

Universidade do Minho

Licenciatura em Engenharia Informática

Comunicação por Computadores - Trabalho Prático nº 1

Grupo 73

Tiago Fernandes Ribeiro - A93203 Francisco Reis Izquierdo - A93241 Francisco Alberto do Fundo Novo - A89567

Ano Letivo - 2021/2022

Índice

1. Questões e Respostas (Parte 1)	3
2. Questões e Respostas (Parte 2)	Erro! Marcador não definido.
3. Anexos	Erro! Marcador não definido.
4. Conclusão	9

1. Questões e Respostas (Parte 1)

2.1) De que forma as perdas e duplicações de pacotes afetaram o desempenho das aplicações? Que camada lidou com as perdas e duplicações: transporte ou aplicação? Responda com base nas experiências feitas e nos resultados observados.

R: Ao comparar as transferências dos ficheiros *file1* e *file2* com os vários protocolos concluímos que a perda e duplicação de pacotes reduz a velocidade e/ou a taxa de transferência da conexão, neste caso Portátil → Servidor e Grilo → Servidor. Isto pode ser comprovado com os tempos de transferência dos ficheiros, sendo a camada de transporte que lida com as perdas e duplicações de transporte.

Figura 1: Tabela de transferência de ficheiros

	Portátil1				Grilo			
	Ficheiro 1		Ficheiro 2		Ficheiro 1		Ficheiro 2	
Protocolo	Tempo de exec.	Taxa de transf.	Tempo de exec.	Taxa de transf.	Tempo de execu.	Taxa de transf.	Tempo de exec.	Taxa de transf.
SFTP	00:00	162.9KB/s	00:00	11.7MB/s	00:00	19.3KB/s	00:00	560KB/s
FTP	00:00	1.3435MB/s	00:00	27.8756MB/s	00:00	1.3026MB/S	00:02	6.2654MB/S
TFTP	-----	-----	-----	-----	-----	-----	-----	-----
HTTP	00:00	17,4MB/s	00:005	26,9MB/s	00:00	20,7MB/s	00:02	6,19MB/s

Nota: É de salientar que Grilo tem uma conectividade cuja largura de banda é de 100Mbps enquanto Portatil1 tem uma conectividade cuja largura de banda é de 1Gbps.

Além disso, não conseguimos obter o tempo de execução e a taxa de transferência dos ficheiros por TFTP.

Na secção **Anexos** constam as imagens das experiências efetuadas, nomeadamente a transferência do *file1* e *file2* segundo cada um dos protocolos em causa.

2.2) Obtenha a partir do wireshark, ou desenhe manualmente, um diagrama temporal para a transferência de file1 por FTP. Foque-se apenas na transferência de dados [ftp-data] e não na conexão de controlo, pois o FTP usa mais que uma conexão em simultâneo. Identifique, se aplicável, as fases de início de conexão, transferência de dados e fim de conexão. Identifique também os tipos de segmentos trocados e os números de sequência usados quer nos dados como nas confirmações.

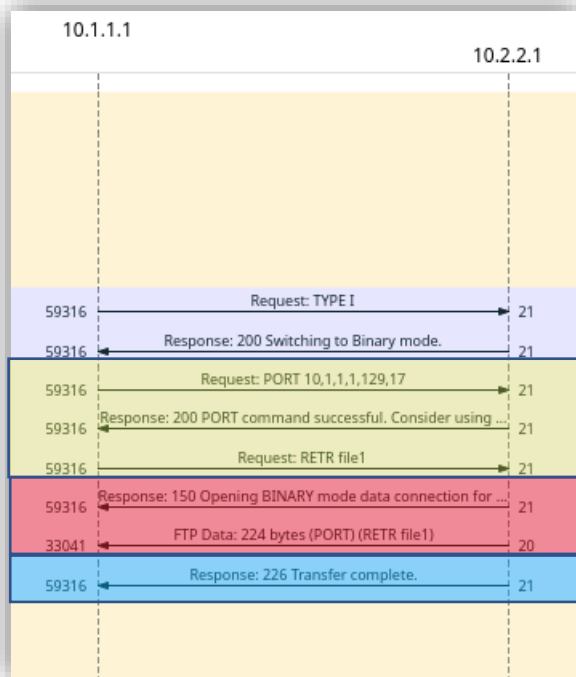


Figura 2: Diagrama temporal da transferência de file1 por FTP para Portátil

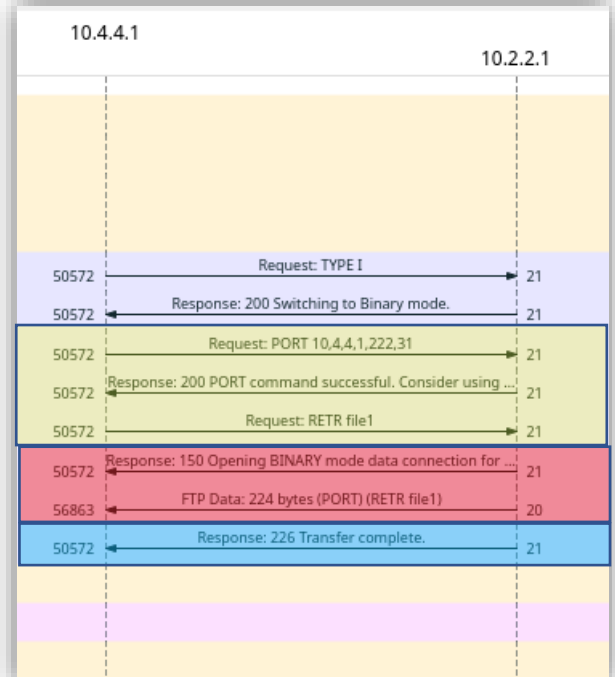


Figura 3: Diagrama temporal da transferência de file1 por FTP para Grilo

R: Após obter o diagrama temporal da transferência do *file1* através do wireshark, conclui-se o seguinte:

- Fase de início de conexão: sombreados a amarelo;
- Fase de transferência de dados: sombreado a vermelho;
- Fase de fim de conexão: sombreado a azul;

Isto advém do facto de o protocolo de aplicação FTP usar como protocolo de transporte o TCP que é orientado à conexão.

Além disso, ao analisar o cabeçalho de transporte conseguimos identificar os tipos de segmentos trocados, sendo estes segmentos TCP e também os números de sequência usados, sendo que nos dados o número de sequência é o **1** e nas confirmações o número de sequência usado é o **1, 9, 32, 83 e 147**.

2.3) Obtenha a partir do wireshark, ou desenhe manualmente, um diagrama temporal para a transferência de file1 por TFTP. Identifique, se aplicável, as fases de início de conexão, transferência de dados e fim de conexão. Identifique também os tipos de segmentos trocados e os números de sequência usados quer nos dados como nas confirmações.

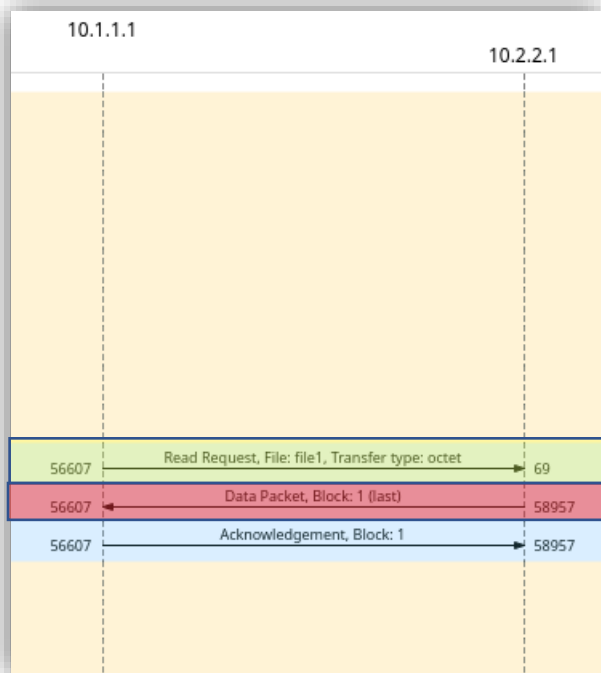


Figura 5: Diagrama temporal da transferência de file1 por TFTP para Portátil

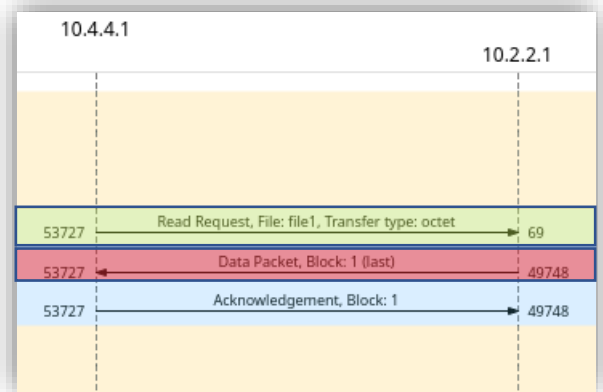


Figura 4: Diagrama temporal da transferência de file1 por TFTP para Grilo

R: Após obter o diagrama temporal da transferência do file1 através do wireshark, conclui-se o seguinte:

- Fase de início de conexão: sombreados a amarelo;
- Fase de transferência de dados: sombreado a vermelho;
- Fase de fim de conexão: sombreado a azul;

Isto advém do facto de o protocolo de aplicação TFTP usar como protocolo de transporte o UDP que é não é orientado à conexão.

Além disso, ao analisar o cabeçalho de transporte conseguimos identificar os tipos de segmentos trocados, sendo estes segmentos UDP e por isso não têm números de sequência.

2.4) Compare sucintamente as quatro aplicações de transferência de ficheiros que usou nos seguintes pontos:

- (i) uso da camada de transporte;
- (ii) eficiência;
- (iii) complexidade;
- (iv) segurança;

(i) Ao analisar as quatro aplicações de transferência de ficheiros usadas, verificamos que as aplicações SFTP, FTP e HTTP usam TCP como protocolo de camada de transporte, usando conexões uma vez que TCP é um protocolo com conexão, enquanto TFTP usa UDP como protocolo de camada de transporte, não usando conexões, uma vez que UDP é um protocolo sem conexões.

(ii) A eficiência das aplicações, tem por base os mecanismos implementados nas mesmas bem como o protocolo da camada de transporte usado. Assim, temos de forma crescente de eficiência:

TFTP → apesar de usar o protocolo UDP (que à partida seria mais rápido para transferências), os pacotes têm de ser reconhecidos (*acknowledged*) pela aplicação, pois o protocolo UDP não capta todos os erros em causa, sendo a aplicação a ter de verificar os mesmos.

HTTP → é eficiente na transferência de ficheiros pequenos. Uma vez que usa o protocolo da camada de transporte TCP, é a camada de transporte que faz a verificação de erros, mas apenas usa uma conexão.

FTP → além de usar o protocolo da camada de transporte TCP, sendo os erros verificados por este, usa duas conexões (canais) separados, um para estabelecer comunicações simples entre os clientes e outro que só é aberto quando necessário para transferir dados (é criada uma *thread* específica para o cliente em causa, para o transporte de dados de forma rápida e eficiente).

SFTP → é o mais eficaz de todos os protocolos de aplicação.

(iii) Em termos de complexidade das quatro aplicações é de salientar a forma como as aplicações estabelecem a ligação entre cliente e servidor bem como os mecanismos implementados nas mesmas e o protocolo da camada de transporte usado. Assim, temos de forma crescente de complexidade:

TFTP → em relação à transferência dos ficheiros foi necessário executar o servidor manualmente na *bash* do mesmo e preparar a diretoria onde se encontravam o(s) ficheiro(s) a transferir. Apesar disso, usa o protocolo de camada de transporte mais simples o protocolo UDP, que é uma conexão mais simples.

HTTP → em relação à transferência dos ficheiros foi necessário executar o servidor manualmente na *bash* do mesmo e preparar a diretoria onde se encontravam o(s) ficheiro(s) a transferir. Além disso, usa o protocolo de camada de transporte mais complexo, o protocolo TCP, mas apenas usa uma conexão.

FTP → em relação à transferência dos ficheiros foi necessário executar o servidor manualmente na *bash* do mesmo. Além disso, usa o protocolo de camada de transporte mais complexo, o protocolo TCP e usa mais do que uma conexão, fazendo uso de *threads* de forma a não encher o servidor.

SFTP → apesar de não ser necessário executar o servidor manualmente na *bash* do mesmo nem preparar a diretoria onde se encontravam o(s) ficheiro(s) a transferir, usa o protocolo de camada de transporte mais complexo, o protocolo TCP, fazendo uso de mais do que uma conexão e além disso tem de criptografar os dados transferidos.

(iv) Após uma breve pesquisa sobre as quatro aplicações, verificamos que as aplicações que oferecem mecanismos de autenticação e criptografia são as aplicações SFTP e FTP, sendo que os ficheiros transferidos são criptografados apenas na aplicação SFTP, contrariamente às aplicações FTP (que transfere dados em modo texto e/ou binário), TFTP (usa UDP para enviar e receber dados, sendo pouco seguro) e HTTP (que transfere dados em modo texto). Assim, a aplicação SFTP é a mais segura das quatro e a menos segura é a aplicação TFTP.

2. Questões e Respostas (Parte 2)

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	DNS	-----	-----	-----
tracert	DNS	UDP	33434	8
telnet	Telnet	TCP	23	20
ftp	FTP	TCP	20	20
tftp	TFTP	UDP	69	8
http (browser)	HTTP	TCP	80	20
ssh	SSHv2	TCP	22	20

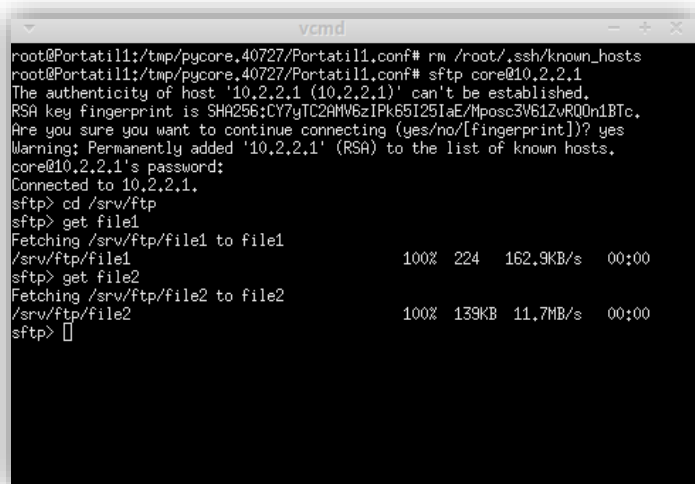
```

Frame 18: 290 bytes on wire (2320 bits), 290 bytes captured (2320 bits) on interface veth1.2.88, id
Ethernet II, Src: 00:00:00_aa:00:14 (00:00:00:aa:00:14), Dst: 00:00:00_aa:00:10 (00:00:00:aa:00:10)
Internet Protocol Version 4, Src: 10.2.2.1, Dst: 10.1.1.1
Transmission Control Protocol, Src Port: 20, Dst Port: 33041, Seq: 1, Ack: 1, Len: 224
  Source Port: 20
  Destination Port: 33041
  [Stream index: 1]
  [TCP Segment Len: 224]
  Sequence number: 1 (relative sequence number)
  Sequence number (raw): 1494660896
  [Next sequence number: 225 (relative sequence number)]
  Acknowledgment number: 1 (relative ack number)
  Acknowledgment number (raw): 2968223317
  1000 .... = Header Length: 32 bytes (8)
  ▶ Flags: 0x018 (PSH, ACK)
  Window size value: 502
  [Calculated window size: 64256]
  [Window size scaling factor: 128]
  Checksum: 0x8c4a [unverified]
  [Checksum Status: Unverified]
  Urgent pointer: 0
  ▶ Options: (12 bytes), No-Operation (NOP), No-Operation (NOP), Timestamps
  ▶ [SEQ/ACK analysis]
  ▶ [Timestamps]

```

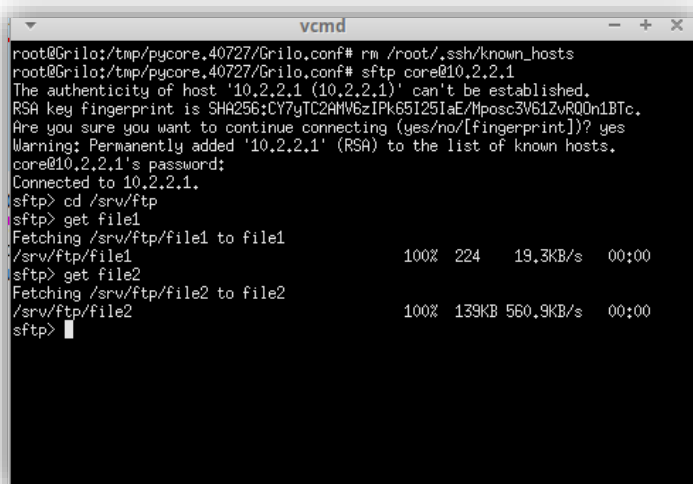
1:Exemplo de análise de pacotes por wireshark (conexão FTP)

3. Anexos



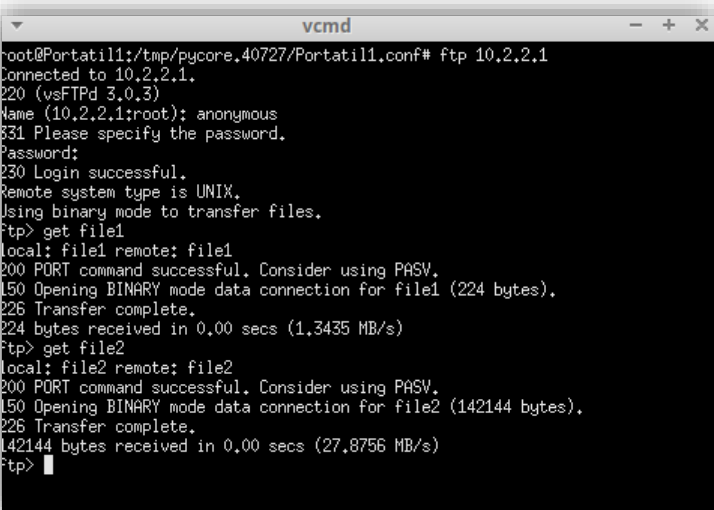
```
vcmd
root@Portatil1:/tmp/pycore.40727/Portatil1.conf# rm /root/.ssh/known_hosts
root@Portatil1:/tmp/pycore.40727/Portatil1.conf# sftp core@10.2.2.1
The authenticity of host '10.2.2.1 (10.2.2.1)' can't be established.
RSA key fingerprint is SHA256:CY7yTC2AMV6zIPk65I25IaE/Mposc3V61ZvRQ0n1BTc.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '10.2.2.1' (RSA) to the list of known hosts.
core@10.2.2.1's password:
Connected to 10.2.2.1.
sftp> cd /srv/ftp
sftp> get file1
Fetching /srv/ftp/file1 to file1
/srv/ftp/file1 100% 224 162,9KB/s 00:00
sftp> get file2
Fetching /srv/ftp/file2 to file2
/srv/ftp/file2 100% 139KB 11,7MB/s 00:00
sftp>
```

Figura 6: Transferências dos ficheiros por SFTP para Portatil1



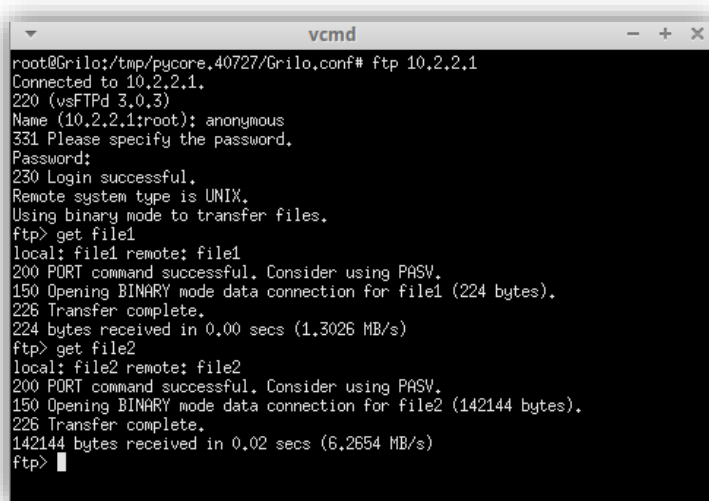
```
vcmd
root@Grilo:/tmp/pycore.40727/Grilo.conf# rm /root/.ssh/known_hosts
root@Grilo:/tmp/pycore.40727/Grilo.conf# sftp core@10.2.2.1
The authenticity of host '10.2.2.1 (10.2.2.1)' can't be established.
RSA key fingerprint is SHA256:CY7yTC2AMV6zIPk65I25IaE/Mposc3V61ZvRQ0n1BTc.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '10.2.2.1' (RSA) to the list of known hosts.
core@10.2.2.1's password:
Connected to 10.2.2.1.
sftp> cd /srv/ftp
sftp> get file1
Fetching /srv/ftp/file1 to file1
/srv/ftp/file1 100% 224 19,3KB/s 00:00
sftp> get file2
Fetching /srv/ftp/file2 to file2
/srv/ftp/file2 100% 139KB 560,9KB/s 00:00
sftp>
```

Figura 7: Transferências dos ficheiros por SFTP para Grilo



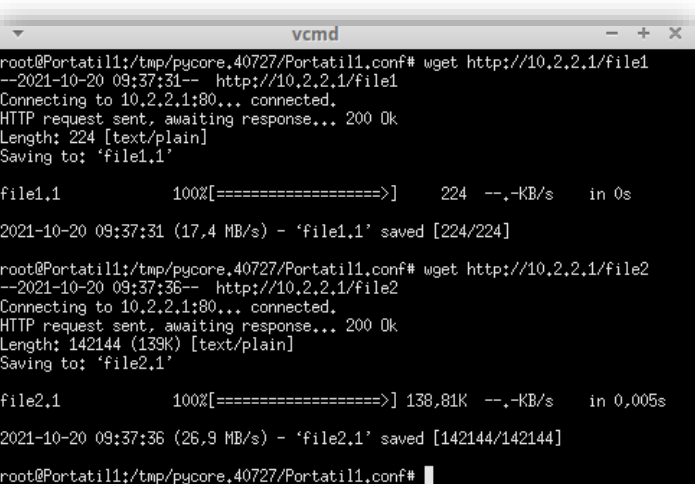
```
vcmd
root@Portatil1:/tmp/pycore.40727/Portatil1.conf# ftp 10.2.2.1
Connected to 10.2.2.1.
220 (vsFTPd 3.0.3)
Name (10.2.2.1:root): anonymous
331 Please specify the password.
Password:
230 Login successful.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> get file1
local: file1 remote: file1
200 PORT command successful. Consider using PASV.
150 Opening BINARY mode data connection for file1 (224 bytes).
226 Transfer complete.
224 bytes received in 0.00 secs (1,3435 MB/s)
ftp> get file2
local: file2 remote: file2
200 PORT command successful. Consider using PASV.
150 Opening BINARY mode data connection for file2 (142144 bytes).
226 Transfer complete.
142144 bytes received in 0.00 secs (27,8756 MB/s)
ftp>
```

Figura 11: Transferências dos ficheiros por FTP para Portatil1



```
vcmd
root@Grilo:/tmp/pycore.40727/Grilo.conf# ftp 10.2.2.1
Connected to 10.2.2.1.
220 (vsFTPd 3.0.3)
Name (10.2.2.1:root): anonymous
331 Please specify the password.
Password:
230 Login successful.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> get file1
local: file1 remote: file1
200 PORT command successful. Consider using PASV.
150 Opening BINARY mode data connection for file1 (224 bytes).
226 Transfer complete.
224 bytes received in 0.00 secs (1,3026 MB/s)
ftp> get file2
local: file2 remote: file2
200 PORT command successful. Consider using PASV.
150 Opening BINARY mode data connection for file2 (142144 bytes).
226 Transfer complete.
142144 bytes received in 0.02 secs (6,2654 MB/s)
ftp>
```

Figura 10: Transferências dos ficheiros por FTP para Grilo



```
vcmd
root@Portatil1:/tmp/pycore.40727/Portatil1.conf# wget http://10.2.2.1/file1
--2021-10-20 09:37:31-- http://10.2.2.1/file1
Connecting to 10.2.2.1:80... connected.
HTTP request sent, awaiting response... 200 Ok
Length: 224 [text/plain]
Saving to: 'file1.1'

file1.1 100%[=====] 224 --,-KB/s in 0s

2021-10-20 09:37:31 (17,4 MB/s) - 'file1.1' saved [224/224]

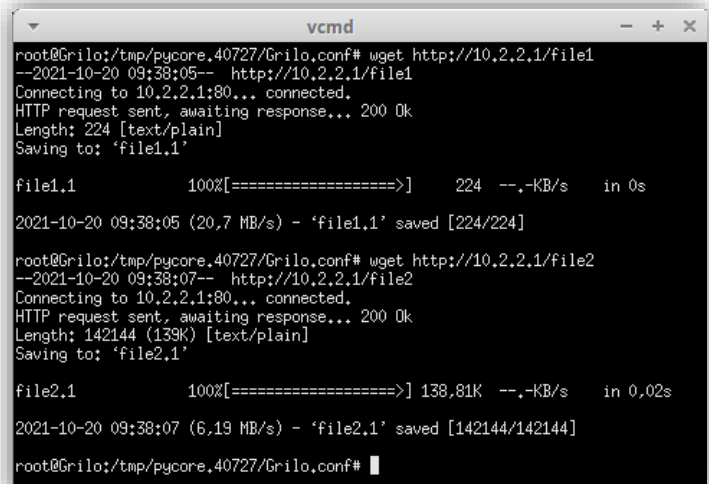
root@Portatil1:/tmp/pycore.40727/Portatil1.conf# wget http://10.2.2.1/file2
--2021-10-20 09:37:36-- http://10.2.2.1/file2
Connecting to 10.2.2.1:80... connected.
HTTP request sent, awaiting response... 200 Ok
Length: 142144 (139K) [text/plain]
Saving to: 'file2.1'

file2.1 100%[=====] 138,81K --,-KB/s in 0,005s

2021-10-20 09:37:36 (26,9 MB/s) - 'file2.1' saved [142144/142144]

root@Portatil1:/tmp/pycore.40727/Portatil1.conf#
```

Figura 9: Transferências dos ficheiros por HTTP para Portatil1



```
vcmd
root@Grilo:/tmp/pycore.40727/Grilo.conf# wget http://10.2.2.1/file1
--2021-10-20 09:38:05-- http://10.2.2.1/file1
Connecting to 10.2.2.1:80... connected.
HTTP request sent, awaiting response... 200 Ok
Length: 224 [text/plain]
Saving to: 'file1.1'

file1.1 100%[=====] 224 --,-KB/s in 0s

2021-10-20 09:38:05 (20,7 MB/s) - 'file1.1' saved [224/224]

root@Grilo:/tmp/pycore.40727/Grilo.conf# wget http://10.2.2.1/file2
--2021-10-20 09:38:07-- http://10.2.2.1/file2
Connecting to 10.2.2.1:80... connected.
HTTP request sent, awaiting response... 200 Ok
Length: 142144 (139K) [text/plain]
Saving to: 'file2.1'

file2.1 100%[=====] 138,81K --,-KB/s in 0,02s

2021-10-20 09:38:07 (6,19 MB/s) - 'file2.1' saved [142144/142144]

root@Grilo:/tmp/pycore.40727/Grilo.conf#
```

Figura 8: Transferências dos ficheiros por HTTP para Grilo

4. Conclusão

Em suma, com este trabalho prático conseguimos aprender as diferenças entre protocolos da camada de transporte e protocolos da camada de aplicação, bem como identificar que protocolos da camada de transporte são usados pelas aplicações. Conseguimos também perceber as diferenças entre os dois protocolos da camada de transporte, TCP e UDP, e como influenciam as diferentes aplicações de transferência de ficheiros ao nível da camada de transporte, a sua eficiência, a sua complexidade e a sua segurança. Concluimos que o protocolo TCP é orientado à conexão enquanto o protocolo UDP não é orientado à conexão, em termos de segurança TCP é mais seguro para transferência de ficheiros uma vez que a verificação de erros é feita na camada de transporte enquanto com o protocolo UDP a verificação da maioria dos erros tem de ser feita na camada de aplicação, isto é, erros como a perda e duplicação de pacotes.

Com isto, as aplicações ao usarem um dos protocolos supramencionados, implementam o seu protocolo de aplicação, que tem como parâmetros a segurança, eficiência e complexidade. Assim, ao longo do projeto fomos capazes de comparar as diferentes aplicações no âmbito dos parâmetros acima referidos.

Numa primeira abordagem ao trabalho prático, mostrámos ter dificuldade sobre a matéria em geral, tal como o que eram protocolos da camada de transporte/aplicação, as transferências de pacotes entre cliente e servidor e como funcionavam as aplicações. Além disso, outro obstáculo que tivemos de superar foi a análise crítica na transferência de pacotes pelo *wireshark*, pois este mostrou ser fulcral para a boa realização do trabalho prático.