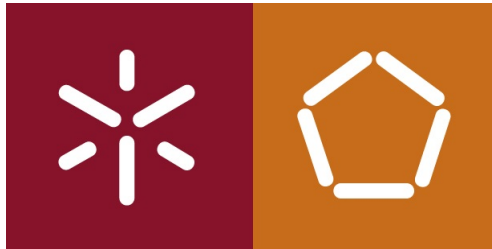


UNIVERSIDADE DO MINHO

MESTRADO INTEGRADO EM ENGENHARIA INFORMÁTICA



---

## TP1 - Protocolos da Camada de Transporte

---

COMUNICAÇÕES POR COMPUTADOR

PL2 GRUPO 4



Adriana Meireles  
A82582



Helena Martins  
A82500



Mariana Pereira  
A81146

4 de Março de 2020

# Conteúdo

0.1	Questões e Respostas . . . . .	2
0.1.1	Questão 1: . . . . .	2
0.1.2	Questão 2: . . . . .	6
0.1.3	Questão 3: . . . . .	9
0.1.4	Questão 4: . . . . .	9
0.2	Conclusões . . . . .	10

## 0.1 Questões e Respostas

### 0.1.1 Questão 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	Protocolo de Aplicações	Protocolo de Transporte	Porta de Atendimento	Overhead de Transporte em bytes
Ping	PING	-	-	-
Traceroute	TRACEROUTE	UDP	33443	$(8/40)*100$
Telnet	TELNET	TCP	23	$(20/47)*100$
FTP	FTP	TCP	21	$(20/26)*100$
TFTP	TFPT	UDP	69	$(8/52)*100$
Browser/http	HTTP	TCP	80	$(20/331)*100$
nslookup	DNS	UDP	53	$(8/39)*100$
SSH	SSH	TCP	22	$(20/332)*100$

- Ping:

4718	3084.59362	64.227.1.239	10.0.2.15	ICMP	98 Echo (ping) reply	id=0x0d66, seq=4/1024, ttl=64
4723	3084.59362	10.0.2.15	64.227.1.239	ICMP	98 Echo (ping) request	id=0x0d66, seq=4/1024, ttl=64
4724	3084.73580	64.227.1.239	10.0.2.15	ICMP	98 Echo (ping) reply	id=0x0d66, seq=4/1024, ttl=41

▶ Frame 4723: 98 bytes on wire (784 bits), 98 bytes captured (784 bits)

▶ Ethernet II, Src: CadmusCo\_78:e5:64 (08:00:27:78:e5:64), Dst: RealtekU\_12:35:02 (52:54:00:12:35:02)

▼ Internet Protocol Version 4, Src: 10.0.2.15 (10.0.2.15), Dst: 64.227.1.239 (64.227.1.239)

Version: 4  
Header length: 20 bytes

▶ Differentiated Services Field: 0x00 (DSCP 0x00: Default; ECN: 0x00: Not-ECT (Not ECN-Capable Transport))

Total Length: 84  
Identification: 0x7a6f (31343)

▶ Flags: 0x02 (Don't Fragment)  
Fragment offset: 0  
Time to live: 64  
Protocol: ICMP (1)

▶ Header checksum: 0x7159 [correct]  
Source: 10.0.2.15 (10.0.2.15)  
Destination: 64.227.1.239 (64.227.1.239)

▶ Internet Control Message Protocol

Figura 1: Captura de Tráfego Ping

O protocolo que foi utilizado é o PING. Como o ping trabalha diretamente com a camada de rede, o protocolo de transporte não é aplicável pelo que não existe porta de atendimento nem overhead de transporte.

## • Traceroute:

No.	Time	Source	Destination	Protocol	Length	Info
4227	1725.55890	10.0.2.15	193.136.19.254	UDP	74	Source port: 49108 Destination port: 33440
4228	1725.55892	10.0.2.15	193.136.19.254	UDP	74	Source port: 43341 Destination port: 33441
4229	1725.55893	10.0.2.15	193.136.19.254	UDP	74	Source port: 60876 Destination port: 33442
4230	1725.55901	10.0.2.2	10.0.2.15	ICMP	70	Time-to-live exceeded (Time to live exceeded in transit)
4231	1725.55902	10.0.2.2	10.0.2.15	ICMP	70	Time-to-live exceeded (Time to live exceeded in transit)
4232	1725.55902	10.0.2.2	10.0.2.15	ICMP	70	Time-to-live exceeded (Time to live exceeded in transit)
4233	1725.55906	10.0.2.15	193.136.19.254	UDP	74	Source port: 48146 Destination port: 33443
4234	1725.55908	10.0.2.15	193.136.19.254	UDP	74	Source port: 58969 Destination port: 33444

▶ Frame 4233: 74 bytes on wire (592 bits), 74 bytes captured (592 bits)
▶ Ethernet II, Src: CadmusCo_78:e5:64 (08:00:27:78:e5:64), Dst: RealtekU_12:35:02 (52:54:00:12:35:02)
▼ Internet Protocol Version 4, Src: 10.0.2.15 (10.0.2.15), Dst: 193.136.19.254 (193.136.19.254)
Version: 4
Header length: 20 bytes
▶ Differentiated Services Field: 0x00 (DSCP 0x00: Default; ECN: 0x00: Not-ECT (Not ECN-Capable Transport))
Total Length: 60
Identification: 0x1050 (4176)
▶ Flags: 0x00
Fragment offset: 0
▶ Time to live: 4
Protocol: UDP (17)
▶ Header checksum: 0xc4cc [correct]
Source: 10.0.2.15 (10.0.2.15)
Destination: 193.136.19.254 (193.136.19.254)
▼ User Datagram Protocol, Src Port: 48146 (48146), Dst Port: 33443 (33443)
Source port: 48146 (48146)
▶ Destination port: 33443 (33443)
Length: 40
▶ Checksum: 0xe1ce [validation disabled]
▶ Data (32 bytes)

Figura 2: Captura de Tráfego Traceroute

O protocolo de aplicação utilizado foi o TRACEROUTE. Com base na figura 2 podemos observar que o protocolo de transporte é o UDP. A porta de atendimento para a trama é a 33443. Uma vez que o protocolo de transporte é o UDP, possui um cabeçalho de tamanho fixo de 8 bytes, logo o overhead é  $(8/40) * 100 = 20\%$

## • Telnet:

No.	Time	Source	Destination	Protocol	Length	Info
426	246.726155	10.0.2.15	193.137.16.65	DNS	75	Standard query A cc2020.ddns.net
427	246.783842	193.137.16.65	10.0.2.15	DNS	172	Standard query response A 193.136.9.183
428	246.784190	10.0.2.15	193.136.9.183	TCP	74	53709 > telnet [SYN] Seq=0 Win=14600 Len=0 MSS=1460 SACK_PERM=1 TSval=166839 TSecr=0 WS=16
429	246.835325	193.136.9.183	10.0.2.15	TCP	60	telnet > 53709 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1460
430	246.835378	10.0.2.15	193.136.9.183	TCP	54	53709 > telnet [ACK] Seq=1 Ack=1 Win=14600 Len=0
431	246.835915	10.0.2.15	193.136.9.183	TELNET	81	Telnet Data ...
432	246.836460	193.136.9.183	10.0.2.15	TCP	60	telnet > 53709 [ACK] Seq=1 Ack=28 Win=65535 Len=0
433	261.872271	193.136.9.183	10.0.2.15	TELNET	66	Telnet Data ...
434	261.872271	10.0.2.15	193.136.9.183	TCP	54	53709 > telnet [ACK] Seq=28 Ack=13 Win=14600 Len=0

Version: 4
Header length: 20 bytes
▶ Differentiated Services Field: 0x10 (DSCP 0x04: Unknown DSCP; ECN: 0x00: Not-ECT (Not ECN-Capable Transport))
Total Length: 67
Identification: 0x176b (5995)
▶ Flags: 0x02 (Don't Fragment)
Fragment offset: 0
Time to live: 64
Protocol: TCP (6)
▶ Header checksum: 0x4bec [correct]
Source: 10.0.2.15 (10.0.2.15)
Destination: 193.136.9.183 (193.136.9.183)
▼ Transmission Control Protocol, Src Port: 53709 (53709), Dst Port: telnet (23), Seq: 1, Ack: 1, Len: 27
Source port: 53709 (53709)
Destination port: telnet (23)
[Stream index: 27]
Sequence number: 1 (relative sequence number)
[Next sequence number: 28 (relative sequence number)]
Acknowledgement number: 1 (relative ack number)
Header length: 20 bytes

Figura 3: Captura de Tráfego Telnet

O protocolo de aplicação utilizado é o TELNET. O protocolo de transporte é o TCP e a porta de atendimento da trama é a 23. O overhead é  $(20/(67 - 20)) \times 100 = 42.5\%$

### • FTP:

No.	Time	Source	Destination	Protocol	Length	Info
4540	2493.10786	193.136.9.183	10.0.2.15	FTP	74	Response: 220 (vsFTPD 2.3.5)
4542	2496.87141	10.0.2.15	193.136.9.183	FTP	63	Request: USER cc
4544	2496.87744	193.136.9.183	10.0.2.15	FTP	88	Response: 331 Please specify the password.
4546	2500.22522	10.0.2.15	193.136.9.183	FTP	67	Request: PASS cc2020
4548	2500.30411	193.136.9.183	10.0.2.15	FTP	77	Response: 230 Login successful.
4550	2500.30437	10.0.2.15	193.136.9.183	FTP	60	Request: SYST
4552	2500.30872	193.136.9.183	10.0.2.15	FTP	73	Response: 215 UNIX Type: L8

```

▶ Frame 4550: 60 bytes on wire (480 bits), 60 bytes captured (480 bits)
▶ Ethernet II, Src: CadmusCo_78:e5:64 (08:00:27:78:e5:64), Dst: RealtekU_12:35:02 (52:54:00:12:35:02)
▼ Internet Protocol Version 4, Src: 10.0.2.15 (10.0.2.15), Dst: 193.136.9.183 (193.136.9.183)
  Version: 4
  Header length: 20 bytes
  ▶ Differentiated Services Field: 0x10 (DSCP 0x04: Unknown DSCP; ECN: 0x00: Not-ECT (Not ECN-Capable Transport))
  Total Length: 46
  Identification: 0x1e3e (7742)
  ▶ Flags: 0x02 (Don't Fragment)
  Fragment offset: 0
  Time to live: 64
  Protocol: TCP (6)
  ▶ Header checksum: 0x452e [correct]
  Source: 10.0.2.15 (10.0.2.15)
  Destination: 193.136.9.183 (193.136.9.183)
▼ Transmission Control Protocol, Src Port: 49167 (49167), Dst Port: ftp (21), Seq: 23, Ack: 78, Len: 6
  Source port: 49167 (49167)
  Destination port: ftp (21)
  [Stream index: 76]
  Sequence number: 23 (relative sequence number)
  [Next sequence number: 29 (relative sequence number)]
  Acknowledgement number: 78 (relative ack number)
  Header length: 20 bytes
  ▶ Flags: 0x018 (PSH, ACK)

```

Figura 4: Captura de Tráfego FTP

Na figura 4 verificamos que o protocolo de aplicação utilizado é o FTP. O protocolo de transporte é o TCP e a porta de atendimento da trama é a 21. O overhead é  $(20/(46 - 20)) \times 100 = 76.9\%$

### • TFTP:

No.	Time	Source	Destination	Protocol	Length	Info
4528	2390.48805	10.0.2.15	193.136.9.183	TFTP	86	Read Request, File: file1, Transfer type: octet, tsize\000=0\000, blksize\000=512\000, timeout\000=6\000

```

▶ Frame 4528: 86 bytes on wire (688 bits), 86 bytes captured (688 bits)
▶ Ethernet II, Src: CadmusCo_78:e5:64 (08:00:27:78:e5:64), Dst: RealtekU_12:35:02 (52:54:00:12:35:02)
▼ Internet Protocol Version 4, Src: 10.0.2.15 (10.0.2.15), Dst: 193.136.9.183 (193.136.9.183)
  Version: 4
  Header length: 20 bytes
  ▶ Differentiated Services Field: 0x00 (DSCP 0x00: Default; ECN: 0x00: Not-ECT (Not ECN-Capable Transport))
  Total Length: 72
  Identification: 0xc2ff (49919)
  ▶ Flags: 0x02 (Don't Fragment)
  Fragment offset: 0
  Time to live: 64
  Protocol: UDP (17)
  ▶ Header checksum: 0xa057 [correct]
  Source: 10.0.2.15 (10.0.2.15)
  Destination: 193.136.9.183 (193.136.9.183)
▼ User Datagram Protocol, Src Port: 46546 (46546), Dst Port: tftp (69)
  Source port: 46546 (46546)
  Destination port: tftp (69)
  Length: 52
  ▶ Checksum: 0xd793 [validation disabled]
▶ Trivial File Transfer Protocol

```

Figura 5: Captura de Tráfego TFTP

Como podemos visualizar na figura 5, o protocolo de aplicação é o TFTP. O protocolo de transporte é o UDP que tem cabeçalho de tamanho fixo de 8

bytes como foi dito anteriormente. A porta de atendimento da trama é a 69. O overhead é  $(8/52) * 100 = 15.4$

- **Browser/HTTP:**

No.	Time	Source	Destination	Protocol	Length	Info
26	15.069137	10.0.2.15	193.136.9.240	HTTP	365	GET /disciplinas/CC-MIEI/ HTTP/1.1
40	15.088506	193.136.9.240	10.0.2.15	HTTP	1134	HTTP/1.1 200 OK (text/html)
43	15.101775	10.0.2.15	193.136.9.240	HTTP	409	GET /disciplinas/CC-MIEI/created-with-vim.png HTTP/1.1
50	15.121060	193.136.9.240	10.0.2.15	HTTP	1162	HTTP/1.1 200 OK (PNG)

▶ Frame 26: 365 bytes on wire (2920 bits), 365 bytes captured (2920 bits)
▶ Ethernet II, Src: CadmusCo_78:e5:64 (08:00:27:78:e5:64), Dst: RealtekU_12:35:02 (52:54:00:12:35:02)
▼ Internet Protocol Version 4, Src: 10.0.2.15 (10.0.2.15), Dst: 193.136.9.240 (193.136.9.240)
Version: 4
Header length: 20 bytes
▶ Differentiated Services Field: 0x00 (DSCP 0x00: Default; ECN: 0x00: Not-ECT (Not ECN-Capable Transport))
Total Length: 351
Identification: 0x1483 (5251)
▶ Flags: 0x02 (Don't Fragment)
Fragment offset: 0
Time to live: 64
Protocol: TCP (6)
▶ Header checksum: 0x4d8f [correct]
Source: 10.0.2.15 (10.0.2.15)
Destination: 193.136.9.240 (193.136.9.240)
▼ Transmission Control Protocol, Src Port: 32890 (32890), Dst Port: http (80), Seq: 1, Ack: 1, Len: 311
Source port: 32890 (32890)
Destination port: http (80)
[Stream index: 6]
Sequence number: 1 (relative sequence number)
[Next sequence number: 312 (relative sequence number)]
Acknowledgement number: 1 (relative ack number)
Header length: 20 bytes
▶ Flags: 0x018 (PSH, ACK)
Window size value: 14600
[Calculated window size: 14600]
[Window size scaling factor: -2 (no window scaling used)]

Figura 6: Captura de Tráfego HTTP

Como podemos observar na figura 6, o protocolo de aplicação é o HTTP. O protocolo de transporte utilizado é o TCP e a porta de atendimento é a 80. De acordo com os dados da figura, o overhead é  $(20/(351 - 20)) * 100 = 6\%$

- **nslookup:**

4262	15.220749	10.0.2.15	193.137.16.65	DNS	61	Standard query R 193.137.16.65
4263	2102.19473	10.0.2.15	193.137.16.65	DNS	73	Standard query A www.uminho.pt
4264	2102.20090	193.137.16.65	10.0.2.15	DNS	345	Standard query response A 193.137.9.114

▶ Frame 4263: 73 bytes on wire (584 bits), 73 bytes captured (584 bits)
▶ Ethernet II, Src: CadmusCo_78:e5:64 (08:00:27:78:e5:64), Dst: RealtekU_12:35:02 (52:54:00:12:35:02)
▼ Internet Protocol Version 4, Src: 10.0.2.15 (10.0.2.15), Dst: 193.137.16.65 (193.137.16.65)
Version: 4
Header length: 20 bytes
▶ Differentiated Services Field: 0x00 (DSCP 0x00: Default; ECN: 0x00: Not-ECT (Not ECN-Capable Transport))
Total Length: 59
Identification: 0x6f3f (28479)
▶ Flags: 0x00
Fragment offset: 0
Time to live: 64
Protocol: UDP (17)
▶ Header checksum: 0x2d9a [correct]
Source: 10.0.2.15 (10.0.2.15)
Destination: 193.137.16.65 (193.137.16.65)
▼ User Datagram Protocol, Src Port: 52974 (52974), Dst Port: domain (53)
Source port: 52974 (52974)
Destination port: domain (53)
Length: 39
▶ Checksum: 0xde11 [validation disabled]
▶ Domain Name System (query)

Figura 7: Captura de Tráfego DNS

Como podemos observar na figura 7, o protocolo de aplicação utilizado é o DNS. O protocolo de transporte utilizado é o UDP e a porta de atendimento da trama é a 53. O overhead é  $(8/39) * 100 = 20.5\%$

## • SSH:

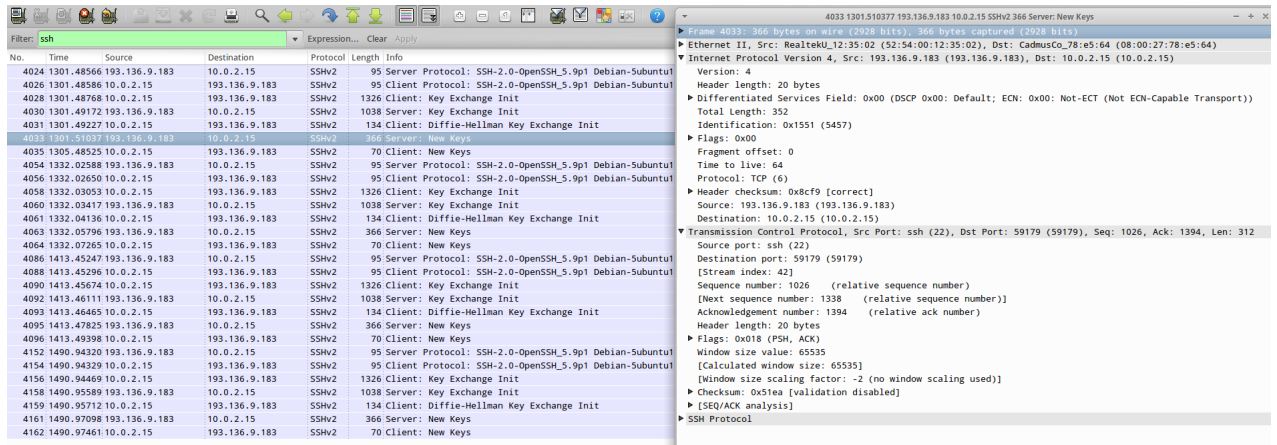


Figura 8: Captura de Tráfego SSH

Como podemos verificar na figura 8 o protocolo de aplicação utilizado é o SSH. O protocolo de transporte é o TCP e a porta de atendimento da trama é a 22. O overhead é  $(20/(352 - 20)) * 100 = 6\%$

### 0.1.2 Questão 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. (Nota: a transferência por FTP envolve mais que uma conexão FTP, nomeadamente uma de controlo [ftp] e outra de dados [ftp-data]). Faça o diagrama apenas para a conexão de transferência de dados do ficheiro mais pequeno)

564	1526.09674	10.1.1.1	10.0.3.1	FTP	76 Request: REIR file1
567	1526.09708	10.3.3.1	10.1.1.1	TCP	74 ftp-data > 52360 [SYN] Seq=0 Win=14600 Len=0 MSS=1460 SACK_PERM=1 TSval=1278153 TSecr=0 WS=16
568	1526.09726	10.1.1.1	10.3.3.1	TCP	74 52360 > ftp-data [SYN, ACK] Seq=0 Ack=1 Win=14480 Len=0 MSS=1460 SACK_PERM=1 TSval=1278154 TSecr=1278153 WS=16
569	1526.09742	10.3.3.1	10.1.1.1	TCP	66 ftp-data > 52360 [ACK] Seq=1 Ack=1 Win=14608 Len=0 TSval=1278154 TSecr=1278154
570	1526.09748	10.3.3.1	10.1.1.1	FTP	130 Response: 150 Opening BINARY mode data connection for file1 (193 bytes).
571	1526.09748	10.3.3.1	10.1.1.1	FTP-DAT	259 FTP Data: 193 bytes
572	1526.09754	10.3.3.1	10.1.1.1	TCP	66 ftp-data > 52360 [FIN, ACK] Seq=194 Ack=1 Win=14608 Len=0 TSval=1278154 TSecr=1278154
573	1526.09784	10.1.1.1	10.3.3.1	TCP	66 52360 > ftp-data [ACK] Seq=1 Ack=194 Win=15552 Len=0 TSval=1278154 TSecr=1278154
574	1526.09857	10.1.1.1	10.3.3.1	TCP	66 52360 > ftp-data [FIN, ACK] Seq=1 Ack=195 Win=15552 Len=0 TSval=1278154 TSecr=1278154
575	1526.09893	10.3.3.1	10.1.1.1	TCP	66 ftp-data > 52360 [ACK] Seq=195 Ack=2 Win=14608 Len=0 TSval=1278154 TSecr=1278154
576	1526.09914	10.3.3.1	10.1.1.1	FTP	90 Response: 226 Transfer complete.

Figura 9: Captura de transferência via ftp

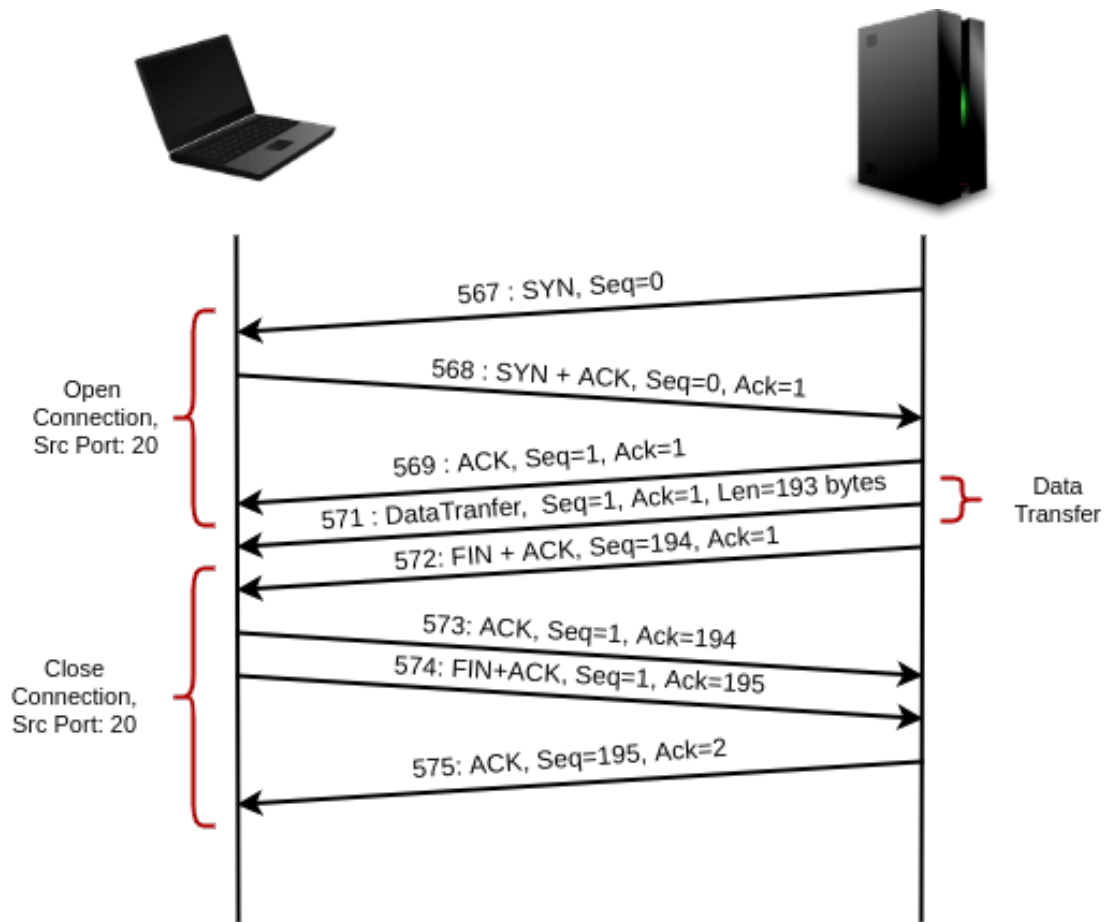


Figura 10: Diagrama Temporal ftp

Primeiramente, o servidor começa a conexão através da porta 20, enviando um pacote SYN cujo número de sequência é 0. O cliente responde com um SYN+ACK. De seguida, o servidor envia um ACK de volta ao cliente. Depois de enviar os dados pretendidos, o servidor inicia a fase de encerramento de conexão, que é constituída por quatro etapas. A flag FIN indica que se pretende terminar a conexão, ficando à espera de uma resposta ACK. Podemos verificar isso através do diagrama temporal apresentado, tendo este sido construído com base na captura de tráfego observada na Figura 9.

647 1843.98608 10.1.1.1	10.3.3.1	TFTP	50 Read Request, File: file1, Transfer type: octet
650 1843.98684 10.3.3.1	10.1.1.1	TFTP	239 Data Packet, Block: 1 (last)
651 1843.98719 10.1.1.1	10.3.3.1	TFTP	46 Acknowledgement, Block: 1

Figura 11: Captura de transferência via tftp



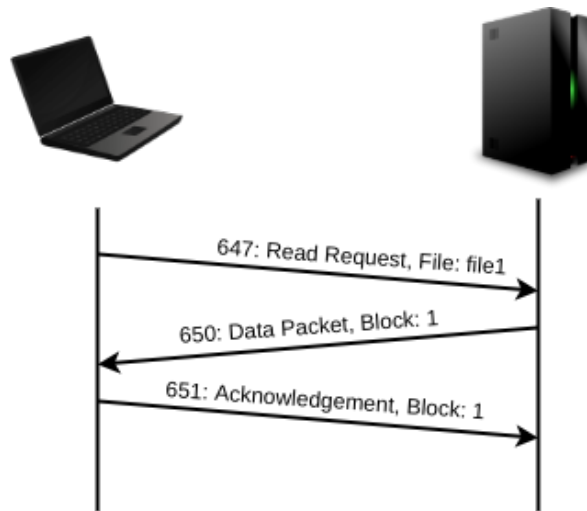


Figura 12: Diagrama temporal tftp

O cliente envia um Read Request para o servidor que irá conter diversas informações, entre as quais o nome do ficheiro. O servidor irá responder com um pacote de dados. Após a receção dos dados por parte do cliente, este irá enviar um pacote ACK.

### 0.1.3 Questão 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 é comprometida pela fiabilidade da transferência de dados.	Fiável, levando a um maior overhead.	Como não se responsabiliza por entregar os dados, apresenta um overhead menor.	Apresenta grande eficiência. Como usa pipelining é bastante rápido o que o torna uma escolha ideal.
Complexidade	É muito complexo devido ao facto de apresentar muitas funcionalidades.	Como garante segurança na transferência de dados, é complexo.	O protocolo de transporte é o UDP e as funcionalidades presentes são mais pequenas. Deste modo, é uma versão simplificada do FTP	Pouco complexo.
Segurança	Como recorre à autenticação e encriptação dos dados, é seguro.	Apesar de utilizar autenticação, é pouco seguro uma vez que a informação não é encriptada, mas sim passada como texto.	É pouco seguro, uma vez que não fornece mecanismos de autenticação ou encriptação dos dados.	É pouco seguro pois apenas recorre à autenticação, não utilizando encriptação dos dados. É vulnerável a ataques "man-in-the-middle", o que permite a atacantes o acesso a contas e informações confidenciais.

### 0.1.4 Questão 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).

Tal como podemos observar na figura que apresentamos a seguir, foi realizado um ping na LAN3 onde 10% dos pacotes foram perdidos e um pacote foi duplicado. Isto acontece devido à existência de problemas a nível de rede.

Para estes problemas de rede, os protocolos de transporte UDP e TCP têm formas diferentes de reagir. O protocolo TCP tem mecanismos de deteção e recuperação, para quando houver uma perda de pacote este seja retransmitido, garantindo que todos os pacotes são entregues, ao contrário do UDP que não dispõe desses mecanismos. Caso haja perda de pacotes, são os protocolos que correm em cima do UDP que poderão corrigir e identificar estes erros.

Posto isto, no caso do host Alfa, apresentado na figura seguinte, os pacotes podem ser recuperados pois podem ser reenviados devido ao protocolo TCP.

```
root@Alfa: /tmp/pycore.57341/Alfa.conf
64 bytes from 10.3.3.1: icmp_req=1 ttl=61 time=10.6 ms
64 bytes from 10.3.3.1: icmp_req=2 ttl=61 time=5.46 ms
64 bytes from 10.3.3.1: icmp_req=3 ttl=61 time=5.60 ms
64 bytes from 10.3.3.1: icmp_req=4 ttl=61 time=5.40 ms
64 bytes from 10.3.3.1: icmp_req=5 ttl=61 time=5.85 ms
64 bytes from 10.3.3.1: icmp_req=7 ttl=61 time=5.51 ms
64 bytes from 10.3.3.1: icmp_req=7 ttl=61 time=5.52 ms (DUP!)
64 bytes from 10.3.3.1: icmp_req=8 ttl=61 time=5.49 ms
64 bytes from 10.3.3.1: icmp_req=10 ttl=61 time=5.51 ms
64 bytes from 10.3.3.1: icmp_req=11 ttl=61 time=5.50 ms
64 bytes from 10.3.3.1: icmp_req=12 ttl=61 time=5.32 ms
64 bytes from 10.3.3.1: icmp_req=13 ttl=61 time=5.51 ms
64 bytes from 10.3.3.1: icmp_req=14 ttl=61 time=6.43 ms
64 bytes from 10.3.3.1: icmp_req=15 ttl=61 time=5.48 ms
64 bytes from 10.3.3.1: icmp_req=16 ttl=61 time=6.04 ms
64 bytes from 10.3.3.1: icmp_req=17 ttl=61 time=6.19 ms
64 bytes from 10.3.3.1: icmp_req=18 ttl=61 time=6.22 ms
64 bytes from 10.3.3.1: icmp_req=19 ttl=61 time=5.38 ms
64 bytes from 10.3.3.1: icmp_req=20 ttl=61 time=5.43 ms

--- 10.3.3.1 ping statistics ---
20 packets transmitted, 18 received, +1 duplicates, 10% packet loss, time 19045m
rtt min/avg/max/mdev = 5.325/5.924/10.644/1.157 ms
root@Alfa:/tmp/pycore.57341/Alfa.conf#
```

Figura 13: Ping

## 0.2 Conclusões

Com este trabalho estudámos o modo como as várias aplicações recorrem aos serviços da camada inferior. Para isso foram considerados os diversos protocolos de aplicação e transporte, portas de atendimento e o overhead associado ao transporte.

Começámos por realizar a instalação, configuração e utilização de serviços de transferência de ficheiros. Deste modo, foi transferido um mesmo ficheiro com recurso a quatro serviços diferentes SFTP, FTP, TFTP e HTTP. Foi realizado um diagrama temporal para o FTP e TFTP com a identificação das fases de estabelecimento de conexão, transferência de dados e fim de conexão. Posteriormente, foram analisadas as características de cada um. Por último, observámos o modo como o TCP e o UDP lidam com a perda e duplicação de pacotes.

Em suma, este trabalho permitiu-nos consolidar os conhecimentos acerca da camada de transporte.