

UNIVERSIDADE DO MINHO

LICENCIATURA EM ENGENHARIA INFORMÁTICA

Comunicação por Computadores - TP2
Grupo 27

Luís Faria (A93209) Rui Moreira (A93232)
Gonçalo Braz (A93178)

Ano Lectivo 2021/2022



Conteúdo

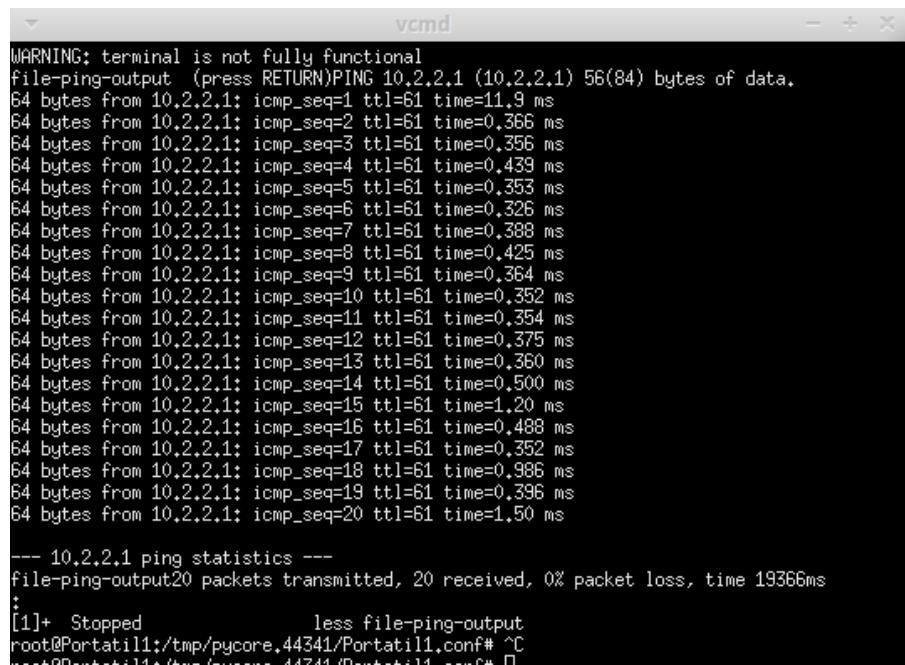
1	Questões e Respostas	3
1.1	Parte 1	3
1.1.1	Exercício 1	3
1.1.2	Exercício 2	5
1.1.3	Exercício 3	5
1.1.4	Exercício 4	6
1.2	Parte 2	10
2	Conclusões	15

Capítulo 1

Questões e Respostas

1.1 Parte 1

1.1.1 Exercício 1



```
WARNING: terminal is not fully functional
file-ping-output (press RETURN)PING 10.2.2.1 (10.2.2.1) 56(84) bytes of data.
64 bytes from 10.2.2.1: icmp_seq=1 ttl=61 time=11.9 ms
64 bytes from 10.2.2.1: icmp_seq=2 ttl=61 time=0.366 ms
64 bytes from 10.2.2.1: icmp_seq=3 ttl=61 time=0.356 ms
64 bytes from 10.2.2.1: icmp_seq=4 ttl=61 time=0.439 ms
64 bytes from 10.2.2.1: icmp_seq=5 ttl=61 time=0.353 ms
64 bytes from 10.2.2.1: icmp_seq=6 ttl=61 time=0.326 ms
64 bytes from 10.2.2.1: icmp_seq=7 ttl=61 time=0.388 ms
64 bytes from 10.2.2.1: icmp_seq=8 ttl=61 time=0.425 ms
64 bytes from 10.2.2.1: icmp_seq=9 ttl=61 time=0.364 ms
64 bytes from 10.2.2.1: icmp_seq=10 ttl=61 time=0.352 ms
64 bytes from 10.2.2.1: icmp_seq=11 ttl=61 time=0.354 ms
64 bytes from 10.2.2.1: icmp_seq=12 ttl=61 time=0.375 ms
64 bytes from 10.2.2.1: icmp_seq=13 ttl=61 time=0.360 ms
64 bytes from 10.2.2.1: icmp_seq=14 ttl=61 time=0.500 ms
64 bytes from 10.2.2.1: icmp_seq=15 ttl=61 time=1.20 ms
64 bytes from 10.2.2.1: icmp_seq=16 ttl=61 time=0.488 ms
64 bytes from 10.2.2.1: icmp_seq=17 ttl=61 time=0.352 ms
64 bytes from 10.2.2.1: icmp_seq=18 ttl=61 time=0.986 ms
64 bytes from 10.2.2.1: icmp_seq=19 ttl=61 time=0.396 ms
64 bytes from 10.2.2.1: icmp_seq=20 ttl=61 time=1.50 ms

--- 10.2.2.1 ping statistics ---
file-ping-output20 packets transmitted, 20 received, 0% packet loss, time 19366ms
:
[1]+ Stopped less file-ping-output
root@Portatil1:/tmp/pycore.44341/Portatil1.conf# ^C
```

Figura 1.1: Ping Portátil 1

```

PING 10.2.2.1 (10.2.2.1) 56(84) bytes of data.
64 bytes from 10.2.2.1: icmp_seq=1 ttl=61 time=10.8 ms
64 bytes from 10.2.2.1: icmp_seq=2 ttl=61 time=5.36 ms
64 bytes from 10.2.2.1: icmp_seq=3 ttl=61 time=5.47 ms
64 bytes from 10.2.2.1: icmp_seq=4 ttl=61 time=5.28 ms
64 bytes from 10.2.2.1: icmp_seq=5 ttl=61 time=5.50 ms
64 bytes from 10.2.2.1: icmp_seq=6 ttl=61 time=5.42 ms
64 bytes from 10.2.2.1: icmp_seq=7 ttl=61 time=8.97 ms
64 bytes from 10.2.2.1: icmp_seq=8 ttl=61 time=5.50 ms
64 bytes from 10.2.2.1: icmp_seq=8 ttl=61 time=5.51 ms (DUP!)
64 bytes from 10.2.2.1: icmp_seq=9 ttl=61 time=5.34 ms
64 bytes from 10.2.2.1: icmp_seq=10 ttl=61 time=5.29 ms
64 bytes from 10.2.2.1: icmp_seq=12 ttl=61 time=6.60 ms
64 bytes from 10.2.2.1: icmp_seq=13 ttl=61 time=5.58 ms
64 bytes from 10.2.2.1: icmp_seq=13 ttl=61 time=5.59 ms (DUP!)
64 bytes from 10.2.2.1: icmp_seq=14 ttl=61 time=5.40 ms
64 bytes from 10.2.2.1: icmp_seq=15 ttl=61 time=8.54 ms
64 bytes from 10.2.2.1: icmp_seq=16 ttl=61 time=5.38 ms
64 bytes from 10.2.2.1: icmp_seq=17 ttl=61 time=5.42 ms
64 bytes from 10.2.2.1: icmp_seq=18 ttl=61 time=5.33 ms
64 bytes from 10.2.2.1: icmp_seq=19 ttl=61 time=5.46 ms
64 bytes from 10.2.2.1: icmp_seq=20 ttl=61 time=6.28 ms
64 bytes from 10.2.2.1: icmp_seq=20 ttl=61 time=6.28 ms (DUP!)

--- 10.2.2.1 ping statistics ---
20 packets transmitted, 19 received, +3 duplicates, 5% packet loss, time 19053ms
rtt min/avg/max/mdev = 5.279/6.106/10.833/1.418 ms
root@Grilo:/tmp/pycore,44341/Grilo.conf# 

```

Figura 1.2: Ping Grilo

Observando a figura 1.2, podemos ver que existiu 5% de perdas de pacotes e 3 duplicações. A camada que lidou com estas perdas e duplicações foi a camada de transporte. Existem dois principais protocolos de transporte que tratam este problema de diferente forma, sendo estes o TCP e o UDP.

O primeiro garante que todos os pacotes são enviados na ordem correta e sem erros. Para isto são trocados vários pacotes durante a transferência de dados, o que acaba por exigir mais da rede. Quando nos encontramos numa rede de menor qualidade é provável que se percam pacotes.

O UDP por não fazer esta verificação, exigindo menos da rede, é de certa forma mais eficiente, sendo um protocolo menos complexo que o TCP. Esta parte de garantir que os dados são enviados/recebidos corretamente deve ser feita pela aplicação.

1.1.2 Exercício 2

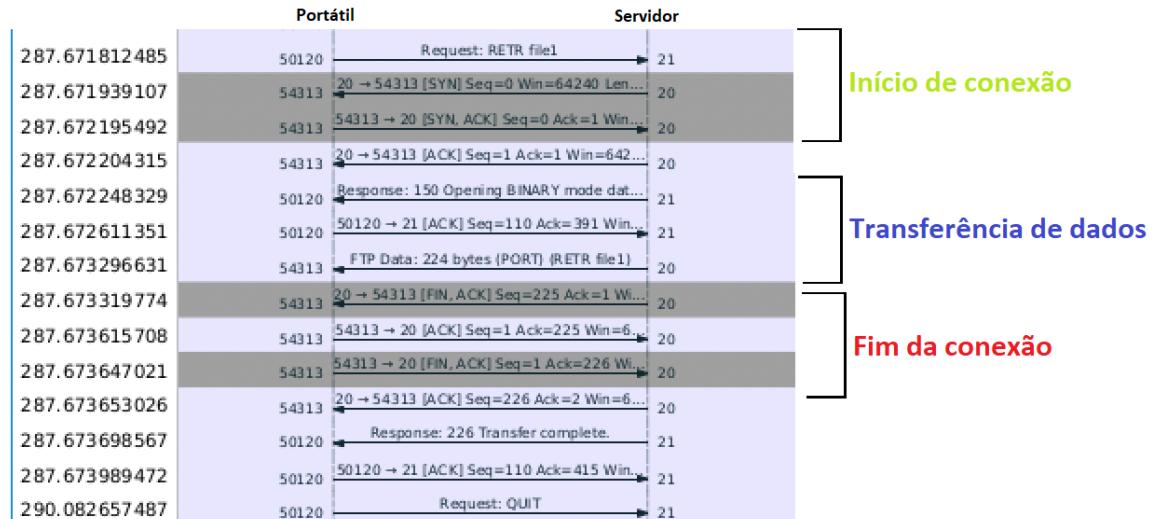


Figura 1.3: Diagrama temporal para transferência de file1 por ftp

1.1.3 Exercício 3

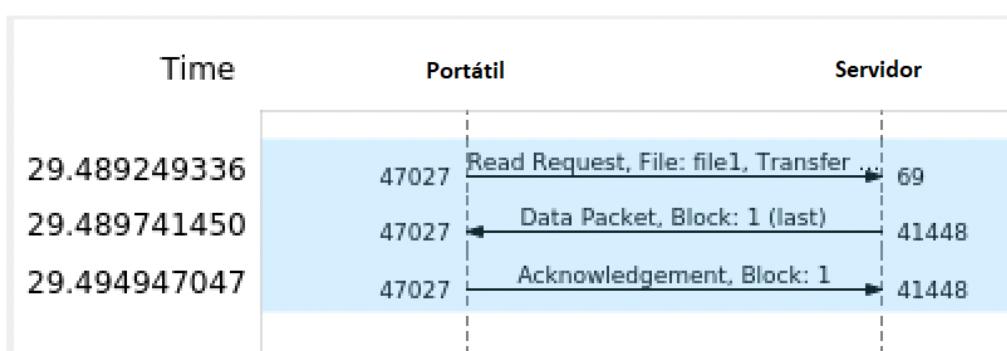


Figura 1.4: Diagrama temporal para transferência de file1 por tftp

1.1.4 Exercício 4

SFTP

- i) Protocolo TCP
- ii) Este protocolo é o menos eficiente de todos os protocolos testados
- iii) Este protocolo é complexo pois tem várias funcionalidades
- iv) Este protocolo é seguro uma vez que recorre sempre à autenticação do cliente e à encriptação de dados

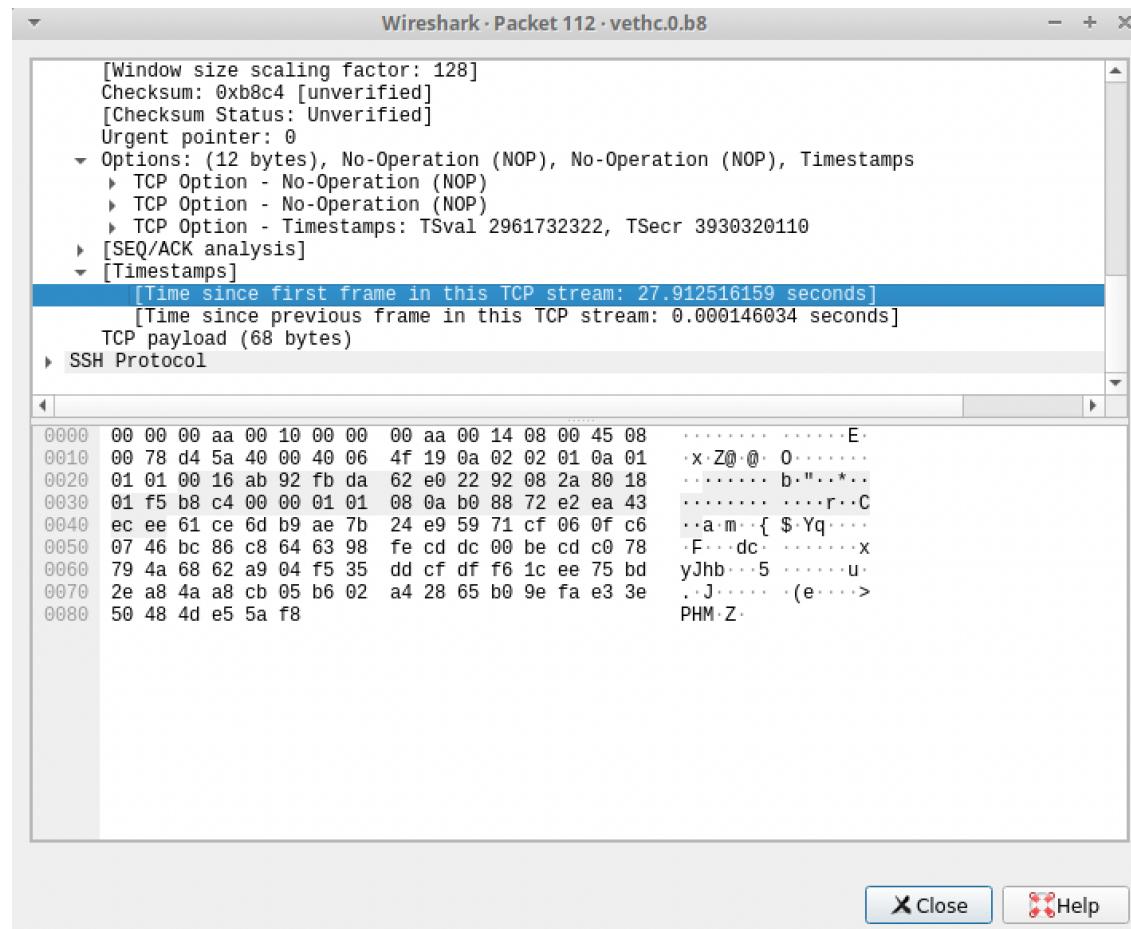


Figura 1.5: SFTP

FTP

- i) Protocolo TCP
- ii) Este é o terceiro mais rápido mas extamente mais rápido que o sftp
- iii) Este protocolo é complexo pois garante segurança na transferência de dados
- iv) Este protocolo requer apenas autenticação

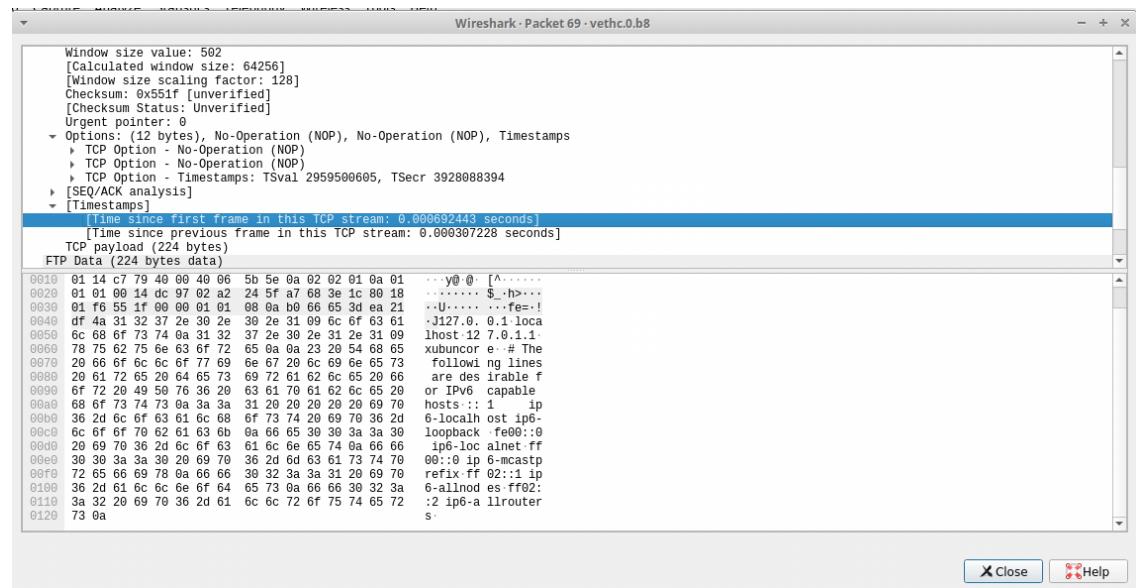


Figura 1.6: FTP

TFTP

- i) Protocolo UDP
- ii) Este protocolo é segundo mais rápido contudo não é fiável a transferência de ficheiros visto que usa o protocolo UDP
- iii) Visto que usa o protocolo UDP, não é muito complexo.
- iv) Este protocolo não utiliza nem autenticação nem encriptação, sendo pouco seguro

```
► Frame 10: 270 bytes on wire (2160 bits), 270 bytes captured (2160 bits) on interface vethc.0.87, id 0
  ▶ 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)
    ▶ Destination: 00:00:00_aa:00:10 (00:00:00:aa:00:10)
      Address: 00:00:00_aa:00:10 (00:00:00:aa:00:10)
        ....0. .... .... .... = LG bit: Globally unique address (factory default)
        ....0. .... .... .... = IG bit: Individual address (unicast)
    ▶ Source: 00:00:00_aa:00:14 (00:00:00:aa:00:14)
      Type: IPv4 (0x0800)
  ▶ Internet Protocol Version 4, Src: 10.2.2.1, Dst: 10.1.1.1
  ▶ User Datagram Protocol, Src Port: 58189, Dst Port: 43134
    Source Port: 58189
    Destination Port: 43134
    Length: 236
    Checksum: 0x1bc8 [unverified]
      [Checksum Status: Unverified]
      [Stream index: 1]
    ▶ [Timestamps]
      [Time since first frame: 0.000000000 seconds]
      [Time since previous frame: 0.000000000 seconds]
  ▶ Trivial File Transfer Protocol
  ▶ Data (224 bytes)

0000  00 00 00 aa 00 10 00 00 00 aa 00 14 08 00 45 00 .....E.
0010  01 00 4b 18 40 00 40 11 d7 d0 0a 02 02 01 0a 01 ..K@@.....
0020  01 01 e3 4d a8 7e 00 ec 1b c8 00 03 00 01 b1 32 ..M~...12
0030  37 2e 30 2e 30 2e 31 09 6c 6f 63 61 6c 68 6f 73 7.0.0.1.localhos
0040  74 0a 31 32 37 2e 30 2e 31 2e 31 09 78 75 62 75 t:127.0.1.1.xubu
0050  6e 63 6f 72 65 0a 0a 23 20 54 68 65 20 66 6f 6c ncore.# The fol
0060  6c 6f 77 69 66 67 20 6c 69 6e 65 73 20 61 72 65 lowing lines are
0070  20 64 65 73 69 72 61 62 6c 65 20 66 6f 72 20 49 |desirab le for I
0080  50 76 36 20 63 61 70 61 62 6c 65 20 68 6f 73 74 Pv6 capa ble host
0090  73 0a 3a 3a 31 20 20 20 20 20 69 70 36 2d 6c 6f s::1 ip6-loop
00a0  63 61 6c 68 6f 73 74 20 69 70 36 2d 6c 6f 6f 70 calhost ip6-loop
00b0  62 61 63 6b 0a 66 65 30 30 3a 3a 30 20 69 70 36 back-fe0 0::0 ip6
00c0  2d 6c 6f 63 61 6c 6e 65 74 0a 66 66 30 30 3a 3a -localne t:ff00::0
00d0  30 20 69 70 36 2d 6d 63 61 73 74 70 72 65 66 69 0 ip6-mc astprefi
00e0  78 0a 66 66 30 32 3a 3a 31 20 69 70 36 2d 61 6c x:ff02::1 ip6-al
00f0  6c 6e 6f 64 65 73 0a 66 66 30 32 3a 3a 32 20 69 lnodes.f f02::2 i
0100  70 36 2d 61 6c 6c 72 6f 75 74 65 72 73 0a p6-allro uters.
```

Figura 1.7: TFTP

HTTP

- i) Protocolo TCP
- ii) Este protocolo é o mais rápido tendo tempos muito próximos de 0 (grande eficiência)
- iii) Usa protocolo TCP, porém podemos considerar pouco complexo visto que permite a legibilidade da informação transmitida facilmente para agentes exteriores
- iv) Este protocolo apenas utiliza autenticação logo não é muito seguro

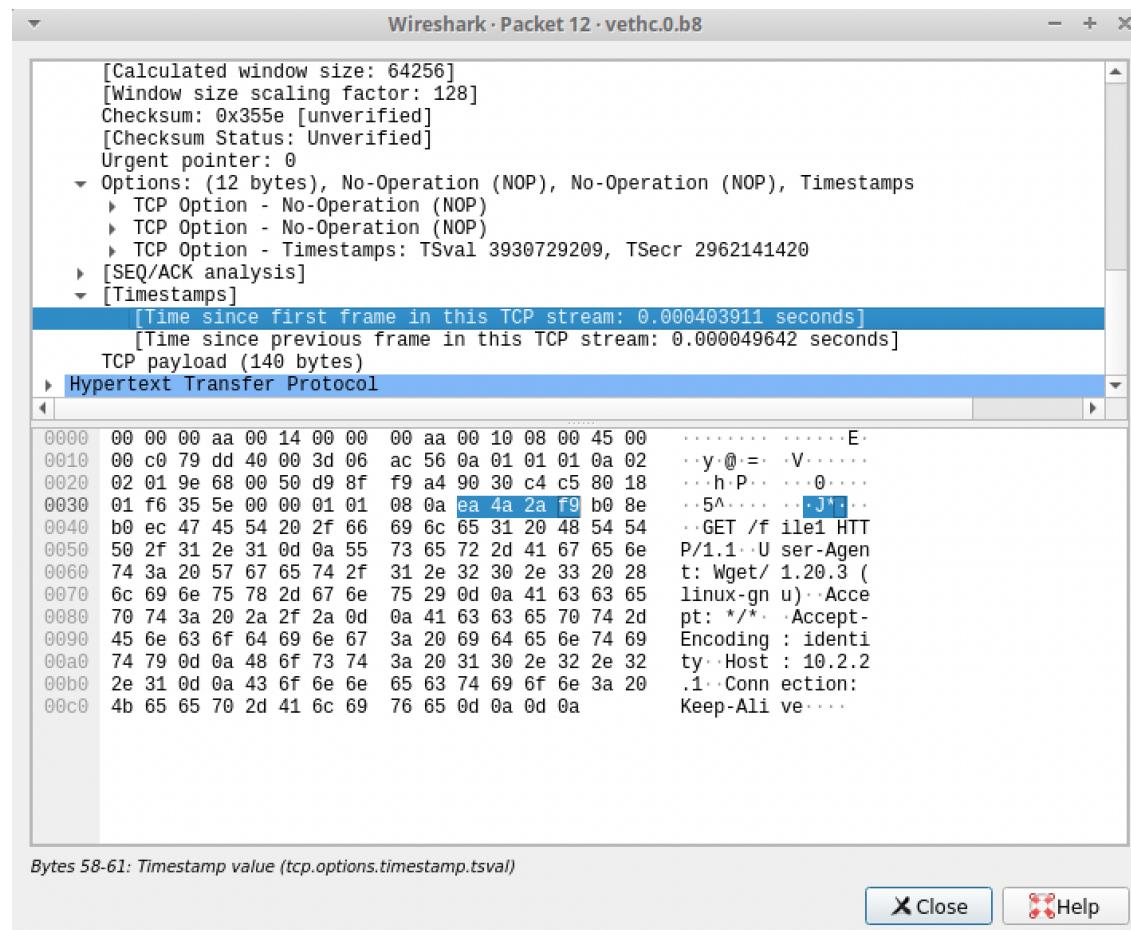


Figura 1.8: HTTP

1.2 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	-	-	-	-
Traceroute	-	UDP	33442	-
Telnet	TELNET	TCP	23	20
FTP	FTP	TCP	21	20
TFTP	TFTP	UPD	69	8
HTTP(browser)	HTTP	TCP	80	20
NSLookup	DNS	TCP	53	20
SSH	SSHv2	TCP	22	20

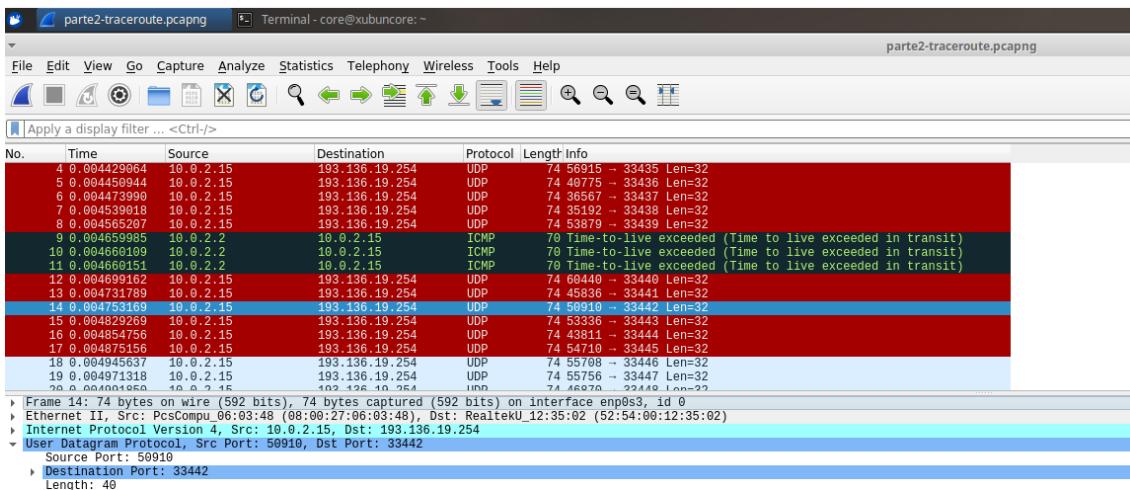


Figura 1.9: Traceroute

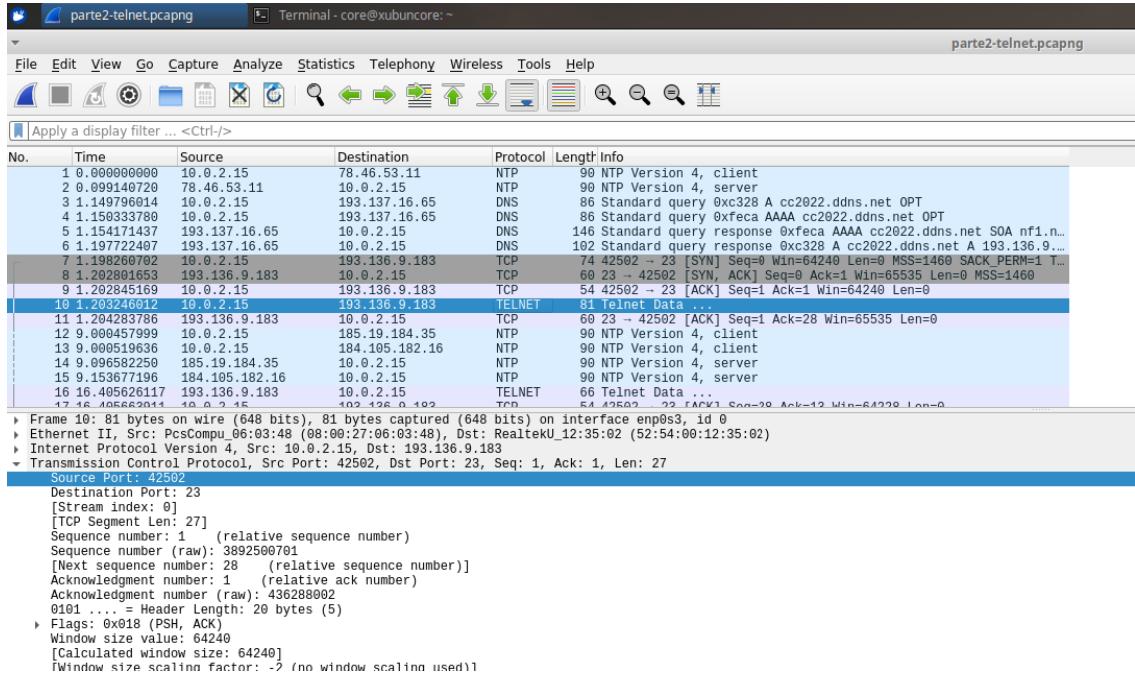


Figura 1.10: TELNET

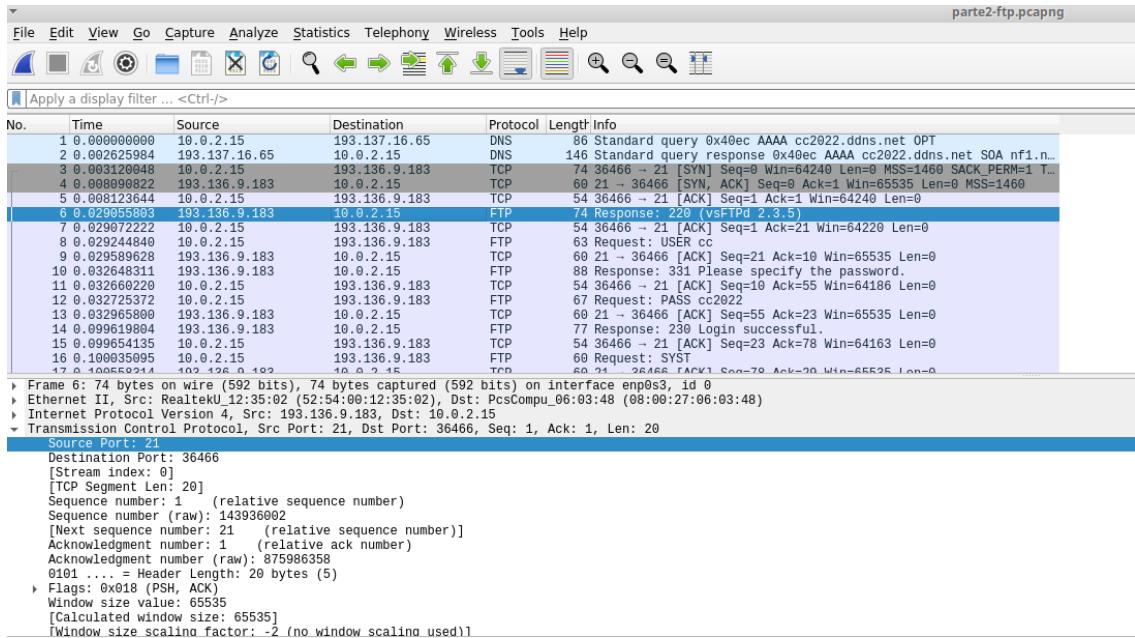


Figura 1.11: FTP

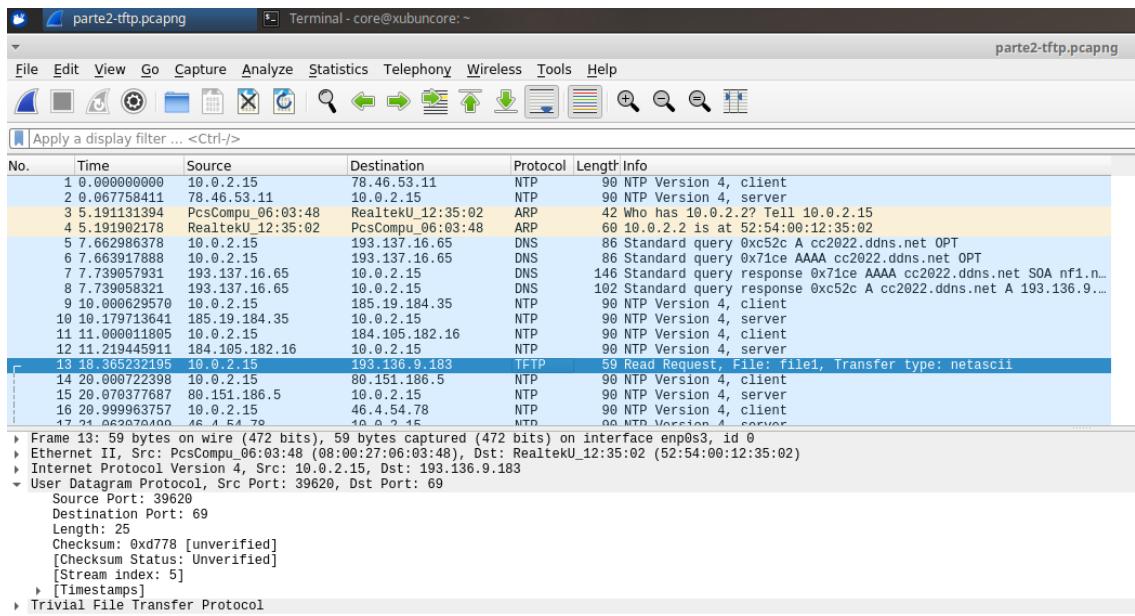


Figura 1.12: TFTP

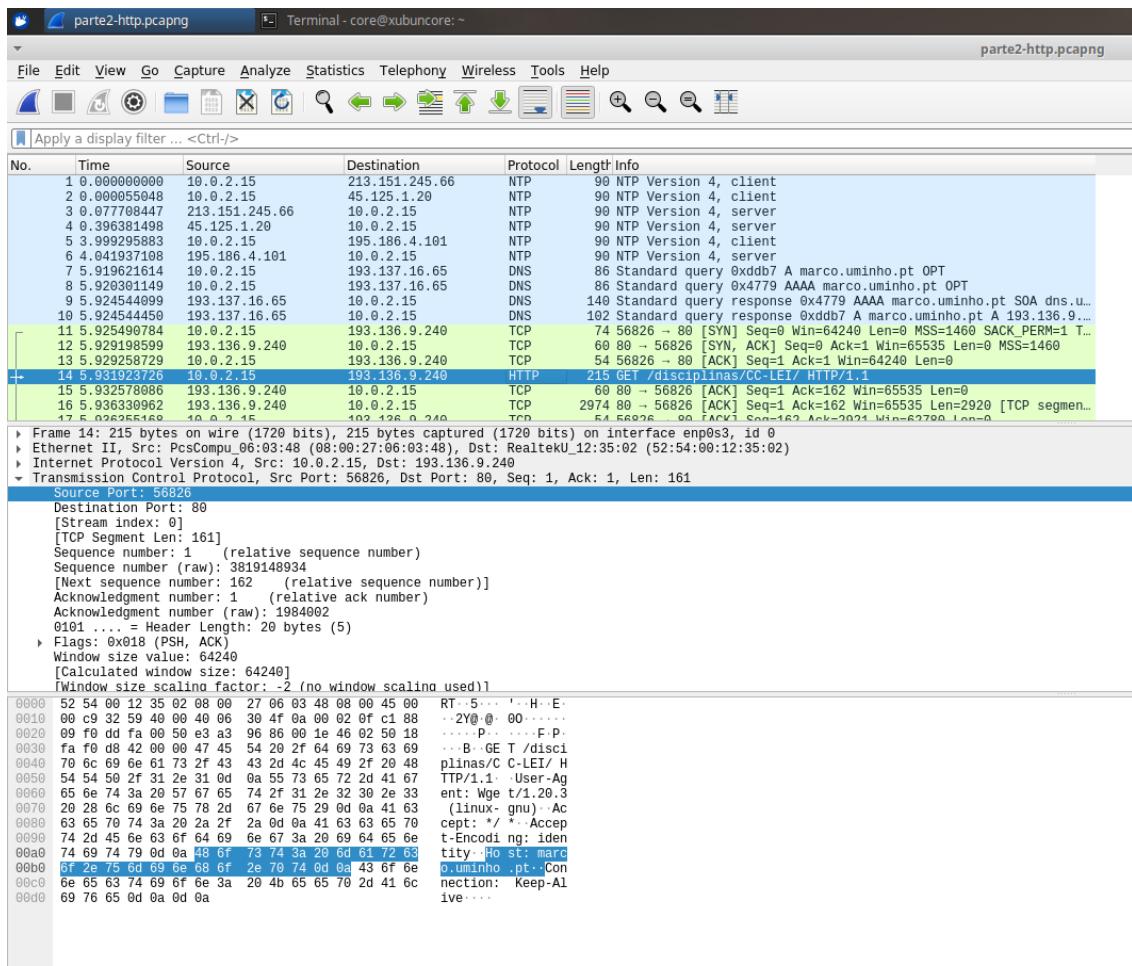


Figura 1.13: HTTP

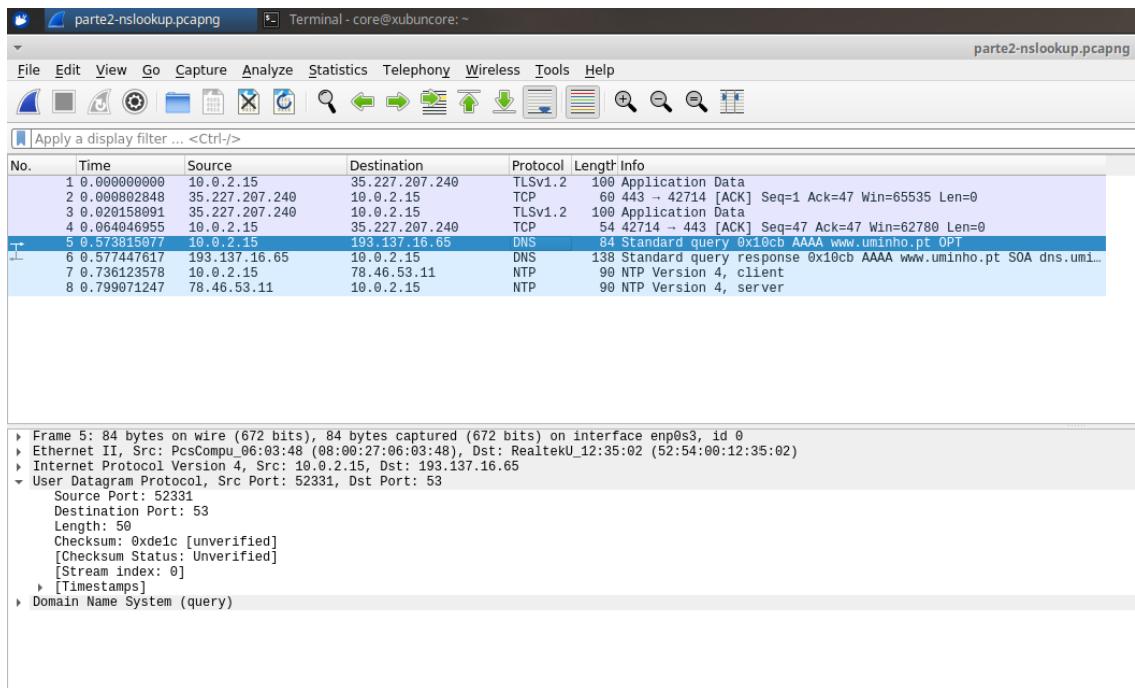


Figura 1.14: NSLookup

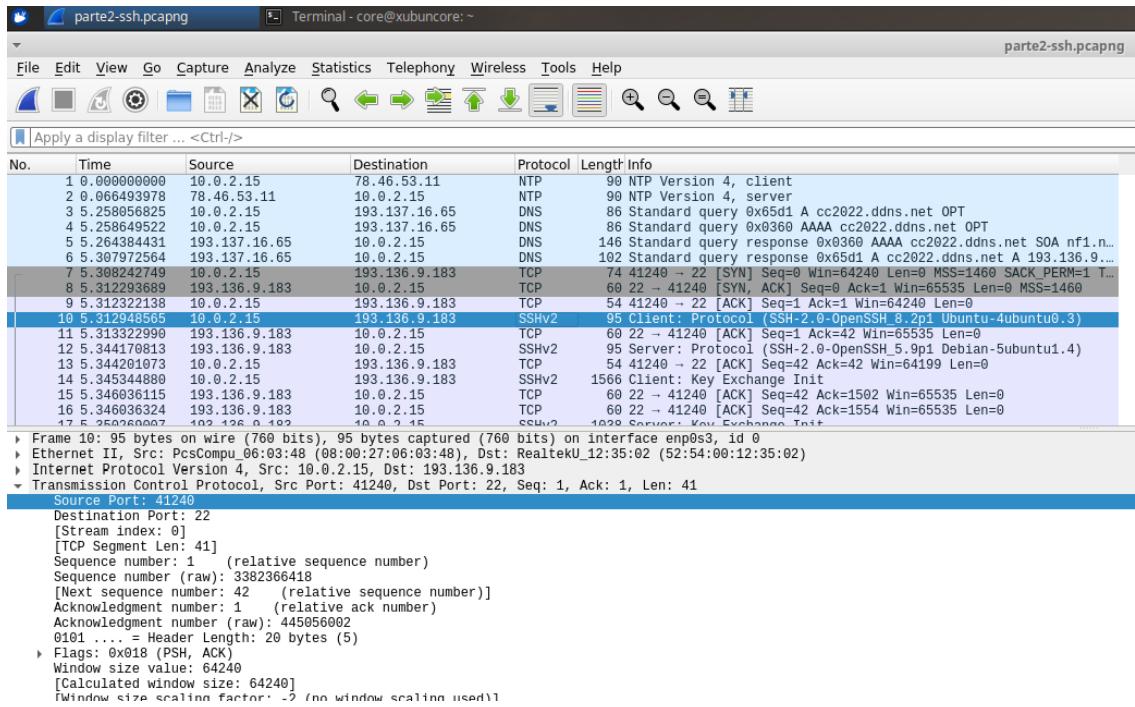


Figura 1.15: SSH

Capítulo 2

Conclusões

Com a execução deste trabalho, julgamos ter percebido as diferenças entre os protocolos da camada de transporte, ou seja, os seus diferentes desempenhos consequentes de diferentes níveis de proteção e trabalhos dos respetivos dados. Além disso, experenciamos a utilização de ferramentas como o Core e o Wireshark essenciais para o estudo deste trabalho prático.