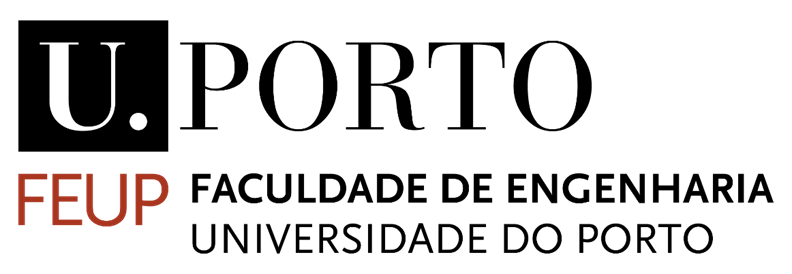
**Configuração de uma Rede**

**e Desenvolvimento de uma Aplicação de Download**

**Relatório do 2º Trabalho Laboratorial**

****

Mestrado integrado em Engenharia Informática e Computação

**Redes de Computadores**

**3MIEIC04**

**Bruno Marques** up201405781

**João Loureiro** up201405652

**José Aleixo Cruz** up201403526

**José Costa** up201402717

Faculdade de Engenharia da Universidade do Porto

Rua Roberto Frias, sn, 4200-465 Porto, Portugal

23 de dezembro de 2016

# Sumário

Este relatório foi elaborado no seguimento do segundo projeto da unidade curricular Redes de Computadores. Aqui é documentada a criação de um programa em linguagem C, que implementa o protocolo FTP (*File Transfer Protocol*) para transferir ficheiros através da Internet. É também descrita a configuração de duas redes virtuais, capazes de transferir dados entre si e com redes exteriores.

Conteúdo

[Sumário 2](#_Toc470203026)

[1. Introdução 4](#_Toc470203027)

[2. Parte 1 – Aplicação de Download 4](#_Toc470203028)

[2.1. url.c 4](#_Toc470203029)

[2.2. ftp.c 5](#_Toc470203030)

[2.3. main.c 6](#_Toc470203031)

[2.4. Exemplo de um download bem-sucedido 6](#_Toc470203032)

[3. Parte 2 – Configuração de Análise da Rede 7](#_Toc470203033)

[3.1. Experiência 1: configurar uma rede IP 7](#_Toc470203034)

[3.2. Experiência 2: implementar duas redes virtuais num switch 8](#_Toc470203035)

[3.3. Experiência 3: configurar um router em Linux 8](#_Toc470203036)

[3.3.1. Resultados da experiência 9](#_Toc470203037)

[3.3.2. Endereços MAC e endereços IP dos pacotes ICMP 10](#_Toc470203038)

[3.4. Experiência 4: configurar um router comercial e implementar NAT 10](#_Toc470203039)

[3.5. Experiência 5: DNS 12](#_Toc470203040)

[3.6. Experiência 6: conexões TCP 13](#_Toc470203041)

[3.7. Experiência 7: implementar NAT em Linux 13](#_Toc470203042)

[4. Conclusões 14](#_Toc470203043)

# Introdução

O segundo projeto laboratorial de Redes de Computadores esmiúça protocolos utilizados na comunicação entre computadores e divide-se em duas partes com os seguintes objetivos:

* **Parte 1 – Aplicação de download**: recorrendo o *File Transfer Protocol* (FTP), criar uma aplicação em linguagem C capaz de transferir qualquer ficheiro de um servidor para um cliente.
* **Parte 2 – Configuração e análise de rede**: utilizando *switches* e *routers*, configurar duas redes virtuais de computadores capazes de comunicar entre si localmente (*Local Area Network –* LAN), com redes exteriores e mesmo com a Internet.

A segunda parte deste projeto segmenta-se num conjunto de seis experiências. Cada experiência possui um guião, indicando como os computadores e as redes devem ser ajustadas e monitorizadas, com vista em responder a perguntas relativas aos protocolos que intervêm na transmissão de dados.

# Parte 1 – Aplicação de Download

Esta parte do trabalho destina-se à realização de um programa capaz de fazer *download* de ficheiros utilizando FTP(*File Transfer Protocol*) . Para tal, basta o utilizador da aplicação inserir um *link* compatível com FTP de um ficheiro, com ou sem informações de *login*, e de forma automática vai ser realizada a transferência do ficheiro pretendido.

Esta aplicação divide-se em três módulos: ***main.c***, [***ftp.c***](file:///C:\Users\Admin\Downloads\ftp.c) e ***url.c***. O módulo *[ftp.c](C:\\Users\\Admin\\Downloads\\ftp.c)* trata das funções necessárias para a interação servidor/cliente invocando comandos FTP para estabelecer a conexão e permitir a transmissão de ficheiros. O módulo *url.c* carrega numa estrutura de dados as informações necessárias para a ligação FTP, obtidas a partir do URL que se usou como argumento. O main.c é responsável por gerir ambos os módulos, de forma a que seja possível interligá-los e executar o programa, mantendo organização no código.

## *url.c*

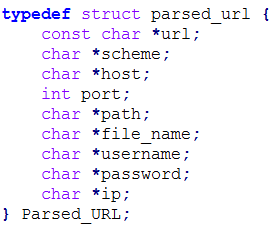
O argumento utilizado na nossa aplicação para fazer download de um ficheiro é o seu URL (*Uniform Resource Locator*) na Internet, com *scheme* FTP.

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

ftp://<user>@<host>/<url-path>

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

O módulo url.c é responsável por pegar no argumento fornecido pelo utilizador e decifrá-lo de forma a obter as informações necessárias para uma ligação FTP, guardando-as numa *struct* com o nome “*Parsed\_URL”*, que é composta da seguinte forma:



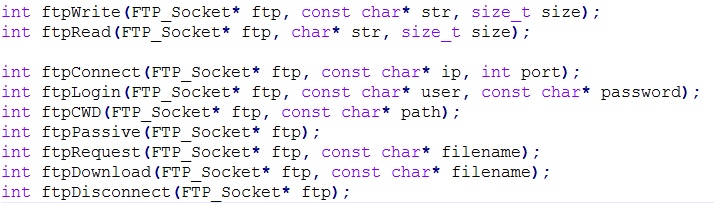
A abordagem tomada consiste em ir percorrendo a *string* que representa o URL e, através dos separadores descritos na sintaxe, obter os campos que estão explícitos no URL, como o *host* e a *path* do ficheiro.Há certas informações que são opcionais no URL, pelo que se não constarem nele são pedidas mais tarde no main.c, como o utilizador e a palavra-passe.

Além dos campos explícitos no URL ou fornecidos pelo utilizador, o url.c é responsável também por obter o endereço IP, a partir do *host* presente no *link* fornecido. Este host é processado pela função hostToIP, que através da execução de duas funções, gethostbyname e inet\_ntoa consegue fazer a dita conversão.

A porta pré-definida para a conexão do *socket* de controlo é a porta 21, que fica definida na struct, se não for especificada nenhuma outra.

## *ftp.c*

Neste ficheiro estão presentes todas as funções utilizadas, relativamente à interação de cliente e servidor, ou seja, envio e recepção de pedidos, bem como à respetiva ligação e desconexão.



Estas são as funções presentes no ficheiro. O ***ftpConnect*** é responsável por fazer a ligação ao servidor FTP, inicializando o descritor de ficheiro para a socket de controlo do *FTP\_Socket\* ftp*, permitindo a troca de informação, em modo ativo, entre servidor e cliente.

O ***ftpWrite*** e o ***ftpRead*** são responsáveis por enviar comandos, e receber mensagens de validação do servidor, em modo ativo. Estas vão ser utilizadas por todas as outras funções deste ficheiro, com excepção do *ftpDownload*, e são essenciais para a aplicação funcionar. De notar que, a cada mensagem enviada, é sempre recebida uma mensagem de resposta. É importante ver sempre qual o *output* do servidor para poder analisar se o pedido foi realizado com sucesso.

O ***ftpPassive*** envia um comando para o servidor informando este de que deseja entrar em modo passivo. O servidor vai então abrir uma nova porta aleatória e enviar como resposta seis números. Com esses seis números, consegue-se determinar qual o endereço IP e a nova porta a que o cliente tem de se conectar. Depois desta resposta, vai ser efetuada uma nova ligação, inicializando o descritor de ficheiro para a socket de *data*. Vai ser através desta que se vai efetuar a transferência do ficheiro pretendido. É necessário entrar em modo passivo para poder fazer *download* de ficheiros, caso contrário estes seriam filtrados pela *firewall*.

Após a realização do *ftpPassive* vais ser possível executar o ***ftpRequest*** e o ***ftpDownload***, sendo que o primeiro apenas envia o comando de *retrieve* do ficheiro especificado, e depois a segunda função vai ser responsável por ler conteúdo do descritor de *data*, criando assim no computador o ficheiro desejado.

O ***ftpLogin*** é responsável por efetuar login no servidor, o ***ftpCWD*** é responsável para mudar o atual diretório de trabalho, para um pedido em *path*, e o ***ftpDisconnect*** serve para enviar a mensagem de desconexão do servidor. A execução desta última declara que a aplicação foi concluída.

## *main.c*

Este ficheiro é responsável apenas por executar, de forma ordenada, as funções dos dois ficheiros anteriores, de forma a que possa interligá-los, e manter a ordem correta de execução de funções (primeirro prepara o IP para qual se vai ligar, de seguida liga-se ao IP, etc...).

A única interação que este módulo faz com o utilizador, é caso não tenha sido introduzido informações acerca do login (ou seja, se pretende utilizar o modo anónimo), o main.c vai pedir que o utilizador introduza a password do modo anónimo.

## Exemplo de um *download* bem-sucedido

Correr a aplicação com o endereço <ftp://ftp.up.pt/pub/robots.txt> resultou no seguinte *output* na consola, bem como a transferência bem-sucedida do ficheiro.

**Nota**: Em modo anónimo é pedido que o utilizador introduza uma password. No caso do FTP da UP, qualquer uma servirá.

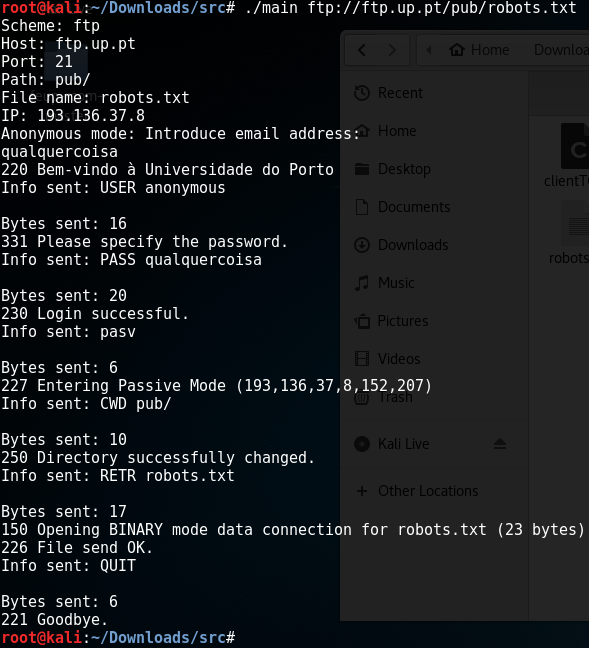


Figura 1 - Output na consola da execução da aplicação de download FTP

# Parte 2 – Configuração de Análise da Rede

## Experiência 1: configurar uma rede IP

Na primeira experiência realizada procedemos à configuração de uma rede de comunicação utilizando *Internet Protocol* para estabelecer a ligação entre um ou mais computadores. Iniciou-se a configuração do tuxy1 e tuxy4, utilizando os comandos:

ifconfig eth0 up

ifconfig eth0 172.16.y0.1/24 (**tuxy1**)

ifconfig eth0 172.16.y0.254/24 (**tuxy4**)

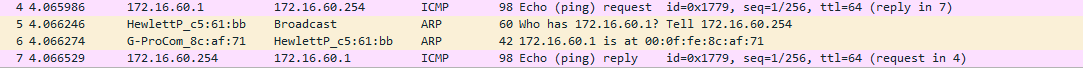
route add default gw 172.16.y0.254

Este último adicionou a *default gateway* no tuxy1, e de seguida utilizamos o comando *ping* de forma a verificar a ligação entre os computadores:

ping 172.16.y0.254

Procedemos a eliminar os pacotes ARP com

arp -d 172.16.y0.254

e a realizar novamente o ping do tuxy1 para tuxy4 utilizando o WireShark para capturar os *requests* e *replies* como se observa na imagem abaixo:

Aqui podemos perceber a importância do ARP (*Address Resolution Protocol*) que permitiu encontrar um endereço na camada de ligação de dados a partir da camada de rede, ou seja, através do IP foi nos permitido obter o endereço MAC. Nos pacotes de ARP é contido um endereço que varia de acordo com o protocolo usado. Após a sua resposta são gerados pacotes de ICMP através do comando *ping*. Nos pacotes de ARP o endereço IP é o endereço do computador na rede local enquanto que o MAC é o endereço físico do computador.

Nos pacotes gerados pelo *ping,* o endereço IP destino é o do computador alvo e o MAC é o endereço do *router*. É enviado também um pacote de controlo com o tamanho do pacote recebido. Quanto ao protocolo usado, o ARP tem informação relativa a esse mesmo nos bytes 2 e 3.

A interface de *loopback* é um canal de comunicação que retransmite toda a informação sem a modificar para a fonte de envio, assim este canal pode ser útil para realizar testes quanto ao estado do dispositivo de comunicação a ser utilizado.

## Experiência 2: implementar duas redes virtuais num *switch*

Nesta experiência foram criadas duas redes virtuais, a **vlany0** e **vlany1**.

Adicionamos ambos os tuxy1 e tuxy4 à vlany0 no *switch*, através dos comandos:

conf t

vlan y0

end

conf t

vlan y1

end

Configurando também o tuxy2, IP e MAC adress apresentado de seguida:

C:\Users\Bruno\AppData\Local\Microsoft\Windows\INetCacheContent.Word\iptuxy2.pngC:\Users\Bruno\AppData\Local\Microsoft\Windows\INetCacheContent.Word\mactxy2.png

Procedeu-se então à configuração das portas da vlan60 – tuxy1 e tuxy4 e 61- tuxy2:

conf t

interface fastethernet0/<PORT>

switchport mode access

switchport access vlan 60 <ou 61>

end

São então efetuados os pings nos diferentes tuxs podendo-se observar que o tuxy1 e o tuxy4 conseguem comunicar entre si enquanto que o tuxy2 não consegue visto que se encontra numa VLAN diferente.

É possível observar isto consultando os logs do WireShark em anexo.

## Experiência 3: configurar um *router* em Linux

Nesta experiência, o computador tuxy4 é programado para servir de *router*. Um *router* é um dispositivo capaz de transmitir dados entre **redes** de computadores, ao invés do *switch* que apenas maneja *packets* entre computadores pertencentes à mesma LAN.

A porta *Eth0* do tuxy1 encontra-se ligada à porta *Eth0* do tuxy4, através da rede virtual *Vlan0* criada no *switch*, pelo que a porta *Eth0* do tuxy4 é o seu *gateway* para a rede Vlan0. Sobra configurar o *gateway* do tuxy4 para a rede *Vlan1*, da qual o tuxy2 faz parte, e permitir que o tuxy4 reencaminhe *packets* IP.

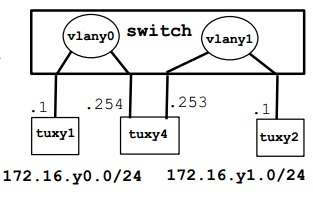


Figura 2 - Esquema da experiência 3

Assim, as *routes* que devem existir nos computadores são as seguintes:

**Tuxy1**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Source | Destination | Gateway | Genmask | Interface |
| 172.16.y0.1 | 0.0.0.0 | 172.16.y0.254 | 0.0.0.0 | *Eth0* |

**Tuxy2**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Source | Destination | Gateway | Genmask | Interface |
| 172.16.y1.1 | 172.16.y0.0 | 172.16.y1.253 | 255.255.255.0 | *Eth0* |

**Tuxy4**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Source | Destination | Gateway | Genmask | Interface |
| 172.16.y0.254 | 172.16.y0.0 | 0.0.0.0 | 255.255.255.0 | *Eth0* |
| 172.16.y1.253 | 172.16.y1.0 | 0.0.0.0 | 255.255.255.0 | *Eth1* |

A tabela de *forwarding* possui os dados das *routes*, permitindo escolher o melhor caminho para um pacote de informação, segundo o endereço IP do pacote.

Para tornar possível que o tuxy4 redirija pacotes IP e permita mensagens de *broadcast*, é necessário executar os seguinte comandos:

# echo 1 > /proc/sys/net/ipv4/ip\_forward

# echo 0 > /proc/sys/net/ipv4/icmp\_echo\_ignore\_broadcasts

### Resultados da experiência

Ao iniciar a comunicação entre o tuxy1 e o tuxy2, nenhum dos computadores consegue associar o outro ao endereço IP, pois não existem entradas na tabela ARP. Para resolver o endereço IP num endereço MAC, é enviado um pacote ARP em *broadcast* perguntando qual é a máquina que tem o endereço IP definido pelas rotas. Essa máquina responde de volta com o seu endereço MAC e quem fez o pedido atualiza a sua tabela ARP.

Por exemplo, quando o tuxy1 faz *ping* ao tuxy2 através do tuxy4, ele desconhece o endereço de *layer* 2 da porya *Eth0* do tuxy4. Assim, faz um *broadcast* para a sua LAN a perguntar quem tem o IP 172.16.y0.254, que é o seu *default gateway*. O tuxy4 responde, enviando o seu endereço MAC para o IP do tuxy1. O tuxy1 adiciona este endereço às suas entradas ARP e passa a poder comunicar com o tuxy4 ao nível local.



Figura 3 - Pedido ARP feito pelo tuxy1 capturado na porta Eth0 do tuxy4

Além de pacotes ARP, são também enviados pacotes ICMP derivados do comando “ping”. Quando se executa esta ordem, o computador envia um *echo request* para tentar obter resposta de um dispositivo com um determinado endereço IP. Caso essa mensagem chegue ao destino, é enviada uma *reply* para notificar a estabilidade da conexão. Caso o *host* ou *router* não tenha sido alcançado, reporta erro.

Neste caso, o *request* é enviado pelo tuxy1, que obtém a *reply* do tuxy2.

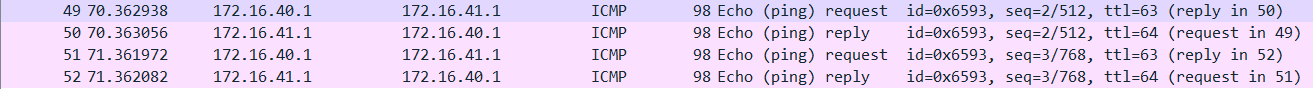


Figura 4 - Pacotes ICMP capturados na porta Eth1 do tuxy4 quando o tuxy1 "pinga" o tuxy2

### Endereços MAC e endereços IP dos pacotes ICMP

Para os pacotes ICMP enviados como sendo *echo request*:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Vlan0 | | | | Vlan1 | | | |
| *Source* | | ***Destination*** | | ***Source*** | | ***Destination*** | |
| IP | MAC | IP | MAC | IP | MAC | IP | MAC |
| Tuxy1 | Tuxy1 | Tuxy2 | Tuxy4 (*Eth0*) | Tuxy1 | Tuxy4 (*Eth1*) | Tuxy2 | Tuxy2 |

Para os pacotes ICMP enviados como sendo *echo reply*:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Vlan0 | | | | Vlan1 | | | |
| *Source* | | ***Destination*** | | ***Source*** | | ***Destination*** | |
| IP | MAC | IP | MAC | IP | MAC | IP | MAC |
| Tuxy2 | Tuxy4  (*Eth0*) | Tuxy1 | Tuxy1 | Tuxy2 | Tuxy2 | Tuxy1 | Tuxy4  (*Eth1*) |

O que isto nos permite concluir é que na camada de ligação de dados, os endereços IP de destino e de origem mantém-se inalterados, mas os endereços MAC são adaptados aos dispositivos dentro da LAN que transmitem a informação.

Por exemplo, neste caso, a resposta do tuxy2 chega ao tuxy1 via tuxy4. O pacote entra na Vlan0 através da porta *Eth0* do tuxy4, pelo que este é o MAC de origem. Dentro da Vlan0, o destino é o endereço MAC do tuxy1, que também possui o IP destino.

## Experiência 4: configurar um *router* comercial e implementar NAT

Depois de estar estabelecida a ligação entre as redes Vlan0 e Vlan1, o passo seguinte do projeto é conectar a segunda rede a uma rede exterior através de um *router* comercial, de forma a permitir a ligação à Internet.

Para tal, é necessário configurar o router para que se ligue à Vlan1 e permita o tráfego de informação entre ela e uma rede exterior:

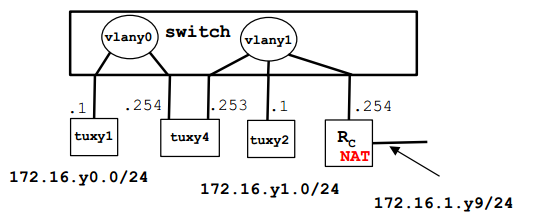


Figura 5 - Esquema da experiência 4 com redes virtuais

Depois de configurar o *switch* para incluir o router na Vlan1 e de configurar as portas do *router* para se ligarem ao *switch* e à rede exterior, definindo os seus IPs, é necessário adicionar as rotas que farão parte da tabela do *router*. Estas rotas podem ser estáticas ou dinâmicas, conforme sejam declaradas manualmente ou através de um protocolo de *routing*. No projeto, foram acrescentadas rotas estáticas ao router através dos seguintes comandos:

ip route 0.0.0.0 0.0.0.0 172.16.1.254

ip route 172.16.Y0.0 255.255.255.0 172.16.Y1.253

O primeiro define o *default* *gateway* para informação que chegue ao *router*, que, neste caso, é a porta que se encontra ligada à rede exterior. O segundo define que pacotes que cheguem ao router com destino à rede Vlan0 devem ser redirecionados para o tuxy4, que funciona como *router* entre a Vlan1 e Vlan0. Assim, pacotes com outro destino que não a Vlan0 vão para o *default gateway*.

Esta gestão de rotas depende de pacotes ICMP (ICMP *redirects*) que controlam o trajeto dos pacotes e notificam os *hosts* da camada 2 das rotas apropriadas para a informação.

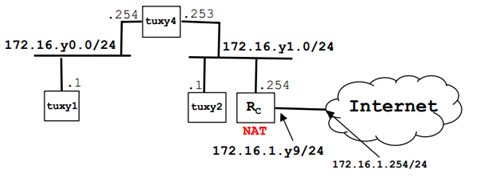


Figura 6 - Esquema da experiência 4 com dispositivos

De seguida, é necessário adicionar as rotas para o *router* nos computadores da Vlan1. Para tal, estabelece-se que a rota por defeito de pacotes enviados a partir do tuxy4 e do tuxy2 é o router comercial:

# route add default gw 172.16.41.254

Resta permitir que o *router* se possa ligar a endereços da Internet, ativando o protocolo NAT (*Network Address Translation*) no *router*. A NAT utiliza uma tabela de *hash* para modificar o endereço IP, de acordo com a rede onde teve origem. Desta forma, o pacote pode circular na Internet e chegar ao destino. Se uma resposta for enviada, o endereço do pacote passa de novo por *hashing* e chega à rede original. O objetivo do NAT é reduzir o número de endereços IP únicos na Internet, que, antes do aparecimento do IPv6, começavam a escassear.

Para configurar o NAT é necessário:

* Indicar qual porta representa o contacto com a rede externa e qual representa o contacto com a rede interna:
  + ip nat inside
  + ip nat outside
* Configurar o NAT *overload*, isto é, permitir que o IP público atribuído ao router possa ser usado por vários *hosts* internamente de forma concorrente:
  + ip nat pool ovrld 172.16.1.**Y**9 172.16.1.**Y**9 prefix 24
  + ip nat inside source list 1 pool ovrld overload
* Indicar quais são os IPs que têm acesso ao IP público do router:
  + access-list 1 permit 172.16.**Y**0.0 0.0.0.7
  + access-list 1 permit 172.16.**Y**1.0 0.0.0.7

Na configuração exemplo, computadores pertencentes à Vlan0 ou Vlan1 que tenham como último número de IP um número entre 0 e 7 podem usufruir do IP público atribuído ao router 172.16.1.Y9.

## Experiência 5: DNS

O objetivo desta experiência era aceder a redes externas, conseguindo assim, aceder à Internet através da rede interna criada. Para tal, foi necessário configurar o DNS.

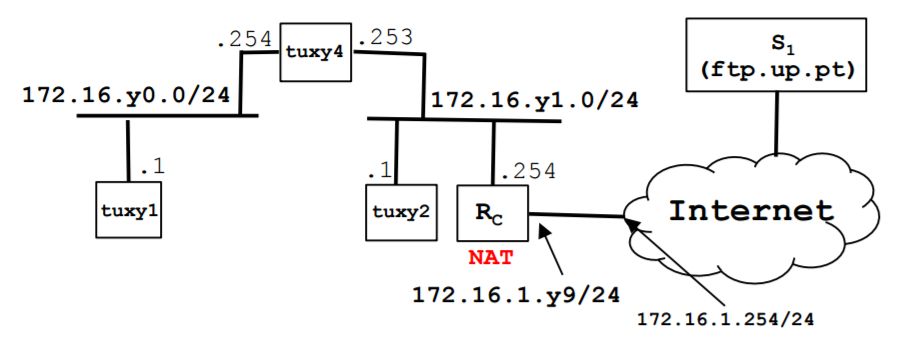


Figura 7 - Esquema da Experiência 5

Para essa configuração é necessário aceder e editar o ficheiro *etc/resolv.conf* utilizando os seguintes comandos.

search netlab.fe.up.pt

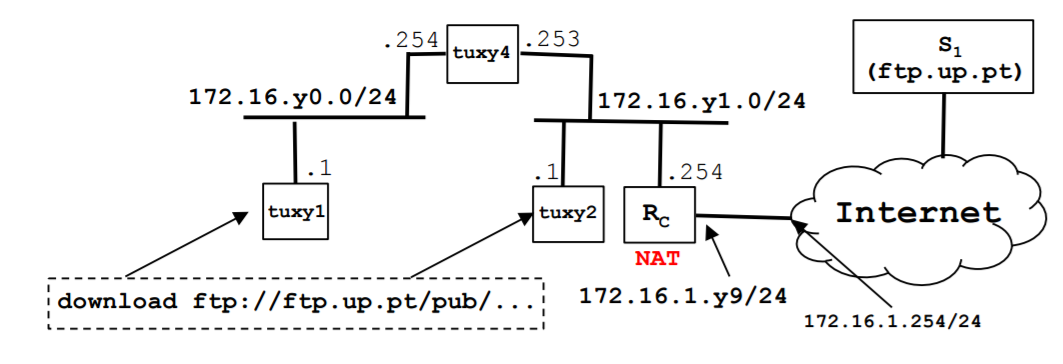
nameserver 172.16.1.2

O parâmetro *search* permite a introdução de *hostnames* curtos que serão resolvidos dentro do domínio especificado, e o parâmetro *nameserver* representa o endereço IP que funcionará como DNS.

Como teste foi feito *ping*, usando [www.google.com](http://www.google.com), que resultou no DNS perguntar a informação contida num dado *domain name* e a obter como resposta o tempo de vida e o tamanho do pacote de dados.

## Experiência 6: conexões TCP

Nesta experiência foi usada a aplicação de download desenvolvida na primeira parte do projeto. A aplicação foi compilada e executada usando um servidor FTP e efetuando o download de um ficheiro. O download foi efetuado corretamente, o que demonstrou que a rede estava bem configurada, não trazendo qualquer problema no acesso por protocolo FTP, assim como a utilização de um servidor exterior à rede.



TCP utiliza *Selective Repeat ARQ*, que é semelhante ao GO-BACK-N ARQ, com a diferença de o receto não deixar de processar os *frames* recebidos quando deteta um erro. Quando a falha de um *frame* é detetada o recetor continua um *acknowledgement*  com o número do *frame* que falhou. Continua a receber e processar os *frames* seguintes, enviando sempre, no *acknowledgemente*, o número do *frame* que falhou primeiro. No final do envio, o emissor verifica os *acknowledgements* e reenvia os *frames* perdido.

## Experiência 7: implementar NAT em Linux

Tendo-se já configurado NAT num *router*, segue-se a configuração de NAT num computador com *kernel* Linux, mais precisamente no tuxy4, que já tem *IP forwarding* ativo.

*Iptables* é uma *firewall* que vem instalada numa distribuição Linux e que permite controlar o tráfego de informação num computador. Para “instalar” a NAT, é necessário atualizar as regras da *iptables* para que faça o *masquerading* dos pacotes IP e os redirecione da rede interior para a rede exterior.

Imagine-se que agora se envia, a partir do tuxy1, um pacote para o IP 8.8.8.8 ([www.google.com](http://www.google.com)). Ao capturar *Eth0,* ligada à rede interior, e *Eth1*, ligada à rede exterior, verifica-se o seguinte:

|  |  |  |  |
| --- | --- | --- | --- |
| Eth0 | | Eth1 | |
| *Source* | ***Destination*** | ***Source*** | ***Destination*** |
| Tuxy1 | 8.8.8.8 | IP público do tuxy4 | 8.8.8.8 |

E ao receber um pacote de volta:

|  |  |  |  |
| --- | --- | --- | --- |
| Eth0 | | Eth1 | |
| *Source* | ***Destination*** | ***Source*** | ***Destination*** |
| 8.8.8.8 | Tuxy1 | 8.8.8.8 | IP público do tuxy4 |

O tuxy4, ao receber o pacote, altera-lhe o endereço IP de origem conforme o endereço público que lhe foi atribuído, guardando informação sobre o IP e a *port* de destino e o IP de origem local. Quando recebe uma resposta de volta, acede aos dados que tinha guardado e determina que o pacote vindo de 8.8.8.8 tem como destino o tuxy1, redirecionando-o.

# Conclusões

A interligação que se observa entre todos os *gadgets* de hoje em dia é permitida pela atuação invisível de vários protocolos de comunicação, conjugados para criar redes computacionais e transferir dados por todo o globo. Os documentos RFC (*Request for Comments*) escritos pela Internet Engineering Task Force contribuem para uma uniformização de normas utilizadas, que, por sua vez, permitem ligar criar a *Web* que nos liga.