

TP1 - Protocolos da Camada de Transporte

Comunicações por Computador - Grupo 7

André Carvalho da Cunha Martins A89586
Bárbara Ferreira Teixeira A89610
Pedro Miguel de Soveral Pacheco Barbosa A89529



Fig. 1. A89586



Fig. 2. A89610



Fig. 3. A89529

16 de março de 2021

Parte 1

André Martins, Bárbara Teixeira, and Pedro Barbosa

University of Minho, Department of Informatics, 4710-057 Braga, Portugal
e-mail: {a89586,a89610,a89529}@alunos.uminho.pt

1 Questões e Respostas

1.1 Pergunta 1

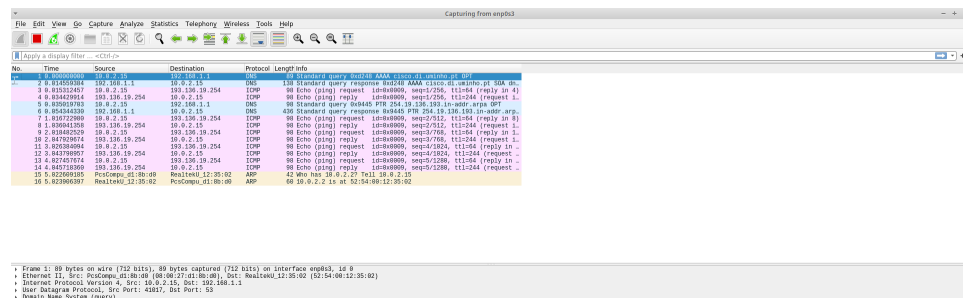
Identifique, numa tabela, para cada comando executado, qual o protocolo de aplicação, o protocolo de transporte, porta de atendimento e overhead de transporte:

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	PING			
Traceroute	TRACEROUTE	UDP	33446	8
Telnet	TELNET	TCP	23	20
FTP	FTP	TCP	21	20
TFTP	TFTP	UDP	69	8
Browser/http	HTTP	TCP	80	32
Nslookup	DNS	UDP	53	8
SSH	SSH	TCP	22	20

Todos os comandos associados ao protocolo de transporte UDP têm overhead de transporte fixo, com o tamanho igual a 8 bytes. Os restantes bytes já dizem respeito à informação transportada.

Já no protocolo TCP, temos um overhead de 20 bytes, que pode ser acrescido se forem alocados mais alguns bytes para opções, ou seja, este valor pode variar consoante o campo options esteja, ou não, a ser utilizado (Exemplo do HTTP)

1.2 Ping



The screenshot shows a Wireshark packet capture of ICMP Echo (ping) traffic. The packet list on the left shows several packets, with the selected packet (No. 1) being an ICMP Echo request from 192.168.1.1 to 192.168.1.2. The packet details pane on the right shows the structure of the ICMP Echo request, including the type (0), code (0), identifier (0), and sequence number (1). The packet bytes pane at the bottom shows the raw data of the packet, which is a 32-byte ICMP Echo request.

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	192.168.1.1	192.168.1.2	ICMP	32	Echo (ping) request
2	0.000000	192.168.1.2	192.168.1.1	ICMP	32	Echo (ping) reply
3	0.000000	192.168.1.1	192.168.1.2	ICMP	32	Echo (ping) request
4	0.000000	192.168.1.2	192.168.1.1	ICMP	32	Echo (ping) reply
5	0.000000	192.168.1.1	192.168.1.2	ICMP	32	Echo (ping) request
6	0.000000	192.168.1.2	192.168.1.1	ICMP	32	Echo (ping) reply
7	0.000000	192.168.1.1	192.168.1.2	ICMP	32	Echo (ping) request
8	0.000000	192.168.1.2	192.168.1.1	ICMP	32	Echo (ping) reply
9	0.000000	192.168.1.1	192.168.1.2	ICMP	32	Echo (ping) request
10	0.000000	192.168.1.2	192.168.1.1	ICMP	32	Echo (ping) reply
11	0.000000	192.168.1.1	192.168.1.2	ICMP	32	Echo (ping) request
12	0.000000	192.168.1.2	192.168.1.1	ICMP	32	Echo (ping) reply
13	0.000000	192.168.1.1	192.168.1.2	ICMP	32	Echo (ping) request
14	0.000000	192.168.1.2	192.168.1.1	ICMP	32	Echo (ping) reply
15	0.000000	192.168.1.1	192.168.1.2	ICMP	32	Echo (ping) request
16	0.000000	192.168.1.2	192.168.1.1	ICMP	32	Echo (ping) reply
17	0.000000	192.168.1.1	192.168.1.2	ICMP	32	Echo (ping) request
18	0.000000	192.168.1.2	192.168.1.1	ICMP	32	Echo (ping) reply
19	0.000000	192.168.1.1	192.168.1.2	ICMP	32	Echo (ping) request
20	0.000000	192.168.1.2	192.168.1.1	ICMP	32	Echo (ping) reply

1.3 Traceroute

[illegible]

1.4 Telnet

[illegible]

1.5 FTP

[illegible]

1.6 TFTP

The image shows a Wireshark packet capture analysis of a DNS query and response. The top toolbar includes icons for File, Edit, View, Go, Capture, Statistics, Telephony, Wireless, Tools, and Help. The main display area shows a list of packets with columns for No., Time, Source, Destination, Protocol, and Length. The selected packet is packet 6, a DNS Standard query response from 192.168.1.1 to 192.168.1.10.

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	192.168.1.10	192.168.1.1	DNS	86	Standard query 0x2008 A c2002.dns.net OPT
2	0.000001195	192.168.1.10	192.168.1.1	DNS	86	Standard query 0x2008 AAAA c2002.dns.net OPT
3	0.000177021	192.168.1.1	192.168.1.10	DNS	146	Standard query response 0x2008 AAAA c2002.dns.net 50M rfi.n...
4	0.000000072	192.168.1.1	192.168.1.10	DNS	100	Standard query response 0x2008 A c2002.dns.net A 192.168.1...
5	0.000000042	192.168.1.10	192.168.1.1	DNS	86	Standard query 0x2008 AAAA c2002.dns.net OPT

The packet details pane for packet 6 shows the following structure:

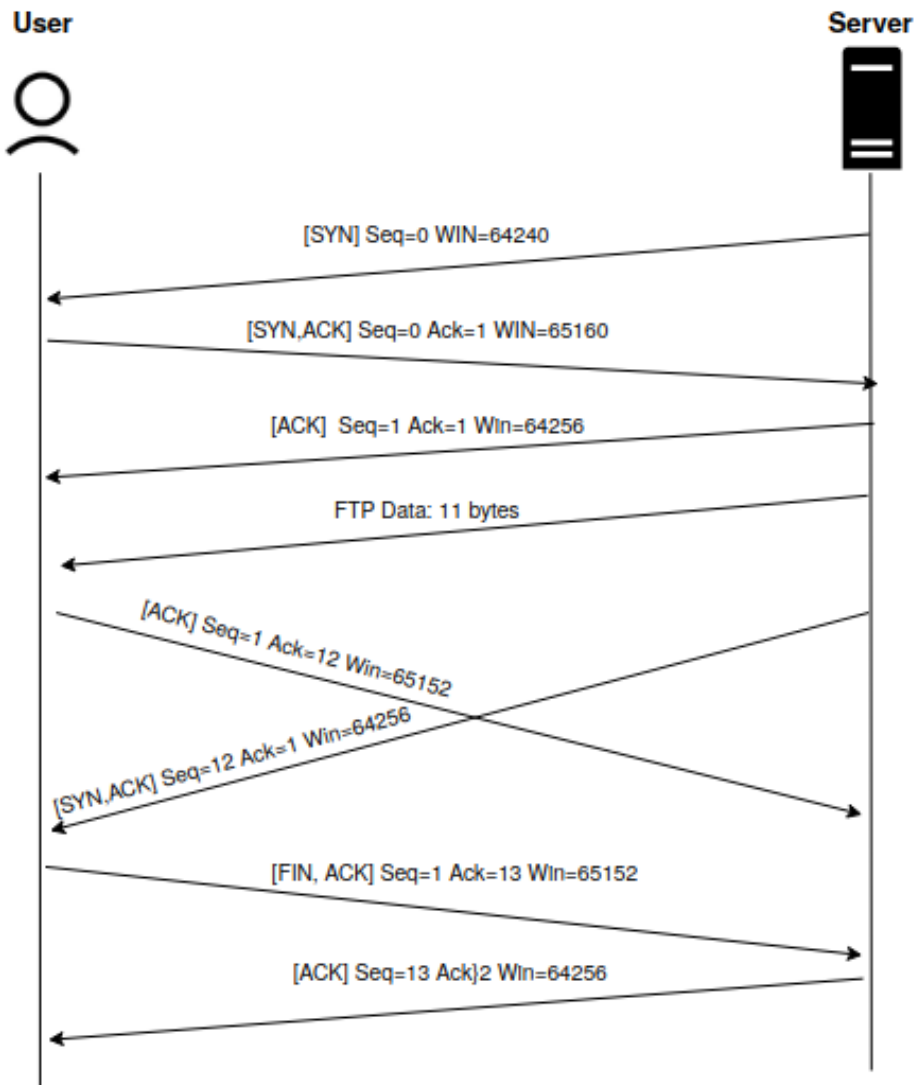
- Frame 5: 86 bytes on wire (688 bits), 86 bytes captured (688 bits) on interface ethp03, 0 s
- Ethernet II, Src: Realtek-UTP-B0-00, Dst: Realtek-UTP-B0-02 (12-14-00-12-30-02)
- Internet Protocol Version 4, Src: 192.168.1.10, Dst: 192.168.1.1
- User Datagram Protocol, Src Port: 60559, Dst Port: 69

2 Pergunta 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.

2.1 FTP

48	55.155572724	10.1.1.1	10.4.4.1	TCP	74	20	41819	[SYN]	Seq=0	Win=64240	Len=0	MSS=1460	SACK_PERM=1	TSval=2181091709	TSecr=0	WS=120
49	55.155786761	10.4.4.1	10.1.1.1	TCP	74	41819	20	[SYN, ACK]	Seq=0	Ack=1	Win=65160	Len=0	MSS=1460	SACK_PERM=1	TSval=3325706606	TSecr=218
50	55.155950887	10.1.1.1	10.4.4.1	TCP	66	20	41819	[ACK]	Seq=1	Ack=1	Win=64256	Len=0	TSval=2181091709	TSecr=3325706606		
51	55.156227260	10.1.1.1	10.4.4.1	FTP	105			Response: 150	Here comes the directory listing.							
52	55.156347579	10.1.1.1	10.4.4.1	FTP-DA	192			FTP Data: 126 bytes (PORT) (LIST)								
53	55.156352024	10.1.1.1	10.4.4.1	TCP	66	20	41819	[FIN, ACK]	Seq=127	Ack=1	Win=64256	Len=0	TSval=2181091710	TSecr=3325706606		
54	55.156587637	10.4.4.1	10.1.1.1	TCP	66	41819	20	[ACK]	Seq=1	Ack=127	Win=65152	Len=0	TSval=3325706607	TSecr=2181091710		
55	55.156814995	10.4.4.1	10.1.1.1	TCP	66	41819	20	[FIN, ACK]	Seq=1	Ack=128	Win=65152	Len=0	TSval=3325706607	TSecr=2181091710		
56	55.156963615	10.1.1.1	10.4.4.1	TCP	66	20	41819	[ACK]	Seq=128	Ack=2	Win=64256	Len=0	TSval=2181091710	TSecr=3325706607		
57	55.157029343	10.1.1.1	10.4.4.1	FTP	90			Response: 226	Directory send OK.							
58	55.157311113	10.4.4.1	10.1.1.1	TCP	66	40874	21	[ACK]	Seq=86	Ack=245	Win=64256	Len=0	TSval=3325706608	TSecr=2181091709		
59	59.905209076	10.4.4.1	10.1.1.1	FTP	74			Request: TYPE I								
60	59.905487331	10.1.1.1	10.4.4.1	FTP	97			Response: 200	Switching to Binary mode.							
61	59.905866257	10.4.4.1	10.1.1.1	FTP	88			Request: PORT 10,4,4,1,174,81								
62	59.906256558	10.1.1.1	10.4.4.1	FTP	117			Response: 200	PORT command successful. Consider using PASV.							
63	59.906531052	10.4.4.1	10.1.1.1	FTP	78			Request: RETR file1								
64	59.907085349	10.1.1.1	10.4.4.1	TCP	74	20	44625	[SYN]	Seq=0	Win=64240	Len=0	MSS=1460	SACK_PERM=1	TSval=2181090621	TSecr=0	WS=120
65	59.907277991	10.4.4.1	10.1.1.1	TCP	74	44625	20	[SYN, ACK]	Seq=0	Ack=1	Win=65160	Len=0	MSS=1460	SACK_PERM=1	TSval=3325711418	TSecr=218
66	59.907423650	10.1.1.1	10.4.4.1	TCP	66	20	44625	[ACK]	Seq=1	Ack=1	Win=64256	Len=0	TSval=2181090621	TSecr=3325711418		
67	59.907526834	10.1.1.1	10.4.4.1	FTP	129			Response: 150	Opening BINARY mode data connection for file1 (11 bytes).							
68	59.907692400	10.1.1.1	10.4.4.1	FTP-DA	77			FTP Data: 11 bytes (PORT) (RETR file1)								
69	59.907753060	10.1.1.1	10.4.4.1	TCP	66	20	44625	[FIN, ACK]	Seq=12	Ack=1	Win=64256	Len=0	TSval=2181090621	TSecr=3325711418		
70	59.907875334	10.4.4.1	10.1.1.1	TCP	66	44625	20	[ACK]	Seq=1	Ack=12	Win=65152	Len=0	TSval=3325711418	TSecr=2181090621		
71	59.907986727	10.4.4.1	10.1.1.1	TCP	66	44625	20	[FIN, ACK]	Seq=1	Ack=13	Win=65152	Len=0	TSval=3325711418	TSecr=2181090621		
72	59.908121090	10.1.1.1	10.4.4.1	TCP	66	20	44625	[ACK]	Seq=13	Ack=2	Win=64256	Len=0	TSval=2181090621	TSecr=3325711418		
73	59.908332556	10.1.1.1	10.4.4.1	FTP	90			Response: 226	Transfer complete.							
74	59.908580248	10.4.4.1	10.1.1.1	TCP	66	40874	21	[ACK]	Seq=128	Ack=414	Win=64256	Len=0	TSval=3325711419	TSecr=2181090621		
75	60.008079800	10.1.1.254	224.0.0.5	OSPF	78			Hello Packet								
76	60.008296906	fe80::200:ff:feaa:10	ff02::5	OSPF	99			Hello Packet								
77	65.748301286	10.4.4.1	10.1.1.1	FTP	72			Request: QUIT								
78	65.748582296	10.1.1.1	10.4.4.1	FTP	89			Response: 221	Goodbye.							
79	65.748666662	10.1.1.1	10.4.4.1	TCP	66	20	80874	[FIN, ACK]	Seq=1	Ack=1	Win=64256	Len=0	TSval=3325711420	TSecr=2181090621		



O protocolo FTP possui dois modos de execução, o modo passivo e o modo ativo. Estes modos são determinados por quem inicia a conexão, ou seja, se esta é iniciada pelo cliente, ou se esta é iniciada pelo servidor. A conexão com a porta 20 do servidor é a segunda conexão a ser criada, sendo a primeira a conexão com a porta 21 do servidor. Os servidores escutam os clientes através da sua porta 21. Os cliente conectam-se a essa porta para iniciar as operações de transferência, contudo, a porta 20 é necessária e essencial para essa transferência ser realizada.

Modo ativo: Se a conexão for inicializada pelo servidor, esta será ativa, isto é, tanto o servidor como o cliente necessitam de abrir portas para receber tráfego.

Modo passivo: Se a conexão for iniciada por um cliente, a conexão será passiva, portanto, apenas é necessário o servidor abrir portas para o tráfego. A grande maioria dos servidores FTP prefere a conexão passiva devido a problemas de segurança.

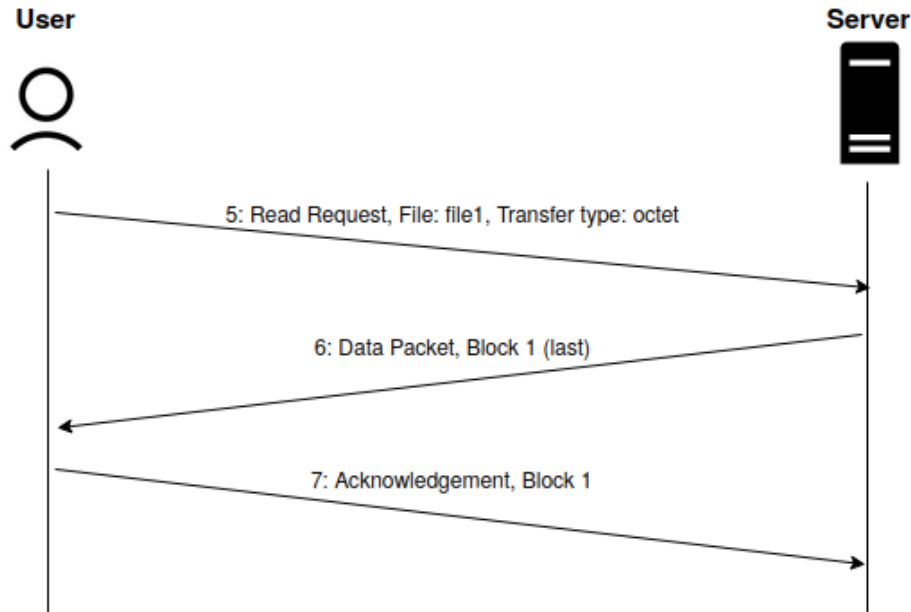
Na figura apresentada, o servidor inicia a conexão através da porta 20 (estamos perante uma conexão FTP ativa). Depois do pedido feito pelo cliente, a conexão é iniciada pelo servidor através de um pacote SYN, com o número de sequência 0. Após este envio, o cliente responde com um pacote SYN e com um pacote ACK, demonstrando sucesso na recepção do pacote SYN enviado pelo servidor.

O servidor responde com um pacote ACK. Após esta fase, o servidor enviará para o cliente os dados pretendidos, começando assim a transferência de dados.

Na última fase, para dar como encerrada a conexão, o servidor irá enviar um pacote FIN, informando que terminou a transferência de dados. O cliente responde com um pacote ACK e com um pacote FIN, junto de outro pacote ACK, referente ao primeiro FIN, enviado pelo servidor. Para finalizar, o servidor envia um ACK, dando assim por terminada a conexão.

2.2 TFTP

40	180.0103000556	10.1.1.254	224.0.0.5	OSPF	78	Hello Packet
50	180.104525931	fe80::200:ff:feaa:10	ff02::5	OSPF	98	Hello Packet
51	182.560691085	00:00:00:aa:00:10	Broadcast	ARP	42	Who has 10.1.1.1? Tell 10.1.1.254
52	182.560856932	00:00:00:aa:00:14	00:00:00:aa:00:10	ARP	42	10.1.1.1 is at 00:00:00:aa:00:14
53	182.560856932	10.4.4.1	10.1.1.1	TFTP	58	Read Request, File: file1, Transfer type: octet
54	182.560856932	10.1.1.1	10.4.4.1	TFTP	57	Data Packet, Block: 1 (last)
55	182.560856932	10.4.4.1	10.1.1.1	TFTP	46	Acknowledgement, Block: 1
56	184.641923693	fe80::34a6:94ff:fed...	ff02::2	ICMPv6	70	Router Solicitation from 26:4b:a7:8c:d9:10
57	184.641923693	fe80::34a6:94ff:fed...	ff02::2	ICMPv6	70	Router Solicitation from 26:4b:a7:8c:d9:10
58	187.713988078	00:00:00:aa:00:14	00:00:00:aa:00:10	ARP	42	Who has 10.1.1.254? Tell 10.1.1.1
59	187.713988432	00:00:00:aa:00:10	00:00:00:aa:00:14	ARP	42	10.1.1.254 is at 00:00:00:aa:00:10



Neste exemplo, o cliente começa por enviar um Read Request. Posteriormente, o servidor envia um pacote que contém os dados pretendidos. Ao receber estes dados, o cliente irá enviar um ACK, informando o servidor do sucesso na receção dos dados.

Neste protocolo, todas as transferências se iniciam com um pedido de leitura ou de escrita num ficheiro, servindo este pedido também para estabelecer uma conexão. Caso o servidor aceda a esta conexão, os ficheiros serão emitidos em blocos de 512 bytes. Caso o tamanho do pacote seja superior a esses 512 bytes, verificar-se-á uma fragmentação, sendo enviados vários pacotes, com o mesmo limite de comprimento.

3 Pergunta 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.

3.1 i) Uso da camada de transporte

SFTP: Protocolo TCP

FTP: Protocolo TCP

TFTP: Protocolo UDP

HTTP: Protocolo TCP

3.2 ii) Eficiência na transferência

SFTP: Semelhante a FTP, possuindo dados encriptados.

FTP: Devido ao uso do protocolo FTP, garante-se que os dados são transmitidos, devido ao uso de ACK, contudo, este procedimento perde eficiência, uma vez que é necessário esperar pelo ACK. Devido à falta de encriptação, fica mais suscetível a interferências por parte de terceiros. Este protocolo revela alguma perda de eficiência, uma vez que, dando o exemplo de um pequeno pacote de dados, iriam ser gastos imensos recursos no envio de ACK e SYN, podendo até estes superar o tamanho dos dados que se pretendia enviar.

TFTP: Uma vez que utiliza o protocolo UDP, torna-se menos viável. Devido à não utilização de ACK, este protocolo torna-se menos viável, não sendo possível confirmar se o pacote foi entregue com sucesso. Contudo, se for sempre bem sucedido, torna-se mais eficiente do que o FTP.

HTTP: Permite que vários HTTP requests sejam enviados numa única ligação TCP, não havendo necessidade de esperar pelas respostas correspondentes.

3.3 iii) Complexidade

SFTP: Este protocolo permite acesso, transferência e gestão de dados, revelando-se bastante complexo, pois todas estas funcionalidades revelam elevados custos de processamento.

FTP: Este protocolo suporta vários pedidos para transferir dados em paralelo, em que cada transferência realiza uma nova conexão, originando, assim, diferentes velocidades de transferência. Revela-se, também, um protocolo bastante complexo. O envio de ACK também torna este protocolo bastante complexo, uma vez que uma pequena transferência de dados requer o envio de bastantes pacotes para que essa transferência seja bem sucedida.

TFTP: Este protocolo é uma alternativa simplificada do protocolo FTP. Ao contrário do protocolo FTP, este possui muito menos funcionalidades do que a versão mais complexa. O facto de ser baseado em UDP também se reflete no facto de ser menos complexo, uma vez que não existe uma necessidade de envio de pacotes ACK.

HTTP: Este protocolo é capaz de garantir confiança e escalabilidade de sistemas. Além disso, oferece um suporte para hipermedia para redes complexas ou para redes diferentes e não confiáveis. Considera-se, então, um protocolo complexo.

3.4 iv) Segurança

SFTP: O SSH permite-nos dizer que este protocolo é seguro. Para entender a segurança deste protocolo, podemos explicar sucintamente como funciona o SSH. O SSH utiliza uma arquitetura em camadas, sendo elas uma camada de transporte, uma camada de autenticação e uma camada de conexão. A camada de transporte é executada através de TCP/IP, fornecendo encriptação, autenticação de servidor e proteção de dados, enquanto que a camada de autenticação é responsável por autenticar os clientes.

FTP: Este protocolo já é conhecido por ter diversas falhas de segurança, muito devido à falta de encriptação dos dados. Ao não existir essa encriptação das transmissões, qualquer indivíduo consegue realizar captura de pacotes na rede, tornando, assim, este protocolo bastante inseguro.

TFTP: Este protocolo não fornece autenticação, logo, não existe proteção dos dados que se pretende transferir. Conclui-se, assim, que não é um protocolo seguro.

HTTP: Neste protocolo, toda a informação é representada em texto, não existindo encriptação. Assim, tanto os dados do cliente como do servidor podem ser alterados por terceiros, tornando este protocolo inseguro.

4 Pergunta 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). Na LAN3 verifica-se uma situação de problemas a nível das lig-

ações de rede, ocorrendo situações de perda e duplicação de pacotes IP. Os dois protocolos estudados, UDP e TCP, lidam de maneira diferente com esta situação.

No caso do TCP, e no caso de existir alguma perda de pacotes, existirá um reenvio desses mesmos pacotes, havendo uma redução do débito de transmissão.

No caso do UDP, não existe nenhum mecanismo que assegura a receção bem sucedida dos pacotes enviados (Inexistência do envio de ACK), não sendo possível detetar a perda de pacotes. Contudo, e dependendo do protocolo que está a executar acima do UDP, os pacotes podem possuir uma identificação, sendo possível analisar manualmente os pacotes recebidos e a suposta falta de alguns pacotes.

Como é possível analisar através da imagem, o ping efetuado na LAN3 registou duplicação de pacotes. Se estivessemos a lidar com TCP, iríamos assistir a um constante reenvio de pacotes, para garantir que estes chegavam ao seu destino.

Em contexto UDP, existem pacotes que vão sendo perdidos, podendo ou não ser reenviados, dependendo do protocolo de aplicação, o que faz com que este protocolo de transporte não dê garantias da entrega dos pacotes.

```
<037/Laptop1.conf# ping -c 20 10.1.1.1 | tee file-ping-output
PING 10.1.1.1 (10.1.1.1) 56(84) bytes of data.
64 bytes from 10.1.1.1: icmp_seq=1 ttl=61 time=0.439 ms
64 bytes from 10.1.1.1: icmp_seq=2 ttl=61 time=0.392 ms
64 bytes from 10.1.1.1: icmp_seq=3 ttl=61 time=0.388 ms
64 bytes from 10.1.1.1: icmp_seq=4 ttl=61 time=0.342 ms
64 bytes from 10.1.1.1: icmp_seq=5 ttl=61 time=0.355 ms
64 bytes from 10.1.1.1: icmp_seq=6 ttl=61 time=0.483 ms
64 bytes from 10.1.1.1: icmp_seq=7 ttl=61 time=0.381 ms
64 bytes from 10.1.1.1: icmp_seq=8 ttl=61 time=0.397 ms
64 bytes from 10.1.1.1: icmp_seq=9 ttl=61 time=0.309 ms
64 bytes from 10.1.1.1: icmp_seq=10 ttl=61 time=0.354 ms
64 bytes from 10.1.1.1: icmp_seq=11 ttl=61 time=0.449 ms
64 bytes from 10.1.1.1: icmp_seq=12 ttl=61 time=0.864 ms
64 bytes from 10.1.1.1: icmp_seq=13 ttl=61 time=0.324 ms
64 bytes from 10.1.1.1: icmp_seq=14 ttl=61 time=0.344 ms
64 bytes from 10.1.1.1: icmp_seq=15 ttl=61 time=0.381 ms
64 bytes from 10.1.1.1: icmp_seq=16 ttl=61 time=0.393 ms
64 bytes from 10.1.1.1: icmp_seq=17 ttl=61 time=0.372 ms
64 bytes from 10.1.1.1: icmp_seq=18 ttl=61 time=0.540 ms
64 bytes from 10.1.1.1: icmp_seq=19 ttl=61 time=0.304 ms
64 bytes from 10.1.1.1: icmp_seq=20 ttl=61 time=0.358 ms
```

```
PING 10.1.1.1 (10.1.1.1) 56(84) bytes of data.
64 bytes from 10.1.1.1: icmp_seq=1 ttl=61 time=5.23 ms
64 bytes from 10.1.1.1: icmp_seq=2 ttl=61 time=5.32 ms
64 bytes from 10.1.1.1: icmp_seq=3 ttl=61 time=5.39 ms
64 bytes from 10.1.1.1: icmp_seq=4 ttl=61 time=5.39 ms
64 bytes from 10.1.1.1: icmp_seq=5 ttl=61 time=5.24 ms
64 bytes from 10.1.1.1: icmp_seq=6 ttl=61 time=5.42 ms
64 bytes from 10.1.1.1: icmp_seq=7 ttl=61 time=5.25 ms
64 bytes from 10.1.1.1: icmp_seq=8 ttl=61 time=5.40 ms
64 bytes from 10.1.1.1: icmp_seq=9 ttl=61 time=5.40 ms
64 bytes from 10.1.1.1: icmp_seq=10 ttl=61 time=5.33 ms
64 bytes from 10.1.1.1: icmp_seq=10 ttl=61 time=5.43 ms (DUP!)
64 bytes from 10.1.1.1: icmp_seq=11 ttl=61 time=5.35 ms
64 bytes from 10.1.1.1: icmp_seq=12 ttl=61 time=5.35 ms
64 bytes from 10.1.1.1: icmp_seq=13 ttl=61 time=5.29 ms
64 bytes from 10.1.1.1: icmp_seq=14 ttl=61 time=5.31 ms
64 bytes from 10.1.1.1: icmp_seq=15 ttl=61 time=5.38 ms
64 bytes from 10.1.1.1: icmp_seq=16 ttl=61 time=5.32 ms
64 bytes from 10.1.1.1: icmp_seq=17 ttl=61 time=5.38 ms
64 bytes from 10.1.1.1: icmp_seq=18 ttl=61 time=5.52 ms
64 bytes from 10.1.1.1: icmp_seq=18 ttl=61 time=5.68 ms (DUP!)
64 bytes from 10.1.1.1: icmp_seq=19 ttl=61 time=5.20 ms
64 bytes from 10.1.1.1: icmp_seq=20 ttl=61 time=5.35 ms

--- 10.1.1.1 ping statistics ---
20 packets transmitted, 20 received, +2 duplicates, 0% packet loss, time 19025ms
```

5 Conclusões

A realização deste trabalho prático permitiu-nos aprofundar conhecimentos sobre protocolos da camada de transporte, nomeadamente UDP e TCP. Estudamos os diferentes comportamentos e características de cada um e as suas respetivas vantagens e desvantagens.

Recorrendo à utilização do core e de ferramentas como o Wireshark, foi possível observar a teoria estudada, nomeadamente a diferença de comportamentos de protocolo TCP para protocolo UDP, analisando como se processa a transferência de dados em ambos.

Portanto, e de uma forma resumida e sucinta, podemos aferir que, se pretendemos garantir que todos os pacotes são entregues e não existe perda de dados, devemos optar pelo protocolo TCP, uma vez que este possui mecanismos capazes de oferecer essas garantias. Caso a perda de pacotes seja um problema insignificante, o UDP revela-se uma melhor aposta, pois é possível usufruir de uma maior velocidade de transferência de dados.