Trabalho 2

Configuração de uma rede e desenvolvimento de uma aplicação de download

Mestrado Integrado em Engenharia Informática e Computação

Redes de Computadores

Turma 6:
Bernardo Santos - up201706534
David Silva - up201705373
Luís Cunha - up201706736

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

23 de Dezembro de 2019

Resumo

O trabalho 1 da disciplina de RCOM incidiu, essencialmente, na aplicação de conceitos da camada de ligação de dados das redes de computadores. No seguimento desse trabalho, o presente projecto incide sobretudo na camada de rede e de transporte. Este trabalho foi desenvolvido em duas partes. Na primeira parte foi criada uma aplicação que permite, através de uma ligação *TCP* e do uso do protocolo *FTP*, efetuar o download de um ficheiro de um servidor FTP. A segunda parte passa por construir uma rede de acesso local, subdivida em duas redes, na qual pudesse ser testada a aplicação desenvolvida na primeira parte. A criação desta rede implicou a configuração dos computadores intervenientes, de um *switch* e de um *router* comercial.

Conteúdo

1 Introdução **4**

2 Parte 1 - Aplicação de download **4**
2.1 Arquitetura 4

3 Parte 2 - Configuração da rede e análise **5**
3.1 Experiência 1 - Configurar uma rede IP 5
3.2 Experiência 2 - Implementar duas VLAN's num switch 6
3.3 Experiência 3 - Configurar um router em Linux 7
3.4 Experiência 4 - Configurar um router comercial e implementar NAT 8
3.5 Experiência 5 - DNS 10
3.6 Experiência 6 - Ligações TCP 10

4 Conclusões **11**

5 Anexos **12**

A Download bem sucedido **12**

B Tabelas de encaminhamento da experiência 3 **12**

C Mecanismo de ARQ numa conexão TCP **13**

D Gráfico de throughput da experiência 6.4 **13**

E Gráficos de throughput da experiência 6.5 **14**

1 Introdução

O segundo trabalho laboratorial da unidade curricular de Redes de Computadores, teve, como principal objectivo compreender o funcionamento de tecnologias que estão por trás da *internet*. Para tal, foi-nos proposto desenvolver uma aplicação capaz de descarregar um ficheiro através do protocolo FTP e configurar/estudar uma rede através da qual fosse testada a aplicação. Para a rede, foi necessário configurar vários aspectos dos computadores envolvidos, um dos quais a funcionar como *router*, um *switch*, no qual foram configuradas duas VLANs e um *router* comercial, no qual foi implementado NAT. A ligação da rede configurada ao *router* do laboratório e a configuração do DNS permitiram aos computadores da rede ter ligação à *internet* e assim testar a aplicação desenvolvida. O relatório encontra-se dividido em três partes. Na primeira, encontram-se descritos aspetos da arquitectura e utilização da aplicação de *download*. Na segunda, existem seis secções, cada uma dedicada à descrição de uma das experiências da configuração e análise da rede. A última parte consiste na conclusão, onde refletimos sobre os objetivos do trabalho.

2 Parte 1 - Aplicação de download

A aplicação aceita como argumento um endereço de um ficheiro presente num servidor FTP no seguinte formato: ftp://[<user>:<password>@]<host>/<url-path>. E descarrega esse mesmo ficheiro (qualquer tipo de ficheiro pode ser descarregado). Todas as respostas do servidor são impressas na consola, assim como uma barra de progresso para que o utilizador saiba em qualquer momento o estado do programa. Durante o desenvolvimento da aplicação foram consultados o RFC959, referente ao protocolo de transferência de dados e o RFC1738, referente ao tratamento de informação proveniente de *URLs*.

No anexo A está presente um exemplo de um download bem sucedido.

2.1 Arquitetura

Inicialmente é feito o processamento do URL fornecido, de forma a assegurar que se encontra com o formato correto, a função responsável por esta tarefa é denominada de *parse_arguments*. E a informação extraída é colocada na seguinte estrutura:

```
1  struct ftp_st {
2      // ftp://[<user>:<password>@]<host>/<url-path>
3      char* user;
4      char* password;
5      char* host;
6      char* url_path;
7  };
```

No caso de não serem especificados o nome de utilizador e a palavra-passe o programa irá correr em modo anónimo(a variável *user* e *password* assumem o valor *anonymous*). O endereço de IP do servidor é obtido através do *host* com recurso à função *getip* fornecida pelos professores. A porta utilizada é sempre a 21. De seguida é aberto o socket através do qual é feita a comunicação entre o cliente e o servidor e são enviados os comandos *USER user* e *PASS pass* para

efectuar o *login*. Se as credenciais forem válidas é enviado o comando *PASV* que efectua a entrada em modo passivo, ao processar a informação recebida do servidor obtemos a porta necessária para a abertura do segundo socket que tem como finalidade a transferência de dados. É enviado o comando *RETR url_path* que faz o pedido do ficheiro especificado ao servidor e finalmente é efectuada a transferência do ficheiro com a chamada da função *download_file*. No final da execução é enviado o comando *QUIT* para o servidor e após a receber uma resposta são fechadas todas as conexões.

Uma das principais funções é a *send_command* que envia um comando para o servidor e interpreta a resposta através de uma chamada à função *read_response*. Através do primeiro digito recebido é possível saber se a resposta é positiva (1, 2, 3) ou negativa (4 e 5). No caso de ser 1 o *read_response* é chamado novamente para ler a resposta seguinte, caso seja 2 ou 3 retorna, caso seja 4 tenta ler a resposta novamente e caso seja 5 termina a execução do programa.

3 Parte 2 - Configuração da rede e análise

3.1 Experiência 1 - Configurar uma rede IP

A primeira experiência consistiu em configurar os endereços IP e tabelas de encaminhamento de dois computadores de modo a permitir comunicação entre eles.

Os endereços *IP* das interfaces *eth0* de ambos os computadores foram configurados utilizando o comando **ifconfig**. O computador tux61 ficou com endereço IP 127.16.60.1 e endereço MAC 00:0f:fe:8c:af:71 e o computador tux64 ficou com endereço IP 127.16.60.254 e endereço MAC 00:21:5a:c5:61:bb. Após a configuração, verificou-se, através do comando ***ping*** que os computadores estavam conectados. Não foi necessário configurar rotas, pois ambos os computadores estavam na mesma LAN. Na imagem abaixo encontram-se as tabelas de encaminhamento(onde se pode verificar, no tux61, que o *gateway* para a rede 172.16.60.0, que é a rede onde se encontram os computadores, é 0.0.0.0, ou seja, a rede está ligada directamente à interface) e de ARP (onde, no tux61 se pode verificar qual o endereço físico correspondente ao IP do computador tux64, e vice-versa).

Após limpar a entrada da tabela ARP(caso isto não fosse feito o o tux61 não enviaria um pacote ARP, pois teria o endereço MAC do tux64 na tabela ARP), utilizando o programa wireshark na interface eth0 do tux61 para capturar os pacotes enviados ao fazer o comando ping do tux61 para o tux64, verificou-se que o tux61 começou por enviar por broadcast um pacote ARP(bytes 12 e 13 do cabeçalho da trama ethernet iguais a 0x0806), com o objectivo de obter o endereço MAC associado ao IP da interface eth0 do tux64. Os endereços IP e MAC de origem deste pacote correspondem aos do tux61, o endereço MAC de destino é 00:00:00:00:00:00, pois o endereço do computador de destino é desconhecido, e o IP de destino corresponde ao tux64. A resposta foi um pacote ARP com endereços IP e MAC de origem e de destino correspondentes aos tux64 e tux61, respectivamente, a informar qual o endereço MAC do tux64.

Conhecendo agora o endereço MAC do tux64, os pacotes gerados pelo comando ***ping*** são pacotes IP(bytes 12 e 13 do cabeçalho da trama ethernet iguais a 0x0800) protocolo ICMP(byte 23 igual a 0x01), com endereços MAC e IP de origem correspondentes ao da máquina que envia o pacote e de destino correspondentes ao da máquina para o qual se envia o pacote.

A interface loopback é uma interface de rede virtual em que as mensagens enviadas para esta interface são recebidas pela máquina que as enviou. Esta interface pode ser útil para efeitos de teste, bem como para ser cliente de um servidor da própria máquina.

O tamanho de uma trama ethernet pode ser obtido através dos bytes números 12 e 13 do cabeçalho da trama. Segundo a norma *IEEE 802.3*, estes bytes contém o tamanho da payload da trama que tem um tamanho mínimo de 46 e máximo de 1500 bytes. Para valores superiores a 1500, estes bytes indicam qual é o protocolo que é encapsulado pela trama ethernet (usualmente IP).

3.2 Experiência 2 - Implementar duas VLAN's num switch

Na segunda experiência foi introduzido um novo computador (tux62), o qual foi configurado de uma forma semelhante à vista na experiência 1. A novidade nesta experiência é a criação (e configuração) de duas VLANs, numa das quais estão conectados os computadores tux61 e tux64 e na outra o computador tux62. O objectivo da experiência é o de estudar as ligações entre os vários computadores associados a VLANs.

A configuração de tux62 foi feita com recurso ao comando ifconfig tendo sido atribuído ao computador o endereço de IP 172.16.61.1/24 (escolhido por nós) e o endereço de MAC 00:21:5a:5a:7d:9c (característico da carta ethernet).

O processo de criação das VLANs foi efectuado através do computador tux61 que estava conectado com o switch da Cisco através de uma ligação na porta série. Para criar as duas VLANs foi utilizado o comando "vlan id", tendo sido criada uma VLAN com id=60 e uma VLAN com id=61. Daqui em diante, iremos referir-nos a estas VLANS como vlan60 e vlan61 respectivamente. Criadas as VLANs, foi necessário adicionar as portas do switch onde estavam conectados os três computadores às VLANs corretas. Estando o computador tux61 ligado à porta #1, tux62 ligado à porta #2 e tux64 ligado à porta #3, foi necessário adicionar a porta #1 e #3 à vlan60 e a porta #2 à vlan61. para o efeito foram utilizados os seguintes comandos:

```
1    ...
2    interface fastethernet 0/X
3    switchport mode access
4    switchport acces vlan 6Z
5    ...
```

Em que X representa a porta correspondente e Z o último digito da VLAN pretendida. Por exemplo, para adicionar tux61 à vlan60, X=1 e Z=0, visto tux61 estar ligado à porta #1 o switch e esta pertencer à vlan60.

Efectuando um ping a partir de tux61 para o IP de tux64 e tux62 foi possível ver que apenas o ping para tux64 (172.16.60.254) obteve resposta. De igual modo, efectuando um ping broadcast a partir de tux61 para o endereço 172.16.60.255 (comando ping -b 172.16.60.255) foi possível observar que apenas em tux64 se detectava a chegada de pacotes ICMP. Ao efectuar um ping broadcast a partir de tux62 para o endereço 172.16.61.255, não foi recebido nenhum pacote de ICMP nos restantes dois computadores.

Os resultados das experiências efectuadas vieram confirmar que as VLANs foram configuradas correctamente pois foi possível observar que embora os três computadores estivessem ligados ao mesmo switch, não existia uma conexão

entre os computadores tux61/tux64 e o computador tux62 dado a se encontrarem em VLANs distintas, existindo portanto dois domínios de broadcast. Com isto é possível perceber a utilidade do uso de VLANs permitindo a criação de sub-redes lógicas numa rede de área local física.

3.3 Experiência 3 - Configurar um router em Linux

A experiência 3 tinha como objectivo configurar a máquina Linux tux64 como um router entre as duas VLANs criadas na experiência anterior.

Para começar, foi preciso configurar a interface eth1 do tux64, adicionando-a à rede do tux62, tendo esta interface ficado com o endereço MAC 00:01:02:a1:35:69 e IP 172.16.61.253 (diferentes da interface eth0). Para além disto, e de modo a permitir comunicação entre as duas sub-redes e a que o tux64 funcione como router, foi ainda preciso activar IP forwarding e desactivar ICMP echo-ignore-broadcast, que estava activado por motivos de segurança.

Após configurar tux64, foi necessário adicionar uma rota em tux61 que lhe permitisse comunicar com a rede de tux62, utilizando o comando *route add -net 172.16.61.0/24 gw 172.16.60.254*. Esta rota indica ao tux61 que para enviar pacotes para endereços da gama 172.16.61.0/24 (onde se encontra o tux62), deve encaminhá-los para o gateway 172.16.60.254 (interface eth0 do tux64). Por outro lado, adicionou-se ao tux62 uma rota que lhe permitisse comunicar com a rede do tux61, com o comando route add -net 172.16.60.0/24 gw 172.16.61.253. Esta rota indica a tux62 que para enviar pacotes para endereços da gama 172.16.60.0/24 (onde se encontra o tux61), deve encaminhá-los para o gateway 172.16.61.253 (interface eth1 do tux64). Consultando as tabelas de encaminhamento dos três tuxes, observam-se os endereços para os quais cada mensagem deve ser encaminhada (gateway) de modo a chegar a um endereço de uma determinada rede de destino (sendo esta definida por um endereço e uma máscara) e qual a interface de rede a usar. Para além disso, são ainda mostradas flags que transmitem informação sobre a rota (por exemplo, U significa que a rota está válida, e G, como se pode ver na rota do tux62 para o tux64, que indica que a rota não é direta), uma métrica que indica o custo da rota em relação a outras para o mesmo destino(Metric), o número de utilizações ativas da rota(Ref) e o número de pacotes enviados através da rota(Use). As tabelas de encaminhamento podem ser vistas no anexo B.

Depois destas configurações, foi possível enviar um ping para o tux62 a partir do tux61, e vice-versa. Capturando pacotes no wireshark relativos tanto à interface eth0 e eth1 do tux64, verifica-se, em primeiro lugar, pedidos ARP nas duas interfaces. Na interface referente à rede onde se encontra o tux61, é recebido um pedido para saber qual o MAC associado ao IP da interface eth0 do tux64, que é o gateway para onde são reencaminhadas as mensagens dirigidas para a sub-rede onde se encontra o tux62. Na interface referente à rede onde se encontra o tux62, é enviado um pedido para saber o MAC associado ao IP do tux62. Estas mensagens são necessárias para resolver os endereços físicos necessários às mensagens ICMP enviadas pelo comando ping. Estas mensagens ICMP são observadas nas capturas de ambas interfaces, com IP de destino e origem sempre iguais ao IP do tux61 e tux62, respetivamente, mas endereços MAC de origem e destino iguais aos das máquinas entre as quais está a ser enviado o pacote. Ou seja, a mensagem enviada do tux61 para o tux62 é primeiramente reencaminhado para a interface do tux64 que se encontra na rede do tux61, com endereços MAC de origem e destino correspondentes aos do tux61 e tux64, respetivamente. O tux64 reencaminha através da interface eth1, que está na

sub-rede do tux62, a mensagem para o tux62, com endereços MAC de origem e destino correspondentes aos do tux64 e tux62, respetivamente.

3.4 Experiência 4 - Configurar um router comercial e implementar NAT

Na quarta experiência foi introduzido no sistema um ***router* comercial** da *Cisco* (Rc). Os objectivos da experiência são configurar **Rc**, implementar no mesmo a técnica de *Network Address Translation* (*NAT*) e acomodar os restantes computadores (tux61 , tux62, tux64) à existência desse *router*.

A introdução de **Rc** é um ponto de partida para conferir aos computadores do sistema acesso à *internet*. Para configurar o *router* comercial foi necessário definir os endereços IP de ambas as suas portas, uma porta que serve de interface para a "rede interna"(endereço 172.16.61.254/24) e uma porta que serve de interface para a "rede externa"(endereço 172.16.1.69/24) a qual está ligada a rede do laboratório (e à *internet*). Os comandos de configuração são introduzidos através da consola do *router* ligada a um computador através da porta série:

```
interface gigabitethernet 0/0
ip address 172.16.61.254 255.255.255.0
no shutdown
...
interface gigabitethernet 0/1
ip address 172.16.1.69 255.255.255.0
no shutdown
...
```

O segundo passo é o de adicionar aos computadores tux61, tux62 e tux64 as **rotas default**. O *router default* de tux61 é tux64, e Rc é o *router default* de tux62 e tux64:

```
// In tux61
route add default gw 172.16.60.254

// In tux62 and tux64
route add default gw 172.16.61.254
```

Para além disso foi necessário adicionar a Rc a rota para a rede 172.16.60.0. Isto pode ser feito adicionando uma **rota estática** para a rede na consola de Rc:

```
ip route 172.16.60.0 255.255.255.0 172.16.61.253
```

O comando usado define que para a gama de endereços definido pelo par **endereço+mascara** dados como primeiro e segundo argumento, o endereço do próximo "hop"deverá ser o endereço dado pelo terceiro argumento.

Através de vários comandos de *ping* foi possível verificar que todas as interfaces (de tux62, tux64 e ambas as interfaces de Rc) estavam atingíveis a partir

de tux61. Também foi possível verificar que a rede 172.16.60.0 estava atingível a partir de tux62. Assim confirmou-se que todas as rotas estavam bem definidas.

Ao retirar a rota para a rede 172.16.60.0 e desligando o **redireccionamento de ICMP** em tux62, efectuando um *traceroute* para tux61 verificou-se que os pacotes enviados seguiam primeiro para Rc (172.16.61.254) e de seguida para tux64. Isto acontece porque Rc é o *default gateway* de tux62 e Rc possui uma rota para a rede 172.16.60.0 através de tux64. Através dos *logs* foi possível observar a chegada de mensagens de *ICMP redirect* provenientes de Rc.

Activando o redireccionamento de ICMP em tux62 e efectuando *traceroute* novamente foi possível observar que, mesmo sem possuir a rota para a rede 172.16.60.0, tux62 conseguiu "**aprender**"que tux64 seria uma melhor rota para a rede 172.16.60.0.

Ao efectuar um *ping* de tux61 para o *router* do laboratório e analisando os *logs*, verificou-se que o *router* não estava atingível. Isto deve-se ao facto de, dado a Rc não ter a funcionalidade de *NAT* implementada, os pacotes *ping request* chegam ao *router* do laboratório com um endereço de origem da rede interna (neste caso 172.16.60.1) e, portanto, este *router* não tem conhecimento de para onde deve enviar os pacotes da resposta (não possui nenhuma rota para a "rede interna").

Para adicionar a funcionalidade de *Network Address Translation* ao *router* Cisco, são utilizados os seguintes comandos na consola de Rc:

```
1     // Definir a porta 0 do router com IP 172.16.61.254/24 e declará-la
2     // como a porta "interior" do NAT (porta que contacta com a rede interna)
3     interface gigabitethernet 0/0
4     ip address 172.16.61.254 255.255.255.0
5     no shutdown
6     ip nat inside
7     ...
8
9     // Definir a porta 1 do router com IP 172.16.1.69/24 e declará-la
10    // como a porta "exterior" do NAT (porta que contacta com a rede
      ↪  externa/internet)
11    interface gigabitethernet 0/1
12    ip address 172.16.1.69 255.255.255.0
13    no shutdown
14    ip nat outside
15    ...
16
17    // Definir uma "pool" de endereços globais com o nome de "ovrld" que podem
18    // ser alocados pelo NAT
19    ip nat pool ovrld 172.16.1.69 172.16.1.69 prefix 24
20    // Definir as gamas de endereços que têm permissão serem traduzidos pelo NAT.
21    // Neste caso endereços como 172.16.61.253 (tux64), estão fora desta gama
22    // e portanto não podem usufruir do NAT (não têm acesso à internet).
23    ip nat inside source list 1 pool ovrld overload
24    access-list 1 permit 172.16.60.0 0.0.0.7
25    access-list 1 permit 172.16.61.0 0.0.0.7
26    // Definir a default gateway do router e adicionar uma rota para a rede
      ↪  172.16.60.0
27    ip route 0.0.0.0 0.0.0.0 172.16.1.254
28    ip route 172.16.60.0 255.255.255.0 172.16.61.253
29    ...
```

A adição de *NAT* em Rc permite a **tradução dos endereços da rede interna** (rede constituída pelos computadores tux6X) no endereço exterior do router Cisco, o qual está mapeado no *router* do laboratório que eventualmente tem "ligação à *internet*". Deste modo é possível a **máquinas exteriores à rede receber e enviar pacotes** para estes computadores. Para qualquer pacote com endereço de origem na rede interna (salvo endereços excluídos pela configuração do *NAT*), Rc irá traduzir o endereço IP num par composto pelo seu "endereço global"(172.16.1.69) mais o número da porta alocada a esse endereço interno, enviando o pacote "**modificado**"para o seu destino. Inversamente, à chegada de um pacote com o IP de destino 172.16.1.69 e um número de uma porta, RC traduz este par num endereço da rede interna, enviando o pacote pela rota configurada.

3.5 Experiência 5 - DNS

De modo a conseguir aceder a domínios da Internet exteriores à rede criada até agora, pretendia-se, nesta experiência, configurar o DNS.

Para tal, foi necessário, em cada um dos três computadores, adicionar ao ficheiro /etc/resolv.conf as linhas ***search netlab.fe.up.pt*** e ***nameserver 172.16.1.1***. A primeira linha indica um domínio com o qual um pedido com menos do que 1 ponto é concatenado, por exemplo, neste caso, ao fazer *ping www*, o *ping* seria feito para *www.netlab.fe.up.pt*. A última indica qual o servidor a usar para traduzir domínios em endereços IP.

Observando os pacotes capturados pelo *wireshark*, verifica-se que são trocados pacotes com uma *query* e uma *response* DNS através do protocolo UDP. Na *query* é indicado o domínio, cujo IP é indicado na *response*.

3.6 Experiência 6 - Ligações TCP

A sexta (e última) experiência efectuada reúne toda a configuração feita nas experiências anteriores e submete a rede ao processo de descarregamento de um ficheiro do servidor de FTP (*File Transfer Protocol*) da Universidade do Porto. O objectivo da experiência é o de estudar o protocolo *TCP* e observar uma utilização mais "realista"da rede montada.

Após efectuar um pedido de *DNS* para averiguar qual é o endereço de IP correspondente ao servidor de *FTP* usado (ftp.up.pt), são abertas duas conexões *TCP* pela aplicação *FTP*. A conexão que é aberta em primeiro lugar tem como propósito **enviar os comandos *FTP* ao servidor** e receber as respostas a esses mesmos comandos (informação de controlo), enquanto que a segunda **recebe os dados enviados pelo servidor**. Uma conexão *TCP* tem **três fases**: o estabelecimento da conexão, a troca de dados/informação e o encerramento da conexão.

O **mecanismo *ARQ*** (*Automatic Repeat Request*) *TCP* é uma variação de *Go-Back-N* em que o receptor aquando da recepção de uma *frame* na qual detecta erros, não deixa de processar as seguintes. O receptor continua a receber as frames seguintes e envia no *ACK* o número de sequência da primeira *frame* que recebeu com erros. No final do envio o emissor verifica os *ACK* e reenvia as *frames* perdidas. Os logs encontram-se no anexo C.

O **mecanismo de controlo de congestionamento** utilizado pelo protocolo *TCP* utiliza várias técnicas como ***additive increase/multiplicative***

decrease, em conjunto com mecanismos como ***slow start*** e ***congestion window***. Para este controlo é definida uma nova variável associada à conexão denominada de "***Congestion Window***". Esta janela **limita a quantidade de tráfego existente de modo a assegurar a eficiência da conexão e um uso "justo"da rede**. Assim, quando o congestionamento da rede aumenta, o tamanho da janela diminui, inversamente, quando o congestionamento da rede diminui, o tamanho da janela aumenta. A fonte avalia a congestão da rede através dos timeouts/perda dos pacotes enviados (quantos mais timeouts, maior a congestão). Inicialmente, a janela de congestionamento tem um tamanho base. Utilizando um mecanismo de "***Slow Start***"o tamanho da janela é duplicado por cada **Round Trip Time(RTT)**, sendo que estes tempos são contados através da recepção de *ACKS*. Quando se dá a perda de um pacote, o tamanho da janela volta ao inicial e é definido um "*threshold*"com um valor igual a **metade** do valor da janela aquando da perda. A partir daí a transmissão é feita novamente utilizando "***Slow Start***"até o tamanho da janela atingir o threshold, momento em que a conexão entra na fase de "congestion avoidance". Nesta fase, a janela de congestão aumenta em **um segmento** por cada RTT (*additive increase*) e diminui para **metade** em caso de perda de um pacote (*multiplicative decrease*).

Nos *logs* foi possível observar que o ***throughput*** de informação **aumenta** rapidamente no início da transmissão e *estabiliza* à volta dos 90 Mbit/s. Alguns segundos depois onde se dá uma queda acentuada do *throughput* para volta dos 55 Mbit/s, de onde a transmissão volta a subir para os 90Mbit/s. Para além disso, foi possível observar que ao longo da transmissão forem detectados valores do tamanho do segmento a oscilar entre valores que rondam dos 1300, 2700 e 4400 bytes. Os valores observados vão de encontro aos mecanismos estudados que promovem um crescimento rápido da quantidade de informação enviada (por factores de 2) de modo a atingir a **eficiência**. O gráficos capturado no *Wireshark* está disponível no anexo D.

Ao efectuar uma transmissão FTP no computador tux62 durante a transmissão no computador tux61, foi possível observar que quando a transmissão em tux62 inicia, o *throughput* da ligação de tux61 **diminui acentuadamente**. Eventualmente o *throughput* de ambas estabiliza por volta dos 50 Mbit/s. Quando a transmissão em tux61 termina, o valor do throughput em tux62 torna a subir até cerca de 90Mbit/s. Este resultado vem confirmar o **uso "justo"** da rede que é pretendido com o mecanismo de congestionamento do protocolo *TCP*. Os gráficos capturados no *Wireshark* estão disponíveis no anexo E.

4 Conclusões

Através das experiências efectuadas foi possível adquirir as bases necessárias para construir uma rede de computadores que, embora simples, seja funcional e entender o seu funcionamento. Para além disso, a primeira parte do projecto deu-nos a conhecer o processo de desenvolvimento de aplicações que funcionem de acordo com protocolos pré-estabelecidos (neste caso o protocolo FTP). Em conjunto com o primeiro projecto, o trabalho efectuado serve como uma boa introdução aos conceitos das várias camadas que compõem o modelo de uma rede de computadores: física, ligação de dados, rede, transporte e aplicação.

5 Anexos

A Download bem sucedido

```
[bernas@bernas-pc ftp_app]$ ./ftp ftp://ftp.up.pt/pub/CPAN/ENDINGS
 - Username : anonymous
 - Password : anonymous
 - Host : ftp.up.pt
 - URL Path : pub/CPAN/ENDINGS
 - IP Address : 193.137.29.15

220-Welcome to the University of Porto's mirror archive (mirrors.up.pt)
220----------------------------------------------------------
220-
220-All connections and transfers are logged. The max number of connections is 200.
220-
220-For more information please visit our website: http://mirrors.up.pt/
220-Questions and comments can be sent to mirrors@uporto.pt
220-
220-
220
 > Connection established

 > Sending Username
331 Please specify the password.
 > Sending Password

230 Login successful.
227 Entering Passive Mode (193,137,29,15,215,185).
150 Opening BINARY mode data connection for pub/CPAN/ENDINGS (3954 bytes).
ENDINGS : 100% [#########################################################################################################]
226 Transfer complete.
221 Goodbye.
```

Figura 1: Download bem sucedido

B Tabelas de encaminhamento da experiência 3

```
tux61:~# route -n
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
172.16.60.0     0.0.0.0         255.255.255.0   U     0      0        0 eth0
172.16.61.0     172.16.60.254   255.255.255.0   UG    0      0        0 eth0
tux61:~#
```

Figura 2: Tabela de encaminhamento de tux61

```
root@tux62:~# route -n
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
172.16.60.0     172.16.61.253   255.255.255.0   UG    0      0        0 eth0
172.16.61.0     0.0.0.0         255.255.255.0   U     0      0        0 eth0
root@tux62:~#
```

Figura 3: Tabela de encaminhamento de tux62

```
tux64:~# route -n
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
172.16.60.0     0.0.0.0         255.255.255.0   U     0      0        0 eth0
172.16.61.0     0.0.0.0         255.255.255.0   U     0      0        0 eth1
tux64:~#
```

Figura 4: Tabela de encaminhamento de tux61

C Mecanismo de ARQ numa conexão TCP

```
1528 2.910849905  193.137.29.15  172.16.60.1    FTP-DA…  2802 FTP Data: 2736 bytes (PASV) (retr )
1529 2.910875119  172.16.60.1    193.137.29.15  TCP        66 40574 → 58015 [ACK] Seq=1 Ack=2183329 Win=340864 Len=0 TSval=1016453 TSecr=760892637
1530 2.911098166  193.137.29.15  172.16.60.1    FTP-DA…  1434 FTP Data: 1368 bytes (PASV) (retr )
1531 2.911108562  193.137.29.15  172.16.60.1    FTP-DA…  1434 [TCP Previous segment not captured] FTP Data: 1368 bytes (PASV) (retr )
1532 2.911131600  172.16.60.1    193.137.29.15  TCP        78 40574 → 58015 [ACK] Seq=1 Ack=2184697 Win=340736 Len=0 TSval=1016453 TSecr=760892637 SLE=2192905 SRE=2194273
1533 2.911349504  193.137.29.15  172.16.60.1    FTP-DA…  2802 [TCP Previous segment not captured] FTP Data: 2736 bytes (PASV) (retr )
1534 2.911376452  172.16.60.1    193.137.29.15  TCP        86 [TCP Dup ACK 1532#1] 40574 → 58015 [ACK] Seq=1 Ack=2184697 Win=340736 Len=0 TSval=1016453 TSecr=760892637 SLE=2195641 SRE=2198377 SLE=2192905 SRE=2194273
1535 2.911600185  193.137.29.15  172.16.60.1    FTP-DA…  2802 FTP Data: 2736 bytes (PASV) (retr )
1536 2.911624752  172.16.60.1    193.137.29.15  TCP        86 [TCP Dup ACK 1532#2] 40574 → 58015 [ACK] Seq=1 Ack=2184697 Win=340736 Len=0 TSval=1016453 TSecr=760892637 SLE=2195641 SRE=2201113 SLE=2192905 SRE=2194273
1537 2.911852310  193.137.29.15  172.16.60.1    FTP-DA…  4170 FTP Data: 4104 bytes (PASV) (retr )
1538 2.911876331  172.16.60.1    193.137.29.15  TCP        86 [TCP Dup ACK 1532#3] 40574 → 58015 [ACK] Seq=1 Ack=2184697 Win=340736 Len=0 TSval=1016453 TSecr=760892637 SLE=2195641 SRE=2205217 SLE=2192905 SRE=2194273
1539 2.912097889  193.137.29.15  172.16.60.1    FTP-DA…  1434 FTP Data: 1368 bytes (PASV) (retr )
1540 2.912108015  193.137.29.15  172.16.60.1    FTP-DA…  1434 [TCP Previous segment not captured] FTP Data: 1368 bytes (PASV) (retr )
```

Figura 5: Mecanismo de ARQ numa conexão TCP

D Gráfico de throughput da experiência 6.4

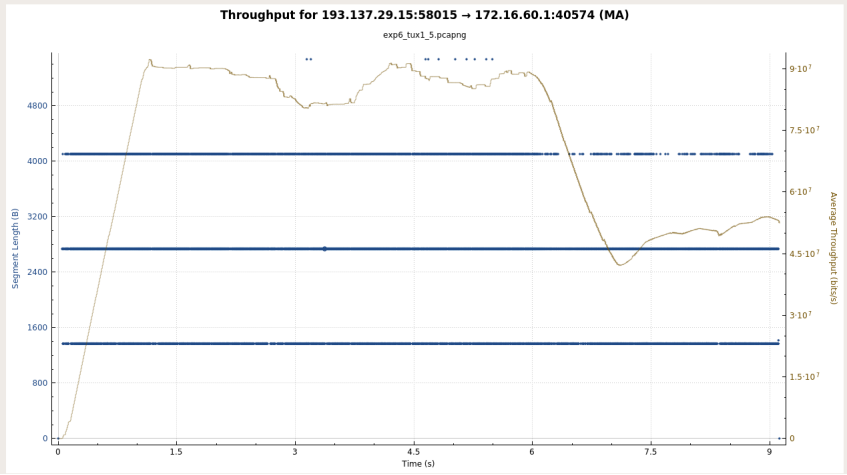

Figura 6: Throughput (laranja) e Segment Size (azul) de tux61

E Gráficos de throughput da experiência 6.5

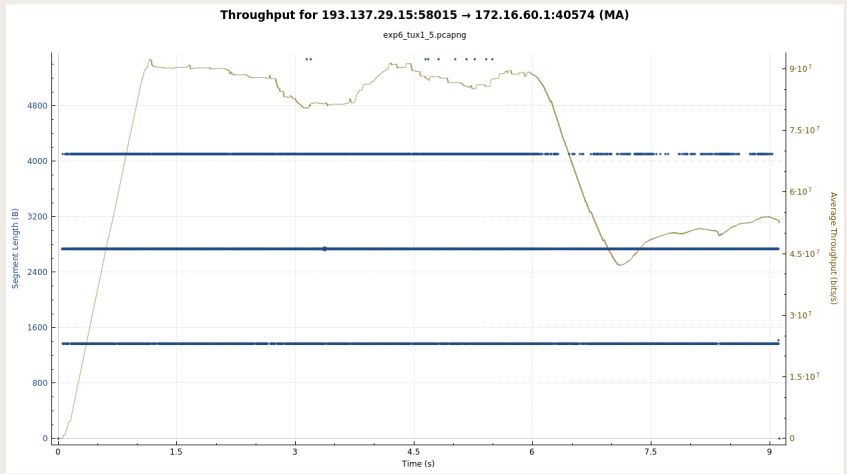

Figura 7: Throughput (laranja) e Segment Size (azul) de tux61

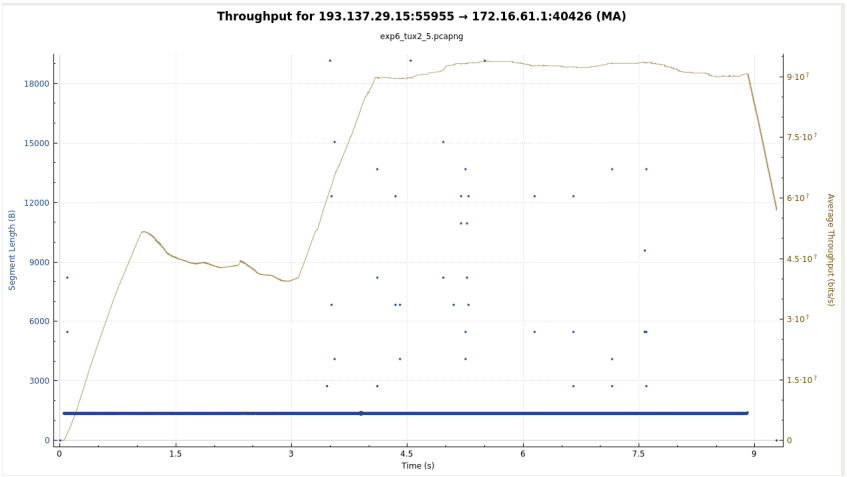

Figura 8: Throughput (laranja) e Segment Size (azul) de tux62