
Protocolos da Camada de Transporte

TRABALHO REALIZADO POR:

BRUNO FILIPE DE SOUSA DIAS

FRANCISCO ALVES ANDRADE

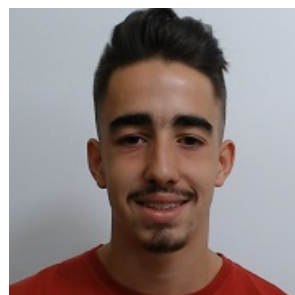
PAULO SILVA SOUSA



A89513
Francisco Andrade



A89583
Bruno Dias



A89465
Paulo Sousa

Índice

1	Questões e Respostas	1
1.1	Pergunta 1	1
1.2	Pergunta 2	6
1.2.1	FTP	6
1.2.2	TFTP	7
1.3	Pergunta 3	8
1.3.1	Uso da camada de transporte	8
1.3.2	Eficiência na transferência	8
1.3.3	Complexidade	9
1.3.4	Segurança	9
1.4	Pergunta 4	10
2	Conclusões	12

1 Questões e Respostas

1.1 Pergunta 1

Inclua no relatório uma tabela em que identifique, para cada comando executado, qual o protocolo de aplicação, o protocolo de transporte, porta de atendimento e overhead de transporte.

De forma a poder responder com transparência a esta questão, foi construída uma tabela semelhante à exposta no enunciado. Para teste dos vários comandos foi utilizado o terminal e o Wireshark como programa de análise do tráfego de rede.

Comado usado	Protocolo de Aplicação	Protocolo de Transporte	Porta de Atendimento	Overhead de transporte em bytes
Ping	PING	-	-	-
Traceroute	TRACEROUTE	UDP	33446	8
Telnet	TELNET	TCP	23	20
Ftp	FTP	TCP	21	20
Tftp	TFTP	UDP	69	8
Browser/http	HTTP	TCP	80	20
Nslookup	DNS	UDP	53	8
Ssh	SSH	TCP	22	20

Table 1: Pergunta 1

Para podermos completar da maneira mais correta a tabela, precisamos de perceber alguns aspetos cruciais. Um deles é o facto de que todos os protocolos que utilizem o protocolo UDP como protocolo de transporte possuírem um *overhead* constante, independentemente da circunstância, de exatamente 8 bytes. O campo TCP possui um *overhead* de normalmente 20 bytes, no entanto, existem casos em que este *overhead* pode ser maior, caso seja utilizado o campo *options* ou não (neste estudo não se verificou o mesmo em nenhum caso).

Para complementar as respostas dadas, seguem seguidamente imagens relativas ao tráfego captado no Wireshark que foi utilizado para obter algumas soluções.

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	fe80::9d95:d407:c7a...	ff02::fb	NDNS	107	Standard query 0x0000 PTR _ipps._tcp.local, "QM" question PTR _ipp._tcp.local, "QM" q...
2	0.000044295	10.0.2.15	224.0.0.251	NDNS	87	Standard query 0x0000 PTR _ipps._tcp.local, "QM" question PTR _ipp._tcp.local, "QM" q...
3	1.284602299	10.0.2.15	192.168.1.1	DNS	86	Standard query 0x9d19 A cc2021.ddns.net OPT
4	1.284715882	10.0.2.15	192.168.1.1	DNS	86	Standard query 0x1ef0 AAAA cc2021.ddns.net OPT
5	1.314911879	192.168.1.1	10.0.2.15	DNS	146	Standard query response 0x1ef0 AAAA cc2021.ddns.net SOA nf1.no-ip.com OPT
6	1.335436353	192.168.1.1	10.0.2.15	DNS	102	Standard query response 0x9d19 A cc2021.ddns.net A 193.136.9.183 OPT
7	1.335717256	10.0.2.15	193.136.9.183	ICMP	98	Echo (ping) request id=0x0001, seq=1/256, ttl=64 (reply in 8)
8	1.356412832	193.136.9.183	10.0.2.15	ICMP	98	Echo (ping) reply id=0x0001, seq=1/256, ttl=51 (request in 7)
9	1.356704851	10.0.2.15	192.168.1.1	DNS	97	Standard query 0xe9ba PTR 183.9.136.193.in-addr.arpa OPT
10	1.401304475	192.168.1.1	10.0.2.15	DNS	128	Standard query response 0xe9ba PTR 183.9.136.193.in-addr.arpa PTR dhcp-43.uminho.pt 0...
11	2.337614341	10.0.2.15	193.136.9.183	ICMP	98	Echo (ping) request id=0x0001, seq=2/512, ttl=64 (reply in 12)
12	2.357661899	193.136.9.183	10.0.2.15	ICMP	98	Echo (ping) reply id=0x0001, seq=2/512, ttl=51 (request in 11)
13	3.338719924	10.0.2.15	193.136.9.183	ICMP	98	Echo (ping) request id=0x0001, seq=3/768, ttl=64 (reply in 14)
14	3.358312528	193.136.9.183	10.0.2.15	ICMP	98	Echo (ping) reply id=0x0001, seq=3/768, ttl=51 (request in 13)
15	4.369362439	10.0.2.15	193.136.9.183	ICMP	98	Echo (ping) request id=0x0001, seq=4/1024, ttl=64 (reply in 16)

Frame 11: 98 bytes on wire (784 bits), 98 bytes captured (784 bits) on interface enp0s3, id 0
 Ethernet II, Src: PcsCompu_d1:8b:d0 (08:00:27:d1:8b:d0), Dst: RealtekU_12:35:02 (52:54:00:12:35:02)
 Internet Protocol Version 4, Src: 10.0.2.15, Dst: 193.136.9.183
 0100 = Version: 4
 0101 = Header Length: 20 bytes (5)
 Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
 Total Length: 84
 Identification: 0xb618 (46616)
 Flags: 0x0000, Don't fragment
 Fragment offset: 0
 Time to live: 64
 Protocol: ICMP (1)
 Header checksum: 0xad42 [validation disabled]
 [Header checksum status: Unverified]
 Source: 10.0.2.15
 Destination: 193.136.9.183
 Internet Control Message Protocol

Figure 1: Captura Ping

O Ping trabalha diretamente com a camada de rede, e portanto não é aplicado qualquer protocolo de transporte não existindo dessa forma Porta de Atendimento nem Over-head de Transporte.

No.	Time	Source	Destination	Protocol	Length	Info
16	0.070818160	10.0.2.15	193.136.9.183	UDP	74	39358 → 33442 Len=32
17	0.070836789	10.0.2.15	193.136.9.183	UDP	74	41566 → 33443 Len=32
18	0.070873128	10.0.2.15	193.136.9.183	UDP	74	55150 → 33444 Len=32
19	0.070895434	10.0.2.15	193.136.9.183	UDP	74	58785 → 33445 Len=32
20	0.070944090	10.0.2.15	193.136.9.183	UDP	74	44554 → 33446 Len=32
21	0.070961011	10.0.2.15	193.136.9.183	UDP	74	56939 → 33447 Len=32
22	0.070995775	10.0.2.15	193.136.9.183	UDP	74	50700 → 33448 Len=32
23	0.071007992	10.0.2.15	193.136.9.183	UDP	74	56236 → 33449 Len=32
24	0.071323422	10.0.2.15	192.168.1.1	DNS	92	Standard query 0x3021 PTR 2.2.0.10.in-addr.arpa OPT
25	0.072824115	10.0.2.2	10.0.2.15	ICMP	70	Destination unreachable (Port unreachable)
26	0.073897417	10.0.2.2	10.0.2.15	ICMP	70	Destination unreachable (Port unreachable)
27	0.073907170	10.0.2.2	10.0.2.15	ICMP	70	Destination unreachable (Port unreachable)

Frame 20: 74 bytes on wire (592 bits), 74 bytes captured (592 bits) on interface enp0s3, id 0
 Ethernet II, Src: PcsCompu_d1:8b:d0 (08:00:27:d1:8b:d0), Dst: RealtekU_12:35:02 (52:54:00:12:35:02)
 Internet Protocol Version 4, Src: 10.0.2.15, Dst: 193.136.9.183
 0100 = Version: 4
 0101 = Header Length: 20 bytes (5)
 Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
 Total Length: 60
 Identification: 0x1678 (5752)
 Flags: 0x0000
 Fragment offset: 0
 Time to live: 5
 Protocol: UDP (17)
 Header checksum: 0xc7eb [validation disabled]
 [Header checksum status: Unverified]
 Source: 10.0.2.15
 Destination: 193.136.9.183
 User Datagram Protocol, Src Port: 44554, Dst Port: 33446
 Source Port: 44554
 Destination Port: 33446
 Length: 40
 Checksum: 0xd707 [unverified]
 [Checksum Status: Unverified]
 [Stream index: 14]
 [Timestamps]
 Data (32 bytes)

Figure 2: Captura Traceroute

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	10.0.2.15	192.168.1.1	DNS	86	Standard query 0x19ab A cc2021.ddns.net OPT
2	0.000128519	10.0.2.15	192.168.1.1	DNS	86	Standard query 0xb835 AAAA cc2021.ddns.net OPT
3	0.029902531	192.168.1.1	10.0.2.15	DNS	146	Standard query response 0xb835 AAAA cc2021.ddns.net SOA nf1.n...
4	0.060488360	192.168.1.1	10.0.2.15	DNS	102	Standard query response 0x19ab A cc2021.ddns.net A 193.136.9...
5	0.060727391	10.0.2.15	193.136.9.183	TCP	74	44318 → 23 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 T...
6	0.081303290	193.136.9.183	10.0.2.15	TCP	60	23 → 44318 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1460
7	0.081332870	10.0.2.15	193.136.9.183	TCP	54	44318 → 23 [ACK] Seq=1 Ack=1 Win=64240 Len=0
8	0.081466066	10.0.2.15	193.136.9.183	TELNET	81	Telnet Data ...
9	0.081694267	193.136.9.183	10.0.2.15	TCP	60	23 → 44318 [ACK] Seq=1 Ack=28 Win=65535 Len=0
10	9.934047864	10.0.2.15	193.136.9.183	TELNET	56	Telnet Data ...
11	9.934279536	193.136.9.183	10.0.2.15	TCP	60	23 → 44318 [ACK] Seq=1 Ack=30 Win=65535 Len=0
12	14.163171533	10.0.2.15	193.136.9.183	TELNET	58	Telnet Data ...
13	14.163402878	193.136.9.183	10.0.2.15	TCP	60	23 → 44318 [ACK] Seq=1 Ack=34 Win=65535 Len=0
14	15.130810368	193.136.9.183	10.0.2.15	TELNET	66	Telnet Data ...
15	15.130839075	10.0.2.15	193.136.9.183	TCP	54	44318 → 23 [ACK] Seq=34 Ack=13 Win=64228 Len=0
▶ Frame 14: 66 bytes on wire (528 bits), 66 bytes captured (528 bits) on interface enp0s3, id 0 ▶ Ethernet II, Src: RealtekU_12:35:02 (52:54:00:12:35:02), Dst: PcsCompu_d1:8b:d0 (08:00:27:d1:8b:d0) ▶ Internet Protocol Version 4, Src: 193.136.9.183, Dst: 10.0.2.15 ▼ Transmission Control Protocol, Src Port: 23, Dst Port: 44318, Seq: 1, Ack: 34, Len: 12 Source Port: 23 Destination Port: 44318 [Stream index: 0] [TCP Segment Len: 12] Sequence number: 1 (relative sequence number) Sequence number (raw): 64002 [Next sequence number: 13 (relative sequence number)] Acknowledgment number: 34 (relative ack number) Acknowledgment number (raw): 2186708540 6101 ... = Header Length: 20 bytes (5) ▶ Flags: 0x018 (PSH, ACK) Window size value: 65535 [Calculated window size: 65535] [Window size scaling factor: -2 (no window scaling used)] Checksum: 0x2643 [unverified] [Checksum Status: Unverified] Urgent pointer: 0 ▶ [SEQ/ACK analysis] ▶ [Timestamps] TCP payload (12 bytes) ▼ Telnet ▶ Do Terminal Type ▶ Do Terminal Speed ▶ Do X Display Location ▶ Do New Environment Option						

Figure 3: Captura Telnet

No.	Time	Source	Destination	Protocol	Length	Info
8	0.124630398	193.136.9.183	10.0.2.15	FTP	74	Response: 220 (vsFTPd 2.3.5)
9	0.124657158	10.0.2.15	193.136.9.183	TCP	54	40324 → 21 [ACK] Seq=1 Ack=21 Win=64220 Len=0
10	4.640976969	10.0.2.15	193.136.9.183	FTP	63	Request: USER cc
11	4.641175238	193.136.9.183	10.0.2.15	TCP	60	21 → 40324 [ACK] Seq=21 Ack=10 Win=65535 Len=0
12	4.654090642	193.136.9.183	10.0.2.15	FTP	88	Response: 331 Please specify the password.
13	4.665001560	10.0.2.15	193.136.9.183	TCP	54	40324 → 21 [ACK] Seq=10 Ack=55 Win=64186 Len=0
14	8.376931207	10.0.2.15	193.136.9.183	FTP	67	Request: PASS cc2021
15	8.377180236	193.136.9.183	10.0.2.15	TCP	60	21 → 40324 [ACK] Seq=55 Ack=23 Win=65535 Len=0
16	8.504725810	193.136.9.183	10.0.2.15	FTP	77	Response: 230 Login successful.
17	8.504737940	10.0.2.15	193.136.9.183	TCP	54	40324 → 21 [ACK] Seq=23 Ack=78 Win=64163 Len=0
18	8.504800779	10.0.2.15	193.136.9.183	FTP	60	Request: SYST
19	8.504930393	193.136.9.183	10.0.2.15	TCP	60	21 → 40324 [ACK] Seq=78 Ack=29 Win=65535 Len=0
20	8.524700470	193.136.9.183	10.0.2.15	FTP	73	Response: 215 UNIX Type: L8
21	8.524725609	10.0.2.15	193.136.9.183	TCP	54	40324 → 21 [ACK] Seq=29 Ack=97 Win=64144 Len=0
▶ Frame 20: 73 bytes on wire (584 bits), 73 bytes captured (584 bits) on interface enp0s3, id 0 ▶ Ethernet II, Src: RealtekU_12:35:02 (52:54:00:12:35:02), Dst: PcsCompu_d1:8b:d0 (08:00:27:d1:8b:d0) ▶ Internet Protocol Version 4, Src: 193.136.9.183, Dst: 10.0.2.15 ▼ Transmission Control Protocol, Src Port: 21, Dst Port: 40324, Seq: 78, Ack: 29, Len: 19 Source Port: 21 Destination Port: 40324 [Stream index: 0] [TCP Segment Len: 19] Sequence number: 78 (relative sequence number) Sequence number (raw): 7616079 [Next sequence number: 97 (relative sequence number)] Acknowledgment number: 29 (relative ack number) Acknowledgment number (raw): 113108614 6101 ... = Header Length: 20 bytes (5) ▶ Flags: 0x018 (PSH, ACK) Window size value: 65535 [Calculated window size: 65535] [Window size scaling factor: -2 (no window scaling used)] Checksum: 0xaf7a [unverified] [Checksum Status: Unverified] Urgent pointer: 0 ▶ [SEQ/ACK analysis] ▶ [Timestamps] TCP payload (19 bytes) ▼ File Transfer Protocol (FTP) ▶ 215 UNIX Type: L8\r\n Response code: NAME system type (215) Response arg: UNIX Type: L8 [Current working directory:]						

Figure 4: Captura FTP

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	10.0.2.15	192.168.1.1	DNS	86	Standard query 0x8b86 A cc2021.ddns.net OPT
2	0.000167499	10.0.2.15	192.168.1.1	DNS	86	Standard query 0xaade AAAA cc2021.ddns.net OPT
3	0.041681735	192.168.1.1	10.0.2.15	DNS	102	Standard query response 0x8b86 A cc2021.ddns.net A 193.136.9...
4	0.067884466	192.168.1.1	10.0.2.15	DNS	146	Standard query response 0xaade AAAA cc2021.ddns.net SOA nf1.n...
5	0.068146650	10.0.2.15	193.136.9.183	TFTP	86	Read Request, File: file1, Transfer type: octet, tsize=0, blk...
6	7.339309931	10.0.2.15	193.136.9.183	TFTP	86	Read Request, File: file1, Transfer type: octet, tsize=0, blk...
7	12.603113949	PcsCompu_d1:8b:d0	RealtekU_12:35:02	ARP	42	Who has 10.0.2.2? Tell 10.0.2.15
8	12.603347245	RealtekU_12:35:02	PcsCompu_d1:8b:d0	ARP	60	10.0.2.2 is at 52:54:00:12:35:02
9	13.398378260	10.0.2.15	193.136.9.183	TFTP	86	Read Request, File: file1, Transfer type: octet, tsize=0, blk...
10	20.414203456	10.0.2.15	193.136.9.183	TFTP	86	Read Request, File: file1, Transfer type: octet, tsize=0, blk...
11	27.440175875	10.0.2.15	193.136.9.183	TFTP	86	Read Request, File: file1, Transfer type: octet, tsize=0, blk...
12	31.199479446	fe80::9d95:d407:c7a...	ff02::fb	MDNS	107	Standard query 0x0000 PTR _ipps._tcp.local, "QM" question PTR...
13	31.199565554	10.0.2.15	224.0.0.251	MDNS	87	Standard query 0x0000 PTR _ipps._tcp.local, "QM" question PTR...
14	34.451234105	10.0.2.15	193.136.9.183	TFTP	86	Read Request, File: file1, Transfer type: octet, tsize=0, blk...
15	41.463123724	10.0.2.15	193.136.9.183	TFTP	86	Read Request, File: file1, Transfer type: octet, tsize=0, blk...
Frame 10: 86 bytes on wire (688 bits), 86 bytes captured (688 bits) on interface enp0s3, id 0						
Ethernet II, Src: PcsCompu_d1:8b:d0 (08:00:27:d1:8b:d0), Dst: RealtekU_12:35:02 (52:54:00:12:35:02)						
Internet Protocol Version 4, Src: 10.0.2.15, Dst: 193.136.9.183						
User Datagram Protocol, Src Port: 42917, Dst Port: 69						
Source Port: 42917						
Destination Port: 69						
Length: 52						
Checksum: 0xd793 [unverified]						
[Checksum Status: Unverified]						
[Stream index: 2]						
[Timestamps]						
Trivial File Transfer Protocol						
Opcode: Read Request (1)						
Source File: file1						
Type: octet						
Option: tsize = 0						
Option: blksize = 512						
Option: timeout = 6						

Figure 5: Captura TFTP

No.	Time	Source	Destination	Protocol	Length	Info
15	3.914770302	10.0.2.15	192.168.1.1	DNS	84	Standard query 0x514f AAAA www.uminho.pt OPT
16	3.949152460	192.168.1.1	10.0.2.15	DNS	138	Standard query response 0x514f AAAA www.uminho.pt SOA dns.umi...
17	3.949469832	10.0.2.15	193.137.9.114	TCP	74	55716 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 T...
18	3.949593595	10.0.2.15	193.137.9.114	TCP	74	55718 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 T...
19	3.972089532	193.137.9.114	10.0.2.15	TCP	60	80 → 55716 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1460
20	3.972118917	10.0.2.15	193.137.9.114	TCP	54	55716 → 80 [ACK] Seq=1 Ack=1 Win=64240 Len=0
21	3.972635093	10.0.2.15	193.137.9.114	HTTP	426	GET / HTTP/1.1
22	3.972670827	193.137.9.114	10.0.2.15	TCP	60	80 → 55716 [ACK] Seq=1 Ack=373 Win=65535 Len=0
23	3.972971528	193.137.9.114	10.0.2.15	TCP	60	80 → 55718 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1460
24	3.972985627	10.0.2.15	193.137.9.114	TCP	54	55718 → 80 [ACK] Seq=1 Ack=1 Win=64240 Len=0
25	3.994605095	193.137.9.114	10.0.2.15	HTTP	180	HTTP/1.0 302 Moved Temporarily
26	3.994622906	10.0.2.15	193.137.9.114	TCP	54	55716 → 80 [ACK] Seq=373 Ack=127 Win=64114 Len=0
27	4.001513239	10.0.2.15	192.168.1.1	DNS	84	Standard query 0xf04b AAAA www.uminho.pt OPT
28	4.027160513	192.168.1.1	10.0.2.15	DNS	138	Standard query response 0xf04b AAAA www.uminho.pt SOA dns.umi...
Frame 21: 426 bytes on wire (3408 bits), 426 bytes captured (3408 bits) on interface enp0s3, id 0						
Ethernet II, Src: PcsCompu_d1:8b:d0 (08:00:27:d1:8b:d0), Dst: RealtekU_12:35:02 (52:54:00:12:35:02)						
Internet Protocol Version 4, Src: 10.0.2.15, Dst: 193.137.9.114						
Transmission Control Protocol, Src Port: 55716, Dst Port: 80, Seq: 1, Ack: 1, Len: 372						
Source Port: 55716						
Destination Port: 80						
[Stream index: 6]						
[TCP Segment Len: 372]						
Sequence number: 1 (relative sequence number)						
Sequence number (raw): 1847118856						
[Next sequence number: 373 (relative sequence number)]						
Acknowledgment number: 1 (relative ack number)						
Acknowledgment number (raw): 50496002						
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: 0xd898 [unverified]						
[Checksum Status: Unverified]						
Urgent pointer: 0						
[SEQ/ACK analysis]						
[Timestamps]						
TCP payload (372 bytes)						
Hypertext Transfer Protocol						
GET / HTTP/1.1\r\n						
Host: www.uminho.pt\r\n						
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:86.0) Gecko/20100101 Firefox/86.0\r\n						
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8\r\n						
Accept-Language: en-US,en;q=0.5\r\n						
Accept-Encoding: gzip, deflate\r\n						
Connection: keep-alive\r\n						
Cookie: _ga=GAI.2.369414041.1614126369\r\n						
Upgrade-Insecure-Requests: 1\r\n						
\r\n						
[Full request URI: http://www.uminho.pt/]						
[HTTP request 1/1]						
[Response in frame: 25]						

Figure 6: Captura Browser/HTTP

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	10.0.2.15	192.168.1.1	DNS	84	Standard query 0xf216 A www.uminho.pt OPT
2	0.020986048	192.168.1.1	10.0.2.15	DNS	100	Standard query response 0xf216 A www.uminho.pt A 193.137.9.11
3	0.021554284	10.0.2.15	192.168.1.1	DNS	84	Standard query 0x4810 AAAA www.uminho.pt OPT
4	0.056119032	192.168.1.1	10.0.2.15	DNS	138	Standard query response 0x4810 AAAA www.uminho.pt SOA dns.umi

▶ Frame 2: 100 bytes on wire (800 bits), 100 bytes captured (800 bits) on interface enp0s3, id 0
 ▶ Ethernet II, Src: RealtekU_12:35:02 (52:54:00:12:35:02), Dst: PcsCompu_d1:8b:d0 (08:00:27:d1:8b:d0)
 ▶ Internet Protocol Version 4, Src: 192.168.1.1, Dst: 10.0.2.15
 ▶ User Datagram Protocol, Src Port: 53, Dst Port: 41896
 Source Port: 53
 Destination Port: 41896
 Length: 66
 Checksum: 0x41b1 [unverified]
 [Checksum Status: Unverified]
 [Stream index: 0]
 [Timestamps]
 Domain Name System (response)
 Transaction ID: 0xf216
 Flags: 0x8100 Standard query response, No error
 Questions: 1
 Answer RRs: 1
 Authority RRs: 0
 Additional RRs: 1
 Queries
 Answers
 Additional records
 [Request In: 1]
 [Time: 0.020986048 seconds]

Figure 7: Captura Nslookup

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	10.0.2.15	192.168.1.1	DNS	86	Standard query 0x0455 A cc2021.ddns.net OPT
2	0.000142689	10.0.2.15	192.168.1.1	DNS	86	Standard query 0xa279 AAAA cc2021.ddns.net OPT
3	0.031752864	192.168.1.1	10.0.2.15	DNS	146	Standard query response 0xa279 AAAA cc2021.ddns.net SOA nf1.n...
4	0.050777826	192.168.1.1	10.0.2.15	DNS	102	Standard query response 0x0455 A cc2021.ddns.net A 193.136.9...
5	0.050992043	10.0.2.15	193.136.9.183	TCP	74	37054 → 22 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 T...
6	0.070627823	193.136.9.183	10.0.2.15	TCP	60	22 → 37054 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1460
7	0.070660938	10.0.2.15	193.136.9.183	TCP	54	37054 → 22 [ACK] Seq=1 Ack=1 Win=64240 Len=0
8	0.070967817	10.0.2.15	193.136.9.183	SSHv2	95	Client: Protocol (SSH-2.0-OpenSSH 8.2p1 Ubuntu-4ubuntu0.1)
9	0.071216233	193.136.9.183	10.0.2.15	TCP	60	22 → 37054 [ACK] Seq=1 Ack=42 Win=65535 Len=0
10	0.125945519	193.136.9.183	10.0.2.15	SSHv2	95	Server: Protocol (SSH-2.0-OpenSSH 5.9p1 Debian-Subuntu1.4)
11	0.125958699	10.0.2.15	193.136.9.183	TCP	54	37054 → 22 [ACK] Seq=42 Ack=42 Win=64199 Len=0
12	0.126297330	10.0.2.15	193.136.9.183	SSHv2	1566	Client: Key Exchange Init
13	0.126596793	193.136.9.183	10.0.2.15	TCP	60	22 → 37054 [ACK] Seq=42 Ack=1502 Win=65535 Len=0
14	0.126596764	193.136.9.183	10.0.2.15	TCP	60	22 → 37054 [ACK] Seq=42 Ack=1554 Win=65535 Len=0
15	0.148215867	193.136.9.183	10.0.2.15	SSHv2	1038	Server: Key Exchange Init

▶ Frame 10: 95 bytes on wire (760 bits), 95 bytes captured (760 bits) on interface enp0s3, id 0
 ▶ Ethernet II, Src: RealtekU_12:35:02 (52:54:00:12:35:02), Dst: PcsCompu_d1:8b:d0 (08:00:27:d1:8b:d0)
 ▶ Internet Protocol Version 4, Src: 193.136.9.183, Dst: 10.0.2.15
 ▶ Transmission Control Protocol, Src Port: 22, Dst Port: 37054, Seq: 1, Ack: 42, Len: 41
 Source Port: 22
 Destination Port: 37054
 [Stream index: 0]
 [TCP Segment Len: 41]
 Sequence number: 1 (relative sequence number)
 Sequence number (raw): 28032002
 [Next sequence number: 42 (relative sequence number)]
 Acknowledgment number: 42 (relative ack number)
 Acknowledgment number (raw): 1143975211
 0101... = Header Length: 20 bytes (5)
 Flags: 0x018 (PSH, ACK)
 Window size value: 65535
 [Calculated window size: 65535]
 [Window size scaling factor: -2 (no window scaling used)]
 Checksum: 0xa658 [unverified]
 [Checksum Status: Unverified]
 Urgent pointer: 0
 [SEQ/ACK analysis]
 [Timestamps]
 TCP payload (41 bytes)
 SSH Protocol

Figure 8: Captura SSH

1.2 Pergunta 2

Uma representação num diagrama temporal das transferências da file1 por FTP e TFTP respetivamente. Se for caso disso, identifique as fases de estabelecimento de conexão, transferência de dados e fim de conexão. Identifica também claramente os tipos de segmentos trocados e os números de sequência usados quer nos dados como nas confirmações.

1.2.1 FTP

Um cliente realiza uma conexão TCP para a porta 21 do servidor. Essa conexão, chamada de conexão de controlo, permanece aberta ao longo da sessão enquanto uma segunda conexão, chamada conexão de dados, é estabelecida na porta 20 do servidor e em alguma porta do cliente, como requisitado para a transferência de arquivos. Esta última é extremamente importante no processo de realização da transferência dos ficheiros, apesar dos clientes se conectarem à porta 21 dos servidores remotos para iniciarem a opera

O protocolo FTP pode ser executado em dois modos distintos que determinam como é que a conexão de dados é estabelecida. No modo **Ativo**, o cliente envia para o servidor o endereço IP e o número da porta na qual ele irá ouvir e então o servidor inicia a conexão TCP.

Em situações onde o cliente se encontra atrás de uma *firewall* e inapto para aceitar entradas de conexão, é utilizado o modo **Passivo**. O cliente envia um comando PASV para o servidor e recebe um endereço IP e um número de porta como resposta, os quais o cliente utiliza para abrir a conexão de dados com o servidor.

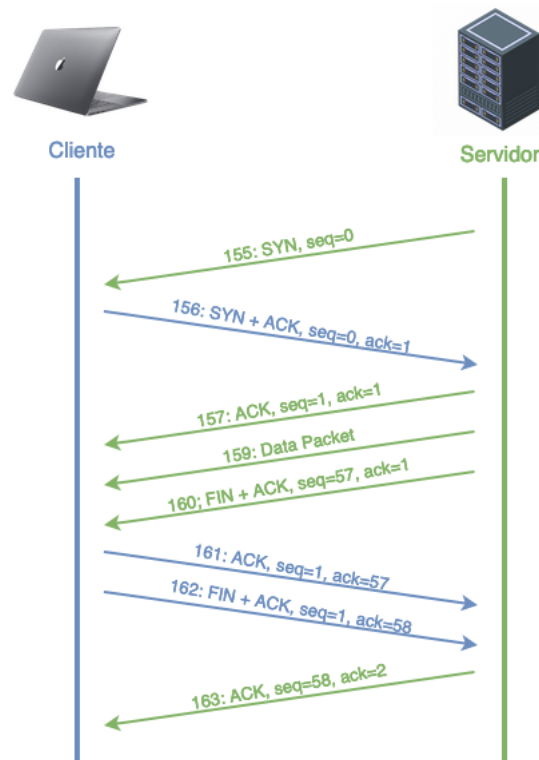


Figure 9: Diagrama FTP

155	34.532098249	10.1.1.1	10.4.4.1	TCP	74	20	-	40201	[SYN]	Seq=0	Win=64240	Len=0	MSS=1460	SACK_PERM=1	TSval=2882493277	TSecr=0	-
156	34.5320916188	10.4.4.1	10.1.1.1	TCP	74	40201	-	20	[SYN, ACK]	Seq=0	Ack=1	Win=65160	Len=0	MSS=1460	SACK_PERM=1	TSval=328993899	TSecr=0
157	34.535094156	10.1.1.1	10.4.4.1	TCP	66	20	-	40201	[ACK]	Seq=1	Ack=1	Win=64256	Len=0	TSval=2882493280	TSecr=328993899		
158	34.535096778	10.1.1.1	10.4.4.1	FTP	129				Response: 150	Opening BINARY mode data connection for file1 (56 bytes).							
159	34.535098085	10.1.1.1	10.4.4.1	FTP-DA	122				FTP Data: 56 bytes (PORT) (RETR file1)								
160	34.535099246	10.1.1.1	10.4.4.1	TCP	66	20	-	40201	[FIN, ACK]	Seq=57	Ack=1	Win=64256	Len=0	TSval=2882493280	TSecr=328993899		
161	34.535114796	10.4.4.1	10.1.1.1	TCP	66	40201	-	20	[ACK]	Seq=1	Ack=57	Win=65152	Len=0	TSval=3289938962	TSecr=2882493280		
162	34.535325795	10.4.4.1	10.1.1.1	TCP	66	40201	-	20	[FIN, ACK]	Seq=1	Ack=58	Win=65152	Len=0	TSval=3289938962	TSecr=2882493280		
163	34.535589417	10.1.1.1	10.4.4.1	TCP	66	20	-	40201	[ACK]	Seq=58	Ack=2	Win=64256	Len=0	TSval=2882493281	TSecr=3289938962		

Figure 10: Captura FTP

1.2.2 TFTP

Neste protocolo transferência começa com um de dois pedidos possíveis, de leitura ou de escrita, num arquivo, pedido esse que serve também como solicitação de conexão. Consequentemente, o servidor pode conceder o pedido, e nesse caso a conexão estará aberta e o arquivo é emitido em blocos fixos de 512 bytes. Exceccionalmente, se o pacote for superior aos 512 bytes de limite, este será fragmentado em pacotes de um tamanho máximo de 512 bytes, até completar a quantidade de dados a transmitir.

Tal como podemos ver na figura 11 do diagrama temporal, o Cliente inicia o processo com um *Read Request* ao servidor, ao qual este responde com um *Data Packet* com os dados de envio pretendidos. Para finalizar o processo, o Cliente envia um *Acknowledgement (ACK)*, relatando o sucesso da operação realizada, isto é, os dados terem sido recebidos sem problemas.

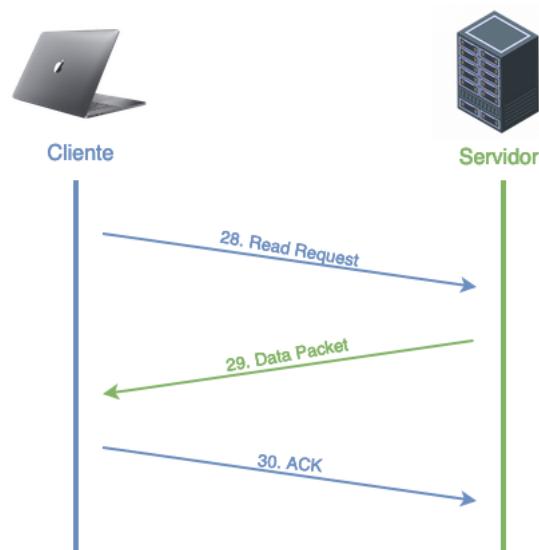


Figure 11: Diagrama TFTP

28	43.405566219	10.4.4.1	10.1.1.1	TFTP	56	Read Request, File: file1, Transfer type: octet
29	43.411891258	10.1.1.1	10.4.4.1	TFTP	102	Data Packet, Block: 1 (last)
30	43.411939854	10.4.4.1	10.1.1.1	TFTP	46	Acknowledgement, Block: 1

Figure 12: Captura TFTP

1.3 Pergunta 3

Com base nas experiências realizadas, distinga e compare sucintamente as quatro aplicações de transferência de ficheiros que usou nos seguintes pontos (i) uso da camada de transporte; (ii) eficiência na transferência; (iii) complexidade; (iv) segurança;

Para respondermos à seguinte pergunta, precisamos de saber as aplicações de transferência de ficheiros que pretendemos estudar. Estas são: **SFTP** (SSH File Transfer Protocol), **FTP** (File Transfer Protocol), **TFTP** (Trivial File Transfer Protocol) e **HTTP** (Hypertext Transfer Protocol).

1.3.1 Uso da camada de transporte

- **SFTP** - Protocolo utilizado é o TCP
- **FTP** - Protocolo utilizado é o TCP
- **TFTP** - Protocolo utilizado é o UDP
- **HTTP** - Protocolo utilizado é o TCP

1.3.2 Eficiência na transferência

- **SFTP** - Comporta-se da mesma maneira que o FTP, sendo entanto um pouco mais forte e ainda mais fiável, dado que os dados se encontram encriptados.
- **FTP** - Bastante fiável no processo de transferências de dados, no entanto, peca no seu custo de eficiência. Esta fiabilidade é adquirida através do uso de *Acknowledges* garantido a transmissão de um segmento de Data, e consequentemente os pacotes irão levar um maior overhead. No entanto, e mesmo por usarmos *Acknowledges*, este processo precisa dos mesmos para conseguir prosseguir corretamente. Os dados não são encontrados encriptados, criando uma brecha para possíveis ataques e interceções.
- **TFTP** - Dado que é utilizado o protocolo UDP, o protocolo TFTP perde bastante da sua fiabilidade. Isto acontece, porque ao contrário do que acontece com o SFTP ou o FTP, o protocolo TFTP não faz qualquer uso de *Acknowledges* tornando-se assim impossível perceber e confirmar se um pacote é entregue com sucesso e sem qualquer erro. Como não possui *Acknowledges* os pacotes irão levar um menor overhead. Dado que este protocolo possui este comportamento, é então necessário enviar o mesmo pacote várias vezes, para existir uma certeza de que, pelo menos um pacote, irá chegar ao seu destino corretamente. Além disso, e pelo facto de não utilizar *Acknowledges*, perde fiabilidade, mas coincidentemente não necessita de haver tempos de espera por confirmações de pacotes que chegaram com sucesso e assim este pacote pode ganhar um pouco na sua velocidade, tornando-se mais rápido.

-
- **HTTP** - O protocolo HTTP permite que vários HTTP requests sejam enviados ao mesmo tempo, para uma mesma ligação TCP (como referido acima) sem ser necessário aguardar qualquer resposta. Isto acontece devido à utilização de pipelining, que para além de ter esse benefício, torna tudo muito mais rápido e mais eficiente.

1.3.3 Complexidade

- **SFTP** - Este protocolo possui uma grande complexidade, devido à vasta amplitude de funcionalidade que oferece. Estas funcionalidades são: acesso a dados, transferência de dados e gestão de dados. Todas estas funcionalidades possuem um grande custo ao nível de processamento e por isso tornam o protocolo em questão bastante complexo.
- **FTP** - A grande fiabilidade deste protocolo, referida anteriormente faz com que, tal como é esperado, este protocolo seja portador de uma grande complexidade. Além da fiabilidade do mesmo, este ainda possui a funcionalidade de transferir dados em paralelo, havendo a necessidade de estabelecer novas conexões que podem diferir muito na sua taxa e velocidade de transferência de dados e pacotes. Tudo isto faz com que o protocolo seja bastante complexo.
- **TFTP** - Este protocolo passa por ser uma versão mais simplificada do protocolo FTP, sendo mesmo confirmado pelo seu nome Trivial FTP. Uma das diferenças é o facto de ser utilizado o protocolo UDP em substituição do protocolo TCP, o que nos mostra logo que é um processo muito menos complexo. Além disso, todo o protocolo possui um menor leque de funcionalidades, comparando ao protocolo FTP. Assim sendo, este protocolo revela-se pouco complexo.
- **HTTP** - O protocolo HTTP é um protocolo utilizado por sistemas de informação distribuídos e colaborativos, e é também ele um protocolo que se banha de várias (e complexas) funcionalidades. Este permite escalabilidade de sistemas, desacoplamento do mesmo, entre várias outras funcionalidades, o que faz com que este protocolo seja muito complexo.

1.3.4 Segurança

- **SFTP** - O protocolo SFTP é de todos os protocolos em estudo o que providencia uma maior segurança. Este utiliza uma arquitetura em camadas, que juntamente fornecem encriptação e proteção da integridade de dados, autenticação do servidor e autenticação e manipulação dos clientes. Assim, podemos perceber o porquê do protocolo fornecer uma segurança tão grande.
- **FTP** - Este protocolo, fornece mecanismos de autenticação, no entanto, é extremamente inseguro. Isto acontece principalmente por não haver encriptação de dados. Dessa maneira, qualquer pessoa capaz de captar os pacotes, quer sendo normalmente, quer sendo através de qualquer tipo de ataque, irá ter acesso a toda a informação, estando englobada aquela que não deveria poder ver, incluindo-se usernames, passwords, detalhes de uma conta bancária, entre outros. Assim, revela uma grande precariedade quanto à sua segurança.

-
- **TFTP** - Este protocolo não fornece qualquer tipo de mecanismos de autenticação, e para mais, não oferece também nenhum tipo de encriptação de dados. Assim, podemos facilmente concluir que este protocolo possui um nível de segurança muito fraco.
 - **HTTP** - Este protocolo, apesar de usufruir e de recorrer à autenticação, não possui mais uma vez, mecanismos de encriptação de dados (dado que se processa maioritariamente para realizar transferências de dados na Internet, estando esta representada sobre a forma de texto). Assim, existe a possibilidade de existirem ataques, que permitam a visualização de dados confidenciais, ou até mesmo a alteração dos mesmos. Dessa forma, este protocolo, confirma-se como sendo pouco seguro.

1.4 Pergunta 4

As características das ligações de rede têm uma enorme influência nos níveis de Transporte e de Aplicação. Discuta, relacionando a resposta com as experiências realizadas, as influências das situações de perda ou duplicação de pacotes IP no desempenho global de Aplicações fiáveis (se possível, relacionando com alguns dos mecanismos de transporte envolvidos).

No que toca aos níveis de Transporte e de Aplicação existe muitas das vezes perda e/ou duplicação de pacotes. Os protocolos TCP e UDP tentam então combater estes problemas, recorrendo a algumas soluções (reagindo de maneiras diferentes). Estas situações acontecem por exemplo na LAN3 da Topologia fornecida para a realização deste projeto (à qual vamos recorrer para responder a esta questão).

Para poder construir uma resposta sólida a esta perquisição, é primeiramente necessário reforçar as funcionalidades e características, tanto do protocolo TCP, como do protocolo UDP, relembrando conceitos acima referidos (ou estudados noutros anos):

- **TCP** - O Protocolo TCP realiza deteção e correção de erros, tentando retransmitir um mesmo pacote caso este não chegue com sucesso e sem anomalias ao seu destino, garantido que todos os pacotes são entregues. Dado que existe todo este mecanismo, quando ocorrem retransmissão de pacotes, o débito de transmissão diminui, dado que estamos a reexpedir um pacote já anteriormente enviado. FTP trabalha à base deste protocolo de transporte.
- **UDP** - O Protocolo UDP realiza também deteção de erros, no entanto, não possui qualquer funcionalidade de correção dos mesmos, e assim, se algum pacote possuir alguma anomalia este protocolo apenas o descarta, dado que não possui mecanismos de recuperação ou correção. Isto faz com que não seja possível reconhecer qualquer perda de pacotes. Na eventualidade de se visar a correção destes erros, ficamos inteiramente dependentes de protocolos que se encontram acima do protocolo UDP. Estes fazem uso do número de sequência do pacote, por exemplo, de forma a verificar se a totalidade dos pacotes foi recebida e se esta não possui qualquer irregularidade. TFTP trabalha à base deste protocolo de transporte.

Ficando compreendidos estes conceitos, são testadas então as transferências de um ficheiro, através de FTP e de TFTP, utilizando Sistemas da LAN3, de modo a compreender o que se sucede na eventualidade de acontecer um erro.

124	90.167476525	10.3.3.3	10.1.1.1	FTP	88 Request: PORT 10.3.3.3,211,13
125	90.174579310	10.1.1.1	10.3.3.3	FTP	117 Response: 200 PORT command successful. Consider using PASV.
126	90.174579208	10.3.3.3	10.1.1.1	TCP	66 50408 - 21 [ACK] Seq=171 Ack=635 Win=502 Len=0 TSval=268577921 TSecr=3927648749
127	90.174608399	10.3.3.3	10.1.1.1	FTP	78 Request: RETR file1
128	90.180397860	10.1.1.1	10.3.3.3	TCP	78 21 - 50408 [ACK] Seq=635 Ack=183 Win=510 Len=0 TSval=3927648755 TSecr=268577921 SLE=0
129	90.180487300	10.1.1.1	10.3.3.3	TCP	74 20 - 54029 [SYN] Seq=0 Win=0 Len=0 MSS=4256 SACK_PERM=1 TSval=3927648755 TSecr=0
130	90.180495259	10.3.3.3	10.1.1.1	TCP	74 54029 - 20 [SYN, ACK] Seq=0 Ack=1 Win=65160 Len=0 MSS=1460 SACK_PERM=1 TSval=268577921 TSecr=3927648755
131	90.186129835	10.1.1.1	10.3.3.3	TCP	66 20 - 54029 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=3927648761 TSecr=268577927
132	90.186136511	10.1.1.1	10.3.3.3	TCP	66 [RST] Seq=1 Win=0 Len=0 TSval=3927648761 TSecr=268577927
133	90.186137677	10.1.1.1	10.3.3.3	TCP	129 Response: 150 Opening BINARY mode data connection for file1 (56 bytes)
134	90.186171962	10.3.3.3	10.1.1.1	TCP	66 50408 - 21 [ACK] Seq=183 Ack=698 Win=502 Len=0 TSval=268577933 TSecr=3927648761
135	90.186299785	10.1.1.1	10.3.3.3	FTP-DA	122 FTP Data: 56 bytes (PORT) (RETR file1)
136	90.186301649	10.1.1.1	10.3.3.3	TCP	66 20 - 54029 [FIN, ACK] Seq=57 Ack=1 Win=64256 Len=0 TSval=3927648761 TSecr=268577927
137	90.186307240	10.3.3.3	10.1.1.1	TCP	66 54029 - 20 [ACK] Seq=1 Ack=57 Win=65152 Len=0 TSval=268577933 TSecr=3927648761
138	90.186386568	10.3.3.3	10.1.1.1	TCP	66 54029 - 20 [FIN, ACK] Seq=1 Ack=58 Win=65152 Len=0 TSval=268577933 TSecr=3927648761
139	90.369932913	10.3.3.254	224.0.0.5	OSPF	78 Hello Packet
140	90.369144237	10.3.3.3	10.1.1.1	TCP	66 [TCP Retransmission] Seq=57 Ack=58 Win=65152 Len=0 TSval=2685781 TSecr=3927648975
141	90.400218937	10.1.1.1	10.3.3.3	TCP	66 20 - 54029 [ACK] Seq=58 Ack=2 Win=64256 Len=0 TSval=3927648975 TSecr=268578141
142	90.400260485	10.1.1.1	10.3.3.3	FTP	90 Response: 226 Transfer complete.
143	90.400267027	10.3.3.3	10.1.1.1	TCP	66 50408 - 21 [ACK] Seq=183 Ack=722 Win=502 Len=0 TSval=268578147 TSecr=3927648975

Figure 13: Sistema Corvo faz download do file1 através de FTP

40	34.648096633	10.1.1.1	10.3.3.3	TFTP	102 Data Packet, Block: 1 (last)
41	34.648110069	10.3.3.3	10.1.1.1	ICMP	130 Destination unreachable (Port unreachable)

Figure 14: Sistema Corvo faz download do file1 através de TFTP

As figuras apresentadas acima, exibem o tráfego de rede capturado pelo Wireshark em momentos em que foram feitos downloads do file1, numa através de FTP e na outra através de TFTP. Em ambos os casos é utilizado o Sistema Corvo, da LAN3, fazendo download do file1 através do Serevr1. Em ambos os casos aconteceram erros (objeto de estudo), no entanto, é importante realçar que estes foram poucos e que na maioria das vezes todo o processo ocorria corretamente e com sucesso. Analisando então os casos em estudo:

- No caso da utilização do **FTP**, podemos ver na Figura 13 dois erros que acontecem algumas vezes. Por sorte conseguimos captar ambos os erros num mesmo "ciclo". O primeiro erro que podemos visualizar é o facto de existirem seguimentos ACK repetidos, mas que não interfere no bom sucesso do processo, chegando o pacote ao seu destino final na mesma. O segundo erro é quando um pacote não chega ao seu destino final corretamente: como podemos ver, o Server1 envia um (FIN+ACK) ao sistema Corvo de modo a encerrar a conexão, no entanto o Corvo não responde com nenhum ACK, e então, depois do Server1 esperar um tempo e não receber um ACK, ele voltar a enviar o mesmo pacote, ficando assim explicada a retransmissão de pacotes. Esta complexidade e tentativa de deteção e correção de erros, faz com que a transmissão de pacotes se torne um pouco mais lenta.
- No caso da utilização do **TFTP**, podemos ver que os dados não conseguem chegar ao seu destino final corretamente, mas como não há qualquer deteção ou correção de erros, o próprio Host demora demasiado tempo a constatar que o pacote enviado não foi entregue corretamente, perdendo alguma eficácia. Ao contrário do FTP e dado que não existem quaisquer mecanismos de correção de erros, faz com que a transmissão de pacotes se torne bastante mais rápida.

Assim, ficamos a perceber que, nos casos em que todo o processo ocorria de forma perfeita sem existir qualquer cenário de erro, o processo de transferência dos arquivos era mais rápido através de TFTP do que de FTP. No entanto, quando ocorriam erros, o TFTP perdia bastante tempo na perceção do mesmo e não conseguia recuperar nenhuma informação perdida, ao contrário do FTP que conseguia recuperar de quaisquer eventualidades de erros, dado que possui os devidos mecanismos de deteção e correção dos mesmos. Assim, ficamos comprometidos entre um processo mais rápido contudo sem garantias (TFTP), e um processo um pouco mais demorado, mas que nos garante que o ficheiro e os dados chegam ao seu destino corretamente (FTP). Cabe a nós decidir a rota a tomar, face a cada situação e à primazia que damos a cada caso em específico (velocidade ou certeza de que tudo é entregue corretamente).

2 Conclusões

De modo a elaborar este trabalho, utilizamos a máquina virtual disponibilizada pelos docentes para ter acesso às ferramentas utilizadas para o seu desenvolvimento: o Core e o Wireshark.

Após realizar este projeto conseguimos sedimentar bastantes conhecimentos sobre Protocolos da Camada de Transportes e Protocolos da Camada da Aplicação. Além disso, conseguimos explorar o comportamento destes protocolos e as suas vantagens e desvantagens, tendo em conta a sua aplicação.