



Universidade do Minho
Escola de Engenharia

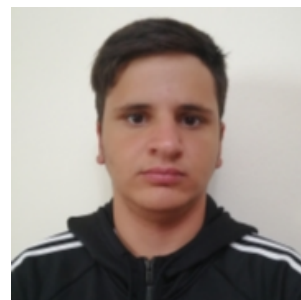
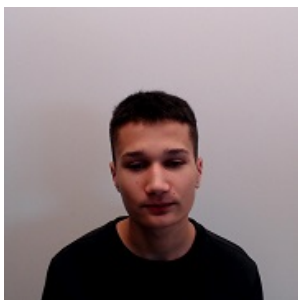
Comunicações por Computador

PL7 Grupo 4

Benjamim Meleiro Rodrigues A93323

Lara Beatriz Pinto Ferreira A95454

João Pedro Machado Ribeiro A95719



2 de Outubro de 2023

Índice

1	Parte I	2
1.1	Questão 1	2
1.2	Questão 2	4
1.3	Questão 3	6
1.4	Questão 4	7
2	Parte II	9
2.1	Questão 1	9
3	Conclusão	16

1 Parte I

1.1 Questão 1

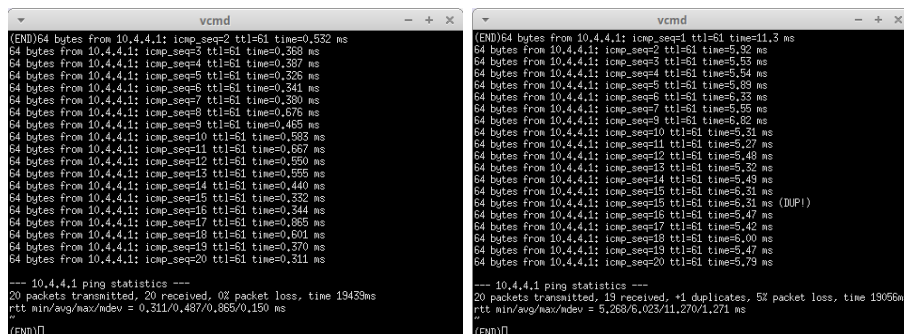
De que forma as perdas e duplicações de pacotes afetaram o desempenho das aplicações? Que camada lidou com as perdas e duplicações: transporte ou aplicação? Responda com base nas experiências feitas e nos resultados observados.

- **STFTP, FTP e HTTP** - usam o protocolo TCP na camada de transporte, sendo este protocolo responsável por lidar com perdas e duplicações de pacotes.

O TCP foi concebido para proporcionar uma entrega fiável e ordenada de dados. Quando ocorrem perdas de pacotes, o TCP inicia retransmissões. O mesmo acontece quando o “receiver” recebe pacotes fora de ordem, avisando o “sender” de forma a que este retransmita os pacotes. No entanto, esta duplicação de pacotes leva a um aumento de congestionamento na rede e redução da disponibilidade de largura de banda. É fácil concluir que o mecanismo de retransmissão é indispensável para garantir a integridade dos dados, mas leva ao aumento do débito e da latência, degradando o desempenho das aplicações.

- **TFTP** - usa o protocolo UDP.

O protocolo UDP não possui mecanismos incorporados para controlo de erros, não garantindo, portanto, o tratamento de perdas e duplicações de pacotes. A perda e duplicação de pacotes no UDP podem resultar em problemas de integridade dos dados e interrupções em aplicações em tempo real. As aplicações que utilizam o UDP devem ser concebidas para lidar com estes problemas, utilizando mecanismos de tratamento de erros e recuperação, se necessário, para proporcionar de forma eficaz uma experiência satisfatória ao utilizador. Sendo assim podemos concluir que, no caso da aplicação TFTP, é a camada de aplicação encarregada de tratar destes problemas.



```
vcmd                                     vcmd
(END)64 bytes from 10.4.4.1: icmp_seq=2 ttl=61 time=0.532 ms
64 bytes from 10.4.4.1: icmp_seq=3 ttl=61 time=0.387 ms
64 bytes from 10.4.4.1: icmp_seq=4 ttl=61 time=0.387 ms
64 bytes from 10.4.4.1: icmp_seq=5 ttl=61 time=0.325 ms
64 bytes from 10.4.4.1: icmp_seq=6 ttl=61 time=0.341 ms
64 bytes from 10.4.4.1: icmp_seq=7 ttl=61 time=0.380 ms
64 bytes from 10.4.4.1: icmp_seq=8 ttl=61 time=0.676 ms
64 bytes from 10.4.4.1: icmp_seq=9 ttl=61 time=0.465 ms
64 bytes from 10.4.4.1: icmp_seq=10 ttl=61 time=0.593 ms
64 bytes from 10.4.4.1: icmp_seq=11 ttl=61 time=0.667 ms
64 bytes from 10.4.4.1: icmp_seq=12 ttl=61 time=0.550 ms
64 bytes from 10.4.4.1: icmp_seq=13 ttl=61 time=0.555 ms
64 bytes from 10.4.4.1: icmp_seq=14 ttl=61 time=0.440 ms
64 bytes from 10.4.4.1: icmp_seq=15 ttl=61 time=0.332 ms
64 bytes from 10.4.4.1: icmp_seq=16 ttl=61 time=0.344 ms
64 bytes from 10.4.4.1: icmp_seq=17 ttl=61 time=0.865 ms
64 bytes from 10.4.4.1: icmp_seq=18 ttl=61 time=0.801 ms
64 bytes from 10.4.4.1: icmp_seq=19 ttl=61 time=0.370 ms
64 bytes from 10.4.4.1: icmp_seq=20 ttl=61 time=0.311 ms
--- 10.4.4.1 ping statistics ---
20 packets transmitted, 20 received, 0% packet loss, time 1943ms
rtt min/avg/max/med = 0.311/0.487/0.865/0.150 ms
(END)

(END)64 bytes from 10.4.4.1: icmp_seq=1 ttl=61 time=11.3 ms
64 bytes from 10.4.4.1: icmp_seq=2 ttl=61 time=5.32 ms
64 bytes from 10.4.4.1: icmp_seq=3 ttl=61 time=5.53 ms
64 bytes from 10.4.4.1: icmp_seq=4 ttl=61 time=5.54 ms
64 bytes from 10.4.4.1: icmp_seq=5 ttl=61 time=5.68 ms
64 bytes from 10.4.4.1: icmp_seq=6 ttl=61 time=5.33 ms
64 bytes from 10.4.4.1: icmp_seq=7 ttl=61 time=5.55 ms
64 bytes from 10.4.4.1: icmp_seq=8 ttl=61 time=5.32 ms
64 bytes from 10.4.4.1: icmp_seq=9 ttl=61 time=5.31 ms
64 bytes from 10.4.4.1: icmp_seq=10 ttl=61 time=5.27 ms
64 bytes from 10.4.4.1: icmp_seq=11 ttl=61 time=5.48 ms
64 bytes from 10.4.4.1: icmp_seq=12 ttl=61 time=5.32 ms
64 bytes from 10.4.4.1: icmp_seq=13 ttl=61 time=5.49 ms
64 bytes from 10.4.4.1: icmp_seq=14 ttl=61 time=5.31 ms
64 bytes from 10.4.4.1: icmp_seq=15 ttl=61 time=5.31 ms
64 bytes from 10.4.4.1: icmp_seq=16 ttl=61 time=5.47 ms
64 bytes from 10.4.4.1: icmp_seq=17 ttl=61 time=5.42 ms
64 bytes from 10.4.4.1: icmp_seq=18 ttl=61 time=5.00 ms
64 bytes from 10.4.4.1: icmp_seq=19 ttl=61 time=5.47 ms
64 bytes from 10.4.4.1: icmp_seq=20 ttl=61 time=5.79 ms
--- 10.4.4.1 ping statistics ---
20 packets transmitted, 19 received, 5% packet loss, time 1905ms
rtt min/avg/max/med = 5.280/5.023/11.270/4.271 ms
(DUP!)
(END)
```

Figura 1: Portatill na bash da esquerda — PC1 na bash da direita

No.	Time	Source	Destination	Protocol	Length	Info
188	179.493084083	10.2.2.1	10.4.4.1	TCP	74	53008 → 22 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 T...
189	179.493263884	10.4.4.1	10.2.2.1	TCP	74	22 → 53008 [ACK] Seq=1 Ack=0 Win=5160 Len=0 MSS=1460 S...
190	179.498424527	10.2.2.1	10.4.4.1	TCP	66	53008 → 22 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=519404655 ...
191	179.498593988	10.2.2.1	10.4.4.1	SSHv2	107	Client: Protocol (SSH-2.0-OpenSSH_8.2p1 Ubuntu-4ubuntu0.9)
192	179.503074400	10.4.4.1	10.2.2.1	TCP	66	22 → 53008 [ACK] Seq=1 Ack=42 Win=65152 Len=0 TSval=515530262.
193	179.512696305	10.4.4.1	10.2.2.1	SSHv2	107	Server: Protocol (SSH-2.0-OpenSSH_8.2p1 Ubuntu-4ubuntu0.9)
194	179.517676602	10.2.2.1	10.4.4.1	TCP	66	53008 → 22 [ACK] Seq=42 Ack=42 Win=64256 Len=0 TSval=51948467...
195	179.518245544	10.2.2.1	10.4.4.1	TCP	66	53008 → 22 [ACK] Seq=42 Ack=42 Win=64256 Len=0 TSval=51948467...
196	179.518024500	10.4.4.1	10.2.2.1	SSHv2	1090	Server: Key Exchange Init
197	179.520997348	10.2.2.1	10.4.4.1	TCP	1514	53008 → 22 [ACK] Seq=42 Ack=42 Win=64256 Len=1448 TSval=51948...
198	179.520998391	10.2.2.1	10.4.4.1	SSHv2	130	Client: Key Exchange Init
199	179.521183212	10.4.4.1	10.2.2.1	TCP	66	22 → 53008 [ACK] Seq=1066 Ack=1554 Win=64128 Len=0 TSval=9155...
200	179.527090308	10.2.2.1	10.4.4.1	SSHv2	114	Client: Diffie-Hellman Key Exchange Init
201	179.538348217	10.4.4.1	10.2.2.1	SSHv2	1182	Server: Diffie-Hellman Key Exchange Reply, New Keys, Encrypte...
202	179.587084792	10.2.2.1	10.4.4.1	TCP	66	53008 → 22 [ACK] Seq=1602 Ack=2182 Win=64128 Len=0 TSval=5194...
204	180.671547789	10.2.2.1	10.4.4.1	SSHv2	82	Client: New Keys
205	180.713747497	10.4.4.1	10.2.2.1	TCP	66	22 → 53008 [ACK] Seq=2182 Ack=1618 Win=64128 Len=0 TSval=9155...
206	180.718990361	10.2.2.1	10.4.4.1	SSHv2	110	Client: Encrypted packet (Len=44)
207	180.719327564	10.4.4.1	10.2.2.1	TCP	66	22 → 53008 [ACK] Seq=2182 Ack=1662 Win=64128 Len=0 TSval=9155...
208	180.719336667	10.4.4.1	10.2.2.1	SSHv2	110	Server: Encrypted packet (Len=44)
209	180.724690846	10.2.2.1	10.4.4.1	TCP	66	53008 → 22 [ACK] Seq=1662 Ack=2226 Win=64128 Len=0 TSval=5194...
210	180.724880273	10.2.2.1	10.4.4.1	SSHv2	126	Client: Encrypted packet (Len=60)
211	180.726870371	10.4.4.1	10.2.2.1	SSHv2	110	Server: Encrypted packet (Len=52)
212	180.908430700	10.4.4.1	10.2.2.1	TCP	130	[TCP Retransmission] 22 → 53008 [PSH, ACK] Seq=2226 Ack=1722...
213	180.976918539	10.2.2.1	10.4.4.1	TCP	78	53008 → 22 [ACK] Seq=1722 Ack=2278 Win=64128 Len=0 TSval=5194...
215	182.558632118	10.2.2.1	10.4.4.1	SSHv2	150	Client: Encrypted packet (Len=84)
216	182.577262038	10.4.4.1	10.2.2.1	SSHv2	94	Server: Encrypted packet (Len=28)
217	182.586326165	10.2.2.1	10.4.4.1	TCP	66	53008 → 22 [ACK] Seq=1806 Ack=2306 Win=64128 Len=0 TSval=5194...
218	182.586326028	10.2.2.1	10.4.4.1	SSHv2	178	Client: Encrypted packet (Len=112)
219	182.636165046	10.4.4.1	10.2.2.1	TCP	66	22 → 53008 [ACK] Seq=2306 Ack=1918 Win=64128 Len=0 TSval=9155...
220	182.675235892	10.4.4.1	10.2.2.1	SSHv2	534	Server: Encrypted packet (Len=468)
221	182.723133211	10.2.2.1	10.4.4.1	TCP	66	53008 → 22 [ACK] Seq=1918 Ack=2774 Win=64128 Len=0 TSval=5194...
222	182.723282310	10.4.4.1	10.2.2.1	SSHv2	110	Server: Encrypted packet (Len=44)
223	182.729145237	10.2.2.1	10.4.4.1	TCP	66	53008 → 22 [ACK] Seq=1918 Ack=2818 Win=64128 Len=0 TSval=5194...
224	182.729283203	10.2.2.1	10.4.4.1	SSHv2	822	Client: Encrypted packet (Len=756)
225	182.738035023	10.4.4.1	10.2.2.1	TCP	66	22 → 53008 [ACK] Seq=2818 Ack=2674 Win=64128 Len=0 TSval=9155...
226	182.735800405	10.4.4.1	10.2.2.1	SSHv2	138	Server: Encrypted packet (Len=72)
227	182.741491267	10.2.2.1	10.4.4.1	SSHv2	110	Client: Encrypted packet (Len=44)
228	182.741791135	10.4.4.1	10.2.2.1	SSHv2	288	Server: Encrypted packet (Len=238)
229	182.902808848	10.4.4.1	10.2.2.1	TCP	208	[TCP Retransmission] 22 → 53008 [PSH, ACK] Seq=2080 Ack=2718...
230	182.907348026	10.2.2.1	10.4.4.1	TCP	78	[TCP Previous segment not captured] 53008 → 22 [ACK] Seq=2778...
231	182.970877446	10.2.2.1	10.4.4.1	TCP	118	[TCP Retransmission] 53008 → 22 [PSH, ACK] Seq=2718 Ack=3119...
232	182.970881020	10.2.2.1	10.4.4.1	TCP	118	[TCP Retransmission] 53008 → 22 [PSH, ACK] Seq=2718 Ack=3119...
233	182.971275193	10.4.4.1	10.2.2.1	TCP	78	22 → 53008 [ACK] Seq=3119 Ack=2770 Win=64128 Len=0 TSval=9155...

Figura 2: Transferência do file1 por SFTP no PC1

Podemos verificar que o "PC1" está conectado a uma rede privada suscetível à perda de pacotes. Logo, é possível que ocorram situações em que pacotes duplicados sejam enviados na tentativa de recuperação de pacotes perdidos. Com a presença de mais pacotes, o tempo de envio aumenta, o que provocará uma diminuição de desempenho nas aplicações, uma vez que estas têm que permanecer em execução por mais tempo.

Podemos verificar também, na figura 2, o protocolo TCP em ação quando acontecem erros na transferência por SFTP.

1.2 Questão 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 controle, 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 os tipos de segmentos trocados e os números de sequência usados quer nos dados como nas confirmações.

FTP usa o protocolo TCP na camada de transporte de maneira a assegurar a eficiente transmissão de arquivos entre um cliente e um servidor. Este protocolo implica o estabelecimento de uma conexão entre os dois intervenientes de forma a poder ocorrer trocas de dados.

- **Início da conexão:**

O protocolo TCP inicia uma conexão seguindo um processo conhecido por "3-way handshake".

Inicialmente, o servidor deseja estabelecer uma ligação com o cliente, por isso envia um segmento TCP SYN, que contém o número de sequência que deve ser sincronizado de forma a se inicializar a conexão. O cliente recebe o segmento e responde ao pedido do cliente com um segmento TCP SYN-ACK, sendo a flag ACK usado para confirmar o sucesso do envio anterior. O servidor confirma a resposta do cliente através de um segmento ACK, sendo estabelecida uma ligação fiável com a qual iniciarão a transferência real de dados.

- **Transmissão de dados:**

A partir deste momento o servidor pode enviar os dados desejados, iniciando assim a fase de transmissão, onde este envia os dados pedidos pelo cliente.

- **Fim da conexão:**

A fase final, ou seja, de fecho da conexão, inicia-se após o envio dos dados, através do envio de um segmento TCP FIN por parte do servidor, de forma a informar o cliente sobre o encerramento do envio de dados. Após esse procedimento, o cliente responde com segmento TCP ACK para indicar que recebeu os dados. Além disso, o cliente envia um segmento TCP FIN-ACK, em resposta ao segmento TCP FIN enviado pelo servidor. Para finalizar, o servidor envia um segmento de confirmação TCP ACK para confirmar que recebeu o último segmento enviado pelo cliente e terminar assim a conexão.

47	40.476957212	10.4.4.1	10.1.1.1	TCP	74 20 -> 51921 [SYN] Seq=0 Win=0 Len=0 MSS=1460 SACK_PERM=1 TSval=3216312750 TSecr=0 WS=128
48	40.476957212	10.1.1.1	10.4.4.1	TCP	74 51921 -> 20 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1460 SACK_PERM=1 TSval=3216312750 TSecr=0 WS=128
49	40.477077138	10.4.4.1	10.1.1.1	TCP	66 20 -> 51921 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=3216312751 TSecr=3361781894
50	40.477153851	10.4.4.1	10.1.1.1	FTP	130 Response: 150 opening BINARY mode data connection for file1 (224 bytes).
51	40.477528031	10.4.4.1	10.1.1.1	FTP-DA	290 FTP Data: 224 bytes (PORT) (RETR file1)
52	40.477522386	10.4.4.1	10.1.1.1	TCP	66 20 -> 51921 [FIN, ACK] Seq=225 Ack=1 Win=0 Len=0 TSval=3216312751 TSecr=3361781894
53	40.477715215	10.1.1.1	10.4.4.1	TCP	66 51921 -> 20 [ACK] Seq=1 Ack=225 Win=65535 Len=0 TSval=3361781894 TSecr=3216312751
54	40.477888447	10.1.1.1	10.4.4.1	TCP	66 51921 -> 20 [FIN, ACK] Seq=1 Ack=226 Win=65535 Len=0 TSval=3361781894 TSecr=3216312751
55	40.478022862	10.4.4.1	10.1.1.1	TCP	66 20 -> 51921 [ACK] Seq=226 Ack=2 Win=64256 Len=0 TSval=3216312752 TSecr=3361781894

Figura 3: Captura FTP

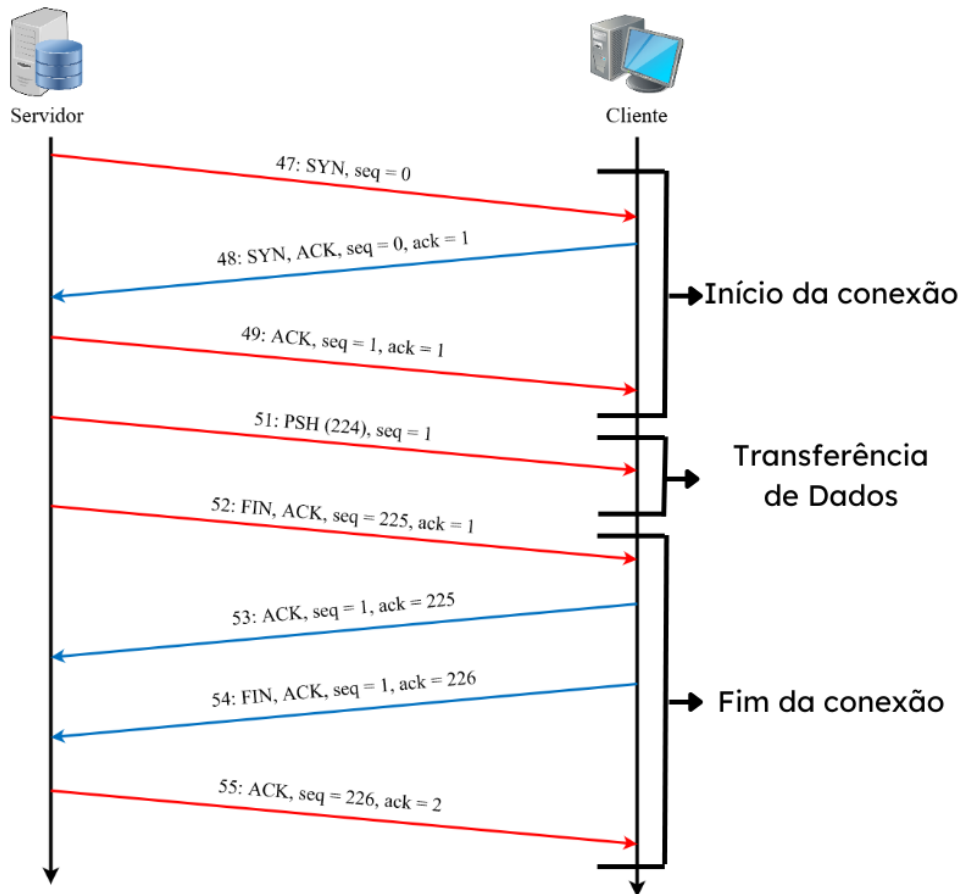


Figura 4: Diagrama Temporal FTP

1.3 Questão 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.

TFTP usa o protocolo UDP na camada de transporte.

O protocolo UDP é leve, o que o torna adequado para transferências de ficheiros rápidas. O UDP atinge o seu objetivo de uma forma simples: envia pacotes diretamente para um computador de destino, sem estabelecer uma conexão primeiro, ao contrário do TCP, que requer o estabelecimento de uma conexão inicial.

Em relação à transferência do ficheiro, o processo inicia com o cliente a enviar um Read Request. Em seguida, o servidor envia um pacote contendo os dados solicitados pelo cliente. O processo conclui com o cliente enviando uma trama ACK, informando ao servidor que os dados foram recebidos com sucesso.

No.	Time	Source	Destination	Protocol	Length	Info
51	83.085019883	10.1.1.1	10.4.4.1	TFTP	56	Read Request, File: file1, Transfer type: octet
52	83.085480703	10.4.4.1	10.1.1.1	TFTP	270	Data Packet, Block: 1 (last)
53	83.085832540	10.1.1.1	10.4.4.1	TFTP	46	Acknowledgement, Block: 1

Figura 5: Captura TFTP

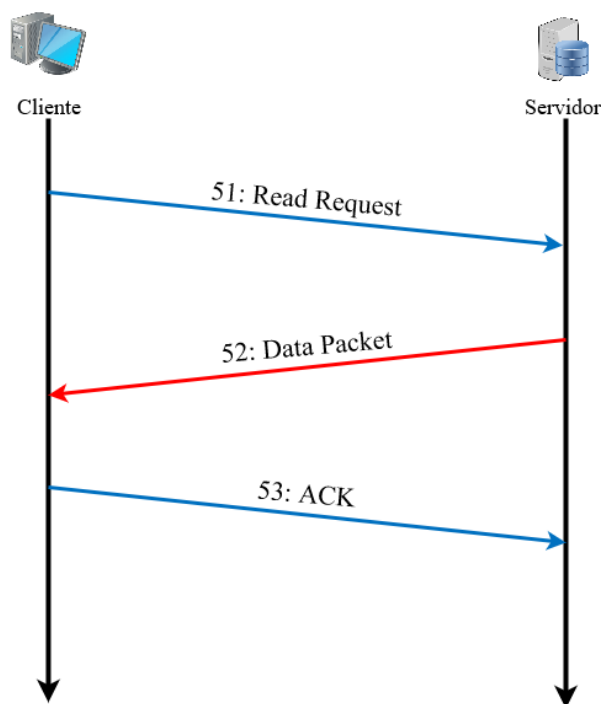


Figura 6: Diagrama Temporal TFTP

1.4 Questão 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;

(i) Uso da Camada de Transporte

- **FTP:** O FTP emprega uma conexão de controlo separada (porta 21) e uma conexão de dados (porta 20), tornando-o um protocolo de duas conexões. Isto requer o gerenciamento de duas conexões para transferência de arquivos. É usado o protocolo TCP para ambas as conexões.
- **TFTP:** TFTP usa o protocolo UDP na camada de transporte, que é um protocolo leve e não orientado à conexão.
- **SFTP:** SFTP usa o protocolo TCP, seguro e confiável como protocolo da camada de transporte subjacente.
- **HTTP:** HTTP usa o protocolo TCP, seguro e confiável como protocolo da camada de transporte subjacente.

(ii) Eficiência

- **FTP:** O FTP oferece bom tratamento de erros e confiabilidade devido ao seu uso do TCP, que fornece entrega confiável e correção de erros. O FTP geralmente oferece altas velocidades de transferência, especialmente em redes de alta largura de banda, mas a velocidade varia devido ao protocolo TCP, dado a processos como Acknowledges e controlo de erros, que levam a tempos de espera.
- **TFTP:** Uma vez que o TFTP utiliza o protocolo UDP, não implementa correção de erros, o que afeta a sua fiabilidade. No entanto, a falta de fiabilidade permite que seja mais rápido do que FTP quando não há erros.
- **SFTP:** Semelhante ao FTP mas, devido à segurança oferecida pelo processo de criptografia/descriptografia providenciada pelo protocolo SSH, pode apresentar velocidades de transferência ligeiramente reduzidas e maior overhead em comparação com o FTP.
- **HTTP:** O HTTP, tal como o FTP, oferece uma transferência de dados fiável com mecanismos robustos de tratamento de erros e recuperação proporcionados pelo TCP.

O HTTP permite que vários requests HTTP sejam enviados através de uma única ligação TCP, sem esperar pelas respostas correspondentes. Isto significa que o cliente pode enviar vários requests através da mesma ligação persistente sem esperar pelas respostas dos requests anteriores.

(iii) Complexidade

- **FTP:** O FTP é um protocolo robusto para transferir ficheiros entre sistemas, oferecendo uma variedade de capacidades de gestão de ficheiros. O uso de ligações de controlo de dados separadas e vários modos de operação adicionam à sua complexidade.
- **TFTP:** O TFTP é o mais simples e menos complexo dos quatro protocolos. Tal como o nome indica, é uma versão simplificada do FTP. O seu design é intencionalmente simples, como podemos ver pelo facto de depender de UDP como protocolo de transporte ao invés de TCP.
- **SFTP:** O SFTP é mais complexo do que o FTP básico devido às funcionalidades de segurança adicionadas. Utiliza o SSH para autenticação e criptografia segura, tornando-o adequado para transferências de ficheiros seguras. Estas funcionalidades de segurança adicionais justificam o aumento da complexidade.
- **HTTP:** O protocolo HTTP na sua forma básica não é particularmente complexo. Envolve principalmente o envio de pedidos de um cliente para um servidor e a receção de respostas em troca. No entanto, a complexidade pode surgir das diferentes funcionalidades adicionais, capacidade de lidar com vários tipos de dados e outros componentes que foram adicionados e evoluíram ao longo do tempo.

(iv) Segurança

- **FTP:** O protocolo tradicional de transferência de ficheiros (FTP) é uma forma simples de transferir dados, mas não oferece nenhuma proteção em termos de dados. Os ficheiros são transferidos sem encriptação, tornando os dados legíveis para qualquer pessoa que os intercepte.
- **TFTP:** O TFTP é considerado inseguro principalmente porque carece de encriptação, deixando os dados vulneráveis a interceptações. Para além disto não possui autenticação. Mesmo o FTP, que não é a opção mais segura para transferência de ficheiros, utiliza autenticação.
- **SFTP:** O SFTP é um protocolo seguro usado para transferir ficheiros entre dois computadores através de uma ligação segura. É um protocolo que fornece encriptação, autenticação e integridade dos dados para transferências de ficheiros.

O SFTP incorpora o protocolo Secure Shell (SSH) no processo de armazenamento e transferência. Os dados são encriptados no servidor e durante a transmissão. Caso os dados sejam roubados durante uma transferência SFTP, o ladrão não será capaz de lê-los sem quebrar a encriptação.

- **HTTP:** HTTP não encripta os dados durante a comunicação cliente-servidor, o que significa que qualquer dado transmitido através de HTTP é enviado em texto simples, sem qualquer encriptação ou mecanismos de segurança. Como resultado, pode ser interceptado e lido por qualquer pessoa com acesso ao tráfego de rede.

2 Parte II

2.1 Questão 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	Protocolo de Aplicação	Protocolo de transporte	Porta de atendimento	Overhead de transporte em bytes
wget, lynx	HTTP	TCP	80	20
ssh, sftp	SSH	TCP	22	20
ftp	FTP	TCP	21	20
tftp	TFTP	UDP	69	8
telnet	TELNET	TCP	23	20
nslookup	DNS	UDP	53	8
ping	PING	N/A	N/A	N/A
traceroute	TRACEROUTE	UDP	33446	8

A partir de uma análise da tabela é possível perceber que todos os comandos cujo protocolo de transporte seja o protocolo UDP possuem um overhead fixo de 8 bytes. Por outro lado, os comandos cujo protocolo de transporte seja o protocolo TCP têm, geralmente, um overhead fixo de 20 bytes, sendo que este possa ser incrementado dependendo se o campo options é utilizado ou não, algo que não se verificou para nenhum dos comandos utilizados pelo nosso grupo.

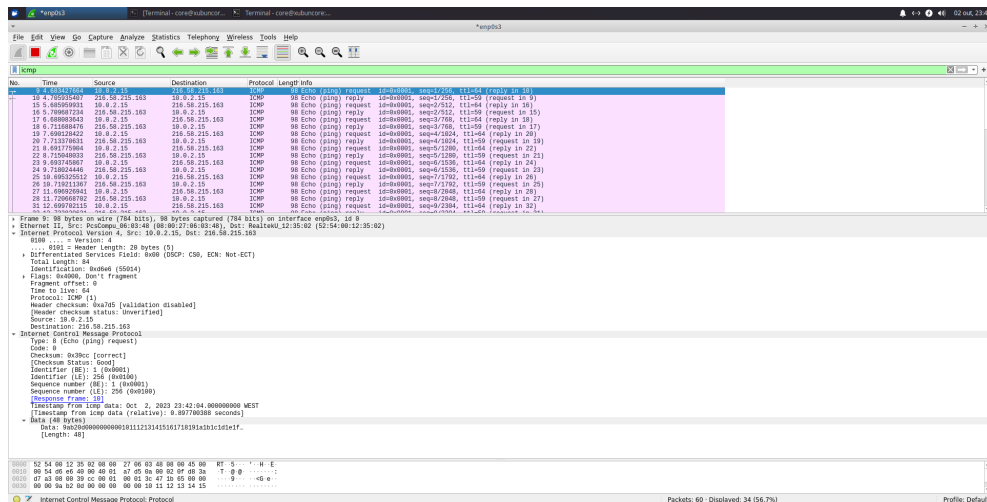


Figura 7: Captura Ping

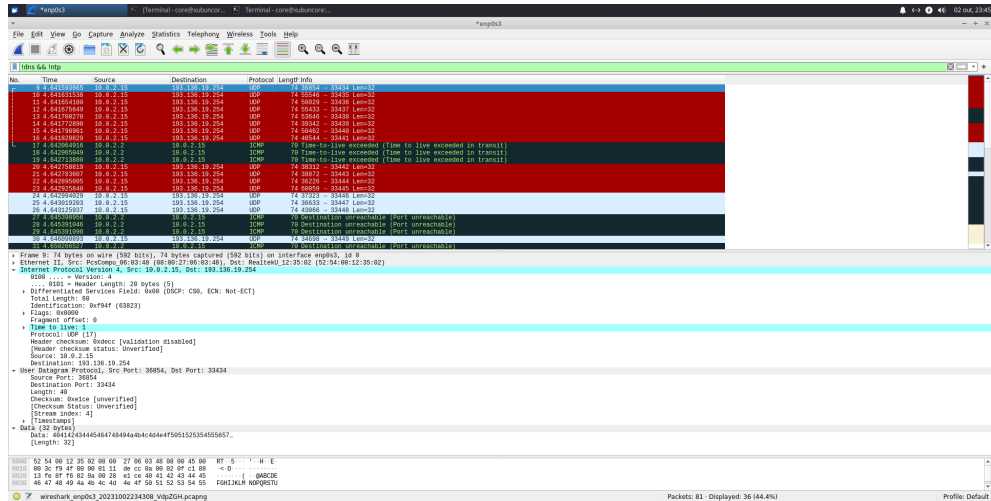


Figura 8: Captura Traceroute

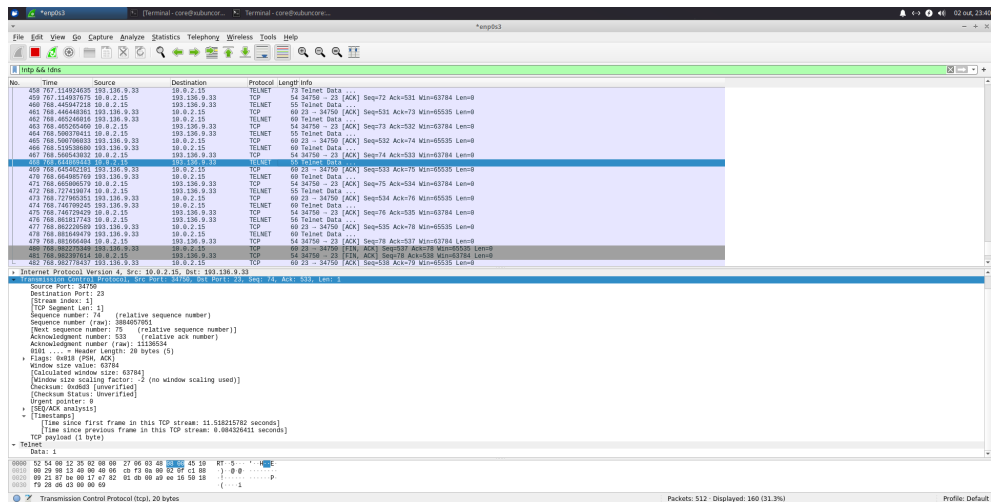


Figura 9: Captura Telnet



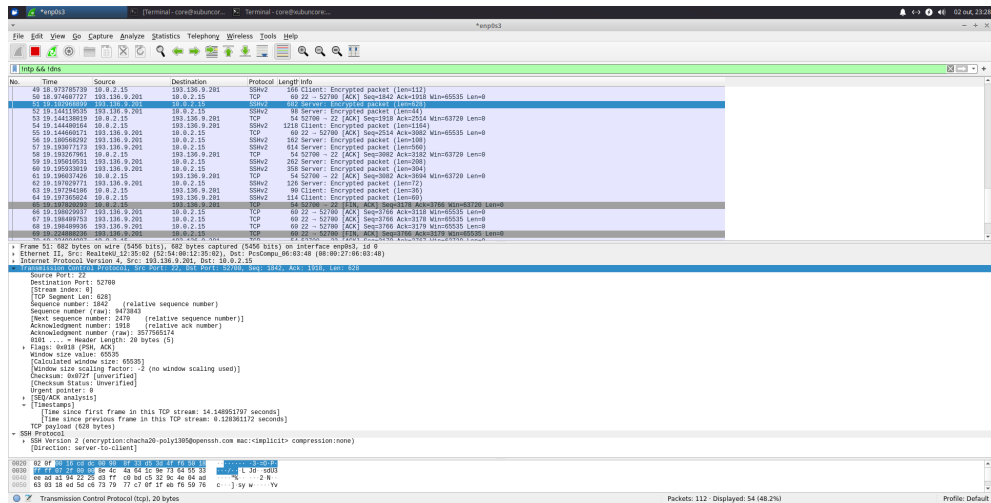


Figura 12: Captura SSH

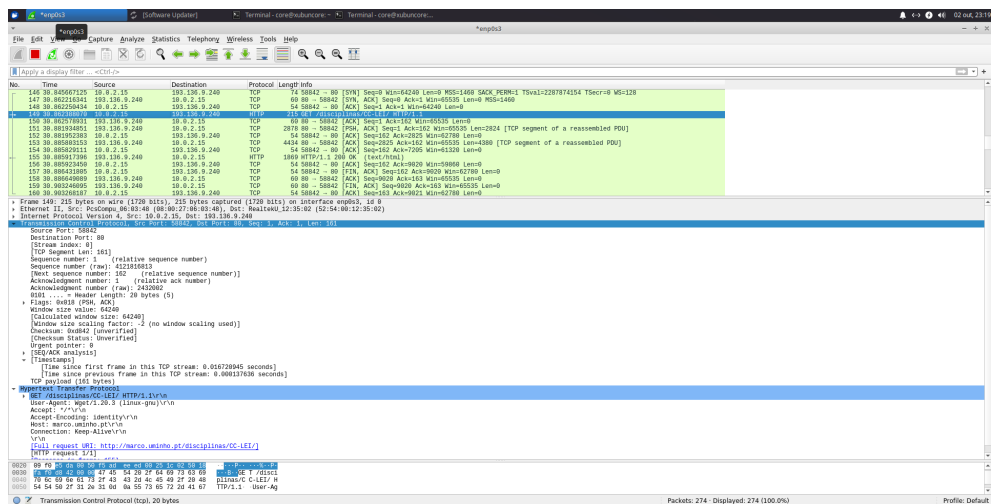


Figura 13: Captura HTTP

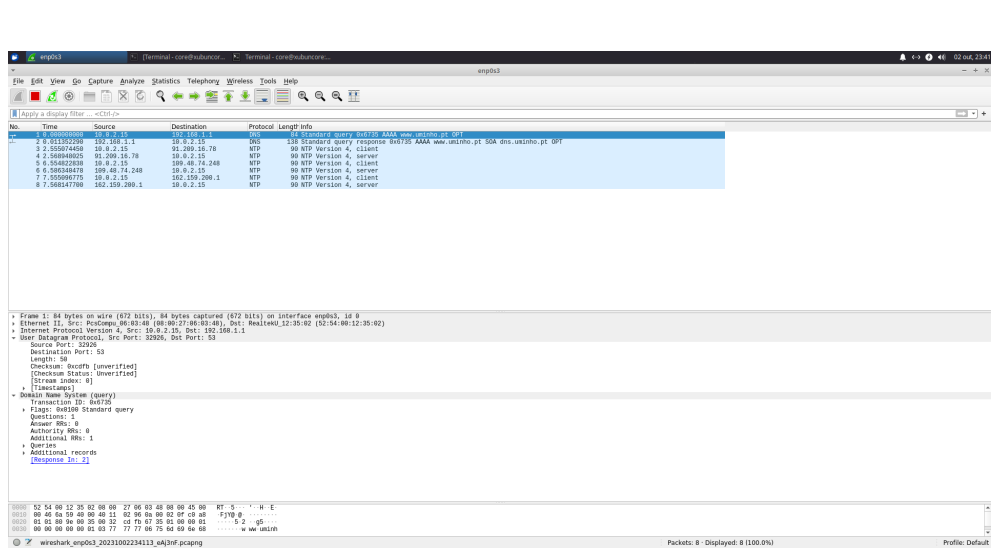


Figura 14: Captura Nslookup

3 Conclusão

Para a realização deste trabalho utilizamos a máquina virtual disponibilizada pelos docentes desta UC, para termos acesso às ferramentas necessárias: Core e Wireshark. Com a realização deste projeto conseguimos ter um melhor entendimento sobre os protocolos tanto da Camada de Transporte como da Camada de Aplicação, e aprendemos também acerca das vantagens e desvantagens de cada um dos seus constituintes.

Com isto, e após tudo o que efetuamos no decorrer do trabalho, podemos afirmar que ao realizar transferências em que a perda de pacotes não seja algo muito importante o melhor protocolo é o UDP, uma vez que é o protocolo de transporte com a melhor velocidade de transferência. Por outro lado, se quisermos garantir que todos os pacotes enviados são recebidos o melhor protocolo a ser utilizado é o TCP, mesmo não sendo o mais eficiente.