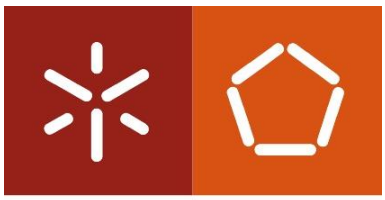


Universidade do Minho
Licenciatura em Engenharia Informática

Comunicações por Computador

Trabalho Prático 1
Protocolos de Camada de Transporte

Grupo 49


António Fonseca
a93167


Diogo Rebelo
a93278


Henrique Alvelos
a93316

30 de outubro de 2021

Conteúdo

Lista de Figuras

Lista de Tabelas

1 Introdução

O presente relatório tenta explorar de uma forma mais profunda alguns conceitos primordiais da Camada de Transporte e de Aplicação e de que forma a primeira se relaciona com a segunda. Naturalmente, tendo sempre como base princípios teóricos, socorremo-nos de uma abordagem mais prática e com o auxílio de outros programas para o realizar com sucesso. Assim sendo, como forma de responder a essa necessidade, este relatório compreende um conjunto de objetivos fundamentais:

- Compreender os protocolos da Camada de Transporte da Internet (UDP & TCP);
- Compreender os princípios subjacentes aos serviços da camada de transporte;
- Testar a conectividade e analisar as características gerais dos links disponibilizados (ligações com diferentes larguras de banda e diferentes atrasos), utilizando o comando "ping" e/ou "traceroute";
- Utilizar o *Wireshark* por forma a obter capturas de tráfego que ajudem a compreender o processo de comunicação (transferência de ficheiros);
- Comparar tempos de transferência de ficheiros entre o Servidor e diferentes clientes.
- Ganhar um maior traquejo em relação à utilização da *Virtual Box* e *Core*.

2 Questões e Respostas

Nesta secção constam todas as questões colocadas e respetivas respostas. Optamos por introduzir alguma informação que consideramos relevante para uma compreensão bem conseguida.

Comece-se, assim sendo, pela **Parte I**.

2.1 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.

2.1.1 Pacotes como unidades de dados e efeitos da sua perda e/ou duplicação

Os pacotes constituem unidades de dados que nem sempre conseguem chegar ao seu endereço de destino, podendo-se mesmo perder na respetiva transferência. Esta perda pode naturalmente afetar as aplicações, todavia, tem mais impacto em aplicações que se socorrem de transferências de dados em tempo real. Mesmo assim, perdas regulares geralmente não têm muito impacto no desempenho geral da rede, mas muitas perdas de pacotes são definitivamente um problema. O aumento de tráfego, os bugs de software, problemas de hardware e software de rede (dispositivos desatualizados) e violações de segurança (ataques e ameaças à rede) são as principais razões pelas quais estas perdas acontecem.

2.1.2 Camada de transporte e efeito consoante o protocolo TCP ou UDP

A camada responsável por lidar com este tipo de perdas e duplicações é a camada de Transporte, com o seu protocolo confiável TCP. Os efeitos destas perdas podem ser analisados consoante o protocolo ou aplicação em questão. Com efeito, são os protocolos TCP (Protocolo de Controlo de Transmissão) e UDP (Protocolo de Datagrama de Usuário) que procuram resolver este problema. Então, considerando as próprias características dos protocolos, é possível inferir o efeito destas perdas e/ou duplicações:

- Um protocolo de transmissão (TCP) é normalmente responsável por resolver estas perdas, sendo orientado à conexão e responsável pelo controlo de fluxo, avalia, identifica e retransmite os pacotes (controlo de erros), perante algum inconveniente.
- Um protocolo de Usuário (UDP), não possuindo mecanismos de controlo de erros (apenas de deteção), descarta diretamente o pacote em vez de o reenviar. Protocolos superiores terão de ser responsáveis por reenviar estes pacotes para que não se percam. No UDP, não há segmentação ou ACKs.

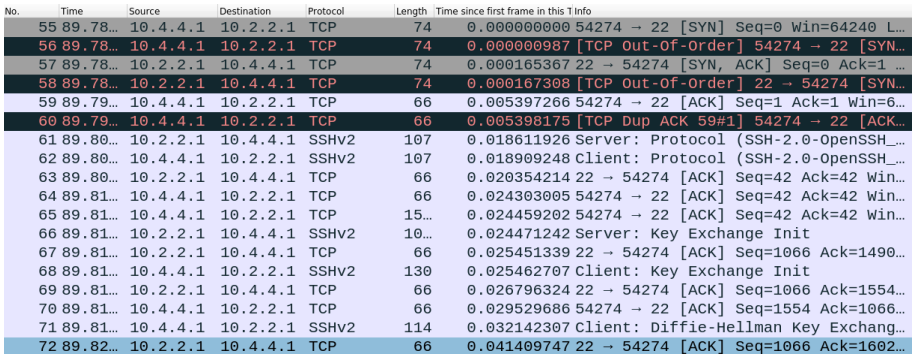
2.1.3 Análise de resultados

Como forma de análise de resultados, podemos utilizar a captura do *Wireshark* relativa à transferência do file 1 no Portátil 1 e no Grilo.

No.	Time	Source	Destination	Protocol	Length	Time since first frame in this T	Info
70	67.19…	10.4.4.1	10.2.2.1	TCP	74	0.005351029	48109 → 20 [SYN, ACK] Seq=0 Ack=1 …
71	67.19…	10.2.2.1	10.4.4.1	TCP	66	0.005516490	20 → 48109 [ACK] Seq=1 Ack=1 Win=6…
72	67.19…	10.2.2.1	10.4.4.1	FTP	105	42.532508956	Response: 150 Here comes the direc…
73	67.19…	10.2.2.1	10.4.4.1	FTP-DATA	192	0.006343186	FTP Data: 126 bytes (PORT) (LIST)
74	67.19…	10.2.2.1	10.4.4.1	TCP	66	0.006345903	20 → 48109 [FIN, ACK] Seq=127 Ack=…
75	67.19…	10.4.4.1	10.2.2.1	TCP	66	42.537609048	55582 → 21 [ACK] Seq=87 Ack=221 Wi…
76	67.19…	10.4.4.1	10.2.2.1	TCP	66	0.011410441	48109 → 20 [ACK] Seq=1 Ack=127 Win…
77	67.19…	10.4.4.1	10.2.2.1	TCP	66	0.011444311	48109 → 20 [FIN, ACK] Seq=1 Ack=12…
78	67.19…	10.2.2.1	10.4.4.1	TCP	66	0.011580613	20 → 48109 [ACK] Seq=128 Ack=2 Win…
79	67.19…	10.2.2.1	10.4.4.1	FTP	90	42.538458668	Response: 226 Directory send OK.
81	67.40…	10.2.2.1	10.4.4.1	TCP	90	42.746792916	[TCP Retransmission] 21 → 55582 [P…
82	67.41…	10.4.4.1	10.2.2.1	TCP	78	42.752045042	55582 → 21 [ACK] Seq=87 Ack=245 Wi…
87	74.87…	10.4.4.1	10.2.2.1	FTP	74	50.211627705	Request: TYPE I
88	74.87…	10.2.2.1	10.4.4.1	FTP	97	50.211813079	Response: 200 Switching to Binary …
89	74.87…	10.4.4.1	10.2.2.1	TCP	66	50.217135374	55582 → 21 [ACK] Seq=95 Ack=276 Wi…
90	74.87…	10.4.4.1	10.2.2.1	FTP	89	50.217136100	Request: PORT 10,4,4,1,132,117
91	74.87…	10.2.2.1	10.4.4.1	FTP	117	50.217373956	Response: 200 PORT command success…
92	74.88…	10.4.4.1	10.2.2.1	TCP	66	50.222477279	55582 → 21 [ACK] Seq=118 Ack=327 W…

Figura 1: Excerto da retransmissão de pacote no Grilo da LAN4 com FTP

A figura acima contém um exemplo de uma retransmissão de um pacote que foi detetada pelo programa. Como se está a utilizar o protocolo TCP as perdas e duplicações são facilmente detetadas, pois o pacote tem de, em algum, momento ser reenviado. Numa lógica de servidor-cliente, o que aconteceu foi que o servidor (10.2.2.1) enviou dados para o respetivo cliente (10.4.4.1), mas o cliente não deu uma resposta, *Acknowledgement* (ACK), no tempo indicado de retransmissão (*Retransmission Timer for Server*). Tal faz com que o servidor pense que o pacote não chegou ao destino, reenviando-o.



No.	Time	Source	Destination	Protocol	Length	Time since first frame in this T	Info
55	89.78…	10.4.4.1	10.2.2.1	TCP	74	0.000000000	54274 → 22 [SYN] Seq=0 Win=64240 L…
56	89.78…	10.4.4.1	10.2.2.1	TCP	74	0.000000987	[TCP Out-Of-Order] 54274 → 22 [SYN…
57	89.78…	10.2.2.1	10.4.4.1	TCP	74	0.000165367	22 → 54274 [SYN, ACK] Seq=0 Ack=1 …
58	89.78…	10.2.2.1	10.4.4.1	TCP	74	0.000167308	[TCP Out-Of-Order] 22 → 54274 [SYN…
59	89.79…	10.4.4.1	10.2.2.1	TCP	66	0.005397266	54274 → 22 [ACK] Seq=1 Ack=1 Win=6…
60	89.79…	10.4.4.1	10.2.2.1	TCP	66	0.005398175	[TCP Dup ACK 59#1] 54274 → 22 [ACK…
61	89.80…	10.2.2.1	10.4.4.1	SSHv2	107	0.018611926	Server: Protocol (SSH-2.0-OpenSSH_…
62	89.80…	10.4.4.1	10.2.2.1	SSHv2	107	0.018909248	Client: Protocol (SSH-2.0-OpenSSH_…
63	89.80…	10.2.2.1	10.4.4.1	TCP	66	0.020354214	22 → 54274 [ACK] Seq=42 Ack=42 Win…
64	89.81…	10.4.4.1	10.2.2.1	TCP	66	0.024303005	54274 → 22 [ACK] Seq=42 Ack=42 Win…
65	89.81…	10.4.4.1	10.2.2.1	TCP	15…	0.024459202	54274 → 22 [ACK] Seq=42 Ack=42 Win…
66	89.81…	10.2.2.1	10.4.4.1	SSHv2	10…	0.024471242	Server: Key Exchange Init
67	89.81…	10.2.2.1	10.4.4.1	TCP	66	0.025451339	22 → 54274 [ACK] Seq=1066 Ack=1490…
68	89.81…	10.4.4.1	10.2.2.1	SSHv2	130	0.025462707	Client: Key Exchange Init
69	89.81…	10.2.2.1	10.4.4.1	TCP	66	0.026796324	22 → 54274 [ACK] Seq=1066 Ack=1554…
70	89.81…	10.4.4.1	10.2.2.1	TCP	66	0.029529686	54274 → 22 [ACK] Seq=1554 Ack=1066…
71	89.81…	10.4.4.1	10.2.2.1	SSHv2	114	0.032142307	Client: Diffie-Hellman Key Exchang…
72	89.82…	10.2.2.1	10.4.4.1	TCP	66	0.041409747	22 → 54274 [ACK] Seq=1066 Ack=1602…

Figura 2: Excerto da duplicação de pacote no Grilo da LAN4 com SFTP

Olhando, agora, para a figura 2, é notória uma duplicação de pacote e vários pacotes *Out-of-Order*. Este último estado quer apenas dizer que os pacotes foram enviados numa ordem diferente daquela pela qual foram recebidos ou vice-versa. Contudo, neste caso, temos o mesmo pacote a ser enviado por múltiplos caminhos: é dado o *SYN (Synchronize)* da porta 54274 para a porta 2 repetidamente e acontece, consequentemente, o mesmo nos *ACKs*. Há, assim, *ACKs* duplicados e estes significam tipicamente perda de pacotes, sendo a comunicação menos eficiente.

Ainda em relação à duplicação de pacotes, reparamos que esta surge na realização do comando *PING* entre o Servidor e o Grilo da LAN4 (figura 3). Em relação ao Portátil 1, o PING observado é muito menor (figura 4).

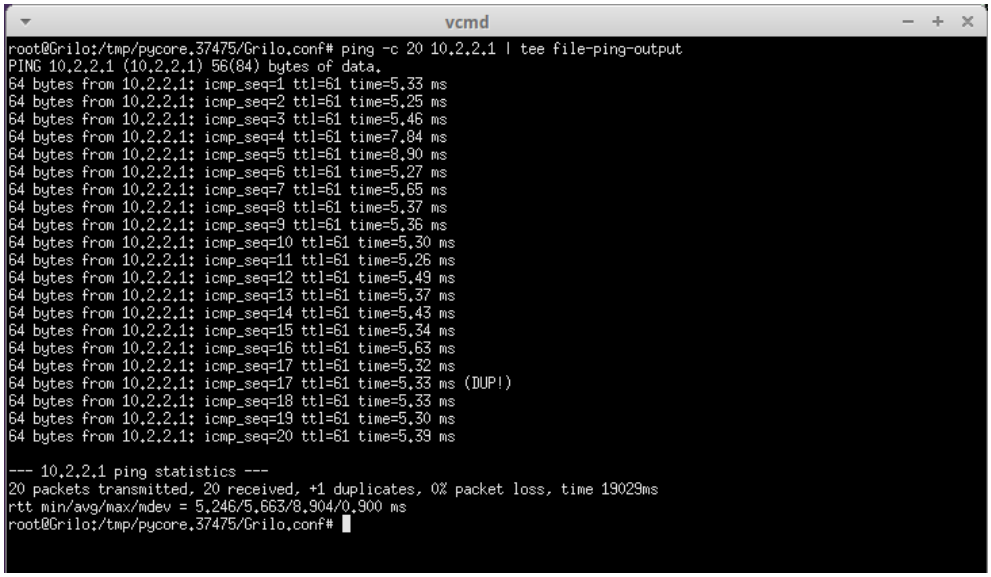


Figura 3: Duplicação de pacote entre o Servidor e o Grilo da LAN4, com o comando PING

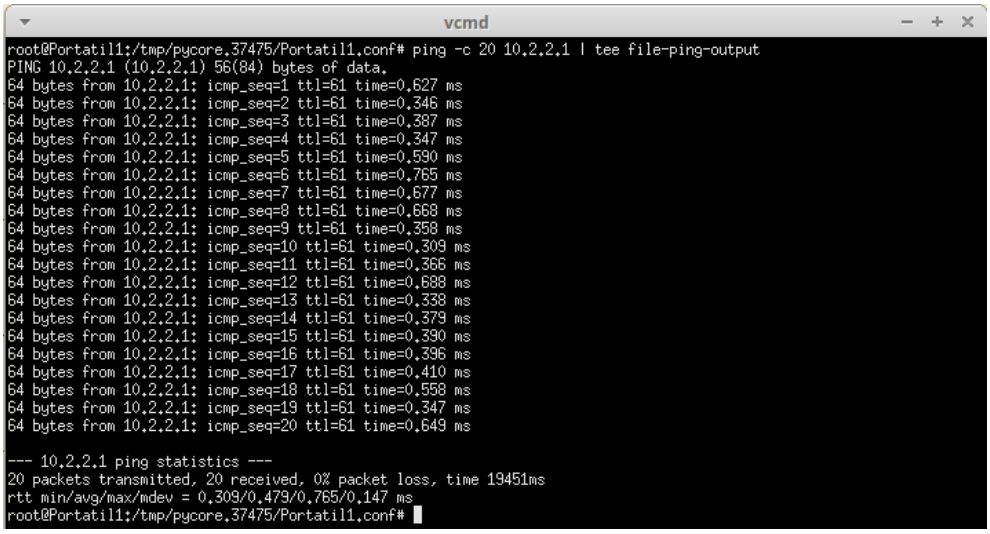


Figura 4: Duplicação de pacote entre o Servidor e o Portátil 1, com o comando PING

2.2 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.

2.2.1 Porta 20 vs Porta 21

O FTP usa dois números de porta TCP bem conhecidos, a porta 20 e a porta 21. A primeira é usada na transferência de dados (envio de ficheiros, no nosso caso), já a segunda é usada para o controlo da conexão (porta de comando).

2.2.2 Diagramas de fluxo da transferência do file 1 por FTP

Para que uma transferência (utilizando FTP) ocorra entre o servidor e cliente, primeiro, o cliente deve pedir acesso ao servidor, normalmente, autenticando-se com um user e password para efeitos de segurança. De seguida, inicia-se a fase inicial de conexão: geralmente, o cliente envia um *TCP SYN* ao servidor, que lhe responde com um *TCP SYN/ACK*. Depois, o cliente envia um *TCP ACK*. A conexão está assim iniciada e o ficheiro pode ser transmitido. Quando a transferência do ficheiro termina, é enviada uma mensagem de "transferência completa"para o cliente, iniciando-se a fase final da conexão: o servidor, utiliza um *TCP FIN* para indicar que o ficheiro foi transferido e para informar que a conexão vai terminar. Então, o cliente responde com um *TCP ACK*. O servidor termina graciosamente a sua conexão, contudo, o cliente deve fazer o mesmo, pelo que envia também um *TCP FIN*, ao qual o servidor responde com um *TCP ACK*. Termina assim a ligação.

No.	Time	Source	Destination	Protocol	Length	Info
88	88.1420…	10.1.1.1	10.2.2.1	TCP	66	41906 → 21 [ACK] Seq=87 Ack=221 Win=64256 Len=0…
89	88.1450…	10.2.2.1	10.1.1.1	TCP	66	20 → 51181 [FIN, ACK] Seq=127 Ack=1 Win=64256 L…
90	88.1451…	10.1.1.1	10.2.2.1	TCP	66	51181 → 20 [ACK] Seq=1 Ack=127 Win=65152 Len=0 …
91	88.1454…	10.1.1.1	10.2.2.1	TCP	66	51181 → 20 [FIN, ACK] Seq=1 Ack=128 Win=65152 L…
92	88.1457…	10.2.2.1	10.1.1.1	TCP	66	20 → 51181 [ACK] Seq=128 Ack=2 Win=64256 Len=0 …
93	88.1457…	10.2.2.1	10.1.1.1	FTP	90	Response: 226 Directory send OK.
94	88.1463…	10.1.1.1	10.2.2.1	TCP	66	41906 → 21 [ACK] Seq=87 Ack=245 Win=64256 Len=0…
107	108.164…	10.1.1.1	10.2.2.1	FTP	74	Request: TYPE I
108	108.164…	10.2.2.1	10.1.1.1	FTP	97	Response: 200 Switching to Binary mode.
109	108.165…	10.1.1.1	10.2.2.1	TCP	66	41906 → 21 [ACK] Seq=95 Ack=276 Win=64256 Len=0…
110	108.165…	10.1.1.1	10.2.2.1	FTP	89	Request: PORT 10,1,1,1,232,121
111	108.165…	10.2.2.1	10.1.1.1	FTP	117	Response: 200 PORT command successful. Consider…
112	108.166…	10.1.1.1	10.2.2.1	TCP	66	41906 → 21 [ACK] Seq=118 Ack=327 Win=64256 Len=…
113	108.166…	10.1.1.1	10.2.2.1	FTP	78	Request: RETR file1
114	108.166…	10.2.2.1	10.1.1.1	TCP	74	20 → 59513 [SYN] Seq=0 Win=64240 Len=0 MSS=1460…
115	108.166…	10.1.1.1	10.2.2.1	TCP	74	59513 → 20 [SYN, ACK] Seq=0 Ack=1 Win=65160 Len…
116	108.166…	10.2.2.1	10.1.1.1	TCP	66	20 → 59513 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TS…
117	108.166…	10.2.2.1	10.1.1.1	FTP	130	Response: 150 Opening BINARY mode data connecti…
118	108.167…	10.2.2.1	10.1.1.1	FTP-DATA	290	FTP Data: 224 bytes (PORT) (RETR file1)
119	108.167…	10.2.2.1	10.1.1.1	TCP	66	20 → 59513 [FIN, ACK] Seq=225 Ack=1 Win=64256 L…
120	108.167…	10.1.1.1	10.2.2.1	TCP	66	41906 → 21 [ACK] Seq=130 Ack=391 Win=64256 Len=…
121	108.167…	10.1.1.1	10.2.2.1	TCP	66	59513 → 20 [ACK] Seq=1 Ack=225 Win=65024 Len=0 …
122	108.167…	10.1.1.1	10.2.2.1	TCP	66	59513 → 20 [FIN, ACK] Seq=1 Ack=226 Win=65024 L…
123	108.167…	10.2.2.1	10.1.1.1	TCP	66	20 → 59513 [ACK] Seq=226 Ack=2 Win=64256 Len=0 …
124	108.167…	10.2.2.1	10.1.1.1	FTP	90	Response: 226 Transfer complete.
125	108.168…	10.1.1.1	10.2.2.1	TCP	66	41906 → 21 [ACK] Seq=130 Ack=415 Win=64256 Len=…
129	112.789…	10.1.1.1	10.2.2.1	FTP	72	Request: QUIT
130	112.790…	10.2.2.1	10.1.1.1	FTP	80	Response: 221 Goodbye.
131	112.790…	10.2.2.1	10.1.1.1	TCP	66	21 → 41906 [FIN, ACK] Seq=429 Ack=136 Win=65280…
132	112.790…	10.1.1.1	10.2.2.1	TCP	66	41906 → 21 [ACK] Seq=136 Ack=429 Win=64256 Len=…
133	112.790…	10.1.1.1	10.2.2.1	TCP	66	41906 → 21 [FIN, ACK] Seq=136 Ack=430 Win=64256…
134	112.791…	10.2.2.1	10.1.1.1	TCP	66	21 → 41906 [ACK] Seq=430 Ack=137 Win=65280 Len=…

Figura 5: Diferentes fases de conexão (sem filtro na porta 20)

Na figura anterior não se aplicou o filtro em relação à porta 20, 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 [ftp-data]. As fases descritas em pormenor na secção 2.2.2 surgem abaixo bem explícitas: é possível ver o inicio e o fim da conexão TCP.

No.	Time	Source	Destination	Protocol	Length	Info
83	88.14…	10.2.2.1	10.1.1.1	TCP	74	20 → 51181 [SYN] Seq=0 Win=64240 Len=0…
84	88.14…	10.1.1.1	10.2.2.1	TCP	74	51181 → 20 [SYN, ACK] Seq=0 Ack=1 Win=…
85	88.14…	10.2.2.1	10.1.1.1	TCP	66	20 → 51181 [ACK] Seq=1 Ack=1 Win=64256…
87	88.14…	10.2.2.1	10.1.1.1	FTP-DATA	192	FTP Data: 126 bytes (PORT) (LIST)
89	88.14…	10.2.2.1	10.1.1.1	TCP	66	20 → 51181 [FIN, ACK] Seq=127 Ack=1 Wi…
90	88.14…	10.1.1.1	10.2.2.1	TCP	66	51181 → 20 [ACK] Seq=1 Ack=127 Win=651…
91	88.14…	10.1.1.1	10.2.2.1	TCP	66	51181 → 20 [FIN, ACK] Seq=1 Ack=128 Wi…
92	88.14…	10.2.2.1	10.1.1.1	TCP	66	20 → 51181 [ACK] Seq=128 Ack=2 Win=642…
114	108.1…	10.2.2.1	10.1.1.1	TCP	74	20 → 59513 [SYN] Seq=0 Win=64240 Len=0…
115	108.1…	10.1.1.1	10.2.2.1	TCP	74	59513 → 20 [SYN, ACK] Seq=0 Ack=1 Win=…
116	108.1…	10.2.2.1	10.1.1.1	TCP	66	20 → 59513 [ACK] Seq=1 Ack=1 Win=64256…
118	108.1…	10.2.2.1	10.1.1.1	FTP-DATA	290	FTP Data: 224 bytes (PORT) (RETR file1)
119	108.1…	10.2.2.1	10.1.1.1	TCP	66	20 → 59513 [FIN, ACK] Seq=225 Ack=1 Wi…
121	108.1…	10.1.1.1	10.2.2.1	TCP	66	59513 → 20 [ACK] Seq=1 Ack=225 Win=650…
122	108.1…	10.1.1.1	10.2.2.1	TCP	66	59513 → 20 [FIN, ACK] Seq=1 Ack=226 Wi…
123	108.1…	10.2.2.1	10.1.1.1	TCP	66	20 → 59513 [ACK] Seq=226 Ack=2 Win=642…

Figura 6: Tráfego com aplicação do filtro à porta 20

Severity	Summary	Group	Protocol	Count
Chat	Connection establish request (SYN): server port 51181	Sequence	TCP	2
83	20 → 51181 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 S…	Sequence	TCP	
114	20 → 59513 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 S…	Sequence	TCP	
Chat	Connection establish acknowledge (SYN+ACK): server port…	Sequence	TCP	2
84	51181 → 20 [SYN, ACK] Seq=0 Ack=1 Win=65160 Len=0 …	Sequence	TCP	
115	59513 → 20 [SYN, ACK] Seq=0 Ack=1 Win=65160 Len=0 …	Sequence	TCP	
Chat	Connection finish (FIN)	Sequence	TCP	4
89	20 → 51181 [FIN, ACK] Seq=127 Ack=1 Win=64256 Len=0…	Sequence	TCP	
91	51181 → 20 [FIN, ACK] Seq=1 Ack=128 Win=65152 Len=0…	Sequence	TCP	
119	20 → 59513 [FIN, ACK] Seq=225 Ack=1 Win=64256 Len=0…	Sequence	TCP	
122	59513 → 20 [FIN, ACK] Seq=1 Ack=226 Win=65024 Len=0…	Sequence	TCP	

Figura 7: Fases de conexão obtidas com recurso ao *Expert Information* do *Wireshark*

Como se observa, a fase inicial de conexão resume-se a [SYN] e [SYN,ACK] (três primeiras linhas da captura da figura 5). Depois ocorre a transferência do ficheiro e termina-se com a fase final, a qual se resume a [FIN,ACK], [ACK] e [FIN,ACK], [ACK] (quatro últimas linhas da mesma captura).

Apresenta-se, então, de seguida, o diagrama temporal da transferência do file 1. Quanto à transferência no Grilo da LAN4, as fases são exatamente iguais, sendo o diagrama extremamente parecido; as flags/tipos do segmento são exatamente iguais, mudando os números de sequência.

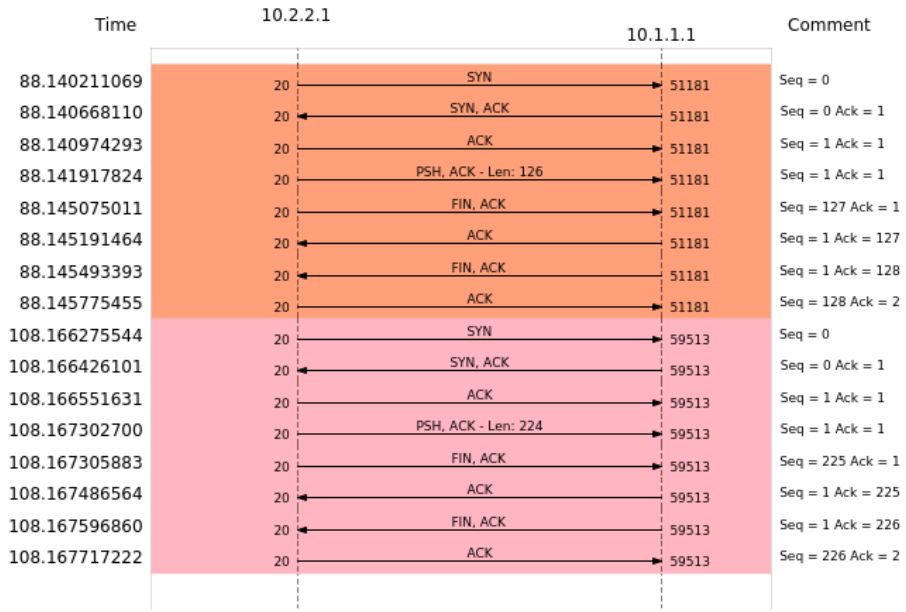


Figura 8: Diagrama temporal da transferência no Portátil 1

2.3 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.

2.3.1 Diagramas de fluxo da transferência do file 1 por TFTP

O protocolo UDP não tem número de segmento, nem *ACKs*, estes só existem a nível aplicacional. Quanto às diferentes fases identificadas, observa-se que, primeiro, o cliente envia um pedido de leitura (*Read Request*) ao servidor, informando-o que pretende transferir o ficheiro. Então, o servidor dá inicio à transferência, enviando o *Data Packet* especificado. Por fim, o cliente envia um *ACK*, para informar o servidor que o *packet* anterior foi recebido com sucesso. As diferentes fases ilustram-se abaixo.

No.	Time	Source	Destination	Protocol	Length	Info
38	60.6606…	10.1.1.1	10.2.2.1	TFTP	56	Read Request, File: file1, Transfer type: octet
39	60.6615…	10.2.2.1	10.1.1.1	TFTP	270	Data Packet, Block: 1 (last)
40	60.6628…	10.1.1.1	10.2.2.1	TFTP	46	Acknowledgement, Block: 1

Figura 9: Tráfego de transferência no Portátil 1

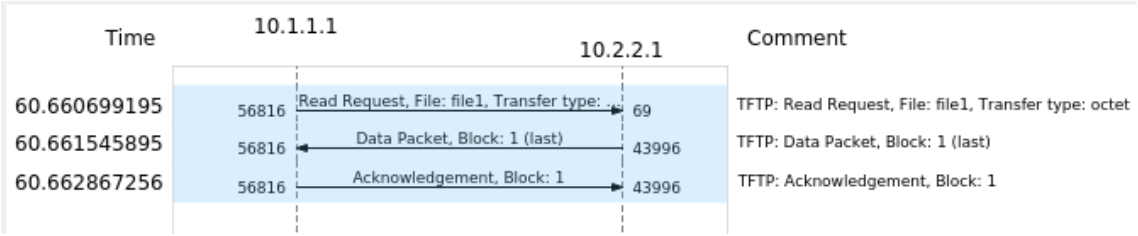


Figura 10: Diagrama temporal da transferência no Portátil 1

Em relação à transferência no Grilo da LAN4, o diagrama é exatamente igual, havendo apenas mais um *Acknowledgement*, como se observa de seguida.

No.	Time	Source	Destination	Protocol	Length	Info
45	71.3201…	10.4.4.1	10.2.2.1	TFTP	56	Read Request, File: file1, Transfer type: octet
46	71.3204…	10.2.2.1	10.4.4.1	TFTP	270	Data Packet, Block: 1 (last)
47	71.3255…	10.4.4.1	10.2.2.1	TFTP	46	Acknowledgement, Block: 1
48	71.3255…	10.4.4.1	10.2.2.1	TFTP	46	Acknowledgement, Block: 1

Figura 11: Tráfego de transferência no Grilo da LAN4

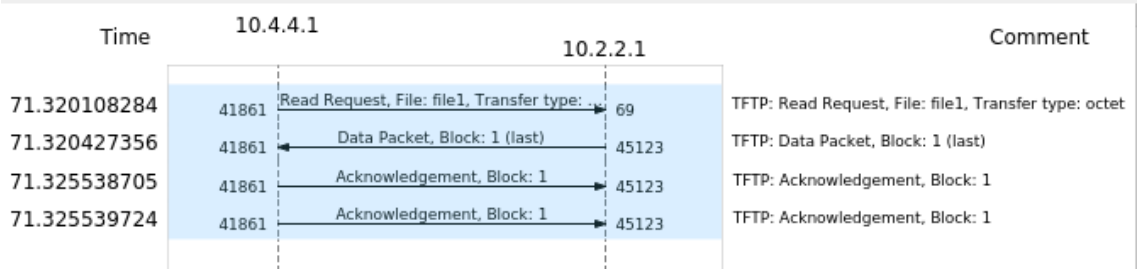


Figura 12: Diagrama temporal da transferência no Grilo da LAN4

2.4 Questão 4

> Compare sucintamente as quatro aplicações de transferência de ficheiros que usou nos seguintes pontos: uso da camada de transporte; eficiência; complexidade; segurança.

2.4.1 Quatro aplicações de transferência de ficheiros

Aplicações	SFTP	FTP	TFTP	HTTP
Camada de Transporte	TCP	TCP	UDP	TCP
Eficiência na Transferência	Semelhante ao FTP mas os dados são encriptados, logo é mais seguro.	Os dados não são encriptados. Assim a transmissão é suscetível de captura por terceiros . Contudo, usa o protocolo TCP, que garante a transmissão correta dos dados.	Menos fiável porque usa o protocolo UDP (não envia mensagens de Acknowledgement).	Vários pacotes podem ser transmitidos ao mesmo tempo. Usa o protocolo TCP, o que conduz a um aumento do débito de informação e de fiabilidade da mesma.
Complexidade	Fiável, permite gestão de acessos, transferência e gestão de dados. Bastante complexo.	Permite transferir ficheiros em paralelo. Cada um cria uma conexão, resultando em várias velocidades de transferência. Bastante complexo.	Como usa o protocolo UDP, a complexidade é muito reduzida e apresenta menos funcionalidades.	Garante confiança, escalabilidade, suporta redes heterogéneas e não confiáveis.
Segurança	Usa SSH, garantindo segurança através da utilização de uma camada de transporte, autenticação e conexão. A camada de transporte é executada com o auxílio do TCP/IP, fornecendo encriptação, autenticação do servidor e proteção de integridade de dados, enquanto que a camada de autenticação é responsável por manipular a autenticação dos clientes.	Não fornece mecanismos de autenticação. Qualquer pessoa pode fazer captura de pacotes.	Acontece o mesmo que no FTP, pois é uma versão mais simples.	Informação está guardada em texto puro, não encriptada, logo é fácil de ler e manipular.

Tabela 1: Explicação sucinta das quatro aplicações de transferência de ficheiros

2.5 Questão 1 - Parte II

> 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.

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[1]				
traceroute	DNS	UDP	53	14,55
telnel	DNS/TELNET	TCP	23	42,55
ftp	FTP	TCP	21	50
tftp	TFTP	UDP	69	15,38
http (browser)	DNS	TCP	80	11,05
nslookup	DNS	UDP	53	16
ssh	DNS/SSHv2	TCP	22	32,78

Tabela 2: Resposta à questão 1 no modo tabela

Deixam-se de seguida as capturas que conduziram aos resultados apresentados na tabela.

2.5.1 Traceroute

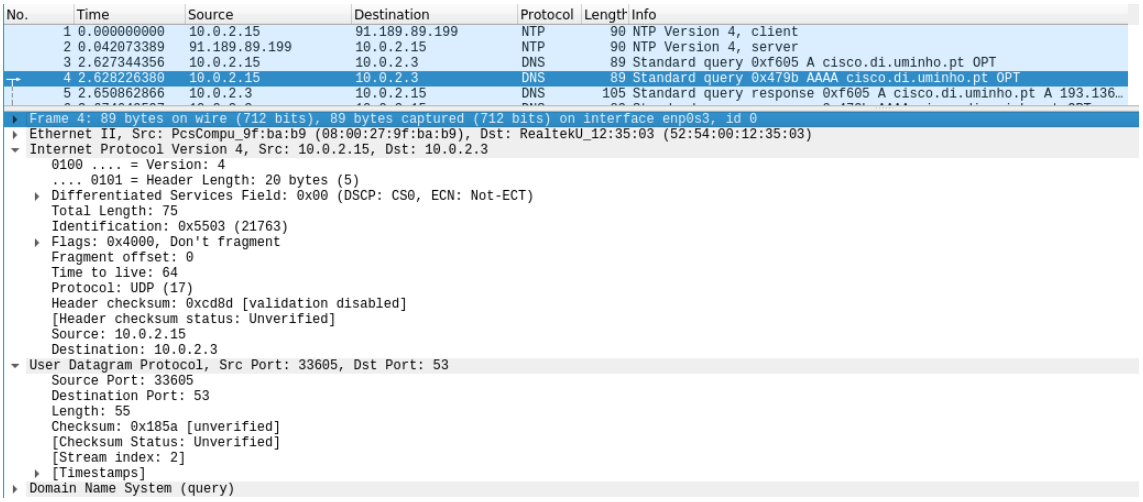


Figura 13: Informação sobre o comando *Traceroute*

Observando a figura, em *User Datagram Protocol*, reparamos que a porta de atendimento/destino é a 53 e que o *Total Lenght* é 75. Assim sendo, o overhead é dado por 8/(75 - 20) * 100 = 14,55.

[1] O PING utiliza o protocolo ICMP, pelo que, não serão retiradas informações.

2.5.2 Telnet

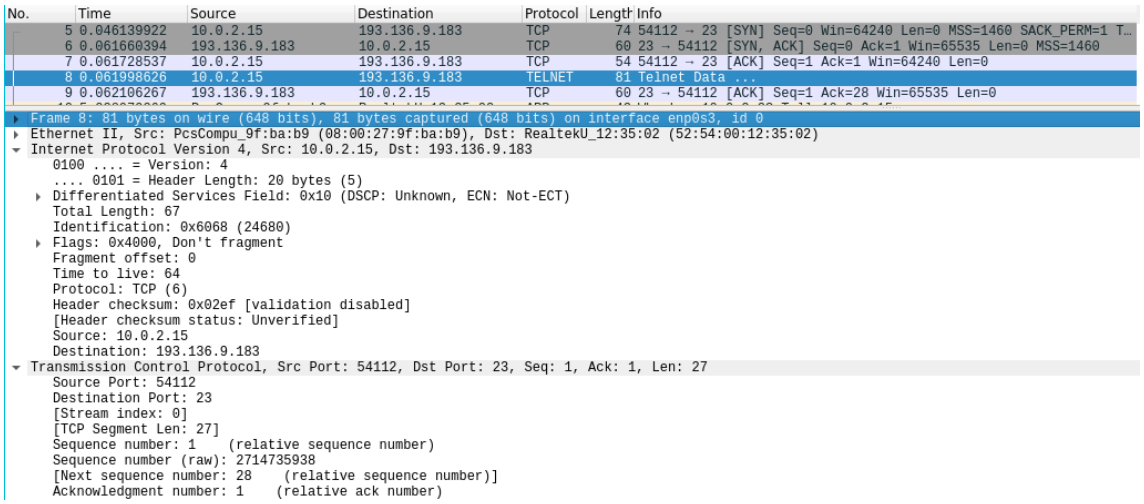


Figura 14: Informação sobre o comando *Telnet*

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

2.5.3 FTP

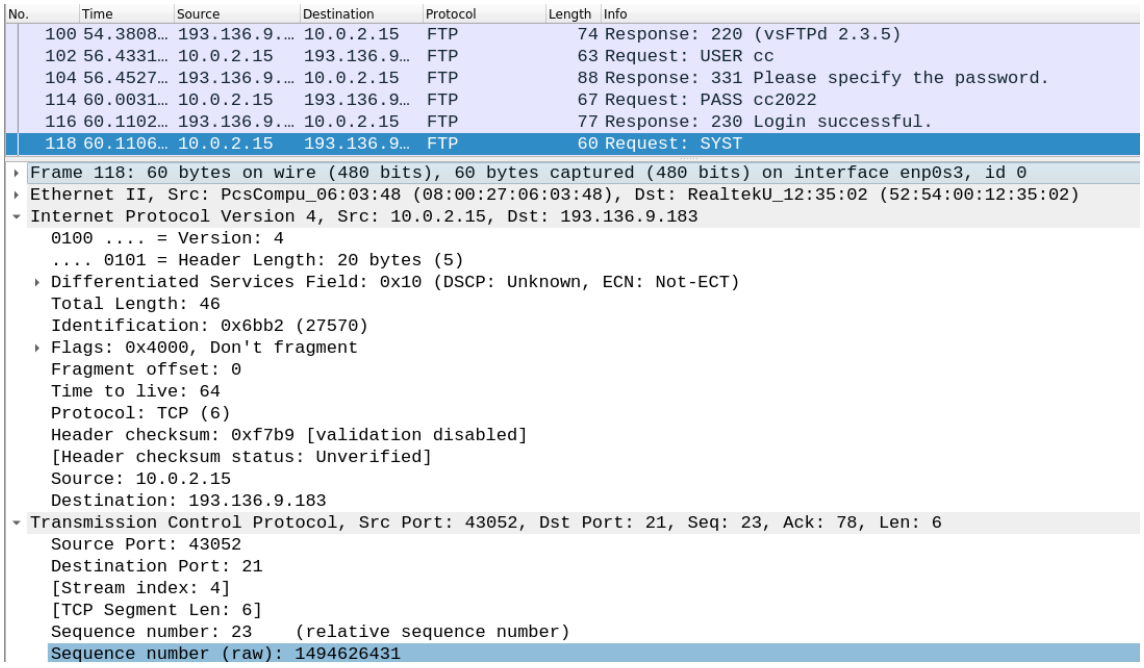


Figura 15: Informação sobre o comando *ftp*

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

2.5.4 TFTP

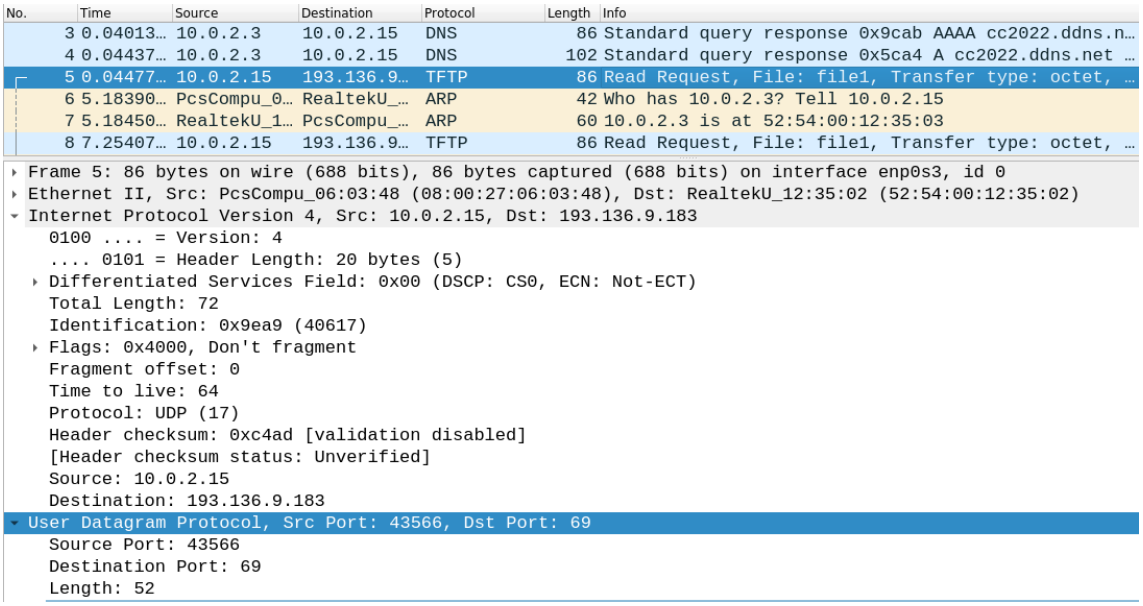


Figura 16: Informação sobre o comando *TFTP*

Observando a figura, em *User Datagram Protocol*, reparamos que a porta de atendimento/destino é a 69 e que o *Total Lenght* é 72. Assim sendo, o overhead é dado por 8/(72 - 20) * 100 = 50.

2.5.5 Browser/HTTP

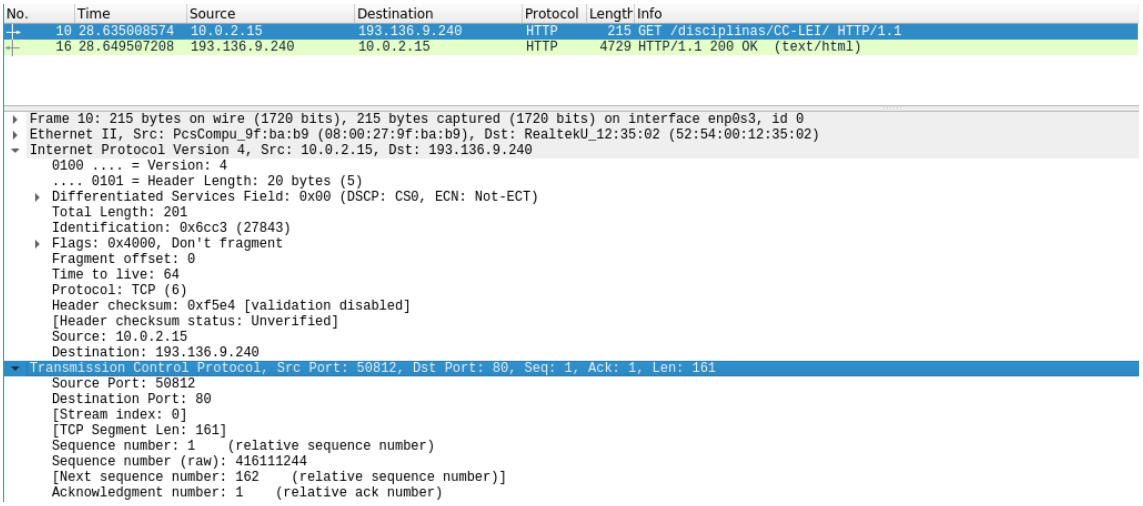


Figura 17: Informação sobre o comando *wget*

Observando a figura, em *Transmission Control Protocol*, reparamos que a porta de atendimento/destino é a 80 e que o *Total Lenght* é 201. Assim sendo, o overhead é dado por 20/(201 - 20) * 100 = 11,05.

2.5.6 SSH

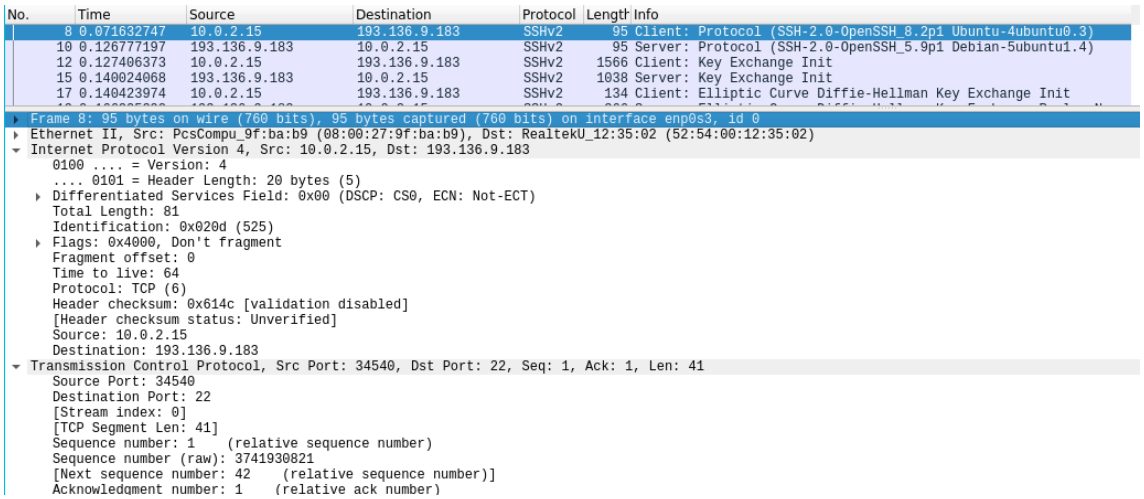


Figura 18: Informação sobre o comando *ssh*

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

2.5.7 Nslookup

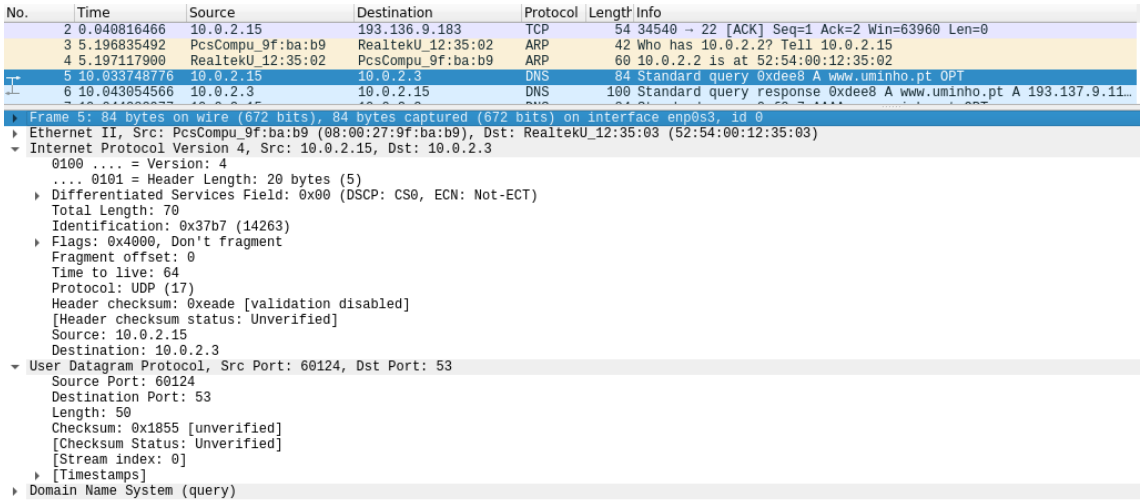


Figura 19: Informação sobre o comando *nslookup*

Observando a figura, em *User Datagram Protocol*, reparamos que a porta de atendimento/destino é a 53 e que o *Total Lenght* é 70. Assim sendo, o overhead é dado por 8/(70 - 20) * 100 = 16.

3 Conclusões

O presente relatório permitiu, inicialmente, consolidar o conhecimento alusivo à camada de transporte e suas aplicações, passando pelo conjunto dos protocolos que sobre esta camada atuam. Segundamente, permitiu também entender bem as diferenças entre os protocolos muito bem explorados aqui, TCP & UDP, e o seu modo de lidar com possíveis dificuldades que possam aparecer numa rede.
Por forma a sumariar alguns aspetos relevantes seguem-se duas tabelas. A primeira visa distinguir os dois protocolos TCP & UDP. A segunda sintetiza as funções da camada de transporte.

Protocolo	**TCP**	**UDP**
Cabeçalho (tamanho)	Variável (mínimo de 20 bytes)	8 bytes
Velocidade	Mais lento	Mais rápido
Controlo de fluxo	Sim	Não
Orientado à conexão	Sim	Não
Controlo de erros	Sim	Não
Controlo de congestão	Sim	Não
Deteção de erros	Sim	Sim

Tabela 3: Distinção entre TCP e UDP (aspetos principais)

Funções da Camada de Transporte
Deteção de erros
Multiplexagem (porta de origem) / Desmultiplexagem (porta de destino)
Encapsulamento
Controlo de Fluxo e de Conexão (início, transferência de dados, fim)

Tabela 4: Funções relevantes da camada de transporte

O grupo considera o conjunto de objetivos iniciais propostos realizados com sucesso e com suporte a nível prático. Consideramos que, apesar da realização do trabalho prático 1 ocorrer antes do lecionamento da cadeira de Redes de Computadores e de Base de Dados, os conceitos necessários para a sua realização foram bem compreendidos.