**Trabalho 2**

Configuração de uma Rede e Desenvolvimento de uma Aplicação de Download

**Relatório Final**



Mestrado Integrado em Engenharia Informática e Computação

**Redes de Computadores**

**Grupo 4 :**

Ana Rita Torres, [up201406093@fe.up.pt](mailto:up201406093@fe.up.pt)

Catarina Correia,  [up201405765@fe.up.pt](mailto:up201405765@fe.up.pt)

Ricardo Neves, [up201405868@fe.up.pt](mailto:up201405868@fe.up.pt)

Faculdade de Engenharia da Universidade do Porto

R. Dr. Roberto Frias, 4200-464 Porto

20 de dezembro de 2016

# Sumário

Este relatório tem como objetivo explicar o segundo projeto da Unidade Curricular de Redes de Computadores. Este projeto dividiu-se em duas partes: numa primeira parte, foi desenvolvida uma aplicação de *download;* numa segunda parte, foram realizadas seis experiências especificadas no enunciado com o intuito de configurar uma rede.

Nas secções que se seguem, será explorado o desenvolvimento da referida aplicação e será feita uma análise das experiências mencionadas.

# Índice

# Introdução

O segundo projeto de Redes de Computadores, como já foi referido, divide-se um duas partes principais. Na primeira parte, foi pedida a elaboração de uma aplicação de *download* que procedesse à transferência de um ficheiro, implementando o protocolo *FTP*. Na segunda parte, foi pedida a configuração de uma rede. Este configuração foi dividida em várias experiências e respeita a seguinte ordem:

1. Configuração de um *IP* de rede;
2. Implementação de duas redes *LAN’s* virtuais num *switch*;
3. Configuração de um *router* em *LINUX*;
4. Configuração de um *router* comercial com e implementação de *NAT*;
5. *DNS;*
6. Conexões *TCP;*

# Parte 1 – Aplicação de Download

Para a realização desta primeira parte, o grupo baseou-se em vários documentos destacando-se os seguintes: o ficheiro *RFC959* que descreve o protocolo de transferência de dados (*FTP*) e o ficheiro *RFC1738* que informa acerca do uso de *URL* e o seu devido tratamento.

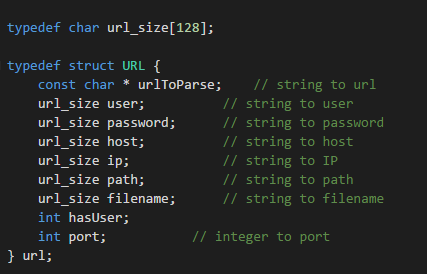
## Arquitetura

Para uma melhor organização e estrututa, a aplicação divide-se em duas camadas: a de processamento do *URL* e a do cliente *FTP*.

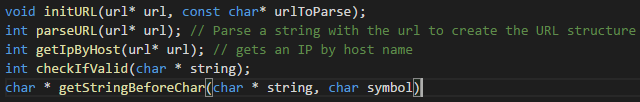
A aplicação desenvolvida aceita um *link* como argumento, especificado na linha de comandos e permite que o *download* seja feito de forma anónima ou não. Caso não se pretenda o anonimato, basta fornecer no *URL* o utilizador, seguido por ‘:’, a palavra-passe e por fim ‘@’. Se tais dados não forem fornecidos, assume-se o utilizador como anónimo (“*anonymous”*) e a palavra-passe como sendo nula.

/\* imagens de input com e sem utilizador \*/

Relativamente ao processamento do *URL,* foi criada uma *struct* que guarda as várias informações representadas no *link*: *urlToParse, user, password, host, ip, path, filename, hasUser* e *port*. O atributo *port* é sempre 21, pois é o número de controlo do protocolo *FTP.*



Na execução do programa, são chamadas várias funções:



A função ***initURL*** guarda o *URL* recebido e aloca memória para os vários atributos. Em seguida, a ***parseURL*** processa a variável *urlToParse* (*link* recebido na linha de comandos) *e* guarda toda a informação necessária. Por último, a função ***getIPByHost***, que chama a função *gethostbyname* com o *host* da *struct,* converte o *hostname* para um endereço *IP.*

As funções ***checkIfValid*** e ***getStringBeforeChar*** são funções auxiliares da *parseURL* e verificam se o *URL* recebido é constituído por caracteres válidos e obtêm uma *string* antes de um determinado caracter, respetivamente.

O cliente FTP é representado por uma estrutura que contém dois descritores, um relativo ao *socket* de controlo e outro ao de dados.

/inserir imagens da struct ftp/

Depois de interpretar a informação introduzida pelo utilizador, isto é, após o processamento do *URL*, procede-se à ligação do cliente FTP ao servidor FTP através de um *socket* TCP através da função ***ftpConnect***.

/super inacabado/

## Resultados de Download

A aplicação desenvolvida foi testada não só em modo normal, ou seja, com um utilizador e palavra-passe, mas também em modo anónimo. Para efeito de teste, foram realizados diversos *dowloads*, todos bem sucedidos, tendo o maior ficheiro testado 500MB.

Caso ocorra algum erro, a aplicação termina e o erro é impresso na consola. Caso contrário, é impresso na consola uma mensagem a dizer que a transferência foi bem sucedida e qual o tamanho do ficheiro.

# Parte 2 – Configuração da Rede e análise

## Configuração de um IP de Rede

# Conceitos

Esta experiência teve como objetivo não só a compreensão da configuração de IP’s em máquinas diferentes, numa mesma rede, mas também a identificação e distinção dos diferentes pacotes enviados entre si.

Existem vários tipos de pacotes de dados, como por exemplo os ARP. Este protocolo é responsável por mapear um endereço de rede para um endereço físico (MAC). Quando um pacote chega a um *gateway*, este pede ao ARP para encontrar um *host* físico ou um endereço MAC que corresponda ao endereço IP. Por sua vez, este procura na sua cache e, se nenhuma entrada for encontrada, o ARP transmite um pacote de solicitação, que contém o IP para o qual se pretende saber o MAC, para todas as máquinas da mesma LAN. Se, em alguma máquina, ocorrer uma correspondência, então esta envia um pacote que contém o seu endereço MAC à máquina que solicitou a informação e o ARP guarda na sua tabela os dados para uma próxima ocorrência.

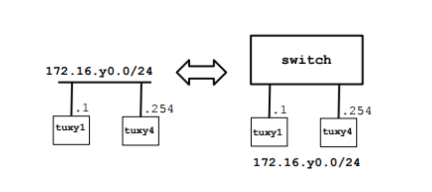
Cada pacote de dados contém, no cabeçalho das tramas enviadas, informação relativa ou ao tipo de protocolo ou ao tamanho da trama.

No primeiro caso, o pacote de dados contém um identificador constituído por 2 bytes, *EtherType*, que permite identificar o protocolo associado. Por exemplo, no caso do ARP, o identificador é 0x0806 e, no caso do IP (IPv4), o identificador corresponde ao valor 0x0800. Nesta circunstância, visto que o tamanho não é especificado, este é obtido através de um delimitador de início da trama e o *interpacket gap*.

Quando uma interface de rede é desconectada, não é possível estabelecer qualquer tipo de comunição com essa interface e, por conseguinte, o próprio computador não consegue comunicar consigo mesmo. De forma a que tal seja evitado, a interface de *loopback* garante a conexão de quaisquer aplicações no computador com servidores do mesmo.

O *loopback* é, portanto, uma interface de rede virtual que o computador usa para comunicar consigo próprio. A sua utilização reflete-se principalmente na realização de diagnósticos, na solução de problemas e ainda na conexão a servidores em execução na máquina local.

# Experiência



Para a realização desta experiência, primeiro foram configurados os IP’s das portas eth0 de dois computadores, o tux41 e o tux44, utilizando o comando ***ifconfig*** e o comando ***route***, necessário para adicionar rotas à tabela de reencaminhamento.

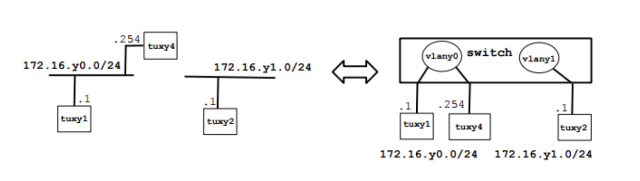
Em seguida, enviou-se o comando ***ping*** para verificar a conetividade entre as duas máquinas configuradas. Este comando gera e envia para o endereço escolhido pacotes ICMP (um protocolo que relata erros caso um determinado serviço ou host não possa ser alcançado para o envio de pacotes) e aguarda por uma resposta.

Após a verificação da ligação entre as duas máquinas, foram apagadas todas as entradas na tabela ARP recorrendo ao comando **arp -d < ip address >**. Por fim, repetiu-se o comando ping, registando os resultados através do *Wireshark*.

Analisando os resultados guardados pelo *Wireshark*, observa-se que é perguntado à rede qual o endereço MAC com um endereço de IP igual a endereço. Por sua vez, o computador responde com o endereço MAC respetivo e, a partir daí, verifica-se que, para cada pedido ICMP, segue-se uma resposta.

/\* Imagem com o log para uma melhor análise \*/

## Implementação de duas Redes *LAN’s* Virtuais num *Switch*



Inicialmente, configurou-se o tux2, tendo em conta as configurações já feitas na primeira experiência.

Em seguida, foram criadas (e configuradas) duas *LAN’s* virtuais diferentes:

* *VLAN* 40 – 172.16.40.0/24 – à qual pertecem os computadores tux41 e o tux44;
* *VLAN* 41 – 172.16.41.0/24 – à qual pertence o computador tux42.

A criação e configuração da *vlan* requer a inserção dos seguintes comandos na consola do *switch:*

1. configure terminal
2. vlan x (x corresponde ao número da *vlan*; no nosso caso, 40)
3. interface fastethernet 0/y (y corresponde ao número da porta no switch que se pretende adicionar à *vlan*)
4. switchport mode access
5. switchport access vlan x
6. end

Enquanto que o primeiro passo é sempre necessário, pois serve para abrir o modo de configuração, o segundo passo cria uma *vlan*. Para adicionar portas à *vlan* é necessário usar os passos 3 a 5, com a porta que se deseja adicionar. Por fim, para sair do modo de configuração, utiliza-se o comando **end**.

Prosseguindo com a experiência, fez-se **ping** do tux1 para o tux4. Tendo em conta que ambos pertencem à mesma rede, verificou-se que não foi enviado qualquer pacote ARP para saber o endereço MAC. Contudo, ao fazer ping do tux1 para o tux2, verificou-se, no *Wireshark*, a inexistência de uma resposta, pois não existe uma forma de comunicação entre as duas redes.

Assim, como só *IPs* dentro da mesma *vlan*  respondem aos *broadcasts,* pode concluir-se que cada *vlan* tem um *brodcast* *domain* diferente.

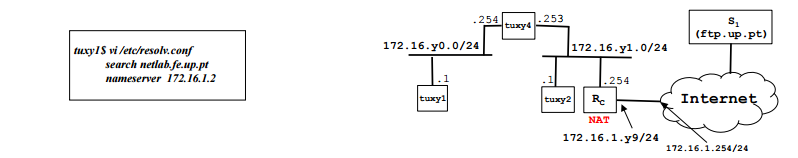
/\* mostras logs \*/

## Configuração de um *router* em LINUX

## Configuração de um *router* comercial e implementação de *NAT*

## *DNS*

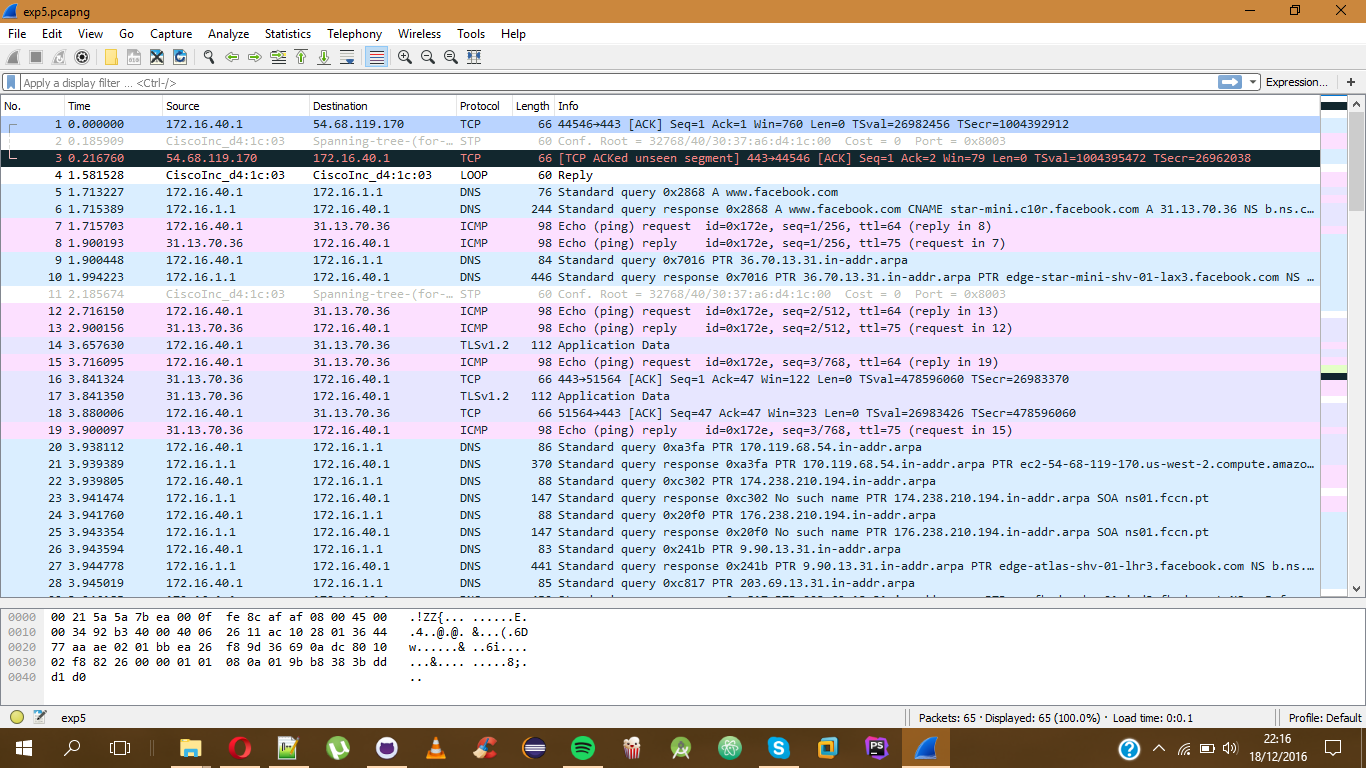
A experiência 5 tem por base a configuração do DNS que, consequentemente, permite a ligação a redes externas, fazendo com que seja possível o acesso à *Internet* através de uma rede interna criada.



A configuração do DNS passa por adicionar estas duas linhas:

* *search netlab.fe.up.pt*
* *nameserver 172.16.1.1*

no ficheiro *resolv.conf* que se encontra no diretório */etc*.

Para testar o funcionamento desta experiência acedeu-se através do *browser* a um *website*, neste caso o *Facebook*.

## Conexões *TCP*