Universidade do Minho
Licenciatura em Engenharia Informática

UC de Comunicações por Computador

Ano Letivo de 2023/2024

Trabalho Prático Nº 1 - Protocolos da Camada de Transporte

Rodrigo Monteiro, Diogo Abreu, e Miguel Gramoso - Grupo 9 - PL7

e-mail: {a100706,a100646,a100835}@alunos.uminho.pt

Índice

Índice de Figuras

Índice de Tabelas

1 Questões (Parte I)

1)

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

Ao testar a conectividade entre o `Portatil1` e o `Servidor1` através do comando *ping*, observamos que não ocorreu perda de pacotes, i.e. foram enviadas 20 ICMP requests e foram recebidas 20 ICMP replies, e que a rota consistiu de links com capacidade ilimitada e capacidade de 1.0 Gbps com delay de 100 µs entre os switches e os routers, o que contribuiu para uma conexão sem perdas e duplicados - apesar disso, fatores como o congestionamento de rede poderiam interferir.

```
--- 10.4.4.1 ping statistics ---
20 packets transmitted, 20 received, 0% packet loss, time 19390ms
rtt min/avg/max/mdev = 0.365/1.144/10.453/2.149 ms
```

Figura 1 - Resultados do comando *ping* (Portatil1)

Por outro lado, ao testar a conectividade entre o `PC1` e o `Servidor1`, também através do comando *ping*, observamos que ocorreu 20% de perda de pacotes (apenas 16 recebidos) e 3 duplicados, cuja causa está no link entre o Router4 e o Switch2 que possui uma largura de banda de 10 Mbps, um delay de 4 ms, uma probabilidade de perda de 10% e uma probabilidade de 5% de duplicação.

```
--- 10.4.4.1 ping statistics ---
20 packets transmitted, 16 received, +1 duplicates, 20% packet loss, time 19115m
s
rtt min/avg/max/mdev = 5.893/11.892/26.321/5.963 ms
```

Figura 2 - Resultados do comando *ping* (PC1)

Por este motivo, os protocolos da camada de transporte usados, o SFTP, FTP, TFTP e HTTP, apesar de possuírem métodos para lidar com a perda de pacotes e duplicados, acabam por fornecer um serviço mais lento à camada aplicacional. Por exemplo, quando se transferiu um ficheiro do `Servidor1` para o `PC1` utilizando SFTP (SSH File Transfer Protocol), a velocidade de transferência foi de 9.0 KB/s, enquanto que, quando se transferiu um ficheiro do `Servidor1` para o Portatil1, a velocidade de transferência foi de 114.6 KB/s; e, quando se transferiu um ficheiro do `Servidor1` para o `PC1` utilizando FTP (File Transfer Protocol), a velocidade de transferência foi de 623.22 KB/s, enquanto que, quando se transferiu um ficheiro do `Servidor1` para o Portatil1, a velocidade de transferência foi de 3.56 MB/s.

Portanto, observa-se que existe uma diferença significativa entre velocidades de transferência do ficheiro devido às perdas de pacotes num dos links.

```
root@Portatil1:/tmp/pycore.35365/Portatil1.conf# sftp core@10.4.4.1
The authenticity of host '10.4.4.1 (10.4.4.1)' can't be established.
RSA key fingerprint is SHA256:ppBOIvrbq+h7uCOAxn16XIs1ER28WP0kDAxGjO1jFhU.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '10.4.4.1' (RSA) to the list of known hosts.
core@10.4.4.1's password:
Connected to 10.4.4.1.
sftp> pwd
Remote working directory: /home/core
sftp> cd /srv/ftp
sftp> dir
file1  file2
sftp> get file1
Fetching /srv/ftp/file1 to file1
/srv/ftp/file1                                100%  224   114.6KB/s   00:00
sftp> quit
```

Figura 3 - Conexão SFTP e resultados da transferência de file1 por SFTP no Portatil1

```
root@PC1:/tmp/pycore.41989/PC1.conf# rm /root/.ssh/known_hosts
root@PC1:/tmp/pycore.41989/PC1.conf# sftp core@10.4.4.1
The authenticity of host '10.4.4.1 (10.4.4.1)' can't be established.
RSA key fingerprint is SHA256:DgwWmfmOrOVbq7ipETlxWyCtOMDp+OuhVkDucQzLpeO.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '10.4.4.1' (RSA) to the list of known hosts.
core@10.4.4.1's password:
Connected to 10.4.4.1.
sftp> pwd
Remote working directory: /home/core
sftp> cd /srv
sftp> cd /srv/ft
sftp> cd /srv/ftp/
sftp> dir
file1  file2
sftp> get file1
sftp> get file1
Fetching /srv/ftp/file1 to file1
/srv/ftp/file1                                100%  224     9.0KB/s   00:00
sftp> quit
```

Figura 4 - Conexão SFTP e resultados da transferência de file1 por SFTP no PC1

```
root@Portatil1:/tmp/pycore.41989/Portatil1.conf# ftp 10.4.4.1
Connected to 10.4.4.1.
220 (vsFTPd 3.0.3)
Name (10.4.4.1:root): anonymous
331 Please specify the password.
Password:
230 Login successful.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> status
Connected to 10.4.4.1.
No proxy connection.
Connecting using address family: any.
Mode: stream; Type: binary; Form: non-print; Structure: file
Verbose: on; Bell: off; Prompting: on; Globbing: on
Store unique: off; Receive unique: off
Case: off; CR stripping: on
Quote control characters: on
Ntrans: off
Nmap: off
Hash mark printing: off; Use of PORT cmds: on
Tick counter printing: off
ftp> pwd
257 "/" is the current directory
ftp> dir
200 PORT command successful. Consider using PASV.
150 Here comes the directory listing.
-rw-r--r--    1 0        0             224 Sep 18 11:45 file1
-rwxr-xr-x    1 0        0          142144 Sep 18 11:45 file2
226 Directory send OK.
ftp> get file1
local: file1 remote: file1
200 PORT command successful. Consider using PASV.
150 Opening BINARY mode data connection for file1 (224 bytes).
226 Transfer complete.
224 bytes received in 0.00 secs (3.5604 MB/s)
ftp> quit
221 Goodbye.
```

```
root@PC1:/tmp/pycore.41989/PC1.conf# ftp 10.4.4.1
Connected to 10.4.4.1.
220 (vsFTPd 3.0.3)
Name (10.4.4.1:root): anonymous
331 Please specify the password.
Password:
230 Login successful.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> status
Connected to 10.4.4.1.
No proxy connection.
Connecting using address family: any.
Mode: stream; Type: binary; Form: non-print; Structure: file
Verbose: on; Bell: off; Prompting: on; Globbing: on
Store unique: off; Receive unique: off
Case: off; CR stripping: on
Quote control characters: on
Ntrans: off
Nmap: off
Hash mark printing: off; Use of PORT cmds: on
Tick counter printing: off
ftp> pwd
257 "/" is the current directory
ftp> dir
200 PORT command successful. Consider using PASV.
150 Here comes the directory listing.
-rw-r--r--    1 0        0             224 Sep 18 11:45 file1
-rwxr-xr-x    1 0        0          142144 Sep 18 11:45 file2
226 Directory send OK.
ftp> get file1
local: file1 remote: file1
200 PORT command successful. Consider using PASV.
150 Opening BINARY mode data connection for file1 (224 bytes).
226 Transfer complete.
224 bytes received in 0.00 secs (623.2194 kB/s)
ftp> quit
221 Goodbye.
```

Figura 5 e 6 - Conexão FTP e resultados da transferência de file1 por FTP no Portatil1 e no PC1

2)

Obtenha a partir do wireshark, ou desenhe manualmente, um diagrama temporal para a transferência de `file1` por FTP. Foque-se apenas na transferência de dados [ftp-data] e não na conexão de controlo, pois 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 quer nos dados como nas confirmações.

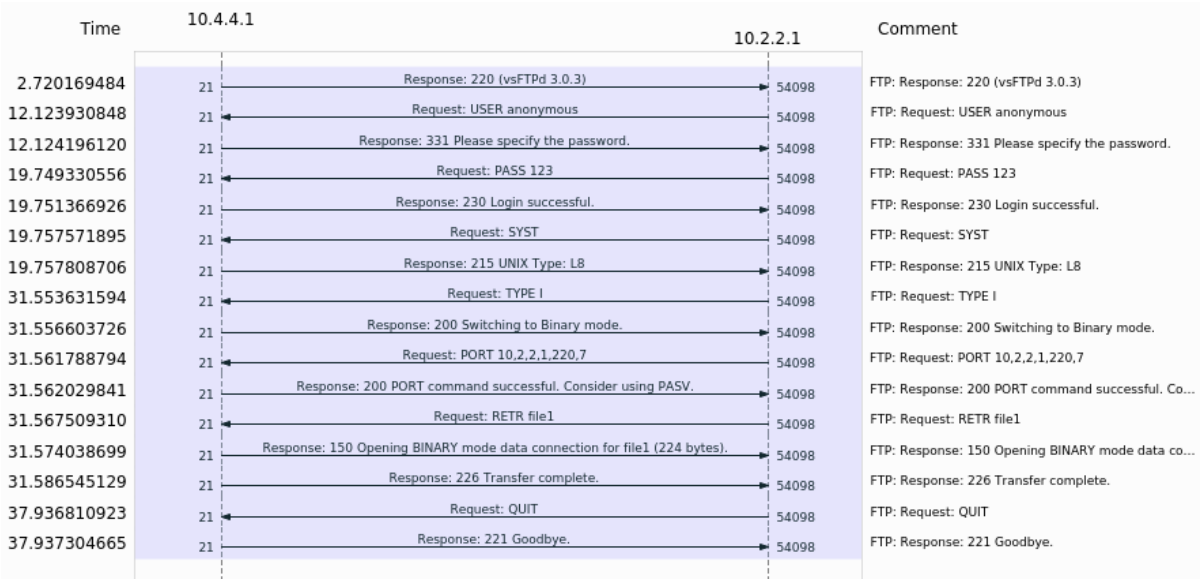

Figura 7 - Flow graph da transferência de file1 por FTP

O primeiro pacote é relativo ao início da conexão, uma resposta com código 220 é enviada quando um novo utilizador se conecta ao servidor FTP, para indicar que este está disponível para o novo cliente.
Os quatro pacotes seguintes são relativos ao processo de login:

- é enviado o username;
- recebido um pedido de password;
- enviada a password;
- e, por fim, recebida uma confirmação de login bem sucedido.

Os próximos nove pacotes são relativos à transferência de dados:

- é solicitada informação sobre o sistema operativo em que o servidor FTP opera;
- a informação é recebida;
- é enviado um pedido de "TYPE I" que indica transferência em modo binário;
- o servidor confirma a mudança para modo binário;
- é enviado um PORT command e este é recebido com sucesso (código 200);
- de seguida, é enviado um comando RETR, um pedido de transferência do ficheiro `file1`;
- é aberta a conexão para a transferência do ficheiro de 224 bytes
- e, por fim, é recebida uma confirmação da conclusão da transferência.

Os últimos dois pacotes são relativos ao fim de conexão:

- é enviado um pedido de fim de conexão,
- e é recebida uma confirmação com o código 221.

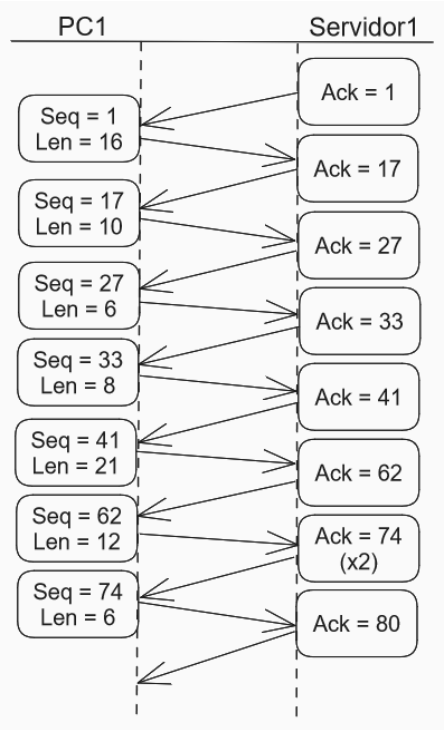

Figura 9 - Progressão dos números de sequência e de confirmação na transferência de file1 por FTP no PC1

O ack é inicialmente igual a 1, possivelmente devido à presença de um pacote anteriormente recebido com uma flag SYN. Este número aumenta progressivamente nos próximos pacotes, pois representa a quantidade de dados recebidos. O número de sequência também aumenta para indicar a posição do primeiro byte que envia. Assim, os números de seq e ack do TCP são importantes para possibilitar uma transferência de dados ordenada e fiável.

> *Nota:* os números seq e ack, inicialmente, são gerados aleatoriamente por motivos de segurança, mas, nesta representação, são utilizados números relativos, tal como aparecem no Wireshark.

3)

Obtenha a partir do wireshark, ou desenhe manualmente, um diagrama temporal para a transferência de `file1` por TFTP. 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 quer nos dados como nas confirmações

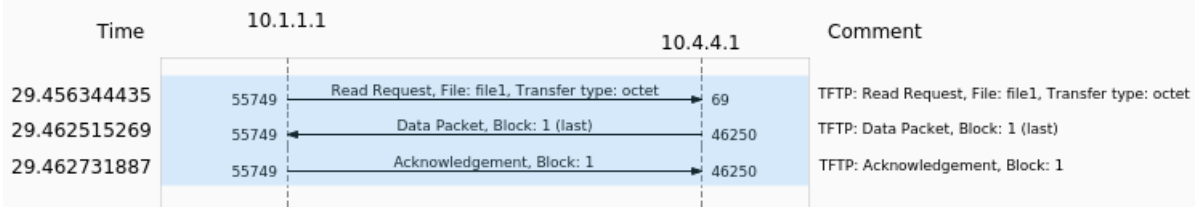

Figura 10 - Flow graph da transferência de file1 por TFTP

O TFTP (Trivial File Transfer Protocol) não tem como protocolo subjacente o TCP, mas sim o UDP (User Datagram Protocol), tornando o processo mais rápido, e stateless, e cujos pacotes têm overhead muito baixo (64 bits); no entanto, não oferece controlo de fluxo, retransmissão, reordenação ou garantia de entrega. Apesar disso, o UDP fornece multiplexagem, e, portanto, existe a seguinte indicação da porta de destino e da porta de origem:

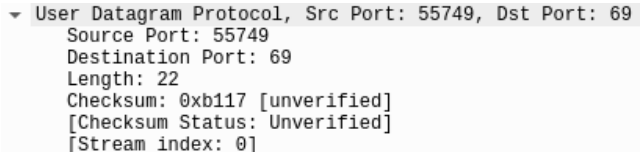

Figura 11 - Dados UDP da read request

O primeiro pacote enviado pelo `Portatil1` ao `Servidor1` é uma read request, um pedido de transferência de tipo octet (toda a informação é tratada como *raw 8-bit bytes*) do `file1`. O segundo é enviado pelo `Servidor1` para o `Portatil1` e contém o único bloco de dados:

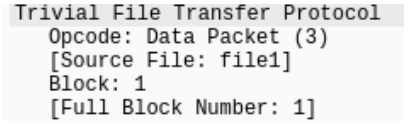

Figura 12 - Dados TFTP do data packet enviado pelo Servidor1

O terceiro é enviado do `Portatil1` para o `Servidor1`, desta vez com porta de destino igual à porta de origem do pacote anterior, que envia uma confirmação.

4)

Compare sucintamente as quatro aplicações de transferência de ficheiros que usou nos seguintes pontos (i) uso da camada de transporte; (ii) eficiência; (iii) complexidade; (iv) segurança.

	Uso da camada de transporte	Eficiência
SFTP	Utiliza TCP.	É eficiente no sentido de ser platform-independent, mas é importante ter em conta que a encriptação introduz mais overhead, e que os dados encriptados são muitas vezes menos comprimíveis levando a um desempenho mais lento.
FTP	Utiliza TCP.	Possui alguma rapidez na transferência de dados, mas este protocolo pode tornar-se progressivamente lento devido à ocorrência de pacotes perdidos (originando pedidos e confirmações repetidas). Este protocolo é, como anteriormente observado nas figuras 3 e 5, geralmente mais rápido do que o SFTP.
TFTP	Utiliza UDP.	Utiliza o protocolo UDP, o que significa que não assegura a integridade dos dados, a retransmissão de pacotes perdidos ou controlo de fluxo. Embora esta simplicidade o torne leve, pode resultar em transferências menos fiáveis e mais lentas, especialmente em redes instáveis ou congestionadas.
HTTP /1.1	Utiliza TCP.	Apresenta vantagens como multiplexagem e compressão de headers, melhorando a velocidade de transferência, mas algumas das suas características de design podem interagir negativamente com o TCP [4]. Apesar disso, na transferência do file1, este protocolo foi o que mostrou a maior velocidade de transferência (Figura 13).

Tabela 1 - Uso da camada de transporte e eficiência dos protocolos testados

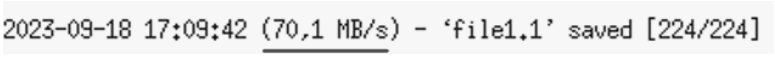

Figura 13 - Velocidade de transferência do file1 por HTTP no Portatil1

	Complexidade	Segurança
SFTP	Permite uma série de operações em ficheiros remotos. Para além disso, utiliza SSH, o que acrescenta uma camada adicional de complexidade em comparação a protocolos não seguros de transferência de ficheiros.	Utiliza o protocolo SSH-2 (ou SSH-1), o que permite encriptação end-to-end, integridade de dados, e mecanismos de autenticação, tornando o SFTP bastante seguro.
FTP	A sua configuração é relativamente simples, tornando-o acessível a utilizadores com diferentes níveis de conhecimento. A complexidade pode aumentar caso se queira aumentar a sua segurança (FTP-SLS).	Não inclui funcionalidades de segurança incorporadas, transmitindo os dados em *plain text*, incluindo nomes de utilizador e palavras-passe, como foi evidenciado anteriormente (Figura 7).
TFTP	Foi concebido para ser simples e de fácil implementação, por isso, não possui a maioria das funcionalidades avançadas oferecidas por protocolos de transferência de ficheiros mais robustos (não consegue listar, apagar ou renomear ficheiros, etc.).	Não oferece quaisquer mecanismos de segurança incorporados, como encriptação ou autenticação. Como resultado, os dados transferidos através do TFTP são enviados em *plain text*, tornando-os vulneráveis, tal como acontece com o FTP.
HTTP /1.1	Por ser um protocolo versátil, torna-se também, portanto, mais complexo. Para além disso, o seu overhead pode levar a um grande consumo de largura de banda, especialmente para pedidos e respostas pequenas.	Não tem recursos sólidos de segurança, uma vez que os dados são transmitidos em *plain text*.

Tabela 2 - Complexidade e segurança dos protocolos testados

2 Questões (Parte II)

1)

Com base no trabalho realizado, tanto na parte I como na parte II, identifique para cada aplicação executada, 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)
wget (Figura 14)	HTTP	TCP	80	Pacote inicial com overhead de 40 bytes, e restantes com 20 bytes.
ssh, sftp (Figura 15)	SSHv2	TCP	80	20
ftp (Figura 16)	FTP	TCP	21	20
tftp (Figura 17)	TFTP	UDP	69	8
telnet (Figura 18)	TELNET	TCP	23	20
nslookup (Figura 19)	DNS	UDP	53	8
ping (Figura 20)	DNS	UDP	53	8
traceroute (Figura 21)	NTP	UDP	123	8

Tabela 3 - Características das aplicações executadas

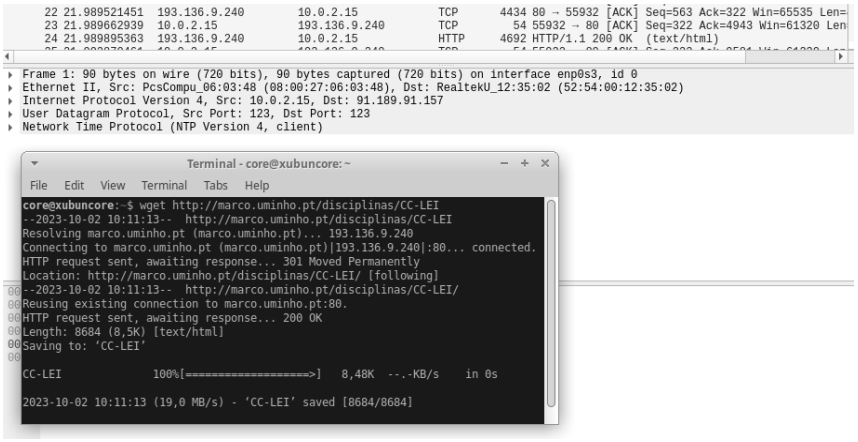

Figura 14 - Transferência de um ficheiro por HTTP (http://marco.uminho.pt/disciplinas/CC-LEI)

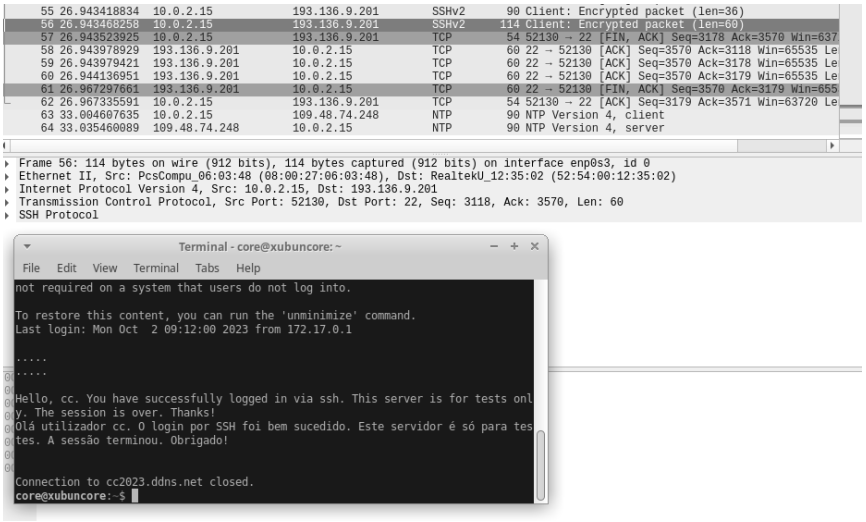

Figura 15 - Transferência de um ficheiro por SSH (cc2023.ddns.net)

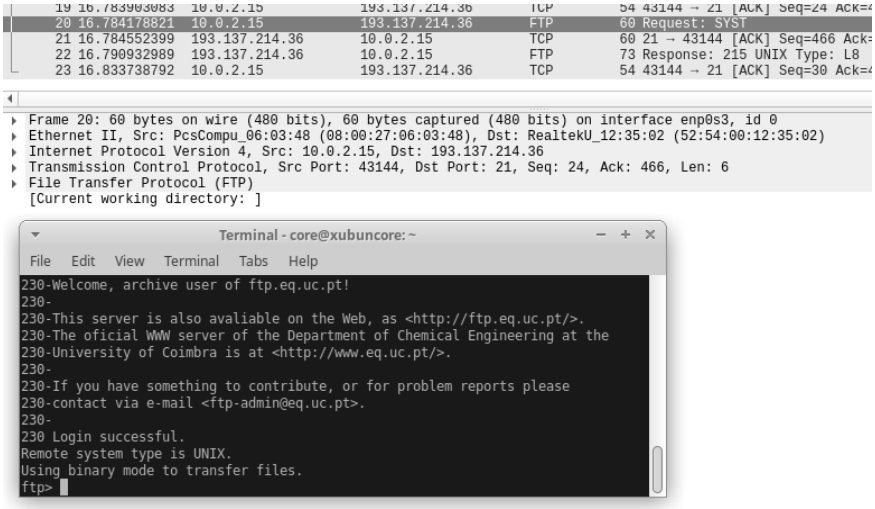

Figura 16 - Transferência de um ficheiro por FTP (ftp.eq.uc.pt)

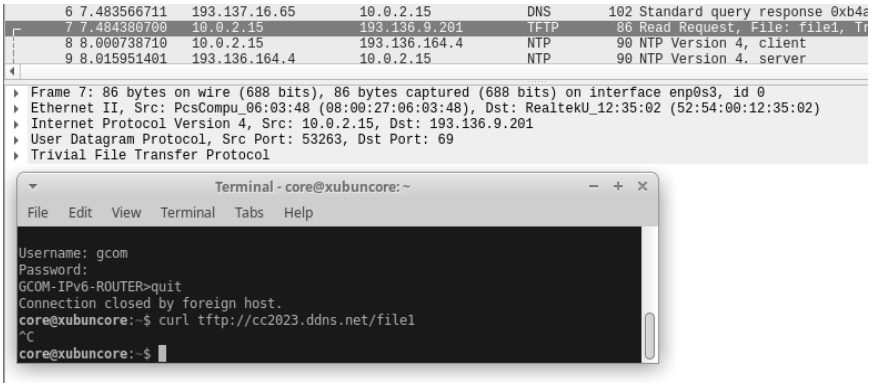

Figura 17 - Transferência de um ficheiro por TFTP (tftp://cc2023.ddns.net/file1)

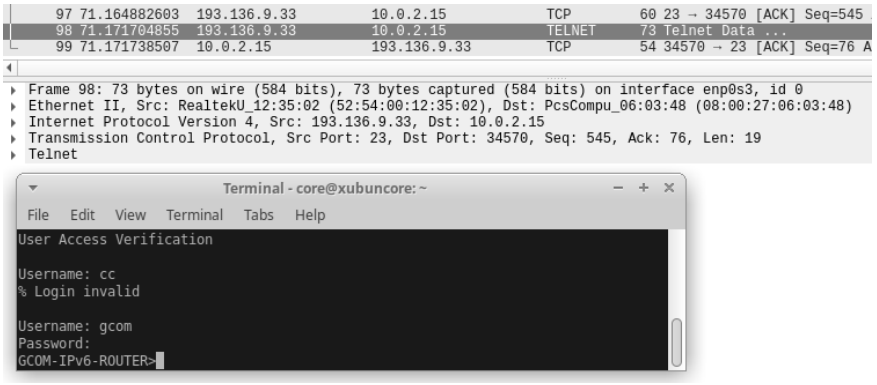

Figura 18 - Resultados do telnet (193.136.9.33)

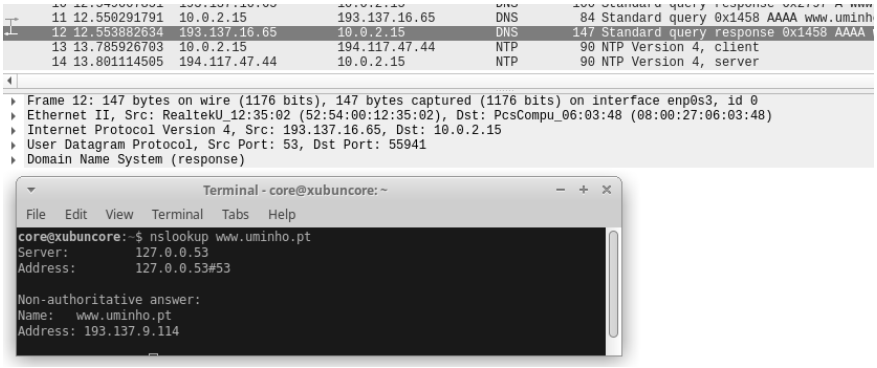

Figura 19 - Resultados do nslookup (www.uminho.pt)

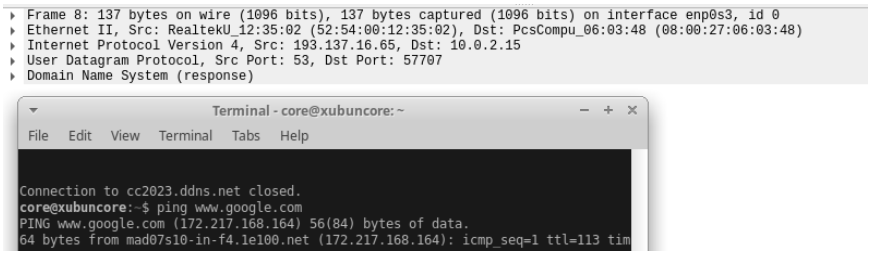

Figura 20 - Resultados do ping (google.com)

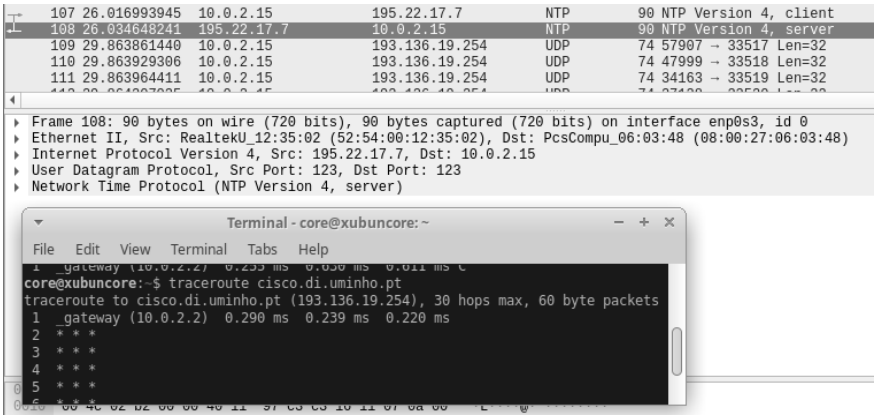

Figura 21 - Resultados do traceroute (cisco.di.uminho.pt)

3 Conclusões

Em síntese, este relatório discute e avalia o desempenho de diferentes protocolos na transferência de um ficheiro, focando um pouco em alguns protocolos como o FTP e o TFTP, e conclui com uma comparação entre os quatro protocolos testados. Deste modo, através da compreensão teórica dos protocolos e diante das conclusões expostas, verificamos a importância da escolha adequada de protocolos de transporte com base nos requisitos e no contexto de cada aplicação.

4 Bibliografia

1. Madpacket - TCP Sequence and Acknowledgement Numbers Explained: https://madpackets.com/2018/04/25/tcp-sequence-and-acknowledgement-numbers-explained/ (acessado em 21 de setembro, 2023)
2. IBM - FTP Performance Considerations: https://www.ibm.com/support/pages/ftp-performance-considerations (acessado em 23 de setembro, 2023)
3. Wikipedia - Trivial File Transfer Protocol: https://en.wikipedia.org/wiki/Trivial_File_Transfer_Protocol (acessado em 23 de setembro de 2023)
4. Simon E Spero - Analysis of HTTP Performance problems: https://www.w3.org/Protocols/HTTP-NG/http-prob.html (acessado em 23 de setembro, 2023)
5. HTTP/2 vs. HTTP/1.1: How do they affect web performance? : https://www.cloudflare.com/learning/performance/http2-vs-http1.1 (acessado em 23 de setembro, 2023)