

Trabalho Prático Nº1 – Protocolo de Camada de Transporte

Carlos Gustavo Silva Pereira a96867, Cláudio Alexandre Freitas Bessa a97063,
João Miguel Ferreira Loureiro a97257

Universidade do Minho - Licenciatura em Engenharia Informática



Índice

1	Questões	3
1.1	Pergunta 1	3
1.2	Pergunta 2	4
1.3	Pergunta 3	6
1.4	Pergunta 4	9
1.5	Pergunta 5	10
2	Conclusão	11

Resumo: O estudo dos protocolos que compõem a camada de transporte é um dos pilares da comunicação por computadores. Este documento aborda noções importantes relativas ao transporte de dados numa rede demonstrando a relevância dos protocolos TCP e UDP em diferentes contextos. Para este trabalho utilizamos auxiliamo-nos de uma máquina virtual com o sistema operativo *Xubuntu* e os *softwares CORE*, para emulação de uma rede, e o *Wireshark*, para visualização das trocas de pacotes.

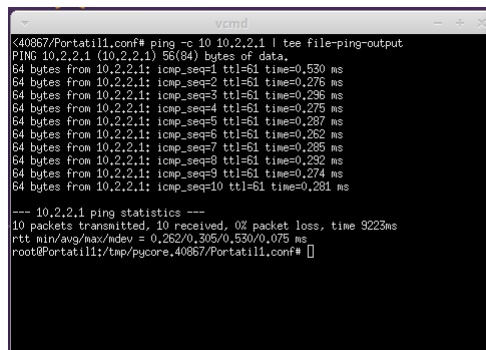
Palavras-chave: Protocolo de Camada de Transporte, Comunicação por Computadores, Segmentos, Pacotes e Datagrama

1 Questões

1.1 Pergunta 1

De que forma as perdas e duplicações de pacotes afetaram o desempenho das aplicações? Que camada lidou com esses problemas: transporte ou aplicação? Responda com base nas experiências feitas e nos resultados observados.

Resposta: Após a verificação de ambos os *pings* nos diferentes ambientes da topologia, verificamos que o computador *Grilo* foi mais lento, uma vez que essa rede tinha associada uma determinada probabilidade de perda e duplicações. A camada que lida com estes tipos de problemas é a camada da aplicação.



```
<40867/Portatil1.conf# ping -c 10 10.2.2.1 | tee file-ping-output
PING 10.2.2.1 (10.2.2.1) 56(84) bytes of data:
64 bytes from 10.2.2.1: icmp_seq=1 ttl=61 time=0.530 ms
64 bytes from 10.2.2.1: icmp_seq=2 ttl=61 time=0.276 ms
64 bytes from 10.2.2.1: icmp_seq=3 ttl=61 time=0.236 ms
64 bytes from 10.2.2.1: icmp_seq=4 ttl=61 time=0.276 ms
64 bytes from 10.2.2.1: icmp_seq=5 ttl=61 time=0.287 ms
64 bytes from 10.2.2.1: icmp_seq=6 ttl=61 time=0.262 ms
64 bytes from 10.2.2.1: icmp_seq=7 ttl=61 time=0.285 ms
64 bytes from 10.2.2.1: icmp_seq=8 ttl=61 time=0.232 ms
64 bytes from 10.2.2.1: icmp_seq=9 ttl=61 time=0.274 ms
64 bytes from 10.2.2.1: icmp_seq=10 ttl=61 time=0.281 ms

--- 10.2.2.1 ping statistics ---
10 packets transmitted, 10 received, 0% packet loss, time 922ms
rtt min/avg/max/mdev = 0.232/0.306/0.530/0.075 ms
root@Portatil1:/tmp/pycore.40867/Portatil1.conf#
```

Figura 1. Resultados obtidos em Portatil1

```

vcmd
klo.conf# ping -c 10 10.2.2.1 | tee file-ping-output
PING 10.2.2.1 (10.2.2.1) 56(84) bytes of data:
64 bytes from 10.2.2.1: icmp_seq=1 ttl=61 time=5.18 ms
64 bytes from 10.2.2.1: icmp_seq=2 ttl=61 time=5.38 ms
64 bytes from 10.2.2.1: icmp_seq=3 ttl=61 time=5.00 ms (DUP!)
64 bytes from 10.2.2.1: icmp_seq=3 ttl=61 time=5.38 ms
64 bytes from 10.2.2.1: icmp_seq=4 ttl=61 time=5.22 ms
64 bytes from 10.2.2.1: icmp_seq=4 ttl=61 time=5.22 ms (DUP!)
64 bytes from 10.2.2.1: icmp_seq=5 ttl=61 time=5.19 ms
64 bytes from 10.2.2.1: icmp_seq=6 ttl=61 time=5.02 ms
64 bytes from 10.2.2.1: icmp_seq=7 ttl=61 time=5.05 ms
64 bytes from 10.2.2.1: icmp_seq=8 ttl=61 time=5.19 ms
64 bytes from 10.2.2.1: icmp_seq=9 ttl=61 time=6.17 ms
64 bytes from 10.2.2.1: icmp_seq=10 ttl=61 time=6.00 ms

--- 10.2.2.1 ping statistics ---
10 packets transmitted, 10 received, +2 duplicates, 0% packet loss, time 901ms
rtt min/avg/max/ndev = 5.182/5.683/6.170/0.412 ms
root@Grilo:/tmp/pgcore.40857/Grilo.conf#

```

Figura 2. Resultados obtidos em Grilo

1.2 Pergunta 2

Obtenha a partir do *Wireshark*, ou desenhe manualmente, um diagrama temporal para a transferência do ficheiro *file1* por *FTP* realizada em A.3. Foque-se apenas na transferência de dados [*ftp-data*] e não na conexão de controlo (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 tanto nos dados como nas confirmações.

Resposta: A transferência do file1 utilizando o protocolo FTP exige um número relativamente baixo de trocas de pacotes, como se conclui através dos diagramas.

Na Figura 3, correspondente à transferência Servidor1-Portátil1, vê-se representada a fase de conexão nos primeiros picos. Observa-se também que a transferência do ficheiro exige mais trocas de pacotes, resultando no pico mais alto do diagrama. O último pico representa o fim da conexão ("Goodbye."). Durante todo o processo de conexão e transferência são trocados segmentos TCP e FTP.

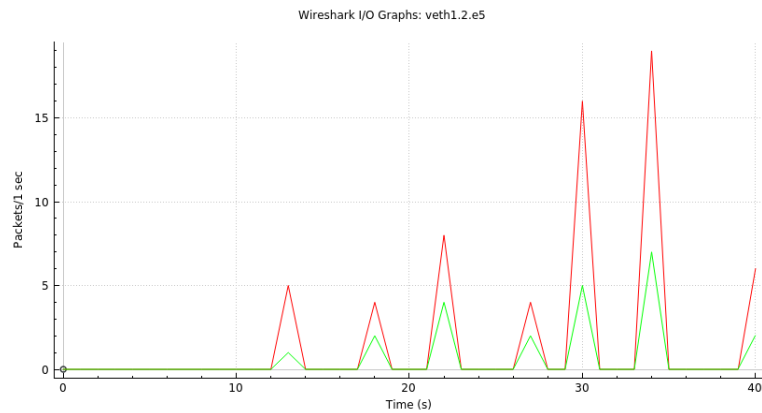


Figura 3. Gráfico do *Wireshark* utilizando TCP (segmentos a vermelhos) e FTP (segmentos a verde) no Portátil 1

No.	Time	Source	Destination	Protocol	Length	Info
12	13.410823185	10.2.2.1	10.1.1.1	FTP	86	Response: 220 (vsFTPd 3.0.3)
18	18.250074248	10.1.1.1	10.2.2.1	FTP	82	Request: USER anonymous
20	18.250265439	10.2.2.1	10.1.1.1	FTP	100	Response: 331 Please specify the password.
24	22.251708145	10.1.1.1	10.2.2.1	FTP	79	Request: PASS a96867
26	22.253171749	10.2.2.1	10.1.1.1	FTP	89	Response: 230 Login successful.
28	22.253680890	10.1.1.1	10.2.2.1	FTP	72	Request: SYST
30	22.253934580	10.2.2.1	10.1.1.1	FTP	85	Response: 215 UNIX Type: L8
35	27.919226536	10.1.1.1	10.2.2.1	FTP	71	Request: PWD
37	27.919561554	10.2.2.1	10.1.1.1	FTP	100	Response: 257 "/" is the current directory
41	30.303436004	10.1.1.1	10.2.2.1	FTP	88	Request: PORT 10,1,1,179,21
42	30.303613081	10.2.2.1	10.1.1.1	FTP	117	Response: 200 PORT command successful. Consider using
44	30.303963260	10.1.1.1	10.2.2.1	FTP	72	Request: LIST
48	30.304506182	10.2.2.1	10.1.1.1	FTP	105	Response: 150 Here comes the directory listing.
55	30.305531349	10.2.2.1	10.1.1.1	FTP	90	Response: 226 Directory send OK.
60	34.442806114	10.1.1.1	10.2.2.1	FTP	74	Request: TYPE I
61	34.442951343	10.2.2.1	10.1.1.1	FTP	97	Response: 200 Switching to Binary mode.
63	34.443130714	10.1.1.1	10.2.2.1	FTP	89	Request: PORT 10,1,1,147,195
64	34.443284820	10.2.2.1	10.1.1.1	FTP	117	Response: 200 PORT command successful. Consider using
66	34.443629358	10.1.1.1	10.2.2.1	FTP	78	Request: RETR file1
70	34.444169155	10.2.2.1	10.1.1.1	FTP	130	Response: 150 Opening BINARY mode data connection for
77	34.445220893	10.2.2.1	10.1.1.1	FTP	90	Response: 226 Transfer complete.
82	40.356684479	10.1.1.1	10.2.2.1	FTP	72	Request: QUIT
83	40.357207223	10.2.2.1	10.1.1.1	FTP	80	Response: 221 Goodbye.

Figura 4. *Wireshark* Portátil 1

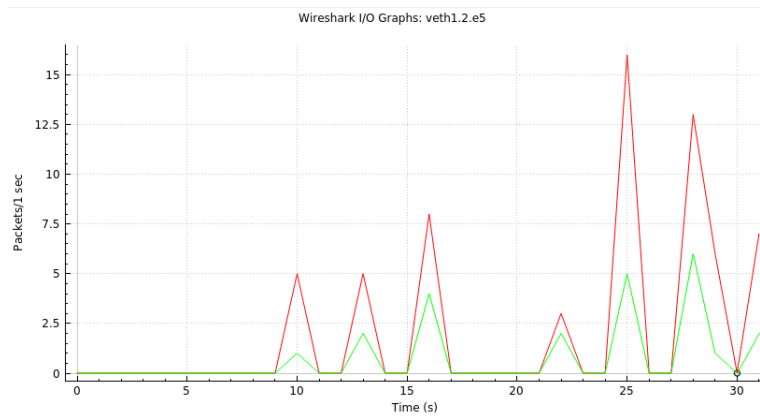


Figura 5. Gráfico do *Wireshark* utilizando TCP (segmentos a vermelhos) e FTP (segmentos a verde) no Grilo

1.3 Pergunta 3

Obtenha a partir do *Wireshark*, ou desenhe manualmente, um diagrama temporal para a transferência do ficheiro *file1* por *TFTP* realizada em A.4. 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 tanto nos dados como nas confirmações.

Resposta: No caso da transferência por *TFTP*, não há uma fase de autenticação, simplesmente uma conexão ao servidor e pedido de transferência de ficheiro, pelo que o diagrama apresenta apenas um pico correspondente a estas ações.

NOTA: na Figura 9 observam-se dois picos que correspondem ao procedimento experimental executado duas vezes (duas transferências do mesmo *file1*).

Durante o processo foram trocados segmentos do tipo:

- Read Request
- Data
- Acknowledgement

No.	Time	Source	Destination	Protocol	Length	Info
12	10.145626608	10.2.2.1	10.4.4.1	FTP	86	Response: 220 (vsFTPd 3.0.3)
15	13.823644520	10.4.4.1	10.2.2.1	FTP	82	Request: USER anonymous
17	13.823817712	10.2.2.1	10.4.4.1	FTP	100	Response: 331 Please specify the password.
22	16.163766358	10.4.4.1	10.2.2.1	FTP	79	Request: PASS a96867
24	16.165186818	10.2.2.1	10.4.4.1	FTP	89	Response: 230 Login successful.
26	16.170309531	10.4.4.1	10.2.2.1	FTP	72	Request: SYST
28	16.170475810	10.2.2.1	10.4.4.1	FTP	85	Response: 215 UNIX Type: L8
34	22.037387306	10.4.4.1	10.2.2.1	FTP	71	Request: PWD
35	22.037539512	10.2.2.1	10.4.4.1	FTP	100	Response: 257 "/" is the current directory
38	25.502597543	10.4.4.1	10.2.2.1	FTP	89	Request: PORT 10,4,4,1,193,253
39	25.502784524	10.2.2.1	10.4.4.1	FTP	117	Response: 200 PORT command successful. Consider using
41	25.507987239	10.4.4.1	10.2.2.1	FTP	72	Request: LIST
45	25.513488156	10.2.2.1	10.4.4.1	FTP	105	Response: 150 Here comes the directory listing.
52	25.519086038	10.2.2.1	10.4.4.1	FTP	90	Response: 226 Directory send OK.
56	28.810574362	10.4.4.1	10.2.2.1	FTP	74	Request: TYPE I
57	28.810738189	10.2.2.1	10.4.4.1	FTP	97	Response: 200 Switching to Binary mode.
59	28.815878988	10.4.4.1	10.2.2.1	FTP	89	Request: PORT 10,4,4,1,165,105
60	28.816038891	10.2.2.1	10.4.4.1	FTP	117	Response: 200 PORT command successful. Consider using
61	28.821120796	10.4.4.1	10.2.2.1	FTP	78	Request: RETR file1
65	28.826592459	10.2.2.1	10.4.4.1	FTP	130	Response: 150 Opening BINARY mode data connection for
73	29.039958181	10.2.2.1	10.4.4.1	FTP	90	Response: 226 Transfer complete.
77	31.555960989	10.4.4.1	10.2.2.1	FTP	72	Request: QUIT
78	31.556339537	10.2.2.1	10.4.4.1	FTP	80	Response: 221 Goodbye.

Figura 6. Wireshark Grilo

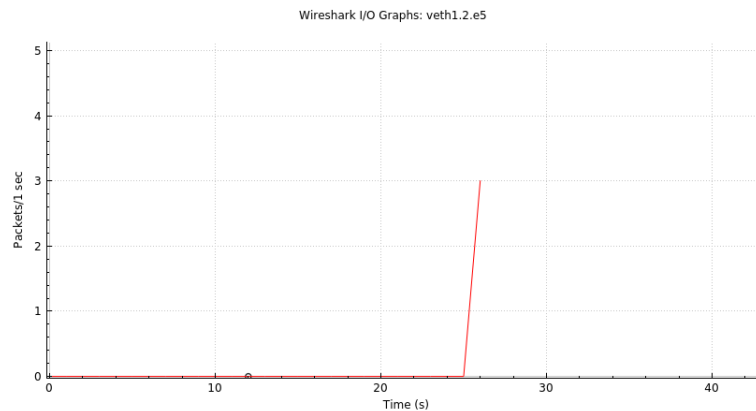
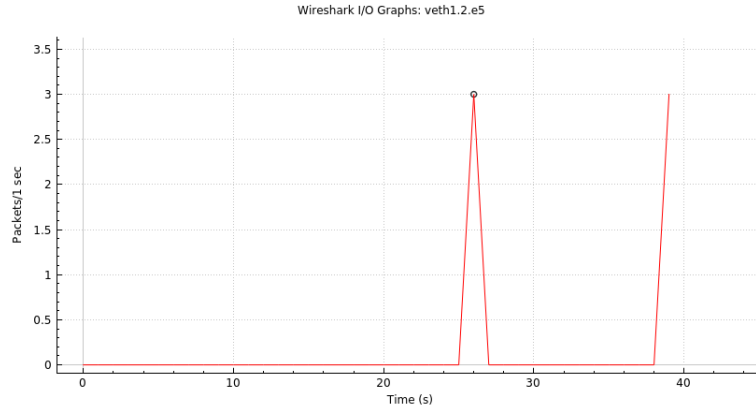


Figura 7. Gráfico Wireshark utilizando TFTP Portátil 1

14	22.002349805	10.2.2.254	224.0.0.5	OSPF	78 Hello Packet
15	24.002410690	10.2.2.254	224.0.0.5	OSPF	78 Hello Packet
16	25.759901253	fe80::200:ff:feaa:10	ff02::5	OSPF	90 Hello Packet
17	26.002730454	10.2.2.254	224.0.0.5	OSPF	78 Hello Packet
18	26.034597505	10.2.2.1	10.2.2.1	TFTP	56 Read Request, File: file1, Transfer type: octet
19	26.973042643	10.2.2.1	10.1.1.1	TFTP	270 Data Packet, Block: 1 (last)
20	26.973278289	10.1.1.1	10.2.2.1	TFTP	46 Acknowledgement, Block: 1
21	28.003106716	10.2.2.254	224.0.0.5	OSPF	78 Hello Packet
22	30.003179228	10.2.2.254	224.0.0.5	OSPF	78 Hello Packet
23	32.003452135	10.2.2.254	224.0.0.5	OSPF	78 Hello Packet
24	37.166145827	00:00:00:00:00:10	00:00:00:00:00:14	ARP	42 Who has 10.2.2.1? Tell 10.2.2.254

Figura 8. Wireshark Portátil 1

Figura 9. Gráfico *Wireshark* utilizando TFTP Grilo

16	26.005748847	10.2.2.254	224.0.0.5	OSPF	78 Hello Packet
17	26.531523735	10.4.4.1	10.2.2.1	TFTP	56 Read Request, File: file1, Transfer type: octet
18	26.531842436	10.2.2.1	10.4.4.1	TFTP	270 Data Packet, Block: 1 (last)
19	26.537355914	10.4.4.1	10.2.2.1	TFTP	46 Acknowledgement, Block: 1
20	27.304222659	fe80::200:ff:feaa:10	ff02::5	OSPF	90 Hello Packet
21	28.000046277	10.2.2.254	224.0.0.5	OSPF	78 Hello Packet
22	30.006474048	10.2.2.254	224.0.0.5	OSPF	78 Hello Packet
23	31.588013531	00:00:00:aa:00:10	00:00:00:aa:00:14	ARP	42 Who has 10.2.2.1? Tell 10.2.2.254
24	31.588159702	00:00:00:aa:00:10	00:00:00:aa:00:10	ARP	42 Who has 10.2.2.254? Tell 10.2.2.1
25	31.588166198	00:00:00:aa:00:10	00:00:00:aa:00:14	ARP	42 10.2.2.254 is at 00:00:00:aa:00:10
26	31.588191442	00:00:00:aa:00:10	00:00:00:aa:00:10	ARP	42 10.2.2.1 is at 00:00:00:aa:00:14
27	32.000602735	10.2.2.254	224.0.0.5	OSPF	78 Hello Packet
28	34.007438350	10.2.2.254	224.0.0.5	OSPF	78 Hello Packet
29	36.007957242	10.2.2.254	224.0.0.5	OSPF	78 Hello Packet
30	37.271421828	fe80::200:ff:feaa:10	ff02::5	OSPF	90 Hello Packet
31	38.008329983	10.2.2.254	224.0.0.5	OSPF	78 Hello Packet
32	39.045936672	10.4.4.1	10.2.2.1	TFTP	56 Read Request, File: file1, Transfer type: octet
33	39.846188995	10.2.2.1	10.4.4.1	TFTP	270 Data Packet, Block: 1 (last)
34	39.851432682	10.4.4.1	10.2.2.1	TFTP	46 Acknowledgement, Block: 1
35	40.009487100	10.2.2.254	224.0.0.5	OSPF	78 Hello Packet
36	42.009859307	10.2.2.254	224.0.0.5	OSPF	78 Hello Packet
37	44.010395716	10.2.2.254	224.0.0.5	OSPF	78 Hello Packet

Figura 10. *Wireshark* Grilo


```

multicast: disabled
mtftp variables
client-port: 76
mcast-ip: 0.0.0.0
listen-delay: 2
timeout-delay: 2
Last command: ---
tftp> get file1
Overwrite local file [y/n]? y
timeout: retrying...
timeout: retrying...
tftp> get file1
Overwrite local file [y/n]? y
tftp> quit
root@Grilo:/tmp/pycore.43599/Grilo.conf#

```

Figura 11. *Timeouts* Grilo

1.4 Pergunta 4

Compare sucintamente as quatro aplicações de transferência de ficheiros que usou, tendo em consideração os seguintes aspetos: (i) identificação da camada de transporte; (ii) eficiência; (iii) complexidade; (iv) segurança.

Resposta:

Protocolos	Uso de camada transporte (i)	Eficiência (ii)	Complexidade (iii)	Segurança (iv)
SFTP	TCP	Baixa	Alta	Alta Encriptação de dados
FTP	TCP	Médio	Média	Baixa Autenticação, fácil descodificação de informação
TFTP	UDP	Alta	Baixo	Baixa
HTTP	TCP	Alta	Baixa	Muito baixa Sem autenticação

O UDP provou ser mais rápido e eficiente uma vez que comparativamente ao TCP não reenvia pacotes previamente perdidos. Justificando alguns acontecimentos, entre eles a alta eficiência do protocolo TFTP em relação aos outros testados para este relatório.

Nota: Relativamente à baixa segurança do FTP, podemos verificar isso através de 4, onde conseguimos visualizar a password do usuário.

1.5 Pergunta 5

Com base no trabalho realizado, construa uma tabela informativa identificando, para cada aplicação executada (*ping*, *traceroute*, *telnet*, *ftp*, *tftp*, *wget/lynx*, *nslookup*, *ssh*, *etc.*), qual o protocolo de aplicação, o protocolo de transporte, a porta de atendimento e o *overhead* de transporte.

Aplicações	Protocolo de Aplicação	Protocolo de Transporte	Porta de Atendimento	Overhead de Transporte (em bytes)
HTTP	HTTP	TCP	80	20
FTP	FTP	TCP	21	20
TFTP	TFTP	UDP	69	8
Telnet	Telnet	TCP	23	20
Nslookup	DNS	UDP	53	8
Ping	—	—	—	—
Traceroute	DNS	UDP	33434	8

2 Conclusão

Com este estudo, tivemos a oportunidade de interiorizar os conceitos lecionados nas aulas teóricas fortificando as bases de Redes de Computadores e revisitamos os nossos conhecimentos associados ao uso de ferramentas como CORE e o *Wireshark*.

Passamos a ter uma noção um pouco mais aprofundada das vantagens e desvantagens associadas à utilização dos diferentes protocolos TCP e UDP em contextos diferentes e aplicações de transferência de ficheiros entre *hosts* como SFTP, FTP, TFTP e HTTP.

O protocolo TCP provou ser menos eficiente, necessitando de mais recursos e sendo mais lento comparativamente ao UDP, estando associado aplicações onde se busca maior fiabilidade e segurança em troca de velocidade, uma vez que este garante a chegada do pacote esperando sempre a chegada de uma mensagem ACK antes do envio do pacote seguinte.

Por outro lado, com UDP, é possível uma transferência mais rápida e eficiente abdicando da garantia de que o pacote chega ao destino, sendo assim um protocolo menos fiável. Geralmente é utilizado em serviços de streaming e VoIP onde a perda de um pacote não compromete totalmente a informação transmitida.