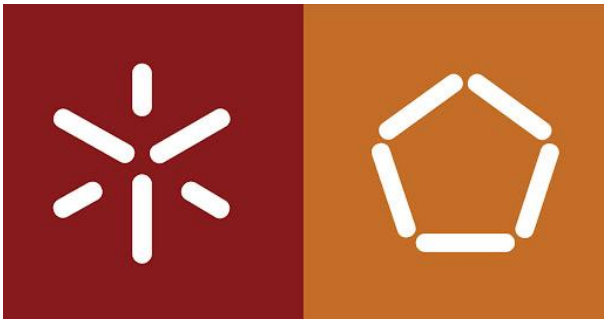


Universidade do Minho
Departamento de Informática
Licenciatura em Engenharia Informática

Comunicações por Computador

Trabalho Prático 1

Grupo Nº 53

Rui Monteiro (A93179)
Diogo Barbosa (A93184)
Joaquim Roque (A93310)

28 de outubro de 2021

Parte I

Questão 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.

Resposta:

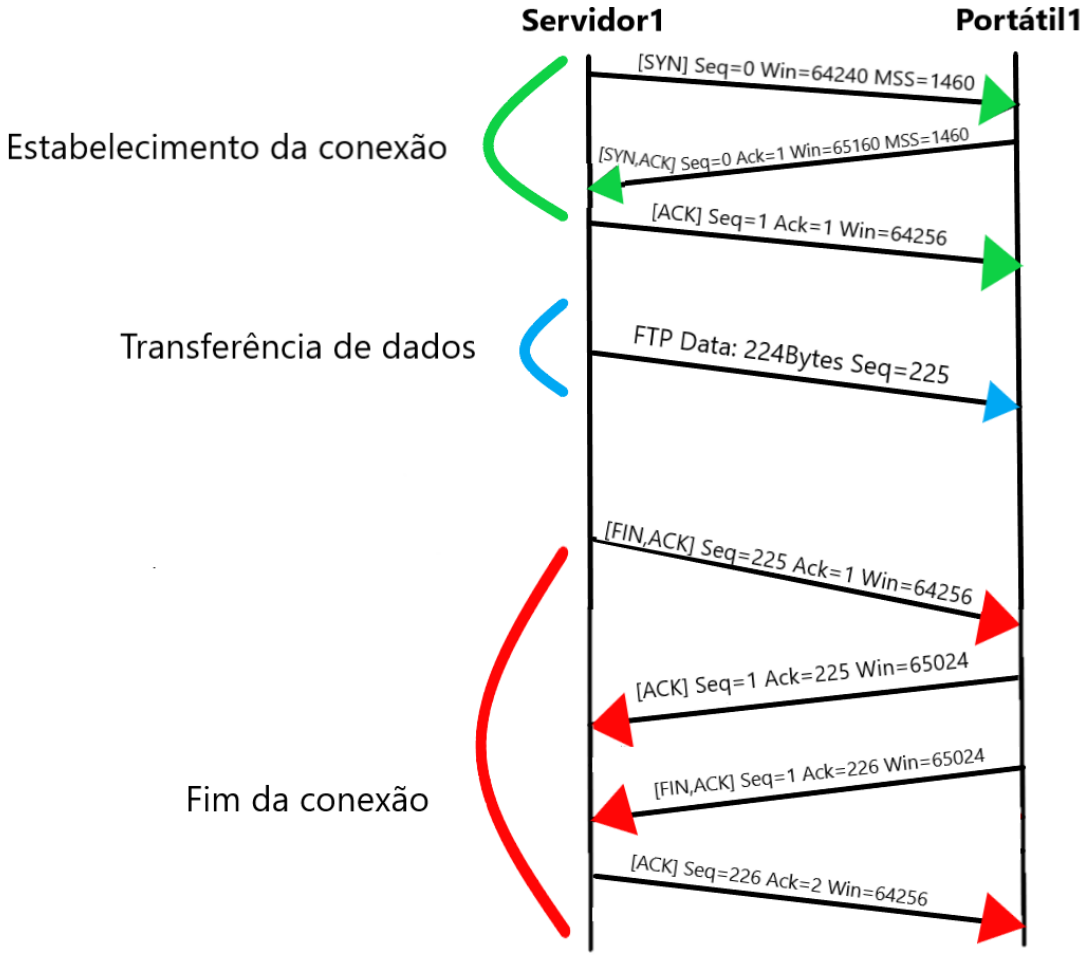
Nas aplicações que usam TCP como protocolo da camada de transporte, qualquer perda ou duplicação de pacotes afetaria o desempenho das aplicações visto que a mesma terá que requerer novamente o pacote perdido, sendo a camada de transporte a lidar com essas situações, tal como foi observado no *wireshark,* visto que o TCP é um protocolo de transporte fiável que procura garantir que todos os pacotes chegam ao seu destino, ou seja, para cada pacote é enviado um *acknowledgement* ao servidor que comprova o sucesso de envio. Quando existe uma retransmissão dos pacotes, leva a atrasos na transmissão dos mesmos, o que afeta o desempenho das aplicações de transferência de ficheiros negativamente.

Pelo contrário, o protocolo de transporte UDP não é orientado à conexão, logo não garante que todos os pacotes enviados chegam ao seu destino, ou seja, sempre que uma falha ocorre, o protocolo não requer um novo pedido para repor os dados perdidos, sendo que neste caso a camada que lida com as perdas e duplicações é a de aplicação. Conclui-se assim que o UDP não é um protocolo de transporte fiável, no entanto devido à sua complexidade reduzida acaba por ter melhor desempenho.

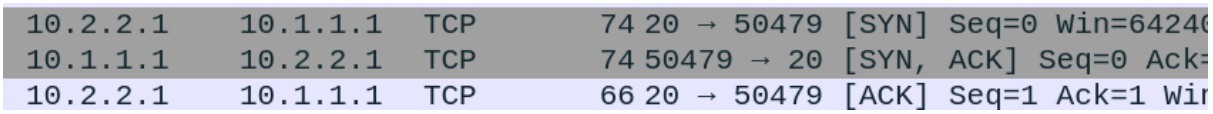
Questão 2

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.

Resposta:

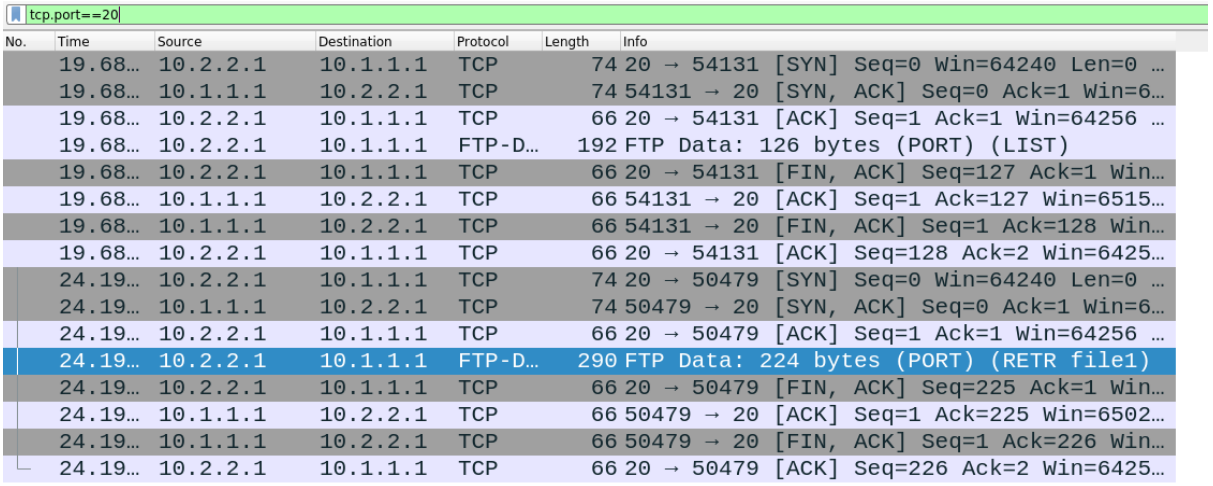


1. Diagrama Temporal - FTP



10.2.2.1	10.1.1.1	TCP	74	20 → 50479 [SYN] Seq=0 Win=64240
10.1.1.1	10.2.2.1	TCP	74	50479 → 20 [SYN, ACK] Seq=0 Ack=
10.2.2.1	10.1.1.1	TCP	66	20 → 50479 [ACK] Seq=1 Ack=1 Wi

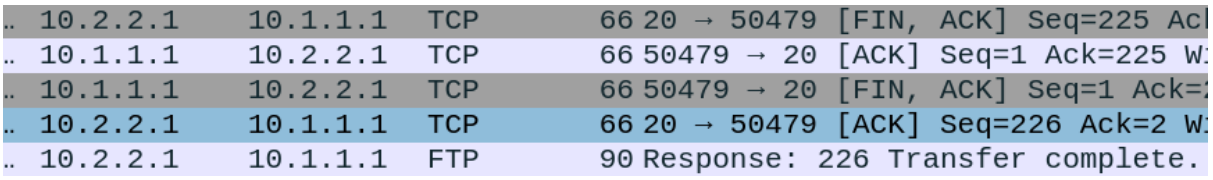
2. Início da conexão



tcp.port==20

No.	Time	Source	Destination	Protocol	Length	Info
	19.68…	10.2.2.1	10.1.1.1	TCP	74	20 → 54131 [SYN] Seq=0 Win=64240 Len=0 …
	19.68…	10.1.1.1	10.2.2.1	TCP	74	54131 → 20 [SYN, ACK] Seq=0 Ack=1 Win=6…
	19.68…	10.2.2.1	10.1.1.1	TCP	66	20 → 54131 [ACK] Seq=1 Ack=1 Win=64256 …
	19.68…	10.2.2.1	10.1.1.1	FTP-D…	192	FTP Data: 126 bytes (PORT) (LIST)
	19.68…	10.2.2.1	10.1.1.1	TCP	66	20 → 54131 [FIN, ACK] Seq=127 Ack=1 Win…
	19.68…	10.1.1.1	10.2.2.1	TCP	66	54131 → 20 [ACK] Seq=1 Ack=127 Win=6515…
	19.68…	10.1.1.1	10.2.2.1	TCP	66	54131 → 20 [FIN, ACK] Seq=1 Ack=128 Win…
	19.68…	10.2.2.1	10.1.1.1	TCP	66	20 → 54131 [ACK] Seq=128 Ack=2 Win=6425…
	24.19…	10.2.2.1	10.1.1.1	TCP	74	20 → 50479 [SYN] Seq=0 Win=64240 Len=0 …
	24.19…	10.1.1.1	10.2.2.1	TCP	74	50479 → 20 [SYN, ACK] Seq=0 Ack=1 Win=6…
	24.19…	10.2.2.1	10.1.1.1	TCP	66	20 → 50479 [ACK] Seq=1 Ack=1 Win=64256 …
	24.19…	10.2.2.1	10.1.1.1	FTP-D…	290	FTP Data: 224 bytes (PORT) (RETR file1)
	24.19…	10.2.2.1	10.1.1.1	TCP	66	20 → 50479 [FIN, ACK] Seq=225 Ack=1 Win…
	24.19…	10.1.1.1	10.2.2.1	TCP	66	50479 → 20 [ACK] Seq=1 Ack=225 Win=6502…
	24.19…	10.1.1.1	10.2.2.1	TCP	66	50479 → 20 [FIN, ACK] Seq=1 Ack=226 Win…
	24.19…	10.2.2.1	10.1.1.1	TCP	66	20 → 50479 [ACK] Seq=226 Ack=2 Win=6425…

3. Transferência de dados (FTP-Data)



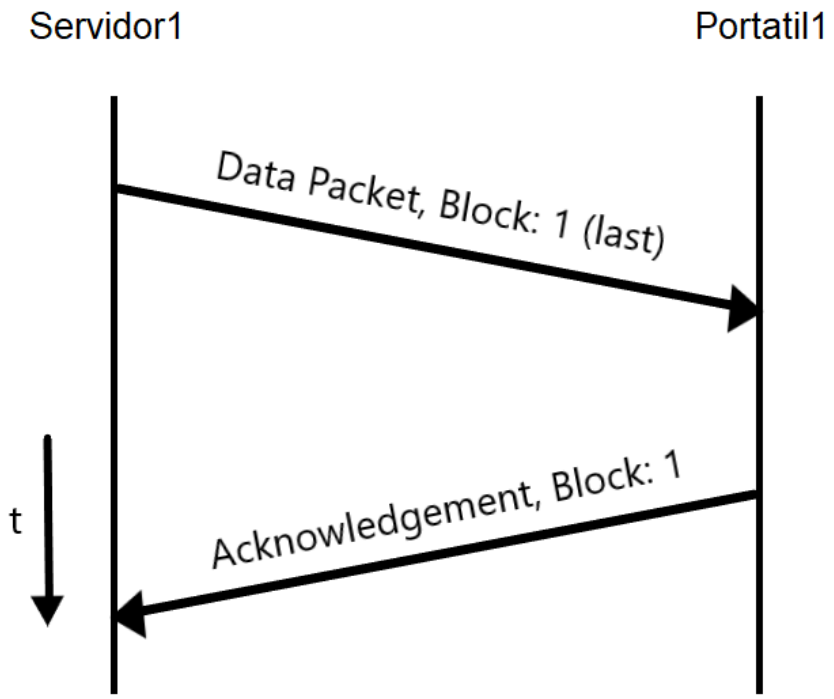
10.2.2.1	10.1.1.1	TCP	66	20 → 50479 [FIN, ACK] Seq=225 Ac
10.1.1.1	10.2.2.1	TCP	66	50479 → 20 [ACK] Seq=1 Ack=225 W
10.1.1.1	10.2.2.1	TCP	66	50479 → 20 [FIN, ACK] Seq=1 Ack=
10.2.2.1	10.1.1.1	TCP	66	20 → 50479 [ACK] Seq=226 Ack=2 W
10.2.2.1	10.1.1.1	FTP	90	Response: 226 Transfer complete.

4. Fim da conexão no wireshark

Questão 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.

Resposta:



5. Diagrama Temporal - TFTP

9.092…	10.1.1.1	10.2.2.1	TFTP	56	Read Request, File: file1, Transfer typ…
9.093…	10.2.2.1	10.1.1.1	TFTP	270	Data Packet, Block: 1 (last)
9.093…	10.1.1.1	10.2.2.1	TFTP	46	Acknowledgement, Block: 1

6. TFTP

Questão 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.

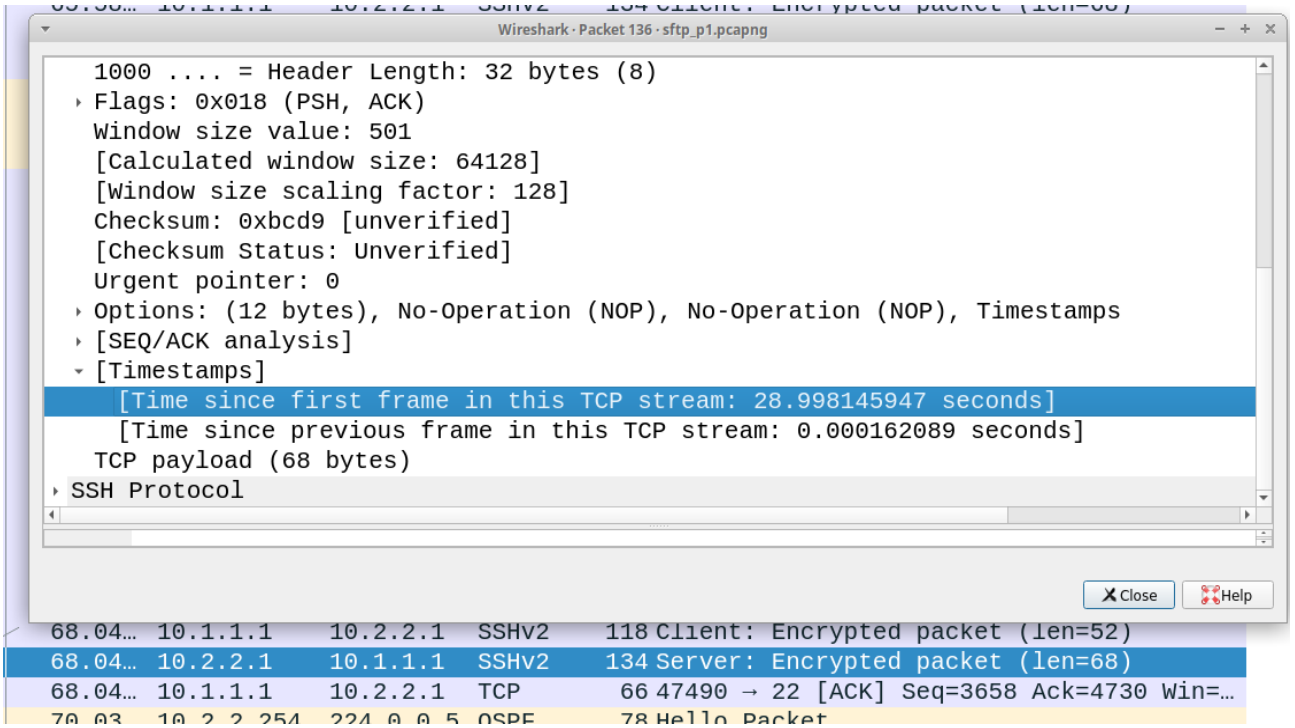
Resposta:

Usando SFTP para realizar a transferência dos ficheiros, verificamos que este faz uso de SSH, necessitando de autenticação por parte do cliente e recorrendo a encriptação dos pacotes, o que a torna numa aplicação mais segura, mas menos eficiente visto que acaba por ser mais complexa que o resto das aplicações, tendo um maior *overhead*. Utiliza TCP como protocolo da camada de transporte.

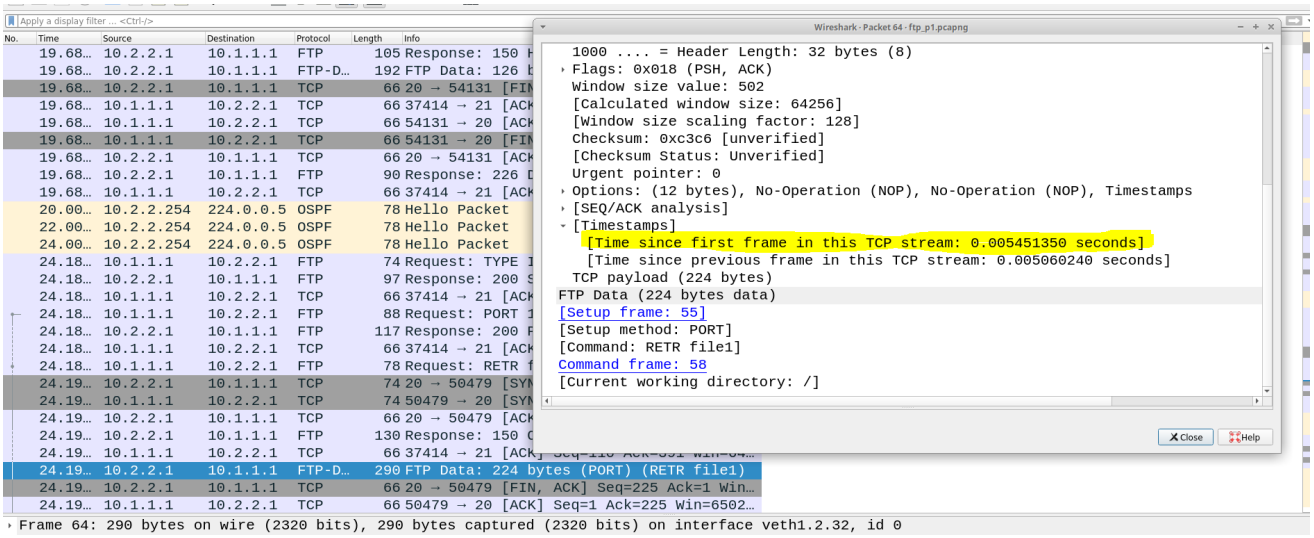
Analisando a transferência de ficheiros por FTP, à semelhança do SFTP utiliza TCP como protocolo da camada de transporte, no entanto não utiliza SSH como método de autenticação, o que a torna menos segura. No que toca à sua complexidade, é uma aplicação menos complexa para a transferência de ficheiros. Possui um *overhead* considerável, afetando de forma notável a sua eficiência.

Quanto à transferência de ficheiros por TFTP, como acontece com FTP, não implementa medidas de segurança adicionais e não dispõe de autenticação por parte do cliente. Utiliza UDP como protocolo da camada de transporte, logo não é fiável, no entanto isto contribui para o seu baixo *overhead*, que o torna mais eficiente. Há portanto um compromisso entre o protocolo da camada de transporte e o programador, em que "sacrificando-se" muito do tratamento de erros do TCP ganha-se em *performance*, ficando o programador encarregue de garantir o normal funcionamento da aplicação.

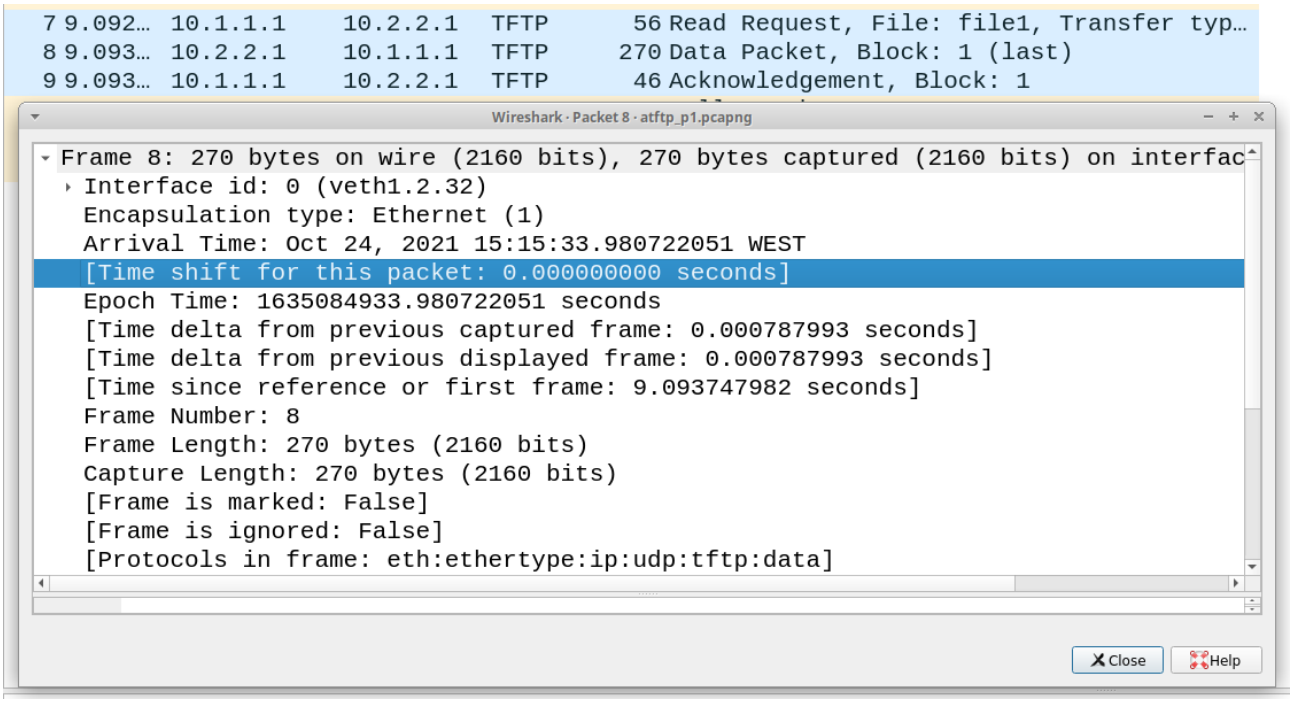
Finalmente, a transferência por HTTP revela ser bastante inseguro visto que as transferências de conteúdo são acessíveis a qualquer pessoa, para além de não requerer qualquer tipo de autenticação por parte do cliente. Utiliza o TCP como protocolo da camada de transporte.



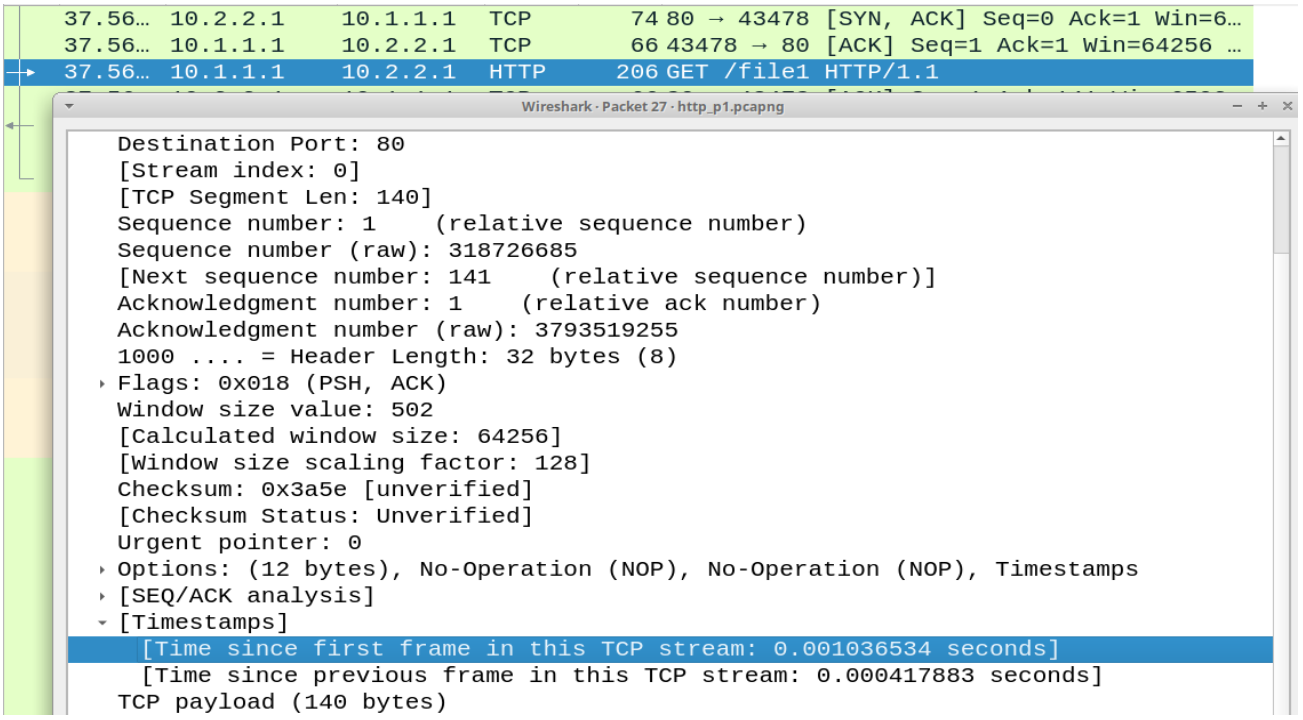
7. Timestamp SFTP



8. Timestamp para FTP



9. Timestamp para TFTP



10. Timestamp para HTTP

Parte II

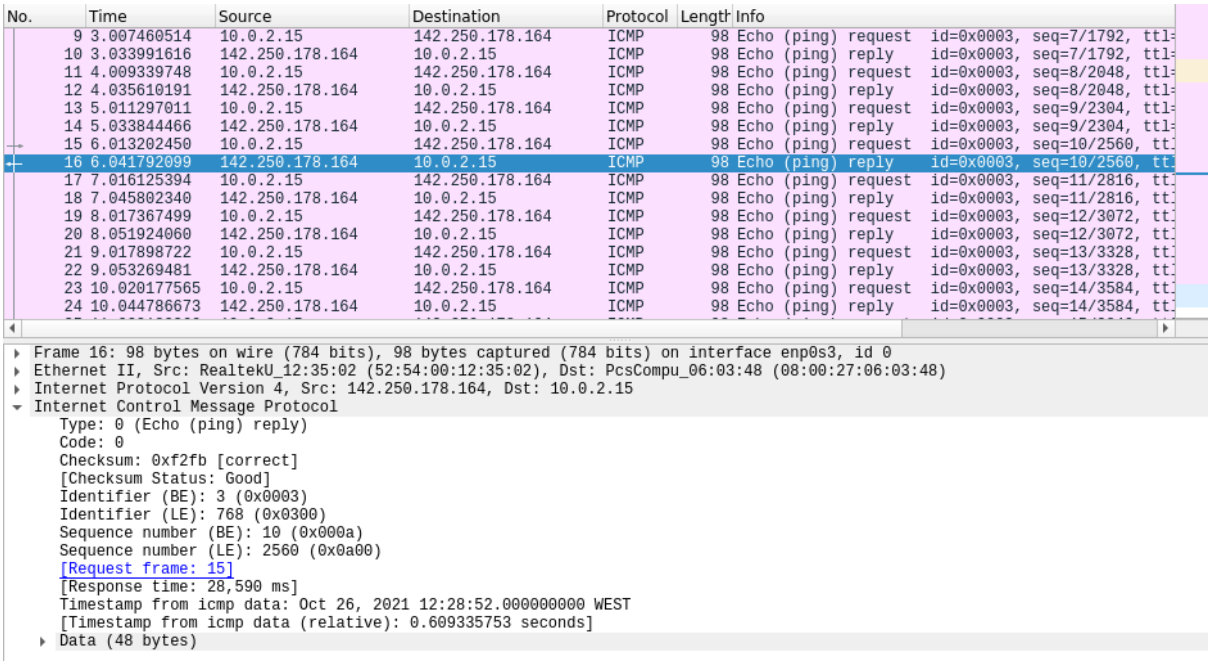
Questão 1

Com base na captura de pacotes feita, preencha a seguinte tabela, identificando para cada aplicação executada, qual o protocolo de aplicação, o protocolo de transporte, porta de atendimento e *overhead* de transporte.

Resposta:

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	-	-	-	-
traceroute	-	udp	*random*	8
telnet	telnet	tcp	23	20
ftp	ftp	tcp	21	20
tftp	tftp	udp	69	8
http(browser)	http	tcp	80	20
nslookup	dns	udp	53	8
ssh	sshv2	tcp	22	20

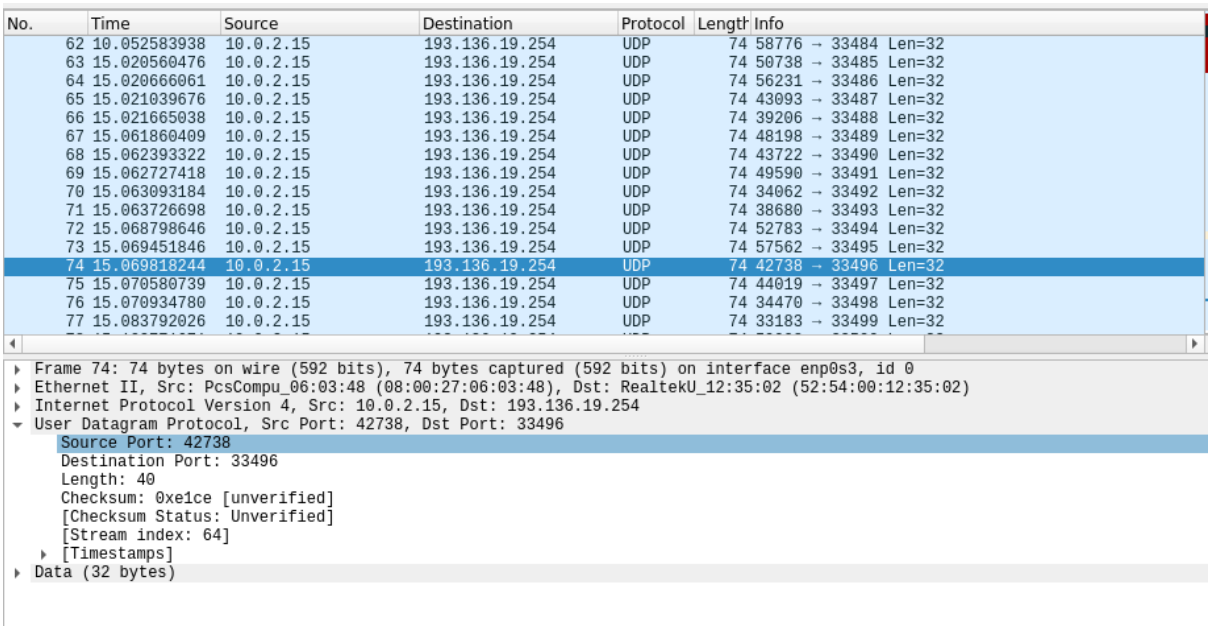
Nota: No que toca ao *overhead,* de salientar que o valor só representa a informação extraordinária dos pacotes de dados. Para ter uma noção do valor total do mesmo, devemos multiplicar esse valor pelo número de pacotes. Se o protocolo usado for o TCP, deve-se considerar ainda as fases de estabelecimento e terminação de conexão, bem como todos os *acknowledgements*.



No.	Time	Source	Destination	Protocol	Length	Info
9	3.007460514	10.0.2.15	142.250.178.164	ICMP	98	Echo (ping) request id=0x0003, seq=7/1792, ttl=
10	3.033991616	142.250.178.164	10.0.2.15	ICMP	98	Echo (ping) reply id=0x0003, seq=7/1792, ttl=
11	4.009339748	10.0.2.15	142.250.178.164	ICMP	98	Echo (ping) request id=0x0003, seq=8/2048, ttl=
12	4.035610191	142.250.178.164	10.0.2.15	ICMP	98	Echo (ping) reply id=0x0003, seq=8/2048, ttl=
13	5.011297011	10.0.2.15	142.250.178.164	ICMP	98	Echo (ping) request id=0x0003, seq=9/2304, ttl=
14	5.033844466	142.250.178.164	10.0.2.15	ICMP	98	Echo (ping) reply id=0x0003, seq=9/2304, ttl=
15	6.013202450	10.0.2.15	142.250.178.164	ICMP	98	Echo (ping) request id=0x0003, seq=10/2560, tt
16	6.041792099	142.250.178.164	10.0.2.15	ICMP	98	Echo (ping) reply id=0x0003, seq=10/2560, tt
17	7.016125394	10.0.2.15	142.250.178.164	ICMP	98	Echo (ping) request id=0x0003, seq=11/2816, tt
18	7.045802340	142.250.178.164	10.0.2.15	ICMP	98	Echo (ping) reply id=0x0003, seq=11/2816, tt
19	8.017367499	10.0.2.15	142.250.178.164	ICMP	98	Echo (ping) request id=0x0003, seq=12/3072, tt
20	8.051924060	142.250.178.164	10.0.2.15	ICMP	98	Echo (ping) reply id=0x0003, seq=12/3072, tt
21	9.017898722	10.0.2.15	142.250.178.164	ICMP	98	Echo (ping) request id=0x0003, seq=13/3328, tt
22	9.053269481	142.250.178.164	10.0.2.15	ICMP	98	Echo (ping) reply id=0x0003, seq=13/3328, tt
23	10.020177565	10.0.2.15	142.250.178.164	ICMP	98	Echo (ping) request id=0x0003, seq=14/3584, tt
24	10.044786673	142.250.178.164	10.0.2.15	ICMP	98	Echo (ping) reply id=0x0003, seq=14/3584, tt

```
▸ Frame 16: 98 bytes on wire (784 bits), 98 bytes captured (784 bits) on interface enp0s3, id 0
▸ Ethernet II, Src: RealtekU_12:35:02 (52:54:00:12:35:02), Dst: PcsCompu_06:03:48 (08:00:27:06:03:48)
▸ Internet Protocol Version 4, Src: 142.250.178.164, Dst: 10.0.2.15
▾ Internet Control Message Protocol
    Type: 0 (Echo (ping) reply)
    Code: 0
    Checksum: 0xf2fb [correct]
    [Checksum Status: Good]
    Identifier (BE): 3 (0x0003)
    Identifier (LE): 768 (0x0300)
    Sequence number (BE): 10 (0x000a)
    Sequence number (LE): 2560 (0x0a00)
    [Request frame: 15]
    [Response time: 28,590 ms]
    Timestamp from icmp data: Oct 26, 2021 12:28:52.000000000 WEST
    [Timestamp from icmp data (relative): 0.609335753 seconds]
  ▸ Data (48 bytes)
```

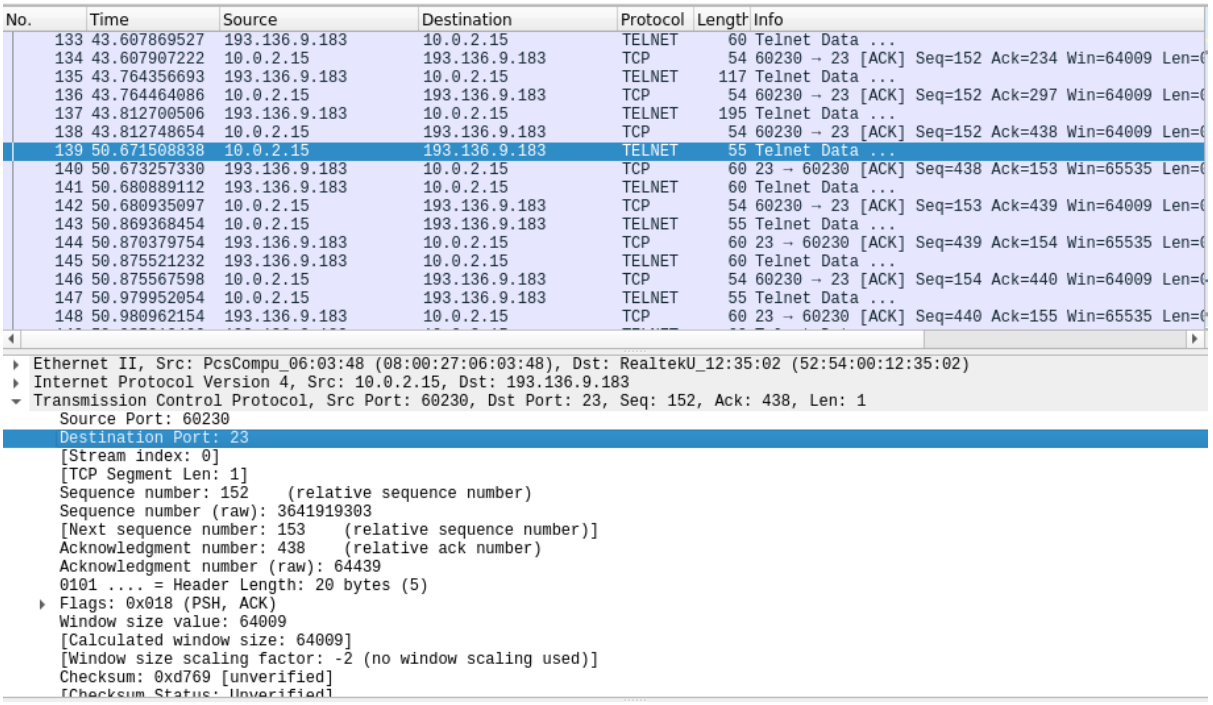
1. ping



No.	Time	Source	Destination	Protocol	Length	Info
62	10.052583938	10.0.2.15	193.136.19.254	UDP	74	58776 → 33484 Len=32
63	15.020560476	10.0.2.15	193.136.19.254	UDP	74	50738 → 33485 Len=32
64	15.020666061	10.0.2.15	193.136.19.254	UDP	74	56231 → 33486 Len=32
65	15.021039676	10.0.2.15	193.136.19.254	UDP	74	43093 → 33487 Len=32
66	15.021665038	10.0.2.15	193.136.19.254	UDP	74	39206 → 33488 Len=32
67	15.061860409	10.0.2.15	193.136.19.254	UDP	74	48198 → 33489 Len=32
68	15.062393322	10.0.2.15	193.136.19.254	UDP	74	43722 → 33490 Len=32
69	15.062727418	10.0.2.15	193.136.19.254	UDP	74	49590 → 33491 Len=32
70	15.063093184	10.0.2.15	193.136.19.254	UDP	74	34062 → 33492 Len=32
71	15.063726698	10.0.2.15	193.136.19.254	UDP	74	38680 → 33493 Len=32
72	15.068798646	10.0.2.15	193.136.19.254	UDP	74	52783 → 33494 Len=32
73	15.069451846	10.0.2.15	193.136.19.254	UDP	74	57562 → 33495 Len=32
74	15.069818244	10.0.2.15	193.136.19.254	UDP	74	42738 → 33496 Len=32
75	15.070580739	10.0.2.15	193.136.19.254	UDP	74	44019 → 33497 Len=32
76	15.070934780	10.0.2.15	193.136.19.254	UDP	74	34470 → 33498 Len=32
77	15.083792026	10.0.2.15	193.136.19.254	UDP	74	33183 → 33499 Len=32

```
▸ Frame 74: 74 bytes on wire (592 bits), 74 bytes captured (592 bits) on interface enp0s3, id 0
▸ Ethernet II, Src: PcsCompu_06:03:48 (08:00:27:06:03:48), Dst: RealtekU_12:35:02 (52:54:00:12:35:02)
▸ Internet Protocol Version 4, Src: 10.0.2.15, Dst: 193.136.19.254
▾ User Datagram Protocol, Src Port: 42738, Dst Port: 33496
    Source Port: 42738
    Destination Port: 33496
    Length: 40
    Checksum: 0xe1ce [unverified]
    [Checksum Status: Unverified]
    [Stream index: 64]
  ▸ [Timestamps]
▸ Data (32 bytes)
```

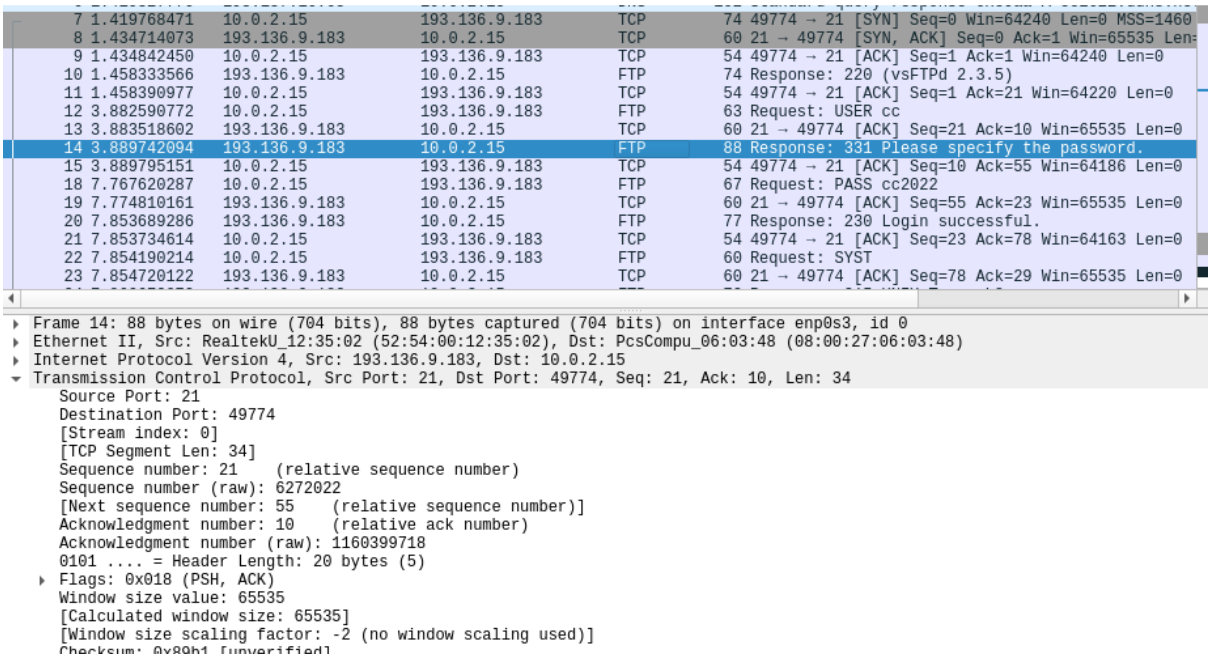
2. traceroute



No.	Time	Source	Destination	Protocol	Length	Info
133	43.607869527	193.136.9.183	10.0.2.15	TELNET	60	Telnet Data ...
134	43.607907222	10.0.2.15	193.136.9.183	TCP	54	60230 → 23 [ACK] Seq=152 Ack=234 Win=64009 Len=0
135	43.764356693	193.136.9.183	10.0.2.15	TELNET	117	Telnet Data ...
136	43.764464086	10.0.2.15	193.136.9.183	TCP	54	60230 → 23 [ACK] Seq=152 Ack=297 Win=64009 Len=0
137	43.812700506	193.136.9.183	10.0.2.15	TELNET	195	Telnet Data ...
138	43.812748654	10.0.2.15	193.136.9.183	TCP	54	60230 → 23 [ACK] Seq=152 Ack=438 Win=64009 Len=0
139	50.671508838	10.0.2.15	193.136.9.183	TELNET	55	Telnet Data ...
140	50.673257330	193.136.9.183	10.0.2.15	TCP	60	23 → 60230 [ACK] Seq=438 Ack=153 Win=65535 Len=0
141	50.680889112	193.136.9.183	10.0.2.15	TELNET	60	Telnet Data ...
142	50.680935097	10.0.2.15	193.136.9.183	TCP	54	60230 → 23 [ACK] Seq=153 Ack=439 Win=64009 Len=0
143	50.869368454	10.0.2.15	193.136.9.183	TELNET	55	Telnet Data ...
144	50.870379754	193.136.9.183	10.0.2.15	TCP	60	23 → 60230 [ACK] Seq=439 Ack=154 Win=65535 Len=0
145	50.875521232	193.136.9.183	10.0.2.15	TELNET	60	Telnet Data ...
146	50.875567598	10.0.2.15	193.136.9.183	TCP	54	60230 → 23 [ACK] Seq=154 Ack=440 Win=64009 Len=0
147	50.979952054	10.0.2.15	193.136.9.183	TELNET	55	Telnet Data ...
148	50.980962154	193.136.9.183	10.0.2.15	TCP	60	23 → 60230 [ACK] Seq=440 Ack=155 Win=65535 Len=0

```
▸ Ethernet II, Src: PcsCompu_06:03:48 (08:00:27:06:03:48), Dst: RealtekU_12:35:02 (52:54:00:12:35:02)
▸ Internet Protocol Version 4, Src: 10.0.2.15, Dst: 193.136.9.183
▾ Transmission Control Protocol, Src Port: 60230, Dst Port: 23, Seq: 152, Ack: 438, Len: 1
    Source Port: 60230
    Destination Port: 23
    [Stream index: 0]
    [TCP Segment Len: 1]
    Sequence number: 152    (relative sequence number)
    Sequence number (raw): 3641919303
    [Next sequence number: 153    (relative sequence number)]
    Acknowledgment number: 438    (relative ack number)
    Acknowledgment number (raw): 64439
    0101 .... = Header Length: 20 bytes (5)
  ▸ Flags: 0x018 (PSH, ACK)
    Window size value: 64009
    [Calculated window size: 64009]
    [Window size scaling factor: -2 (no window scaling used)]
    Checksum: 0xd769 [unverified]
    [Checksum Status: Unverified]
```

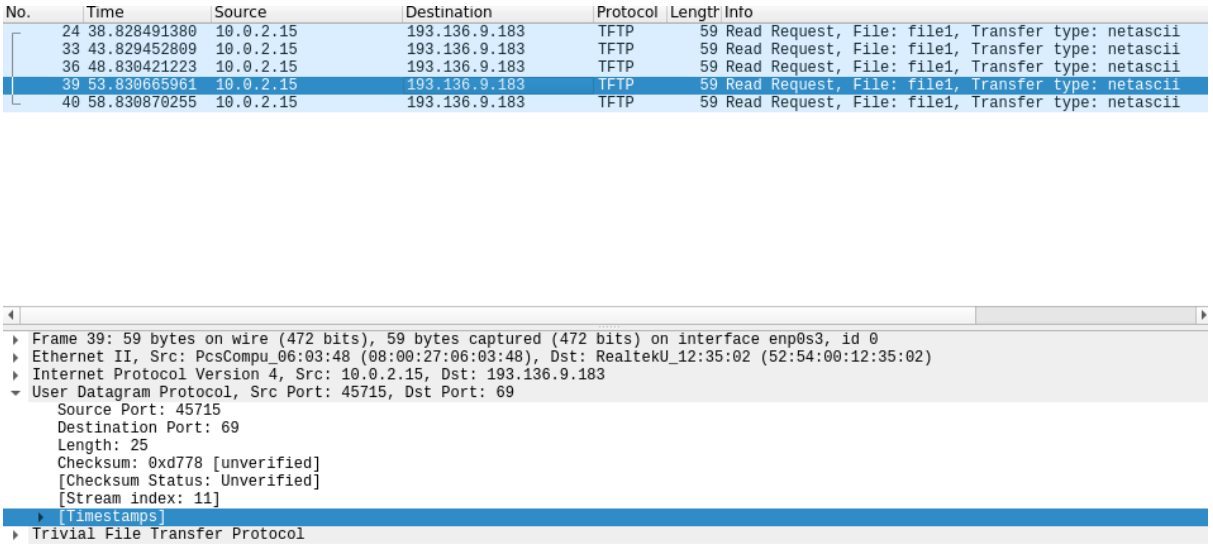
3. telnet



7	1.419768471	10.0.2.15	193.136.9.183	TCP	74	49774 → 21 [SYN] Seq=0 Win=64240 Len=0 MSS=1460
8	1.434714073	193.136.9.183	10.0.2.15	TCP	60	21 → 49774 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0
9	1.434842450	10.0.2.15	193.136.9.183	TCP	54	49774 → 21 [ACK] Seq=1 Ack=1 Win=64240 Len=0
10	1.458333566	193.136.9.183	10.0.2.15	FTP	74	Response: 220 (vsFTPd 2.3.5)
11	1.458390977	10.0.2.15	193.136.9.183	TCP	54	49774 → 21 [ACK] Seq=1 Ack=21 Win=64220 Len=0
12	3.882590772	10.0.2.15	193.136.9.183	FTP	63	Request: USER cc
13	3.883518602	193.136.9.183	10.0.2.15	TCP	60	21 → 49774 [ACK] Seq=21 Ack=10 Win=65535 Len=0
14	3.889742094	193.136.9.183	10.0.2.15	FTP	88	Response: 331 Please specify the password.
15	3.889795151	10.0.2.15	193.136.9.183	TCP	54	49774 → 21 [ACK] Seq=10 Ack=55 Win=64186 Len=0
18	7.767620287	10.0.2.15	193.136.9.183	FTP	67	Request: PASS cc2022
19	7.774810161	193.136.9.183	10.0.2.15	TCP	60	21 → 49774 [ACK] Seq=55 Ack=23 Win=65535 Len=0
20	7.853689286	193.136.9.183	10.0.2.15	FTP	77	Response: 230 Login successful.
21	7.853734614	10.0.2.15	193.136.9.183	TCP	54	49774 → 21 [ACK] Seq=23 Ack=78 Win=64163 Len=0
22	7.854190214	10.0.2.15	193.136.9.183	FTP	60	Request: SYST
23	7.854720122	193.136.9.183	10.0.2.15	TCP	60	21 → 49774 [ACK] Seq=78 Ack=29 Win=65535 Len=0

```
▸ Frame 14: 88 bytes on wire (704 bits), 88 bytes captured (704 bits) on interface enp0s3, id 0
▸ Ethernet II, Src: RealtekU_12:35:02 (52:54:00:12:35:02), Dst: PcsCompu_06:03:48 (08:00:27:06:03:48)
▸ Internet Protocol Version 4, Src: 193.136.9.183, Dst: 10.0.2.15
▾ Transmission Control Protocol, Src Port: 21, Dst Port: 49774, Seq: 21, Ack: 10, Len: 34
    Source Port: 21
    Destination Port: 49774
    [Stream index: 0]
    [TCP Segment Len: 34]
    Sequence number: 21    (relative sequence number)
    Sequence number (raw): 6272022
    [Next sequence number: 55    (relative sequence number)]
    Acknowledgment number: 10    (relative ack number)
    Acknowledgment number (raw): 1160399718
    0101 .... = Header Length: 20 bytes (5)
  ▸ Flags: 0x018 (PSH, ACK)
    Window size value: 65535
    [Calculated window size: 65535]
    [Window size scaling factor: -2 (no window scaling used)]
    Checksum: 0x89b1 [unverified]
```

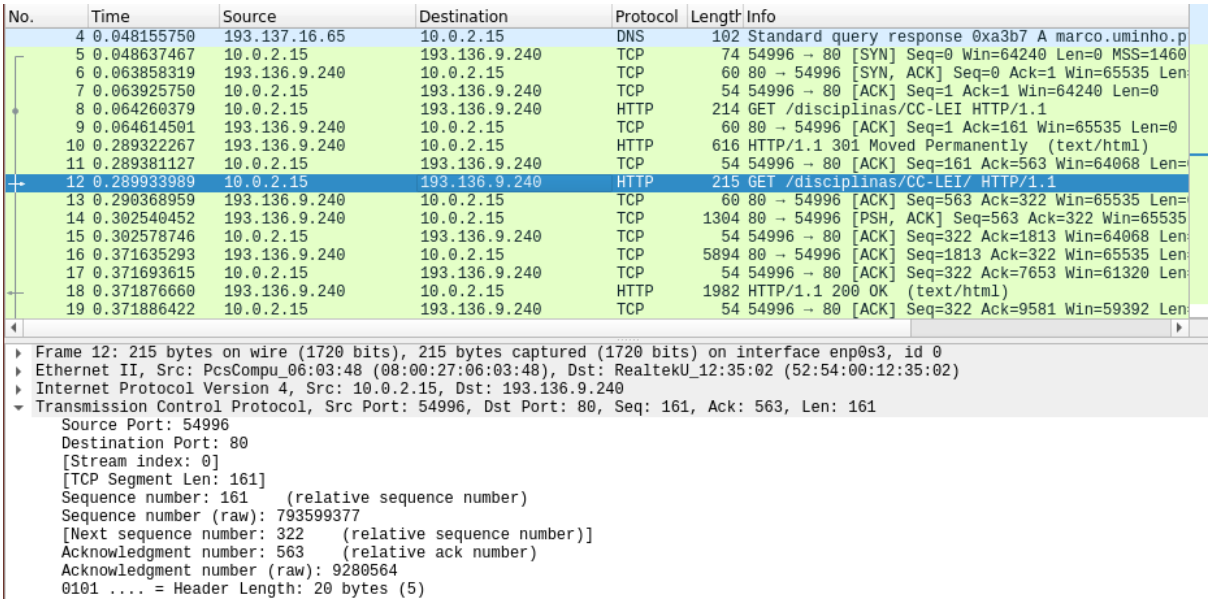
4. ftp



No.	Time	Source	Destination	Protocol	Length	Info
24	38.828491380	10.0.2.15	193.136.9.183	TFTP	59	Read Request, File: file1, Transfer type: netascii
33	43.829452809	10.0.2.15	193.136.9.183	TFTP	59	Read Request, File: file1, Transfer type: netascii
36	48.830421223	10.0.2.15	193.136.9.183	TFTP	59	Read Request, File: file1, Transfer type: netascii
39	53.830665961	10.0.2.15	193.136.9.183	TFTP	59	Read Request, File: file1, Transfer type: netascii
40	58.830870255	10.0.2.15	193.136.9.183	TFTP	59	Read Request, File: file1, Transfer type: netascii

```
▸ Frame 39: 59 bytes on wire (472 bits), 59 bytes captured (472 bits) on interface enp0s3, id 0
▸ Ethernet II, Src: PcsCompu_06:03:48 (08:00:27:06:03:48), Dst: RealtekU_12:35:02 (52:54:00:12:35:02)
▸ Internet Protocol Version 4, Src: 10.0.2.15, Dst: 193.136.9.183
▾ User Datagram Protocol, Src Port: 45715, Dst Port: 69
    Source Port: 45715
    Destination Port: 69
    Length: 25
    Checksum: 0xd778 [unverified]
    [Checksum Status: Unverified]
    [Stream index: 11]
  ▸ [Timestamps]
▸ Trivial File Transfer Protocol
```

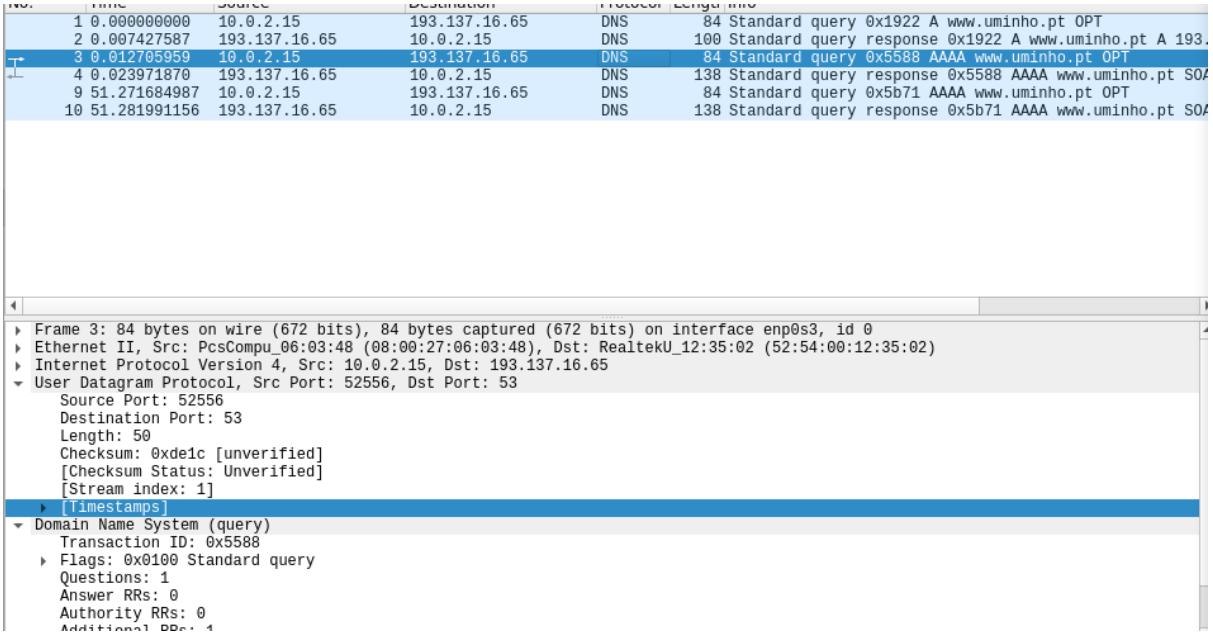
5. tftp



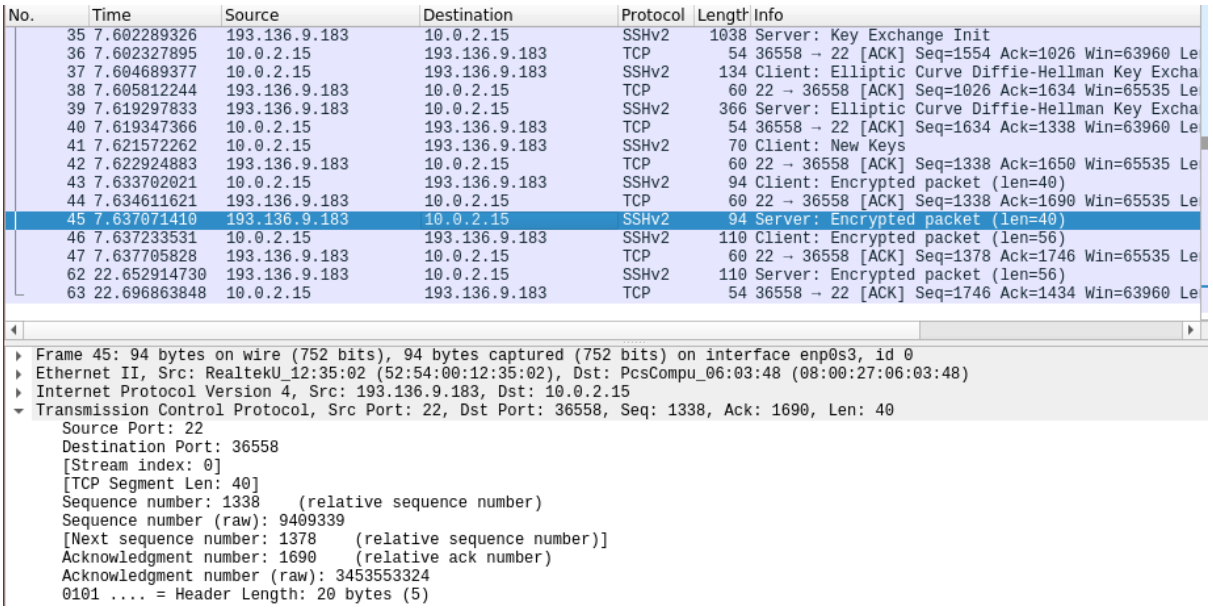
No.	Time	Source	Destination	Protocol	Length	Info
4	0.048155750	193.137.16.65	10.0.2.15	DNS	102	Standard query response 0xa3b7 A marco.uminho.p
5	0.048637467	10.0.2.15	193.136.9.240	TCP	74	54996 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460
6	0.063858319	193.136.9.240	10.0.2.15	TCP	60	80 → 54996 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len
7	0.063925750	10.0.2.15	193.136.9.240	TCP	54	54996 → 80 [ACK] Seq=1 Ack=1 Win=64240 Len=0
8	0.064260379	10.0.2.15	193.136.9.240	HTTP	214	GET /disciplinas/CC-LEI HTTP/1.1
9	0.064614501	193.136.9.240	10.0.2.15	TCP	60	80 → 54996 [ACK] Seq=1 Ack=161 Win=65535 Len=0
10	0.289322267	193.136.9.240	10.0.2.15	HTTP	616	HTTP/1.1 301 Moved Permanently (text/html)
11	0.289381127	10.0.2.15	193.136.9.240	TCP	54	54996 → 80 [ACK] Seq=161 Ack=563 Win=64068 Len=
12	0.289933989	10.0.2.15	193.136.9.240	HTTP	215	GET /disciplinas/CC-LEI/ HTTP/1.1
13	0.290368959	193.136.9.240	10.0.2.15	TCP	60	80 → 54996 [ACK] Seq=563 Ack=322 Win=65535 Len=
14	0.302540452	193.136.9.240	10.0.2.15	TCP	1304	80 → 54996 [PSH, ACK] Seq=563 Ack=322 Win=65535
15	0.302578746	10.0.2.15	193.136.9.240	TCP	54	54996 → 80 [ACK] Seq=322 Ack=1813 Win=64068 Len
16	0.371635293	193.136.9.240	10.0.2.15	TCP	5894	80 → 54996 [ACK] Seq=1813 Ack=322 Win=65535 Len
17	0.371693615	10.0.2.15	193.136.9.240	TCP	54	54996 → 80 [ACK] Seq=322 Ack=7653 Win=61320 Len
18	0.371876660	193.136.9.240	10.0.2.15	HTTP	1982	HTTP/1.1 200 OK (text/html)
19	0.371886422	10.0.2.15	193.136.9.240	TCP	54	54996 → 80 [ACK] Seq=322 Ack=9581 Win=59392 Len

```
▸ Frame 12: 215 bytes on wire (1720 bits), 215 bytes captured (1720 bits) on interface enp0s3, id 0
▸ Ethernet II, Src: PcsCompu_06:03:48 (08:00:27:06:03:48), Dst: RealtekU_12:35:02 (52:54:00:12:35:02)
▸ Internet Protocol Version 4, Src: 10.0.2.15, Dst: 193.136.9.240
▾ Transmission Control Protocol, Src Port: 54996, Dst Port: 80, Seq: 161, Ack: 563, Len: 161
    Source Port: 54996
    Destination Port: 80
    [Stream index: 0]
    [TCP Segment Len: 161]
    Sequence number: 161    (relative sequence number)
    Sequence number (raw): 793599377
    [Next sequence number: 322    (relative sequence number)]
    Acknowledgment number: 563    (relative ack number)
    Acknowledgment number (raw): 9280564
    0101 .... = Header Length: 20 bytes (5)
```

6. http



7. nslookup



8. ssh

Parte III

Conclusão

Na elaboração deste trabalho, consolidamos os conteúdos lecionados nas aulas teóricas, nomeadamente distinção das várias camadas e suas funções, aprofundamento do conhecimento da metodologia de funcionamento dos protocolos da camada de transporte e de que forma estes se relacionam com o desenvolvimento de *software*.