bfd, seq=4/1024,
12	4.894786000	Cisco_3a:f6:03	Cisco_3a:f6:03	LOOP	60	Reply
13	5.481301000	172.16.30.1	172.16.30.254	ICMP	98	Echo (ping) request id=0x0bfd, seq=5/1280,
14	5.481551000	172.16.30.254	172.16.30.1	ICMP	98	Echo (ping) reply id=0x0bfd, seq=5/1280,
15	6.014765000	Cisco_3a:f6:03	Spanning-tree-(for-br	STP	60	Conf. Root = 32768/30/fc:fb:fb:3a:f6:00 Cos
16	6.481292000	172.16.30.1	172.16.30.254	ICMP	98	Echo (ping) request id=0x0bfd, seq=6/1536,
17	6.481539000	172.16.30.254	172.16.30.1	ICMP	98	Echo (ping) reply id=0x0bfd, seq=6/1536,
18	7.608749000	Cisco_3a:f6:03	CDP/VTP/DTP/PAGP/UDLD	CDP	435	Device ID: tux-sw5 Port ID: FastEthernet0/1
19	8.024452000	Cisco_3a:f6:03	Spanning-tree-(for-br	STP	60	Conf. Root = 32768/30/fc:fb:fb:3a:f6:00 Cos
20	10.024537000	Cisco_3a:f6:03	Spanning-tree-(for-br	STP	60	Conf. Root = 32768/30/fc:fb:fb:3a:f6:00 Cos
21	11.873280000	172.16.30.1	172.16.31.1	ICMP	98	Echo (ping) request id=0x0c04, seq=1/256, t
22	11.873830000	172.16.31.1	172.16.30.1	ICMP	98	Echo (ping) reply id=0x0c04, seq=1/256, t
23	12.029339000	Cisco_3a:f6:03	Spanning-tree-(for-br	STP	60	Conf. Root = 32768/30/fc:fb:fb:3a:f6:00 Cos
24	12.873291000	172.16.30.1	172.16.31.1	ICMP	98	Echo (ping) request id=0x0c04, seq=2/512, t
25	12.873809000	172.16.31.1	172.16.30.1	ICMP	98	Echo (ping) reply id=0x0c04, seq=2/512, t
26	13.873292000	172.16.30.1	172.16.31.1	ICMP	98	Echo (ping) request id=0x0c04, seq=3/768, t
27	13.873808000	172.16.31.1	172.16.30.1	ICMP	98	Echo (ping) reply id=0x0c04, seq=3/768, t

### 6.1.6 Experiência 3 - Ping para o tux32 a partir do tux31, visto do tux34

Interface id	Source	Destination	Protocol	Info
0 Cisco_3a:f6:05		Spanning-tree-(for-br	STP	Conf. Root = 32768/30/fc:fb:fb:3a:f6:00 Cost = 0 Port = 0x8005
0 172.16.30.1		172.16.31.1	ICMP	Echo (ping) request id=0x11a, seq=1/256, ttl=64 (no response found!)
0 172.16.31.1		172.16.30.1	ICMP	Echo (ping) reply id=0x11a, seq=1/256, ttl=63 (request in 6)
0 Cisco_3a:f6:05		CDP/VTP/DTP/PAGP/UDLD	CDP	Device ID: tux-sw5 Port ID: FastEthernet0/3
0 Cisco_3a:f6:05		Spanning-tree-(for-br	STP	Conf. Root = 32768/30/fc:fb:fb:3a:f6:00 Cost = 0 Port = 0x8005
0 172.16.30.1		172.16.31.1	ICMP	Echo (ping) request id=0x11a, seq=2/512, ttl=64 (no response found!)
0 172.16.31.1		172.16.30.1	ICMP	Echo (ping) reply id=0x11a, seq=2/512, ttl=63 (request in 10)
0 172.16.30.1		172.16.31.1	ICMP	Echo (ping) request id=0x11a, seq=3/768, ttl=64 (no response found!)
0 172.16.31.1		172.16.30.1	ICMP	Echo (ping) reply id=0x11a, seq=3/768, ttl=63 (request in 12)
0 Cisco_3a:f6:05		Spanning-tree-(for-br	STP	Conf. Root = 32768/30/fc:fb:fb:3a:f6:00 Cost = 0 Port = 0x8005
0 172.16.30.1		172.16.31.1	ICMP	Echo (ping) request id=0x11a, seq=4/1024, ttl=64 (no response found!)
0 172.16.31.1		172.16.30.1	ICMP	Echo (ping) reply id=0x11a, seq=4/1024, ttl=63 (request in 15)
0 172.16.30.1		172.16.31.1	ICMP	Echo (ping) request id=0x11a, seq=5/1280, ttl=64 (no response found!)
0 172.16.31.1		172.16.30.1	ICMP	Echo (ping) reply id=0x11a, seq=5/1280, ttl=63 (request in 17)
0 Cisco_3a:f6:05		Spanning-tree-(for-br	STP	Conf. Root = 32768/30/fc:fb:fb:3a:f6:00 Cost = 0 Port = 0x8005
0 172.16.30.1		172.16.31.1	ICMP	Echo (ping) request id=0x11a, seq=6/1536, ttl=64 (no response found!)
0 172.16.31.1		172.16.30.1	ICMP	Echo (ping) reply id=0x11a, seq=6/1536, ttl=63 (request in 20)
0 Hewlett-_c3:78:70		G-ProCom_8b:e4:a7	APP	Who has 172.16.30.1? Tell 172.16.30.254
0 G-ProCom_8b:e4:a7		Hewlett-_c3:78:70	APP	172.16.30.1 is at 00:0f:fe:8b:e4:a7
0 Cisco_3a:f6:05		Cisco_3a:f6:05	LOOP	Reply
0 Cisco_3a:f6:05		Spanning-tree-(for-br	STP	Conf. Root = 32768/30/fc:fb:fb:3a:f6:00 Cost = 0 Port = 0x8005
0 Cisco_3a:f6:05		Spanning-tree-(for-br	STP	Conf. Root = 32768/30/fc:fb:fb:3a:f6:00 Cost = 0 Port = 0x8005

### 6.1.7 Experiência 3 - Ping para o tux32, tux34.eth0 e tux34.eth1 em simultâneo a partir do tux31

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	Cisco_3a:f6:03	Spanning-tree-(for-br	STP	60	Conf. Root = 32768/30/fc:fb:fb:3a:f6:00 Cos
2	2.004898000	Cisco_3a:f6:03	Spanning-tree-(for-br	STP	60	Conf. Root = 32768/30/fc:fb:fb:3a:f6:00 Cos
3	3.863364000	172.16.30.1	172.16.30.254	ICMP	98	Echo (ping) request id=0x1095, seq=1/256, t
4	3.863548000	172.16.30.254	172.16.30.1	ICMP	98	Echo (ping) reply id=0x1095, seq=1/256, t
5	4.014933000	Cisco_3a:f6:03	Spanning-tree-(for-br	STP	60	Conf. Root = 32768/30/fc:fb:fb:3a:f6:00 Cos
6	4.862801000	172.16.30.1	172.16.30.254	ICMP	98	Echo (ping) request id=0x1095, seq=2/512, t
7	4.863008000	172.16.30.254	172.16.30.1	ICMP	98	Echo (ping) reply id=0x1095, seq=2/512, t
8	5.862813000	172.16.30.1	172.16.30.254	ICMP	98	Echo (ping) request id=0x1095, seq=3/768, t
9	5.863021000	172.16.30.254	172.16.30.1	ICMP	98	Echo (ping) reply id=0x1095, seq=3/768, t
10	6.014728000	Cisco_3a:f6:03	Spanning-tree-(for-br	STP	60	Conf. Root = 32768/30/fc:fb:fb:3a:f6:00 Cos
11	6.207212000	172.16.30.1	172.16.31.253	ICMP	98	Echo (ping) request id=0x1099, seq=1/256, t
12	6.207371000	172.16.31.253	172.16.30.1	ICMP	98	Echo (ping) reply id=0x1099, seq=1/256, t
13	6.864173000	172.16.30.1	172.16.30.254	ICMP	98	Echo (ping) request id=0x1095, seq=4/1024, t
14	6.864381000	172.16.30.254	172.16.30.1	ICMP	98	Echo (ping) reply id=0x1095, seq=4/1024, t
15	7.206830000	172.16.30.1	172.16.31.253	ICMP	98	Echo (ping) request id=0x1099, seq=2/512, t
16	7.207211000	172.16.31.253	172.16.30.1	ICMP	98	Echo (ping) reply id=0x1099, seq=2/512, t
17	7.668118000	Cisco_3a:f6:03	Cisco_3a:f6:03	LOOP	60	Reply
18	7.863170000	172.16.30.1	172.16.30.254	ICMP	98	Echo (ping) request id=0x1095, seq=5/1280, t
19	7.863372000	172.16.30.254	172.16.30.1	ICMP	98	Echo (ping) reply id=0x1095, seq=5/1280, t
20	8.019566000	Cisco_3a:f6:03	Spanning-tree-(for-br	STP	60	Conf. Root = 32768/30/fc:fb:fb:3a:f6:00 Cos
21	8.206813000	172.16.30.1	172.16.31.253	ICMP	98	Echo (ping) request id=0x1099, seq=3/768, t
22	8.206966000	172.16.31.253	172.16.30.1	ICMP	98	Echo (ping) reply id=0x1099, seq=3/768, t
23	8.470976000	172.16.30.1	172.16.31.1	ICMP	98	Echo (ping) request id=0x109a, seq=1/256, t
24	8.471499000	172.16.31.1	172.16.30.1	ICMP	98	Echo (ping) reply id=0x109a, seq=1/256, t
25	8.862813000	172.16.30.1	172.16.30.254	ICMP	98	Echo (ping) request id=0x1095, seq=6/1536, t
26	8.863022000	172.16.30.254	172.16.30.1	ICMP	98	Echo (ping) reply id=0x1095, seq=6/1536, t
27	8.877498000	Hewlett_c3:78:70	G-ProCom_8b:e4:a7	ARP	60	Who has 172.16.30.1? Tell 172.16.30.254

### 6.1.8 Experiência 4 - Ping para o tux32, tux34 e Rc em simultâneo a partir do tux31

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	Cisco_3a:f6:03	Spanning-tree-(for-br	STP	60	Conf. Root = 32768/30/fc:fb:fb:3a:f6:00 Cos
2	1.893631000	172.16.30.1	172.16.1.39	ICMP	98	Echo (ping) request id=0x1245, seq=1/256, t
3	1.894424000	172.16.1.39	172.16.30.1	ICMP	98	Echo (ping) reply id=0x1245, seq=1/256, t
4	1.999960000	Cisco_3a:f6:03	Spanning-tree-(for-br	STP	60	Conf. Root = 32768/30/fc:fb:fb:3a:f6:00 Cos
5	2.893245000	172.16.30.1	172.16.1.39	ICMP	98	Echo (ping) request id=0x1245, seq=2/512, t
6	2.893974000	172.16.1.39	172.16.30.1	ICMP	98	Echo (ping) reply id=0x1245, seq=2/512, t
7	3.085399000	172.16.30.1	172.16.30.254	ICMP	98	Echo (ping) request id=0x1246, seq=1/256, t
8	3.085556000	172.16.30.254	172.16.30.1	ICMP	98	Echo (ping) reply id=0x1246, seq=1/256, t
9	3.893252000	172.16.30.1	172.16.1.39	ICMP	98	Echo (ping) request id=0x1245, seq=3/768, t
10	3.893871000	172.16.30.1	172.16.31.1	ICMP	98	Echo (ping) request id=0x1247, seq=1/256, t
11	3.893934000	172.16.1.39	172.16.30.1	ICMP	98	Echo (ping) reply id=0x1245, seq=3/768, t
12	3.894352000	172.16.31.1	172.16.30.1	ICMP	98	Echo (ping) reply id=0x1247, seq=1/256, t
13	4.004817000	Cisco_3a:f6:03	Spanning-tree-(for-br	STP	60	Conf. Root = 32768/30/fc:fb:fb:3a:f6:00 Cos
14	4.086262000	172.16.30.1	172.16.30.254	ICMP	98	Echo (ping) request id=0x1246, seq=2/512, t
15	4.086462000	172.16.30.254	172.16.30.1	ICMP	98	Echo (ping) reply id=0x1246, seq=2/512, t
16	4.893249000	172.16.30.1	172.16.1.39	ICMP	98	Echo (ping) request id=0x1245, seq=4/1024, t
17	4.893287000	172.16.30.1	172.16.31.1	ICMP	98	Echo (ping) request id=0x1247, seq=2/512, t
18	4.893770000	172.16.31.1	172.16.30.1	ICMP	98	Echo (ping) reply id=0x1247, seq=2/512, t
19	4.893812000	172.16.1.39	172.16.30.1	ICMP	98	Echo (ping) reply id=0x1245, seq=4/1024, t
20	5.085261000	172.16.30.1	172.16.30.254	ICMP	98	Echo (ping) request id=0x1246, seq=3/768, t
21	5.085413000	172.16.30.254	172.16.30.1	ICMP	98	Echo (ping) reply id=0x1246, seq=3/768, t
22	5.893250000	172.16.30.1	172.16.31.1	ICMP	98	Echo (ping) request id=0x1247, seq=3/768, t
23	5.893287000	172.16.30.1	172.16.1.39	ICMP	98	Echo (ping) request id=0x1245, seq=5/1280, t
24	5.893504000	172.16.31.1	172.16.30.1	ICMP	98	Echo (ping) reply id=0x1247, seq=3/768, t
25	5.893967000	172.16.1.39	172.16.30.1	ICMP	98	Echo (ping) reply id=0x1245, seq=5/1280, t
26	6.014698000	Cisco_3a:f6:03	Spanning-tree-(for-br	STP	60	Conf. Root = 32768/30/fc:fb:fb:3a:f6:00 Cos
27	6.085247000	172.16.30.1	172.16.30.254	ICMP	98	Echo (ping) request id=0x1246, seq=4/1024, t

### 6.1.9 Experiência 5 - Ping para *www.google.pt* a partir do tux31

No.	Time	Source	Destination	Protocol	Length	Info
3	4.009422000	Cisco_3a:f6:03	Spanning-tree (for-br	STP	60	Conf. Root = 32768/30/fc:fb:fb:3a:f6:00 Cost = 0 Port = 0x8004
4	4.216029000	Cisco_3a:f6:03	Cisco_3a:f6:03	LOOP	60	Reply
5	5.313999000	172.16.30.1	172.16.1.1	DNS	73	Standard query 0x01e0 A www.google.pt
6	5.315396000	172.16.1.1	172.16.30.1	DNS	475	Standard query response 0x01e0 A 194.210.238.155 A 194.210.238.163 A 194.210.238.165 A 194.210.238.167
7	5.315628000	172.16.30.1	194.210.238.155	ICMP	98	Echo (ping) request id=0x12fe, seq=1/256, ttl=64 (reply in 5)
8	5.322628000	194.210.238.155	172.16.30.1	ICMP	98	Echo (ping) reply id=0x12fe, seq=1/256, ttl=55 (request in 4)
9	5.322801000	172.16.30.1	172.16.1.1	DNS	88	Standard query 0xbdc PTR 155.238.210.194.in-addr.arpa
10	5.330400000	172.16.1.1	172.16.30.1	DNS	147	Standard query response 0xbdc No such name
11	6.019444000	Cisco_3a:f6:03	Spanning-tree (for-br	STP	60	Conf. Root = 32768/30/fc:fb:fb:3a:f6:00 Cost = 0 Port = 0x8004
12	6.318289000	172.16.30.1	194.210.238.155	ICMP	98	Echo (ping) request id=0x12fe, seq=2/512, ttl=64 (no response found!)
13	6.324055000	194.210.238.155	172.16.30.1	ICMP	98	Echo (ping) reply id=0x12fe, seq=2/512, ttl=55 (request in 9)
14	7.320174000	172.16.30.1	194.210.238.155	ICMP	98	Echo (ping) request id=0x12fe, seq=3/768, ttl=64 (no response found!)
15	7.325931000	194.210.238.155	172.16.30.1	ICMP	98	Echo (ping) reply id=0x12fe, seq=3/768, ttl=55 (request in 11)
16	8.019224000	Cisco_3a:f6:03	Spanning-tree (for-br	STP	60	Conf. Root = 32768/30/fc:fb:fb:3a:f6:00 Cost = 0 Port = 0x8004
17	8.322049000	172.16.30.1	194.210.238.155	ICMP	98	Echo (ping) request id=0x12fe, seq=4/1024, ttl=64 (reply in 15)
18	8.327846000	194.210.238.155	172.16.30.1	ICMP	98	Echo (ping) reply id=0x12fe, seq=4/1024, ttl=55 (request in 14)
19	9.329862000	172.16.30.1	194.210.238.155	ICMP	98	Echo (ping) request id=0x12fe, seq=5/1280, ttl=64 (reply in 17)
20	9.329732000	194.210.238.155	172.16.30.1	ICMP	98	Echo (ping) reply id=0x12fe, seq=5/1280, ttl=55 (request in 16)
21	10.024077000	Cisco_3a:f6:03	Spanning-tree (for-br	STP	60	Conf. Root = 32768/30/fc:fb:fb:3a:f6:00 Cost = 0 Port = 0x8004
22	10.323520000	Hewlett_c3:78:70	G-ProCom_8b:e4:a7	ARP	60	Who has 172.16.30.1? Tell 172.16.30.254
23	10.323541000	G-ProCom_8b:e4:a7	Hewlett_c3:78:70	ARP	42	172.16.30.1 is at 00:0f:fe:8b:e4:a7
24	10.325801000	172.16.30.1	194.210.238.155	ICMP	98	Echo (ping) request id=0x12fe, seq=6/1536, ttl=64 (reply in 20)
25	10.331511000	194.210.238.155	172.16.30.1	ICMP	98	Echo (ping) reply id=0x12fe, seq=6/1536, ttl=55 (request in 19)
26	11.327651000	172.16.30.1	194.210.238.155	ICMP	98	Echo (ping) request id=0x12fe, seq=7/1792, ttl=64 (reply in 23)
27	11.333449000	194.210.238.155	172.16.30.1	ICMP	98	Echo (ping) reply id=0x12fe, seq=7/1792, ttl=55 (request in 21)
28	12.033976000	Cisco_3a:f6:03	Spanning-tree (for-br	STP	60	Conf. Root = 32768/30/fc:fb:fb:3a:f6:00 Cost = 0 Port = 0x8004
29	14.033718000	Cisco_3a:f6:03	Spanning-tree (for-br	STP	60	Conf. Root = 32768/30/fc:fb:fb:3a:f6:00 Cost = 0 Port = 0x8004
30	14.218117000	Cisco_3a:f6:03	Cisco_3a:f6:03	LOOP	60	Reply

### 6.1.10 Experiência 5 - Ping para *www.google.pt* a partir do tux32

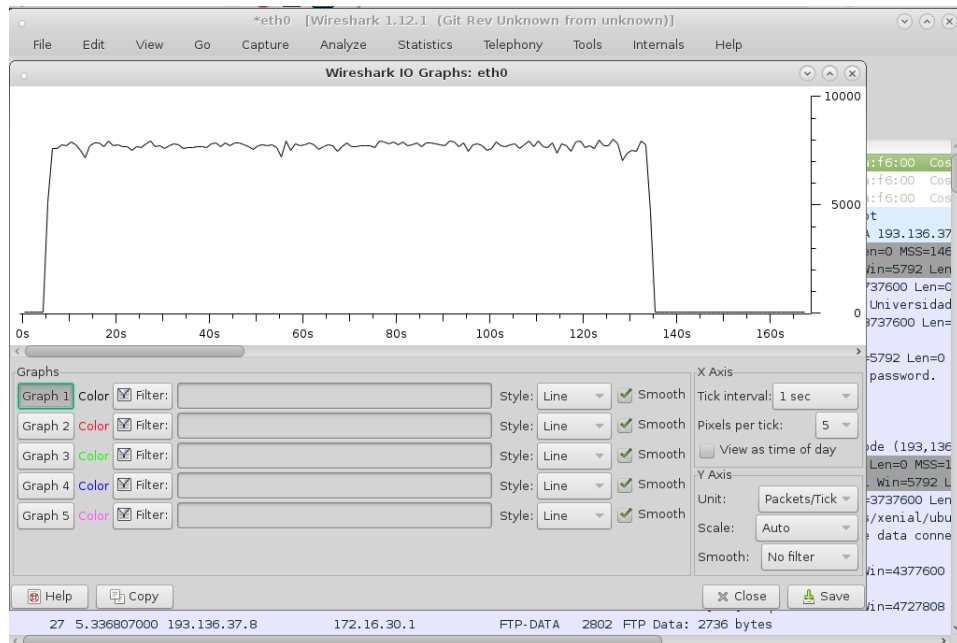
No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	Cisco_3a:f6:04	Spanning-tree (for-br	STP	60	Conf. Root = 32768/31/fc:fb:fb:3a:f6:00 Cost = 0 Port = 0x8004
2	1.646086000	172.16.31.1	172.16.1.1	DNS	73	Standard query 0xbf82 A www.google.pt
3	1.647138000	172.16.1.1	172.16.31.1	DNS	475	Standard query response 0xbf82 A 194.210.238.159 A 194.210.238.163 A 194.210.238.165 A 194.210.238.167
4	1.647269000	172.16.31.1	194.210.238.159	ICMP	98	Echo (ping) request id=0x14fb, seq=1/256, ttl=64 (reply in 5)
5	1.653270000	194.210.238.159	172.16.31.1	ICMP	98	Echo (ping) reply id=0x14fb, seq=1/256, ttl=55 (request in 4)
6	1.653394000	172.16.31.1	172.16.1.1	DNS	88	Standard query 0xdd5f PTR 159.238.210.194.in-addr.arpa
7	1.661177000	172.16.1.1	172.16.31.1	DNS	147	Standard query response 0xdd5f No such name
8	2.004959000	Cisco_3a:f6:04	Spanning-tree (for-br	STP	60	Conf. Root = 32768/31/fc:fb:fb:3a:f6:00 Cost = 0 Port = 0x8004
9	2.649285000	172.16.31.1	194.210.238.159	ICMP	98	Echo (ping) request id=0x14fb, seq=2/512, ttl=64 (no response found!)
10	2.654901000	194.210.238.159	172.16.31.1	ICMP	98	Echo (ping) reply id=0x14fb, seq=2/512, ttl=55 (request in 9)
11	3.650978000	172.16.31.1	194.210.238.159	ICMP	98	Echo (ping) request id=0x14fb, seq=3/768, ttl=64 (no response found!)
12	3.657216000	194.210.238.159	172.16.31.1	ICMP	98	Echo (ping) reply id=0x14fb, seq=3/768, ttl=55 (request in 11)
13	4.009722000	Cisco_3a:f6:04	Spanning-tree (for-br	STP	60	Conf. Root = 32768/31/fc:fb:fb:3a:f6:00 Cost = 0 Port = 0x8004
14	4.652282000	172.16.31.1	194.210.238.159	ICMP	98	Echo (ping) request id=0x14fb, seq=4/1024, ttl=64 (reply in 15)
15	4.657867000	194.210.238.159	172.16.31.1	ICMP	98	Echo (ping) reply id=0x14fb, seq=4/1024, ttl=55 (request in 14)
16	5.653932000	172.16.31.1	194.210.238.159	ICMP	98	Echo (ping) request id=0x14fb, seq=5/1280, ttl=64 (reply in 17)
17	5.659481000	194.210.238.159	172.16.31.1	ICMP	98	Echo (ping) reply id=0x14fb, seq=5/1280, ttl=55 (request in 16)
18	6.014507000	Cisco_3a:f6:04	Spanning-tree (for-br	STP	60	Conf. Root = 32768/31/fc:fb:fb:3a:f6:00 Cost = 0 Port = 0x8004
19	6.655553000	172.16.31.1	194.210.238.159	ICMP	98	Echo (ping) request id=0x14fb, seq=6/1536, ttl=64 (reply in 20)
20	6.661126000	194.210.238.159	172.16.31.1	ICMP	98	Echo (ping) reply id=0x14fb, seq=6/1536, ttl=55 (request in 19)
21	8.020193000	Cisco_3a:f6:04	Spanning-tree (for-br	STP	60	Conf. Root = 32768/31/fc:fb:fb:3a:f6:00 Cost = 0 Port = 0x8004
22	8.489541000	Cisco_3a:f6:04	Cisco_3a:f6:04	LOOP	60	Reply



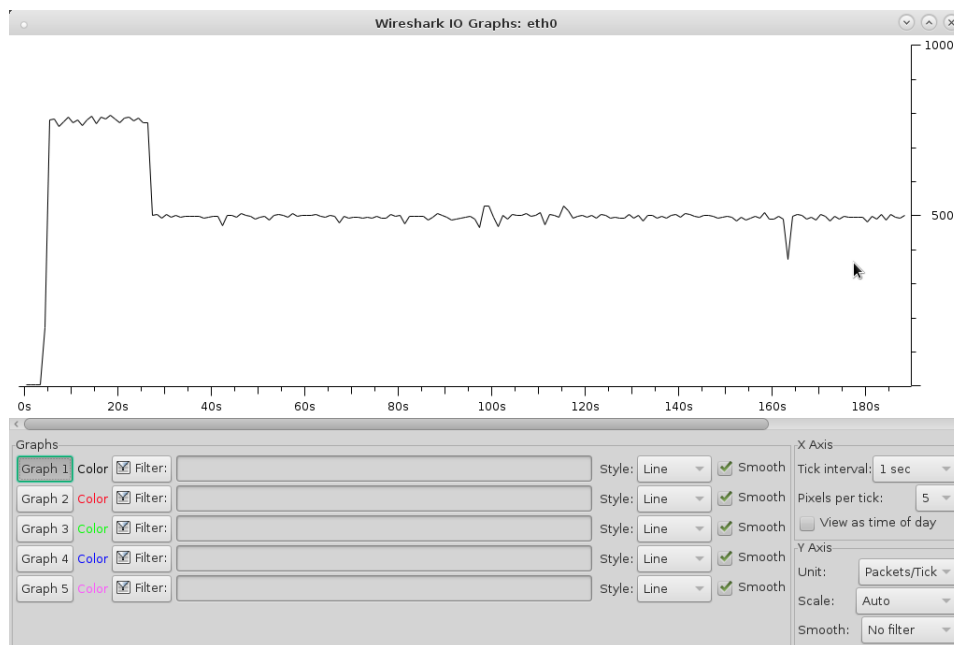
### 6.1.11 Experiência 6 - Download de um ficheiro a partir do tux31

No.	Time	Source	Destination	Protocol	Length	Info
2	0.121419000	Cisco_3a:f6:03	Spanning-tree-(for-br	STP	60	Conf. Root = 32768/30/fc:fb:fb:3a:f6:00 Cos
3	2.126155000	Cisco_3a:f6:03	Spanning-tree-(for-br	STP	60	Conf. Root = 32768/30/fc:fb:fb:3a:f6:00 Cos
4	2.172830000	172.16.30.1	172.16.1.1	DNS	69	Standard query Oxbeffe A ftp.up.pt
5	2.174284000	172.16.1.1	172.16.30.1	DNS	532	Standard query response Oxbeffe A 193.136.37
6	2.174480000	172.16.30.1	193.136.37.8	TCP	74	49039->21 [SYN] Seq=0 Win=29200 Len=0 MSS=146
7	2.176664000	193.136.37.8	172.16.30.1	TCP	70	21->49039 [SYN, ACK] Seq=0 Ack=1 Win=5792 Len=0
8	2.176694000	172.16.30.1	193.136.37.8	TCP	66	49039->21 [ACK] Seq=1 Ack=1 Win=3737600 Len=0
9	2.181413000	193.136.37.8	172.16.30.1	FTP	106	Response: 220 Bem-vindo \303\240 Universidad
10	2.181442000	172.16.30.1	193.136.37.8	TCP	66	49039->21 [ACK] Seq=1 Ack=41 Win=3737600 Len=0
11	2.181514000	172.16.30.1	193.136.37.8	FTP	81	Request: USER anonymous
12	2.183073000	193.136.37.8	172.16.30.1	TCP	66	21->49039 [ACK] Seq=41 Ack=16 Win=5792 Len=0
13	2.183086000	193.136.37.8	172.16.30.1	FTP	100	Response: 331 Please specify the password.
14	2.183150000	172.16.30.1	193.136.37.8	FTP	77	Request: PASS asdas
15	2.186515000	193.136.37.8	172.16.30.1	FTP	89	Response: 230 Login successful.
16	2.186586000	172.16.30.1	193.136.37.8	FTP	71	Request: PASV
17	2.188462000	193.136.37.8	172.16.30.1	FTP	115	Response: 227 Entering Passive Mode (193,136
18	2.188579000	172.16.30.1	193.136.37.8	TCP	74	46227->59973 [SYN] Seq=0 Win=29200 Len=0 MSS=
19	2.190093000	193.136.37.8	172.16.30.1	TCP	70	59973->46227 [SYN, ACK] Seq=0 Ack=1 Win=5792
20	2.190115000	172.16.30.1	193.136.37.8	TCP	66	46227->59973 [ACK] Seq=1 Ack=1 Win=3737600 Le
21	2.190162000	172.16.30.1	193.136.37.8	FTP	129	Request: retr pub/ubuntu-releases/xenial/ubu
22	2.209087000	193.136.37.8	172.16.30.1	FTP	189	Response: 150 Opening BINARY mode data conn
23	2.242549000	193.136.37.8	172.16.30.1	FTP-DATA	1434	FTP Data: 1368 bytes
24	2.242594000	172.16.30.1	193.136.37.8	TCP	66	46227->59973 [ACK] Seq=1 Ack=1369 Win=4027392
25	2.242802000	193.136.37.8	172.16.30.1	FTP-DATA	2802	FTP Data: 2736 bytes
26	2.242832000	172.16.30.1	193.136.37.8	TCP	66	46227->59973 [ACK] Seq=1 Ack=4105 Win=4727808
27	2.244160000	193.136.37.8	172.16.30.1	FTP-DATA	2802	FTP Data: 2736 bytes
28	2.244191000	172.16.30.1	193.136.37.8	TCP	66	46227->59973 [ACK] Seq=1 Ack=6841 Win=5428224

### 6.1.12 Experiência 6 - Gráfico mostrando o número de pacotes por segundo de um download de um ficheiro a partir do tux31



### 6.1.13 Experiência 6 - Gráfico mostrando o número de pacotes por segundo de um download de um ficheiro a partir do tux31, em simultâneo com outro download no tux32



## 6.2 Código Fonte

```
1 #include "clientFTP.h"

3 int main(int argc, char** argv) {
    char cmd[MAX_SIZE] = "";
5     char * response = (char*) malloc(MAX_SIZE);
    int new_port = 0;
7     int parse_ret;

9     ftp_info* info = malloc(sizeof(ftp_info));
    memset(info, 0, sizeof(ftp_info));

11     // verifica se os argumentos
13     check_args(argc, argv);

15     // caso nao seja passado um url
    if (strcmp(argv[2], "default") == 0) {
17         strcpy(info->user, "miguel-teresa-6");
        strcpy(info->password, "rcombuefixe");
19         strcpy(info->host, "ftp.up.pt");
        strcpy(info->path, "file.txt");
21     }
    // caso tenha um url
23     else {
        parse_ret = parser(argv[2], info);
    }
}
```

```

25     if(check_errors(parse_ret)) // deu um erro
26         return parse_ret;

29     printf("user: \">%s%\n", info->user);
30     printf("password: \">%s%\n", info->password);
31     printf("host: \">%s%\n", info->host);
32     printf("url path: \">%s%\n", info->path);
33     printf("file name: \">%s%\n", info->file_name);
34 }

35 // obter o endereço
36 get_address(info);

39 connect_to_server(info, NORMAL, SERVER_PORT);

41 /*send a string to the server*/
42 // Enviar o comando "user BUF1" para o sockfd

43 if (read_response(info, &response) != RECEIVE_CMD_SUCCESS){
44     perror("[Erro Read]\n");
45     exit(-1);
46 }

49 sprintf(cmd, "USER %s\n", info->user);

51 if (write_command(info, cmd, strlen(cmd)) != SEND_CMD_SUCCESS)
52 {
53     perror("[Erro Write]\n");
54     exit(-1);
55 }

57 // Ler o código da string "331 Password required for BUF1" do
58 // sockfd
59 memset(response, 0, MAX_SIZE);
60 if (read_response(info, &response) != RECEIVE_CMD_SUCCESS){
61     perror("[Erro Read]\n");
62     exit(-1);
63 }

64 sprintf(cmd, "PASS %s\n", info->password);

65 if (write_command(info, cmd, strlen(cmd)) != SEND_CMD_SUCCESS)
66 {
67     perror("[Erro Write]\n");
68     exit(-1);
69 }

71 // Ler o código da string "230 User BUF1 logged in"
72 memset(response, 0, MAX_SIZE);
73 if (read_response(info, &response) != RECEIVE_CMD_SUCCESS){
74     perror("[Erro Read]\n");
75     exit(-1);
76 }
77 }

```

```

79 // Estamos logados
81 // Enviar a string "pasv" para o sockfd
82 // Ler os dois ultimos numeros da string "227 Entering Passive
83 // Mode (193,136,28,12,19,91)"
84 // por exemplo percorrer a string ate encontrar um ) e ir
85 // buscar os dois numeros anteriores
86
87 // Com os dois numeros lidos calcular o valor da nova porta.
88 // porta = Num1 * 256 + Num2
89
90 new_port = enter_passive_mode(info, cmd);
91 printf("PORTA: %d\n", new_port);
92
93 // Abrir um sockfd2 com a nova porta que foi calculada
94 connect_to_server(info, DATA, new_port);
95
96 // Enviar a string "retr BUF4" para o sockfd
97 // Ler a string "retr BUF4" no sockfd2
98 check_file(info, cmd);
99
100 // Fazer o download do ficheiro
101 download_file(info);
102
103 close(info->sockfd);
104
105 return 0;
106 }

```

src/main.c

```

1 #include "clientFTP.h"
2
3 void check_args(int argc, char **argv) {
4     if (argc != 3) {
5         printf("Usage: download ftp://[<user>:<password>@]<host>/<
6         url-path>\n");
7         exit(1);
8     }
9 }
10
11 int get_address(ftp_info *info) {
12     if ((h=gethostbyname(info->host)) == NULL) {
13         perror("gethostbyname");
14         exit(1);
15     }
16
17     printf("Host name : %s\n", h->h_name);
18     printf("IP Address : %s\n", inet_ntoa(*((struct in_addr *)h->
19     h_addr)));
20
21     strcpy(info->server_address, inet_ntoa(*((struct in_addr *)h->
22     h_addr)));
23
24     return 0;
25 }

```

```

23 }
24
25 int connect_to_server(ftp_info *info, int type, int port) {
26     struct sockaddr_in server_addr;
27
28     /*server address handling*/
29     bzero((char*)&server_addr, sizeof(server_addr));
30     server_addr.sin_family = AF_INET;
31     server_addr.sin_addr.s_addr = inet_addr(info->server_address);
32     /*32 bit Internet address network byte ordered*/
33     server_addr.sin_port = htons(port); /*server TCP port must
34     be network byte ordered */
35
36     /*open an TCP socket*/
37     if (type == NORMAL) {
38         if ((info->sockfd = socket(AF_INET, SOCK_STREAM, 0)) < 0) {
39             perror("socket()");
40             exit(0);
41         }
42
43         printf("Socket connected port:%d ip:%s\n", info->sockfd,
44             info->server_address);
45
46         /*connect to the server*/
47         if (connect(info->sockfd, (struct sockaddr *)&server_addr,
48             sizeof(server_addr)) < 0) {
49             perror("connect()");
50             exit(0);
51         }
52     }
53     else if (type == DATA) {
54         if ((info->data_sockfd = socket(AF_INET, SOCK_STREAM, 0)) <
55             0) {
56             perror("socket()");
57             exit(0);
58         }
59
60         printf("Socket connected port:%d ip:%s\n", info->data_sockfd,
61             info->server_address);
62
63         /*connect to the server*/
64         if (connect(info->data_sockfd, (struct sockaddr *)&
65             server_addr, sizeof(server_addr)) < 0) {
66             perror("connect()");
67             exit(0);
68         }
69     }
70
71     return 0;
72 }
73
74 /**
75  * @brief Funcao para escrever um comando.
76  * @param info - informacoes sobre o ftp.
77  * @param cmd - comando a enviar.
78  * @param size - tamanho do comando a enviar.

```

```

71  * @return Retorna 0 em caso de sucesso, -1 caso ocorra um erro
    ao enviar
    *      o comando e -2 caso o envio do comando fique
        incompleto.
73 */
int write_command(ftp_info *info, char *cmd, int size) {
75     int bytes;

77     printf("A enviar comando -> %s", cmd);

79     bytes = write(info->sockfd, cmd, size);

81     if (bytes <= 0) {
        printf("[Erro Write] Nao foi possivel enviar o comando: %s",
            cmd);
83         return SEND_CMD_ERROR;
    }

85     if (bytes != size) {
87         printf("[Erro Write] Nao foi possivel enviar todos os dados
        .\n");
        return SEND_CMD_INCOMPLETE;
89     }

91     return SEND_CMD_SUCCESS;

93 }

95 int read_response(ftp_info *info, char **response) {
97     if (read(info->sockfd, *response, MAX_SIZE) <= 0) {
99         printf("[Erro Read] Nao foi possivel receber uma resposta.\n
        ");
        return RECEIVE_CMD_ERROR;
101    }

103    printf("Resposta recebida com sucesso -> %s", *response);

105    return RECEIVE_CMD_SUCCESS;
    }

107 int enter_passive_mode(ftp_info *info, char *cmd) {
109     char *response = (char*) malloc(MAX_SIZE);

111     sprintf(cmd, "PASV\n");

113     if (write_command(info, cmd, strlen(cmd)) != SEND_CMD_SUCCESS)
        {
            perror("[Erro Write] ");
115             exit(-1);
        }

117     memset(response, 0, MAX_SIZE);
119     if (read_response(info, &response) != RECEIVE_CMD_SUCCESS) {
        perror("[Erro Read] ");
    }

```

```

121     exit(-1);
122 }
123
124     return calculate_port(response);
125 }
126
127 int calculate_port(char *response) {
128     char *str2;
129     char *str = malloc(MAX_SIZE);
130     int port;
131
132     str = strtok(response, "("); //retira texto antes do "("
133     str = strtok(NULL, ")"); //retira ")"
134
135     strtok(str, ","); //retira as ","
136     strtok(NULL, ",");
137     strtok(NULL, ",");
138     strtok(NULL, ",");
139     str2 = strtok(NULL, ",");
140     port = strtol(str2, NULL, 10) * 256;
141     str2 = strtok(NULL, ",");
142     port += strtol(str2, NULL, 10);
143
144     return port;
145 }
146
147 int check_file(ftp_info *info, char *cmd) {
148     char *response = (char*)malloc(MAX_SIZE);
149
150     // Enviar a string "retr BUF4" para o sockfd
151     sprintf(cmd, "retr %s\n", info->path);
152     if (write_command(info, cmd, strlen(cmd)) != SEND_CMD_SUCCESS)
153     {
154         perror("[Erro Write]\n");
155         exit(-1);
156     }
157
158     // Ler a string "retr BUF4" no sockfd2
159     memset(response, 0, MAX_SIZE);
160     if (read_response(info, &response) != RECEIVE_CMD_SUCCESS) {
161         perror("[Erro Read]\n");
162         exit(-1);
163     }
164
165     strtok(response, " ");
166
167     if (strcmp(response, "150") != 0) {
168         printf("[Erro File] Ficheiro nao encontrado.\n");
169         exit(-2);
170     }
171
172     return 0;
173 }
174
175 int download_file(ftp_info * info) {
176     FILE *fd_file;

```

```

177 int bytes;
char *buf = (char*) malloc(MAX_SIZE);

179 fd_file = fopen(info->file_name, "w");

181 if (fd_file == NULL) {
182     printf("[Erro File] Nao foi possivel abrir o ficheiro.\n");
183     exit(-1);
184 }

185 printf("Downloading %s ... \n", info->file_name);

187 while ((bytes = read(info->data_sockfd, buf, sizeof(char)*
188     MAX_SIZE)) > 0) {
189     fwrite(buf, bytes, 1, fd_file);
190 }

191 printf("Download finished.\n");

193 close(info->data_sockfd);
194 fclose(fd_file);
195 return 0;
196 }

197
198
199 int check_errors(int parse_ret) {
200     switch (parse_ret) {
201         case REG_ERROR:
202             printf("[Erro Regex] Nao foi possivel compilar a expressao
203             regular.\n");
204             return REG_ERROR;
205             break;
206         case URL_NOTVALID:
207             printf("[Erro Regex] URL inserido nao e valido.\n");
208             return URL_NOTVALID;
209             break;
210         case REGEX_URL_ERROR:
211             printf("[Erro Regex] A expressao regular nao encontrou
212             correspondencia.\n");
213             return REGEX_URL_ERROR;
214             break;
215         case URL_ERROR:
216             printf("[Erro URL] Nao foi possivel fazer parse.\n");
217             return URL_ERROR;
218             break;
219         case URL_USER_ERROR:
220             printf("[Erro User] Nao foi possivel fazer parse.\n");
221             return URL_USER_ERROR;
222             break;
223         case URL_PASS_ERROR:
224             printf("[Erro Password] Nao foi possivel fazer parse.\n");
225             return URL_PASS_ERROR;
226             break;
227         case URL_HOST_ERROR:
228             printf("[Erro Host] Nao foi possivel fazer parse.\n");
229             return URL_HOST_ERROR;
230             break;

```



```

229     default:
230         break;
231 }
232
233 return 0;
234 }

```

src/clientFTP.c

```

#include <stdio.h>
2 #include <sys/types.h>
#include <sys/socket.h>
4 #include <netinet/in.h>
#include <arpa/inet.h>
6 #include <stdlib.h>
#include <unistd.h>
8 #include <signal.h>
#include <netdb.h>
10 #include <string.h>
#include <errno.h>
12 #include "parser.h"

14 /**
 * @brief Funcao que verifica os argumentos passados pelo
 *        utilizador.
16 * @param argc - numero de argumentos.
 * @param argv - argumentos inseridos.
18 */
void check_args(int argc, char **argv);

20
22 /**
 * @brief Funcao obter o endereco ip.
 * @param info - informacoes sobre o ftp.
24 * @return Retorna 0 em caso de sucesso.
 */
26 int get_address();

28 /**
 * @brief Funcao para conectar ao servidor.
30 * @param info - informacoes sobre o ftp.
 * @param type - tipo de conexao.
32 * @param port - porta.
 * @return Retorna 0 em caso de sucesso.
34 */
int connect_to_server(ftp_info *info, int sockfd, int port);

36
38 /**
 * @brief Funcao para escrever um comando.
 * @param info - informacoes sobre o ftp.
40 * @param cmd - comando a enviar.
 * @param size - tamanho do comando a enviar.
42 * @return Retorna 0 em caso de sucesso, -1 caso ocorra um erro
 *        ao enviar
 *        o comando e -2 caso o envio do comando fique
 *        incompleto.
44 */
int write_command(ftp_info *info, char *cmd, int size);

```

```

46  /**
48  * @brief Funcao para ler uma resposta.
49  * @param info - informacoes sobre o ftp.
50  * @param response - resposta recebida.
51  * @return Retorna 0 em caso de sucesso e -1 caso contrario.
52  */
53  int read_response(ftp_info *info, char **response);
54
55  /**
56  * @brief Funcao para entrar no modo passivo.
57  * @param info - informacoes sobre o ftp.
58  * @param cmd - comando a ser enviado.
59  * @return Retorna o valor da nova porta.
60  */
61  int enter_passive_mode(ftp_info *info, char *cmd);
62
63  /**
64  * @brief Funcao para calcular o valor da nova porta.
65  * @param response - resposta recebida que contem os
66  *                  valores para calcular o valor da nova
67  *                  porta.
68  * @return Retorna o valor da nova porta.
69  */
70  int calculate_port(char *response);
71
72  /**
73  * @brief Funcao para verificar se o ficheiro existe.
74  * @param info - informacoes sobre o ftp.
75  * @param cmd - comando a ser enviado.
76  * @return Retorna 0 em caso de sucesso.
77  */
78  int check_file(ftp_info *info, char *cmd);
79
80  /**
81  * @brief Funcao para fazer o download de um ficheiro
82  * @param info - informacoes sobre o ftp.
83  * @return Retorna 0 em caso de sucesso.
84  */
85  int download_file(ftp_info *info);
86
87  /**
88  * @brief Funcao para imprimir uma mensagem de erro de acordo
89  *        com o codigo de retorno passado.
90  * @param parse_ret - codigo de retorno.
91  * @return Retorna o codigo de erro correspondente.
92  */
93  int check_errors(int parse_ret);

```

src/clientFTP.h

```

1  #include "parser.h"
2
3
4  int parser(char *url, ftp_info *info) {
5      int check_return;

```

```

7   check_return = check_regex(url);

9   if (check_return == URL_VALID)
    return url_parser(url, info);

11  else
13      return check_return;
15  }

17  int check_regex(char *url) {
18      regex_t regex;
19      int reti;

21      /* Compila a expressao regular */
22      /* ftp://[<user>:<password>@]<host>/<url-path>
23         <user> -> [[A-Za-z0-9_-]+
24         <password> -> [A-Za-z0-9]+
25         <host> -> [A-Za-z0-9\\\.]+
26         <url-path> -> ([A-Za-z0-9_-\\\.]+\\\/)+
27      */
28      reti = regcomp(&regex, "ftp://\\[[A-Za-z0-9_-]+:[A-Za-z0-9]+@
29          \\[A-Za-z0-9\\\.]+\\\/([A-Za-z0-9_-]+\\\/)+([A-Za-z0-9_-]+\\\.)"
30          , REG_EXTENDED);
31      if (reti)
32          return REG_ERROR;

33      /* Executa a expressao regular */
34      // exemplo de regex a funcionar
35      reti = regexec(&regex, url, 0, NULL, 0);

36      /* Validate regexSimple */
37      if (!reti) {
38          printf("URL valido\n");
39          return URL_VALID;
40      }
41      else if (reti == REG_NOMATCH)
42          return URL_NOTVALID;

43      else
44          return REGEX_URLERROR;

45      /* Liberta a memoria alocada */
46      regfree(&regex);
47  }

49  int url_parser(char* url, ftp_info * info) {
50      const char d1[2] = "[";
51      const char d2[2] = ":";
52      const char d3[2] = "@";
53      const char d4[2] = "/";
54      char *token = (char*)malloc(MAX_SIZE);
55      char *user = (char*)malloc(MAX_SIZE);
56      char *password = (char*)malloc(MAX_SIZE);
57      char *host = (char*)malloc(MAX_SIZE);
58      char *path = (char*)malloc(MAX_SIZE);

```

```

61 //ftp://[<user>:<password>@]<host>/<url-path>
63 token = strtok(url, d1);
64 token = strtok(NULL, d1);
65
66 if (token != NULL) {
67     // USER
68     user = strtok(token, d2);
69     token = strtok(NULL, d2);
70
71     if (token != NULL) {
72         // PASSWORD
73         password = strtok(token, d3);
74         token = strtok(NULL, d3);
75
76         if (token != NULL) {
77             // HOST
78             host = strtok(token, d4);
79             token = strtok(NULL, "");
80
81             if (token != NULL) {
82                 // URL PATH
83                 path = token;
84             }
85             else return URLHOST_ERROR;
86         }
87         else return URLPASS_ERROR;
88     }
89     else return URLUSER_ERROR;
90 }
91 else return URLError;
92
93 strcpy(info->user, user);
94 strcpy(info->password, password);
95 strcpy(info->host, host);
96 strcpy(info->path, path);
97
98 token = strrchr(path, '/');
99
100 if (token && *(token + 1))
101     strcpy(info->file_name, token+1);
102
103 return SUCCESS_PARSER;
104 }

```

src/parser.c

```

#include <stdio.h>
2 #include <stdlib.h>
#include <string.h>
4 #include <regex.h>
#include "Utilities.h"
6
/**
8  * @brief Funcao que faz parse do url inserido pelo utilizador.
9  * @param url - url inserido pelo utilizador.
10 * @param info - informacoes sobre o ftp.

```

```

12  * @return Retorna 0 em caso de sucesso.
13 */
14 int parser(char *url, ftp_info *info);
15
16 /**
17  * @brief Funcao para verificar se o url inserido faz
18  *        match com ftp://[<user>:<password>@]<host>/<url-path
19  *        >.
20  * @param url - url inserido pelo utilizador.
21  * @return Retorna 0 em caso de sucesso.
22 */
23 int check_regex(char *url);
24
25 /**
26  * @brief Funcao para obter as informacoes atraves do url.
27  * @param url - url inserido pelo utilizador.
28  * @param info - informacoes sobre o ftp.
29  * @return Retorna 0 em caso de sucesso.
30 */
31 int url_parser(char *url, ftp_info *info);

```

src/parser.h

```

1  #define MAX_SIZE 1024
2
3  /* REGEX RETURNS */
4  #define URL_VALID 0
5  #define REG_ERROR 1
6  #define URL_NOTVALID 2
7  #define REGEX_URL_ERROR 3
8
9  /* PARSER RETURNS */
10 #define SUCCESS_PARSER 0
11 #define URL_ERROR 4
12 #define URL_USER_ERROR 5
13 #define URL_PASS_ERROR 6
14 #define URL_HOST_ERROR 7
15
16 /* CLIENT FTP RETURNS */
17 #define SEND_CMD_SUCCESS 0
18 #define SEND_CMD_ERROR -1
19 #define SEND_CMD_INCOMPLETE -2
20 #define RECEIVE_CMD_SUCCESS 0
21 #define RECEIVE_CMD_ERROR -1
22
23
24 /* CLIENT FTP */
25 #define IP_SIZE 15
26 #define SERVER_PORT 21
27 #define DATA 1
28 #define NORMAL 2
29
30 struct hostent *h;
31
32 typedef struct {
33     char server_address[IP_SIZE];
34     int sockfd;

```

```

35 int data_sockfd;
   char file_name[MAX_SIZE];
37 char user[MAX_SIZE];
   char password[MAX_SIZE];
39 char host[MAX_SIZE];
   char path[MAX_SIZE];
41 } ftp_info;

```

src/Utilities.h

```

1 all:
   gcc -Wall -o clientFTP clientFTP.c parser.c main.c
3 clean:
   rm clientFTP

```

src/Makefile

## 6.3 Scripts

### 6.3.1 Configuração do tux31

```

#!/bin/bash
2 ifconfig eth0 down
  ifconfig eth0 up 172.16.30.1/24
4 route add default gw 172.16.30.254
  route add -net 172.16.31.0/24 gw 172.16.30.254
6 echo 0 > /proc/sys/net/ipv4/icmp_echo_ignore_broadcasts

```

scripts/tux1.sh

### 6.3.2 Configuração do tux32

```

#!/bin/bash
2 ifconfig eth0 down
  ifconfig eth0 up 172.16.31.1/24
4 route add default gw 172.16.31.253
  route add -net 172.16.30.0/24 gw 172.16.31.253
6 echo 0 > /proc/sys/net/ipv4/icmp_echo_ignore_broadcasts

```

scripts/tux2.sh

### 6.3.3 Configuração do tux34

```

#!/bin/bash
2 ifconfig eth0 down
  ifconfig eth1 down
4 ifconfig eth0 up 172.16.30.254/24
  ifconfig eth1 up 172.16.31.253/24
6 route add default gw 172.16.31.254
  echo 1 > /proc/sys/net/ipv4/ip_forward
8 echo 0 > /proc/sys/net/ipv4/icmp_echo_ignore_broadcasts

```

scripts/tux4.sh

## 6.4 Configurações do router e switch

```
// Configure Switch
2
en
4 password: 8nortel

6 // Create 2 vlans (ids = 30, 31)
configure terminal
8 vlan 30
end
10 configure terminal
vlan 31
12 end
show vlan 30
14 show vlan 31

16
// Add ports to vlans
18
// tux31 E0 on port 1
20 configure terminal
interface fastEthernet 0/1
22 switchport mode access
switchport access vlan 30
24 end

26 // tux32 E0 on port 2
configure terminal
28 interface fastEthernet 0/2
switchport mode access
30 switchport access vlan 31
end

32
// tux34 E0 on port 3
34 configure terminal
interface fastEthernet 0/3
36 switchport mode access
switchport access vlan 30
38 end

40 // tux34 E1 on port 4
configure terminal
42 interface fastEthernet 0/4
switchport mode access
44 switchport access vlan 31
end

46
// Configure Router
48 username: root
password: 8nortel
50
configure terminal
52 interface gigabitethernet 0/0
ip address 172.16.31.254 255.255.255.0
54 no shutdown
```

```

56 ip nat inside
56 exit

58 interface gigabitethernet 0/1
ip address 172.16.1.39 255.255.255.0
60 no shutdown
ip nat outside
62 exit

64 configure terminal
interface gigabitethernet 0/0
66 ip address 172.16.31.254 255.255.255.0
no shutdown
68 ip nat inside source list 1 pool ovrld overload
exit
70

72 configure terminal
access-list 1 permit 172.16.30.0 0.0.0.255
74 access-list 1 permit 172.16.31.0 0.0.0.255
ip route 0.0.0.0 0.0.0.0 172.16.1.254
76 ip route 172.16.30.0 255.255.255.0 172.16.31.253
end
78

// Remove iptables no terminal do tux4
80 iptables -F
iptables -t nat -F

```

scripts/config.txt