**Faculdade de Engenharia da Universidade do Porto**



2º Trabalho Laboratorial de Redes de Computadores:

Desenvolvimento de uma aplicação de *download*  
&   
Configuração de uma rede

**20 de dezembro de 2016**

**Turma 7, Grupo 5:**

Andreia Rodrigues - [up201404691@fe.up.pt](mailto:up201404691@fe.up.pt)

Eduardo Leite - [gei12068@fe.up.pt](mailto:gei12068@fe.up.pt)

Gonçalo Leão - [up201406036@fe.up.pt](mailto:up201406036@fe.up.pt)

Francisco Queirós - [up201404326@fe.up.pt](mailto:up201404326@fe.up.pt)

# 

# 

# 

# **Sumário**

No âmbito da unidade curricular de Redes de Computadores, foi-nos proposto um segundo trabalho laboratorial composto por duas partes. A primeira parte teve como objetivo desenvolver uma aplicação de *download* de ficheiros usando o *File Transfer Protocol* (FTP), enquanto que a segunda parte consistiu na configuração de uma rede de computadores.

Os elementos do grupo retiram como principais conclusões que este trabalho permitiu consolidar uma vasta gama de tópicos abordados nas aulas teóricas de Redes de Computadores, incluindo *switching* de pacotes ao nível da camada de rede, endereços *Internet Protocol* (IP), *Domain Name System* (DNS) e *Network Address Network* (NAT).

# **Índice**

[**Sumário**](#_bl62uwdgp9z5)

[**Índice**](#_pavhkiez8t0c)

[**Introdução**](#_xx07k0wmuh35)

[**Desenvolvimento da aplicação de download**](#_rxslv5x9s9nf)

[**Arquitetura**](#_rgrzsr3w8yw1)

[**Fluxo de execução do programa**](#_ouhopqlq4x8w)

[**Resultados de um download**](#_kuk15awxqpdn)

[**Configuração e análise de uma rede**](#_ozk9l9wxlk3f)

[**Experiência 1 - Configurar de uma rede IP**](#_tgyo5mkhf7da)

[**Experiência 2 - Implementar duas LANs virtuais num switch**](#_o0wlqnemuotq)

[**Experiência 3 - Configurar um router em Linux**](#_q3b1qgyf5ove)

[**Experiência 4 - Configurar um router comercial e implementar o NAT**](#_qv3x9bfqrh76)

[**Experiência 5 - DNS**](#_texe7uftpxng)

[**Experiência 6 - Ligações TCP**](#_wxi1orl177pk)

[**Experiência 7 - Implementar o NAT em Linux**](#_5z2x9u6zpya)

[**Conclusões**](#_yyrhb9skg2n2)

# 

# **Introdução**

O segundo trabalho laboratorial da unidade curricular de Redes de Computadores, do primeiro semestre do terceiro ano do Mestrado Integrado em Engenharia Informática e Computação, teve como objetivo a elaboração de uma aplicação para *download de ficheiros* e a configuração de uma rede de computadores.

Para a aplicação de *download*, desenvolvida em linguagem C e em ambiente Linux, o objectivo em concreto era desenvolver um cliente FTP capaz de comunicar com um servidor remoto usando *sockets* TCP para descarregar um ficheiro da Internet, identificado por um *Uniform Resource Locator* (URL). Para isso, foi necessário a leitura de dois *Request for Comments’s* (RFCs). O RFC 1738 específica os *standards* da Internet para URL’s, enquanto que o RFC 959 descreve em detalhe o funcionamento do protocolo FTP. A tradução do nome de domínio para o endereço IP correspondente exigiu o conhecimento do serviço fornecido pelo *Domain Name System* (DNS).

A configuração da rede de computadores teve como principal objetivo correr a aplicação de *download* num dos nós da rede. No guião, esta parte do trabalho foi dividida em diversas experiências que detalham a construção de redes progressivamente mais complexas que envolvem o uso de um *switch* ligado a vários computadores (com DNS configurado) e a um *router*, o qual tinha NAT configurada e estava ligado à Internet. Estas experiências pretendem complementar diversos tópicos abordados nas aulas teóricas sobre as camadas mais altas do modelo TCP/IP (camadas de rede, de transporte e de aplicação).

O objetivo deste relatório é apresentar, de forma sucinta, os passos e as decisões tomadas para o desenvolvimento da aplicação e da rede de computadores.

O seguimento deste relatório subdivide-se nas seguintes partes:

* **Desenvolvimento da aplicação de *download***, onde será apresentada a arquitetura do programa desenvolvido e o resultado da sua execução.
* **Configuração e análise de uma rede**, onde são descritos, para cada experiência realizada, a arquitetura da rede, os objetivos e os principais comandos de configuração. As explicações das experiências são acompanhadas de diversas figuras obtidas ao longo das aulas práticas.
* **Conclusões**.

# **Desenvolvimento da aplicação de *download***

Nesta secção, será descrita a aplicação de cliente FTP desenvolvida para descarregar ficheiros de servidores remotos. Após compilar o programa (poderá ser usado o comando make no mesmo diretório do Makefile fornecido), este poderá ser executado escrevendo na linha de comandos “download <url>”, onde url é o endereço URL do ficheiro a descarregar. A sintaxe do endereço URL é conforme ao RFC 1738 e tem o seguinte formato:

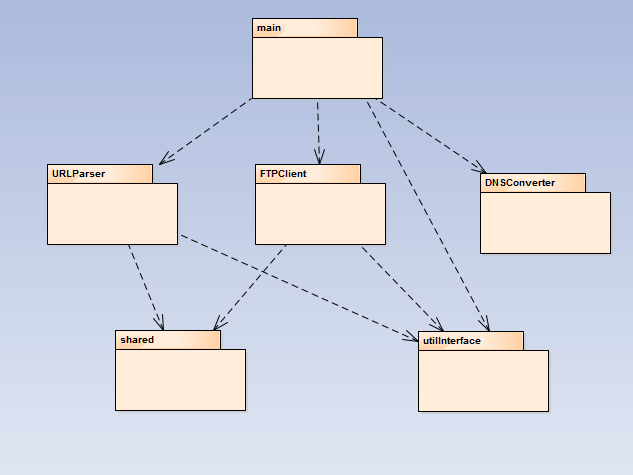
ftp://[<user>:<password>@]<host>[:<port>]/<url-path>

No formato apresentado acima, os campos delimitados por ‘[‘ e ‘]’ representam partes do URL que podem ser omitidas. Em um dado campo opcional não seja fornecido, o programa utiliza um valor por omissão:

* *username* é por omissão “anonymous”.
* *password* é por omissão “mail@domain”.
* *port* é por omissão 21.

## Arquitetura

No diagrama abaixo, pode-se ver o diagrama UML de *packages* da programa, onde cada *package* representa um módulo/bloco funcional (associado a um ficheiro .c e, eventualmente, a um ficheiro .h). As setas representam as dependências entre módulos.



*Figura 1: Diagrama UML de “packages” do programa*

A tabela seguinte descreve a responsabilidade de cada módulo.

|  |  |
| --- | --- |
| **Módulo** | **Descrição** |
| main | Módulo principal do programa. Serve de ponto de partida da sua execução e invoca serviços fornecidos pelos restantes módulos. |
| URLParser | Módulo responsável por fazer *parsing* de URL’s e de armazenar a sua informação num objeto do tipo url\_data. |
| FTPClient | Módulo responsável pela comunicação com um servidor remoto usando o protocolo FTP. |
| DNSConverter | Módulo responsável pela conversação de nomes de *hosts* em endereços IP. |
| shared | Módulo elementar com *macros* MAX\_SIZE, que define o tamanho máximo de diversos campos dos endereços URL’s (como o *path* do ficheiro a descarregar). |
| utilInterface | Módulo com *macros* utilitários para a escrita na consola, nomeadamente para mudar a sua cor de texto e de fundo. |

*Tabela 1 - Responsabilidade de cada módulo da aplicação*

## Fluxo de execução do programa

Para começar, é feito “parsing” ao único argumento do programa, o *Uniform Resource Locator* (URL) associado ao ficheiro a descarregar. Este “parsing” é feito usando a função parseURL do módulo URLParser. Caso o URL esteja conforme ao esquema URL para FTP, especificado pelo RFC 1738, a informação nele contida (*username*, *password*, *host*, *port* e *path*) é armazenada numa struct do tipo url\_data.

É depois feita a conversão do nome do *host* no seu endereço IP usando a função host2Ip do módulo DNSConverter. Esta função usa o método gethostbyname de netdb.h.

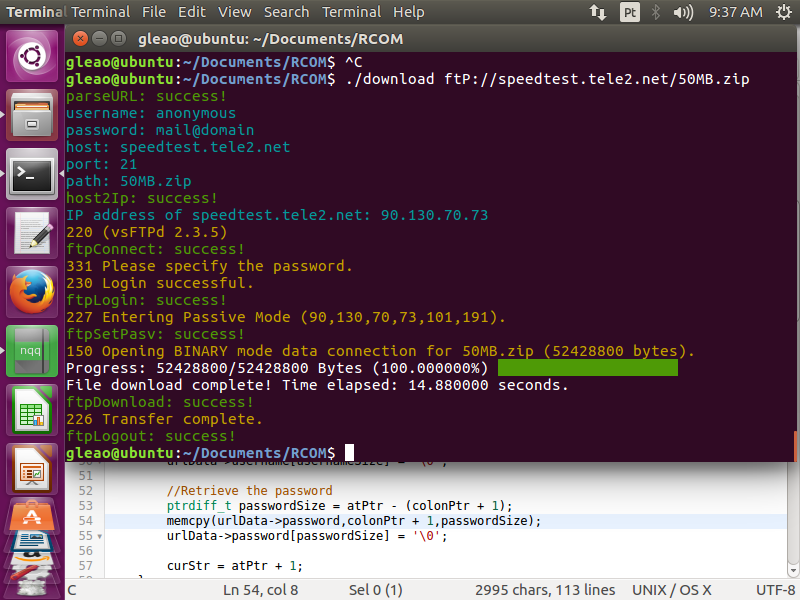
De seguida, são invocadas successivamente funções do módulo FTPClient para comunicar com o servidor remoto para transferir o ficheiro especificado. A sequência de comandos e respostas entre o cliente e o servidor estão conformes ao RFC 959, relativo ao protocolo de aplicação FTP. As funções invocadas são as seguintes (na ordem indicada):

* ftpConnect é usado para estabelecer uma conexão com o servidor. Em particular, é criado um socket que é ligado ao servidor, usando as funções socket e connect. Este socket fica associado ao descritor controlSocketFd, contido num objeto da struct ftp\_data.
* ftpLogin envia os comandos USER e PASS ao servidor, indicando como argumentos, respetivamente, o *username* e a *password* do URL enviado pela linha de comandos.
* ftpSetPasv envia o comando PASV ao servidor sem mais nenhum argumento. Isto tem como efeito colocar o *Data Transfer Process* (DTP) em modo passivo. Como resposta, o servidor especifica o endereço IP e o *port* onde se encontra à espera que seja estabelecida uma segunda conexão para transmitir o ficheiro. Usando esse endereço IP e *port*, um segundo socket, associado ao descritor dataSocketFd em ftp\_data, é conectado ao servidor.
* ftpDownload envia o comando RETR (sem argumentos) através do primeiro socket (socket de controlo) para iniciar a transferência do ficheiro. De seguida, é aberto um ficheiro para guardar a informação recebida do servidor e é recebida a informação do ficheiro, de forma ordenada, através do segundo socket (socket de dados). No final, é fechado tanto o ficheiro de *output* como o *socket* de dados.
* ftpLogout envia o comando QUIT (sem argumentos) para terminar a ligação com o servidor e é fechado o socket de controlo.

Durante a execução do programa, caso ocorra um erro e algo inesperado (por exemplo, o servidor envia uma resposta associada a um erro), o programa imprime uma mensagem de erro e termina.

## Resultados de um *download*

Para demonstrar o bom funcionamento da aplicação desenvolvida, foram realizadas várias experiências de *download* de diferentes ficheiros de diversos servidores. Na figura abaixo, é apresentado o *output* para a consola após ter sido escrito o comando ./download ftP://speedtest.tele2.net/50MB.zip.



*Figura 2: Resultados de um “download” usando o cliente FTP*

# **Configuração e análise de uma rede**

## Experiência 1 - Configurar de uma rede IP

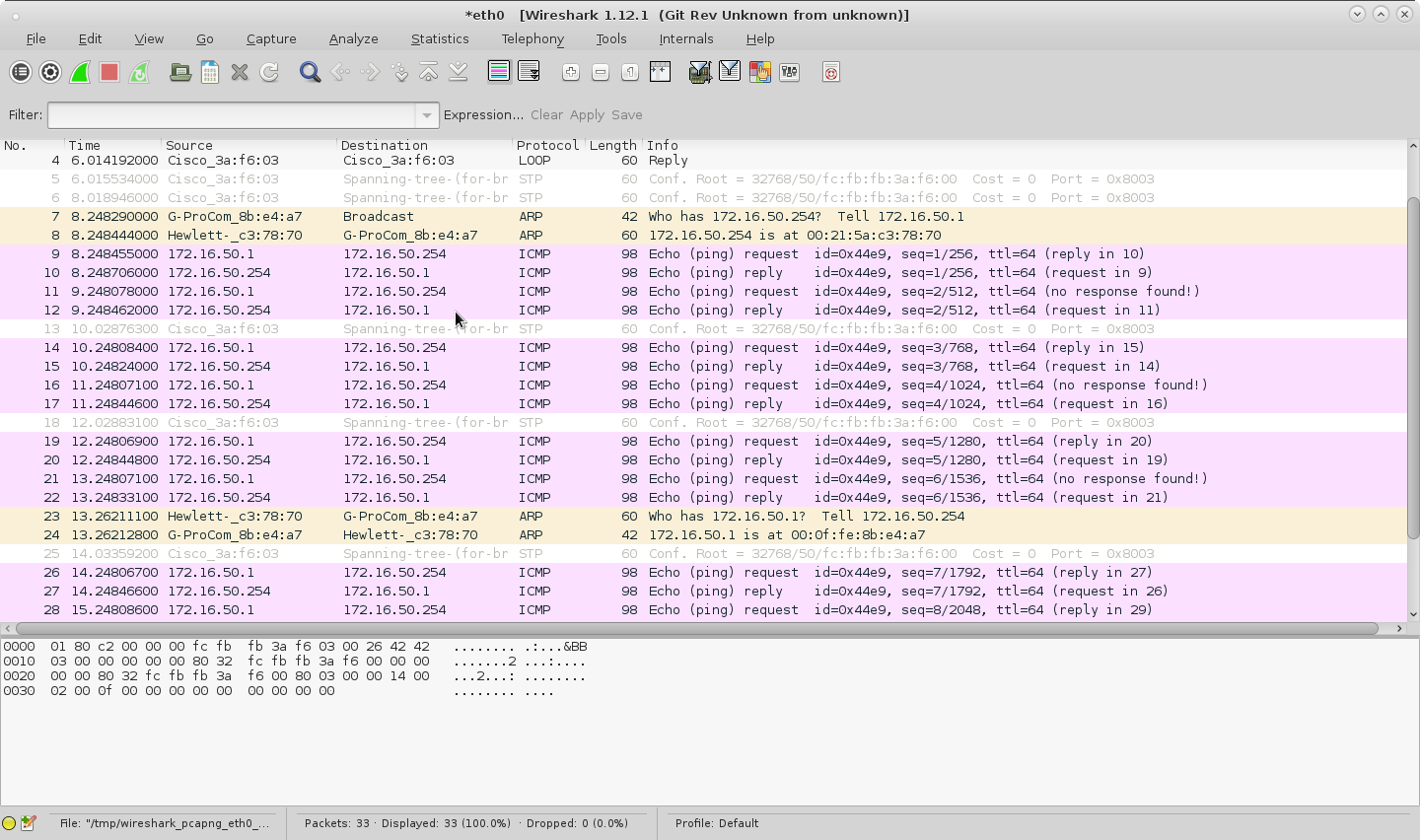
**Objetivos**: - associar endereços IP a *interfaces* de computadores, para que estes possam comunicar entre si.

**Arquitetura da rede**: A rede é composta por duas máquinas (máquina 1 e 4) ligados a um *switch*.

Para configurar o endereço IP associado a um dado interface de uma máquina, basta usar o comando ifconfig com a seguinte sintaxe: ifconfig <interface> <endereço IP>/<máscara da rede>

Para enviar um pacote, um elemento da rede precisa de conhecer o endereço MAC (ao nível da ligação de dados) do destinatário, sabendo o seu endereço IP (ao nível da rede). Para isso, o emissor envia um pacote de *Address Resolution Protocol* (ARP) por *broadcast* com o endereço IP do destinatário e aguarda por um pacote ARP de resposta com o endereço MAC correspondente.

Na máquina 1, foi usado o comando ping 172.16.Y0.254 (sendo Y o número da bancada) para enviar pacotes para a *interface* eth0 da máquina 4. Com o comando *ping*, são gerados pacotes de *Internet Control Message Protocol* (ICMP).



*Figura 3: Captura em Wireshark na interface eth0 da máquina 1 dos pacotes gerados pelo comando ping*

Os pacotes ARP enviados pela máquina 1 têm como endereços MAC e IP de origem os endereços respectivos da eth0 da máquina 1. Os endereços MAC e IP de destino são endereços de *broadcast* pois não se sabe à partida que computador tem o endereço MAC procurado.

A partir de uma trama Ethernet, para determinar o protocolo de rede (IP, ARP, …) do pacote do seu *payload*, é preciso ler o cabeçalho (mais precisamente, os 21º e 22º bits). Caso se trate de um pacote IP, o seu protocolo de camada de transporte (ICMP, …) pode ser determinado pelos bits 73 a 80.

Sendo o tamanho do *header* e *trailer* da trama Ethernet constantes, o que varia de comprimento é o seu *payload*. Caso se trate de um pacote IP, o seu tamanho é indicado pelos bits 17 a 32.

O *loopback interface* é usado para um computador enviar pacotes para ele mesmo sem necessidade de ligações externas. Isto permite diagnosticar eventuais problemas ou aceder aos seus próprios serviços de rede, sendo o mesmo computador cliente e servidor simultaneamente.

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

**Objetivos**: - configurar VLANs num *switch*.

**Arquitetura da rede**: A rede é composta por três máquinas (máquinas 1, 2 e 4) ligados a um switch, com duas *virtual LANs* (VLANs). Uma VLAN tem as máquinas 1 e 4, a outra, a máquina 2.

Para configurar o *switch*, insere-se o comando configure terminal. De seguida, para definir as duas VLAN’s, escreve-se o comando vlan <número da vlan>, seguido de exit, para sair da configuração dessa VLAN. Assim, para definir a vlan Y0, escreve-se vlan Y0 e exit. Por fim, para associar as portas do *switch* a uma dada VLAN, executa-se o seguinte conjunto de comandos: interface fastethernet 0/<número da porta do switch>, switchport mode access, switchport access vlan <número da vlan>, exit

Existem dois domínios de *broadcast*, correspondentes à cada VLAN. Para mostrar a existência destes domínios, tentou-se fazer *ping* do máquina 2 a partir da máquina 1, este tendo falhado, o que mostra que existem pelo menos dois domínios distintos.

## Experiência 3 - Configurar um *router* em Linux

**Objetivos**: - configurar um computador para servir de *router* para que se haja comunicação entre as duas VLANs.

**Arquitetura da rede**: Similar à experiência 2, mas incluindo a máquina4 também na VLAN Y1 (usando outra *interface*).

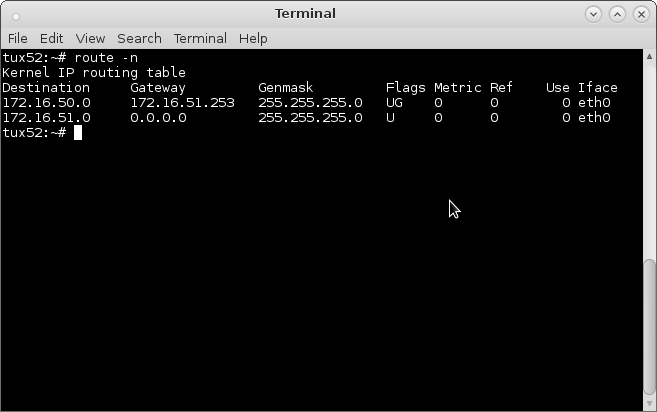
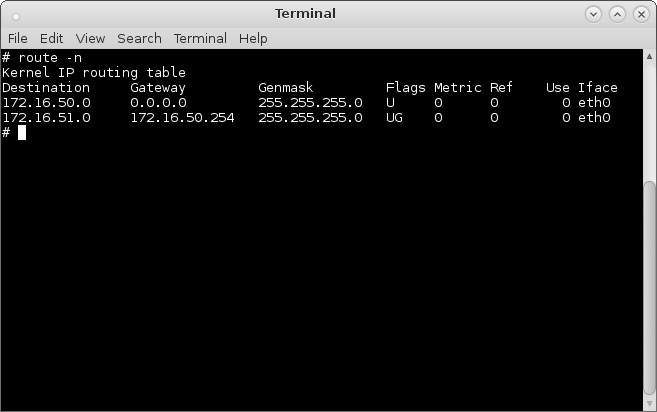
Começa-se por ligar a *interface* eth1 da máquina 4 à VLAN1 de forma análoga ao descrito na experiência 2.

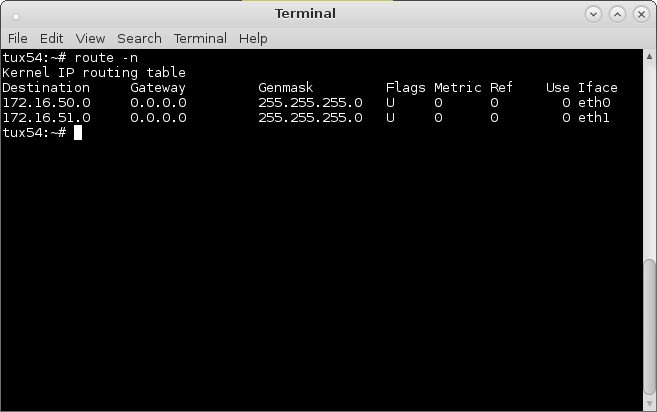
Nas máquinas das “extremidades” (máquinas 1 e 2), é preciso configurar as rotas de encaminhamento de pacotes usando o comando route.

Na máquina 1, escreve-se o comando route add -net 172.16.Y1.0/24 gw 172.16.Y0.254. O primeiro endereço especifica o conjunto de endereços IP para os quais será definido o encaminhamento (neste caso, os endereço do domínio Y1), com o uso de uma máscara. O segundo endereço indica para onde se devem encaminhar os pacotes (neste caso, para a *interface* eth0 da máquina 4).

Na máquina 2, escreve-se o comando route add -net 172.16.Y0.0/24 gw 172.16.Y1.253.

Para ativar o encaminhamento de pacotes, é preciso executar o comando echo 1> /proc/sys/net/ipv4/ip\_forward

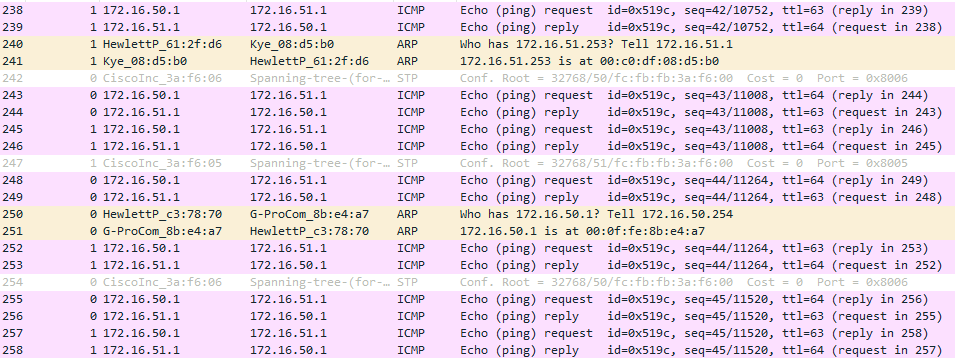




*Figuras 4, 5 e 6: Tabelas de encaminhamento de pacotes as máquinas 1, 2 e 4, respetivamente (obtidas usando o comando route -n)*

As informações mais importantes contidas na entrada da tabela de encaminhamento são os endereços para os quais será definido o encaminhamento (*Gateway*, juntamente com a máscara, em *Genmask*), o endereço IP de destino de encaminhamento (*Destination*) e a *interface* de saída dos pacotes (*Iface*).

Para testar se o *router* foi bem configurado, foi executado o comando ping 172.16.Y1.1 a partir da máquina 1.



*Figura 7: Captura no Wireshark nos interfaces eth0 e eth1 da máquina 4 dos pacotes gerados pelo comando ping*

A máquina 1 começa por consultar a sua tabela de encaminhamento para saber para quem enviar o pacote e conclui que pacotes para a máquina 2 devem ser enviados para a *interface* eth0 da máquina 4, pelo que envia um pacote ARP a pedir o endereço MAC desta *interface*. Ao receber um pacote ARP de resposta por parte da máquina 4, a máquina 1 envia um pacote ICMP à máquina 4.

Na máquina 4, é feito um processamento semelhante: primeiro é consultada a tabela de encaminhamento de pacotes, depois é enviado um comando ARP a pedir o endereço MAC da *interface* eth0 da máquina 2, e por fim é enviado um pacote ICMP a esta *interface*.

A partir da máquina 2, é feito o percurso inverso e de forma análoga, para que a máquina 1 receba pacotes ICMP de *reply*.

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

**Objetivos**: - configurar um *router* comercial.  
 - implementar NAT no *router*.

**Arquitetura da rede**: Similar à experiência 3, mas incluindo um *router* comercial na VLAN Y1, ligado pela porta 0. O *router* está ligado à rede do laboratório pela porta 1.

Para configurar o *router*, insere-se o comando conf t.

De seguida, para associar endereços IP às portas do *router*, executa-se o seguinte conjunto de comandos.

Para a porta 0: interface gigabitethernet 0/0, ip address 172.16.Y1.254 255.255.255.0, no shutdown, ip nat inside e exit

Para a porta 1: interface gigabitethernet 0/1, ip address 172.16.1.Y9 255.255.255.0, no shutdown, ip nat outside e exit

O comando no shutdown é usado para ativar a porta.

A penúltima linha das duas configurações acima define que a porta 0 faça parte da rede privada e que a porta 1 faça interface com o exterior (para a NAT).

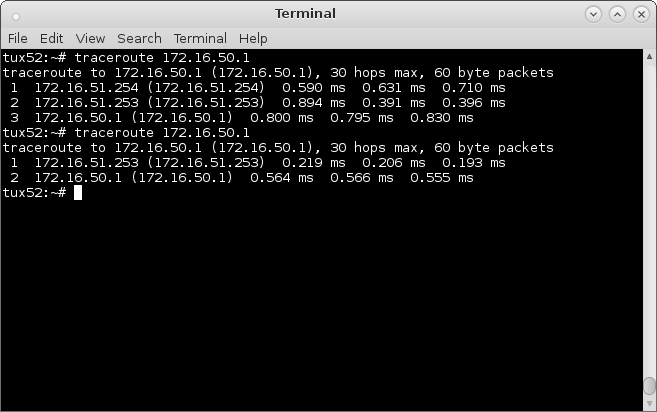
Para configurar as rotas estáticas, foram executados os comandos ip route 0.0.0.0 0.0.0.0 172.16.1.254 e ip route 172.16.Y0.0 255.255.255.0 172.16.Y1.253

O primeiro comando indica que por omissão os pacotes devem ser enviados para o *router* da rede do laboratório.

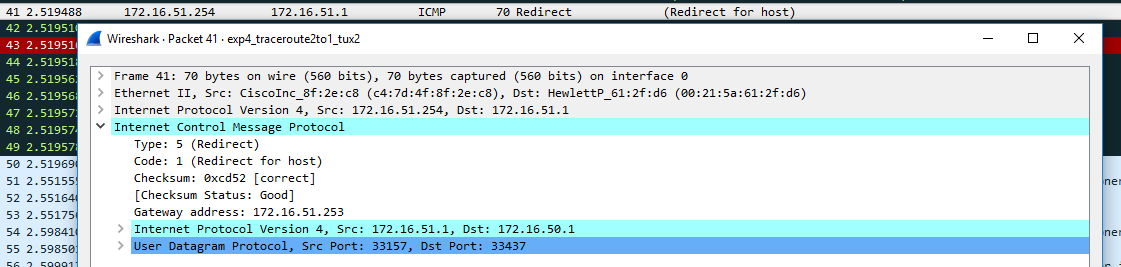
O segundo comando indica que, caso o pacote tenha como destino um elemento da VLAN Y0, deve ser redirecionado para a *interface* eth1 da máquina 4.

Por fim, é preciso indicar que, por omissão, os pacotes da máquina 2 são redirecionados para a porta 0 do *router* (route add default gw 172.16.Y1.254).

Se for retirada a rota da máquina 2 que redireciona pacotes da VLAN Y0 para o eth1 da máquina 4 (route delete -net 172.16.Y1.0/24) e mas mantendo ativo o redirecionamento de pacotes (echo 1 > /proc/sys/net/ipv4/conf/eth0/accept\_redirects seguido de echo 1 > /proc/sys/net/ipv4/conf/all/accept\_redirects) quando se faz *traceroute* da máquina 2 para eth0 da máquina 1, os pacotes passam primeiro pelo *router* comercial, tal como se pode ver na figura 8. Isto deve-se a já não haver uma rota direta a ligar a máquina 2 à máquina 4. Contudo, se o comando *traceroute* for executado uma segunda vez, tal como se pode ver na figura 8, os pacotes já não passam pelo *router* pois, da primeira vez que este comando foi executado, a máquina 2 recebeu um pacote ICMP redirect do *router* (figura 9) a indicar que os pacotes para a máquina 1 podem ser diretamente entregues à máquina 4, o que atualizou a tabela de encaminhamento de pacotes da máquina 2.



*Figura 8: Output do traceroute para a máquina 2 a partir da máquina 1*



*Figura 9: Conteúdo do pacote ICMP redirect enviado à máquina 2 pelo router*

Para configurar a gama de endereços externos a serem usados pela NAT, executou-se o comando ip nat pool ovrld 172.16.1.Y9 172.16.1.Y9 prefix 24.

Para definir os endereços internos que podem usar a NAT, foram executados os comandos: ip nat inside source list 1 pool ovrld overload, access-list 1 permit 172.16.Y0.0 0.0.0.255 e access-list 1 permit 172.16.Y1.0 0.0.0.255

O objetivo da *Network Address Translation* (NAT) é permitir que todos os elementos da rede interna comuniquem com o exterior usando um número reduzido de endereços IP externos. A NAT distingue os elementos da rede interna pela porta que é usada na comunicação com esses endereços IP.

## Experiência 5 - DNS

**Objetivos**: - configurar o DNS nas máquinas.

**Arquitetura da rede**: Igual à experiência 4.

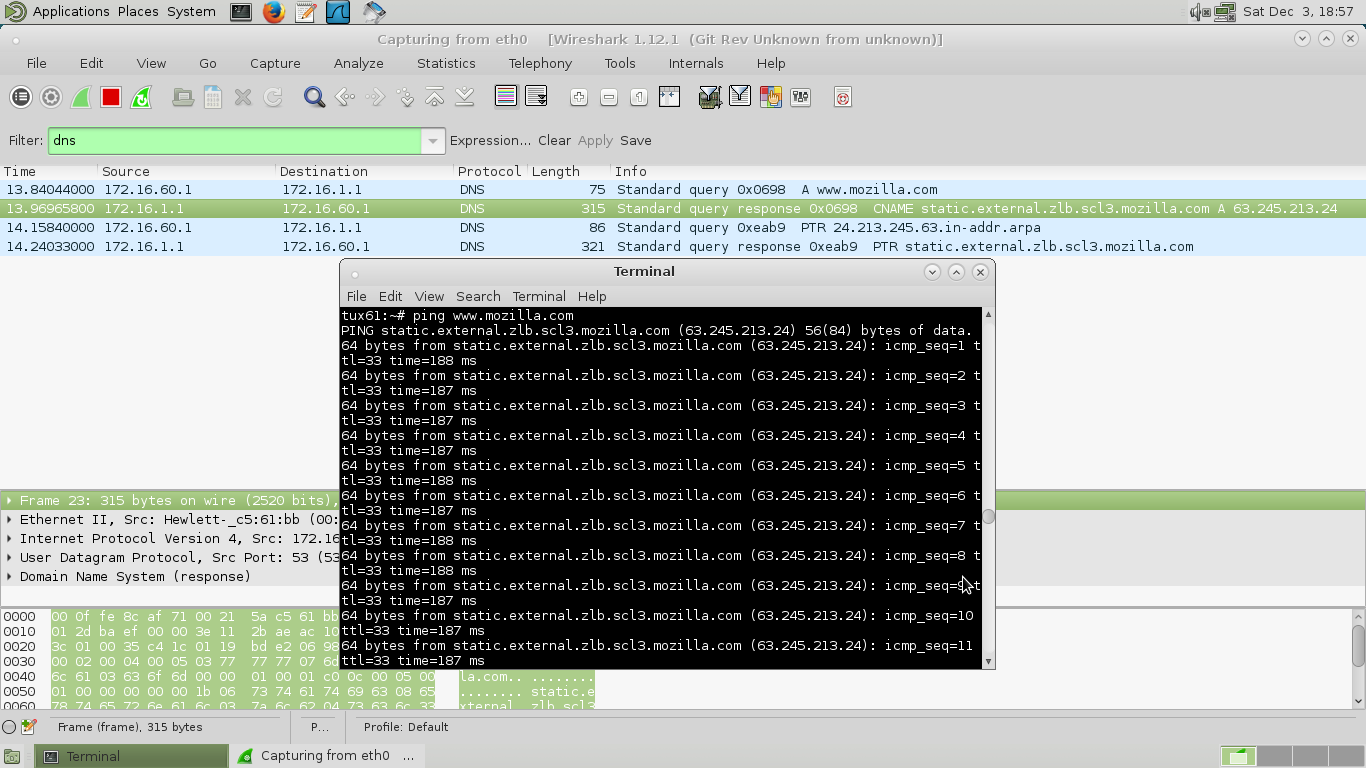
Com o *Domain Name System* (DNS), torna-se possível aceder a domínios da Internet pelo seu nome em vez de ter de se especificar o seu endereço IP, o que é mais prático para humanos.

Para configurar o DNS num dado computador, é necessário editar o ficheiro de configuração do *DNS resolver* do sistema operativo, /etc/resolv.conf, e incluir as seguintes linhas: search netlab.fe.up.pt e nameserver 172.16.1.1.

A primeira linha é uma diretiva para acrescentar uma entrada à *search list* usada pelo *DNS resolver*. Sempre que o *resolver* receber um nome de domínio com menos que um certo número de pontos (‘.’) (por omissão, menos que um ponto), é procurado o endereço IP do domínio cujo nome é o obtido concatenando o nome dado com um ponto (‘.’) seguido de cada entrada da *search list*, individualmente.

A segunda linha é uma diretiva para indicar o endereço IP de um *name server* (neste caso, o da Netlab da FEUP). Os *name servers* são usados pelo *resolver* para fazer a conversão de um *domain name* no seu endereço IP.

Para testar se o DNS foi bem configurado, foi corrido o comando ping www.mozilla.com na máquina 1. Os pacotes trocados usando o protocolo DNS são apresentados na figura 10.



*Figura 10: Captura pelo Wireshark na interface eth0 da máquina 1 dos pacotes DNS gerados pelo comando ping*

Numa primeira fase, a máquina 1 envia ao *name server* um pacote DNS para pedir o endereço IP de www.mozilla.com (*query*).

Numa segunda fase, a máquina obtém uma resposta do servidor (*reply*) com o endereço IP pedido (63.245.213.24) no campo A e com o nome canónico de www.mozilla.com (static.external.zlb.scl3.mozilla.com) no campo CNAME.

## Experiência 6 - Ligações TCP

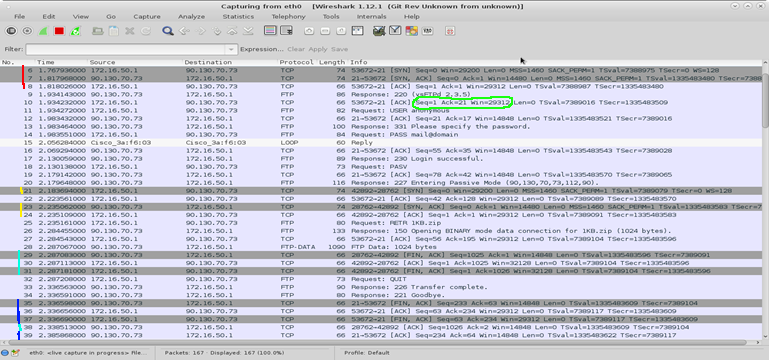
**Objetivos**: - correr o cliente FTP desenvolvido para fazer *download* de um ficheiro da Internet

**Arquitetura da rede**: Igual à experiência 4.

Ao correr a aplicação *File Transfer Protocol* (FTP), são abertas duas conexões de *Transmission Control Protocol* (TCP), uma para cada conexão FTP estabelecida. Uma das conexões serve para o cliente enviar comandos de controle para o servidor (na figura 11, são usadas as portas 53672 e 21 para o cliente e servidor, respetivamente), enquanto que a outra é usada para transferir o ficheiro (na figura.11, usam-se as portas 42892 e 28762 para o cliente e servidor, respetivamente).

Dado que o protocolo TCP é orientado à conexão, uma conexão deste protocolo tem três fases:

* estabelecimento da conexão (onde é usado um *three-way handshake*).
* envio de dados.
* término da conexão (onde é usado um *four-way handshake*).



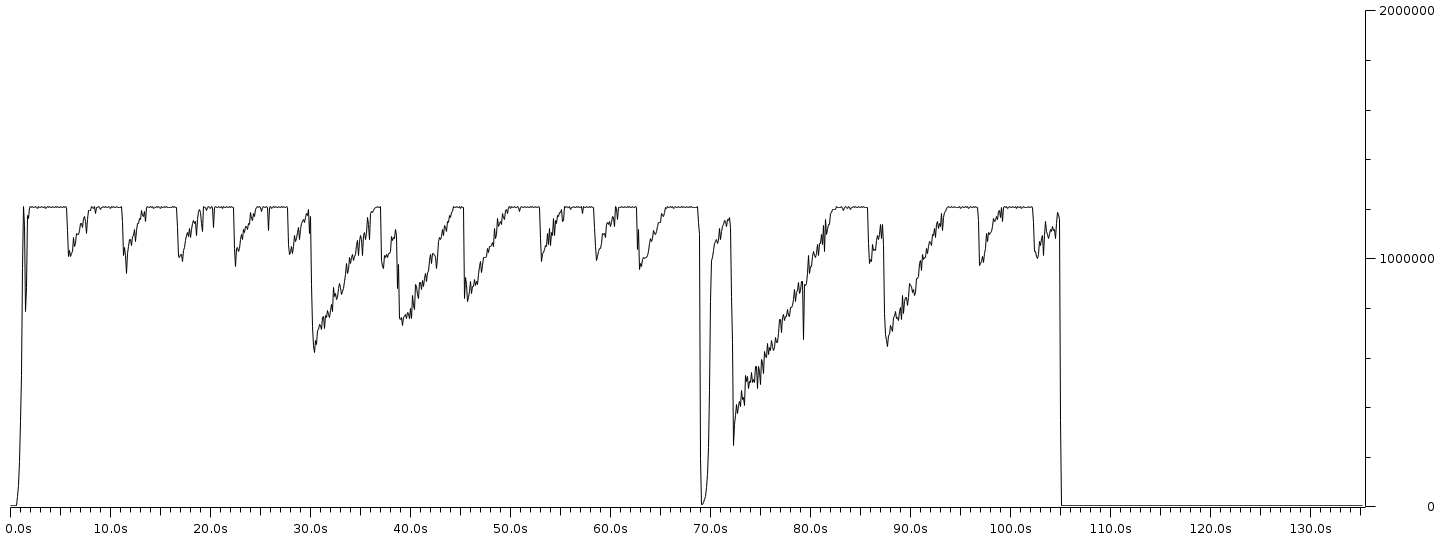
*Figura 11: Captura no Wireshark na interface eth0 da máquina 1 dos pacotes gerados pela aplicação de download (obtida com o comando ./download ftp://speedtest.tele2.net/1KB.zip na máquina 1)*

Na figura 11, os pacotes assinalados a vermelho e amarelo representam os pacotes usados para estabelecimento das ligações TCP de controle e de transferência do ficheiro, respetivamente. Os pacotes a azul claro e azul escuro representam os pacotes respetivos de término da ligação de controle e de transferência do ficheiro.

O mecanismo de Automatic Repeat reQuest (ARQ) usado em TCP é uma variação do Go Back N, fundamentado sobre o conceito de “janela” de sequências de bytes. Tanto o receptor como o transmissor vão ter uma janela que depende de fatores como bytes enviados, recebidos, lidos, entre outros. Do lado do emissor, permite saber que pacotes (re)transmitir e do lado do receptor, que pacotes se espera receber.

Os campos dos pacotes TCP relevantes para o mecanismo de ARQ são o *Sequence Number*, o *Acknowledgement Number* e o *Window Size* (estes campos num dos pacotes estão assinalados a verde na figura 11).

Em TCP, o mecanismo de controlo de congestão usa os ACKs recebidos para regular a transmissão de novos pacotes. O mecanismo faz uso de uma “janela de congestão”, que corresponde a uma estimativa da quantidade de bytes que podem ser enviados. O campo dos pacotes TCP mais relevante para este mecanismo é o *Window Size*.



*Figura 12: Número de bytes recebidos por segundo pela máquina 1 ao longo do tempo (obtida com o comando ./download ftp://speedtest.tele2.net/1GB.zip na máquina 1)*

Ao fazer o *download* de um ficheiro usando o cliente FTP desenvolvido, o número de bytes recebidos por segundo tem tendência a ir aumentando ao longo do tempo com algumas quedas abruptas, conferindo ao gráfico respetivo um aspeto de “dente de serra”. Esta observação é coerente com os mecanismos de controlo de congestão usados em TCP, como o *Additive Increase/Multiplicative Decrease*, em que a perda de um pacote faz encolher bastante a dimensão da janela de congestão, enquanto que incrementos ao comprimento desta são mais graduais e de menor valor.

Ao correr simultâneamente o cliente FTP nas máquinas 1 e 2, constata-se que o número de bytes recebidos no máquina 1 cada segundo é bastante menor pois é mais quebras no tamanho da janela de congestão por haver bastante mais tráfego. Como consequência, a transferência de um ficheiro demora mais tempo.

## Experiência 7 - Implementar o NAT em Linux

**Objetivos**: - configurar NAT num computador.

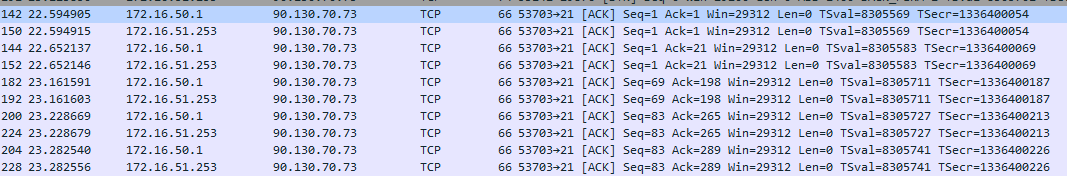
**Arquitetura da rede**: Igual à experiência 4.

Para implementar NAT na máquina 4, foi usado o comando iptables. Os comandos utilizados foram: iptables -t nat -A POSTROUTING -o eth1 -j MASQUERADE, iptables -A FORWARD -i eth1 -m state --state NEW,INVALID -j DROP

O primeiro comando especifica que a NAT é implementado em modo MASQUERADE, para que pacotes vindos de elementos (máquina 1) da rede interna, ao passarem pela máquina 4, passaram a ter como endereço IP de origem o endereço da eth1 da máquina 4, que não tem de ser especificado, o que é útil caso se esteja a usar endereços dinâmicos (DHCP).

O segundo comando implementa uma regra de segurança que especifica que todas as ligações novas ou inválidas do exterior para o interior da rede privada devem ser ignoradas.

Para mostrar o funcionamento da NAT, geraram-se diferentes tipos de pacotes na máquina 1 para a Internet, usando diferentes comandos (*wget* para TCP, *traceroute* para UDP e *ping* para ICMP). Na figura 13, é possível ver alguns pacotes gerados usando o comando wget ftp://speedtest.tele2.net/1KB.zip a partir da máquina 1. A captura por Wireshark ocorreu nas interfaces eth0 e eth1 da máquina 4, onde se pode observar que os pacotes vindos da máquina 1 (para a eth0 da máquina 4) saem pela eth1 com o endereço público da NAT (172.16.Y1.253) no campo do endereço de origem.



*Figura 13: Captura pelo Wireshark nas interfaces eth0 e eth1 da máquina 4 de pacotes TCP gerados pelo wget*

# **Conclusões**

Com o cliente FTP desenvolvido, foi possível descarregar com sucesso ficheiros da Internet usando *sockets*. Esta primeira parte do trabalho permitiu aos elementos do grupo perceber melhor o funcionamento de um protocolo do nível de aplicação (FTP) e também do nível de transporte (TCP) de um ponto de vista mais prático. Esta tarefa também permitiu ilustrar a importância da leitura de RFC’s para conhecer as convenções e padrões usados na Internet.

As diversas experiências realizadas no laboratório culminaram no uso do cliente FTP desenvolvido para ilustrar o bom funcionamento da rede construída. Os objetivos de aprendizagem desta segunda parte também foram atingidos pois todos os membros do grupo conseguiram aprofundar os seus conhecimentos acerca de diversos tópicos do nível de rede, de transporte e de aplicação da pilha de protocolos TCP/IP, complementando assim os conhecimentos obtidos no primeiro trabalho laboratorial.