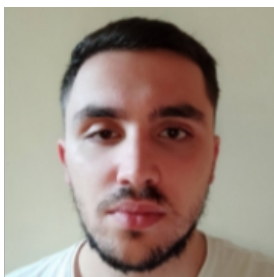

Comunicações por Computadores

Trabalho Prático 1

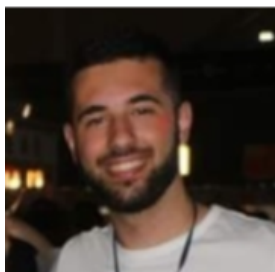
TRABALHO REALIZADO POR:

FRANCISCO PINTO LAMEIRÃO
LUÍS MIGUEL MOREIRA FERREIRA
PEDRO DANTAS DA CUNHA PEREIRA

PL 3 - GRUPO 8



A97504
Francisco Lameirão



A95111
Luís Ferreira



A97396
Pedro Dantas

Índice

1	Exercícios Parte 1	2
1.1	Exercício 1	2
1.2	Exercício 2	3
1.3	Exercício 3	5
1.4	Exercício 4	6
2	Exercícios Parte 2	8
2.1	Exercício 1	8
3	Conclusões	12

1 Exercícios Parte 1

1.1 Exercício 1

Enunciado: 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: Neste caso de estudo, quando ocorrem perdas de pacotes e existe uma tentativa da recuperação dos mesmos, acaba por se verificar o envio de pacotes duplicados que, em torno, contribuem para um intervalo de tempo de envio mais elevado, visto que o próprio número total de pacotes também aumenta. Como consequência, o desempenho das aplicações acaba por piorar, visto que o seu tempo de execução total será mais elevado do que o normal.

```
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=1.43 ms  
64 bytes from 10.4.4.1: icmp_seq=2 ttl=61 time=0.471 ms  
64 bytes from 10.4.4.1: icmp_seq=3 ttl=61 time=2.98 ms  
64 bytes from 10.4.4.1: icmp_seq=4 ttl=61 time=0.623 ms  
64 bytes from 10.4.4.1: icmp_seq=5 ttl=61 time=3.41 ms  
64 bytes from 10.4.4.1: icmp_seq=6 ttl=61 time=1.06 ms  
64 bytes from 10.4.4.1: icmp_seq=7 ttl=61 time=0.810 ms  
64 bytes from 10.4.4.1: icmp_seq=8 ttl=61 time=0.526 ms  
64 bytes from 10.4.4.1: icmp_seq=9 ttl=61 time=3.46 ms  
64 bytes from 10.4.4.1: icmp_seq=10 ttl=61 time=0.691 ms  
64 bytes from 10.4.4.1: icmp_seq=11 ttl=61 time=2.83 ms  
64 bytes from 10.4.4.1: icmp_seq=12 ttl=61 time=0.668 ms  
64 bytes from 10.4.4.1: icmp_seq=13 ttl=61 time=2.70 ms  
64 bytes from 10.4.4.1: icmp_seq=14 ttl=61 time=0.508 ms  
64 bytes from 10.4.4.1: icmp_seq=15 ttl=61 time=3.19 ms  
64 bytes from 10.4.4.1: icmp_seq=16 ttl=61 time=1.05 ms  
64 bytes from 10.4.4.1: icmp_seq=17 ttl=61 time=0.811 ms  
64 bytes from 10.4.4.1: icmp_seq=18 ttl=61 time=0.690 ms  
64 bytes from 10.4.4.1: icmp_seq=19 ttl=61 time=3.19 ms  
64 bytes from 10.4.4.1: icmp_seq=20 ttl=61 time=0.562 ms  
  
--- 10.4.4.1 ping statistics ---  
20 packets transmitted, 20 received, 0% packet loss, time 19152ms  
rtt min/avg/max/mdev = 0.471/1.583/3.463/1.150 ms  
~
```

Figure 1: Ping Portatill

```
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=11.1 ms
64 bytes from 10.4.4.1: icmp_seq=2 ttl=61 time=5.70 ms
64 bytes from 10.4.4.1: icmp_seq=3 ttl=61 time=6.36 ms
64 bytes from 10.4.4.1: icmp_seq=4 ttl=61 time=5.80 ms
64 bytes from 10.4.4.1: icmp_seq=5 ttl=61 time=6.13 ms
64 bytes from 10.4.4.1: icmp_seq=6 ttl=61 time=6.93 ms
64 bytes from 10.4.4.1: icmp_seq=7 ttl=61 time=5.52 ms
64 bytes from 10.4.4.1: icmp_seq=10 ttl=61 time=6.36 ms
64 bytes from 10.4.4.1: icmp_seq=11 ttl=61 time=5.58 ms
64 bytes from 10.4.4.1: icmp_seq=12 ttl=61 time=6.23 ms
64 bytes from 10.4.4.1: icmp_seq=12 ttl=61 time=6.28 ms (DUP!)
64 bytes from 10.4.4.1: icmp_seq=14 ttl=61 time=6.74 ms
64 bytes from 10.4.4.1: icmp_seq=15 ttl=61 time=6.01 ms
64 bytes from 10.4.4.1: icmp_seq=16 ttl=61 time=6.35 ms
64 bytes from 10.4.4.1: icmp_seq=17 ttl=61 time=6.49 ms
64 bytes from 10.4.4.1: icmp_seq=18 ttl=61 time=5.37 ms
64 bytes from 10.4.4.1: icmp_seq=19 ttl=61 time=5.83 ms
64 bytes from 10.4.4.1: icmp_seq=20 ttl=61 time=6.55 ms

--- 10.4.4.1 ping statistics ---
20 packets transmitted, 17 received, +1 duplicates, 15% packet loss, time 19085ms
rtt min/avg/max/mdev = 5.374/6.406/11.095/1.211 ms
~
```

Figure 2: Ping PC1

Em relação à camada à qual será atribuída a tarefa de lidar com as perdas e duplicações dos pacotes, podemos dizer que esta estará dependente do tipo de protocolo de transporte a ser utilizado.

UDP: No caso de ser escolhido um protocolo no qual se utilize um formato UDP, será a camada de aplicação a responsável pelos problemas de perda e duplicação de pacotes. Isto porque o formato em questão considera como fator principal a velocidade à qual os pacotes são enviados, ignorando possíveis erros nos mesmos, sendo que, caso exista de facto algum erro, o pacote em questão é apenas descartado.

TCP: Caso seja utilizado o protocolo TCP, a camada responsável pelos problemas referidos será a camada de transporte, visto que, ao contrário do formato UDP, este dá prioridade à detecção e correção de erros em relação à velocidade à qual os pacotes são enviados. Quando é encontrado algum tipo de erro, dá-se uma tentativa de retransmissão dos pacotes.

1.2 Exercício 2

Enunciado: 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:

FTP: O protocolo FTP poderá ser executado em modo passivo ou modo ativo, sendo a escolha entre estes dois modos feita com base em quem inicia a conexão entre cliente e servidor. Neste protocolo, é utilizado um par de conexões entre servidor e cliente, sendo que a primeira conexão a ser criada é a conexão com a porta 21 do servidor e a segunda

com a porta 20 do servidor. Após a abertura da porta 21 dos servidores é iniciada a escuta das conexões de entrada do cliente. Os clientes realizam a conexão com a porta 21 dos servidores remotos, de modo a iniciar as transferências de ficheiros. A porta 20 é necessária para garantir que estas transferências são, de facto, realizadas.

Modo Passivo: Caso seja o cliente a iniciar a conexão de dados, o protocolo FTP será executado em modo passivo. Isto significa que apenas o servidor será necessário portas para o tráfego de dados. De modo a evitar problemas de segurança, a maioria dos servidores utilizam uma conexão FTP passiva.

Modo Ativo: No caso da conexão de dados ser iniciada pelo servidor, a conexão FTP será executada em modo ativo. Isto traduz-se na necessidade de tanto o servidor como o cliente abrirem portas de modo a receber tráfego.

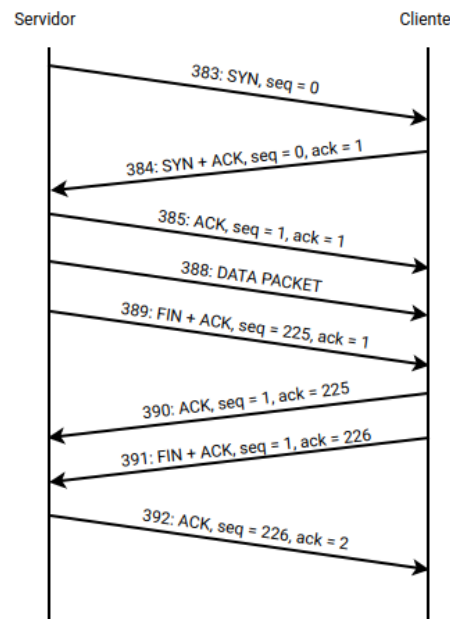


Figure 3: Diagrama Temporal FTP

383	474.073527754	10.4.4.1	10.1.1.1	TCP	74	20 → 41503	[SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK
384	474.073713496	10.1.1.1	10.4.4.1	TCP	74	41503 → 20	[SYN, ACK] Seq=0 Ack=1 Win=65160 Len=0 MS
385	474.074049475	10.4.4.1	10.1.1.1	TCP	66	20 → 41503	[ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=2
386	474.074051308	10.4.4.1	10.1.1.1	FTP	130	Response: 150	Opening BINARY mode data connection fc
387	474.074998407	10.1.1.1	10.4.4.1	TCP	66	39572 → 21	[ACK] Seq=112 Ack=391 Win=64256 Len=0 TSv
388	474.075549504	10.4.4.1	10.1.1.1	FTP-DA...	290	FTP Data: 224	bytes (PORT) (RETR file1)
389	474.075553181	10.4.4.1	10.1.1.1	TCP	66	20 → 41503	[FIN, ACK] Seq=225 Ack=1 Win=64256 Len=0
390	474.075825017	10.1.1.1	10.4.4.1	TCP	66	41503 → 20	[ACK] Seq=1 Ack=225 Win=65024 Len=0 TSval
391	474.075825378	10.1.1.1	10.4.4.1	TCP	66	41503 → 20	[FIN, ACK] Seq=1 Ack=226 Win=65024 Len=0
392	474.076073038	10.4.4.1	10.1.1.1	TCP	66	20 → 41503	[ACK] Seq=226 Ack=2 Win=64256 Len=0 TSval
393	474.076074932	10.4.4.1	10.1.1.1	FTP	90	Response: 226	Transfer complete.

Figure 4:

Tal como podemos verificar nas imagens apresentadas anteriormente, a conexão inicia-se com o envio de um segmento TCP SYN por parte do servidor, indicando quais os números de sequência a ser sincronizados de modo a iniciar a conexão. Assim que recebido pelo cliente, é dada uma resposta pelo mesmo, através do envio de um segmento TCP SYN e uma trama ACK de forma a confirmar a receção do segmento anteriormente enviado pelo servidor. Seguidamente, o servidor envia também uma trama ACK de modo a confirmar a receção do SYN enviado pelo cliente.

Após o estabelecimento e a verificação desta conexão, é então inicializada a fase de transmissão, sendo que esta começa com o envio dos dados pretendidos por parte do servidor.

Finalmente, passamos para a fase de ... que começa com o envio de um segmento TCP FIN, de modo a informar o cliente que todos os dados foram enviados. Como resposta, o cliente envia uma trama ACK e um segmento TCP FIN com uma trama ACK adicional, referente ao TCP FIN enviado pelo servidor anteriormente. Finalmente, é enviada uma última trama ACK pelo servidor, indicando o sucesso na recepção do segmento TCP FIN enviado pelo cliente.

1.3 Exercício 3

Enunciado: 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:

Como é possível verificar nas imagens abaixo apresentadas, inicialmente é enviado um Read Request ao servidor por parte do cliente. Após isto, o servidor envia um pacote no qual estão contidos os dados que pretende enviar. Finalmente, o cliente envia uma trama ACK de modo a confirmar o sucesso da recepção dos dados.

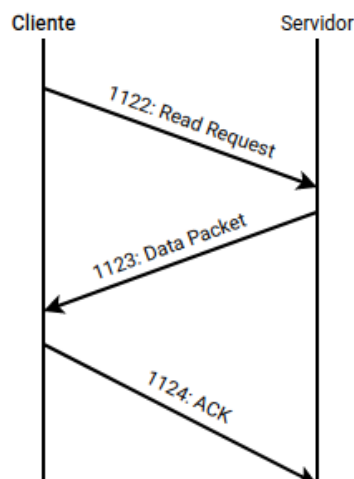


Figure 5: Diagrama Temporal TFTP

1122	1519.4491837	10.1.1.1	10.4.4.1	TFTP	56 Read Request, File: file1, Transfer type: octet
1123	1519.4608812	10.4.4.1	10.1.1.1	TFTP	270 Data Packet, Block: 1 (last)
1124	1519.4615311	10.1.1.1	10.4.4.1	TFTP	46 Acknowledgement, Block: 1
1125	1520.5427506	fe80::200:ff:feaa:10	ff02::5	OSPF	90 Hello Packet
1126	1520.7163504	10.4.4.254	224.0.0.5	OSPF	78 Hello Packet
1127	1522.7167424	10.4.4.254	224.0.0.5	OSPF	78 Hello Packet
1128	1524.6011804	00:00:00_aa:00:10	00:00:00_aa:00:14	ARP	42 Who has 10.4.4.1? Tell 10.4.4.254
1129	1524.6014519	00:00:00_aa:00:14	00:00:00_aa:00:10	ARP	42 Who has 10.4.4.254? Tell 10.4.4.1
1130	1524.6014540	00:00:00_aa:00:14	00:00:00_aa:00:10	ARP	42 10.4.4.1 is at 00:00:00_aa:00:14

Frame 1122: 56 bytes on wire (448 bits), 56 bytes captured (448 bits) on interface veth1.3.eb, id 0
 Ethernet II, Src: 00:00:00_aa:00:10 (00:00:00_aa:00:10), Dst: 00:00:00_aa:00:14 (00:00:00_aa:00:14)
 Internet Protocol Version 4, Src: 10.1.1.1, Dst: 10.4.4.1
 User Datagram Protocol, Src Port: 51665, Dst Port: 69
 Trivial File Transfer Protocol

Figure 6: Captura TFTP

No protocolo TFTP, todas as transferências são iniciadas através de um pedido de leitura de um ficheiro (ou de escrita no mesmo), sendo que este mesmo pedido será também utilizado como pedido de conexão. Caso o pedido seja aceite por parte do servidor, é estabelecida a conexão entre servidor e cliente e o ficheiro será transmitido através de blocos com 512 bytes, sendo este um limite de tamanho fixo. Na eventualidade de o tamanho do pacote ser superior a 512 bytes, este sofrerá fragmentação, sendo então enviados vários pacotes com o mesmo limite de tamanho.

1.4 Exercício 4

Enunciado: 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:

(i) Uso da Camada de Transporte

- **SFTP** - Protocolo TCP;
- **FTP** - Protocolo TCP;
- **TFTP** - Protocolo UDP;
- **HTTP** - Protocolo TCP;

(ii) Eficiência

- **SFTP** - Semelhante ao FTP, mas como tem os dados encriptados é mais segura;
- **FTP** - O protocolo FTP utiliza *acknowledges* de modo a confirmar e garantir a transmissão dos segmentos. No entanto, o facto de ter de esperar pelos referidos *acknowledges* para continuar acaba por baixar o nível de eficiência deste protocolo.
- **TFTP** - Visto que o TFTP usa o protocolo UDP, é considerado menos viável, tendo em conta que, sem o uso de *acknowledges*, é impossível confirmar a receção dos segmentos enviados. Caso seja bem sucedido, é mais rápido que o protocolo FTP, no entanto, a maioria das vezes existe a necessidade de retransmissão dos pacotes múltiplas vezes.
- **HTTP** - Permite que vários HTTP requests sejam enviados numa única ligação TCP sem ser preciso esperar pelas respostas individuais de cada um.

(iii) Complexidade

- **SFTP** - Este protocolo permite gestão, acesso e transferência de dados. Estas funcionalidades têm todas custos de processamento elevados, pelo que o protocolo é considerado bastante complexo.
- **FTP** - O protocolo FTP suporta a possibilidade de existirem vários pedidos de transferência em paralelo, nos quais se realiza uma nova conexão de dados em cada transferência. Isto leva a que existam diferentes velocidades de transferência. Tendo em conta a frequência com a qual estas novas conexões são criadas, o protocolo FTP é considerado bastante complexo.

-
- **TFTP** - É uma alternativa do protocolo FTP, sendo uma versão mais "trivial" (tal como indicado no nome) e simples. Posto isto, é também um protocolo com menos funcionalidades que o FTP, para além de utilizar o protocolo UDP ao invés do protocolo TCP. Com todos estes fatores em mente, podemos afirmar que é um protocolo com um nível de complexidade baixo.
 - **HTTP** - Implementa uma garantia de confiança, escalabilidade e encerramento de sistemas. Por ter todas estas capacidades assumimos que se trata de um protocolo bastante complexo.

(iv) Segurança

- **SFTP** - Sendo que este protocolo usa SSH (*Secure Shell*), podemos afirmar que se trata de um protocolo bastante seguro. Isto porque o SSH utiliza uma arquitetura em camadas, nas quais a camada de transporte (com auxílio do protocolo TCP/IP) fornece encriptação, autenticação do servidor e proteção da integridade dos dados, e a camada de autenticação tem como objetivo gerir a autenticação dos clientes.
- **FTP** - Conhecido por falhas na segurança, pois não fornece encriptação de dados. Assim sendo, qualquer pessoa é capaz de capturar pacotes na rede e obter assim nomes de utilizadores, palavras-passe, etc. Isto faz com que o protocolo FTP seja considerado muito inseguro.
- **TFTP** - Não fornece autenticação. Assim sendo, não protege dados que se pretendam transferir, fazendo com que este não seja um protocolo seguro.
- **HTTP** - É um protocolo da camada de aplicação e é frequentemente usado para transferências de dados na Internet. Apesar disto, atendendo ao facto de que a informação é representada em texto e não é encriptada, os dados podem ser alterados.

2 Exercícios Parte 2

2.1 Exercício 1

Enunciado: 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.

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)
wget, lynx ou via browser	HTTP	TCP	80	20
ssh, sftp	SSH	TCP	22	20
ftp	FTP	TCP	21	20
Tftp	TFTP	UDP	69	8
telnet	Telnet	TCP	23	20
nslookup ou dig	DNS	UDP	53	8
Ping	Ping	-	-	-
Traceroute	Traceroute	UDP	33437	8
Outras:				

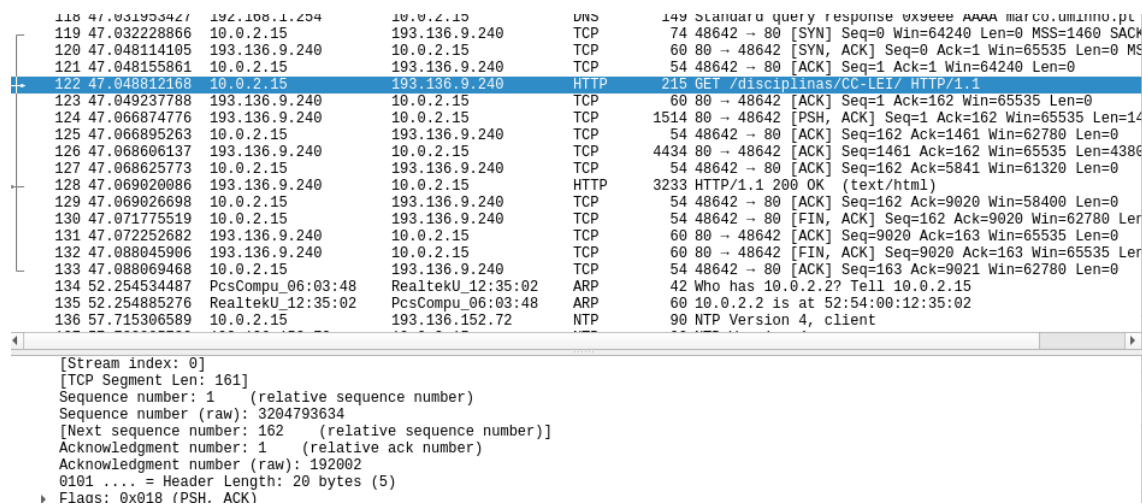


Figure 7: Captura Wget

No.	Time	Source	Destination	Protocol	Length	Info
7	0.281864242	193.136.9.201	10.0.2.15	TCP	60	22 → 36230 [ACK] Seq=1 Ack=42 Win=65535 Len=0
8	0.325336944	193.136.9.201	10.0.2.15	SSHv2	95	Server: Protocol (SSH-2.0-OpenSSH_8.9p1 Ubuntu-8ubuntu2) 193.136.9.201
9	0.325360454	10.0.2.15	193.136.9.201	TCP	54	36230 → 22 [ACK] Seq=42 Ack=42 Win=64199 Len=0
10	0.326360312	10.0.2.15	193.136.9.201	SSHv2	1566	Client: Key Exchange Init
11	0.326597630	193.136.9.201	10.0.2.15	TCP	60	22 → 36230 [ACK] Seq=42 Ack=1502 Win=65535 Len=0
12	0.326597720	193.136.9.201	10.0.2.15	TCP	60	22 → 36230 [ACK] Seq=42 Ack=1554 Win=65535 Len=0
13	0.328408721	193.136.9.201	10.0.2.15	SSHv2	1134	Server: Key Exchange Init
14	0.328414661	10.0.2.15	193.136.9.201	TCP	54	36230 → 22 [ACK] Seq=1554 Ack=1122 Win=6372 Len=0
15	0.330123306	10.0.2.15	193.136.9.201	SSHv2	102	Client: Diffie-Hellman Key Exchange Init
16	0.330475742	193.136.9.201	10.0.2.15	TCP	60	22 → 36230 [ACK] Seq=1122 Ack=1602 Win=65535 Len=0
17	0.352068593	193.136.9.201	10.0.2.15	SSHv2	650	Server: Diffie-Hellman Key Exchange Reply,
18	0.352092434	10.0.2.15	193.136.9.201	TCP	54	36230 → 22 [ACK] Seq=1602 Ack=1718 Win=6372 Len=0
19	0.354285742	10.0.2.15	193.136.9.201	SSHv2	70	Client: New Keys
20	0.354651079	193.136.9.201	10.0.2.15	TCP	60	22 → 36230 [ACK] Seq=1718 Ack=1618 Win=65535 Len=0
21	0.355036181	10.0.2.15	193.136.9.201	SSHv2	98	Client: Encrypted packet (len=44)
22	0.355495800	193.136.9.201	10.0.2.15	TCP	60	22 → 36230 [ACK] Seq=1722 Ack=1662 Win=65535 Len=0
23	0.378152635	193.136.9.201	10.0.2.15	SSHv2	98	Server: Encrypted packet (len=44)
24	0.378177207	10.0.2.15	193.136.9.201	TCP	54	36230 → 22 [ACK] Seq=1662 Ack=1762 Win=6372 Len=0
25	0.378666407	10.0.2.15	193.136.9.201	SSHv2	114	Client: Encrypted packet (len=60)
26	0.378892826	193.136.9.201	10.0.2.15	TCP	60	22 → 36230 [ACK] Seq=1762 Ack=1722 Win=65535 Len=0
27	0.403401964	193.136.9.201	10.0.2.15	SSHv2	106	Server: Encrypted packet (len=52)
28	0.454036172	10.0.2.15	193.136.9.201	TCP	54	36230 → 22 [ACK] Seq=1722 Ack=1814 Win=6372 Len=0
29	0.525314334	10.0.2.15	193.136.9.201	SSHv2	138	Client: Encrypted packet (len=84)

SSH Version 2 (encryption:chacha20-poly1305@openssh.com mac:<implicit> compression:none)
Packet Length: 1076
Padding Length: 7
Key Exchange
Message Code: Key Exchange Init (20)
Algorithms
Cookie: eed311ba4681392a956dc650464c0bdf
kex algorithms length: 265

```

0030 ff ff e8 b6 00 00 00 04 34 07 14 ee d3 11 ba .....4.....
0040 46 81 39 2a 95 6d c6 50 46 4c 0b d7 00 00 01 09 F-9-m-P FL.....
0050 63 75 72 76 65 32 35 35 31 39 2d 73 68 61 32 35 curve25519-sha256
0060 36 2c 63 75 72 76 65 32 35 35 31 39 2d 73 68 61 6,curve25519-sha256
0070 32 35 36 40 6c 69 62 73 73 68 2e 6f 72 67 2c 65 256@libs.sh.org,e
0080 63 64 68 2d 73 68 61 32 2d 6e 69 73 74 70 32 35 cdh-sha2-nistp256
0090 36 2c 65 63 64 68 2d 73 68 61 32 2d 6e 69 73 74 6,ecdh-sha2-nist
00a0 70 33 38 34 2c 65 63 64 68 2d 73 68 61 32 2d 6e p384,ecd-h-sha2-n
00b0 69 73 74 70 35 32 31 2c 73 6e 74 72 75 70 37 36 istp521, sntrup76
00c0 31 78 32 35 35 31 39 2d 73 68 61 35 31 32 40 6f 1x25519-sha512o
00d0 70 65 6e 73 73 68 2e 63 6f 6d 2c 64 69 66 66 69 penssh.com,diffi
00e0 65 2d 68 65 6c 6c 6d 61 6e 2d 67 72 6f 75 70 2d e-hellman-group
00f0 65 78 63 68 61 6e 67 65 2d 73 68 61 32 35 36 2c exchange-sha256,
0100 64 69 66 66 69 65 2d 68 65 6c 6d 61 6e 2d 67 diffie-hellman-gro
0110 72 6f 75 70 31 36 2d 73 68 61 35 31 32 2c 64 69 oup16-sha512,d
0120 66 66 69 65 2d 68 65 6c 6c 6d 61 6e 2d 67 72 6f ffie-hellman-gro

```

Figure 8: Captura SSH

341	541.982821364	10.0.2.15	193.137.214.36	TCP	74	[TCP Retransmission] 40060 → 17071 [SYN] Seq=0 Win=0 Len=0
342	541.999901216	193.137.214.36	10.0.2.15	TCP	60	17071 → 40060 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0
343	541.999947317	10.0.2.15	193.137.214.36	TCP	54	40060 → 17071 [ACK] Seq=1 Ack=1 Win=64240 Len=0
344	542.000531693	10.0.2.15	193.137.214.36	FTP	67	Request: RETR README
345	542.000866304	193.137.214.36	10.0.2.15	TCP	60	21 → 56490 [ACK] Seq=621 Ack=104 Win=65535 Len=0
346	542.018893580	193.137.214.36	10.0.2.15	FTP-DA...	397	FTP Data: 343 bytes (PASV) (PASV)
347	542.018928209	10.0.2.15	193.137.214.36	TCP	54	40060 → 17071 [ACK] Seq=1 Ack=344 Win=63897 Len=0
348	542.018894111	193.137.214.36	10.0.2.15	TCP	60	17071 → 40060 [FIN, ACK] Seq=344 Ack=1 Win=65535 Len=0
349	542.019577784	193.137.214.36	10.0.2.15	FTP	119	Response: 150 Opening BINARY mode data connection for
350	542.019583736	10.0.2.15	193.137.214.36	TCP	54	56490 → 21 [ACK] Seq=104 Ack=686 Win=64062 Len=0
351	542.020001212	10.0.2.15	193.137.214.36	TCP	54	40060 → 17071 [FIN, ACK] Seq=1 Ack=345 Win=63897 Len=0
352	542.022482090	193.137.214.36	10.0.2.15	TCP	60	17071 → 40060 [ACK] Seq=345 Ack=2 Win=65535 Len=0
353	542.034754031	193.137.214.36	10.0.2.15	FTP	78	Response: 226 Transfer complete.
354	542.034772447	10.0.2.15	193.137.214.36	TCP	54	56490 → 21 [ACK] Seq=104 Ack=710 Win=64062 Len=0
355	556.527788792	10.0.2.15	193.137.214.36	FTP	60	Request: QUIT
356	556.528212649	193.137.214.36	10.0.2.15	TCP	60	21 → 56490 [ACK] Seq=710 Ack=110 Win=65535 Len=0
357	556.543807886	193.137.214.36	10.0.2.15	FTP	68	Response: 221 Goodbye.
358	556.543825832	10.0.2.15	193.137.214.36	TCP	54	56490 → 21 [ACK] Seq=110 Ack=724 Win=64062 Len=0
359	556.543808167	193.137.214.36	10.0.2.15	TCP	60	21 → 56490 [FIN, ACK] Seq=724 Ack=110 Win=65535 Len=0
360	556.544415290	10.0.2.15	193.137.214.36	TCP	54	56490 → 21 [FIN, ACK] Seq=110 Ack=725 Win=64062 Len=0
361	556.544641652	193.137.214.36	10.0.2.15	TCP	60	21 → 56490 [ACK] Seq=725 Ack=111 Win=65535 Len=0
362	557.715245545	10.0.2.15	91.189.91.157	NTP	90	NTP Version 4, client
363	557.815265682	91.189.91.157	10.0.2.15	NTP	90	NTP Version 4, server

Frame 344: 67 bytes on wire (536 bits), 67 bytes captured (536 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
Transmission Control Protocol, Src Port: 56490, Dst Port: 21, Seq: 91, Ack: 621, Len: 13
Source Port: 56490
Destination Port: 21
[Stream index: 1]
[TCP Segment Len: 13]
Sequence number: 91 (relative sequence number)

Figure 9: Captura FTP

No.	Time	Source	Destination	Protocol	Length	Info
63	35.652951232	193.136.9.33	10.0.2.15	TCP	60	23 → 34478 [ACK] Seq=505 Ack=66 Win=65535 Len=0
64	35.669716977	193.136.9.33	10.0.2.15	TELNET	60	Telnet Data ...
65	35.669730789	10.0.2.15	193.136.9.33	TCP	54	34478 → 23 [ACK] Seq=66 Ack=506 Win=63784 Len=0
66	36.048452891	10.0.2.15	193.136.9.33	TELNET	55	Telnet Data ...
67	36.048750770	193.136.9.33	10.0.2.15	TCP	60	23 → 34478 [ACK] Seq=506 Ack=67 Win=65535 Len=0
68	36.065830539	193.136.9.33	10.0.2.15	TELNET	60	Telnet Data ...
69	36.065855059	10.0.2.15	193.136.9.33	TCP	54	34478 → 23 [ACK] Seq=67 Ack=507 Win=63784 Len=0
70	36.194933446	10.0.2.15	193.136.9.33	TELNET	55	Telnet Data ...
71	36.195358704	193.136.9.33	10.0.2.15	TCP	60	23 → 34478 [ACK] Seq=507 Ack=68 Win=65535 Len=0
72	36.221368738	193.136.9.33	10.0.2.15	TELNET	60	Telnet Data ...
73	36.221387689	10.0.2.15	193.136.9.33	TCP	54	34478 → 23 [ACK] Seq=68 Ack=508 Win=63784 Len=0
74	36.262052334	10.0.2.15	193.136.9.33	TELNET	55	Telnet Data ...
75	36.262556461	193.136.9.33	10.0.2.15	TCP	60	23 → 34478 [ACK] Seq=508 Ack=69 Win=65535 Len=0
76	36.311622122	193.136.9.33	10.0.2.15	TELNET	60	Telnet Data ...
77	36.311655226	10.0.2.15	193.136.9.33	TCP	54	34478 → 23 [ACK] Seq=69 Ack=509 Win=63784 Len=0
78	36.897898819	10.0.2.15	193.136.9.33	TELNET	56	Telnet Data ...
79	36.898191770	193.136.9.33	10.0.2.15	TCP	60	23 → 34478 [ACK] Seq=509 Ack=71 Win=65535 Len=0
80	36.920848897	193.136.9.33	10.0.2.15	TELNET	66	Telnet Data ...
81	36.920866907	10.0.2.15	193.136.9.33	TCP	54	34478 → 23 [ACK] Seq=71 Ack=521 Win=63784 Len=0
82	37.702957275	10.0.2.15	193.136.9.33	TELNET	55	Telnet Data ...
83	37.703404280	193.136.9.33	10.0.2.15	TCP	60	23 → 34478 [ACK] Seq=521 Ack=72 Win=65535 Len=0
84	38.197395238	10.0.2.15	193.136.9.33	TELNET	55	Telnet Data ...
85	38.197905326	193.136.9.33	10.0.2.15	TCP	60	23 → 34478 [ACK] Seq=521 Ack=73 Win=65535 Len=0
Acknowledgment number (raw): 21248509						
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: 0xd6d3 [unverified]						
[Checksum Status: Unverified]						

Figure 10: Captura Telnet

No.	Time	Source	Destination	Protocol	Length	Info
67	8.843765669	10.0.2.15	88.157.128.22	NTP	90	NTP Version 4, client
68	8.852268528	88.157.128.22	10.0.2.15	NTP	90	NTP Version 4, server
69	10.843314391	10.0.2.15	88.157.128.22	NTP	90	NTP Version 4, client
70	10.853584267	88.157.128.22	10.0.2.15	NTP	90	NTP Version 4, server
71	12.843520623	10.0.2.15	88.157.128.22	NTP	90	NTP Version 4, client
72	12.854859251	88.157.128.22	10.0.2.15	NTP	90	NTP Version 4, server
73	14.843522490	10.0.2.15	88.157.128.22	NTP	90	NTP Version 4, client
74	14.853932759	88.157.128.22	10.0.2.15	NTP	90	NTP Version 4, server
75	17.843544810	10.0.2.15	109.48.74.248	NTP	90	NTP Version 4, client
76	17.869145538	109.48.74.248	10.0.2.15	NTP	90	NTP Version 4, server
77	19.969203602	10.0.2.15	192.168.1.254	DNS	84	Standard query 0x00ab A www.uminho.pt OPT
78	19.971448756	192.168.1.254	10.0.2.15	DNS	100	Standard query response 0x00ab A www.uminho.pt
79	19.972377940	10.0.2.15	192.168.1.254	DNS	84	Standard query 0xcd94 AAAA www.uminho.pt OPT
80	19.983773047	192.168.1.254	10.0.2.15	DNS	147	Standard query response 0xcd94 AAAA www.uminho.pt
81	20.071639720	PcsCompu_06:03:48	RealtekU_12:35:02	ARP	42	Who has 10.0.2.2? Tell 10.0.2.15
82	20.072574569	RealtekU_12:35:02	PcsCompu_06:03:48	ARP	60	10.0.2.2 is at 52:54:00:12:35:02
83	57.844381041	10.0.2.15	192.168.1.254	DNS	92	Standard query 0xab63 A 1.ubuntu.pool.ntp.org
84	57.844757299	10.0.2.15	192.168.1.254	DNS	92	Standard query 0x15a7 AAAA 1.ubuntu.pool.ntp.org
85	57.912742227	192.168.1.254	10.0.2.15	DNS	140	Standard query response 0xab63 A 1.ubuntu.pool.ntp.org
86	57.924336076	192.168.1.254	10.0.2.15	DNS	162	Standard query response 0x15a7 AAAA 1.ubuntu.pool.ntp.org
87	57.924598140	10.0.2.15	194.117.47.44	NTP	90	NTP Version 4, client
88	57.940056318	194.117.47.44	10.0.2.15	NTP	90	NTP Version 4, server
89	61.843629059	10.0.2.15	194.117.47.44	NTP	90	NTP Version 4, client
Domain Name System (query)						
Transaction ID: 0x00ab						
Flags: 0x0100 Standard query						
Questions: 1						
Answer RRs: 0						
Authority RRs: 0						
Additional RRs: 1						
Queries						

Figure 11: Captura NSLookup

4	0.014954832	192.168.1.254	10.0.2.15	DNS	112 Standard query response 0x/ce4 AAAA www.goog
5	0.015483480	10.0.2.15	142.250.201.67	ICMP	98 Echo (ping) request id=0x0001, seq=1/256, t
6	0.035128460	142.250.201.67	10.0.2.15	ICMP	98 Echo (ping) reply id=0x0001, seq=1/256, t
7	0.035434210	10.0.2.15	192.168.1.254	DNS	98 Standard query 0x0dce PTR 67.201.250.142.in-
8	0.041080703	192.168.1.254	10.0.2.15	DNS	125 Standard query response 0x0dce PTR 67.201.25
9	0.041640298	10.0.2.15	192.168.1.254	DNS	87 Standard query 0xd1f5 PTR 67.201.250.142.in-
10	0.045099495	192.168.1.254	10.0.2.15	DNS	114 Standard query response 0xd1f5 PTR 67.201.25
11	1.016958323	10.0.2.15	142.250.201.67	ICMP	98 Echo (ping) request id=0x0001, seq=2/512, t
12	1.043169481	142.250.201.67	10.0.2.15	ICMP	98 Echo (ping) reply id=0x0001, seq=2/512, t
13	1.952176980	10.0.2.15	193.136.152.71	NTP	90 NTP Version 4, client
14	1.961719128	193.136.152.71	10.0.2.15	NTP	90 NTP Version 4, server
15	2.018705582	10.0.2.15	142.250.201.67	ICMP	98 Echo (ping) request id=0x0001, seq=3/768, t
16	2.037875599	142.250.201.67	10.0.2.15	ICMP	98 Echo (ping) reply id=0x0001, seq=3/768, t
17	2.952590438	10.0.2.15	194.117.47.42	NTP	90 NTP Version 4, client
18	2.962699728	194.117.47.42	10.0.2.15	NTP	90 NTP Version 4, server
19	3.021197997	10.0.2.15	142.250.201.67	ICMP	98 Echo (ping) request id=0x0001, seq=4/1024, t
20	3.040128708	142.250.201.67	10.0.2.15	ICMP	98 Echo (ping) reply id=0x0001, seq=4/1024, t
21	3.952781304	10.0.2.15	194.8.30.16	NTP	90 NTP Version 4, client
22	3.969998937	194.8.30.16	10.0.2.15	NTP	90 NTP Version 4, server
23	4.022643695	10.0.2.15	142.250.201.67	ICMP	98 Echo (ping) request id=0x0001, seq=5/1280, t

▶ Frame 5: 98 bytes on wire (784 bits), 98 bytes captured (784 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: 142.250.201.67
 0100 = Version: 4
 0101 = Header Length: 20 bytes (5)
 ▶ Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
 Total Length: 84
 Identification: 0xb49f (46239)

0000	52 54 00 12 35 02 08 00	27 06 03 48 08 00 45 00	RT..5...H..E..
0010	00 54 b4 9f 40 00 40 01	21 bd 0a 00 02 0f 8e fa	..T..@..!.....
0020	c9 43 08 00 70 78 00 01	00 01 62 29 1b 65 00 00	..C..px...b).e..
0030	00 00 45 24 06 00 00 00	00 00 10 11 12 13 14 15	...ES.....
0040	16 17 18 19 1a 1b 1c 1d	1e 1f 20 21 22 23 24 25!#\$%&
0050	26 27 28 29 2a 2b 2c 2d	2e 2f 30 31 32 33 34 35	&'()*+,-./012345
0060	36 37	67	

Figure 12: Captura Ping

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	10.0.2.15	192.168.1.254	DNS	78	Standard query 0x53db A cisco.di.uminho.pt
2	0.000371510	10.0.2.15	192.168.1.254	DNS	78	Standard query 0x5894 AAAA cisco.di.uminho.p
3	0.837075292	192.168.1.254	10.0.2.15	DNS	94	Standard query response 0x53db A cisco.di.um
4	0.837075693	192.168.1.254	10.0.2.15	DNS	127	Standard query response 0x5894 AAAA cisco.di
5	0.838017261	10.0.2.15	193.136.19.254	UDP	74	50616 → 33434 Len=32
6	0.838251430	10.0.2.15	10.0.2.15	ICMP	70	Time-to-live exceeded (Time to live exceedec
7	0.838582097	10.0.2.15	193.136.19.254	UDP	74	33852 → 33435 Len=32
8	0.838849765	10.0.2.15	10.0.2.15	ICMP	70	Time-to-live exceeded (Time to live exceedec
9	0.838932314	10.0.2.15	193.136.19.254	UDP	74	34181 → 33436 Len=32
10	0.839154479	10.0.2.15	10.0.2.15	ICMP	70	Time-to-live exceeded (Time to live exceedec
11	0.839235956	10.0.2.15	193.136.19.254	UDP	74	40832 → 33437 Len=32
12	0.839624311	10.0.2.15	193.136.19.254	UDP	74	35945 → 33438 Len=32
13	0.839888872	10.0.2.15	193.136.19.254	UDP	74	34269 → 33439 Len=32
14	0.840149215	10.0.2.15	193.136.19.254	UDP	74	47392 → 33440 Len=32
15	0.840409899	10.0.2.15	193.136.19.254	UDP	74	44189 → 33441 Len=32
16	0.840685944	10.0.2.15	193.136.19.254	UDP	74	46827 → 33442 Len=32
17	0.840953652	10.0.2.15	193.136.19.254	UDP	74	55118 → 33443 Len=32
18	0.841218915	10.0.2.15	193.136.19.254	UDP	74	50780 → 33444 Len=32
19	0.841482615	10.0.2.15	193.136.19.254	UDP	74	55905 → 33445 Len=32
20	0.841957818	10.0.2.15	193.136.19.254	UDP	74	34205 → 33446 Len=32
21	0.842225706	10.0.2.15	193.136.19.254	UDP	74	47029 → 33447 Len=32
22	0.842488274	10.0.2.15	193.136.19.254	UDP	74	53359 → 33448 Len=32
23	0.842934768	10.0.2.15	193.136.19.254	UDP	74	33018 → 33449 Len=32

▶ Frame 11: 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
 0100 = Version: 4
 0101 = Header Length: 20 bytes (5)
 ▶ Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
 Total Length: 60
 Identification: 0xee7e (61054)

0000	52 54 00 12 35 02 08 00	27 06 03 48 08 00 45 00	RT..5...H..E..
0010	00 3c ee 7e 00 00 02 11	e8 9d 0a 00 02 0f c1 88	..<.....
0020	13 fe 9f 80 82 9d 00 28	e1 ce 40 41 42 43 44 45(..[ABCRDE
0030	46 47 48 49 4a 4b 4c 4d	4e 4f 50 51 52 53 54 55	FGHIJKLM NOPQRSTU
0040	56 57 58 59 5a 5b 5c 5d	5e 5f	VWXYZ[\]^_

Figure 13: captura Traceroute

3 Conclusões

Após a realização e conclusão deste trabalho prático, é seguro dizer que foi um ótimo projeto para aprofundar conceitos relativos aos diferentes protocolos estudados, sendo estes referentes tanto à camada de transporte como à camada de aplicação, para além de obtermos uma melhor percepção das vantagens e desvantagens que cada um destes protocolos traz, o que, por consequência, nos ajuda a perceber quando devem ser utilizados. O facto de conseguirmos analisar estes conceitos em tempo real, através do uso das ferramentas aconselhadas, torna toda a experiência mais simples e perceptível.

Desta forma, consideramos que o fizemos um bom trabalho e esperamos poder vir a utilizar e aprofundar todos os conceitos estudados no futuro.