

Faculdade de Engenharia da Universidade do Porto



Rede de Computadores

Grupo 6

Segundo trabalho laboratorial

Estudantes:

- Alexandre Correia (up202007042@edu.fe.up.pt)
- Gonalo Pinto (up202004907@edu.fe.up.pt)

Índice

Índice	2
Sumário	3
Introdução	3
Parte 1 - Aplicação Download	4
Parte 2 - Configuração da Rede	4
Experiência 1 - Configurar uma rede IP	4
Experiência 2 - Implementar duas Bridges num Switch	5
Experiência 3 - Configurar um Router em Linux	6
Experiência 4 - Configurar um Commercial Router e Implementar a NAT	8
Experiência 5 - DNS	9
Experiência 6 - Conexões TCP	9
Conclusões	11
Anexo I - Código Fonte	11

Sumário

Neste projeto, foi criada uma rede de computadores e uma aplicação que permite transferir ficheiros, com o objetivo de perceber com maior detalhe a implementação e configuração de equipamentos como computadores, *switches* e *routers*, bem como protocolos ao nível da transferência de dados.

Com este projeto, foi possível conceber uma aplicação capaz de transferir qualquer ficheiro de um dado servidor, a partir de um computador ligado a uma rede devidamente configurada.

Introdução

A realização deste projeto tem dois objetivos: primeiramente, a criação de uma aplicação que permitisse transferir um ficheiro localizado num servidor FTP ; seguidamente da configuração da rede definida no enunciado no trabalho, que é utilizada na execução da aplicação anterior.

Este relatório tem como objetivo fornecer informações mais detalhadas sobre a configuração da rede e da aplicação que permite estabelecer uma conexão a um servidor FTP.

Na primeira parte, é apresentado o funcionamento da aplicação desenvolvida que após receber como entrada uma *socket*, é feita a autenticação e conexão ao servidor FTP, permitindo assim a transferência do ficheiro indicado na *socket*.

Na segunda parte, a configuração da rede é explicada passando por todas as seis experiências realizadas até a implementação completa da rede pretendida. Durante estas experiências são realizados alguns testes à rede com a aplicação *Wireshark*, de modo a perceber se corresponde ao objetivo pretendido.

Parte 1 - Aplicação *Download*

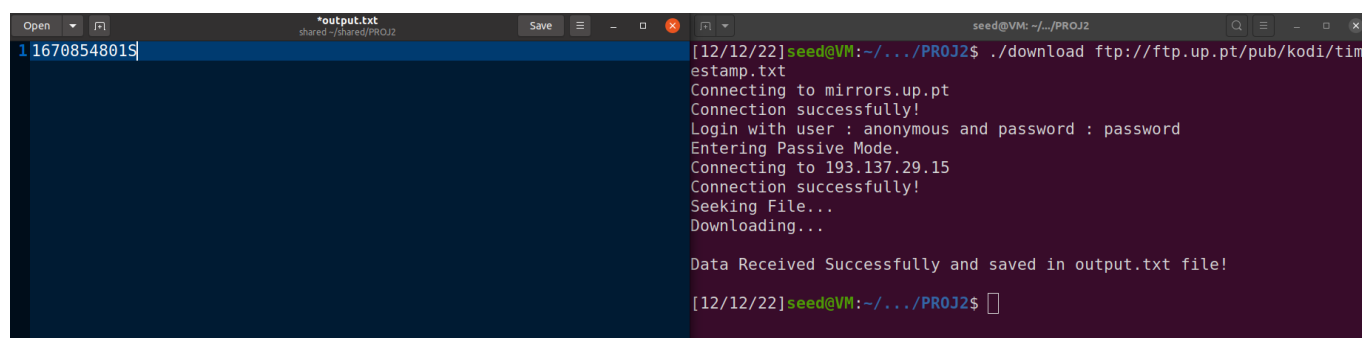
A aplicação *download* tem como objetivo transferir um ficheiro de um servidor FTP, dado o caminho para o mesmo.

Para que a transferência tenha sucesso a aplicação conecta-se ao *host* fornecido (através de uma *socket*) e faz a devida autenticação, caso esta exista.

Depois desta comunicação inicial, a aplicação entrará em *passive mode*, para poder receber informação.

Após o processo do IP enviado pelo servidor FTP, a aplicação abre outra *socket* para receção de dados (IP aleatório).

Finalmente, a aplicação introduz o caminho para o ficheiro e este é transferido para o computador, posteriormente guardado num ficheiro no computador local (**output.txt**) e apresentado no terminal uma mensagem de sucesso.



```
Open  *output.txt  Save  seed@VM: ~/.../PROJ2
1 1670854801$ [12/12/22]seed@VM:~/.../PROJ2$ ./download ftp://ftp.up.pt/pub/kodi/tim
estamp.txt
Connecting to mirrors.up.pt
Connection successfully!
Login with user : anonymous and password : password
Entering Passive Mode.
Connecting to 193.137.29.15
Connection successfully!
Seeking File...
Downloading...

Data Received Successfully and saved in output.txt file!

[12/12/22]seed@VM:~/.../PROJ2$
```

Figura 1 - Exemplo de execução do programa (download do ficheiro **pub/kodi/timespamp.txt**, do servidor **ftp.up.pt**)

Parte 2 - Configuração da Rede

É de notar que todas as experiências foram realizadas na **bancada 5**.

Experiência 1 - Configurar uma rede IP

Com o uso do comando **ipconfig** foi possível configurar o **tux3** e o **tux4** com os IPs **172.16.50.1** e **172.16.50.254**, respetivamente. Estes conseguem comunicar entre si sem a necessidade de criar *routes*.

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	Routerbo_1c:95:c8	Spanning-tree-(for...	STP	60	RST. Root = 32768/0/c4:ad:34:1c:95:c8 Cost = 0 Port = 0x8001
2	2.002181004	Routerbo_1c:95:c8	Spanning-tree-(for...	STP	60	RST. Root = 32768/0/c4:ad:34:1c:95:c8 Cost = 0 Port = 0x8001
3	4.004360820	Routerbo_1c:95:c8	Spanning-tree-(for...	STP	60	RST. Root = 32768/0/c4:ad:34:1c:95:c8 Cost = 0 Port = 0x8001
4	6.006547132	Routerbo_1c:95:c8	Spanning-tree-(for...	STP	60	RST. Root = 32768/0/c4:ad:34:1c:95:c8 Cost = 0 Port = 0x8001
5	7.597018727	HewlettP_61:2c:54	Broadcast	ARP	42	Who has 172.16.50.254? Tell 172.16.50.1
6	7.597135991	HewlettP_19:09:5c	HewlettP_61:2c:54	ARP	60	172.16.50.254 is at 00:22:64:19:09:5c
7	7.597144511	172.16.50.1	172.16.50.254	ICMP	98	Echo (ping) request id=0x0c7a, seq=1/256, ttl=64 (reply in 8)
8	7.597231534	172.16.50.254	172.16.50.1	ICMP	98	Echo (ping) reply id=0x0c7a, seq=1/256, ttl=64 (request in 7)
9	8.008733094	Routerbo_1c:95:c8	Spanning-tree-(for...	STP	60	RST. Root = 32768/0/c4:ad:34:1c:95:c8 Cost = 0 Port = 0x8001
10	8.604513815	172.16.50.1	172.16.50.254	ICMP	98	Echo (ping) request id=0x0c7a, seq=2/512, ttl=64 (reply in 11)
11	8.604616691	172.16.50.254	172.16.50.1	ICMP	98	Echo (ping) reply id=0x0c7a, seq=2/512, ttl=64 (request in 10)
12	9.628510861	172.16.50.1	172.16.50.254	ICMP	98	Echo (ping) request id=0x0c7a, seq=3/768, ttl=64 (reply in 13)
13	9.628641254	172.16.50.254	172.16.50.1	ICMP	98	Echo (ping) reply id=0x0c7a, seq=3/768, ttl=64 (request in 12)

Figura 2 - Experiência 1

Com o auxílio do programa *Wireshark*, consegue-se observar os pacotes transferidos com o uso do comando *ping* (ver figura 2).

Num primeiro momento, pode-se observar pacotes ARP, usados para obter o endereço MAC de um computador, sabendo o endereço IP. A origem pede o MAC de um computador com um certo IP. Esse computador envia uma resposta com o seu MAC.

Os pacotes *ping* podem ser do tipo **ping request** ou do tipo **ping reply**. A origem envia o **ping request** com o seu IP e MAC para o destinatário, que tem um IP e um MAC associado também.

No destinatário é enviado um **ping reply** com a mesma estrutura só que com os IPs e MACs da origem no destino e vice-versa.

O *Wireshark* permite determinar várias informações sobre os pacotes que estão a ser capturados, por exemplo, o tipo de protocolo usado pelo pacote (coluna **Protocol**) e o tamanho (coluna **Length**).

Finalmente, o *loopback interface* é uma interface virtual que está sempre acessível enquanto um dos endereços IP no *switch* for operacional. é bastante importante principalmente para fazer *debug* e testar aplicações.

Experiência 2 - Implementar duas *Bridges* num *Switch*

Nesta experiência foi possível implementar duas *bridges* e ligar o **tux3** e **tux4** a uma e o **tux2** a outra. Verifica-se que os computadores de *bridges* diferentes não conseguem comunicar entre si, pelo que não foram capturados *pings* entre o **tux2** e qualquer outro computador. No entanto, o **tux3** e **tux4** conseguem comunicar normalmente. (ver figura 3)

1	0.000000000	Routerbo_1c:95:c8	Spanning-tree-(for...	STP	60	RST. Root = 32768/0/c4:ad:34:1c:95:c8 Cost = 0 Port = 0x8001
2	0.658247533	0.0.0.0	255.255.255.255	MNDP	159	5678 → 5678 Len=117
3	0.658279101	Routerbo_1c:95:cf	CDP/VTP/DTP/PAGP/UD...	CDP	93	Device ID: Mikrotik Port ID: bridge50
4	0.899498323	172.16.50.1	172.16.50.254	ICMP	98	Echo (ping) request id=0x07df, seq=1/256, ttl=64 (reply in 5)
5	0.899611536	172.16.50.254	172.16.50.1	ICMP	98	Echo (ping) reply id=0x07df, seq=1/256, ttl=64 (request in 4)
6	1.910899564	172.16.50.1	172.16.50.254	ICMP	98	Echo (ping) request id=0x07df, seq=2/512, ttl=64 (reply in 7)
7	1.910995875	172.16.50.254	172.16.50.1	ICMP	98	Echo (ping) reply id=0x07df, seq=2/512, ttl=64 (request in 6)

Figura 3 - Experiência 2

Para configurar uma *bridge*, já no **Mikrotik switch**, com o *reset* feito, primeiro cria-se uma *bridge* (**/interface bridge add name=bridge50**). Depois remove-se as portas da *bridge default* (**/interface bridge port remove [find interface=ether1]**) e adiciona-se essas mesmas portas à *bridge* que foi criada (**/interface bridge port add bridge=bridge50 interface=ether1**).

172.16.50.1	172.16.50.255	ICMP
172.16.50.1	172.16.50.255	ICMP

Figura 4 - Experiência 2 (Broadcast no tux3)

172.16.51.1	172.16.51.255	ICMP
172.16.51.1	172.16.51.255	ICMP

Figura 5 - Experiência 2 (Broadcast no tux2)

Pela análise dos pacotes, conclui-se que há 2 domínios *broadcast*, o **172.16.50.0/24** e o **172.16.51.0/24**

Experiência 3 - Configurar um *Router* em *Linux*

Nesta parte, liga-se o **tux4**, noutra interface, à *bridge* do **tux2**, permitindo, assim, a comunicação entre os computadores das duas *bridges*. Foi necessário fazer *routing*, permitir o **IP Forwarding** e desabilitar o **ICMP echo ignore broadcast**.

9	3.065290635	172.16.50.1	172.16.50.254	ICMP	98 Echo (ping) request	id=0x0de2, seq=4/1024, ttl=64 (reply in 10)
10	3.065392394	172.16.50.254	172.16.50.1	ICMP	98 Echo (ping) reply	id=0x0de2, seq=4/1024, ttl=64 (request in 9)
11	4.089282024	172.16.50.1	172.16.50.254	ICMP	98 Echo (ping) request	id=0x0de2, seq=5/1280, ttl=64 (reply in 12)
12	4.089381757	172.16.50.254	172.16.50.1	ICMP	98 Echo (ping) reply	id=0x0de2, seq=5/1280, ttl=64 (request in 11)
13	4.221396171	Routerbo_1c:95:cf	Spanning-tree-(for-...	STP	60 RST. Root = 32768/0/c4:ad:34:1c:95:cf	Cost = 0 Port = 0x8001
14	5.081254188	HewlettP_61:2c:54	HewlettP_19:09:5c	ARP	42 Who has 172.16.50.254? Tell 172.16.50.1	
15	5.081354410	HewlettP_19:09:5c	HewlettP_61:2c:54	ARP	60 172.16.50.254 is at 00:22:64:19:09:5c	
16	5.085890112	HewlettP_19:09:5c	HewlettP_61:2c:54	ARP	60 Who has 172.16.50.1? Tell 172.16.50.254	
17	5.085904081	HewlettP_61:2c:54	HewlettP_19:09:5c	ARP	42 172.16.50.1 is at 00:21:5a:61:2c:54	
18	6.223642546	Routerbo_1c:95:cf	Spanning-tree-(for-...	STP	60 RST. Root = 32768/0/c4:ad:34:1c:95:cf	Cost = 0 Port = 0x8001
19	6.928623885	0.0.0.0	255.255.255.255	MNDP	159 5678 + 5678 Len=117	
20	6.928654754	Routerbo_1c:95:cf	CDP/VTP/DTP/PAGP/UD...	CDP	93 Device ID: MikroTik	Port ID: bridge50
21	6.991926907	172.16.50.1	172.16.51.253	ICMP	98 Echo (ping) request	id=0x0de6, seq=1/256, ttl=64 (reply in 22)
22	6.992029713	172.16.51.253	172.16.50.1	ICMP	98 Echo (ping) reply	id=0x0de6, seq=1/256, ttl=64 (request in 21)
23	7.993287904	172.16.50.1	172.16.51.253	ICMP	98 Echo (ping) request	id=0x0de6, seq=2/512, ttl=64 (reply in 24)
24	7.993388405	172.16.51.253	172.16.50.1	ICMP	98 Echo (ping) reply	id=0x0de6, seq=2/512, ttl=64 (request in 23)

Figura 6 - Experiência 3 (tux3 a comunicar com as duas interfaces do tux4)

35	15.416015263	172.16.50.1	172.16.51.1	ICMP	98 Echo (ping) request	id=0x0ded, seq=1/256, ttl=64 (reply in 36)
36	15.416271301	172.16.51.1	172.16.50.1	ICMP	98 Echo (ping) reply	id=0x0ded, seq=1/256, ttl=63 (request in 35)
37	16.234838034	Routerbo_1c:95:cf	Spanning-tree-(for-...	STP	60 RST. Root = 32768/0/c4:ad:34:1c:95:cf	Cost = 0 Port = 0x8001
38	16.441292640	172.16.50.1	172.16.51.1	ICMP	98 Echo (ping) request	id=0x0ded, seq=2/512, ttl=64 (reply in 39)
39	16.441531148	172.16.51.1	172.16.50.1	ICMP	98 Echo (ping) reply	id=0x0ded, seq=2/512, ttl=63 (request in 38)
40	17.465286892	172.16.50.1	172.16.51.1	ICMP	98 Echo (ping) request	id=0x0ded, seq=3/768, ttl=64 (reply in 41)
41	17.465511851	172.16.51.1	172.16.50.1	ICMP	98 Echo (ping) reply	id=0x0ded, seq=3/768, ttl=63 (request in 40)
42	19.327094700	Routerbo_1c:95:cf	Spanning-tree-(for-...	STP	60 RST. Root = 32768/0/c4:ad:34:1c:95:cf	Cost = 0 Port = 0x8001

Figura 7 - Experiência 3 (tux3 a comunicar com tux2)

De modo a configurar os caminhos, deve-se definir regras:

- No **tux3**, se o endereço destino for **172.16.51.0/24**, deve ser encaminhado pelo endereço **172.16.50.254 (eth0 do tux4)**.
- No **tux2**, se o endereço destino for **172.16.50.0/24**, deve ser encaminhado pelo endereço **172.16.51.253 (eth1 do tux4)**.

Kernel IP routing table						
Destination	Gateway	Genmask	Flags	Metric	Ref	Use Iface

Figura 8 - Cabeçalho de uma *routing table*

Analisando uma *routing table*, pode-se observar o endereço destino (com a devida máscara), o *gateway* (porta para onde será encaminhado), uma interface e algumas *flags*.

Foi possível verificar a conectividade entre todas as máquinas (**172.16.50.254**, **172.16.51.253** e **172.16.51.1**) e só se é observado mensagens ARP entre o **tux3 (00:21:5a:61:2c:54)** e o **tux4 (00:22:64:19:09:5c)** no ping para **172.16.50.254**, porque estão na mesma rede.

Ethernet II, Src: HewlettP_61:2c:54 (00:21:5a:61:2c:54), Dst: HewlettP_19:09:5c (00:22:64:19:09:5c)
Internet Protocol Version 4, Src: 172.16.50.1, Dst: 172.16.50.254
Ethernet II, Src: HewlettP_61:2c:54 (00:21:5a:61:2c:54), Dst: HewlettP_19:09:5c (00:22:64:19:09:5c)
Internet Protocol Version 4, Src: 172.16.50.1, Dst: 172.16.51.253
Ethernet II, Src: HewlettP_61:2c:54 (00:21:5a:61:2c:54), Dst: HewlettP_19:09:5c (00:22:64:19:09:5c)
Internet Protocol Version 4, Src: 172.16.50.1, Dst: 172.16.51.1

Figura 9 - MACs associados a cada *ping*

Observando as *logs*, quando o **tux3** manda pacotes ICMP (tanto para **tux4** como para **tux2**), o MAC associado ao IP destino é o do **tux4** e o MAC associado ao IP Origem é o do **tux3**.

Experiência 4 - Configurar um *Commercial Router* e Implementar a NAT

Nesta experiência foi configurado um *router*, ligado à *bridge* 51, de modo a conseguir aceder à *Internet* a partir de qualquer computador.

Com o uso do comando *ping* foi verificado que todos os computadores estão ligados.

Para definir uma *static route* deve-se usar o comando:

→ `/ip route add dst-address=<ip_address>/<mask>
gateway=<ip_address>`



Figura 10 - Traceroute (Experiência 4)

Como é possível observar na figura 10, quando é executado o comando *ping* do **tux2** para o **tux3** com e sem *route* verifica-se que:

(sem a route) 172.16.51.1 -> 172.16.51.254 -> 172.16.51.253 -> 172.16.50.1

(com a route) 172.16.51.1 -> 172.16.51.253 -> 172.16.50.1

Isto acontece devido ao facto de **172.15.51.254** ser o *default gateway* do **tux2**. Adicionando a *route* diretamente otimiza a comunicação.

O *Network Address Translation* (NAT) traduz o endereço privado de cada equipamento no endereço público da rede, e quando recebe informação faz o inverso (transforma o endereço público num endereço privado). Desta forma o exterior só conhece o IP público da rede.

Para configurar o NAT, primeiramente deve-se desativar o NAT *default* (`/ip firewall nat disable 0`)

Depois, adiciona-se as regras (`/ip firewall nat add chain=srcnat action=masquerade out-interface=ether1`).

Experiência 5 - DNS

Nesta fase do projeto procedeu-se à configuração do DNS nos vários computadores de forma a facilitar o acesso aos servidores da **FEUP**.

10 0.8887625904	172.16.51.253	172.16.2.1	DNS	/1 Standard query 0xb09c AAAA ftp.up.pt
11 0.888320776	172.16.2.1	172.16.51.253	DNS	109 Standard query response 0x0d94 A ftp.up.pt CNAME mirrors.up.pt A 193.137.29.15
12 0.888353532	172.16.2.1	172.16.51.253	DNS	121 Standard query response 0x0e9c AAAA ftp.up.pt CNAME mirrors.up.pt AAAA 2001:690:2200:1200::15
13 0.925518393	172.16.51.253	193.137.29.15	ICMP	100 Echo (ping) request id=0x0c12, seq=1/256, ttl=64 (reply in 14)
14 0.929155392	193.137.29.15	172.16.51.253	ICMP	100 Echo (ping) reply id=0x0c12, seq=1/256, ttl=58 (request in 13)
15 0.929277615	172.16.51.253	172.16.2.1	DNS	88 Standard query 0x20db PTR 15.29.137.193.in-addr.arpa
16 0.929957383	172.16.2.1	172.16.51.253	DNS	115 Standard query response 0x20db PTR 15.29.137.193.in-addr.arpa PTR mirrors.up.pt
17 1.117565231	Routerbo_eb:24:1d		ARP	62 Who has 172.16.51.253? Tell 172.16.51.254
18 1.117586113	KYE_25:21:9e		ARP	44 172.16.51.253 is at 00:c0:df:25:21:9e
19 1.927069524	172.16.51.253	193.137.29.15	ICMP	100 Echo (ping) request id=0x0c12, seq=2/512, ttl=64 (reply in 20)

Figura 11 - Experiência 5

Pacotes DNS são usados para traduzir um domínio (www.google.com, por exemplo) para um endereço IP. Na imagem pode-se observar o pedido e a sua devida resposta.

Para configurar o DNS, no ficheiro **/etc/resolv.conf**, é usado o comando **nameserver xxx.xxx.xxx.xxx**, de modo a relacionar um nome com um endereço IP.

Experiência 6 - Conexões TCP

Nesta experiência foi ainda realizada a transferência de um ficheiro através da aplicação desenvolvida na **parte 1**. Foi feita a recolha dos pacotes durante apenas **uma transferência e duas transferências em simultâneo**.

5 6.928476671	172.16.50.1	172.16.2.1	DNS	71 Standard query 0x6f2e A ftp.gnu.org
6 6.929347530	172.16.2.1	172.16.50.1	DNS	87 Standard query response 0x6f2e A ftp.gnu.org A 209.51.188.20
7 6.929492872	172.16.50.1	209.51.188.20	TCP	74 45346 → 21 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM TSval=3922898270 TSecr=0 WS=64
8 7.045335839	209.51.188.20	172.16.50.1	TCP	74 21 → 45346 [SYN, ACK] Seq=0 Ack=1 Win=65160 Len=0 MSS=1460 SACK_PERM TSval=1680082279 TSecr=3922898270 WS=128
9 7.045377255	172.16.50.1	209.51.188.20	TCP	66 45346 → 21 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=3922898386 TSecr=1680082279
10 7.045460856	172.16.50.1	209.51.188.20	FTP	81 Request: user anonymous
11 7.160660509	209.51.188.20	172.16.50.1	TCP	66 21 → 45346 [ACK] Seq=1 Ack=16 Win=65152 Len=0 TSval=1680082394 TSecr=3922898386
12 7.207036878	209.51.188.20	172.16.50.1	FTP	93 Response: 220 GNU FTP server ready.
13 7.207050218	172.16.50.1	209.51.188.20	TCP	66 45346 → 21 [ACK] Seq=16 Ack=28 Win=64256 Len=0 TSval=3922898548 TSecr=1680082440
14 7.300878173	209.51.188.20	172.16.50.1	FTP	105 Response: 230-NOTICE (Updated October 15 2021):
15 7.300886554	172.16.50.1	209.51.188.20	TCP	66 45346 → 21 [ACK] Seq=16 Ack=67 Win=64256 Len=0 TSval=3922898642 TSecr=1680082533
16 7.300908484	209.51.188.20	172.16.50.1	FTP	72 Response: 230-

Figura 12 - Experiência 6 (início da transferência)

FTP	67 Request:
TCP	66 42112 → 28360 [FIN, ACK] Seq=1 Ack=31242091 Win=1738240 Len=0 TSval=3487044961 TSecr=1853221159
FTP	90 Response: 226 Transfer complete.

Figura 13 - Experiência 6 (fim da transferência)

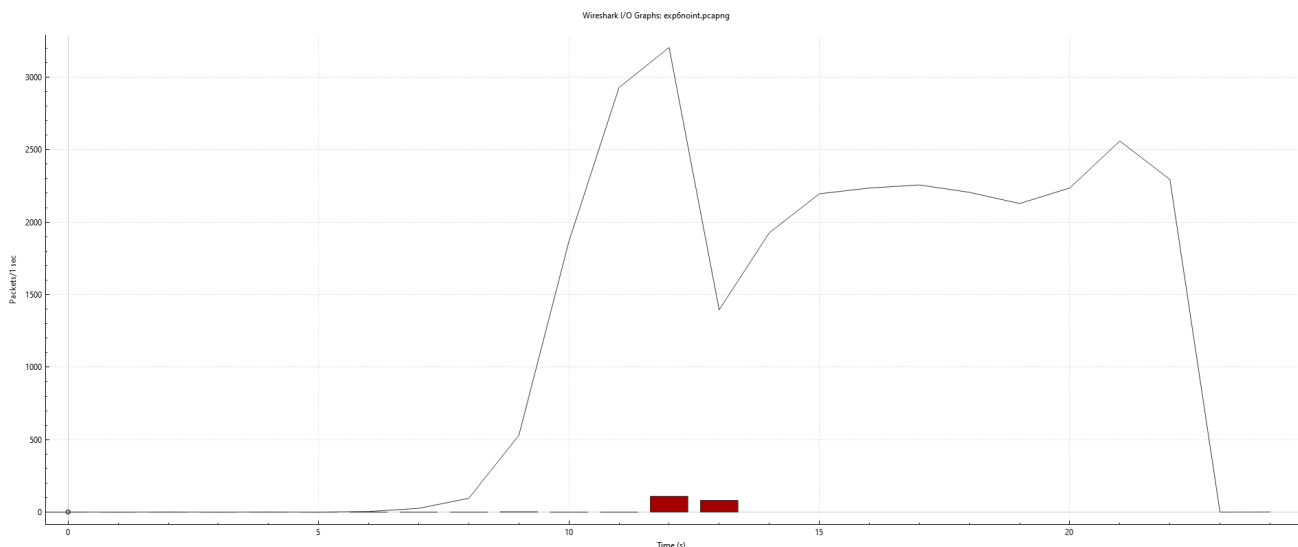


Figura 14 - Experiência 6 (transferência única)

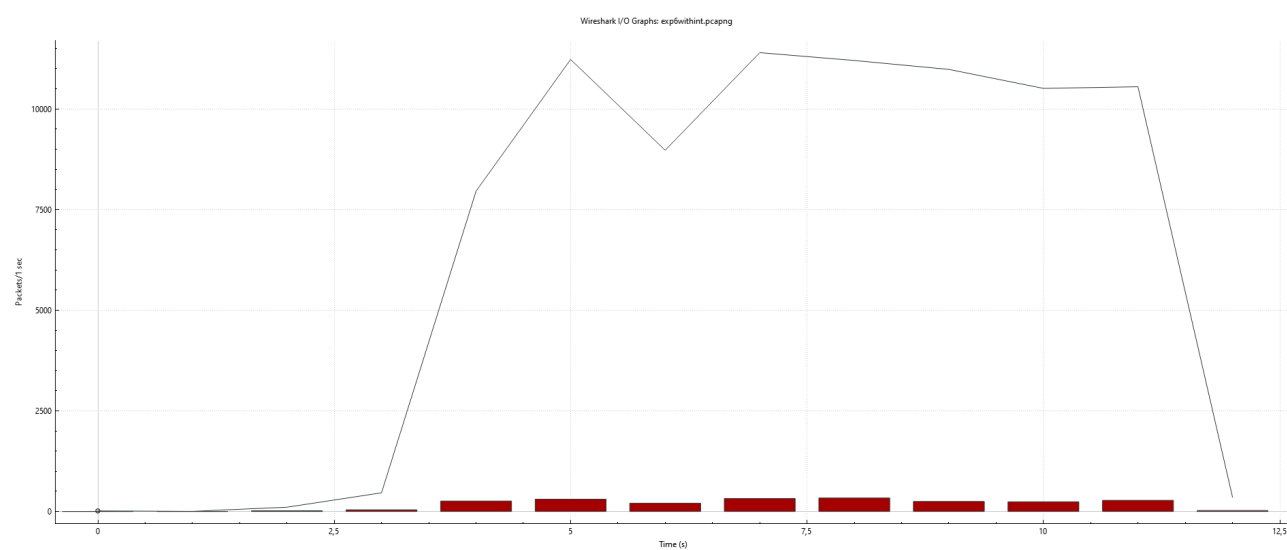


Figura 15 - Experiência 6 (duas transferências em simultâneo)

Uma conexão TCP está dividida em 3 fases, o estabelecimento da conexão (**figura 12**), a transferência de dados (**figura 12**) e o término da conexão (**figura 13**).

Na aplicação **download** são abertas duas conexões, uma para os comandos e uma para os dados. A informação é recebida pela conexão de dados, que é gerada aleatoriamente e transmitida pela conexão dos comandos.

O mecanismo ARQ (*Automatic Repeat ReQuest*) é um mecanismo que pede a retransmissão de pacotes perdidos e/ou com erros. O mecanismo usado no TCP é uma variação do mecanismo **Go-Back-N**, pelo que as mensagens **ACK** têm um

número de sequência e quando há uma falha, os pacotes são retransmitidos um de cada vez. É de realçar, também, que o controlo de erros é baseado na sequência de bytes e não nos pacotes de informação.

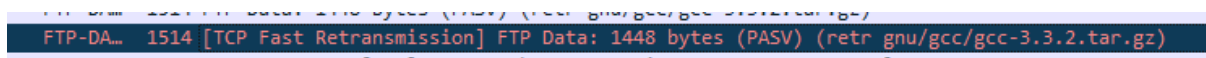


Figura 16 - Retransmissão de um pacote

Na figura 16, é possível observar um exemplo de uma retransmissão de um pacote.

O mecanismo de controlo de congestionamento no TCP gere a transmissão de informação de várias fontes, permitindo um fluxo de informação justo e eficiente.

Analisando os gráficos obtidos das transferências, observam-se as fases do controlo de congestionamento como o **Slow Start** e o **Congestion Avoidance**. Para além das fases, também é possível reparar que a velocidade da transferência diminui quando há transferências em simultâneo.

A velocidade diminuiu de **12500 pacotes/s** para **11500 pacotes/s**. Apesar que, teoricamente, a velocidade deveria baixar para metade, visto que os recursos estão a ser partilhados.

Conclusões

Da análise ao presente projeto de trabalho resulta que é possível configurar uma rede de computadores capaz de comunicar com servidores, ainda que conectados à *Internet*, nomeadamente, realizar uma transferência de um servidor **FTP**.

Na elaboração deste projeto, colocou-se em prática alguns dos procedimentos de redes de computadores, como a configuração de uma **rede IP**, **Routing**, **NAT** e **DNS**.

Anexo I - Código Fonte

O código fonte do trabalho laboratorial encontra-se no ficheiro **RCT05G06.zip**, na pasta **downloadApp**. Nessa mesma pasta encontram-se os ficheiros necessários para executar o projeto.

Para correr a aplicação deve:

1. correr o "gcc" : **gcc download.c -o download;**
2. correr o executável com o devido parâmetro: **./download ftp://[<user>:<password>@]<host>/<url-path>**
3. Um exemplo de execução seria: **./download ftp://ftp.up.pt/pub/kodi/timestamp.txt**