

# Comunicações por Computador - Relatório TP1

João Nunes (A82300)      Luís Braga (A82088)  
Luís Martins (A82298)  
Grupo 52

1 de Março 2018

# 1 Questões e Respostas

1. Inclua no relatório uma tabela em que identifique, para cada comando executado, qual o protocolo de aplicação, o protocolo de transporte, porta de atendimento e *overhead* de transporte, como ilustrado no exemplo seguinte:

De acordo com o enunciado do trabalho prático, foram executados os diversos comandos com os resultados expressos na seguinte tabela:

Comando usado (aplicação)	Protocolo de Aplicação (se aplicável)	Protocolo de transporte (se aplicável)	Porta de atendimento (se aplicável)	Overhead de transporte em bytes (se aplicável)
Ping	-	-	-	-
tracert	-	UDP	33434 a 33534	8
telnet	telnet	TCP	23	20
ftp	ftp	TCP	21	20
Tftp	tftp	UDP	69	8
browser/http	http	TCP	80	20
nslookup	DNS	UDP	53	8
ssh	ssh	TCP	22	20

## Ping:

No caso do ping, este não irá possuir protocolo de aplicação nem de transporte e porta de atendimento. Por consequência, não terá nenhum overhead de transporte visto que não possui protocolo de transporte, uma vez que este opera ao mandar apenas mensagens ICMP para o destino à espera de obter um ICMP echo reply.

Este comportamento poderá ser observado no seguinte printscreen onde foi efetuado um ping para o endereço IP 193.136.9.33.

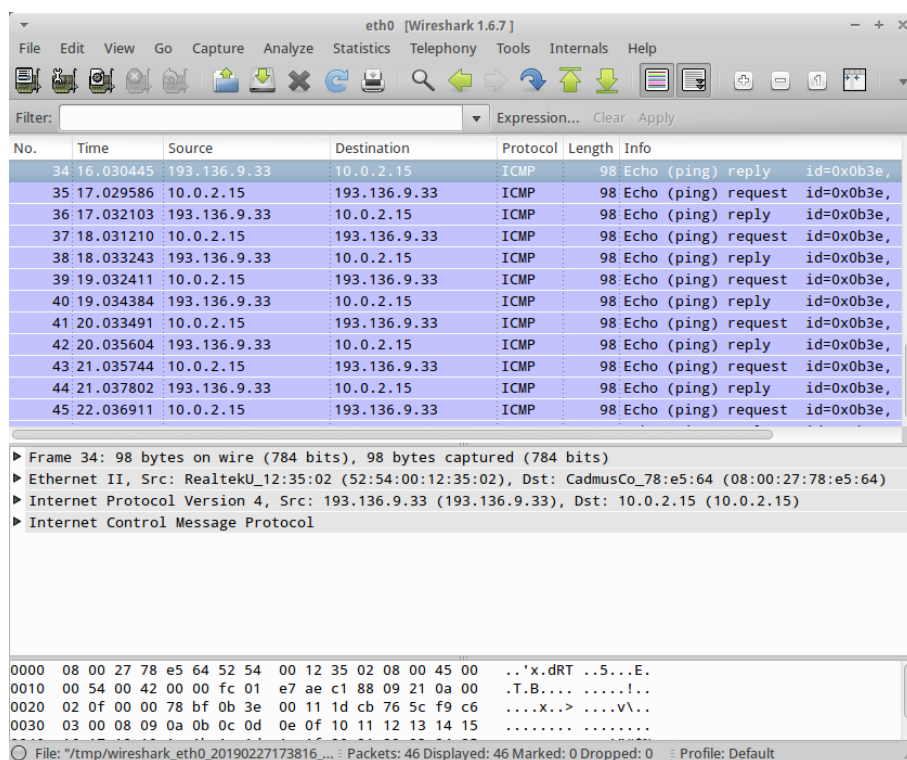


Figura 1: Printscreen das tramas capturadas usando o comando ping.

### traceroute:

No traceroute não existe protocolo de aplicação. Este possui um protocolo de transporte, sendo esse o UDP (User Datagram Protocol). As portas de atendimento também variam entre o 33434 e o 33534 possuindo 8 bytes para o overhead de transporte, o mesmo poderá ser verificado na próxima figura:

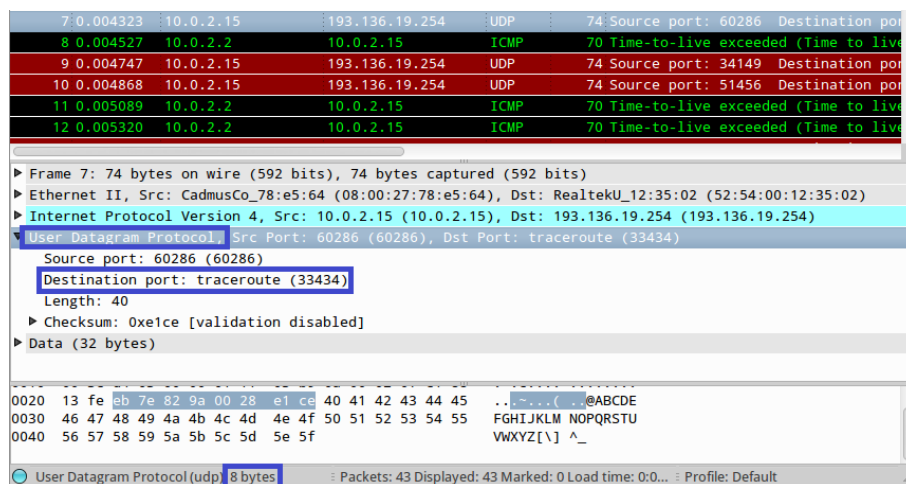


Figura 2: Printscreen das tramas capturadas usando o comando traceroute.

### telnet:

No comando telnet é utilizado o protocolo de aplicação com este mesmo nome. O protocolo é utilizado para providenciar uma comunicação, utilizando uma conexão terminal virtual. O protocolo de transporte é o TCP possuindo uma porta de atendimento 23 com 20 bytes de overhead.

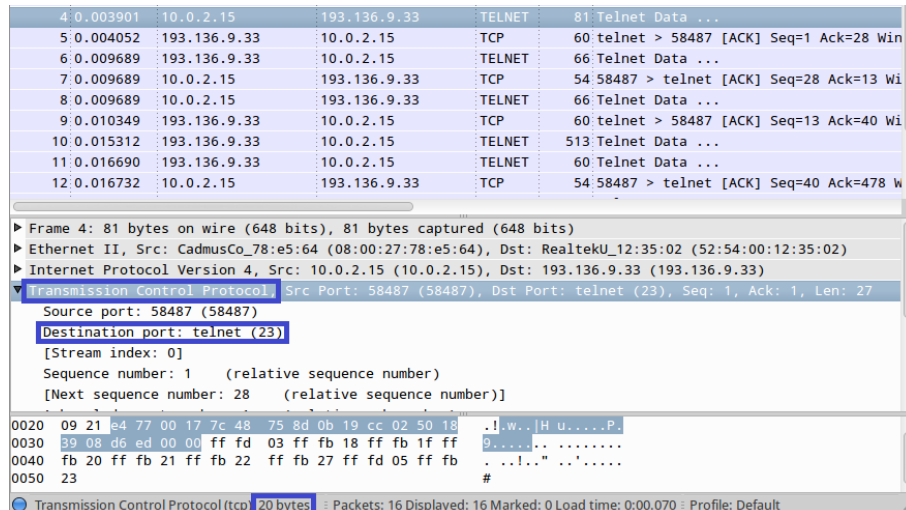


Figura 3: Printscreen das tramas capturadas usando o comando telnet.

### ftp:

O ftp utiliza o protocolo, com o mesmo nome, sendo este usado para a transferência de dados. O protocolo de aplicação é o ftp sendo que o de transporte é o TCP, a porta de atendimento é a 21 e o overhead é de 20 bytes.

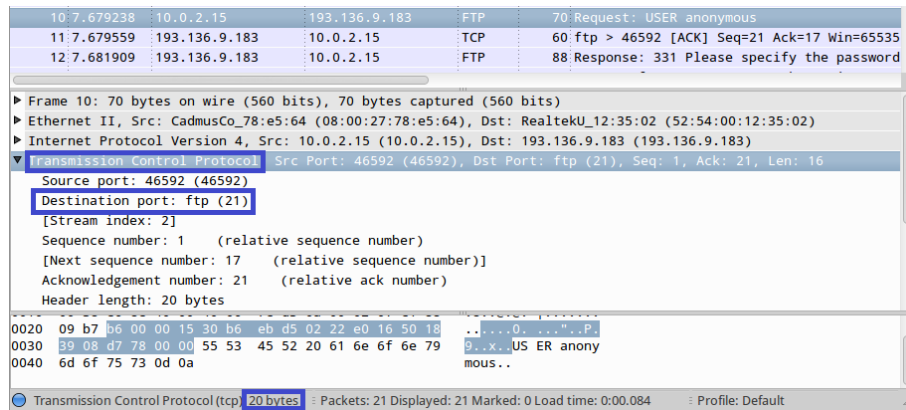


Figura 4: Printscreen das tramas capturadas usando o comando ftp.

### Tftp:

Neste caso foi utilizado o comando `curl tftp://gr2018.ddns.net/file1`, este comando funciona com um protocolo de transferência de dados simples, semelhante ao ftp. Portanto o protocolo de aplicação é o tftp, o de transporte o UDP com a porta de atendimento 69, e um overhead de transporte de 8 bytes.

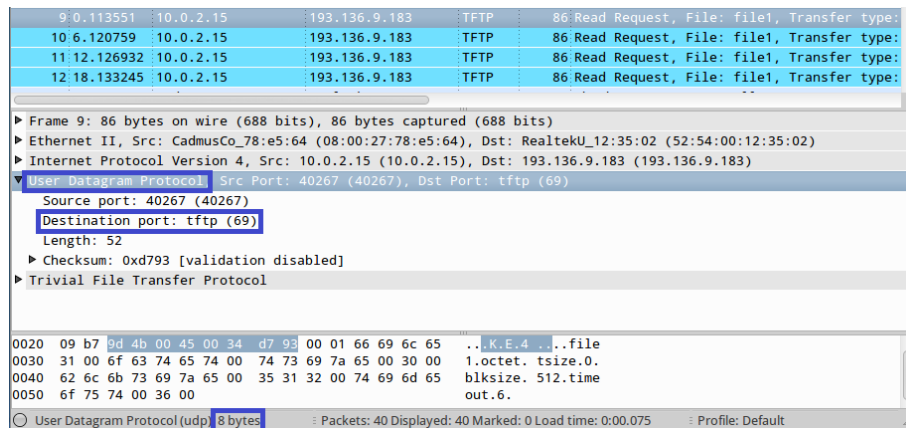


Figura 5: Printscreen das tramas capturadas usando o comando tftp.

## http:

De modo a capturar o tráfego, foi utilizado o comando:

```
wget http://marco.uminho.pt/disciplinas/CC-MIEI/
```

Onde foi possível depois da análise da trama capturada que o protocolo de aplicação é o http e o de transporte o TCP, a porta de atendimento é a 80 e possui 20 bytes de overhead de transporte.

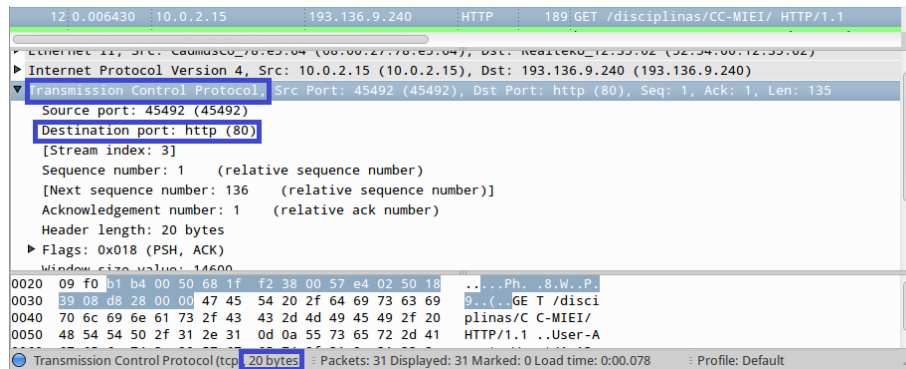


Figura 6: Printscreen da trama capturada usando o comando wget.

## nslookup:

O nslookup funciona como uma ferramenta de resolução de nomes como tal, o protocolo de aplicação é o DNS (Domain Name System), o de transporte é o UDP com porta de atendimento 53 e um overhead de 8 bytes.

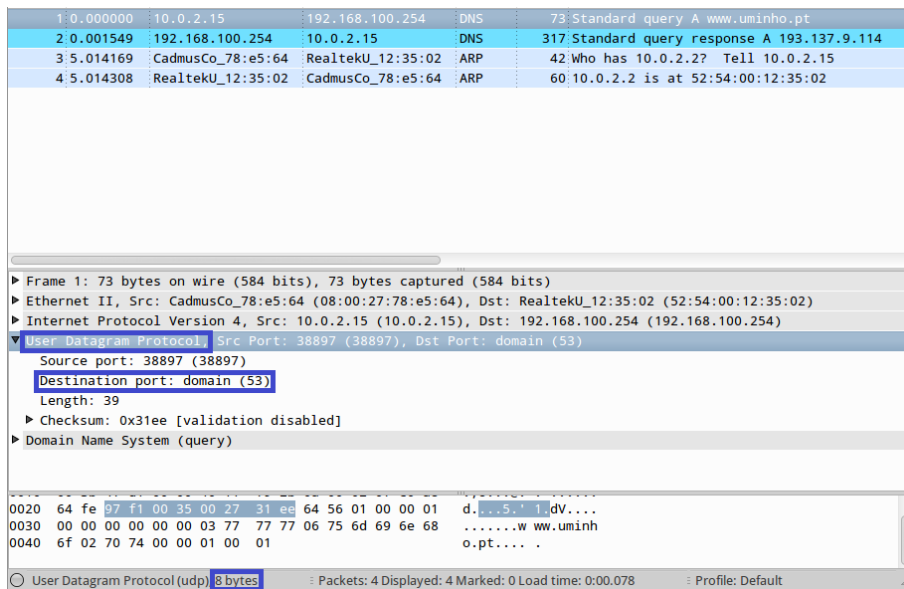


Figura 7: Printscreen da trama capturada usando o comando wget.

### ssh:

O ssh é um protocolo de rede criptográfico cujo intuito é operar em redes de forma segura através de uma rede insegura, tendo sido efetuado um login remoto. O protocolo de aplicação portanto é ssh, o de transporte o TCP, a porta de atendimento a 22 e o overhead é de 20 bytes.

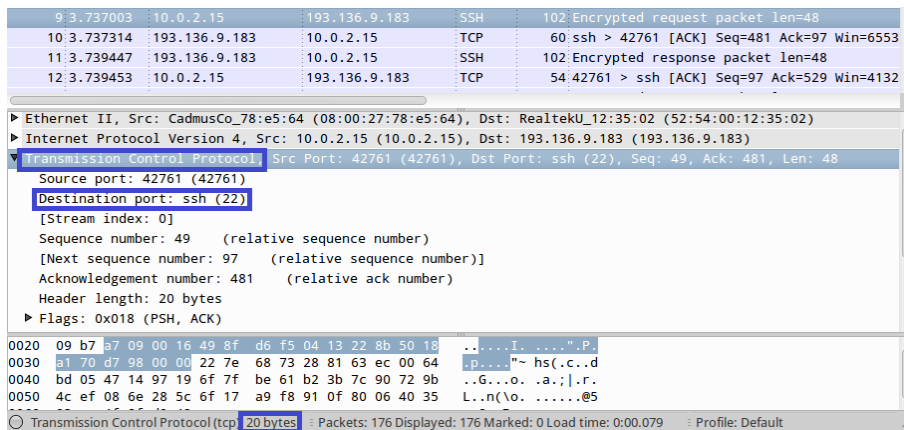


Figura 8: Printscreen da trama capturada usando o comando ssh.

2. Uma representação num diagrama temporal das transferências da file1 por FTP e TFTP respetivamente. Se for caso disso, identifique as fases de estabelecimento de conexão, transferência de dados e fim de conexão. Identifica também claramente os tipos de segmentos trocados e os números de sequência usados quer nos dados como nas confirmações.

#### FTP:

O protocolo de transferência de arquivos, *File Transfer Protocol* (FTP), usa como protocolo de transporte o TCP. Portanto, o procedimento protocolar, poderá ser dividido em três fases distintas:

- Estabelecimento de conexão;
- Transferência de dados;
- Fim da conexão.

Na primeira fase (estabelecimento da conexão), é enviado um **SYN** (Synchronize) do Servidor ao Cliente (neste caso), de maneira a sincronizar e iniciar a conexão entre ambas as entidades. O remetente desse sinal define o número de sequência de segmento.

Na fase de transferência de dados, o TCP possui vários mecanismos que asseguram uma certa robustez, como por exemplo, quando o recetor recebe um segmento de dados, é verificado o **SEQ** e se este for igual ao próximo, então os dados são recebidos em ordem.

Por fim, após a transferência de dados, a conexão termina, sendo a responsabilidade do término desta ligação distribuída entre os dois intervenientes. De modo a fechar a conexão, é necessário ser mandado um **FIN**, a resposta a este FIN será um **ACK**. Por sua vez, o outro interveniente irá proceder da mesma maneira.

Todas estas fases estão representadas no diagrama abaixo.



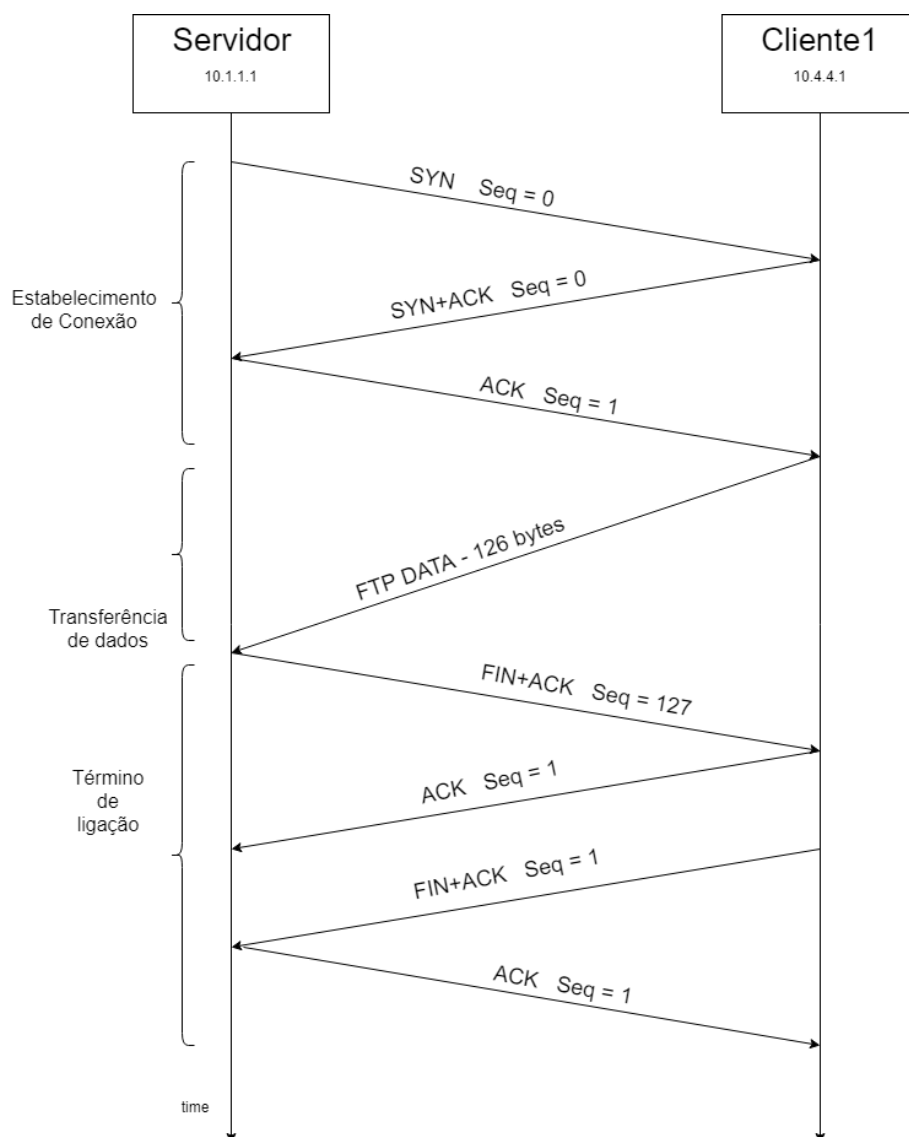


Figura 9: Diagrama temporal da transferência FTP da file1.

Este diagrama foi elaborado, tendo por base a seguinte sequência de captura de tramas:

79	261.666477	10.1.1.1	10.4.4.1	TCP	74 ftp-data > 58085 [SYN] Seq=0 Win=14600 Len=0 MSS=1460 SACK_PERM=1 TSval=380943 TSecr=0 WS=16
80	261.666622	10.4.4.1	10.1.1.1	TCP	74 58085 > ftp-data [SYN, ACK] Seq=0 Ack=1 Win=14480 Len=0 MSS=1460 SACK_PERM=1 TSval=380943 TSecr=380943 WS=16
81	261.666802	10.1.1.1	10.4.4.1	TCP	66 ftp-data > 58085 [ACK] Seq=1 Ack=1 Win=14608 Len=0 TSval=380943 TSecr=380943
82	261.666804	10.1.1.1	10.4.4.1	FTP	105 Response: 150 Here comes the directory listing.
83	261.666981	10.1.1.1	10.4.4.1	FTP-DAT	192 FTP Data: 126 bytes
84	261.666981	10.1.1.1	10.4.4.1	TCP	66 ftp-data > 58085 [FIN, ACK] Seq=127 Ack=1 Win=14608 Len=0 TSval=380943 TSecr=380943
85	261.667743	10.4.4.1	10.1.1.1	TCP	66 58085 > ftp-data [ACK] Seq=1 Ack=127 Win=14480 Len=0 TSval=380943 TSecr=380943
86	261.667743	10.4.4.1	10.1.1.1	TCP	66 58085 > ftp-data [FIN, ACK] Seq=1 Ack=128 Win=14480 Len=0 TSval=380943 TSecr=380943
87	261.667901	10.1.1.1	10.4.4.1	TCP	66 ftp-data > 58085 [ACK] Seq=128 Ack=2 Win=14608 Len=0 TSval=380943 TSecr=380943

Figura 10: Captura Wireshark.

## TFTP:

O **TFTP** usa como protocolo de transporte o UDP.

Primeiramente, é feito um *Read Request* dos dados. De seguida, é transferido o pacote de dados do servidor para o cliente. Por fim, é feito um ACK dos dados recebidos.

O mesmo funcionamento poderá ser verificado na seguinte sequência de tramas capturadas.

19	81.748331	10.4.4.1	10.1.1.1	TFTP	56 Read Request, File: file1, Transfer type: octet
20	81.748634	00:00:00_aa:00:16	Broadcast	ARP	42 Who has 10.1.1.254? Tell 10.1.1.1
21	81.748639	00:00:00_aa:00:12	00:00:00_aa:00:16	ARP	42 10.1.1.254 is at 00:00:00:aa:00:12
22	81.748764	10.1.1.1	10.4.4.1	TFTP	303 Data Packet, Block: 1 (last)
23	81.749049	10.4.4.1	10.1.1.1	TFTP	46 Acknowledgement, Block: 1

Figura 11: Captura Wireshark.

Da figura anterior foi elaborado o seguinte diagrama temporal:

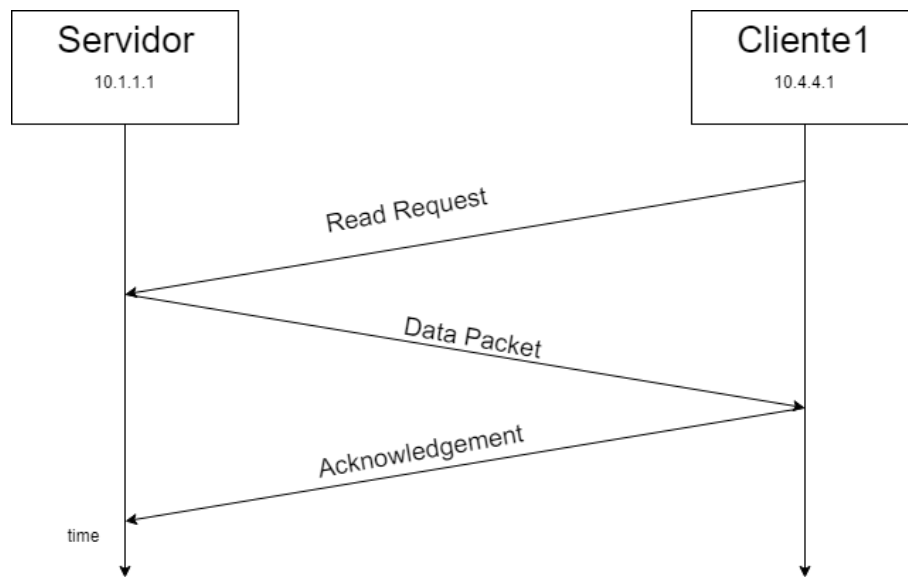


Figura 12: Diagrama temporal da transferência TFTP da file1.

3. Com base nas experiências realizadas, distinga e compare sucintamente as quatro aplicações de transferência de arquivos que usou nos seguintes pontos (i) uso da camada de transporte; (ii) eficiência na transferência; (iii) complexidade; (iv) segurança;

i ) Uso da camada de transporte

1. **SFTP**

- TCP

2. **FTP**

- TCP

3. **TFTP**

- UDP

4. **HTTP**

- TCP

## ii ) Eficiência na Transferência

### 1. SFTP

- É melhor que o protocolo FTP em termos de eficiência, mesmo assim peca em relação ao TFTP e o HTTP em termos de eficiência.

```
root@Cliente1:/tmp/pycore.52804/Cliente1.conf# less file-ping-output
root@Cliente1:/tmp/pycore.52804/Cliente1.conf# rm /root/.ssh/known_hosts
root@Cliente1:/tmp/pycore.52804/Cliente1.conf# sftp core@10.1.1.1
The authenticity of host '10.1.1.1 (10.1.1.1)' can't be established.
RSA key fingerprint is d2:9d:73:1a:65:5d:15:73:56:24:31:36:32:02:b6:eb.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '10.1.1.1' (RSA) to the list of known hosts.
core@10.1.1.1's password:
Connected to 10.1.1.1.
sftp> pwd
Remote working directory: /home/core
sftp> cd /srv/ftp
sftp> dir
file1  file2
sftp> get file1
Fetching /srv/ftp/file1 to file1
/srv/ftp/file1                                100% 257    0.3KB/s   00:00
sftp> quit
root@Cliente1:/tmp/pycore.52804/Cliente1.conf#
```

Figura 13: Transferência do file1 segundo o SFTP.

### 1. FTP

- É ideal para dados de pequeno tamanho, para dados de tamanho maior já demonstra pouca eficiência na transferência de dados.

### 2. TFTP

- É um protocolo de transferência rápido, apesar de pecar em termos de segurança.

### 3. HTTP

- Bastante eficiente e mais rápido que FTP.

## iii ) Complexidade

### 1. SFTP

- O SFTP, sendo uma variante do FTP, é de complexidade elevada tal como o FTP.

## 2. FTP

- O FTP embora tendo sido um dos primeiros protocolos de aplicação a ser desenvolvidos, é bastante complexo. Uma vez que obriga a implementar uma gestão concorrente entre várias conexões TCP mantidas entre o cliente e o servidor.

## 3. TFTP

- É uma aplicação de transferência semelhante ao FTP. Foi construído com a intenção de ser simples. A aplicação TFTP restringe as operações a simples transferências de ficheiros e não necessita de autenticação (como se pode confirmar no seguinte tópico iv - Segurança).

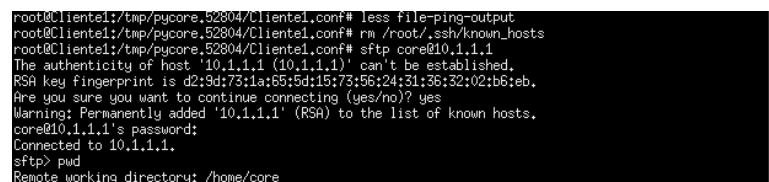
## 4. HTTP

- HTTP foi desenhado para ser simples e possível de ser compreendido facilmente, sendo assim fácil trabalhar com ele.

### iv ) Segurança

#### 1. SFTP

- Os dados são transferidos através de uma conexão a uma shell segura, que é uma network encriptada sendo acedida através de um processo de login.

A terminal window showing the SFTP login process. The user is at a root prompt on a client machine. They run 'less file-ping-output', then 'rm /root/.ssh/known\_hosts', and finally 'sftp core@10.1.1.1'. The terminal shows the SSH warning about the host's authenticity, the user's confirmation to continue, the warning about adding the host to the known hosts list, the password prompt, and the successful connection to 10.1.1.1. The remote working directory is shown as /home/core.

```
root@Cliente1:/tmp/pycore.52804/Cliente1.conf# less file-ping-output
root@Cliente1:/tmp/pycore.52804/Cliente1.conf# rm /root/.ssh/known_hosts
root@Cliente1:/tmp/pycore.52804/Cliente1.conf# sftp core@10.1.1.1
The authenticity of host '10.1.1.1 (10.1.1.1)' can't be established.
RSA key fingerprint is d2:9d:73:1a:65:5d:15:73:56:24:31:36:32:02:b6:eb.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '10.1.1.1' (RSA) to the list of known hosts.
core@10.1.1.1's password:
Connected to 10.1.1.1.
sftp> pwd
Remote working directory: /home/core
```

Figura 14: Processo de login segundo o SFTP.

#### 1. FTP

- De modo aos dados serem transferidos segundo FTP foi necessário passar por um processo de login. Contudo os dados transferidos não são encriptados pelo que, portanto, podem facilmente ser acedidos por outros, sendo então considerado um protocolo inseguro.

```

root@Cliente1:/tmp/pycore.52804/Cliente1.conf# ftp 10.1.1.1
Connected to 10.1.1.1.
220 (vsFTPd 2.3.5)
Name (10.1.1.1:root): anonymous
331 Please specify the password.
Password:
230 Login successful.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> status
Connected to 10.1.1.1.
No proxy connection.
Connecting using address family: any.
Mode: stream; Type: binary; Form: non-print; Structure: file
Verbose: on; Bell: off; Prompting: on; Globbing: on
Store unique: off; Receive unique: off
Case: off; CR stripping: on
Quote control characters: on
Ntrans: off
Nmap: off
Hash mark printing: off; Use of PORT cmds: on
Tick counter printing: off
ftp> pwd
257 "/"
ftp> dir
200 PORT command successful. Consider using PASV.
150 Here comes the directory listing.
-rw-r--r--  1 0      0      257 Feb 28 19:28 file1
-rwxr-xr-x  1 0      0     104508 Feb 28 19:28 file2
226 Directory send OK.
ftp>

```

Figura 15: Processo de login segundo o FTP.

## 1. TFTP

- Não é tão seguro como FTP, pois não possui as mesmas características nem foi usado qualquer tipo de processo de login, como se pode ver na figura abaixo em que se executou o comando *status* logo após o comando *atftp*. Acrescentando o facto de que a directoria no servidor tem de ter acesso para escrita para todos.

```

root@Cliente1:/tmp/pycore.52804/Cliente1.conf# atftp 10.1.1.1
tftp> status
Connected: 10.1.1.1 port 69
Mode:      octet
Verbose:   off
Trace:     off
Options
  tsize:    disabled
  blksize:  disabled
  timeout:  disabled
  multicast: disabled
mtftp variables
  client-port: 76
  mcast-ip:    0.0.0.0
  listen-delay: 2
  timeout-delay: 2
Last command: ---
tftp> get file1
Overwrite local file [y/n]? y
tftp> quit
root@Cliente1:/tmp/pycore.52804/Cliente1.conf#

```

Figura 16: Processo de uso de TFTP.

## 1. HTTP

- A aplicação de transferência de ficheiros HTTP não possui qualquer tipo de mecanismo de autenticação pelo foi só usar o comando *wget*. No entanto, é mais seguro que FTP.

**4. As características das ligações de rede têm uma enorme influência nos níveis de Transporte de Aplicação. Discuta, relacionando a resposta com as experiências realizadas, as influências das situações de perda ou duplicação de pacotes IP no desempenho global de Aplicações fiáveis (se possível, relacionando com alguns dos mecanismos de transporte envolvidos).**

É lógico que, aumentando a percentagem de *package loss* na ligação, se verifique que pacotes são perdidos.

Aplicações de transferência fiáveis como, por exemplo, SFTP, que utiliza protocolo de transporte TCP, têm implementadas funcionalidades que fazem com que todos os dados cheguem ao destino.

Assim, ao aumentar o *package loss*, usando simultaneamente protocolos de transferência fiáveis vamos ter um maior overhead, pois quanto mais perda, mais tráfego vai circular rede para corrigir os erros, o que diminui o desempenho global da rede.

sftp file1Alfa [Wireshark 1.6.7]

File Edit View Go Capture Analyze Statistics Telephony Tools Internals Help

Filter: Expression... Clear Apply

No.	Time	Source	Destination	Protocol	Length	Info
24	4.956631	10.3.3.1	10.1.1.1	SSH	98	Encrypted request packet len=32
25	4.956631	10.3.3.1	10.1.1.1	SSH	130	Encrypted request packet len=64
26	4.956645	10.3.3.1	10.1.1.1	TCP	66	40321 > ssh [FIN, ACK] Seq=529 Ack=801 Win=0 Len=0
27	4.957942	10.1.1.1	10.3.3.1	TCP	66	ssh > 40321 [ACK] Seq=801 Ack=530 Win=156 Len=0
28	4.957944	10.1.1.1	10.3.3.1	TCP	66	ssh > 40321 [FIN, ACK] Seq=801 Ack=530 Win=0 Len=0
29	4.963111	10.3.3.1	10.1.1.1	TCP	66	40321 > ssh [ACK] Seq=530 Ack=802 Win=264 Len=0
30	4.963111	10.3.3.1	10.1.1.1	TCP	66	[TCP Dup ACK 29#1] 40321 > ssh [ACK] Seq=530 Ack=802 Win=264 Len=0
31	4.963296	10.1.1.1	10.3.3.1	TCP	54	ssh > 40321 [RST] Seq=802 Win=0 Len=0
32	6.686112	00:00:00_aa:00:12	00:00:00_aa:00:16	ARP	42	Who has 10.1.1.1? Tell 10.1.1.254
33	6.686283	00:00:00_aa:00:16	00:00:00_aa:00:12	ARP	42	10.1.1.1 is at 00:00:00_aa:00:16
34	10.000999	10.1.1.254	224.0.0.5	OSPF	78	Hello Packet
35	10.137167	fe80::200:ff:feaa:1ff02::5	ff02::5	OSPF	90	Hello Packet

▶ Flags: 0x010 (ACK)  
 Window size value: 2641  
 [Calculated window size: 2641]  
 [Window size scaling factor: -1 (unknown)]  
 ▶ Checksum: 0x3b0d [validation disabled]  
 ▶ Options: (12 bytes)  
 ▼ [SEQ/ACK analysis]  
 ▶ [TCP Analysis Flags]  
 [Duplicate ACK #: 1]  
 ▼ [Duplicate to the ACK in frame: 29]  
 ▶ [Expert Info (Note/Sequence): Duplicate ACK (#1)]

0000 00 00 00 aa 00 16 00 00 00 aa 00 12 08 00 45 00 .....E.  
 0010 00 34 f7 89 40 00 3d 06 2e 35 0a 03 03 01 0a 01 .4..@.=. .5.....  
 0020 01 01 9d 81 00 16 1a c6 8c df 25 71 3a bb 80 10 .....%q:..  
 0030 0a 51 3b 0d 00 00 01 01 08 0a 00 0d 39 eb 00 0d .Q:.....9...

Frame (frame), 66 bytes    Packets: 35 Displayed: 35 Marked: 0 Load time: 0:00.074    Profile: Default

Figura 17: SFTP segundo a conexão com perdas, atrasos e duplicações.



sftpCliente1Serv1 [Wireshark 1.6.7]

File Edit View Go Capture Analyze Statistics Telephony Tools Internals Help

Filter: Expression... Clear Apply

No.	Time	Source	Destination	Protocol	Length	Info
348	911.403714	10.4.4.1	10.1.1.1	TCP	66	40140 > ssh [ACK] Seq=2802 Ack=3682 Win=2
349	915.284517	10.4.4.1	10.1.1.1	TCP	98	[TCP segment of a reassembled PDU]
350	915.284858	10.1.1.1	10.4.4.1	TCP	194	[TCP segment of a reassembled PDU]
351	915.285201	10.4.4.1	10.1.1.1	TCP	66	40140 > ssh [ACK] Seq=2834 Ack=3810 Win=2
352	915.285201	10.4.4.1	10.1.1.1	TCP	98	[TCP segment of a reassembled PDU]
353	915.285201	10.4.4.1	10.1.1.1	TCP	130	[TCP segment of a reassembled PDU]
354	915.285201	10.4.4.1	10.1.1.1	TCP	66	40140 > ssh [FIN, ACK] Seq=2930 Ack=3810
355	915.286554	10.1.1.1	10.4.4.1	TCP	66	ssh > 40140 [ACK] Seq=3810 Ack=2931 Win=2
356	915.286986	10.1.1.1	10.4.4.1	TCP	66	ssh > 40140 [FIN, ACK] Seq=3810 Ack=2931
357	915.287104	10.4.4.1	10.1.1.1	TCP	66	40140 > ssh [ACK] Seq=2931 Ack=3811 Win=2
358	920.019672	fe80::200:ff:feaa:1ff02::5		OSPF	90	Hello Packet
359	920.030622	10.1.1.254	224.0.0.5	OSPF	78	Hello Packet

▼ Transmission Control Protocol, Src Port: 40140 (40140), Dst Port: ssh (22), Seq: 2930, Ack: 3810, Len: 0

Source port: 40140 (40140)  
Destination port: ssh (22)  
[Stream index: 2]  
Sequence number: 2930 (relative sequence number)  
Acknowledgement number: 3810 (relative ack number)  
Header length: 32 bytes

► Flags: 0x011 (FIN, ACK)  
Window size value: 1689  
[Calculated window size: 27024]  
[Window size scaling factor: 16]

0000	00 00 00 aa 00 16 00 00 00 aa 00 12 08 00 45 00	.....E.
0010	00 34 e7 84 40 00 3e 06 3c 39 0a 04 04 01 0a 01	.4..@.>. <9.....
0020	01 01 9c cc 00 16 d1 03 ed f9 cc 83 1a bd 80 11	.....
0030	06 99 19 2d 00 00 01 01 08 0a 00 04 25 df 00 04	.....%...

Frame (frame), 66 bytes      Packets: 359 Displayed: 359 Marked: 0 Load time: 0:00.076      Profile: Default

Figura 18: SFTP na conexão normal.

## 2 Conclusão

Neste trabalho prático, foram consolidados certos aspetos tais como classificar os protocolos de transporte (UDP e TCP) em diversas aplicações diferentes e o overhead de cada um destes protocolos.

De seguida, foram testadas cada uma destas aplicações no CORE através de uma topologia fornecida anteriormente. Foi feita uma análise dos resultados entre as várias aplicações, sendo portanto possível estabelecer uma conexão entre a consequência do uso de diferentes protocolos de transporte em diferentes aplicações.

Finalmente, foi feita uma análise numa situação de package loss de forma a perceber como os mecanismos de correção de erros funcionam.

Portanto, com a elaboração deste trabalho, foram aprofundados os níveis de compreensão e conhecimento em relação às diferentes vertentes da camada de transporte, enriquecendo os conhecimentos já pré existentes em relação a esta.