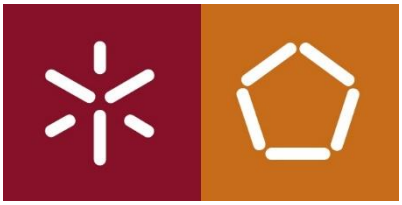
# Comunicações por Computador



## Trabalho prático nº1

4 de março de 2020


**PL4 - Grupo nº2**

Hugo André Coelho Cardoso (A85006)

João da Cunha e Costa (A84775)

Válter Ferreira Picas Carvalho (A84464)



Mestrado Integrado em Engenharia Informática

Universidade do Minho

# 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.**

| 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 | não aplicável | não aplicável | não aplicável | não aplicável |
| traceroute | mdns | UDP | Várias* | 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 |
| Outras? | - | - | - | - |

*o protocolo UDP envia cada pacote para uma porta diferente, incrementando-as

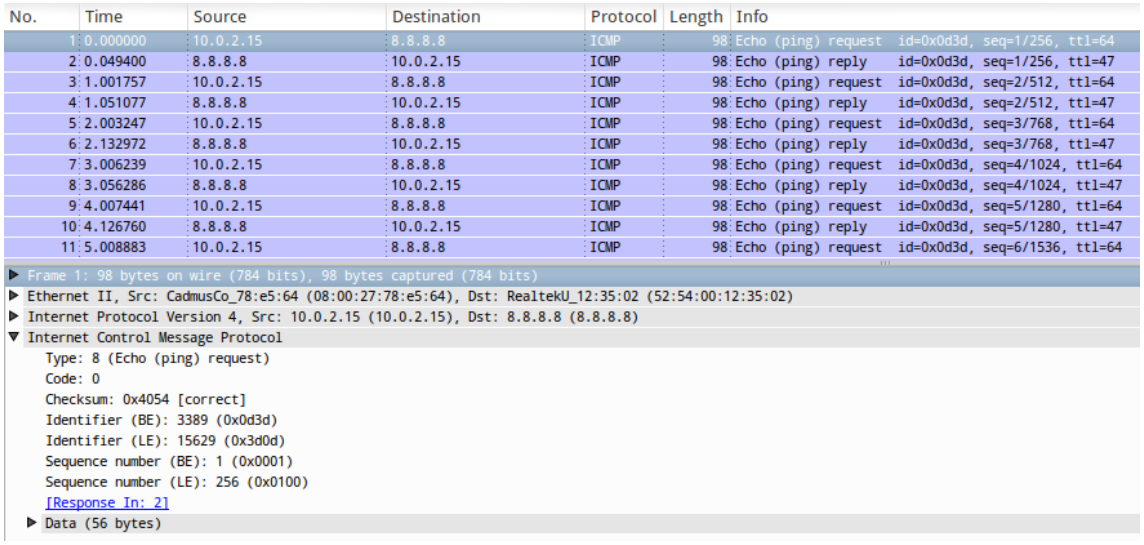



Figura 1 - Ping.

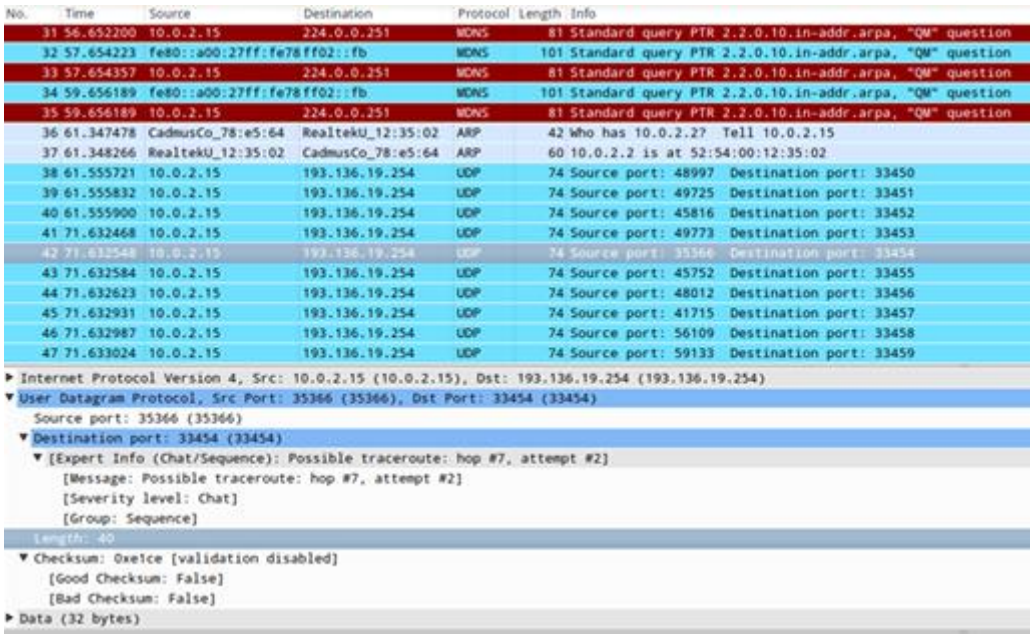



Figura 2 - Traceroute.

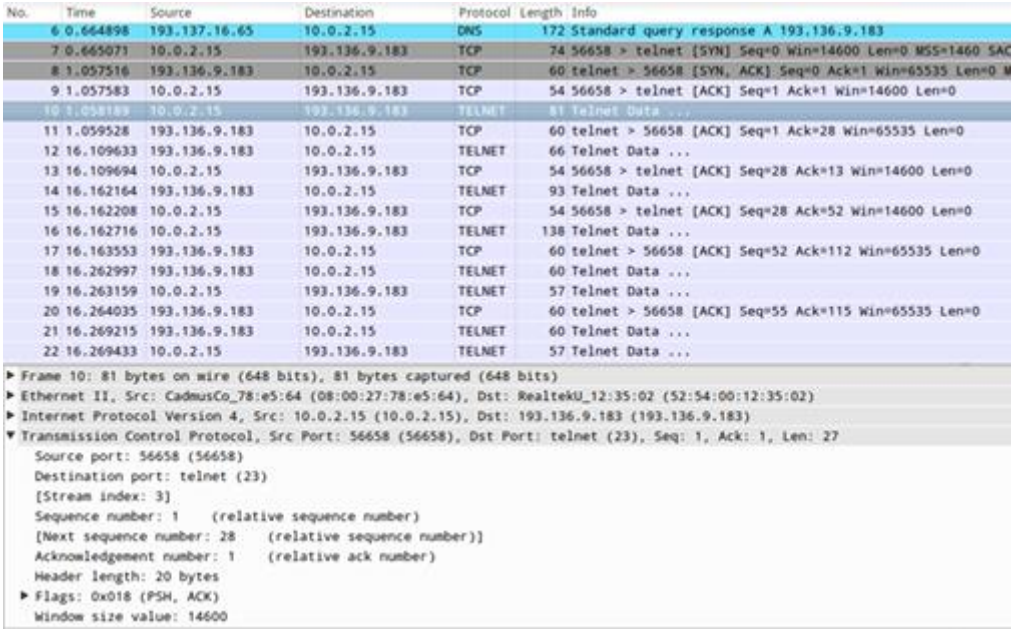
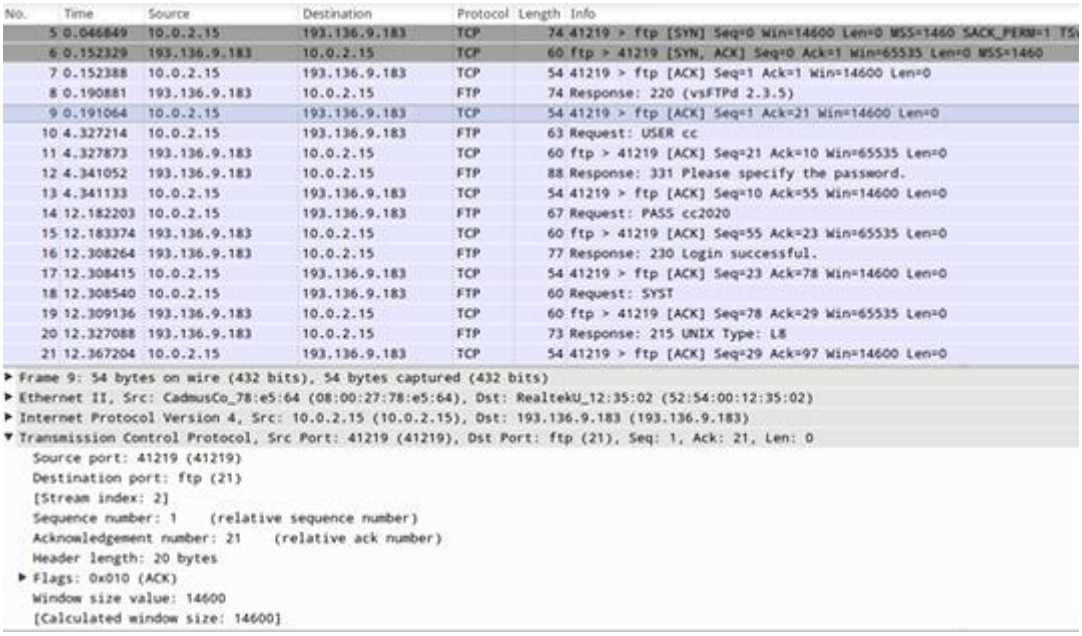


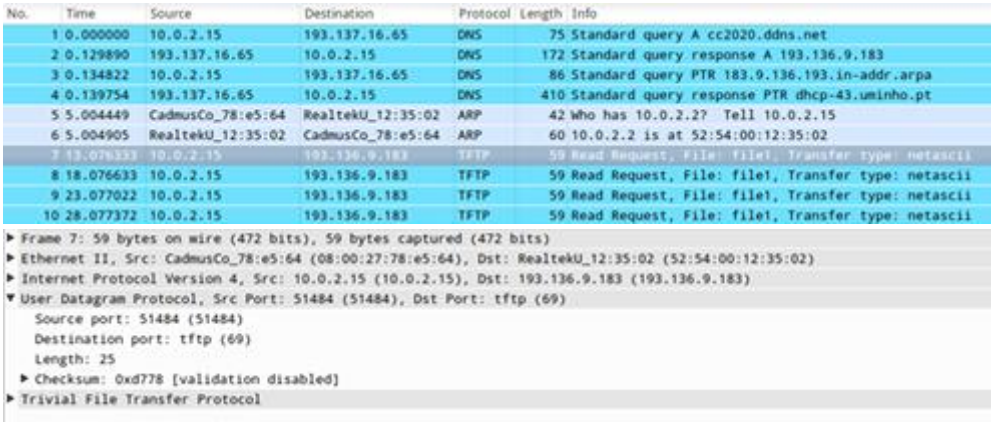

Figura 3 - Telnet.



| No. | Time | Source | Destination | Protocol | Length | Info |
|---|---|---|---|---|---|---|
| 5 | 0.046849 | 10.0.2.15 | 193.136.9.183 | TCP | 74 | 41219 > ftp [SYN] Seq=0 Win=14600 Len=0 MSS=1460 SACK_PERM=1 TS |
| 6 | 0.152329 | 193.136.9.183 | 10.0.2.15 | TCP | 60 | ftp > 41219 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1460 |
| 7 | 0.152388 | 10.0.2.15 | 193.136.9.183 | TCP | 54 | 41219 > ftp [ACK] Seq=1 Ack=1 Win=14600 Len=0 |
| 8 | 0.190881 | 193.136.9.183 | 10.0.2.15 | FTP | 74 | Response: 220 (vsFTPd 2.3.5) |
| 9 | 0.191064 | 10.0.2.15 | 193.136.9.183 | TCP | 54 | 41219 > ftp [ACK] Seq=1 Ack=21 Win=14600 Len=0 |
| 10 | 4.327214 | 10.0.2.15 | 193.136.9.183 | FTP | 63 | Request: USER cc |
| 11 | 4.327873 | 193.136.9.183 | 10.0.2.15 | TCP | 60 | ftp > 41219 [ACK] Seq=21 Ack=10 Win=65535 Len=0 |
| 12 | 4.341052 | 193.136.9.183 | 10.0.2.15 | FTP | 88 | Response: 331 Please specify the password. |
| 13 | 4.341133 | 10.0.2.15 | 193.136.9.183 | TCP | 54 | 41219 > ftp [ACK] Seq=10 Ack=55 Win=14600 Len=0 |
| 14 | 12.182203 | 10.0.2.15 | 193.136.9.183 | FTP | 67 | Request: PASS cc2020 |
| 15 | 12.183374 | 193.136.9.183 | 10.0.2.15 | TCP | 60 | ftp > 41219 [ACK] Seq=55 Ack=23 Win=65535 Len=0 |
| 16 | 12.308264 | 193.136.9.183 | 10.0.2.15 | FTP | 77 | Response: 230 Login successful. |
| 17 | 12.308415 | 10.0.2.15 | 193.136.9.183 | TCP | 54 | 41219 > ftp [ACK] Seq=23 Ack=78 Win=14600 Len=0 |
| 18 | 12.308540 | 10.0.2.15 | 193.136.9.183 | FTP | 60 | Request: SYST |
| 19 | 12.309136 | 193.136.9.183 | 10.0.2.15 | TCP | 60 | ftp > 41219 [ACK] Seq=78 Ack=29 Win=65535 Len=0 |
| 20 | 12.327088 | 193.136.9.183 | 10.0.2.15 | FTP | 73 | Response: 215 UNIX Type: L8 |
| 21 | 12.367204 | 10.0.2.15 | 193.136.9.183 | TCP | 54 | 41219 > ftp [ACK] Seq=29 Ack=97 Win=14600 Len=0 |

```
▶ Frame 9: 54 bytes on wire (432 bits), 54 bytes captured (432 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)
▼ Transmission Control Protocol, Src Port: 41219 (41219), Dst Port: ftp (21), Seq: 1, Ack: 21, Len: 0
    Source port: 41219 (41219)
    Destination port: ftp (21)
    [Stream index: 2]
    Sequence number: 1    (relative sequence number)
    Acknowledgement number: 21    (relative ack number)
    Header length: 20 bytes
  ▶ Flags: 0x010 (ACK)
    Window size value: 14600
    [Calculated window size: 14600]
```

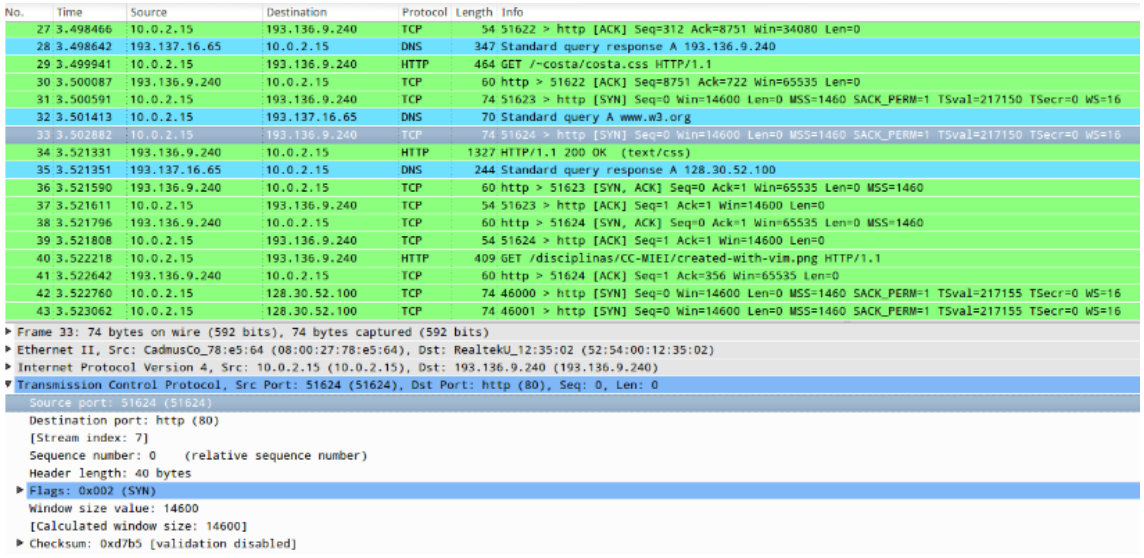
Figura 4 - Ftp.



| No. | Time | Source | Destination | Protocol | Length | Info |
|---|---|---|---|---|---|---|
| 1 | 0.000000 | 10.0.2.15 | 193.137.16.65 | DNS | 75 | Standard query A cc2020.ddns.net |
| 2 | 0.129890 | 193.137.16.65 | 10.0.2.15 | DNS | 172 | Standard query response A 193.136.9.183 |
| 3 | 0.134822 | 10.0.2.15 | 193.137.16.65 | DNS | 86 | Standard query PTR 183.9.136.193.in-addr.arpa |
| 4 | 0.139754 | 193.137.16.65 | 10.0.2.15 | DNS | 410 | Standard query response PTR dhcp-43.uminho.pt |
| 5 | 5.004449 | CadmusCo_78:e5:64 | RealtekU_12:35:02 | ARP | 42 | Who has 10.0.2.2? Tell 10.0.2.15 |
| 6 | 5.004905 | RealtekU_12:35:02 | CadmusCo_78:e5:64 | ARP | 60 | 10.0.2.2 is at 52:54:00:12:35:02 |
| 7 | 13.076333 | 10.0.2.15 | 193.136.9.183 | TFTP | 59 | Read Request, File: file1, Transfer type: netascii |
| 8 | 18.076633 | 10.0.2.15 | 193.136.9.183 | TFTP | 59 | Read Request, File: file1, Transfer type: netascii |
| 9 | 23.077022 | 10.0.2.15 | 193.136.9.183 | TFTP | 59 | Read Request, File: file1, Transfer type: netascii |
| 10 | 28.077372 | 10.0.2.15 | 193.136.9.183 | TFTP | 59 | Read Request, File: file1, Transfer type: netascii |

```
▶ Frame 7: 59 bytes on wire (472 bits), 59 bytes captured (472 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)
▼ User Datagram Protocol, Src Port: 51484 (51484), Dst Port: tftp (69)
    Source port: 51484 (51484)
    Destination port: tftp (69)
    Length: 25
  ▶ Checksum: 0xd778 [validation disabled]
▶ Trivial File Transfer Protocol
```

Figura 5 -Tftp.



| No. | Time | Source | Destination | Protocol | Length | Info |
|---|---|---|---|---|---|---|
| 27 | 3.498466 | 10.0.2.15 | 193.136.9.240 | TCP | 54 | 51622 > http [ACK] Seq=312 Ack=8751 Win=34080 Len=0 |
| 28 | 3.498642 | 193.137.16.65 | 10.0.2.15 | DNS | 347 | Standard query response A 193.136.9.240 |
| 29 | 3.499941 | 10.0.2.15 | 193.136.9.240 | HTTP | 464 | GET /~costa/costa.css HTTP/1.1 |
| 30 | 3.500087 | 193.136.9.240 | 10.0.2.15 | TCP | 60 | http > 51622 [ACK] Seq=8751 Ack=722 Win=65535 Len=0 |
| 31 | 3.500591 | 10.0.2.15 | 193.136.9.240 | TCP | 74 | 51623 > http [SYN] Seq=0 Win=14600 Len=0 MSS=1460 SACK_PERM=1 TSval=217150 TSecr=0 WS=16 |
| 32 | 3.501413 | 10.0.2.15 | 193.137.16.65 | DNS | 70 | Standard query A www.w3.org |
| 33 | 3.502882 | 10.0.2.15 | 193.136.9.240 | TCP | 74 | 51624 > http [SYN] Seq=0 Win=14600 Len=0 MSS=1460 SACK_PERM=1 TSval=217150 TSecr=0 WS=16 |
| 34 | 3.521331 | 193.136.9.240 | 10.0.2.15 | HTTP | 1327 | HTTP/1.1 200 OK  (text/css) |
| 35 | 3.521351 | 193.137.16.65 | 10.0.2.15 | DNS | 244 | Standard query response A 128.30.52.100 |
| 36 | 3.521590 | 193.136.9.240 | 10.0.2.15 | TCP | 60 | http > 51623 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1460 |
| 37 | 3.521611 | 10.0.2.15 | 193.136.9.240 | TCP | 54 | 51623 > http [ACK] Seq=1 Ack=1 Win=14600 Len=0 |
| 38 | 3.521796 | 193.136.9.240 | 10.0.2.15 | TCP | 60 | http > 51624 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1460 |
| 39 | 3.521808 | 10.0.2.15 | 193.136.9.240 | TCP | 54 | 51624 > http [ACK] Seq=1 Ack=1 Win=14600 Len=0 |
| 40 | 3.522218 | 10.0.2.15 | 193.136.9.240 | HTTP | 409 | GET /disciplinas/CC-MIEI/created-with-vim.png HTTP/1.1 |
| 41 | 3.522642 | 193.136.9.240 | 10.0.2.15 | TCP | 60 | http > 51624 [ACK] Seq=1 Ack=356 Win=65535 Len=0 |
| 42 | 3.522760 | 10.0.2.15 | 128.30.52.100 | TCP | 74 | 46000 > http [SYN] Seq=0 Win=14600 Len=0 MSS=1460 SACK_PERM=1 TSval=217155 TSecr=0 WS=16 |
| 43 | 3.523062 | 10.0.2.15 | 128.30.52.100 | TCP | 74 | 46001 > http [SYN] Seq=0 Win=14600 Len=0 MSS=1460 SACK_PERM=1 TSval=217155 TSecr=0 WS=16 |

```
▶ Frame 33: 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: 193.136.9.240 (193.136.9.240)
▼ Transmission Control Protocol, Src Port: 51624 (51624), Dst Port: http (80), Seq: 0, Len: 0
    Source port: 51624 (51624)
    Destination port: http (80)
    [Stream index: 7]
    Sequence number: 0    (relative sequence number)
    Header length: 40 bytes
  ▶ Flags: 0x002 (SYN)
    Window size value: 14600
    [Calculated window size: 14600]
  ▶ Checksum: 0xd7b5 [validation disabled]
```

Figura 6- Http.

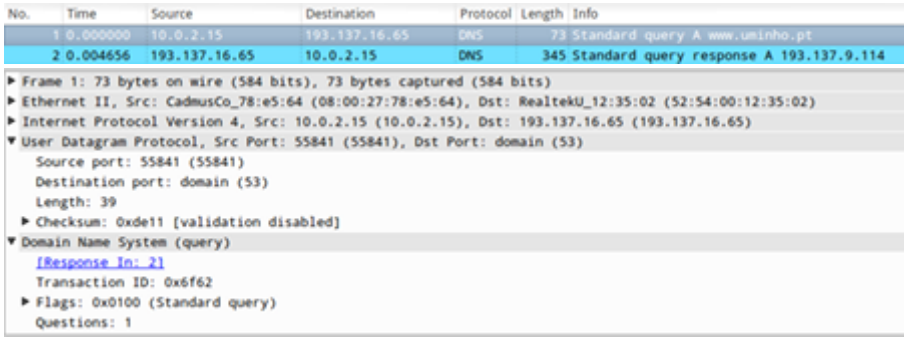



Figura 7 - Nslookup.

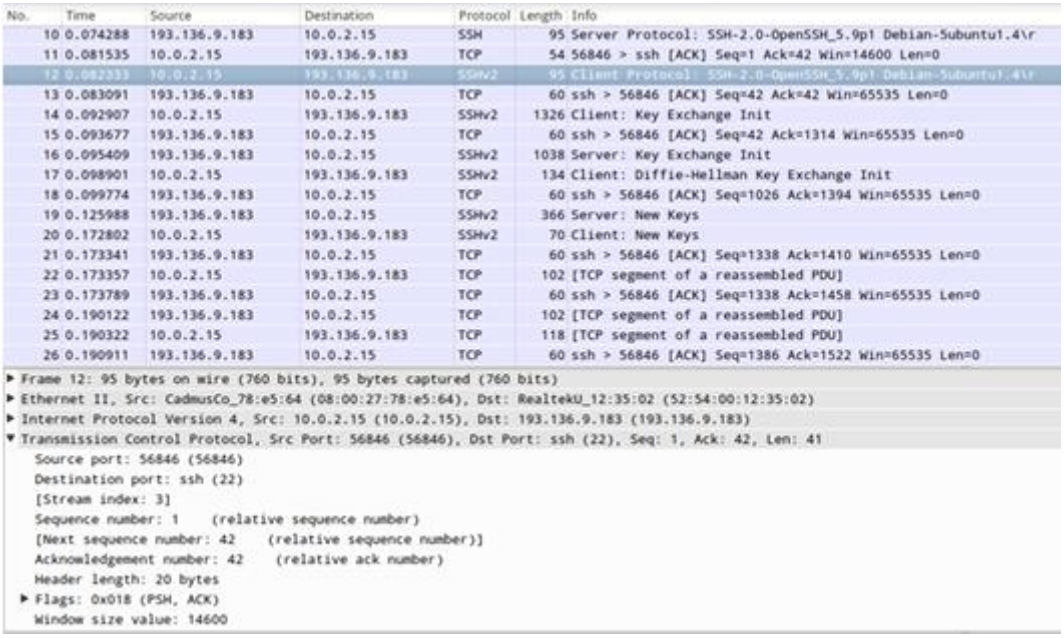
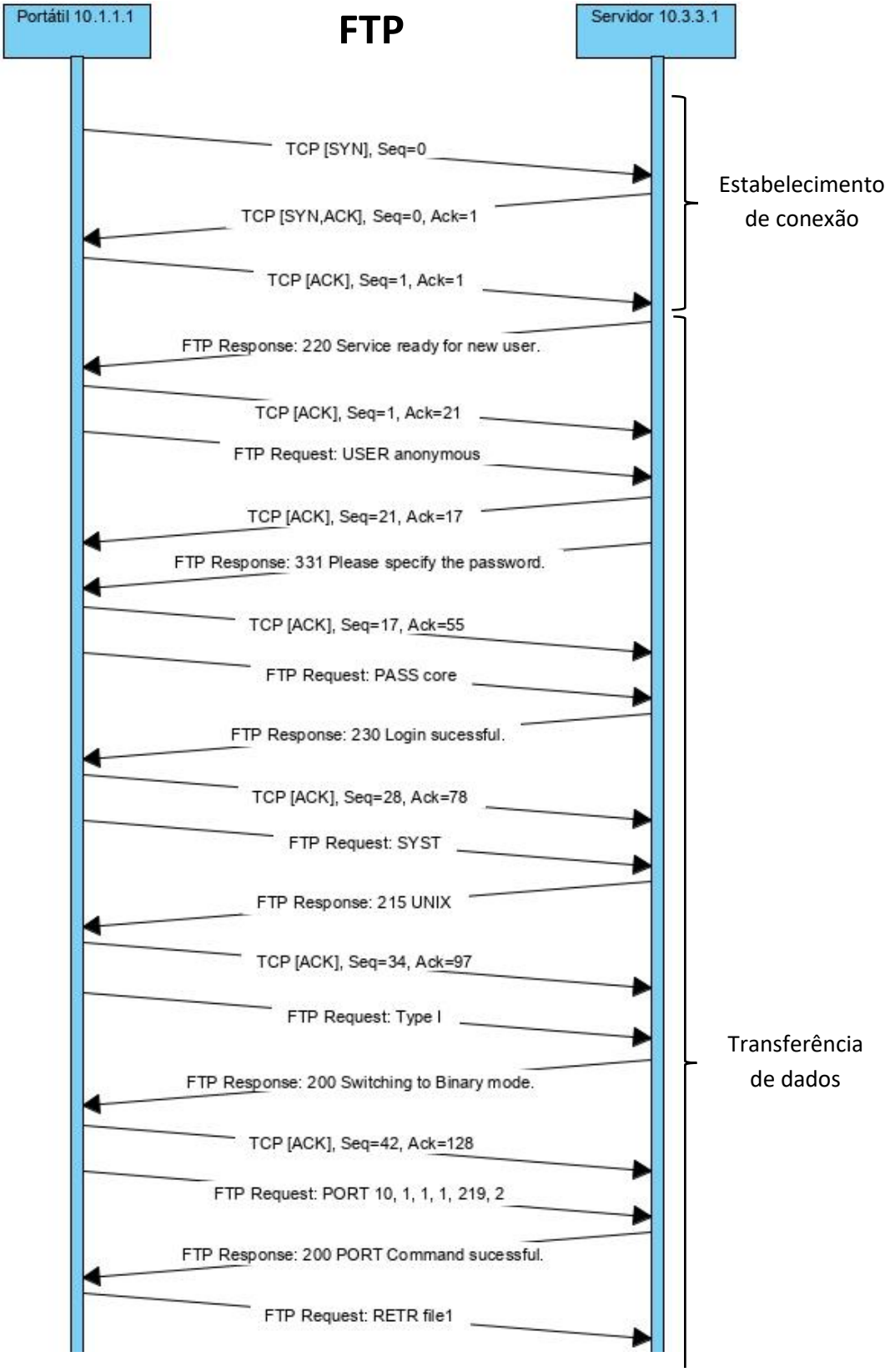
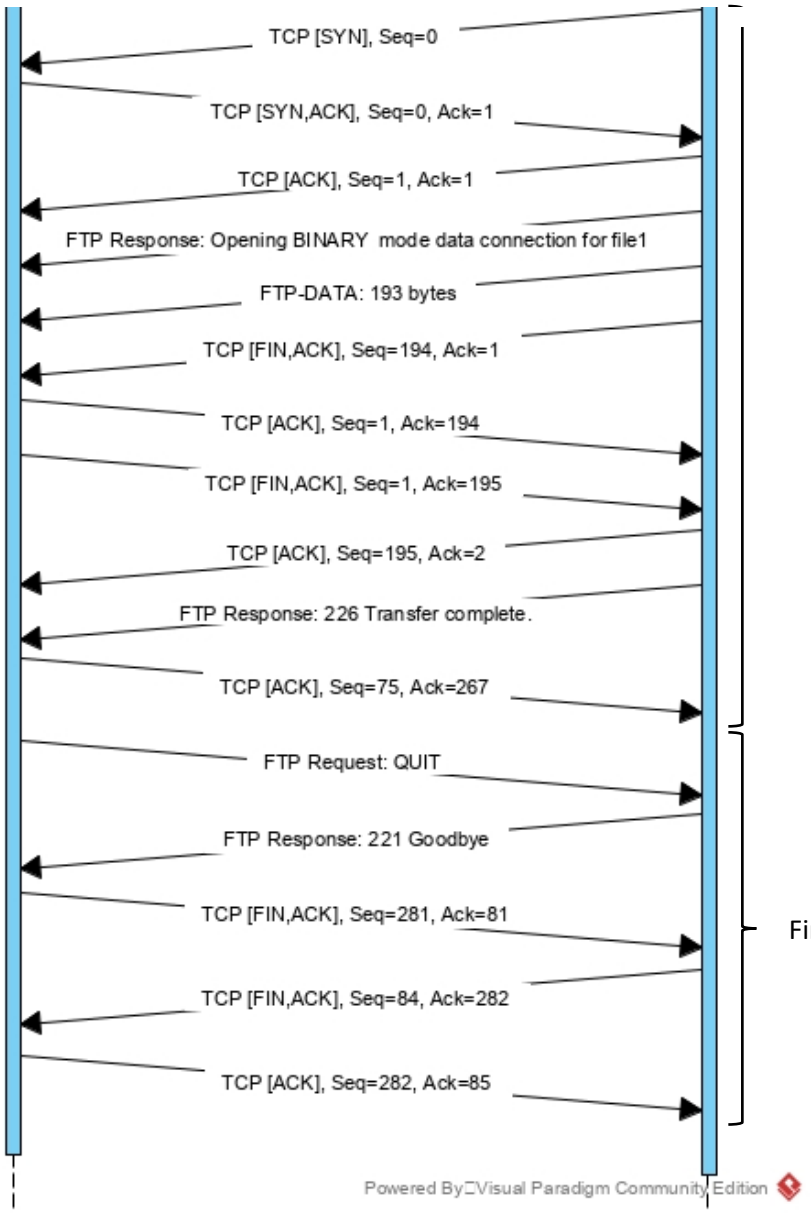
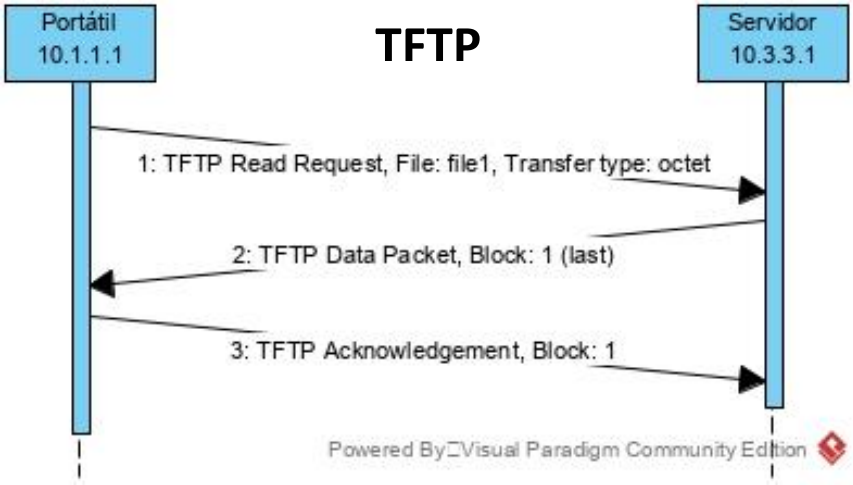



Figura 8 - Ssh.

**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.**




TCP [SYN], Seq=0
TCP [SYN,ACK], Seq=0, Ack=1
TCP [ACK], Seq=1, Ack=1
FTP Response: Opening BINARY mode data connection for file1
FTP-DATA: 193 bytes
TCP [FIN,ACK], Seq=194, Ack=1
TCP [ACK], Seq=1, Ack=194
TCP [FIN,ACK], Seq=1, Ack=195
TCP [ACK], Seq=195, Ack=2
FTP Response: 226 Transfer complete.
TCP [ACK], Seq=75, Ack=267
FTP Request: QUIT
FTP Response: 221 Goodbye
TCP [FIN,ACK], Seq=281, Ack=81
TCP [FIN,ACK], Seq=84, Ack=282
TCP [ACK], Seq=282, Ack=85
Fim de conexão
Powered By Visual Paradigm Community Edition


Portátil
10.1.1.1
TFTP
Servidor
10.3.3.1
1: TFTP Read Request, File: file1, Transfer type: octet
2: TFTP Data Packet, Block: 1 (last)
3: TFTP Acknowledgement, Block: 1
Powered By Visual Paradigm Community Edition

**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.**

**STFP** - protocolo que implementa encriptação (através de SSH) de conexão. Todas as mensagens são estabelecidas entre um cliente e um servidor, com processos de autenticação, e encriptadas para essa mesma conexão, o que assegura um elevado nível de segurança. Porém, todo este overhead de encriptação e afins do SSH leva a uma baixa eficiência e grande complexidade. Utiliza o TCP como protocolo de camada de transporte.

**FTP** - protocolo simples, também de cliente-servidor, mas a conexão não é encriptada, apesar de haver um processo de autenticação. Como não é encriptada, é relativamente vulnerável em termos de segurança porque todos os dados de conexão estão diretamente nos pacotes enviados, pelo que são fáceis de obter por parte de alguém que esteja à escuta na rede. É bastante eficiente (foi o protocolo que transferiu o ficheiro mais rapidamente, dos que usam TCP), mas em termos de complexidade a situação inverte-se, visto que há bastantes "handshakes" para estabelecer estas conexões, o que o torna mais difícil de implementar. Utiliza TCP como protocolo de camada de transporte.

**TFPT** - também um protocolo simples que não implementa mecanismos de autenticação nem de encriptação, pelo que é pouco seguro e complexo. Contudo, é bastante eficiente (foi o protocolo mais rápido de entre os que testamos) e usa UDP como protocolo de camada de transporte.

**HTTP** - muito semelhante ao FTP. Utiliza mecanismos de "handshake" e autenticação, tornando-se relativamente pouco complexo, embora pouco seguro (mais tarde, surgiu o HTTPS – HTTP Secure). É muito eficiente e utiliza TCP como protocolo de camada de transporte.

**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).**

Para aplicações que trabalhem sobre TCP, temos a garantia de que a nível de rede, os pacotes chegam na ordem correta e a sua receção/envio é sempre confirmada por ambas as partes, garantindo assim que não há perda de dados. Porém, todo este processo de trocas de mensagens leva a que este processo seja muito complexo e a que sejam enviados demasiados pacotes de controlo. Evidentemente, se uma rede possui falhas/duplicações, todo este processo vai tornar-se ainda mais complexo e demorar mais tempo (pelos testes executados, a transferência do *file2* demorou 1.88s no protátil Alfa e 0.64s no portátil 1, usando FTP).

Para aplicações que trabalhem sobre UDP, já não podemos ter a certeza de que todos os pacotes chegam ao destino e não há retransmissões. Logo, cabe à aplicação decidir como agir nestes casos. No entanto, em troca desta escassez de controlo, é garantida a rapidez no envio de pacotes, bem como uma menor carga sobre o mecanismo de transporte e, consequentemente, reduz o congestionamento da rede.

Concluímos, portanto, que ambas as opções são perfeitamente válidas para obter uma aplicação fiável. Distinguem-se apenas na maneira e circunstâncias em que são implementadas - UDP é típico para *streams* de vídeo (não faz sentido pensar em retransmissão caso se perca uma *frame*, porque implicaria atrasos na transmissão ao vivo), enquanto TCP é mais habitual em transferências de ficheiros.

# 2. Conclusão

Terminado este trabalho prático, sentimos que conseguimos, num contexto geral, atingir os objetivos pretendidos, estabelecendo uma ponte entre a matéria lecionada nas aulas teóricas e as aplicações práticas dos conceitos e temas propostos.

Percebemos as diferenças entre os protocolos de transporte UDP e TCP, assim como as vantagens e desvantagens de cada um. Para além disto, trabalhamos ainda com aplicações que faziam uso de ambos protocolos, concluindo precisamente que tanto um como outro têm a sua aplicação e contexto na vida real.