



UNIVERSIDADE DO MINHO

DEPARTAMENTO DE INFORMÁTICA

Comunicações por Computador
TP1:Protocolos da Camada de Transporte
Grupo N^o 3 PL6

Gonçalo Almeida (A84610)

Emanuel Rodrigues (A84776)

Lázaro Pinheiro (A86788)

4 de Março de 2020

Conteúdo

1	Questões e Respostas	3
2	Conclusão	15

Capítulo 1

Questões e Respostas

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, como ilustrado no exemplo seguinte:

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	PING	-	-	-
tracert	TRACEROUTE	UDP	33452	33.33%
telnet	TELNET	TCP	23	29.85%
ftp	FTP	TCP	21	30.30%
Tftp	TFTP	UDP	69	47.62%
browser/http	HTTP	TCP	80	10.42%
nslookup	DNS	UDP	53	28.57%
ssh	SSHv2	TCP	22	21.51%

PING

8	1.611899673	fe80::200:ff:feaa:10	ff02::5	OSPF	90 Hello Packet
9	2.003037497	10.1.1.1	10.3.3.1	ICMP	98 Echo (ping) request id=0x001f, seq=3/768, ttl=61 (reply in 10)
10	2.003001281	10.3.3.1	10.1.1.1	ICMP	98 Echo (ping) reply id=0x001f, seq=3/768, ttl=64 (request in 9)

Figura 1.1: Tráfego de pacotes

▶ Frame 9: 98 bytes on wire (784 bits), 98 bytes captured (784 bits) on interface 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: 10.3.3.1
▼ Internet Control Message Protocol
Type: 8 (Echo (ping) request)
Code: 0
Checksum: 0x00a5 [correct]
[Checksum Status: Good]
Identifier (BE): 31 (0x001f)
Identifier (LE): 7936 (0x1f00)
Sequence number (BE): 3 (0x0003)
Sequence number (LE): 768 (0x0300)
[Response frame: 10]
Timestamp from icmp data: Mar 4, 2020 11:29:58.000000000 WET
[Timestamp from icmp data (relative): 0.817223284 seconds]
▶ Data (48 bytes)

Figura 1.2: Trama ping

Este comando utiliza o protocolo de aplicação PING. Como trabalha na camada da rede, não

se aplica o protocolo de transporte, o que implica a não existência de uma porta de atendimento e de um Overhead de transporte.

TRACEROUTE

39	0.130145602	10.0.2.15	193.136.19.254	UDP	74	32777 → 33451	Len=32
40	0.130319126	10.0.2.15	193.136.19.254	UDP	74	58459 → 33452	Len=32
41	0.130474581	10.0.2.15	193.136.19.254	UDP	74	43169 → 33453	Len=32

Figura 1.3: Tráfego de pacotes

▶	Frame 40: 74 bytes on wire (592 bits), 74 bytes captured (592 bits) on interface 0
▶	Ethernet II, Src: PcsCompu_04:03:91 (08:00:27:04:03:91), 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: 0x00db (219)
▶	Flags: 0x0000
	Time to live: 7
	Protocol: UDP (17)
	Header checksum: 0xd141 [validation disabled]
	[Header checksum status: Unverified]
	Source: 10.0.2.15
	Destination: 193.136.19.254
▼	User Datagram Protocol, Src Port: 58459, Dst Port: 33452
	Source Port: 58459
▶	Destination Port: 33452
	Length: 40
	Checksum: 0xe1ce [unverified]
	[Checksum Status: Unverified]
	[Stream index: 22]
▶	Data (32 bytes)

Figura 1.4: Trama traceroute

Este comando utiliza o protocolo de aplicação TRACEROUTE. Podemos verificar que o protocolo de transporte é o UDP (olhando para o campo Protocol), a porta de atendimento é a 33452 (olhando para o campo DST Port) e o Overhead de transporte = Header Length / Total Length = 20 / 60 = 0.3333.

TELNET

9	1.092751714	10.0.2.15	193.136.9.183	TCP	54	43934 → 23 [ACK] Seq=1 Ack=1 Win=29200 Len=0
10	1.093184520	10.0.2.15	193.136.9.183	TELNET	81	Telnet Data ...
11	1.093693398	193.136.9.183	10.0.2.15	TCP	60	23 → 43934 [ACK] Seq=1 Ack=28 Win=65535 Len=0

Figura 1.5: Tráfego de pacotes

```

▶ Frame 10: 81 bytes on wire (648 bits), 81 bytes captured (648 bits) on interface 0
▶ Ethernet II, Src: PcsCompu_04:03:91 (08:00:27:04:03:91), 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: 0x10 (DSCP: Unknown, ECN: Not-ECT)
    Total Length: 67
    Identification: 0x3035 (12341)
    ▶ Flags: 0x4000, Don't fragment
    Time to live: 64
    Protocol: TCP (6)
    Header checksum: 0x3322 [validation disabled]
    [Header checksum status: Unverified]
    Source: 10.0.2.15
    Destination: 193.136.9.183
▼ Transmission Control Protocol, Src Port: 43934, Dst Port: 23, Seq: 1, Ack: 1, Len: 27
    Source Port: 43934
    Destination Port: 23
    [Stream index: 1]
    [TCP Segment Len: 27]
    Sequence number: 1 (relative sequence number)
    [Next sequence number: 28 (relative sequence number)]
    Acknowledgment number: 1 (relative ack number)
    0101 .... = Header Length: 20 bytes (5)
    ▶ Flags: 0x018 (PSH, ACK)
    Window size value: 29200
    [Calculated window size: 29200]
    [Window size scaling factor: -2 (no window scaling used)]
    Checksum: 0xd783 [unverified]
    [Checksum Status: Unverified]
    Urgent pointer: 0
    ▶ [SEQ/ACK analysis]
    ▶ [Timestamps]
    TCP payload (27 bytes)
▶ Telnet

```

Figura 1.6: Trama telnet

Este comando utiliza o protocolo de aplicação TELNET. Podemos verificar que o protocolo de transporte é o TCP (olhando para o campo Protocol), a porta de atendimento é a 23 (olhando para o campo DST Port) e o Overhead de transporte = Header Length / Total Length = 20/67 = 0.2985.

FTP

5	0.063253644	10.1.1.1	10.3.3.1	TCP	66 52784 - 21 [ACK] Seq=1 Ack=21 Win=29312 Len=0 TSval=3727250819 TSecr=1502027481
6	1.699631289	10.1.1.1	10.3.3.1	FTP	80 Request: USER emanuel
7	1.699917241	10.3.3.1	10.1.1.1	TCP	66 21 - 52784 [ACK] Seq=21 Ack=15 Win=29056 Len=0 TSval=1502029118 TSecr=3727252455

Figura 1.7: Tráfego de pacotes

```

▶ Frame 6: 80 bytes on wire (640 bits), 80 bytes captured (640 bits) on interface 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: 10.3.3.1
    0100 .... = Version: 4
    .... 0101 = Header Length: 20 bytes (5)
    ▶ Differentiated Services Field: 0x10 (DSCP: Unknown, ECN: Not-ECT)
    Total Length: 66
    Identification: 0x677c (26492)
    ▶ Flags: 0x4000, Don't fragment
    Time to live: 61
    Protocol: TCP (6)
    Header checksum: 0xbe24 [validation disabled]
    [Header checksum status: Unverified]
    Source: 10.1.1.1
    Destination: 10.3.3.1
▼ Transmission Control Protocol, Src Port: 52784, Dst Port: 21, Seq: 1, Ack: 21, Len: 14
    Source Port: 52784
    Destination Port: 21
    [Stream index: 0]
    [TCP Segment Len: 14]
    Sequence number: 1 (relative sequence number)
    [Next sequence number: 15 (relative sequence number)]
    Acknowledgment number: 21 (relative ack number)
    1000 .... = Header Length: 32 bytes (8)
    ▶ Flags: 0x018 (PSH, ACK)
    Window size value: 229
    [Calculated window size: 29312]
    [Window size scaling factor: 128]
    Checksum: 0x183a [unverified]
    [Checksum Status: Unverified]
    Urgent pointer: 0
    ▶ Options: (12 bytes), No-Operation (NOP), No-Operation (NOP), Timestamps
    ▶ [SEQ/ACK analysis]
    ▶ [Timestamps]
    TCP payload (14 bytes)
▶ File Transfer Protocol (FTP)
    [Current working directory: ]

```

Figura 1.8: Trama ftp

Este comando utiliza o protocolo de aplicação FTP. Podemos verificar que o protocolo de transporte é o TCP (olhando para o campo Protocol), a porta de atendimento é a 21 (olhando para o campo DST Port) e o Overhead de transporte = Header Length / Total Length = 20 / 66 = 0.3030.

TFTP

2	1.525152399	fe80::200:ff:feaa:10	ff02::5	OSPF	90 Hello Packet
3	8.585028189	10.1.1.1	10.3.3.1	TFTP	56 Read Request, File: file1, Transfer type: octet
4	8.585697878	10.3.3.1	10.1.1.1	TFTP	59 Data Packet, Block: 1 (last)

Figura 1.9: Tráfego de pacotes

```

▶ Frame 3: 56 bytes on wire (448 bits), 56 bytes captured (448 bits) on interface 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: 10.3.3.1
    0100 .... = Version: 4
    .... 0101 = Header Length: 20 bytes (5)
    ▶ Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
      Total Length: 42
      Identification: 0x097e (2430)
    ▶ Flags: 0x4000, Don't fragment
      Time to live: 61
      Protocol: UDP (17)
      Header checksum: 0x1c40 [validation disabled]
      [Header checksum status: Unverified]
      Source: 10.1.1.1
      Destination: 10.3.3.1
▼ User Datagram Protocol, Src Port: 33991, Dst Port: 69
    Source Port: 33991
    Destination Port: 69
    Length: 22
    Checksum: 0x182d [unverified]
    [Checksum Status: Unverified]
    [Stream index: 0]
▶ Trivial File Transfer Protocol

```

Figura 1.10: Trama tftp

Este comando utiliza o protocolo de aplicação TFTP. Podemos verificar que o protocolo de transporte é o UDP (olhando para o campo Protocol), a porta de atendimento é a 69 (olhando para o campo DST Port) e o Overhead de transporte = Header Length / Total Length = 20 / 42 = 0.4762.

BROWSER/HTTP

5	6.569314795	10.1.1.1	10.3.3.1	TCP	66 55846 → 80 [ACK] Seq=1 Ack=1 Win=29312 Len=0 TSval=3727693968 TSecr=1502470631
6	6.57089466	10.1.1.1	10.3.3.1	HTTP	206 GET /file1 HTTP/1.1
7	6.570693552	10.3.3.1	10.1.1.1	TCP	66 80 → 55846 [ACK] Seq=1 Ack=141 Win=30080 Len=0 TSval=1502470633 TSecr=3727693970

Figura 1.11: Tráfego de pacotes

```

▶ Frame 6: 206 bytes on wire (1648 bits), 206 bytes captured (1648 bits) on interface 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: 10.3.3.1
    0100 .... = Version: 4
    .... 0101 = Header Length: 20 bytes (5)
    ▶ Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
    Total Length: 192
    Identification: 0xee34 (60980)
    ▶ Flags: 0x4000, Don't fragment
    Time to live: 61
    Protocol: TCP (6)
    Header checksum: 0x36fe [validation disabled]
    [Header checksum status: Unverified]
    Source: 10.1.1.1
    Destination: 10.3.3.1
▼ Transmission Control Protocol, Src Port: 55846, Dst Port: 80, Seq: 1, Ack: 1, Len: 140
    Source Port: 55846
    Destination Port: 80
    [Stream index: 0]
    [TCP Segment Len: 140]
    Sequence number: 1 (relative sequence number)
    [Next sequence number: 141 (relative sequence number)]
    Acknowledgment number: 1 (relative ack number)
    1000 .... = Header Length: 32 bytes (8)
    ▶ Flags: 0x018 (PSH, ACK)
    Window size value: 229
    [Calculated window size: 29312]
    [Window size scaling factor: 128]
    Checksum: 0x18b8 [unverified]
    [Checksum Status: Unverified]
    Urgent pointer: 0
    ▶ Options: (12 bytes), No-Operation (NOP), No-Operation (NOP), Timestamps
    ▶ [SEQ/ACK analysis]
    ▶ [Timestamps]
    TCP payload (140 bytes)
▶ Hypertext Transfer Protocol

```

Figura 1.12: Trama browser/http

Este comando utiliza o protocolo de aplicação HTTP. Podemos verificar que o protocolo de transporte é o TCP (olhando para o campo Protocol), a porta de atendimento é a 80 (olhando para o campo DST Port) e o Overhead de transporte = Header Length / Total Length = 20 / 192 = 0.1042.

NSLOOKUP

1 0.000000000	10.0.2.15	10.0.2.3	DNS	84 Standard query 0xd29b AAAA www.uminho.pt OPT
2 0.000736641	10.0.2.3	10.0.2.15	DNS	84 Standard query response 0xd29b AAAA www.uminho.pt OPT

Figura 1.13: Tráfego de pacotes


```

▶ Frame 1: 84 bytes on wire (672 bits), 84 bytes captured (672 bits) on interface 0
▶ Ethernet II, Src: PcsCompu_04:03:91 (08:00:27:04:03:91), Dst: RealtekU_12:35:03 (52:54:00:12:35:03)
▼ Internet Protocol Version 4, Src: 10.0.2.15, Dst: 10.0.2.3
    0100 .... = Version: 4
    .... 0101 = Header Length: 20 bytes (5)
    ▶ Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
    Total Length: 70
    Identification: 0x37a1 (14241)
    ▶ Flags: 0x4000, Don't fragment
    Time to live: 64
    Protocol: UDP (17)
    Header checksum: 0xeaf4 [validation disabled]
    [Header checksum status: Unverified]
    Source: 10.0.2.15
    Destination: 10.0.2.3
▼ User Datagram Protocol, Src Port: 41552, Dst Port: 53
    Source Port: 41552
    Destination Port: 53
    Length: 50
    Checksum: 0x1855 [unverified]
    [Checksum Status: Unverified]
    [Stream index: 0]
▶ Domain Name System (query)

```

Figura 1.14: Trama nslookup

Este comando utiliza o protocolo de aplicação DNS. Podemos verificar que o protocolo de transporte é o UDP (olhando para o campo Protocol), a porta de atendimento é a 53 (olhando para o campo DST Port) e o Overhead de transporte = Header Length / Total Length = 20 / 70 = 0.2857.

SSH

5	5.084538120	10.1.1.1	10.3.3.1	TCP	66 33600 → 22 [ACK] Seq=1 Ack=1 Win=29312 Len=0 TSval=3726452339 TSecr=1501229001
6	5.087567948	10.1.1.1	10.3.3.1	SSHv2	107 Client: Protocol (SSH-2.0-OpenSSH_7.6p1 Ubuntu-4ubuntu0.3)
7	5.088755513	10.3.3.1	10.1.1.1	TCP	66 22 → 33600 [ACK] Seq=1 Ack=42 Win=29056 Len=0 TSval=1501229006 TSecr=3726452340

Figura 1.15: Tráfego de pacotes

```

▶ Frame 6: 107 bytes on wire (856 bits), 107 bytes captured (856 bits) on interface 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: 10.3.3.1
    0100 .... = Version: 4
    .... 0101 = Header Length: 20 bytes (5)
    ▶ Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
    Total Length: 93
    Identification: 0x3d2d (15661)
    ▶ Flags: 0x4000, Don't fragment
    Time to live: 61
    Protocol: TCP (6)
    Header checksum: 0xe868 [validation disabled]
    [Header checksum status: Unverified]
    Source: 10.1.1.1
    Destination: 10.3.3.1
▼ Transmission Control Protocol, Src Port: 33600, Dst Port: 22, Seq: 1, Ack: 1, Len: 41
    Source Port: 33600
    Destination Port: 22
    [Stream index: 0]
    [TCP Segment Len: 41]
    Sequence number: 1 (relative sequence number)
    [Next sequence number: 42 (relative sequence number)]
    Acknowledgment number: 1 (relative ack number)
    1000 .... = Header Length: 32 bytes (8)
    ▶ Flags: 0x018 (PSH, ACK)
    Window size value: 229
    [Calculated window size: 29312]
    [Window size scaling factor: 128]
    Checksum: 0x1855 [unverified]
    [Checksum Status: Unverified]
    Urgent pointer: 0
    ▶ Options: (12 bytes), No-Operation (NOP), No-Operation (NOP), Timestamps
    ▶ [SEQ/ACK analysis]
    ▶ [Timestamps]
    TCP payload (41 bytes)
▼ SSH Protocol
    Protocol: SSH-2.0-OpenSSH_7.6p1 Ubuntu-4ubuntu0.3

```

Figura 1.16: Trama ssh

Este comando utiliza o protocolo de aplicação SSHv2. Podemos verificar que o protocolo de transporte é o TCP (olhando para o campo Protocol), a porta de atendimento é a 22 (olhando para o campo DST Port) e o Overhead de transporte = Header Length / Total Length = 20 / 93 = 21.51.

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.

FTP

28	16.872149585	10.3.3.1	10.1.1.1	FTP	117 Response: 200 PORT command successful. Consider using PASV.
29	16.872659720	10.1.1.1	10.3.3.1	FTP	78 Request: REIR file1
30	16.874548364	10.3.3.1	10.1.1.1	TCP	74 20 -> 46143 [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TSval=1502044293 TSecr=0 WS=128
31	16.874869718	10.1.1.1	10.3.3.1	TCP	74 46143 -> 20 [SYN, ACK] Seq=0 Ack=1 Win=28960 Len=0 MSS=1460 SACK_PERM=1 TSval=3727267630 TSecr=1502044293 WS=128
32	16.875113744	10.3.3.1	10.1.1.1	TCP	66 20 -> 46143 [ACK] Seq=1 Ack=1 Win=29312 Len=0 TSval=1502044294 TSecr=3727267630
33	16.875370619	10.3.3.1	10.1.1.1	FTP	129 Response: 150 Opening BINARY mode data connection for file1 (13 bytes).
34	16.875528786	10.3.3.1	10.1.1.1	FTP-DA	79 FTP Data: 13 bytes (PORT) (RETR file1)
35	16.876253376	10.3.3.1	10.1.1.1	TCP	66 20 -> 46143 [FIN, ACK] Seq=14 Ack=1 Win=29312 Len=0 TSval=1502044294 TSecr=3727267630
36	16.876330237	10.1.1.1	10.3.3.1	TCP	66 46143 -> 20 [ACK] Seq=1 Ack=14 Win=29056 Len=0 TSval=3727267631 TSecr=1502044294
37	16.877375486	10.1.1.1	10.3.3.1	TCP	66 46143 -> 20 [FIN, ACK] Seq=1 Ack=14 Win=29056 Len=0 TSval=3727267633 TSecr=1502044294
38	16.877784567	10.3.3.1	10.1.1.1	TCP	66 20 -> 46143 [ACK] Seq=15 Ack=2 Win=29312 Len=0 TSval=1502044296 TSecr=3727267633
39	16.878975671	10.3.3.1	10.1.1.1	FTP	90 Response: 226 Transfer complete.

Figura 1.17: Tráfego de pacotes

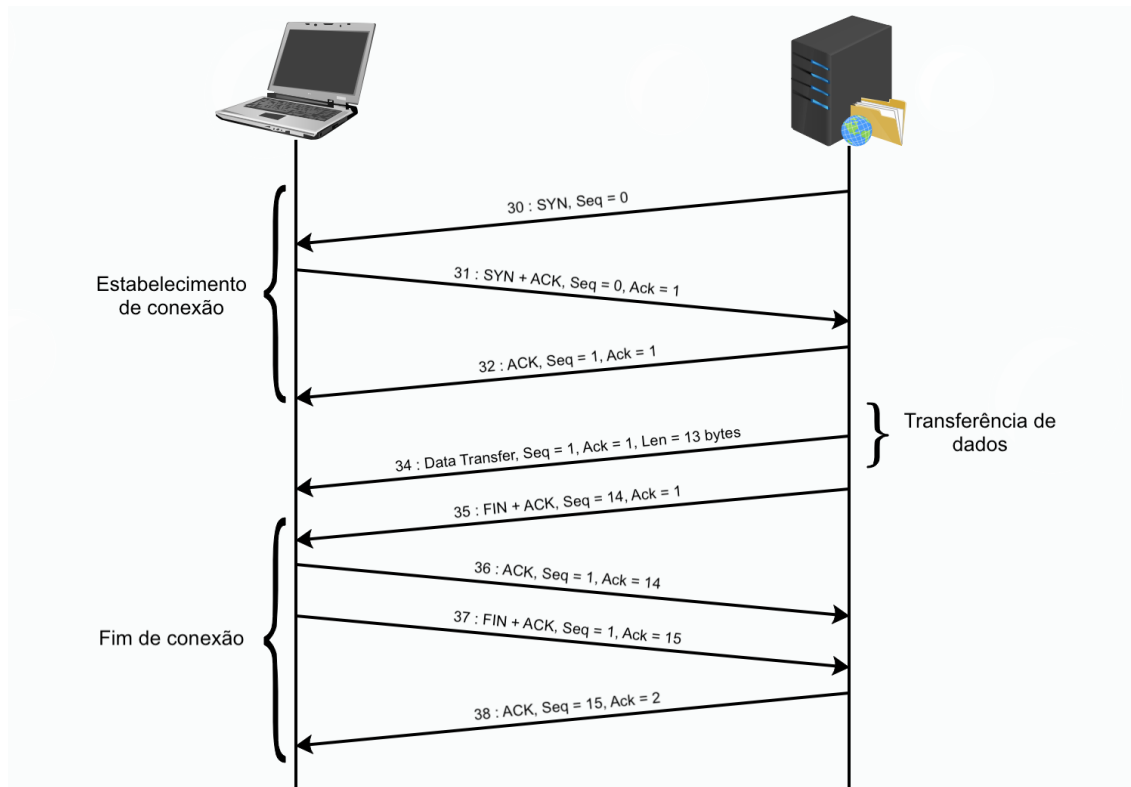


Figura 1.18: Protocolo de controlo de transmissão FTP

Na fase de estabelecimento de conexão, a conexão começa com o envio de um pacote SYN (indica que os números de sequência devem ser sincronizados para se iniciar uma conexão) por parte do servidor com número de sequência igual a 0, ao qual o utilizador responde com um SYN+ACK. Posteriormente, o servidor envia um ACK (indica se o número de sequência de confirmação é válido) de volta ao utilizador.

De seguida, ocorre a fase de transferência de Dados, onde o servidor envia os dados solicitados ao utilizador.

No final do envio, o servidor inicia a fase de fim de conexão (dupla terminação com 4 segmentos). Note-se que, o servidor após emitir o pacote FIN (indica que terminou o envio de dados), fica à espera de um pacote ACK, para poder terminar a conexão com sucesso.

TFTP

2	1.525152399	fe80::200:ff:feaa:10	ff02::5	OSPF	90 Hello Packet
3	8.585028189	10.1.1.1	10.3.3.1	TFTP	56 Read Request, File: file1, Transfer type: octet
4	8.585697878	10.3.3.1	10.1.1.1	TFTP	59 Data Packet, Block: 1 (last)
5	8.586427619	10.1.1.1	10.3.3.1	TFTP	46 Acknowledgement, Block: 1
6	10.001038434	10.3.3.254	224.0.0.5	OSPF	78 Hello Packet

Figura 1.19: Tráfego de pacotes

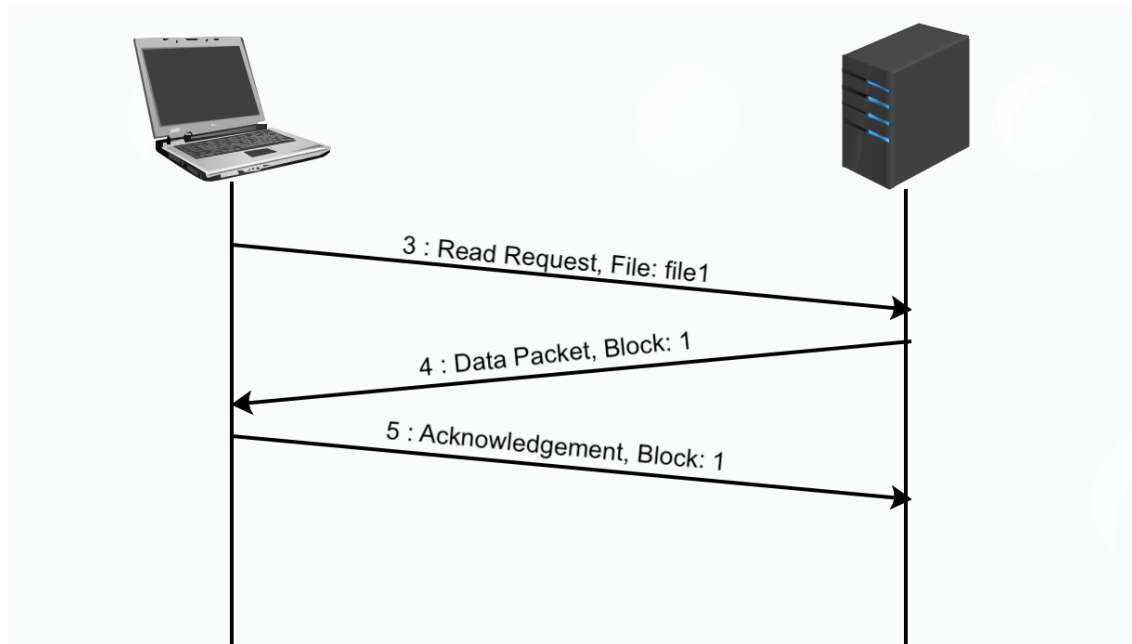


Figura 1.20: Protocolo de controlo de transmissão TFTP

O utilizador envia um Read Request para o servidor, o qual contém o nome do ficheiro, ao qual o servidor responde com um pacote de dados. O utilizador após receber os dados envia um pacote ACK.

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;

	SFTP	FTP	TFTP	HTTP
Uso da camada de transporte	Protocolo TCP	Protocolo TCP	Protocolo UDP	Protocolo TCP
Eficiência na transferência	A eficiência depende da fiabilidade da transferência de dados	Eficiente, visto que é fiável(maior overhead)	Pouco eficiente, pois não se responsabiliza pela entrega dos dados(menor overhead)	Muito eficiente
Complexidade	Muito complexo, pois disponibiliza inúmeras funcionalidades	Complexo, pois existe segurança na transferência de dados	Versão simplificada do FTP, uma vez que o protocolo de transporte é o UDP, disponível menos funcionalidades do que o FTP	Pouco complexo
Segurança	Seguro, pois recorre à autenticação e existe encriptação dos dados	Pouco Seguro, apesar de utilizar autenticação	Pouco seguro, uma vez que não existe autenticação nem encriptação de dados	Pouco seguro, apesar de recorrer à autenticação, não utiliza encriptação dos dados

Figura 1.21: Distinção das aplicações SFTP, FTP, TFTP e HTTP

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).

```
root@Portatil1: /tmp/pycore.32831/Portatil1.conf
root@Portatil1:/tmp/pycore.32831/Portatil1.conf# ping 10.2.2.3
PING 10.2.2.3 (10.2.2.3) 56(84) bytes of data.
64 bytes from 10.2.2.3: icmp_seq=1 ttl=62 time=6.09 ms
64 bytes from 10.2.2.3: icmp_seq=3 ttl=62 time=6.24 ms
64 bytes from 10.2.2.3: icmp_seq=4 ttl=62 time=7.51 ms
64 bytes from 10.2.2.3: icmp_seq=5 ttl=62 time=18.0 ms
64 bytes from 10.2.2.3: icmp_seq=6 ttl=62 time=5.61 ms
64 bytes from 10.2.2.3: icmp_seq=6 ttl=62 time=5.61 ms (DUP!)
64 bytes from 10.2.2.3: icmp_seq=7 ttl=62 time=13.5 ms
64 bytes from 10.2.2.3: icmp_seq=8 ttl=62 time=5.82 ms
64 bytes from 10.2.2.3: icmp_seq=9 ttl=62 time=11.5 ms
64 bytes from 10.2.2.3: icmp_seq=11 ttl=62 time=15.4 ms
64 bytes from 10.2.2.3: icmp_seq=12 ttl=62 time=7.18 ms
64 bytes from 10.2.2.3: icmp_seq=13 ttl=62 time=6.37 ms
64 bytes from 10.2.2.3: icmp_seq=14 ttl=62 time=7.68 ms
64 bytes from 10.2.2.3: icmp_seq=15 ttl=62 time=19.3 ms
64 bytes from 10.2.2.3: icmp_seq=16 ttl=62 time=6.37 ms
^C
--- 10.2.2.3 ping statistics ---
17 packets transmitted, 14 received, +1 duplicates, 17% packet loss, time 16064ms
rtt min/avg/max/mdev = 5.613/9.503/19.397/4.655 ms
root@Portatil1:/tmp/pycore.32831/Portatil1.conf#
```

Figura 1.22: Comando ping

Observando a figura 23, na execução do comando ping ao servidor 1 no portátil 1 vemos que, dos 17 pacotes transmitidos, houve uma perda de 17% dos pacotes e 1 foi duplicado.

Os protocolos de transporte TCP e UDP resolvem os problemas no nível da ligação de rede de maneiras diferentes. O protocolo TCP é o mais usado dos dois, pois garante a entrega de todos os pacotes, e caso haja perda de um pacote, este é retransmitido. Esta garantia é obtida através do método Three Way Handshake (SYN, SYN-ACK, ACK).

Quanto ao protocolo UDP, é mais simples que o anterior, sendo que não é capaz de detetar e recuperar perdas de pacotes. A garantia da entrega dos pacotes fica a cargo das camadas superiores como a aplicação em si. Relativamente ao portátil 1 da figura, a recuperação dos pacotes perdidos pode ser feita através do reenvio destes no caso do TCP.

Capítulo 2

Conclusão

Com a realização deste primeiro trabalho prático aprofundamos o nosso conhecimento sobre as camadas inferiores na utilização de diversas aplicações. Começamos por analisar protocolos de aplicação e de transporte, as portas que estes utilizam e os overheads associados através da análise de tráfego de pacotes e tramas. De seguida comparamos quatro serviços distintos (SFTP, FTP, TFTP e HTTP) na transferência de ficheiros. Por fim analisamos as diferenças entre os protocolos TCP e UDP relativamente à entrega de pacotes. A realização deste trabalho foi de extrema importância para a melhor compreensão da camada de transporte.