

**Trabalho Prático Nº 1**  
**Protocolos da Camada de Transporte**  
-  
**Comunicação por Computadores**

Rafael Silva and José Ramos

University of Minho, Department of Informatics, 4710-057 Braga, Portugal  
e-mail: {a74264,a73855}@alunos.uminho.pt

# 1 Desenvolvimento

## 1.1 Questões e respostas

### 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 (aplicação)	Protocolo de Aplicação (se aplicável)	Protocolo de transporte (se aplicável)	Porta de atendimento (se aplicável)	Overhead de transporte em bytes (se aplicável)
Ping	X	X	10.1.1.1	X
tracert	DNS	UDP	53	X
telnet	TLS <sub>v1.2</sub>	TCP	23	ad 45 01 b6 5d d8 ad d8 8a e1 ef 50 7b 85 20 0a 4c 00 00
ftp	TLS <sub>v1.2</sub>	TCP	443	01 b5 95 0c 09 85 83 f5 6e 31 a0 ea 20 10 f1 ff a3 64 00 00
Tftp	TFTP	UDP	69	X
browser/http	HTTP	TCP	80	00 50 d6 92 08 28 6e 1 f5 3b 08 1a 60 12 ff 00 98 00 00 01 05 4c
nslookup	DNS	UDP	53	X
ssh	SSH <sub>v2</sub>	TCP	22	00 10 b1 8f 0c 83 d4 d5 e1 a7 c3 fe 50 8 ff ff 0a 1e 00 00
Outras:				

Figura 1: Tabela de questão 1

## 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.

**Resposta:**

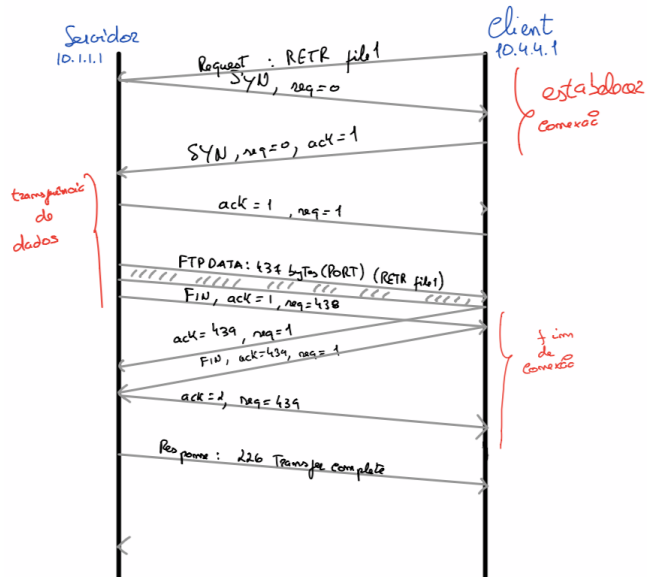


Figura 2: Timeline de transmissão FTP

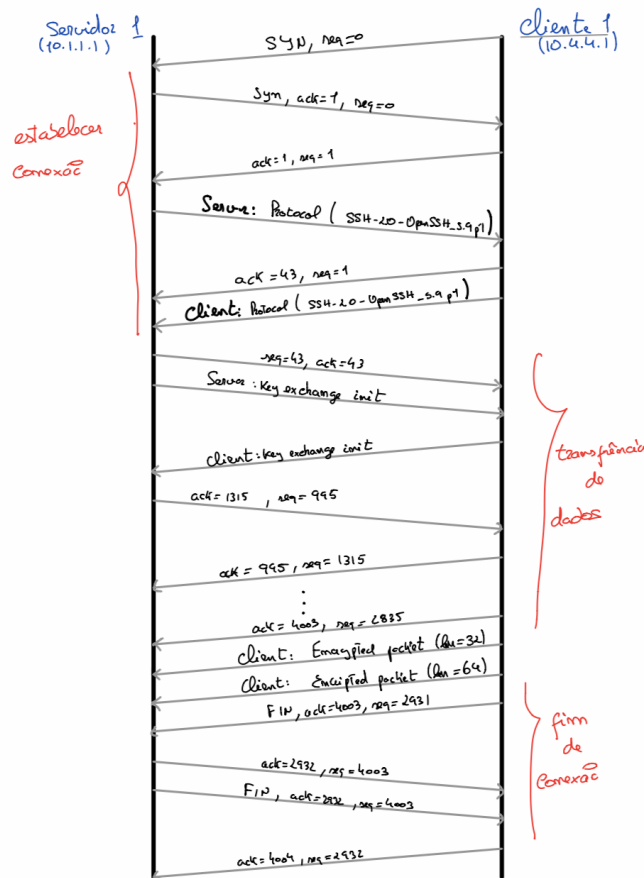


Figura 3: Timeline de transmissão SFTP

### 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;

#### 1. FTP - File Transfer Protocol

(i) uso da camada de transporte:

- O servidor remoto aceita uma conexão de controlo do cliente a nível local. O cliente envia comandos para o servidor e a conexão persiste ao longo de toda a sessão (tratando-se assim de um protocolo TCP);

(ii) eficiência na transferência;

- Não muito eficiente uma vez que envia toda a informação numa só Stream de bits guardada como uma string limitando a capacidade de controlo de erros e ate mesmo velocidade de transmissão.

(iii) complexidade;

- Envia uma Stream de bits guardada como uma única string que contem a informação e toda o overhead necessário para transmissão (ex : nome do ficheiro, tamanho, timestamp)

(iv) *segurança;*

- FTP é considerada uma das opções menos seguras no que toca a transmissão de ficheiros em rede uma vez que não encripta os dados e para a transferência destes não considera a segurança desta, havendo diversos tipos de formas de intercetar transferências em FTP como por exemplo : *FTP Bounce Attack* , *FTP Brute Force Attack* , *Packet Capture (or Sniffing)*, *Spoof Attack* ou *Port Stealing*

## **2. SFTP -Secure File Transfer Protocol**

(i) *uso da camada de transporte:*

A transferência dos dados é realizado através de uma conexão previamente assegurada, que utiliza o protocolo SSH (*Secure Shell protocol*).

(ii) *eficiência na transferência;*

- SFTP é um protocolo baseado em transferência de pacotes ao invés de baseados em texto. Assim ao enviar pequenos pacotes com a informação devidamente dividida e preparada entre si consegue ser um método mais rápido e a capaz de produzir um menor numero de erros e de perda de informação

(iii) *complexidade;*

- Transferências em SFTP são realizadas através de uma conexão SSH e ocorre a preparação da informação em data packages. Além disso, esta transferência é realizada sobre o controlo de conexão principal , eliminando a necessidade de abrir uma nova conexão de dados somente para este evento.

(iv) *segurança;*

- Uma vez que utiliza o protocolo SSH é considerada intrinsecamente segura

## **3. TFTP - Trivial File Transfer Protocol**

(i) *uso da camada de transporte:*

- Utiliza o protocolo de transporte UDP

(ii) *eficiência na transferência;*

- Ao utilizar os protocolo UDP o TFTP peca por não possuir tanto controlo de erros, havendo assim alguma perda na sua eficiência. Para além disso tem também de suportar a sessão criada para o envio da informação.

(iii) *complexidade;*

- Transferência é iniciada pelo cliente pedindo para ler ou escrever um ficheiro num servidor. Se este aceita o pedido o ficheiro é enviado em blocos de tamanho fixo , cada um geralmente possuindo apenas um único pacote IP de modo a evitar a sua fragmentação e deve esperar pelo acknowledgment de modo a poder enviar o pacote seguinte.

(iv) *segurança;*

- O TFTP não possui nenhum mecanismo de controlo de acesso. Assim devemos ter cuidado ao enviar ficheiros de carácter privado e ter em atenção aos direitos garantidos ao servidor TFTP de modo a não violar a segurança d  
o servidor onde o ficheiro se encontra guardado.

#### **4. HTTP - Hypertext Control Protocol**

*(i) uso da camada de transporte:*

- Aplicação de camada de transporte desenhada com a framework do protocolo de Internet em mente, assumindo assim capacidade de transporte segura e eficaz.

*(ii) eficiência na transferência;*

- Eficiente no sentido em que a sua arquitetura é projetada para permitir intermediação entre elementos de rede de modo a melhorar e permitir comunicações entre servidores e clientes. Web sites com muita procura e tráfego normalmente apresentam servers cache que permitem melhorar o tempo de resposta.

*(iii) complexidade;*

- Devido a utilização de servers cache e outros métodos que permitem melhorar a eficiência da transmissão, verificamos que existe um aumento a nível de recursos utilizados e também na complexidade da sua gestão que permite este melhoramento.

*(iv) segurança;*

- O HTTP utiliza vários métodos de autentificação para a ocorrência de transferências, quer seja acessos básicos ou acessos a web sites com informação mais importante os métodos de autentificação e segurança vão melhorando, fornecendo assim um ambiente seguro para o envio de dados.

#### 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).

```
root@Cliente1: /tmp/pycore.46788/Cliente1.conf
PING 10.1.1.1 (10.1.1.1) 56(84) bytes of data:
64 bytes from 10.1.1.1: icmp_req=1 ttl=62 time=0.668 ms
64 bytes from 10.1.1.1: icmp_req=2 ttl=62 time=0.486 ms
64 bytes from 10.1.1.1: icmp_req=3 ttl=62 time=0.369 ms
64 bytes from 10.1.1.1: icmp_req=4 ttl=62 time=0.346 ms
64 bytes from 10.1.1.1: icmp_req=5 ttl=62 time=0.324 ms
64 bytes from 10.1.1.1: icmp_req=6 ttl=62 time=0.390 ms
64 bytes from 10.1.1.1: icmp_req=7 ttl=62 time=0.547 ms
64 bytes from 10.1.1.1: icmp_req=8 ttl=62 time=0.435 ms
64 bytes from 10.1.1.1: icmp_req=9 ttl=62 time=0.380 ms
64 bytes from 10.1.1.1: icmp_req=10 ttl=62 time=0.441 ms
64 bytes from 10.1.1.1: icmp_req=11 ttl=62 time=0.451 ms
64 bytes from 10.1.1.1: icmp_req=12 ttl=62 time=0.557 ms
64 bytes from 10.1.1.1: icmp_req=13 ttl=62 time=0.369 ms
64 bytes from 10.1.1.1: icmp_req=14 ttl=62 time=0.542 ms
64 bytes from 10.1.1.1: icmp_req=15 ttl=62 time=0.501 ms
64 bytes from 10.1.1.1: icmp_req=16 ttl=62 time=0.325 ms
64 bytes from 10.1.1.1: icmp_req=17 ttl=62 time=0.485 ms
64 bytes from 10.1.1.1: icmp_req=18 ttl=62 time=0.556 ms
64 bytes from 10.1.1.1: icmp_req=19 ttl=62 time=0.441 ms
64 bytes from 10.1.1.1: icmp_req=20 ttl=62 time=0.470 ms

--- 10.1.1.1 ping statistics ---
20 packets transmitted, 20 received, 0% packet loss, time 18990ms
rtt min/avg/max/mdev = 0.324/0.453/0.668/0.081 ms
file-ping-output (END)
```

Figura 4: Ping efetuado em Cliente1

```
PING 10.1.1.1 (10.1.1.1) 56(84) bytes of data:
64 bytes from 10.1.1.1: icmp_req=1 ttl=61 time=12.4 ms
64 bytes from 10.1.1.1: icmp_req=2 ttl=61 time=5.39 ms
64 bytes from 10.1.1.1: icmp_req=3 ttl=61 time=5.45 ms
64 bytes from 10.1.1.1: icmp_req=4 ttl=61 time=7.12 ms
64 bytes from 10.1.1.1: icmp_req=4 ttl=61 time=7.15 ms (DUPL)
64 bytes from 10.1.1.1: icmp_req=5 ttl=61 time=6.35 ms
64 bytes from 10.1.1.1: icmp_req=6 ttl=61 time=6.18 ms
64 bytes from 10.1.1.1: icmp_req=7 ttl=61 time=6.09 ms
64 bytes from 10.1.1.1: icmp_req=8 ttl=61 time=6.05 ms
64 bytes from 10.1.1.1: icmp_req=9 ttl=61 time=6.27 ms
64 bytes from 10.1.1.1: icmp_req=10 ttl=61 time=5.43 ms
64 bytes from 10.1.1.1: icmp_req=11 ttl=61 time=5.50 ms
64 bytes from 10.1.1.1: icmp_req=12 ttl=61 time=5.37 ms
64 bytes from 10.1.1.1: icmp_req=13 ttl=61 time=6.41 ms
64 bytes from 10.1.1.1: icmp_req=14 ttl=61 time=6.28 ms
64 bytes from 10.1.1.1: icmp_req=15 ttl=61 time=6.24 ms
64 bytes from 10.1.1.1: icmp_req=16 ttl=61 time=5.44 ms
64 bytes from 10.1.1.1: icmp_req=17 ttl=61 time=6.63 ms
64 bytes from 10.1.1.1: icmp_req=17 ttl=61 time=6.64 ms (DUPL)
64 bytes from 10.1.1.1: icmp_req=18 ttl=61 time=6.45 ms
64 bytes from 10.1.1.1: icmp_req=19 ttl=61 time=6.73 ms
64 bytes from 10.1.1.1: icmp_req=20 ttl=61 time=9.19 ms

--- 10.1.1.1 ping statistics ---
20 packets transmitted, 20 received, +2 duplicates, 0% packet loss, time 19041ms
rtt min/avg/max/mdev = 5.373/6.586/12.441/1.521 ms
file-ping-output (END)
```

Figura 5: Ping efetuado em Alfa

Podemos verificar que o ping efetuado no *Cliente1* foi mais rápido do que no *Alfa*, e também podemos verificar que no *Alfa* existiam dois pacotes duplicados mas em ambos não se perderam pacotes, sendo recebido os vinte.

```
root@Cliente1:/tmp/pycore.46829/Cliente1.conf# wget http://10.1.1.1/file2
--2019-02-28 17:36:19-- http://10.1.1.1/file2
Connecting to 10.1.1.1:80... connected.
HTTP request sent, awaiting response... 200 Ok
Length: 104508 (102K) [text/plain]
Saving to: 'file2'

100%[=====>] 104,508      165K/s   in 0.6s

2019-02-28 17:36:20 (165 KB/s) - 'file2' saved [104508/104508]
```

Figura 6: **http** (file2) efetuado em Cliente1

```
root@Alfa:/tmp/pycore.46829/Alfa.conf# wget http://10.1.1.1/file2
--2019-02-28 17:36:48-- http://10.1.1.1/file2
Connecting to 10.1.1.1:80... connected.
HTTP request sent, awaiting response... 200 Ok
Length: 104508 (102K) [text/plain]
Saving to: 'file2'

100%[=====>] 104,508      40.0K/s   in 2.5s

2019-02-28 17:36:51 (40.0 KB/s) - 'file2' saved [104508/104508]
```

Figura 7: **http** (file2) efetuado em Alfa

Nas imagens apresentadas em cima, ou seja, na **Figura 5** e na **Figura 6** temos o *download* do ficheiro **file2** através do protocolo **http** e podemos de imediato verificar que o *download* do mesmo é mais rápido pelo *Cliente1* do que pelo *Alfa*, isso é devido a conexão que ambos têm com a rede *Backbone*, sendo o mais rápido o *Cliente1*.

Sendo o protocolo **http** de camada de aplicação, usando como meio protocolo de camada de transportes o **TCP**.



```
ftp> get file2
local: file2 remote: file2
200 PORT command successful. Consider using PASV.
150 Opening BINARY mode data connection for file2 (104508 bytes).
226 Transfer complete.
104508 bytes received in 0.63 secs (162.3 kB/s)
ftp> quit
221 Goodbye.
root@Cliente1:/tmp/pycore.46835/Cliente1.conf#
```

Figura 8: **ftp** (file2) efetuado em Cliente1

```
ftp> get file2
local: file2 remote: file2
200 PORT command successful. Consider using PASV.
150 Opening BINARY mode data connection for file2 (104508 bytes).
226 Transfer complete.
104508 bytes received in 2.77 secs (36.8 kB/s)
ftp> quit
221 Goodbye.
root@Alfa:/tmp/pycore.46835/Alfa.conf#
```

Figura 9: **ftp** (file2) efetuado em Alfa

Nas figuras em cima podemos verificar o *download* do *file2* através do protocolo de **FTP** que pertence a camada de aplicação e o mesmo usa p protocolo de transporte **TCP**, ou seja, que iremos ter o servidor que envia o ficheiro em questão e fica *a espera* de uma resposta sempre que envia, dai vemos que o ficheiro foi recebido por ambos, quer pelo *Alfa*, quer pelo *Cliente*.

Porém podemos verificar que o *download* mais rápido é feito pelo Cliente1, isto devido a ligação com a rede *backbone*.

```

root@Cliente1:/tmp/pycore.46835/Cliente1.conf# atftp 10.1.1.1
tftp> status
Connected: 10.1.1.1 port 69
Mode:      octet
Verbose:   off
Trace:     off
Options
  tsize:    disabled
  blksize:  disabled
  timeout:  disabled
  multicast: disabled
mtftp variables
  client-port: 76
  wcast-ip:    0.0.0.0
  listen-delay: 2
  timeout-delay: 2
Last command: quit
tftp> get file2
Overwrite local file [y/n]? y
timeout: retrying...
timeout: retrying...
timeout: retrying...
timeout: retrying...
timeout: retrying...
timeout: retrying...
tftp: aborting

```

Figura 10: **tftp** (file2) efetuado em Cliente1

```

root@Alfa:/tmp/pycore.46835/Alfa.conf# atftp 10.1.1.1
tftp> status
Connected: 10.1.1.1 port 69
Mode:      octet
Verbose:   off
Trace:     off
Options
  tsize:    disabled
  blksize:  disabled
  timeout:  disabled
  multicast: disabled
mtftp variables
  client-port: 76
  wcast-ip:    0.0.0.0
  listen-delay: 2
  timeout-delay: 2
Last command: quit
tftp> get file2
Overwrite local file [y/n]? y
timeout: retrying...
timeout: retrying...
timeout: retrying...
timeout: retrying...
timeout: retrying...
timeout: retrying...
tftp: aborting

```

Figura 11: **tftp** (file2) efetuado em Alfa

Podemos verificar nas figuras em cima que o *download* efetuado do *file2* usa o protocolo **TFTP** e que o mesmo usa como camada de transportes o **UDP**, ou seja, que caso não esteja ninguém no outro lado para receber o que foi enviado pelo servidor, o mesmo ficheiro fica perdido, e não é recebido nem pelo Cliente1 nem pelo Alfa.

Podemos concluir então através da análise das figuras e do que foi dito acima que o protocolo **TCP** é mais seguro no que toca a receção de ficheiro, porém as vezes a velocidade do mesmo é sacrificada, enquanto que o protocolo **UDP** não tem um controlo de chegadas por assim dizer, ou seja, que quando um ficheiro é enviado nunca iremos saber se o mesmo foi entregue ou não, porém a velocidade é superior a do **TCP** em alguns casos.

## **2 Conclusão**

.

O trabalho prático proposto na UC para a abordagem do tema de Protocolos da Camada de Transporte com recurso à ferramenta Wireshark apresentou vários assuntos pertinentes na gestão da troca e procura de informação.

Ao resolver os exercícios, capturamos tráfego em wireshark e analisamo-lo.

Verificamos também a diferença entre os diferentes protocolos a serem usados, bem como as diferentes camadas existentes.

Podemos verificar também que a ligação com uma certa rede pode afetar o seu desempenho.