



UNIVERSIDADE DO MINHO  
LICENCIATURA EM ENGENHARIA INFORMÁTICA

# Comunicações por Computador

Trabalho Prático nº1  
PL3 - GRUPO 9  
**Protocolos da Camada de  
Transporte**

Henrique Pereira (a100831)      Maya Gomes (a100822)      José  
Faria(a95255)

October 3, 2023

# Conteúdos

<b>1</b>	<b>Introdução</b>	<b>3</b>
<b>2</b>	<b>Questões e Respostas</b>	<b>4</b>
2.1	QUESTÕES (Parte I)	4
2.1.1	Questão 1:	4
2.1.2	Questão 2:	9
2.1.3	Questão 3:	12
2.1.4	Questão 4:	13
2.2	Uso da camada de transporte por parte das aplicações (Parte II)	14
2.2.1	Questão 1:	14

# 1 Introdução

Neste exercício pretende-se transferir o mesmo ficheiro usando 4 serviços diferentes: SFTP, FTP, TFTP e HTTP, capturando todos os pacotes trocados durante a transferência com o Wireshark.

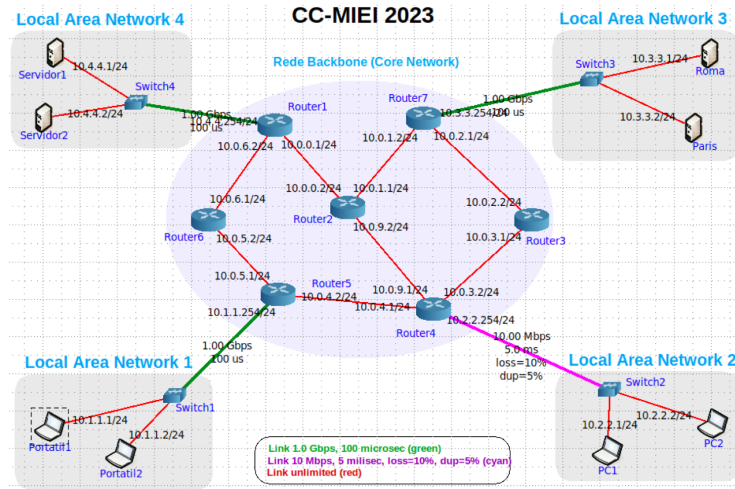


Figure 1: Topologia de Rede (backbone, acesso e local).

## 2 Questões e Respostas

### 2.1 QUESTÕES (Parte I)

#### 2.1.1 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.*

Os pacotes são conjuntos de dados que ocasionalmente não conseguem alcançar o seu destino, e às vezes podem-se perder durante o processo de transferência. Essa perda pode afetar naturalmente várias aplicações, mas tem um impacto mais significativo naquelas que dependem de transferências de dados em tempo real. No entanto, perdas ocasionais geralmente não afetam muito o desempenho geral da rede, mas a ocorrência frequente de perda de pacotes representa efetivamente um problema. Essas perdas podem ser causadas por uma série de razões, como aumento no tráfego, erros de software, problemas de hardware e software de rede desatualizados, bem como violações de segurança, como ataques e ameaças à integridade da rede.

A camada que lida com as perdas e duplicações é a camada de transporte, com o seu protocolo confiável TCP. Os resultados e efeitos destas perdas variam consoante o protocolo em questão. Desta forma, são os protocolos TCP (Protocolo de Controlo de Transmissão) e UDP (Protocolo de Datagrama de Usuário) que procuram resolver este problema. O efeito dessas perdas e/ou duplicações são os seguintes:

- Um protocolo de transmissão (TCP) é responsável por resolver estas perdas. O TCP é um protocolo orientado à conexão e é encarregado de gerenciar o controle de fluxo da comunicação. Ele avalia, identifica e retransmite os pacotes (controle de erros), quando ocorrem problemas específicos, garantindo assim a correção de erros durante a transmissão.
- Um protocolo de Usuário (UDP), não possui recursos para corrigir erros na transmissão, limitando-se a detetá-los. Assim, ele descarta diretamente o pacote em vez de o reenviar. Desta forma, os protocolos superiores terão de ser responsáveis por reenviar os pacotes para que não se percam. Além disso, no UDP, não ocorre segmentação dos dados em pacotes menores, e não são enviados ACKs (confirmações de recebimento) para confirmar que os pacotes foram entregues com sucesso.

```

root@Portatil1:/tmp/pycore.42189/Portatil1.conf# ping -c 20 10.4.4.1 | tee file-ping-output
PING 10.4.4.1 (10.4.4.1) 56(84) bytes of data.
64 bytes from 10.4.4.1: icmp_seq=1 ttl=61 time=0.581 ms
64 bytes from 10.4.4.1: icmp_seq=2 ttl=61 time=0.375 ms
64 bytes from 10.4.4.1: icmp_seq=3 ttl=61 time=0.437 ms
64 bytes from 10.4.4.1: icmp_seq=4 ttl=61 time=0.395 ms
64 bytes from 10.4.4.1: icmp_seq=5 ttl=61 time=0.311 ms
64 bytes from 10.4.4.1: icmp_seq=6 ttl=61 time=0.380 ms
64 bytes from 10.4.4.1: icmp_seq=7 ttl=61 time=0.322 ms
64 bytes from 10.4.4.1: icmp_seq=8 ttl=61 time=0.342 ms
64 bytes from 10.4.4.1: icmp_seq=9 ttl=61 time=0.351 ms
64 bytes from 10.4.4.1: icmp_seq=10 ttl=61 time=0.309 ms
64 bytes from 10.4.4.1: icmp_seq=11 ttl=61 time=0.325 ms
64 bytes from 10.4.4.1: icmp_seq=12 ttl=61 time=0.435 ms
64 bytes from 10.4.4.1: icmp_seq=13 ttl=61 time=2.72 ms
64 bytes from 10.4.4.1: icmp_seq=14 ttl=61 time=0.343 ms
64 bytes from 10.4.4.1: icmp_seq=15 ttl=61 time=0.321 ms
64 bytes from 10.4.4.1: icmp_seq=16 ttl=61 time=0.670 ms
64 bytes from 10.4.4.1: icmp_seq=17 ttl=61 time=0.408 ms
64 bytes from 10.4.4.1: icmp_seq=18 ttl=61 time=0.308 ms
64 bytes from 10.4.4.1: icmp_seq=19 ttl=61 time=0.545 ms
64 bytes from 10.4.4.1: icmp_seq=20 ttl=61 time=0.361 ms

--- 10.4.4.1 ping statistics ---
20 packets transmitted, 20 received, 0% packet loss, time 19402ms
rtt min/avg/max/mdev = 0.308/0.511/2.719/0.515 ms
root@Portatil1:/tmp/pycore.42189/Portatil1.conf#

```

Figure 2: Comando PING Portatil1 - conectividade

```

PC1.conf# ping -c 20 10.4.4.1 | tee file-ping-output
PING 10.4.4.1 (10.4.4.1) 56(84) bytes of data.
64 bytes from 10.4.4.1: icmp_seq=1 ttl=61 time=10.9 ms
64 bytes from 10.4.4.1: icmp_seq=2 ttl=61 time=6.02 ms
64 bytes from 10.4.4.1: icmp_seq=3 ttl=61 time=6.27 ms
64 bytes from 10.4.4.1: icmp_seq=4 ttl=61 time=6.09 ms
64 bytes from 10.4.4.1: icmp_seq=5 ttl=61 time=5.43 ms
64 bytes from 10.4.4.1: icmp_seq=6 ttl=61 time=6.07 ms
64 bytes from 10.4.4.1: icmp_seq=7 ttl=61 time=5.31 ms
64 bytes from 10.4.4.1: icmp_seq=8 ttl=61 time=6.04 ms
64 bytes from 10.4.4.1: icmp_seq=9 ttl=61 time=5.42 ms
64 bytes from 10.4.4.1: icmp_seq=10 ttl=61 time=6.06 ms
64 bytes from 10.4.4.1: icmp_seq=11 ttl=61 time=6.15 ms
64 bytes from 10.4.4.1: icmp_seq=12 ttl=61 time=5.44 ms
64 bytes from 10.4.4.1: icmp_seq=13 ttl=61 time=5.29 ms
64 bytes from 10.4.4.1: icmp_seq=14 ttl=61 time=5.93 ms
64 bytes from 10.4.4.1: icmp_seq=15 ttl=61 time=5.44 ms
64 bytes from 10.4.4.1: icmp_seq=17 ttl=61 time=5.70 ms
64 bytes from 10.4.4.1: icmp_seq=18 ttl=61 time=5.77 ms
64 bytes from 10.4.4.1: icmp_seq=19 ttl=61 time=5.40 ms
64 bytes from 10.4.4.1: icmp_seq=20 ttl=61 time=5.96 ms

--- 10.4.4.1 ping statistics ---
20 packets transmitted, 19 received, 5% packet loss, time 19036ms
rtt min/avg/max/mdev = 5.285/6.038/10.944/1.198 ms

```

Figure 3: Comando PING PC1 - conectividade

No que toca à duplicação de pacotes podemos observar que entre o Portátil da LAN 1 e o PC1 da LAN 2 o PING é diferente sendo muito menor no Portátil.

Notamos com o comando PING ocorre perda de pacotes, pois enquanto que no Portatil1 são transmitidos e recebidos 20 pacotes, no PC1 são transmitidos 20 pacotes mas somente 19 são recebidos.

```

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 (262.9207 kB/s)
ftp>

```

Figure 4: ftp PC1

```

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 (262.9207 kB/s)
ftp>

```

Figure 5: Wireshark ftp file1 no PC1

202	48.311868510	10.2.2.1	10.4.4.1	TCP	66	36703 → 20 [ACK] Seq=1 Ack=82537 Win=128512 Len=0 TSval=40639.
203	48.311869257	10.2.2.1	10.4.4.1	TCP	66	36703 → 20 [ACK] Seq=1 Ack=83985 Win=131328 Len=0 TSval=40639.
204	48.327923023	10.4.4.1	10.2.2.1	FTP-DA	390	[TCP Previous segment not captured] FTP Data: 240 bytes [PORT]
205	48.343238450	10.2.2.1	10.4.4.1	TCP	86	36703 → 20 [ACK] Seq=1 Ack=94121 Win=203776 Len=0 TSval=40639.
206	48.347441400	10.4.4.1	10.2.2.1	FTP	1514	[TCP Previous segment not captured] Seq=1 Win=0 Len=0 Seq=1 Win=0 Len=0
207	48.348591646	10.2.2.1	10.4.4.1	TCP	86	36703 → 20 [ACK] Seq=1 Ack=95569 Win=206592 Len=0 TSval=40639.
208	48.349131510	10.4.4.1	10.2.2.1	TCP	1514	[TCP Retransmission] 20 → 36703 [ACK] Seq=15552 Ack=1 Win=64
209	48.353393556	10.2.2.1	10.4.4.1	TCP	86	36703 → 20 [ACK] Seq=1 Ack=97611 Win=209336 Len=0 TSval=40639.
210	48.354129133	10.4.4.1	10.2.2.1	TCP	1514	[TCP Retransmission] 20 → 36703 [ACK] Seq=97611 Ack=1 Win=64
211	48.354230940	10.4.4.1	10.2.2.1	TCP	1514	[TCP Retransmission] 20 → 36703 [ACK] Seq=98495 Ack=1 Win=64
212	48.354231556	10.4.4.1	10.2.2.1	TCP	1514	[TCP Retransmission] 20 → 36703 [PSH, ACK] Seq=99013 Ack=1 Win=64
213	48.354232094	10.4.4.1	10.2.2.1	TCP	1514	[TCP Retransmission] 20 → 36703 [ACK] Seq=101381 Ack=1 Win=64
214	48.354232579	10.4.4.1	10.2.2.1	TCP	1514	[TCP Retransmission] 20 → 36703 [ACK] Seq=102899 Ack=1 Win=64
215	48.354233066	10.4.4.1	10.2.2.1	TCP	1514	[TCP Retransmission] 20 → 36703 [PSH, ACK] Seq=104257 Ack=1 Win=64
216	48.354233552	10.4.4.1	10.2.2.1	TCP	1514	[TCP Retransmission] 20 → 36703 [ACK] Seq=105705 Ack=1 Win=64
217	48.354233945	10.4.4.1	10.2.2.1	TCP	1514	[TCP Retransmission] 20 → 36703 [ACK] Seq=107153 Ack=1 Win=64
218	48.354234372	10.4.4.1	10.2.2.1	TCP	1514	[TCP Retransmission] 20 → 36703 [ACK] Seq=108601 Ack=1 Win=64
219	48.354234761	10.4.4.1	10.2.2.1	TCP	1514	[TCP Retransmission] 20 → 36703 [PSH, ACK] Seq=109091 Ack=1 Win=64
220	48.354235150	10.4.4.1	10.2.2.1	TCP	1514	[TCP Retransmission] 20 → 36703 [ACK] Seq=134665 Ack=1 Win=64
221	48.354235539	10.4.4.1	10.2.2.1	TCP	1514	[TCP Retransmission] 20 → 36703 [ACK] Seq=136113 Ack=1 Win=64
222	48.354235928	10.4.4.1	10.2.2.1	TCP	1514	[TCP Retransmission] 20 → 36703 [PSH, ACK] Seq=137561 Ack=1 Win=64
223	48.354236317	10.4.4.1	10.2.2.1	TCP	1514	[TCP Retransmission] 20 → 36703 [ACK] Seq=139009 Ack=1 Win=64
224	48.359897806	10.2.2.1	10.4.4.1	TCP	86	36703 → 20 [ACK] Seq=1 Ack=98465 Win=212480 Len=0 TSval=40639.
225	48.359901199	10.2.2.1	10.4.4.1	TCP	86	36703 → 20 [ACK] Seq=1 Ack=99913 Win=215296 Len=0 TSval=40639.
226	48.359901920	10.2.2.1	10.4.4.1	TCP	86	36703 → 20 [ACK] Seq=1 Ack=101361 Win=218240 Len=0 TSval=40639.
227	48.359902615	10.2.2.1	10.4.4.1	TCP	86	36703 → 20 [ACK] Seq=1 Ack=102899 Win=221184 Len=0 TSval=40639.
228	48.359903314	10.2.2.1	10.4.4.1	TCP	86	36703 → 20 [ACK] Seq=1 Ack=104257 Win=224896 Len=0 TSval=40639.
229	48.359904009	10.2.2.1	10.4.4.1	TCP	86	36703 → 20 [ACK] Seq=1 Ack=105705 Win=228640 Len=0 TSval=40639.
230	48.359904696	10.2.2.1	10.4.4.1	TCP	78	36703 → 20 [ACK] Seq=1 Ack=134665 Win=235648 Len=0 TSval=40639.
231	48.359931845	10.2.2.1	10.4.4.1	TCP	78	36703 → 20 [ACK] Seq=1 Ack=136113 Win=238464 Len=0 TSval=40639.
232	48.359932751	10.2.2.1	10.4.4.1	TCP	78	36703 → 20 [ACK] Seq=1 Ack=137561 Win=241408 Len=0 TSval=40639.
233	48.570426994	10.4.4.1	10.2.2.1	TCP	1514	[TCP Retransmission] 20 → 36703 [ACK] Seq=137561 Ack=1 Win=64
234	48.570427474	10.4.4.1	10.2.2.1	TCP	78	[TCP Previous segment not captured] 36703 → 20 [ACK] Seq=2 Ac
235	48.570461157	10.4.4.1	10.2.2.1	FTP	90	Response: 220 Transfer complete.
236	48.581944573	10.2.2.1	10.4.4.1	TCP	66	49346 → 21 [ACK] Seq=78 Ack=313 Win=502 Len=0 TSval=406390996.

Figure 6: Wireshark file2 no PC1

674	780.435661567	10.2.2.1	10.4.4.1	TFTP	56 Read Request, File: file2, Transfer type: octet
675	780.436808539	10.4.4.1	10.2.2.1	TFTP	558 Data Packet, Block: 1
676	780.442586599	10.2.2.1	10.4.4.1	TFTP	46 Acknowledgement, Block: 1
677	780.442670762	10.4.4.1	10.2.2.1	TFTP	558 Data Packet, Block: 2
678	780.448458120	10.2.2.1	10.4.4.1	TFTP	46 Acknowledgement, Block: 2
679	780.449269053	10.4.4.1	10.2.2.1	TFTP	558 Data Packet, Block: 3
680	780.454448139	10.2.2.1	10.4.4.1	TFTP	46 Acknowledgement, Block: 3
681	780.454629664	10.4.4.1	10.2.2.1	TFTP	558 Data Packet, Block: 4
682	780.460062555	10.2.2.1	10.4.4.1	TFTP	46 Acknowledgement, Block: 4
683	780.460228590	10.4.4.1	10.2.2.1	TFTP	558 Data Packet, Block: 5
684	780.465918376	10.2.2.1	10.4.4.1	TFTP	46 Acknowledgement, Block: 5
685	780.466113919	10.4.4.1	10.2.2.1	TFTP	558 Data Packet, Block: 6
686	780.471791580	10.2.2.1	10.4.4.1	TFTP	46 Acknowledgement, Block: 6
687	780.472094684	10.4.4.1	10.2.2.1	TFTP	558 Data Packet, Block: 7
688	780.477627575	10.2.2.1	10.4.4.1	TFTP	46 Acknowledgement, Block: 7
689	780.477722259	10.4.4.1	10.2.2.1	TFTP	558 Data Packet, Block: 8
690	780.483552166	10.2.2.1	10.4.4.1	TFTP	46 Acknowledgement, Block: 8
691	780.483879593	10.4.4.1	10.2.2.1	TFTP	558 Data Packet, Block: 9
692	780.488998987	10.2.2.1	10.4.4.1	TFTP	46 Acknowledgement, Block: 9
693	780.489178642	10.4.4.1	10.2.2.1	TFTP	558 Data Packet, Block: 10
694	780.495205425	10.2.2.1	10.4.4.1	TFTP	46 Acknowledgement, Block: 10
695	780.495384661	10.4.4.1	10.2.2.1	TFTP	558 Data Packet, Block: 11
696	780.501002328	10.2.2.1	10.4.4.1	TFTP	46 Acknowledgement, Block: 11
697	780.501218186	10.4.4.1	10.2.2.1	TFTP	558 Data Packet, Block: 12
698	780.506091735	10.2.2.1	10.4.4.1	TFTP	46 Acknowledgement, Block: 12
699	780.507080152	10.4.4.1	10.2.2.1	TFTP	558 Data Packet, Block: 13
700	780.512737013	10.2.2.1	10.4.4.1	TFTP	46 Acknowledgement, Block: 13
701	780.512921362	10.4.4.1	10.2.2.1	TFTP	558 Data Packet, Block: 14

Figure 7: Wireshark tftp file2 no PC1

1071	498.581509534	10.4.4.1	10.2.2.1	SSHv2	1514 Server: Encrypted packet (len=1448)
1072	498.581510190	10.4.4.1	10.2.2.1	SSHv2	1514 Server: Encrypted packet (len=1448)
1073	498.581510955	10.4.4.1	10.2.2.1	SSHv2	1514 Server: Encrypted packet (len=1448)
1074	498.581511740	10.4.4.1	10.2.2.1	SSHv2	1514 Server: Encrypted packet (len=1448)
1075	498.581512400	10.4.4.1	10.2.2.1	SSHv2	1514 Server: Encrypted packet (len=1448)
1076	498.581513462	10.4.4.1	10.2.2.1	SSHv2	1514 Server: Encrypted packet (len=1448)
1077	498.581514163	10.4.4.1	10.2.2.1	SSHv2	1514 Server: Encrypted packet (len=1448)
1078	498.581514830	10.4.4.1	10.2.2.1	SSHv2	1514 Server: Encrypted packet (len=1448)
1079	498.581515626	10.4.4.1	10.2.2.1	SSHv2	558 Server: Encrypted packet (len=492)
1080	498.581516489	10.4.4.1	10.2.2.1	SSHv2	1514 Server: Encrypted packet (len=1448)
1081	498.581517080	10.4.4.1	10.2.2.1	SSHv2	1514 Server: Encrypted packet (len=1448)
1082	498.581517820	10.4.4.1	10.2.2.1	SSHv2	1514 Server: Encrypted packet (len=1448)
1083	498.581518558	10.4.4.1	10.2.2.1	SSHv2	1514 Server: Encrypted packet (len=1448)
1084	498.581519299	10.4.4.1	10.2.2.1	SSHv2	1514 Server: Encrypted packet (len=1448)
1085	498.581745070	10.4.4.1	10.2.2.1	SSHv2	1514 Server: Encrypted packet (len=1448)
1086	498.581745847	10.4.4.1	10.2.2.1	SSHv2	1514 Server: Encrypted packet (len=1448)
1087	498.581746535	10.4.4.1	10.2.2.1	SSHv2	1514 Server: Encrypted packet (len=1448)
1088	498.581747193	10.4.4.1	10.2.2.1	SSHv2	1514 Server: Encrypted packet (len=1448)
1089	498.581747846	10.4.4.1	10.2.2.1	SSHv2	1514 Server: Encrypted packet (len=1448)
1090	498.581748496	10.4.4.1	10.2.2.1	SSHv2	1514 Server: Encrypted packet (len=1448)
1091	498.581749144	10.4.4.1	10.2.2.1	SSHv2	558 Server: Encrypted packet (len=492)
1092	498.581896506	10.4.4.1	10.2.2.1	SSHv2	1514 Server: Encrypted packet (len=1448)
1093	498.581897288	10.4.4.1	10.2.2.1	SSHv2	1514 Server: Encrypted packet (len=1448)
1094	498.581898011	10.4.4.1	10.2.2.1	SSHv2	1514 Server: Encrypted packet (len=1448)
1095	498.581898724	10.4.4.1	10.2.2.1	SSHv2	1514 Server: Encrypted packet (len=1448)
1096	498.581899444	10.4.4.1	10.2.2.1	SSHv2	1514 Server: Encrypted packet (len=1448)
1097	498.582467242	10.4.4.1	10.2.2.1	SSHv2	1514 Server: Encrypted packet (len=1448)
1098	498.582468070	10.4.4.1	10.2.2.1	SSHv2	1514 Server: Encrypted packet (len=1448)
1099	498.582468817	10.4.4.1	10.2.2.1	SSHv2	1062 Server: Encrypted packet (len=996)
1100	498.586868043	10.2.2.1	10.4.4.1	TCP	66 46248 -> 22 [ACK] Seq=4006 Ack=106342 Win=114944 Len=0 TSval=4..
1101	498.586871424	10.2.2.1	10.4.4.1	TCP	66 46248 -> 22 [ACK] Seq=4006 Ack=109238 Win=120784 Len=0 TSval=4..
1102	498.586872212	10.2.2.1	10.4.4.1	TCP	66 46248 -> 22 [ACK] Seq=4006 Ack=112134 Win=126464 Len=0 TSval=4..
1103	498.586936971	10.2.2.1	10.4.4.1	TCP	66 46248 -> 22 [ACK] Seq=4006 Ack=115030 Win=132224 Len=0 TSval=4..
1104	498.586938718	10.2.2.1	10.4.4.1	TCP	66 46248 -> 22 [ACK] Seq=4006 Ack=117926 Win=138112 Len=0 TSval=4..
1105	498.586938835	10.2.2.1	10.4.4.1	TCP	66 [TCP Dup ACK 1104#1] 46248 -> 22 [ACK] Seq=4006 Ack=117926 Win..
1106	498.587029519	10.2.2.1	10.4.4.1	TCP	66 46248 -> 22 [ACK] Seq=4006 Ack=119866 Win=143872 Len=0 TSval=4..
1107	498.587079497	10.2.2.1	10.4.4.1	TCP	66 46248 -> 22 [ACK] Seq=4006 Ack=122762 Win=149632 Len=0 TSval=4..
1108	498.587119285	10.2.2.1	10.4.4.1	TCP	66 46248 -> 22 [ACK] Seq=4006 Ack=125658 Win=155392 Len=0 TSval=4..
1109	498.613401725	10.4.4.1	10.2.2.1	TCP	1062 [TCP Retransmission] 22 -> 46248 [PSH, ACK] Seq=146422 Ack=400..
1110	498.619663697	10.2.2.1	10.4.4.1	TCP	78 [TCP Previous segment not captured] 46248 -> 22 [ACK] Seq=4074..
1111	498.707973589	10.2.2.1	10.4.4.1	TCP	134 [TCP Retransmission] 46248 -> 22 [PSH, ACK] Seq=4006 Ack=14741..
1112	498.709403356	10.4.4.1	10.2.2.1	SSHv2	134 Server: Encrypted packet (len=66)
1113	498.803648140	10.2.2.1	10.4.4.1	TCP	66 46248 -> 22 [ACK] Seq=4074 Ack=147486 Win=201728 Len=0 TSval=4..
1114	499.021856420	10.2.2.1	10.4.4.1	SSHv2	118 Client: Encrypted packet (len=52)
1115	499.022273266	10.4.4.1	10.2.2.1	SSHv2	134 Server: Encrypted packet (len=68)
1116	499.027427899	10.2.2.1	10.4.4.1	TCP	66 46248 -> 22 [ACK] Seq=4126 Ack=147554 Win=201728 Len=0 TSval=4..

Figure 8: Wireshark sftp file2 no PC1

Olhando, para a figura acima, podemos notar uma duplicação de pacote e vários pacotes *Out-of-Order*. Assim, temos o mesmo pacote a ser enviado por múltiplos caminhos: é dado o SYN (*Synchronize*) da porta 46248 para a porta 22 repetidamente e acontece o mesmo nos ACKs. Desta forma, há ACKs duplicados significando a perda de pacotes e tornando a comunicação menos eficiente. Tal como referimos anteriormente, reparamos que na realização do comando PING visualizamos esta tal perda de pacotes.

```

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 (465.4255 kB/s)
ftp>

```

Figure 9: ftp Portatill

98	3.888278883	10.4.4.1	10.1.1.1	FTP-DA..	1514 FTP Data: 1448 bytes (PORT) (RETR file2)
99	3.888279649	10.4.4.1	10.1.1.1	FTP-DA..	1514 FTP Data: 1448 bytes (PORT) (RETR file2)
100	3.888314555	10.1.1.1	10.4.4.1	TCP	78 40049 → 20 [ACK] Seq=1 Ack=41993 Win=77184 Len=0 TSval=412449..
101	3.888319666	10.1.1.1	10.4.4.1	TCP	66 40049 → 20 [ACK] Seq=1 Ack=78193 Win=53248 Len=0 TSval=412449..
102	3.888333731	10.1.1.1	10.4.4.1	TCP	66 40049 → 20 [ACK] Seq=1 Ack=81089 Win=53248 Len=0 TSval=412449..
103	3.888350806	10.4.4.1	10.1.1.1	FTP-DA..	1514 FTP Data: 1448 bytes (PORT) (RETR file2)
104	3.888360670	10.4.4.1	10.1.1.1	FTP-DA..	1514 FTP Data: 1448 bytes (PORT) (RETR file2)
105	3.888361252	10.4.4.1	10.1.1.1	FTP-DA..	1514 FTP Data: 1448 bytes (PORT) (RETR file2)
106	3.888361815	10.4.4.1	10.1.1.1	FTP-DA..	1514 FTP Data: 1448 bytes (PORT) (RETR file2)
107	3.888362409	10.4.4.1	10.1.1.1	FTP-DA..	1514 FTP Data: 1448 bytes (PORT) (RETR file2)
108	3.888362954	10.4.4.1	10.1.1.1	FTP-DA..	1514 FTP Data: 1448 bytes (PORT) (RETR file2)
109	3.888363499	10.4.4.1	10.1.1.1	FTP-DA..	1514 FTP Data: 1448 bytes (PORT) (RETR file2)
110	3.888367811	10.4.4.1	10.1.1.1	FTP-DA..	1514 FTP Data: 1448 bytes (PORT) (RETR file2)
111	3.888399784	10.4.4.1	10.1.1.1	FTP-DA..	1514 FTP Data: 1448 bytes (PORT) (RETR file2)
112	3.888406941	10.4.4.1	10.1.1.1	FTP-DA..	1514 FTP Data: 1448 bytes (PORT) (RETR file2)
113	3.888406491	10.4.4.1	10.1.1.1	FTP-DA..	1514 FTP Data: 1448 bytes (PORT) (RETR file2)
114	3.888443431	10.4.4.1	10.1.1.1	FTP-DA..	1514 FTP Data: 1448 bytes (PORT) (RETR file2)
115	3.888483850	10.1.1.1	10.4.4.1	TCP	66 40049 → 20 [ACK] Seq=1 Ack=83985 Win=79232 Len=0 TSval=412449..
116	3.888504930	10.4.4.1	10.1.1.1	FTP-DA..	1514 FTP Data: 1448 bytes (PORT) (RETR file2)
117	3.888566799	10.4.4.1	10.1.1.1	FTP-DA..	1514 FTP Data: 1448 bytes (PORT) (RETR file2)
118	3.888567389	10.4.4.1	10.1.1.1	FTP-DA..	1514 FTP Data: 1448 bytes (PORT) (RETR file2)
119	3.888567964	10.4.4.1	10.1.1.1	FTP-DA..	1514 FTP Data: 1448 bytes (PORT) (RETR file2)
120	3.888625337	10.1.1.1	10.4.4.1	TCP	66 40049 → 20 [ACK] Seq=1 Ack=105705 Win=58880 Len=0 TSval=412444..
121	3.888736336	10.1.1.1	10.4.4.1	TCP	78 [TCP Window Update] 40049 → 20 [ACK] Seq=1 Ack=105705 Win=800..
122	3.888736845	10.1.1.1	10.4.4.1	TCP	78 [TCP Dup ACK 120#1] 40049 → 20 [ACK] Seq=1 Ack=105705 Win=800..
123	3.888737326	10.1.1.1	10.4.4.1	TCP	78 [TCP Dup ACK 120#2] 40049 → 20 [ACK] Seq=1 Ack=105705 Win=800..
124	3.888737789	10.1.1.1	10.4.4.1	TCP	78 [TCP Dup ACK 120#3] 40049 → 20 [ACK] Seq=1 Ack=105705 Win=800..
125	3.888769551	10.4.4.1	10.1.1.1	FTP-DA..	1514 FTP Data: 1448 bytes (PORT) (RETR file2)
126	3.888779423	10.4.4.1	10.1.1.1	FTP-DA..	1514 FTP Data: 1448 bytes (PORT) (RETR file2)
127	3.888792791	10.4.4.1	10.1.1.1	FTP-DA..	1514 FTP Data: 1448 bytes (PORT) (RETR file2)
128	3.888793879	10.4.4.1	10.1.1.1	FTP-DA..	1514 FTP Data: 1448 bytes (PORT) (RETR file2)
129	3.888810844	10.4.4.1	10.1.1.1	FTP-DA..	1514 FTP Data: 1448 bytes (PORT) (RETR file2)
130	3.888872139	10.4.4.1	10.1.1.1	FTP-DA..	1514 FTP Data: 1448 bytes (PORT) (RETR file2)
131	3.888873645	10.4.4.1	10.1.1.1	FTP-DA..	1514 FTP Data: 1448 bytes (PORT) (RETR file2)
132	3.888874249	10.4.4.1	10.1.1.1	FTP-DA..	1514 FTP Data: 1448 bytes (PORT) (RETR file2)
133	3.888874824	10.4.4.1	10.1.1.1	FTP-DA..	1514 FTP Data: 1448 bytes (PORT) (RETR file2)
134	3.888875432	10.4.4.1	10.1.1.1	FTP-DA..	1514 FTP Data: 1448 bytes (PORT) (RETR file2)
135	3.888879491	10.4.4.1	10.1.1.1	FTP-DA..	1514 FTP Data: 1448 bytes (PORT) (RETR file2)
136	3.888881155	10.4.4.1	10.1.1.1	FTP-DA..	1514 FTP Data: 1448 bytes (PORT) (RETR file2)
137	3.888882135	10.4.4.1	10.1.1.1	FTP-DA..	1514 FTP Data: 1448 bytes (PORT) (RETR file2)
138	3.888883801	10.4.4.1	10.1.1.1	TCP	1514 [TCP Fast Retransmission] 1514 FTP Data: 1448 bytes (PORT) (RETR f..
139	3.888883822	10.4.4.1	10.1.1.1	TCP	1514 [TCP Out-Of-Order] 20 → 40049 [ACK] Seq=107153 Ack=1 Win=6425..
140	3.888884652	10.4.4.1	10.1.1.1	FTP-DA..	1514 FTP Data: 1448 bytes (PORT) (RETR file2)
141	3.889326518	10.1.1.1	10.4.4.1	TCP	78 40049 → 20 [ACK] Seq=1 Ack=107153 Win=78592 Len=0 TSval=41244..
142	3.889327317	10.1.1.1	10.4.4.1	TCP	78 40049 → 20 [ACK] Seq=1 Ack=108661 Win=77194 Len=0 TSval=41244..
143	3.889328993	10.1.1.1	10.4.4.1	TCP	66 40049 → 20 [ACK] Seq=1 Ack=133217 Win=61952 Len=0 TSval=41244..

Figure 10: Wireshark ftp file2 no Portatill



### 2.1.2 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.*

O FTP emprega dois números de porta TCP amplamente reconhecidos: a porta 20 e a porta 21. A primeira porta é utilizada para a transferência de dados, como o envio de arquivos, enquanto que a segunda porta é destinada ao controlo da conexão, ao atuar como a porta de comando.

Para que uma transferência usando FTP ocorra entre o servidor e cliente, numa primeira fase, o cliente deve pedir acesso ao servidor. Assim, esse autentica-se com um user e uma password para efeitos de segurança. De seguida, inicia-se a fase inicial de conexão. O cliente envia um TCP SYN ao servidor, que lhe responde com um TCP SYN/ACK. De seguida, o cliente envia um TCP ACK. Fica a conexão iniciada e o ficheiro pode ser transmitido. Quando a transferência do ficheiro termina, é enviada uma mensagem de "transferência completa" para o cliente. Inicia-se a fase final da conexão, isto é, o servidor utiliza um TCP FIN para indicar que o ficheiro foi transferido e para informar que a conexão vai terminar. Finalmente, o cliente responde com um TCP ACK. O servidor termina a sua conexão, no entanto, o cliente deve fazer o mesmo, sendo que é enviado um TCP FIN, ao qual o servidor responde com um TCP ACK. Termina assim a ligação.

A figura 5 (da questão 1) ilustra as diferentes fases do processo, tanto em relação ao pedido de autenticação ao servidor, como à transferência completa.

Para melhor visualização das diferentes fases, tanto em relação ao pedido de autenticação ao servidor, como à transferência completa. A imagem abaixo já tem aplicado o filtro **tcp.port == 20**, já que o solicitado foi focar na transferência de dados.

tcp.port == 20						
No.	Time	Source	Destination	Protocol	Length	Info
9	13.299218397	10.2.2.1	10.4.4.1	FTP	89	Request: PORT 10,2,2,1,150,171
10	13.299516922	10.4.4.1	10.2.2.1	FTP	117	Response: 200 PORT command successful. Consider using PASV.
11	13.507705071	10.4.4.1	10.2.2.1	TCP	117	[TCP Retransmission] 21 → 49704 [PSH, ACK] Seq=1 Ack=24 Win=510 Len=51
12	13.513577692	10.2.2.1	10.4.4.1	TCP	78	[TCP Previous segment not captured] 49704 → 21 [ACK] Seq=36 Ack=52 Win=
13	13.516575399	10.2.2.1	10.4.4.1	TCP	78	[TCP Retransmission] 49704 → 21 [PSH, ACK] Seq=24 Ack=52 Win=502 Len=1
14	13.517470983	10.4.4.1	10.2.2.1	TCP	74	20 → 38571 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=3416
15	13.522594449	10.2.2.1	10.4.4.1	TCP	74	38571 → 20 [SYN, ACK] Seq=0 Ack=1 Win=65168 Len=0 MSS=1460 SACK_PERM=1
16	13.523351675	10.4.4.1	10.2.2.1	TCP	66	20 → 38571 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=3416477965 TSecr=26
17	13.523352809	10.4.4.1	10.2.2.1	FTP	130	Response: 150 Opening BINARY mode data connection for file1 (224 bytes)
18	13.523353253	10.4.4.1	10.2.2.1	FTP-DATA	290	FTP Data: 224 bytes (PORT) (PORT 10,2,2,1,150,171)
19	13.523353626	10.4.4.1	10.2.2.1	TCP	66	20 → 38571 [FIN, ACK] Seq=225 Ack=1 Win=64256 Len=0 TSval=3416477965 T
20	13.523353927	10.2.2.1	10.4.4.1	TCP	74	[TCP Out-Of-Order] 38571 → 20 [SYN, ACK] Seq=0 Ack=1 Win=65168 Len=0 M
21	13.523646325	10.4.4.1	10.2.2.1	TCP	66	[TCP Dup ACK 16#1] 20 → 38571 [ACK] Seq=226 Ack=1 Win=64256 Len=0 TSval
22	13.529189437	10.2.2.1	10.4.4.1	TCP	66	38571 → 20 [ACK] Seq=1 Ack=225 Win=65024 Len=0 TSval=2626982244 TSecr=
23	13.731652002	10.4.4.1	10.2.2.1	TCP	130	[TCP Retransmission] 21 → 49704 [PSH, ACK] Seq=52 Ack=36 Win=510 Len=6
24	13.732581631	10.2.2.1	10.4.4.1	TCP	66	38571 → 20 [FIN, ACK] Seq=1 Ack=226 Win=65024 Len=0 TSval=2626982448 T
25	13.733143029	10.4.4.1	10.2.2.1	TCP	66	20 → 38571 [ACK] Seq=226 Ack=2 Win=64256 Len=0 TSval=3416478175 TSecr=
26	13.733144626	10.4.4.1	10.2.2.1	FTP	90	Response: 226 Transfer complete.
27	13.736795154	10.2.2.1	10.4.4.1	TCP	78	49704 → 21 [ACK] Seq=36 Ack=116 Win=502 Len=0 TSval=2626982452 TSecr=34
28	13.738254918	10.2.2.1	10.4.4.1	TCP	66	49704 → 21 [ACK] Seq=36 Ack=140 Win=502 Len=0 TSval=2626982454 TSecr=34

Figure 11: Tráfego com aplicação do filtro à porta 20

Na figura acima conseguimos observar um exemplo de uma retransmissão de um pacote que foi detetada. Como o pacote deve ser reenviado por se estar a utilizar o protocolo TCP, as perdas e duplicações acabam por serem facilmente detetadas.

Numa lógica de servidor-cliente, o que aconteceu foi que o servidor (10.4.4.1) enviou dados para o respetivo cliente (10.2.2.1), mas o cliente não deu uma resposta, ACK, no tempo indicado de retransmissão (Retransmission Timer for Server). Com isso, o servidor pensa que o pacote não chegou ao destino, reenviando-o.

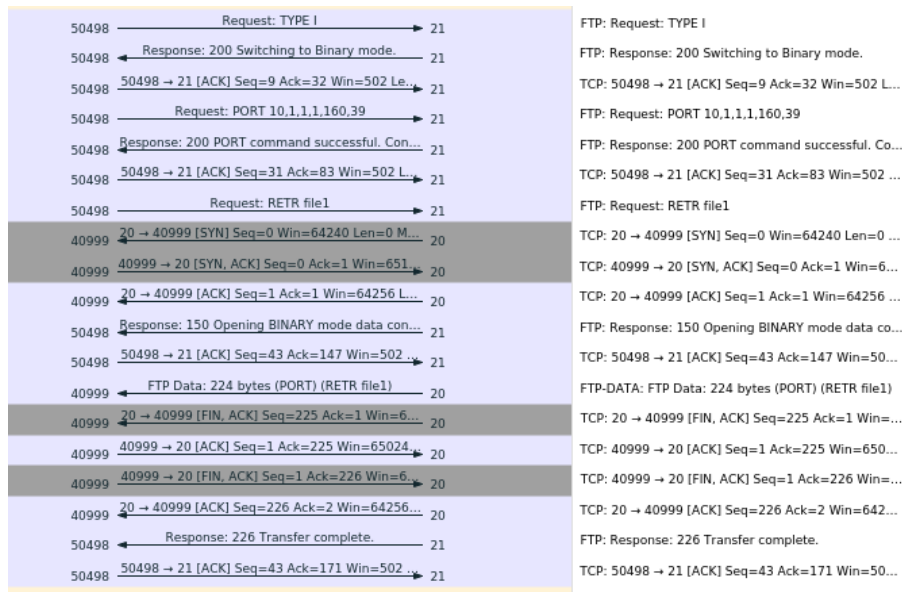


Figure 12: Diagrama temporal da transferência no Portátil 1

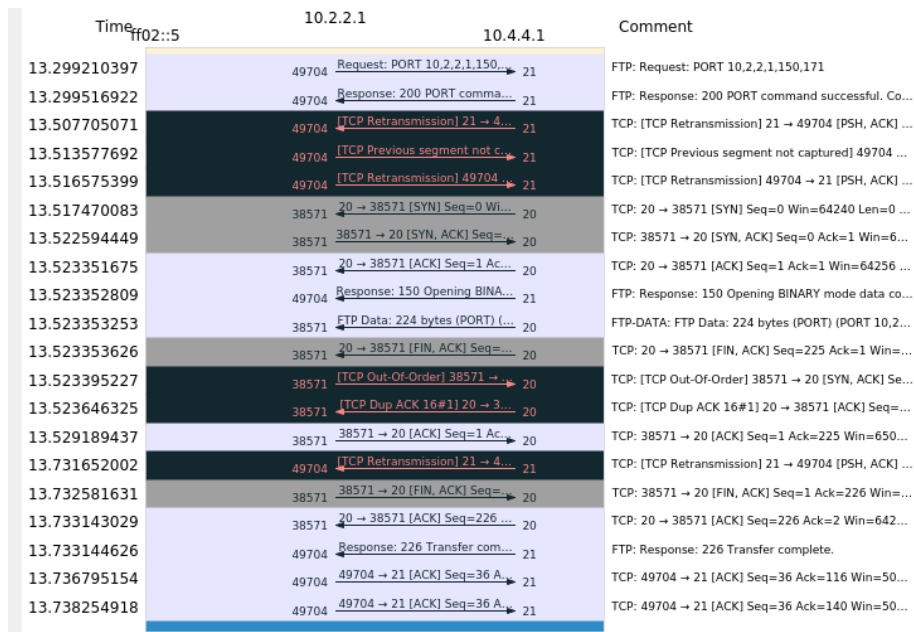


Figure 13: Diagrama temporal da transferência no PC1

### 2.1.3 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.

595	672.468586766	10.1.1.1	10.4.4.1	TFTP	56 Read Request, File: file1, Transfer type: octet
596	672.478246716	10.4.4.1	10.1.1.1	TFTP	270 Data Packet, Block: 1 (last)
597	672.478698119	10.1.1.1	10.4.4.1	TFTP	46 Acknowledgement, Block: 1

Figure 14: Wireshark tftp da transferência no Portátil1

É importante referir que o protocolo UDP não tem número de segmento, nem ACKs, pois estes só existem a nível aplicacional. Na figura acima notamos que, numa primeira fase, o cliente envia um pedido de leitura (*Read Request*) ao servidor, informando-o que pretende transferir o ficheiro. De seguida, o servidor dá início à transferência, enviando o *Data Packet* especificado. Finalmente, o cliente envia um ACK, para informar o servidor que o *packet* anterior foi recebido com sucesso.

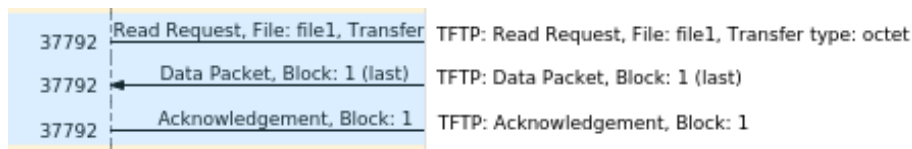


Figure 15: Diagrama temporal da transferência no Portátil1

#### 2.1.4 Questão 4:

*Compare sucintamente as quatro aplicações de transferência de arquivos que usou nos seguintes pontos (i) uso da camada de transporte; (ii) eficiência; (iii) complexidade; (iv) segurança;*

- SFTP : A camada de transporte utilizada é o TCP. Em relação à eficiência e transferência de dados, podemos dizer que é semelhante ao FTP, porém os dados são encriptados, tornando-o mais seguro. Quanto à complexidade, é uma opção fiável gestão de acessos, transferência e gestão de dados. Sendo bastante complexo. Em termos de segurança, utiliza o SSH, garantindo segurança por meio de uma camada de transporte, autenticação e conexão. A camada de transporte é executada com o auxílio do protocolo TCP/IP, fornecendo encriptação, autenticação do servidor e proteção da integridade dos dados. A camada de autenticação é responsável por lidar com a autenticação dos clientes.
- FTP : A camada de transporte utilizada é o TCP. Em relação à eficiência e transferência, é importante observar que os dados não são encriptados, o que torna a transmissão vulnerável à captura por terceiros. No entanto, o protocolo TCP é utilizado para assegurar a transmissão correta dos dados. Quanto à complexidade, esta camada permite a transferência de arquivos em paralelo. Criando uma conexão, resultando em várias velocidades de transferência, tornando-o num sistema bastante complexo. Por fim, em relação à segurança, essa camada não oferece mecanismos de autenticação, o que significa que qualquer pessoa pode capturar pacotes de dados sem restrições.
- TFTP : A camada de transporte utiliza o UDP. Em relação à eficiência e transferência, podemos dizer que é menos confiável, uma vez que utiliza o protocolo UDP, que não envia mensagens de confirmação (*Acknowledgment*) para garantir que os dados foram entregues com sucesso. Quanto à complexidade, devido ao uso do protocolo UDP, a camada é muito simplificada e oferece menos funcionalidades em comparação com outros protocolos. Por fim, em termos de segurança, segue uma abordagem semelhante ao FTP, pois representa uma versão mais simples e não inclui mecanismos avançados de segurança.
- HTTP : A camada de transporte utiliza o protocolo UDP. Em relação à eficiência e transferência, é possível transmitir vários pacotes simultaneamente, o que pode aumentar significativamente a taxa de transferência de dados. É usado o protocolo TCP, o que conduz a um aumento do débito de informação e da fiabilidade da mesma. Sobre a complexidade, é garantida confiança, escalabilidade, e são suportadas as redes heterogêneas e não confiáveis. Por fim, relativamente a segurança, a informação está guardada de uma forma não encriptada, sendo mais fácil de ler e manipular por terceiros.

## 2.2 Uso da camada de transporte por parte das aplicações (Parte II)

### 2.2.1 Questão 1:

Com base no trabalho realizado, tanto na parte I como na parte II, identifique para cada aplicação executada, qual o protocolo de aplicação, o protocolo de transporte, porta de atendimento e overhead de transporte.

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)
wget, lynx ou via browser	DNS	TCP	80	11,05
ssh, sftp	SSH/SFTP	TCP	22	32,78
ftp	FTP	TCP	21	76,92
Tftp	TFTP	UDP	69	50
Telnet	Telnet	TCP	23	42,55
nslookup ou dig	DNS	UDP	53	16
Ping	*	*	*	*
Traceroute	DNS	UDP	53	15,38

\* Como o PING utiliza o protocolo ICMP, não serão retiradas informações.

```

1 0.000000000 10.0.2.15 193.137.16.65 DNS 86 Standard query 0xec09 A cc2023.ddns.net OPT
2 0.000000001 10.0.2.15 193.137.16.65 DNS 16 Standard query 0x11d1 AAAA cc2023.ddns.net CoI
3 0.000000003 193.137.16.0.2.15 DNS 146 Standard query response 0x11d1 AAAA cc2023.ddns.net SOA m1.no-10.com OPT
4 0.106311585 193.137.16.0.2.15 DNS 102 Standard query response 0xec09 A cc2023.ddns.net A 193.136.9.201 OPT
5 0.106490610 10.0.2.15 193.136.9.201 TCP 74 44104 - 22 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=2424272122 TSecr=0 WS=128
6 0.109112374 193.136.10.0.2.15 TCP 60 22 - 44104 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1460
7 0.109177293 10.0.2.15 193.136.9.201 TCP 54 44104 - 22 [ACK] Seq=1 Ack=1 Win=64240 Len=0
8 0.109472122 10.0.2.15 193.136.9.201 SSHv2 95 Client: Protocol (SSH-2.0-OpenSSH_8.2p1 Ubuntu-ubuntu0.3)
9 0.109786642 193.136.10.0.2.15 TCP 60 22 - 44104 [ACK] Seq=1 Ack=42 Win=65535 Len=0
10 0.156651540 193.136.10.0.2.15 SSHv2 95 Server: Protocol (SSH-2.0-OpenSSH_8.9p1 Ubuntu-ubuntu0.4)
11 0.156659824 10.0.2.15 193.136.9.201 TCP 54 44104 - 22 [ACK] Seq=42 Ack=42 Win=64199 Len=0
12 0.157899990 10.0.2.15 193.136.9.201 SSHv2 1566 Client: Key Exchange Init
13 0.157944156 193.136.10.0.2.15 TCP 60 22 - 44104 [ACK] Seq=42 Ack=1562 Win=65535 Len=0
14 0.157944225 193.136.10.0.2.15 TCP 60 22 - 44104 [ACK] Seq=42 Ack=1554 Win=65535 Len=0
15 0.159837339 193.136.10.0.2.15 SSHv2 1134 Server: Key Exchange Init
16 0.162365747 10.0.2.15 193.136.9.201 SSHv2 102 Client: Diffie-Hellman Key Exchange Init
17 0.162666143 193.136.10.0.2.15 TCP 60 22 - 44104 [ACK] Seq=1122 Ack=1602 Win=65535 Len=0
18 0.173724296 193.136.10.0.2.15 SSHv2 650 Server: Diffie-Hellman Key Exchange Reply, New Keys, Encrypted packet (len=316)
19 0.176902626 10.0.2.15 193.136.9.201 SSHv2 70 Client: New Keys
20 0.176253432 193.136.10.0.2.15 TCP 60 22 - 44104 [ACK] Seq=1718 Ack=1618 Win=65535 Len=0
21 0.176606814 10.0.2.15 193.136.9.201 SSHv2 96 Client: Encrypted packet (len=411)
Frame 2: 86 bytes on wire (688 bits), 86 bytes captured (688 bits) on interface enp0s3, id 0
Ethernet II, Src: PcsCompu, 08:00:40:08:00:27, Dst: RealtekU, 12:35:02 (12:34:00:12:35:02)
Internet Protocol Version 4, Src: 10.0.2.15, Dst: 193.137.16.65
0100 ..... = Version: 4
0111 = Header Length: 20 bytes (5)
Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
Total Length: 72
Identification: 0x4dad (19678)
Flags: 0x0000, Don't Fragment
Fragment offset: 0
Time to live: 64
Protocol: UDP (17)
Header checksum: 0x6f26 [validation disabled]
[Header checksum status: Unverified]
Source: 10.0.2.15
Destination: 193.137.16.65
User Datagram Protocol, Src Port: 51464, Dst Port: 53
Domain Name System (query)

```

Figure 16: Informação sobre o comando Traceroute

**Traceroute:** Observando a figura, em *User Datagram Protocol*, reparamos que a porta de atendimento, ou seja, porta de destino é a 53 e que o *Total Length* é 72. Assim sendo, o *overhead* é dado por  $8 / (72 - 20) * 100 = 15,38$ .

5	0.043899119	10.0.2.15	193.136.9.240	TCP	74	33542 → 80	[SYN]	Seq=0	Win=64240	Len=0	MSS=1460	SACK_PERM=1	TSval=1532720629	TSecr=0	WS=128
6	0.046148394	193.136...	10.0.2.15	TCP	60	80 → 33542	[SYN, ACK]	Seq=0	Ack=1	Win=65535	Len=0	MSS=1460			
7	0.046166228	10.0.2.15	193.136.9.240	TCP	54	33542 → 80	[ACK]	Seq=1	Ack=1	Win=64240	Len=0				
8	0.046166351	10.0.2.15	193.136.9.240	TCP	215	80 → 33542	[ACK]	Seq=1	Ack=162	Win=65535	Len=0				
9	0.046095731	193.136...	10.0.2.15	TCP	60	80 → 33542	[ACK]	Seq=1	Ack=162	Win=65535	Len=0				
10	0.050432574	193.136...	10.0.2.15	TCP	2974	80 → 33542	[ACK]	Seq=1	Ack=162	Win=65535	Len=2920				[TCP segment of a reassembled PDU]
11	0.050443276	10.0.2.15	193.136.9.240	TCP	54	33542 → 80	[ACK]	Seq=162	Ack=2921	Win=62780	Len=0				
12	0.050533345	193.136...	10.0.2.15	TCP	4434	80 → 33542	[ACK]	Seq=2921	Ack=162	Win=65535	Len=4380				[TCP segment of a reassembled PDU]
13	0.050543515	10.0.2.15	193.136.9.240	TCP	54	33542 → 80	[ACK]	Seq=162	Ack=7301	Win=59860	Len=0				
▶ Frame 8: 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 0100 .... = Version: 4 00000000 = Header Length: 20 bytes (5)															
▶ Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT) Total Length: 201 Identification: 0xc42f (58223) ▶ Flags: 0x4000, Don't fragment Fragment offset: 0 Time to live: 64 Protocol: TCP (6) Header checksum: 0x9e78 [validation disabled] [Header checksum status: Unverified] Source: 10.0.2.15 Destination: 193.136.9.240															
▶ Transmission Control Protocol, Src Port: 33542, Dst Port: 80, Seq: 1, Ack: 1, Len: 161 Source Port: 33542 Destination Port: 80 [Stream index: 0] [TCP Segment Len: 161] Sequence number: 1 (relative sequence number) Sequence number (raw): 3122966283 [Next sequence number: 162 (relative sequence number)] Acknowledgment number: 1 (relative ack number) Acknowledgment number (raw): 164224002 0101 .... = Header Length: 20 bytes (5) ▶ Flags: 0x018 (PSH, ACK) Window size value: 64240 [Calculated window size: 64240] [Window size scaling factor: -2 (no window scaling used)] Checksum: 0x0842 [unverified] [Checksum Status: Unverified] Urgent pointer: 0 ▶ [SEQ/ACK analysis] ▶ [Timestamps] TCP payload (161 bytes)															
▶ Hypertext Transfer Protocol															

Figure 17: Informação sobre o comando wget

**Browser/HTTP:** Observando a figura, em *User Datagram Protocol*, reparamos que a porta de atendimento, ou seja, porta de destino é a 80 e que o *Total Lenght* é 201. Assim sendo, o *overhead* é dado por  $20/(201 - 20) * 100 = 11,05$

18 2.821273828	193.137...	10.0.2.15	FTP	88 Response: 331 Please specify the password.
19 2.821282581	10.0.2.15	193.137.214...	TCP	54 40056 → 21 [ACK] Seq=11 Ack=55 Win=64186 Len=0
20 4.765953578	10.0.2.15	193.137.214...	FTP	67 Request: PASS cc2023
21 4.766382128	193.137...	10.0.2.15	TCP	60 21 → 40056 [ACK] Seq=55 Ack=24 Win=65535 Len=0
22 4.776696695	193.137...	10.0.2.15	FTP	465 Response: 230-Welcome, archive user of ftp.eq.uc.pt!
23 4.776725021	10.0.2.15	193.137.214...	TCP	54 40056 → 21 [ACK] Seq=24 Ack=466 Win=63784 Len=0
24 4.776914113	10.0.2.15	193.137.214...	FTP	60 Request: SYST
25 4.777462434	193.137...	10.0.2.15	TCP	60 21 → 40056 [ACK] Seq=466 Ack=30 Win=65535 Len=0
26 4.782596126	193.137...	10.0.2.15	FTP	73 Response: 215 UNIX Type: L8
▶ Frame 24: 60 bytes on wire (480 bits), 60 bytes captured (480 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: 193.137.214.36 0100 ..... = Version: 4 0101 = Header Length: 20 bytes (5) ▶ Differentiated Services Field: 0x10 (DSCP: Unknown, ECN: Not-ECT) Total Length: 46 Identification: 0xacf0 (44272) ▶ Flags: 0x4000, Don't fragment Fragment offset: 0 Time to live: 64 Protocol: TCP (6) Header checksum: 0xea0c [validation disabled] [Header checksum status: Unverified] Source: 10.0.2.15 Destination: 193.137.214.36 ▶ Transmission Control Protocol, Src Port: 40056, Dst Port: 21, Seq: 24, Ack: 466, Len: 6 Source Port: 40056 Destination Port: 21 [Stream index: 1] [TCP Segment Len: 6] Sequence number: 24 (relative sequence number) Sequence number (raw): 2252140502 [Next sequence number: 30 (relative sequence number)] Acknowledgment number: 466 (relative ack number) Acknowledgment number (raw): 124480467 0101 ..... = Header Length: 20 bytes (5) ▶ Flags: 0x018 (PSH, ACK) Window size value: 63784 [Calculated window size: 63784] [Window size scaling factor: -2 (no window scaling used)] Checksum: 0xa3dd [unverified] [Checksum Status: Unverified] Urgent pointer: 0 ▶ [SEQ/ACK analysis] ▶ [Timestamps] TCP payload (6 bytes) ▶ File Transfer Protocol (FTP) [Current working directory: ]				

Figure 18: Informação sobre o comando FTP

**FTP:** Observando a figura, em *Transmission Control Protocol*, reparamos que a porta de atendimento, ou seja, porta de destino é a 21 e que o Total *Lenght* é 46. A porta de origem é a 21, que é normalmente a utilizada por este protocolo. Assim sendo, o *overhead* é dado por  $20/(46 - 20) * 100 = 76,92$ .



```

5 0.000105393 10.0.2.15 193.136.9.201 [IP] 66 Read Request File: file1, transfer type: octet, tsize=0, blksize=512, timeout=0
6 2.776466644 10.0.2.15 162.247.243... TLSv1.2 1066 Application Data
7 2.777039568 162.247... 10.0.2.15 TCP 60 443 - 46558 [ACK] Seq=1 Ack=1013 Win=65535 Len=0
8 2.884948028 162.247... 10.0.2.15 TLSv1.2 441 Application Data, Application Data
9 2.884966299 10.0.2.15 162.247.243... TCP 54 46558 - 443 [ACK] Seq=1013 Ack=388 Win=62780 Len=0

> Frame 5: 86 bytes on wire (688 bits), 86 bytes captured (688 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: 193.136.9.201
  0100 .... = Version: 4
  .... 0101 = Header Length: 20 bytes (5)
  > Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
    Total Length: 72
    Identification: 0x843f (33855)
    > Flags: 0x4000, Don't fragment
    Fragment offset: 0
    Time to live: 64
    Protocol: UDP (17)
    Header checksum: 0xdf05 [validation disabled]
    [Header checksum status: Unverified]
    Source: 10.0.2.15
    Destination: 193.136.9.201
  > User Datagram Protocol, Src Port: 56012, Dst Port: 69
    Source Port: 56012
    Destination Port: 69
    Length: 52
    Checksum: 0xd7a5 [unverified]
    [Checksum Status: Unverified]
    [Stream index: 2]
    [Timestamps]
  > Trivial File Transfer Protocol

```

Figure 19: Informação sobre o comando TFTP

**TFTP:** Observando a figura, em *Transmission Control Protocol*, reparamos que a porta de atendimento, ou seja, porta de destino é a 69 e que o Total Length é 72. Assim sendo, *overhead* é dado por  $8 / (72 - 20) * 100 = 50$ .

25	3.769004631	10.0.2.15	193.136.9.33	TELNET	81 Telnet Data ...
26	3.769431962	193.136...	10.0.2.15	TCP	60 23 → 60260 [ACK] Seq=1 Ack=28 Win=65535 Len=0
27	3.774293607	193.136...	10.0.2.15	TELNET	66 Telnet Data ...
▶ Frame 25: 81 bytes on wire (648 bits), 81 bytes captured (648 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: 193.136.9.33 0100 .... = Version: 4 .... 0101 = Header Length: 20 bytes (5) ▶ Differentiated Services Field: 0x10 (DSCP: Unknown, ECN: Not-ECT) Total Length: 67 Identification: 0x735f (29535) ▶ Flags: 0x4000, Don't fragment Fragment offset: 0 Time to live: 64 Protocol: TCP (6) Header checksum: 0xf08d [validation disabled] [Header checksum status: Unverified] Source: 10.0.2.15 Destination: 193.136.9.33 ▶ Transmission Control Protocol, Src Port: 60260, Dst Port: 23, Seq: 1, Ack: 1, Len: 27 Source Port: 60260 Destination Port: 23 [Stream index: 1] [TCP Segment Len: 27] Sequence number: 1 (relative sequence number) Sequence number (raw): 1581687970 [Next sequence number: 28 (relative sequence number)] Acknowledgment number: 1 (relative ack number) Acknowledgment number (raw): 31936002 0101 .... = Header Length: 20 bytes (5) ▶ Flags: 0x018 (PSH, ACK) Window size value: 64240 [Calculated window size: 64240] [Window size scaling factor: -2 (no window scaling used)] Checksum: 0xd6ed [unverified] [Checksum Status: Unverified] Urgent pointer: 0 ▶ [SEQ/ACK analysis] ▶ [Timestamps] ▶ TCP payload (27 bytes) ▶ Telnet					

Figure 20: Informação sobre o comando Telnet

**Telnet:** Observando a figura, em *Transmission Control Protocol*, reparamos que a porta de atendimento, ou seja, porta de destino é a 23 e que o Total Length é 67. Assim sendo, o *overhead* é dado por  $20/(67 - 20) * 100 = 42,55$ .

14	2.586919631	10.0.2.15	193.136.9.201	SSHv2	95 Client: Protocol (SSH-2.0-OpenSSH.8.zpl Ubuntu-4ubuntu0.3)
15	2.587354440	193.136...	10.0.2.15	TCP	60 22 → 58610 [ACK] Seq=1 Ack=42 Win=65535 Len=0
16	2.621506370	193.136...	10.0.2.15	SSHv2	95 Server: Protocol (SSH-2.0-OpenSSH.8.9p1 Ubuntu-3ubuntu0.4)
17	2.621520282	10.0.2.15	193.136.9.201	TCP	54 58610 → 22 [ACK] Seq=42 Ack=42 Win=64199 Len=0
18	2.622086720	10.0.2.15	193.136.9.201	SSHv2	1566 Client: Key Exchange Init
19	2.622368731	193.136...	10.0.2.15	TCP	60 22 → 58610 [ACK] Seq=42 Ack=1502 Win=65535 Len=0
20	2.622368847	193.136...	10.0.2.15	TCP	60 22 → 58610 [ACK] Seq=42 Ack=1554 Win=65535 Len=0

▶ Frame 14: 95 bytes on wire (760 bits), 95 bytes captured (760 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: 193.136.9.201 0100 .... = Version: 4 0101 = Header Length: 20 bytes (5) ▶ Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT) Total Length: 81 Identification: 0xffff (65531) ▶ Flags: 0x4000, Don't fragment Fragment offset: 0 Time to live: 64 Protocol: TCP (6) Header checksum: 0x634b [validation disabled] [Header checksum status: Unverified] Source: 10.0.2.15 Destination: 193.136.9.201 ▶ Transmission Control Protocol, Src Port: 58610, Dst Port: 22, Seq: 1, Ack: 1, Len: 41 Source Port: 58610 Destination Port: 22 [Stream index: 1] [TCP Segment Len: 41] Sequence number: 1 (relative sequence number) Sequence number (raw): 2249833734 [Next sequence number: 42 (relative sequence number)] Acknowledgment number: 1 (relative ack number) Acknowledgment number (raw): 44096002 0101 .... = Header Length: 20 bytes (5) ▶ Flags: 0x018 (PSH, ACK) Window size value: 64240 [Calculated window size: 64240] [Window size scaling factor: -2 (no window scaling used)] Checksum: 0xd7a3 [unverified] [Checksum Status: Unverified] Urgent pointer: 0 ▶ [SEQ/ACK analysis] ▶ [Timestamps] TCP payload (41 bytes) ▶ SSH Protocol
--

Figure 21: Informação sobre o comando SSH

**SSH:** Observando a figura, em *Transmission Control Protocol*, reparamos que a porta de atendimento, ou seja, porta de destino é a 22 e que o Total *Lenght* é 81. Assim sendo, o *overhead* é dado por  $20 / (81 - 20) * 100 = 32,78$ .

5 1.161963965	193.137.16.65	DNS	54 Standard query 0xbe50 A 222.uminho.pt OPT
4 1.165795540	193.137.16.65	DNS	147 Standard query response 0xbe50 No such name A 222.uminho.pt SOA dns.uminho.pt OPT
5 1.165889299	193.137.16.65	DNS	73 Standard query 0xbe50 A 222.uminho.pt
6 1.170881554	193.137.16.65	DNS	136 Standard query response 0xbe50 No such name A 222.uminho.pt SOA dns.uminho.pt
7 1.999435234	193.136.152.1	NTP	90 NTP Version 4, client
8 2.006632100	193.136.152.1	NTP	90 NTP Version 4, server

Frame 3: 84 bytes on wire (672 bits), 84 bytes captured (672 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: 193.137.16.65, Dst: 193.137.16.65
0100 .... = Version: 4
0000 0000 0000 0000 = Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
Total Length: 70
Identification: 0x8675 (1653)
Flags: 0x4000, Don't Fragment
Fragment offset: 0
Time to live: 64
Protocol: UDP (17)
Header checksum: 0x5650 [validation disabled]
[Header checksum status: Unverified]
Source: 193.137.16.65
Destination: 193.137.16.65
User Datagram Protocol, Src Port: 59798, Dst Port: 53
Source Port: 59798
Destination Port: 53
Length: 50
Checksum: 0xdec [unverified]
[Checksum Status: Unverified]
[Stream index: 1]
[Timestamps]
Domain Name System (query)

Figure 22: Informação sobre o comando SSH

**Nslookup:** Observando a figura, em *Transmission Control Protocol*, reparamos que a porta de atendimento, ou seja, porta de destino é a 53 e que o *Total Length* é 70. Assim sendo, o *overhead* é dado por  $8/(70 - 20) * 100 = 16$ .