

Universidade do Minho
Escola de Engenharia

Comunicações por computadores

TP1 - Protocolos da Camada de Transporte

Grupo 9

16 Março de 2021

Ana Luísa Lira Tomé Carneiro A89533

Henrique Manuel Ferreira da Silva Guimarães Ribeiro A89582

Pedro Almeida Fernandes A89574

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	Não	Não	Não	Não
tracert	tracert	UDP	33430	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

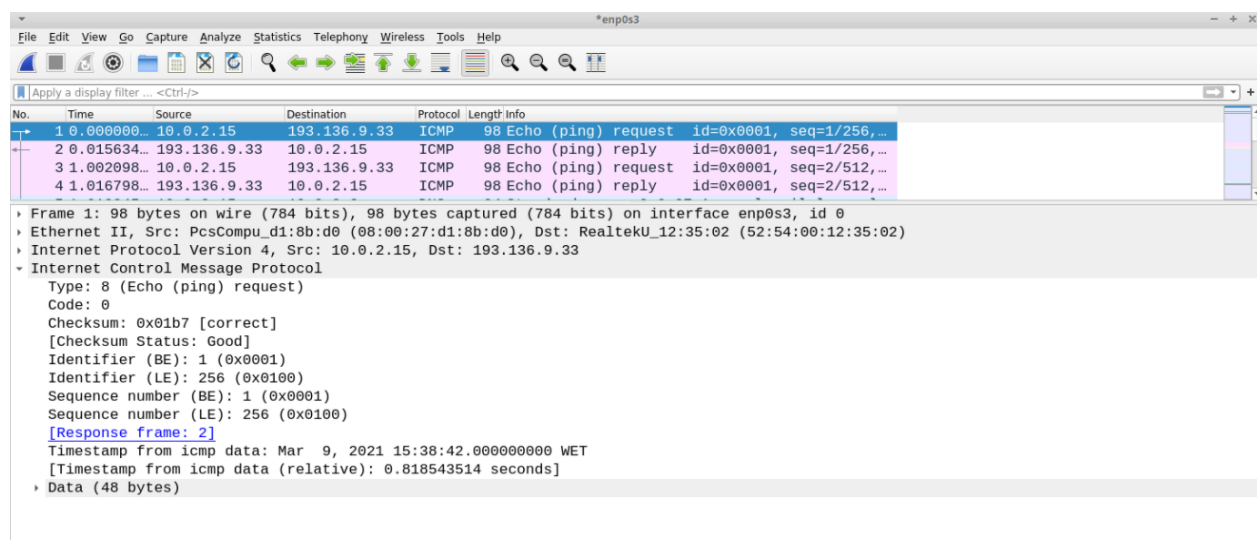


Figura 1: Ping

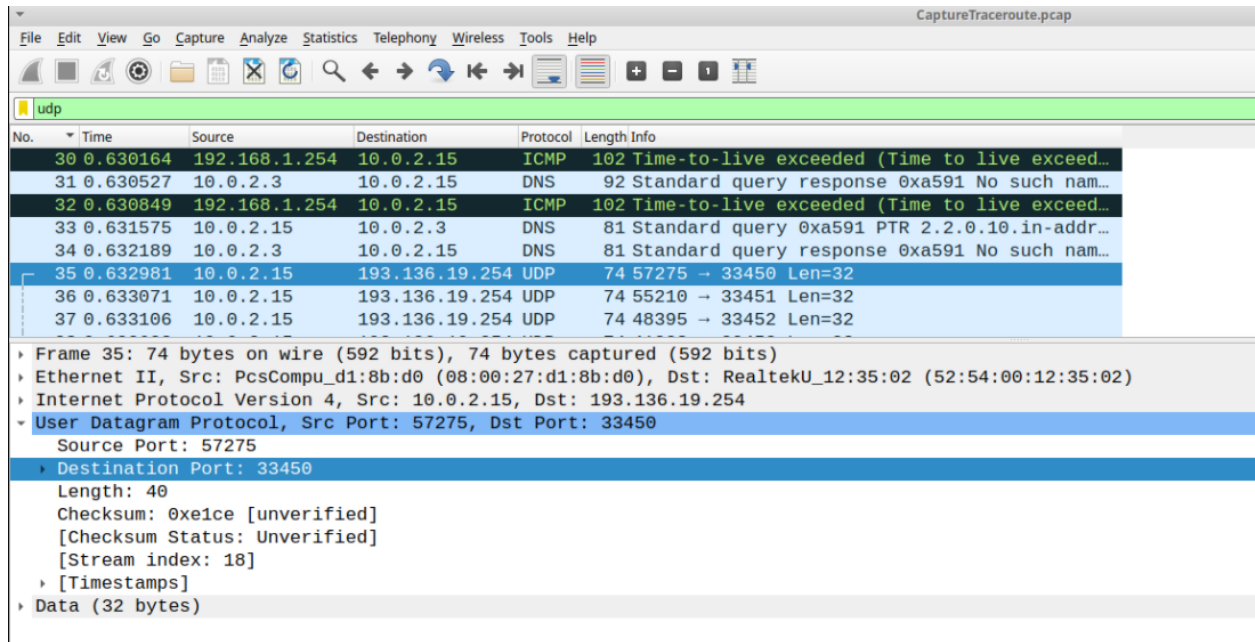


Figura 2: traceroute

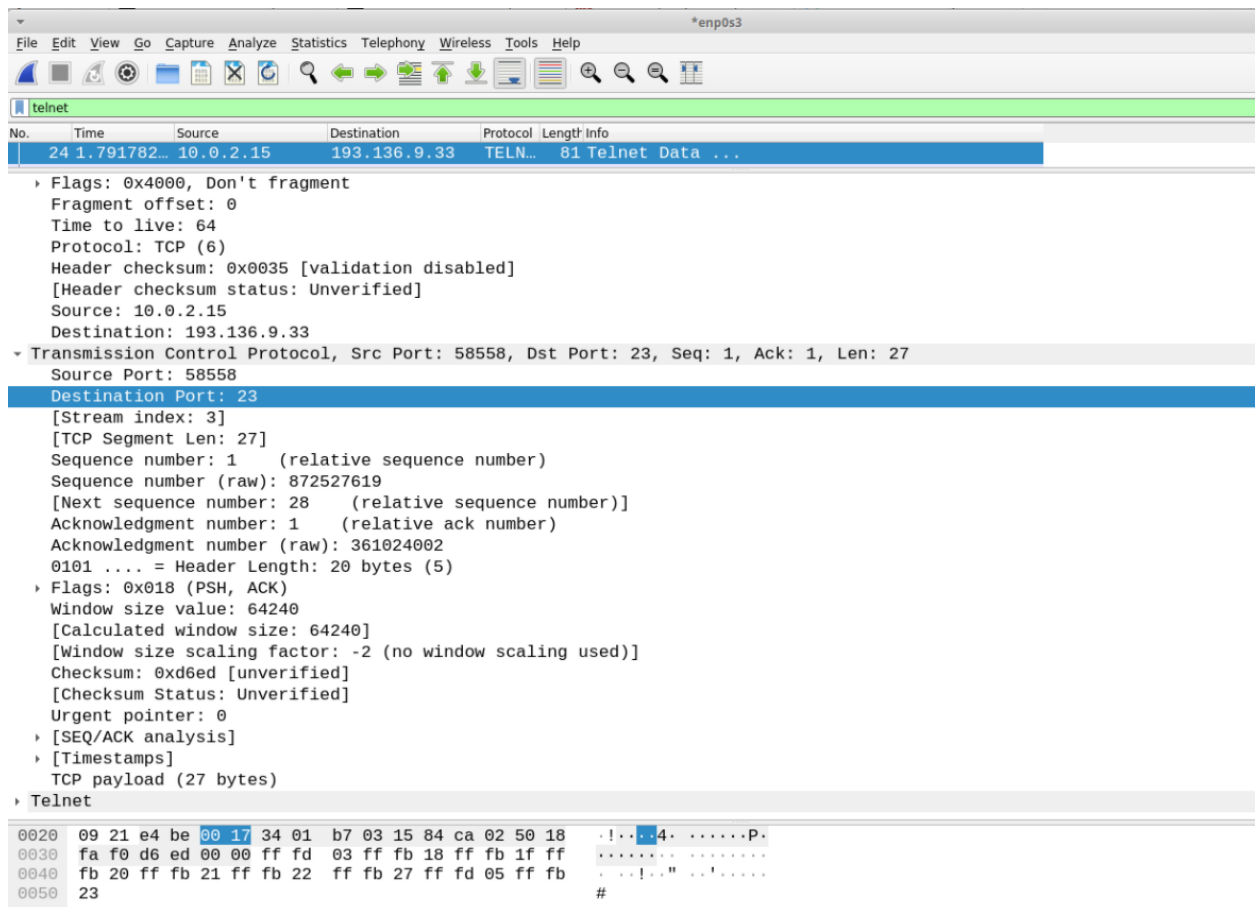


Figura 3: Telnet

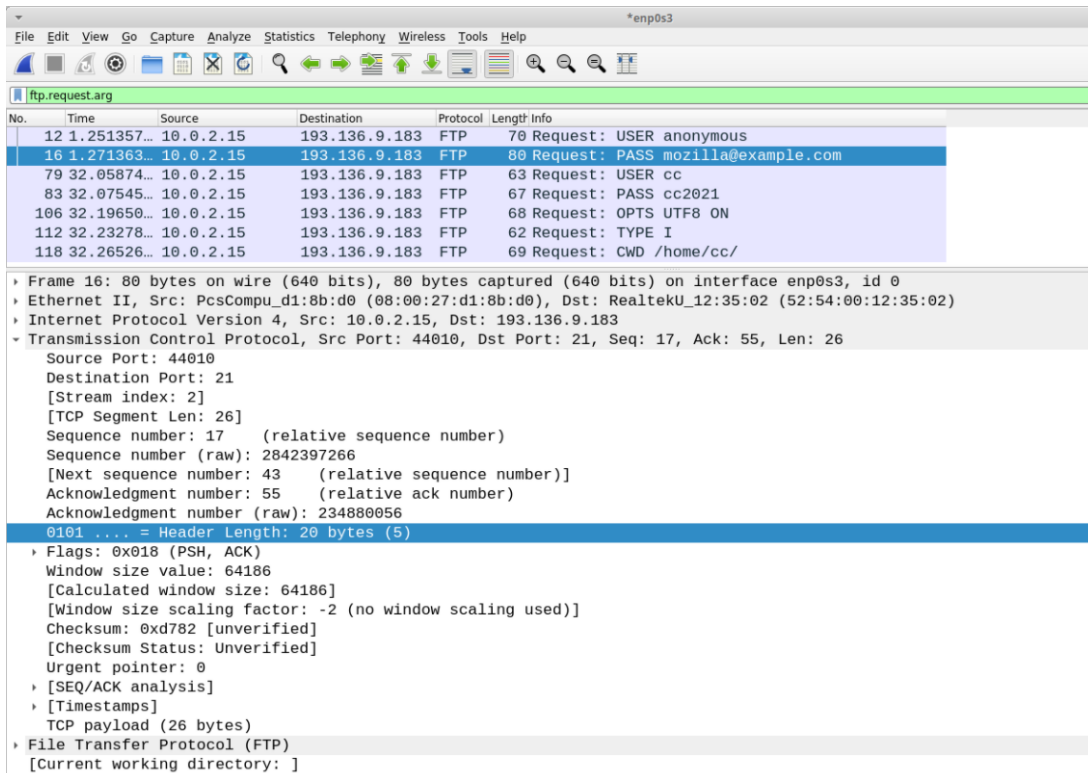


Figura 4: Ftp

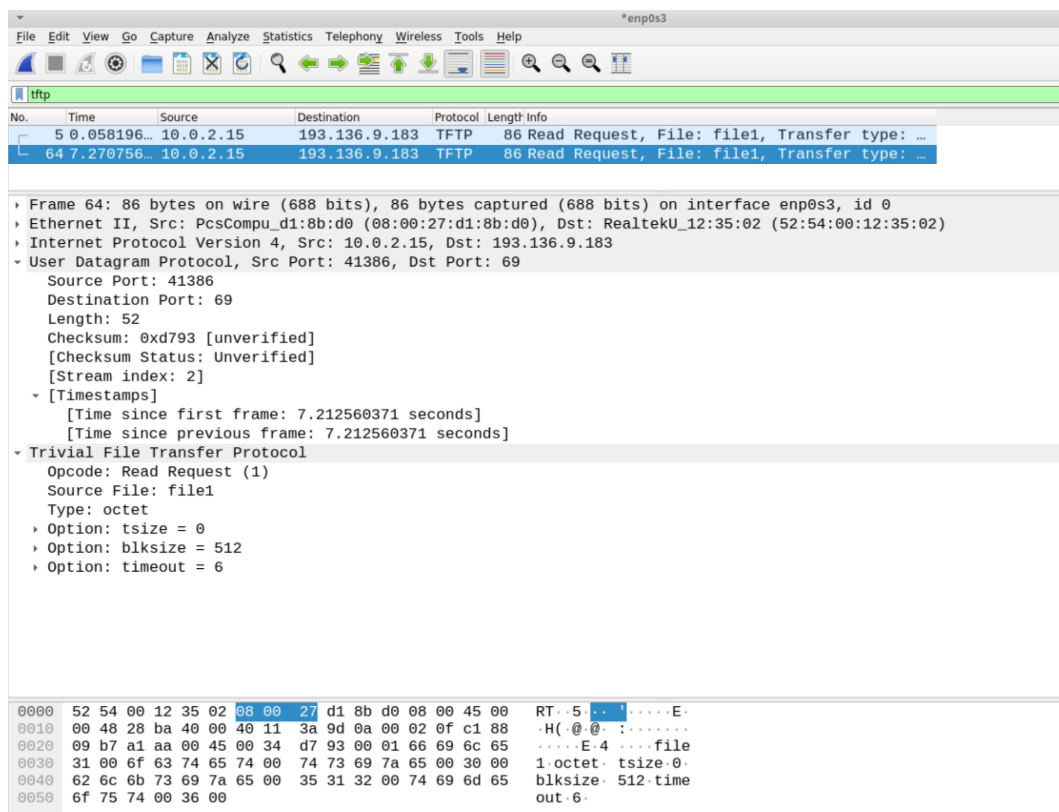


Figura 5: Tftp

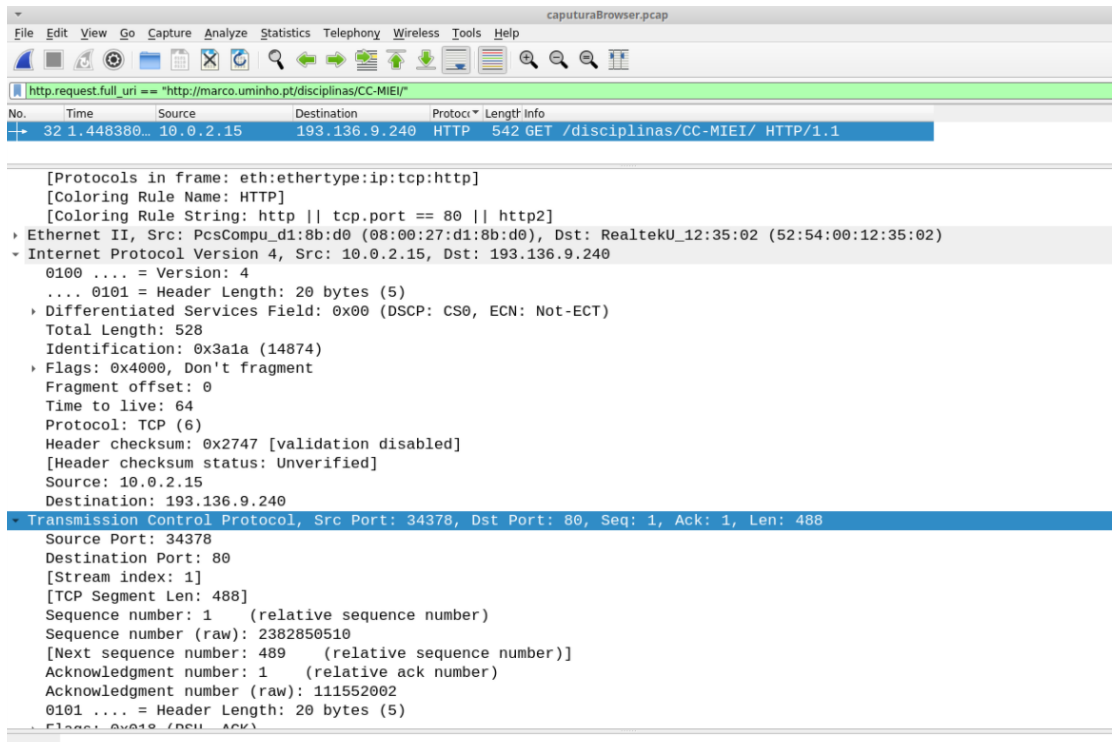


Figura 6: Browser/Http

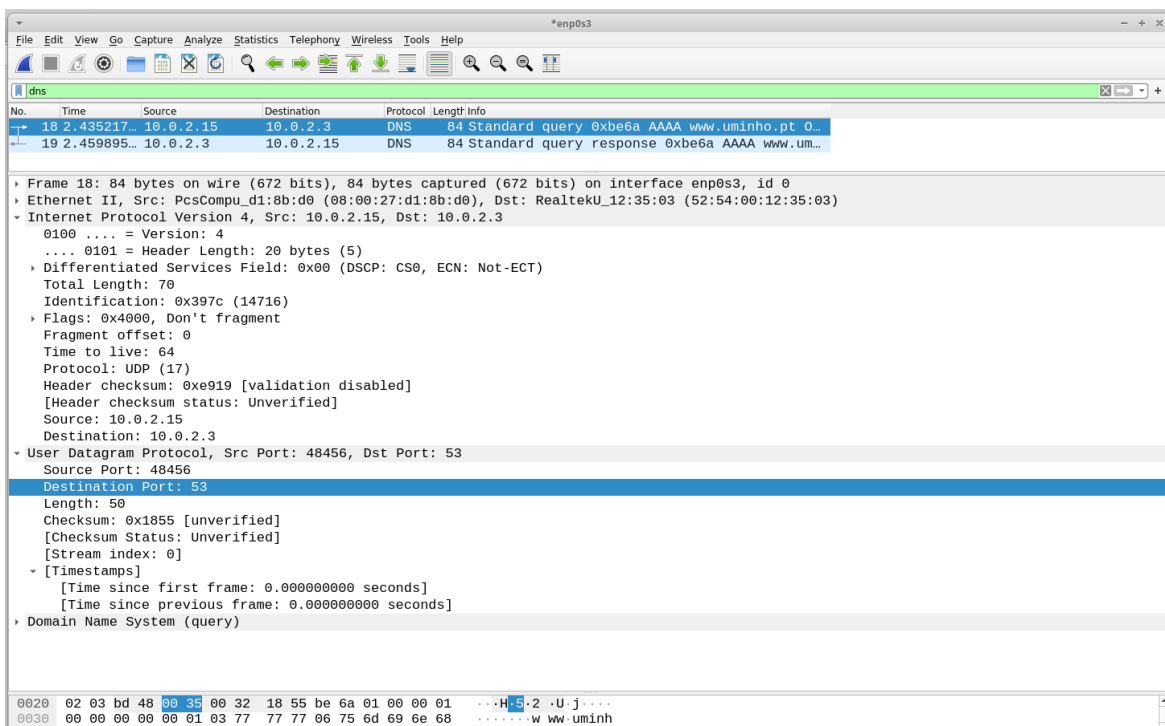


Figura 7: nslookup

*enp0s3					
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help					
ssh					
No.	Time	Source	Destination	Protocol	Length Info
52	9.006884...	193.136.9.183	10.0.2.15	SSH	110 Server: Encrypted packet (len=56)
56	15.33729...	10.0.2.15	193.136.9.183	SSH	190 Client: Encrypted packet (len=136)
58	15.40990...	193.136.9.183	10.0.2.15	SSH	78 Server: Encrypted packet (len=24)
60	15.41018...	10.0.2.15	193.136.9.183	SSH	166 Client: Encrypted packet (len=112)
Flags: 0x4000, Don't fragment Fragment offset: 0 Time to live: 64 Protocol: TCP (6) Header checksum: 0xdf4 [validation disabled] [Header checksum status: Unverified] Source: 10.0.2.15 Destination: 193.136.9.183					
Transmission Control Protocol, Src Port: 49376, Dst Port: 22, Seq: 113, Ack: 97, Len: 136 Source Port: 49376 Destination Port: 22 [Stream index: 2] [TCP Segment Len: 136] Sequence number: 113 (relative sequence number) Sequence number (raw): 274363120 [Next sequence number: 249 (relative sequence number)] Acknowledgment number: 97 (relative ack number) Acknowledgment number (raw): 413057435 0101 = Header Length: 20 bytes (5) Flags: 0x018 (PSH, ACK) Window size value: 63960 [Calculated window size: 63960] [Window size scaling factor: -1 (unknown)] Checksum: 0xd7f0 [unverified] [Checksum Status: Unverified] Urgent pointer: 0 [SEQ/ACK analysis] [Timestamps] TCP payload (136 bytes)					
SSH Protocol					
0020	09 b7 c0 e0	00 16	10 5a	72 f0 18 9e c1 9b 50 18Z r....P.
0030	f9 d8 d7 f0	00 00	45 7a	96 06 ef e8 65 20 10 efEze ..

Figura 8: ssh

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.

(Nota: a transferência por FTP envolve mais que uma conexão FTP, nomeadamente uma de controlo [ftp] e outra de dados [ftp-data]. Faça o diagrama apenas para a conexão de transferência de dados do ficheiro mais pequeno)

tcp.stream eq 1					
No.	Time	Source	Destination	Protocol	Length Info
119	145.006484	10.1.1.1	10.4.4.1	TCP	74 20 → 33773 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=2733536686 TSecr=0 WS=128
120	145.007092	10.4.4.1	10.1.1.1	TCP	74 33773 → 20 [SYN, ACK] Seq=0 Ack=1 Win=65160 Len=0 MSS=1460 SACK_PERM=1 TSval=845749609 TSecr=2733536686 WS=128
121	145.007366	10.1.1.1	10.4.4.1	TCP	66 20 → 33773 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=2733536687 TSecr=845749609
123	145.007713	10.1.1.1	10.4.4.1	FTP-DATA	192 FTP Data: 126 bytes (PORT) (LIST)
124	145.008101	10.1.1.1	10.4.4.1	TCP	66 20 → 33773 [FIN, ACK] Seq=127 Ack=1 Win=64256 Len=0 TSval=2733536688 TSecr=845749609
126	145.008141	10.4.4.1	10.1.1.1	TCP	66 33773 → 20 [ACK] Seq=1 Ack=127 Win=65152 Len=0 TSval=845749610 TSecr=2733536688
127	145.008917	10.4.4.1	10.1.1.1	TCP	66 33773 → 20 [FIN, ACK] Seq=1 Ack=128 Win=65152 Len=0 TSval=845749610 TSecr=2733536688
128	145.009723	10.1.1.1	10.4.4.1	TCP	66 20 → 33773 [ACK] Seq=128 Ack=2 Win=64256 Len=0 TSval=2733536689 TSecr=845749610

Figura 9: Transferência por FTP

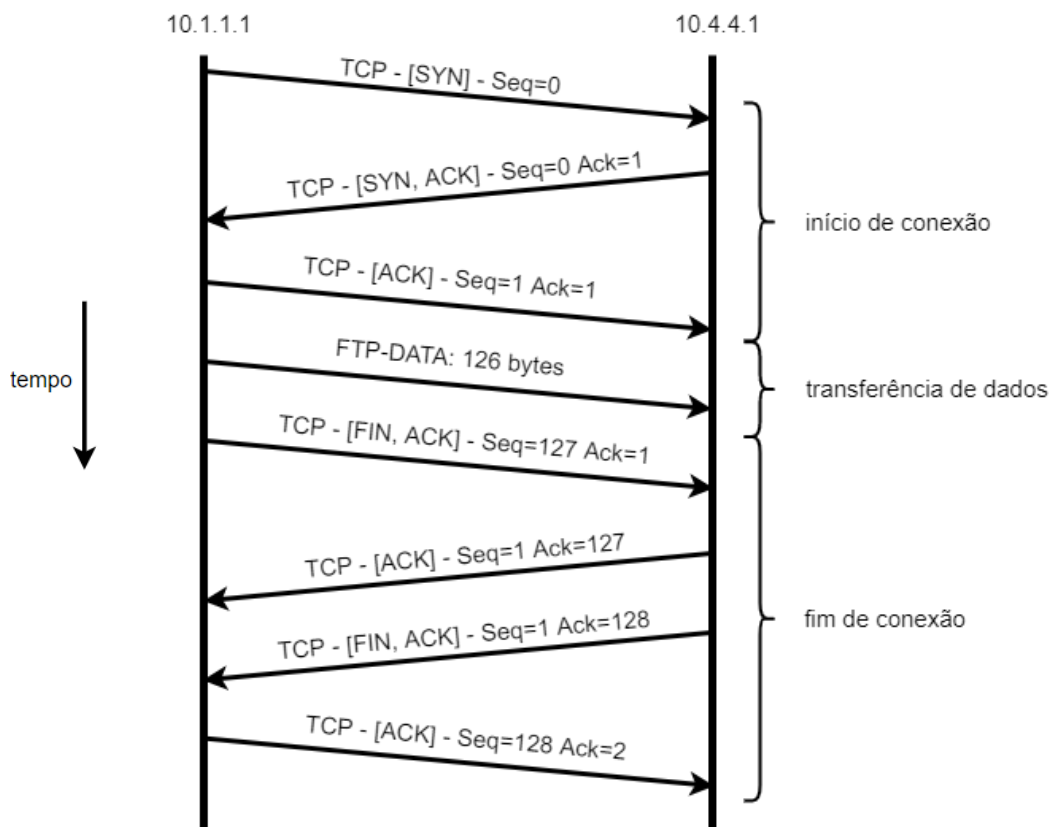


Figura 10: Diagrama Temporal da transferencia por FTP

543	888.3587...	10.1.1.254	224.0.0.5	OSPF	78 Hello Packet
544	888.4222...	10.4.4.1	10.1.1.1	TFTP	56 Read Request, File: file1, Transfer type: octet
545	888.4241...	10.1.1.1	10.4.4.1	TFTP	276 Data Packet, Block: 1 (last)
546	888.4246...	10.4.4.1	10.1.1.1	TFTP	46 Acknowledgement, Block: 1
547	890.3583...	10.1.1.254	224.0.0.5	OSPF	78 Hello Packet
548	892.3600...	10.1.1.254	224.0.0.5	OSPF	78 Hello Packet

Figura 11: Transferência por TFTP

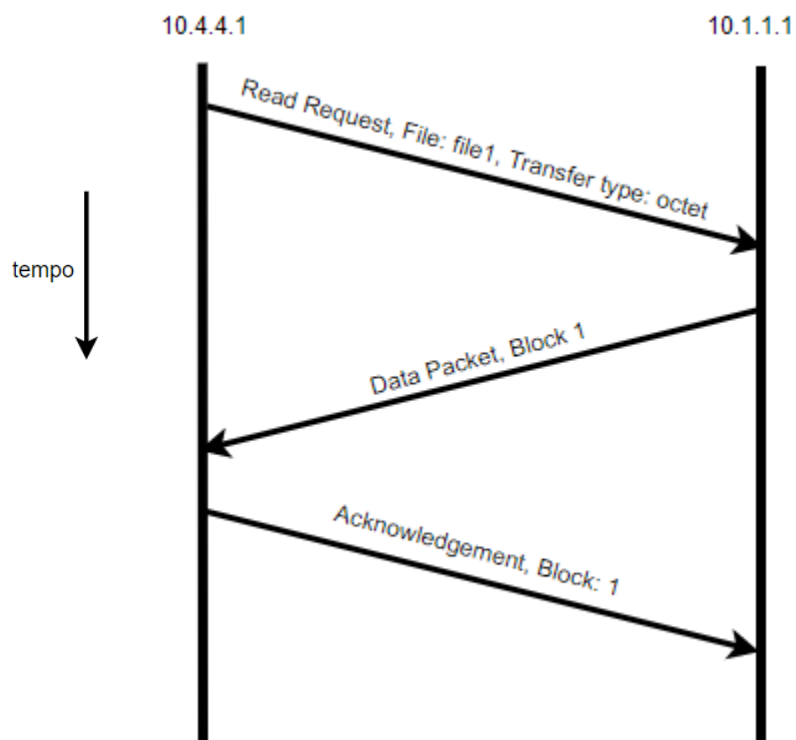


Figura 12: Diagrama Temporal da transferencia por TFTP

QUESTAO 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;

O HTTP para protocolo da camada de transporte utiliza TCP. Este protocolo possui vários esquemas de autenticação. Em contrapartida, se uma transferência de ficheiros estiver em curso, uma pessoa desde que esteja na mesma rede pode ver o conteúdo desses ficheiros antes que eles cheguem ao receptor, o que torna o HTTP inseguro.

O FTP usa como protocolo da camada de transporte o TCP e é um serviço básico de transferência de ficheiros fiável. No entanto, não adiciona nenhuma camada adicional de segurança uma vez que até se pode ver a password em texto normal num processo de autenticação. Isto torna-o bastante simples embora possua um grande leque de comandos e opções. Uma vez que possui uma elevada overhead tem alguns problemas em termos de eficiência.

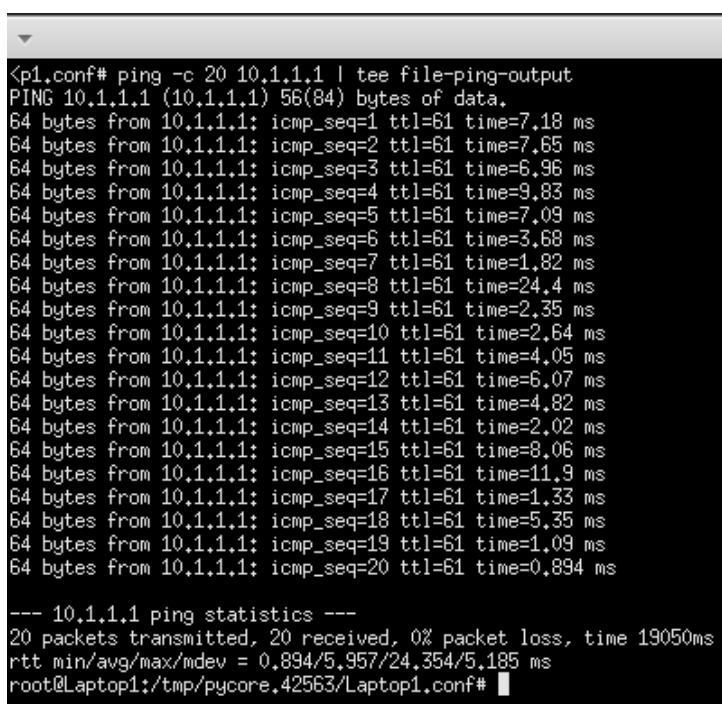
O SFTP tal como o FTP requer uma autenticação entre o cliente e o servidor e utiliza o TCP como protocolo na camada de transporte. Este é um protocolo muito seguro e mais seguro que o FTP uma vez que usa ligações SSH. Estas ligações vão ter um impacto negativo na eficiência uma vez que causam overhead.

O TFTP usa o protocolo UDP na camada de transporte o que o torna um serviço básico não fiável de transferência de ficheiros. Este protocolo é bastante simples uma vez que não implementa nenhuma segurança adicional nem tem mecanismos de autenticação. Este protocolo possui baixa overhead o que o torna muito eficiente na transmissão de ficheiros.

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).

Nota: Para responder a esta pergunta deve em primeiro lugar efetuar as transferências pedidas no enunciado, quer a partir do sistema Laptop1 na LAN4, quer do sistema Corvo a LAN3, pois só assim poderá ligar esta resposta à prática. Na topologia, o sistema Corvo tem conectividade ao Backbone através de um link que funciona com perdas, atrasos e duplicações, que é o link entre o switch SwitchLan3 e o router Router4. Nos testes podem mesmo ajustar esses parâmetros.



```
<p1.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=7.18 ms
64 bytes from 10.1.1.1: icmp_seq=2 ttl=61 time=7.65 ms
64 bytes from 10.1.1.1: icmp_seq=3 ttl=61 time=6.96 ms
64 bytes from 10.1.1.1: icmp_seq=4 ttl=61 time=9.83 ms
64 bytes from 10.1.1.1: icmp_seq=5 ttl=61 time=7.09 ms
64 bytes from 10.1.1.1: icmp_seq=6 ttl=61 time=3.68 ms
64 bytes from 10.1.1.1: icmp_seq=7 ttl=61 time=1.82 ms
64 bytes from 10.1.1.1: icmp_seq=8 ttl=61 time=24.4 ms
64 bytes from 10.1.1.1: icmp_seq=9 ttl=61 time=2.35 ms
64 bytes from 10.1.1.1: icmp_seq=10 ttl=61 time=2.64 ms
64 bytes from 10.1.1.1: icmp_seq=11 ttl=61 time=4.05 ms
64 bytes from 10.1.1.1: icmp_seq=12 ttl=61 time=6.07 ms
64 bytes from 10.1.1.1: icmp_seq=13 ttl=61 time=4.82 ms
64 bytes from 10.1.1.1: icmp_seq=14 ttl=61 time=2.02 ms
64 bytes from 10.1.1.1: icmp_seq=15 ttl=61 time=8.06 ms
64 bytes from 10.1.1.1: icmp_seq=16 ttl=61 time=11.9 ms
64 bytes from 10.1.1.1: icmp_seq=17 ttl=61 time=1.33 ms
64 bytes from 10.1.1.1: icmp_seq=18 ttl=61 time=5.35 ms
64 bytes from 10.1.1.1: icmp_seq=19 ttl=61 time=1.09 ms
64 bytes from 10.1.1.1: icmp_seq=20 ttl=61 time=0.894 ms

--- 10.1.1.1 ping statistics ---
20 packets transmitted, 20 received, 0% packet loss, time 19050ms
rtt min/avg/max/mdev = 0.894/5.957/24.354/5.185 ms
root@Laptop1:/tmp/pycore.42563/Laptop1.conf#
```

Figura 13: Conectividade do 'Laptop 1'

Na figura 13, podemos ver que não houve packet loss nem duplicação de packets, contudo na figura 14 podemos ver que na transferência de dados para o servidor 1 a partir do laptop 'Corvo' existe packet loss de 5% e num conjunto de 20 transferências, 4 delas são duplicadas o que equivale a 20% de duplicação. Isto deve-se ao facto que a ligação entre o router 4 e a switchLan3 que liga ao laptop 'Corvo' tem uma packet Loss de 5% e um packet duplication (DUP) de 10%.

```
<vo.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=13.3 ms
64 bytes from 10.1.1.1: icmp_seq=1 ttl=61 time=13.3 ms (DUP!)
64 bytes from 10.1.1.1: icmp_seq=2 ttl=61 time=5.24 ms
64 bytes from 10.1.1.1: icmp_seq=4 ttl=61 time=5.41 ms
64 bytes from 10.1.1.1: icmp_seq=4 ttl=61 time=5.41 ms (DUP!)
64 bytes from 10.1.1.1: icmp_seq=5 ttl=61 time=5.54 ms
64 bytes from 10.1.1.1: icmp_seq=5 ttl=61 time=5.55 ms (DUP!)
64 bytes from 10.1.1.1: icmp_seq=6 ttl=61 time=5.56 ms
64 bytes from 10.1.1.1: icmp_seq=7 ttl=61 time=5.51 ms
64 bytes from 10.1.1.1: icmp_seq=8 ttl=61 time=5.57 ms
64 bytes from 10.1.1.1: icmp_seq=9 ttl=61 time=5.55 ms
64 bytes from 10.1.1.1: icmp_seq=10 ttl=61 time=5.50 ms
64 bytes from 10.1.1.1: icmp_seq=11 ttl=61 time=5.56 ms
64 bytes from 10.1.1.1: icmp_seq=12 ttl=61 time=5.40 ms
64 bytes from 10.1.1.1: icmp_seq=13 ttl=61 time=5.38 ms
64 bytes from 10.1.1.1: icmp_seq=14 ttl=61 time=5.36 ms
64 bytes from 10.1.1.1: icmp_seq=15 ttl=61 time=5.51 ms
64 bytes from 10.1.1.1: icmp_seq=16 ttl=61 time=5.52 ms
64 bytes from 10.1.1.1: icmp_seq=17 ttl=61 time=8.97 ms
64 bytes from 10.1.1.1: icmp_seq=18 ttl=61 time=7.76 ms
64 bytes from 10.1.1.1: icmp_seq=19 ttl=61 time=9.23 ms
64 bytes from 10.1.1.1: icmp_seq=20 ttl=61 time=9.65 ms
64 bytes from 10.1.1.1: icmp_seq=20 ttl=61 time=9.66 ms (DUP!)

--- 10.1.1.1 ping statistics ---
20 packets transmitted, 19 received, +4 duplicates, 5% packet loss, time 19059ms
rtt min/avg/max/mdev = 5.235/6.934/13.319/2.472 ms
root@Corvo:/tmp/pycore.42563/Corvo.conf#
```

Figura 14: Conectividade do 'Laptop Corvo'

Estas duas características podem causar problemas na transferência de dados e tem influência no nível de aplicação. A packet duplication (DUP) pode não causar grande problema na execução de uma aplicação, pelo menos comparado com os efeitos que a packet loss tem, visto que na camada TCP são descartados os pacotes duplicados. Assim, o único efeito negativo será uma pequena redução da largura de banda disponível. O packet loss causa uma diminuição na taxa de transferência devido à perda de informação, contudo o protocolo TCP consegue detectar packet loss e efectuar retransmissões, à custa de um aumento de latência. Devido a estas duas características na figura 14 podemos ver que a média do return time (rtt), isto é, o tempo de obtenção de uma respostas é mais lento quando comparado com o laptop 1 da figura 13.

Concluimos assim, que de forma a ter uma conexão viável, segura e rápida tanto na camada de transporte como na aplicação é necessário que não haja latências devido às transmissão de dados e que não haja perda de informação.

CONCLUSÃO

Este primeiro trabalho serviu como consolidação da matéria lecionada nas aulas teóricas acerca de protocolos na camada de transporte.

Dado como terminado este relatório, foi possível ao longo de todo este trabalho prático adquirir e consolidar conhecimento sobre diversos protocolos como o DNS, TCP, UDP, TELNET entre outros. Além disso, foi também possível conhecer vários servidores como SFTP, FTP, TFTP e HTTP. Foi possível também através de experiências realizadas no programa Wireshark e Core Network Emulator distinguir e comparar aplicações de transferências de ficheiros. Finalmente, foi também possível adquirir conhecimento sobre camadas de aplicação e transporte e de como são influenciadas pela packet loss e packet duplication.