Comunicações por Computador

Trabalho Prático Nº.1 - Protocolos da Camada de Transporte

Eduardo Coelho, Henrique Neto, Júlio Alves
e-mail: {a89616,a89618,a89468}@alunos.uminho.pt

16 de março de 2021

Questões e Respostas

1. 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	—	—	—	—
traceroute	—	UDP	33447	8
telnet	TELNET	TCP	23	20
ftp	FTP	TCP	58788	20
Tftp	TFTP	UDP	69	8
browser/http	HTTP	TCP	80	20
nslookup	DNS	UDP	60615	8
ssh	SSHv2	TCP	34980	20

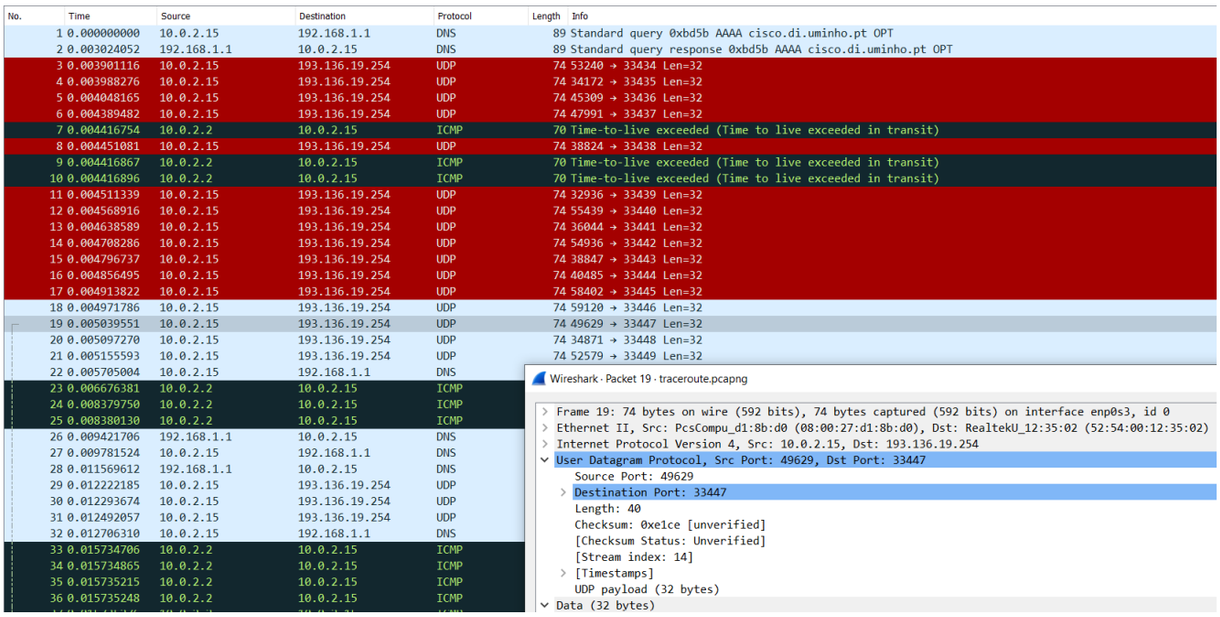
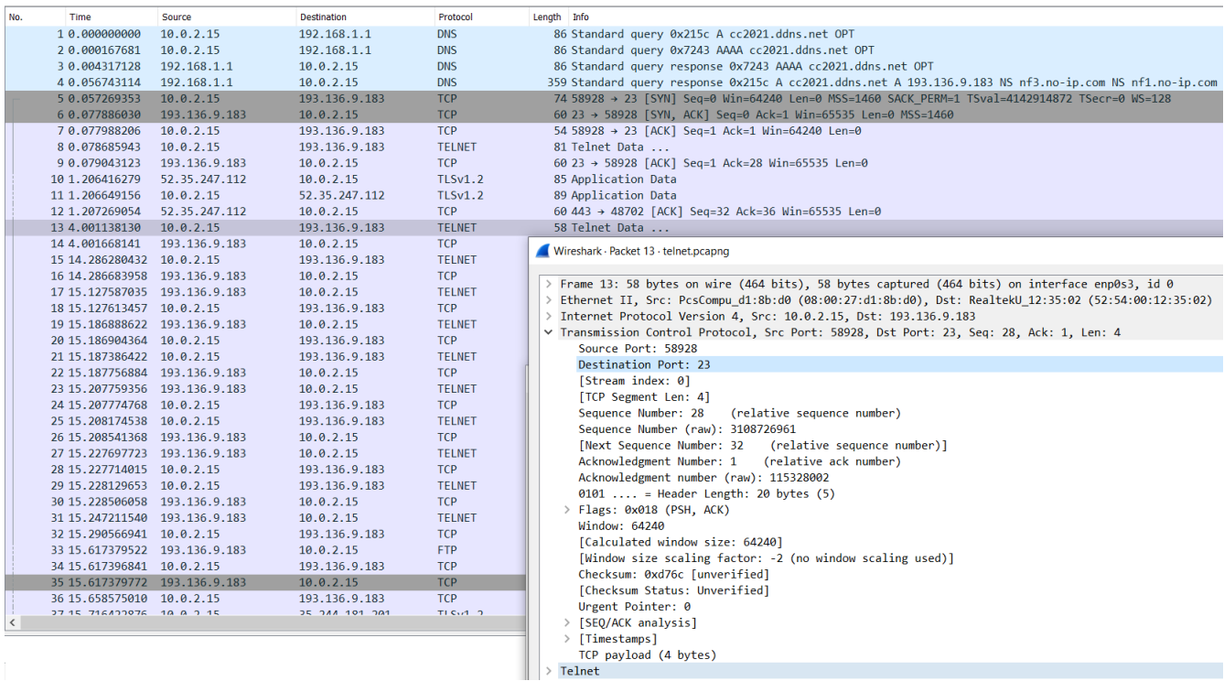


Figura 1: Traceroute

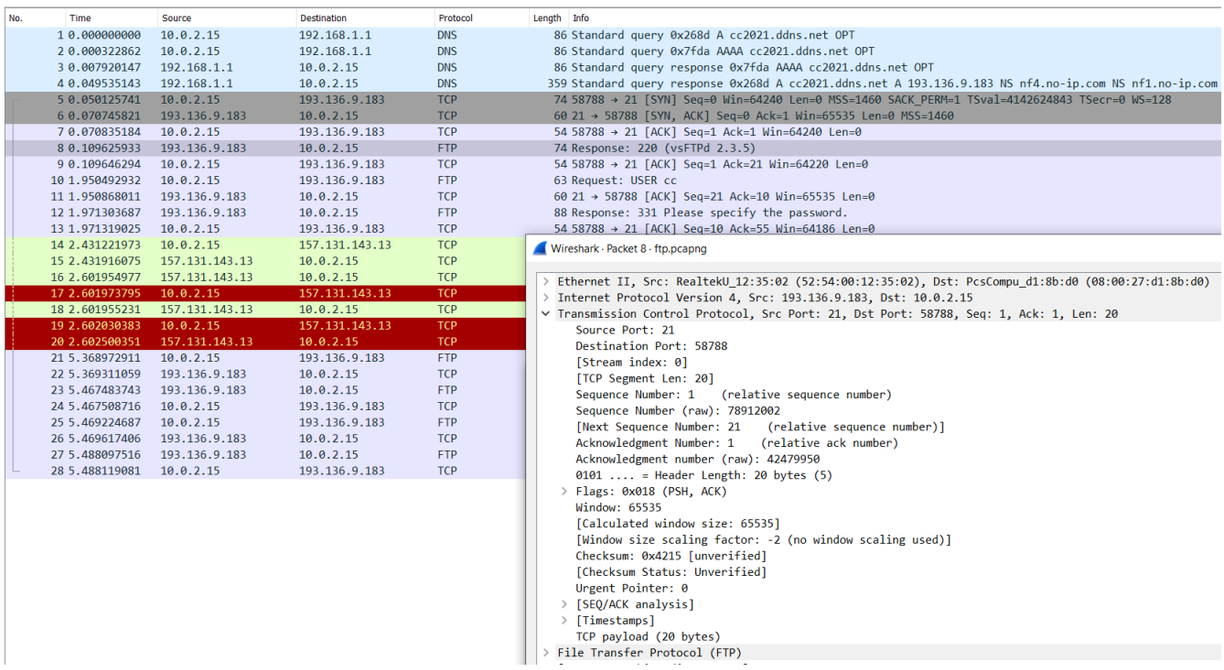


No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	10.0.2.15	192.168.1.1	DNS	86	Standard query 0x215c A cc2021.ddns.net OPT
2	0.000167681	10.0.2.15	192.168.1.1	DNS	86	Standard query 0x7243 AAAA cc2021.ddns.net OPT
3	0.004317128	192.168.1.1	10.0.2.15	DNS	86	Standard query response 0x7243 AAAA cc2021.ddns.net OPT
4	0.056743114	192.168.1.1	10.0.2.15	DNS	359	Standard query response 0x215c A cc2021.ddns.net A 193.136.9.183 NS nf3.no-ip.com NS nf1.no-ip.com
5	0.057269353	10.0.2.15	193.136.9.183	TCP	74	58928 → 23 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=4142914872 TSecr=0 WS=128
6	0.077886030	193.136.9.183	10.0.2.15	TCP	60	23 → 58928 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1460
7	0.077988206	10.0.2.15	193.136.9.183	TCP	54	58928 → 23 [ACK] Seq=1 Ack=1 Win=64240 Len=0
8	0.078685943	10.0.2.15	193.136.9.183	TELNET	81	Telnet Data ...
9	0.079043123	193.136.9.183	10.0.2.15	TCP	60	23 → 58928 [ACK] Seq=1 Ack=28 Win=65535 Len=0
10	1.206416279	52.35.247.112	10.0.2.15	TLSv1.2	85	Application Data
11	1.206649156	10.0.2.15	52.35.247.112	TLSv1.2	89	Application Data
12	1.207269054	52.35.247.112	10.0.2.15	TCP	60	443 → 48702 [ACK] Seq=32 Ack=36 Win=65535 Len=0
13	4.001138130	10.0.2.15	193.136.9.183	TELNET	58	Telnet Data ...
14	4.001668141	193.136.9.183	10.0.2.15	TCP		
15	14.286280432	10.0.2.15	193.136.9.183	TELNET		
16	14.286683958	193.136.9.183	10.0.2.15	TCP		
17	15.127587035	193.136.9.183	10.0.2.15	TELNET		
18	15.127613457	10.0.2.15	193.136.9.183	TCP		
19	15.186888622	193.136.9.183	10.0.2.15	TELNET		
20	15.186904364	10.0.2.15	193.136.9.183	TCP		
21	15.187386422	10.0.2.15	193.136.9.183	TELNET		
22	15.187756884	193.136.9.183	10.0.2.15	TCP		
23	15.207759356	193.136.9.183	10.0.2.15	TELNET		
24	15.207774768	10.0.2.15	193.136.9.183	TCP		
25	15.208174538	10.0.2.15	193.136.9.183	TELNET		
26	15.208541368	193.136.9.183	10.0.2.15	TCP		
27	15.227697723	193.136.9.183	10.0.2.15	TELNET		
28	15.227714015	10.0.2.15	193.136.9.183	TCP		
29	15.228129653	10.0.2.15	193.136.9.183	TELNET		
30	15.228506058	193.136.9.183	10.0.2.15	TCP		
31	15.247211540	193.136.9.183	10.0.2.15	TELNET		
32	15.290566941	10.0.2.15	193.136.9.183	TCP		
33	15.617379522	193.136.9.183	10.0.2.15	FTP		
34	15.617396841	10.0.2.15	193.136.9.183	TCP		
35	15.617379772	193.136.9.183	10.0.2.15	TCP		
36	15.658575010	10.0.2.15	193.136.9.183	TCP		

Wireshark · Packet 13 · telnet.pcapng

```
> Frame 13: 58 bytes on wire (464 bits), 58 bytes captured (464 bits) on interface enp0s3, id 0
> Ethernet II, Src: PcsCompu_d1:8b:d0 (08:00:27:d1:8b:d0), Dst: RealtekU_12:35:02 (52:54:00:12:35:02)
> Internet Protocol Version 4, Src: 10.0.2.15, Dst: 193.136.9.183
v Transmission Control Protocol, Src Port: 58928, Dst Port: 23, Seq: 28, Ack: 1, Len: 4
    Source Port: 58928
    Destination Port: 23
    [Stream index: 0]
    [TCP Segment Len: 4]
    Sequence Number: 28    (relative sequence number)
    Sequence Number (raw): 3108726961
    [Next Sequence Number: 32    (relative sequence number)]
    Acknowledgment Number: 1    (relative ack number)
    Acknowledgment number (raw): 115328002
    0101 .... = Header Length: 20 bytes (5)
  > Flags: 0x018 (PSH, ACK)
    Window: 64240
    [Calculated window size: 64240]
    [Window size scaling factor: -2 (no window scaling used)]
    Checksum: 0xd76c [unverified]
    [Checksum Status: Unverified]
    Urgent Pointer: 0
  > [SEQ/ACK analysis]
  > [Timestamps]
    TCP payload (4 bytes)
> Telnet
```

Figura 2: Telnet



No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	10.0.2.15	192.168.1.1	DNS	86	Standard query 0x268d A cc2021.ddns.net OPT
2	0.000322862	10.0.2.15	192.168.1.1	DNS	86	Standard query 0x7fda AAAA cc2021.ddns.net OPT
3	0.007920147	192.168.1.1	10.0.2.15	DNS	86	Standard query response 0x7fda AAAA cc2021.ddns.net OPT
4	0.049535143	192.168.1.1	10.0.2.15	DNS	359	Standard query response 0x268d A cc2021.ddns.net A 193.136.9.183 NS nf4.no-ip.com NS nf1.no-ip.com
5	0.050125741	10.0.2.15	193.136.9.183	TCP	74	58788 → 21 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=4142624843 TSecr=0 WS=128
6	0.070745821	193.136.9.183	10.0.2.15	TCP	60	21 → 58788 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1460
7	0.070835184	10.0.2.15	193.136.9.183	TCP	54	58788 → 21 [ACK] Seq=1 Ack=1 Win=64240 Len=0
8	0.109625933	193.136.9.183	10.0.2.15	FTP	74	Response: 220 (vsFTPd 2.3.5)
9	0.109646294	10.0.2.15	193.136.9.183	TCP	54	58788 → 21 [ACK] Seq=1 Ack=21 Win=64220 Len=0
10	1.950492932	10.0.2.15	193.136.9.183	FTP	63	Request: USER cc
11	1.950868011	193.136.9.183	10.0.2.15	TCP	60	21 → 58788 [ACK] Seq=21 Ack=10 Win=65535 Len=0
12	1.971303687	193.136.9.183	10.0.2.15	FTP	88	Response: 331 Please specify the password.
13	1.971319025	10.0.2.15	193.136.9.183	TCP	54	58788 → 21 [ACK] Seq=10 Ack=55 Win=64186 Len=0
14	2.431221973	10.0.2.15	157.131.143.13	TCP		
15	2.431916075	157.131.143.13	10.0.2.15	TCP		
16	2.601954977	157.131.143.13	10.0.2.15	TCP		
17	2.601973795	10.0.2.15	157.131.143.13	TCP		
18	2.601955231	157.131.143.13	10.0.2.15	TCP		
19	2.602030383	10.0.2.15	157.131.143.13	TCP		
20	2.602500351	157.131.143.13	10.0.2.15	TCP		
21	5.368972911	10.0.2.15	193.136.9.183	FTP		
22	5.369311059	193.136.9.183	10.0.2.15	TCP		
23	5.467483743	193.136.9.183	10.0.2.15	FTP		
24	5.467508716	10.0.2.15	193.136.9.183	TCP		
25	5.469224687	10.0.2.15	193.136.9.183	FTP		
26	5.469617406	193.136.9.183	10.0.2.15	TCP		
27	5.488097516	193.136.9.183	10.0.2.15	FTP		
28	5.488119081	10.0.2.15	193.136.9.183	TCP		

Wireshark · Packet 8 · ftp.pcapng

```
> Ethernet II, Src: RealtekU_12:35:02 (52:54:00:12:35:02), Dst: PcsCompu_d1:8b:d0 (08:00:27:d1:8b:d0)
> Internet Protocol Version 4, Src: 193.136.9.183, Dst: 10.0.2.15
v Transmission Control Protocol, Src Port: 21, Dst Port: 58788, Seq: 1, Ack: 1, Len: 20
    Source Port: 21
    Destination Port: 58788
    [Stream index: 0]
    [TCP Segment Len: 20]
    Sequence Number: 1    (relative sequence number)
    Sequence Number (raw): 78912002
    [Next Sequence Number: 21    (relative sequence number)]
    Acknowledgment Number: 1    (relative ack number)
    Acknowledgment number (raw): 42479950
    0101 .... = Header Length: 20 bytes (5)
  > Flags: 0x018 (PSH, ACK)
    Window: 65535
    [Calculated window size: 65535]
    [Window size scaling factor: -2 (no window scaling used)]
    Checksum: 0x4215 [unverified]
    [Checksum Status: Unverified]
    Urgent Pointer: 0
  > [SEQ/ACK analysis]
  > [Timestamps]
    TCP payload (20 bytes)
> File Transfer Protocol (FTP)
```

Figura 3: FTP

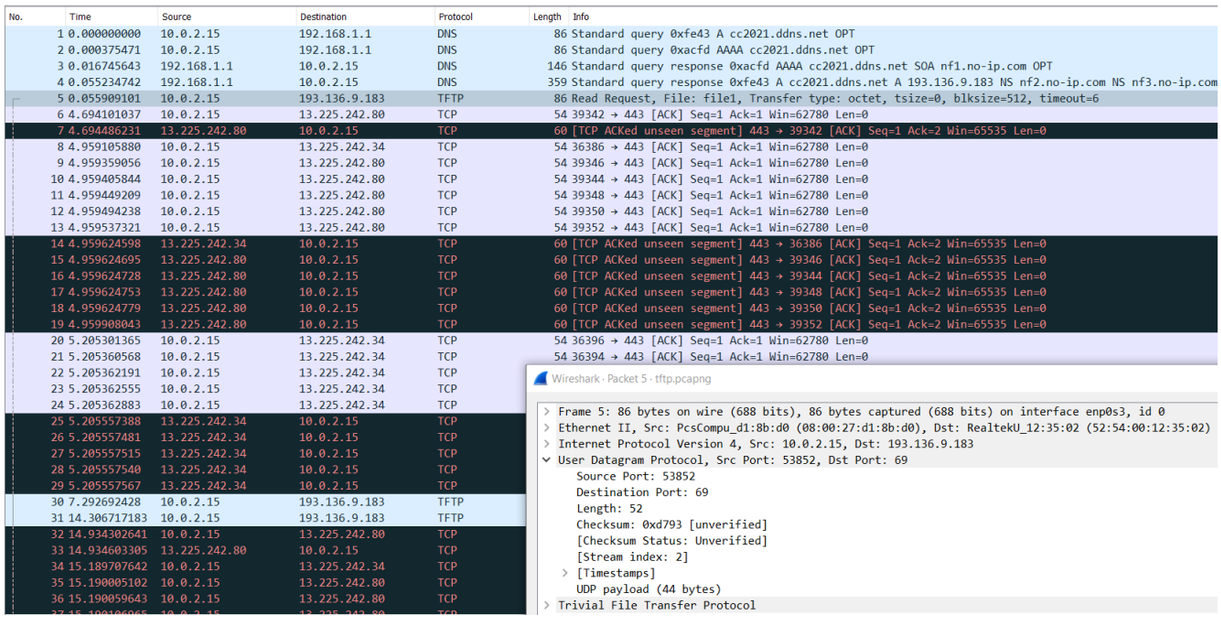


Figura 4: TFTP

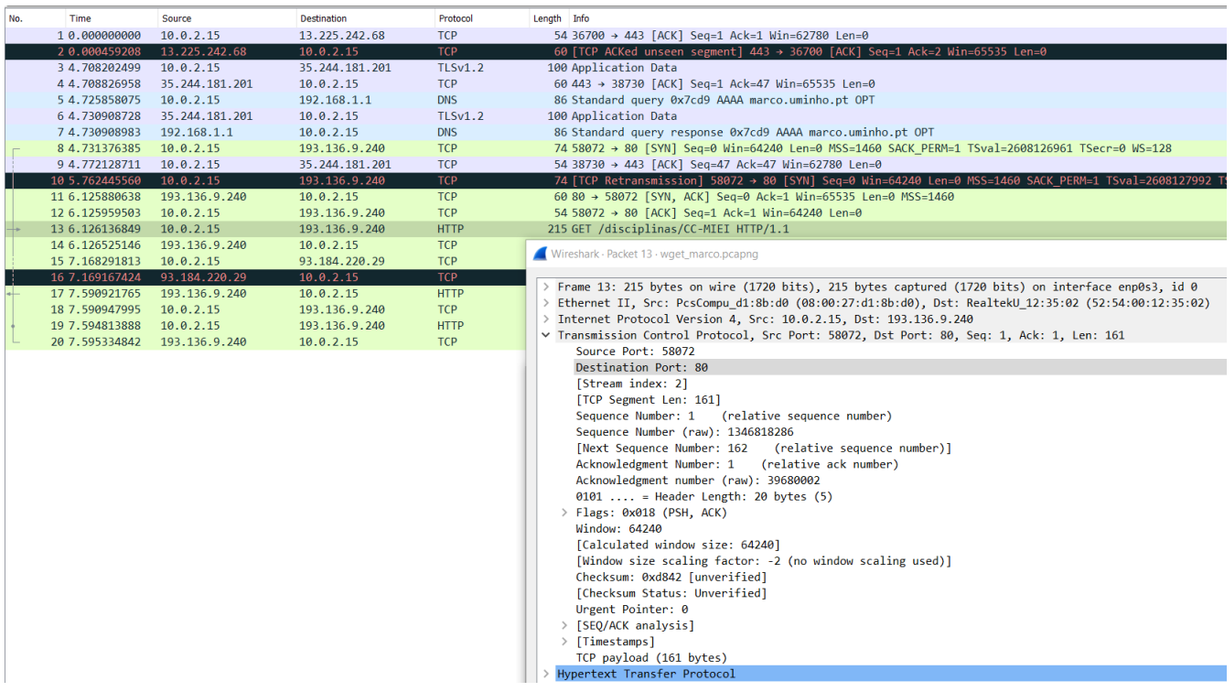


Figura 5: Browser/HTTP

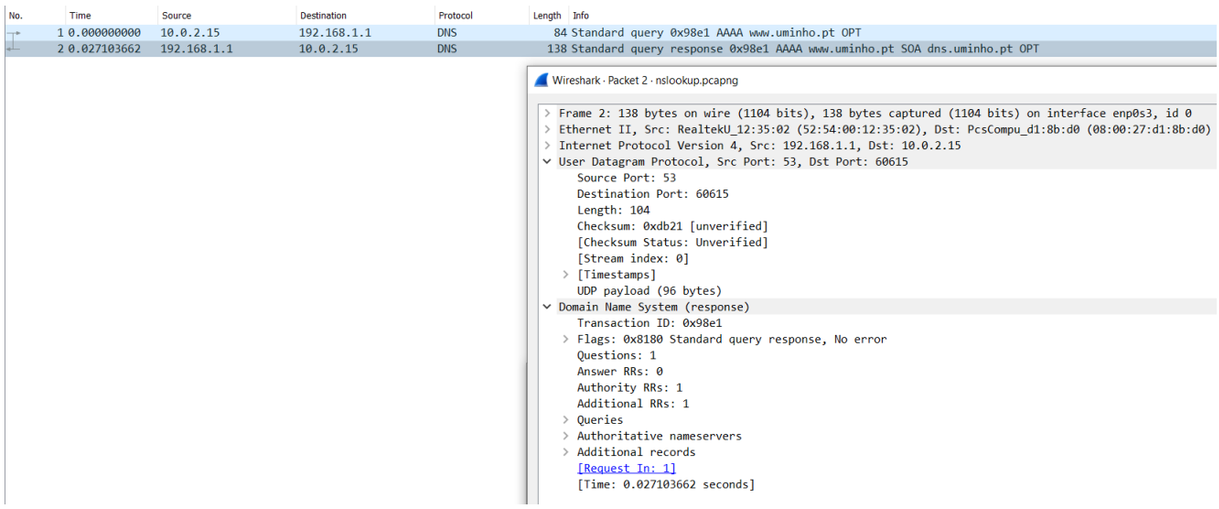


Figura 6: nslookup

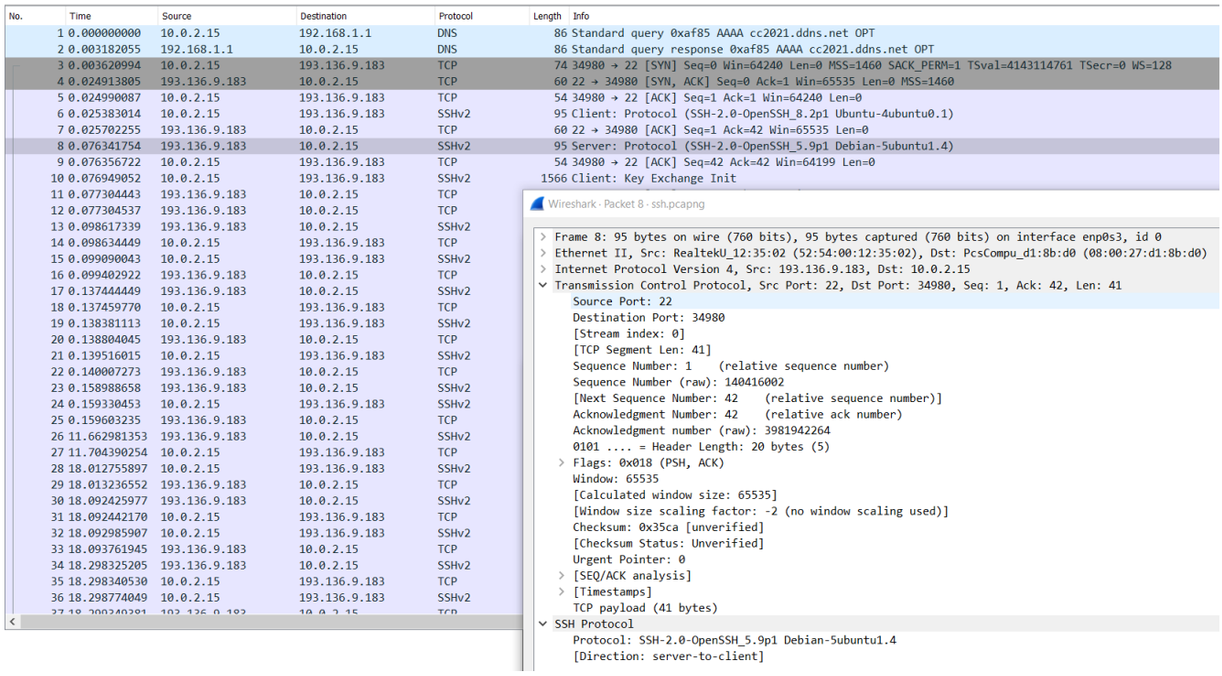


Figura 7: SSH

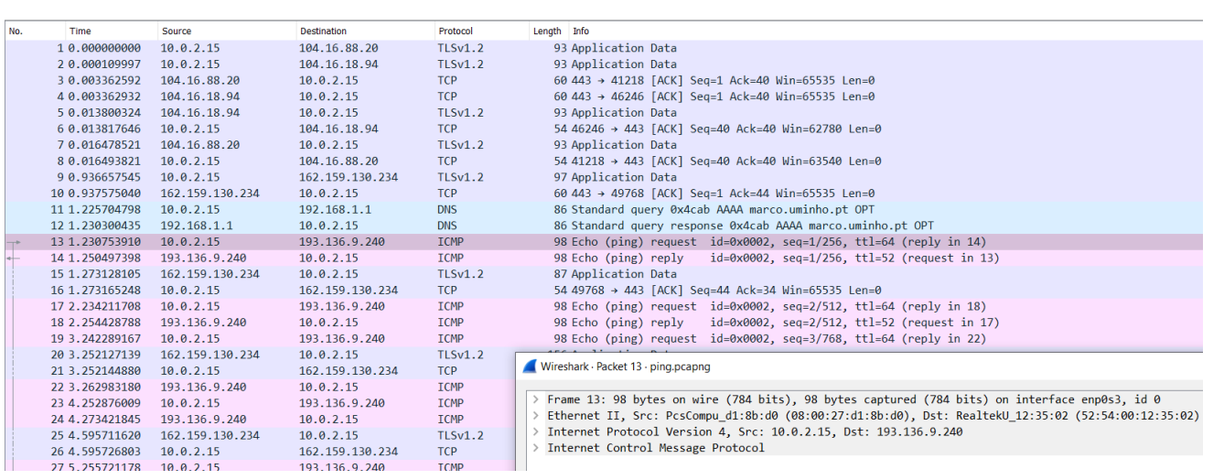


Figura 8: Ping

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)

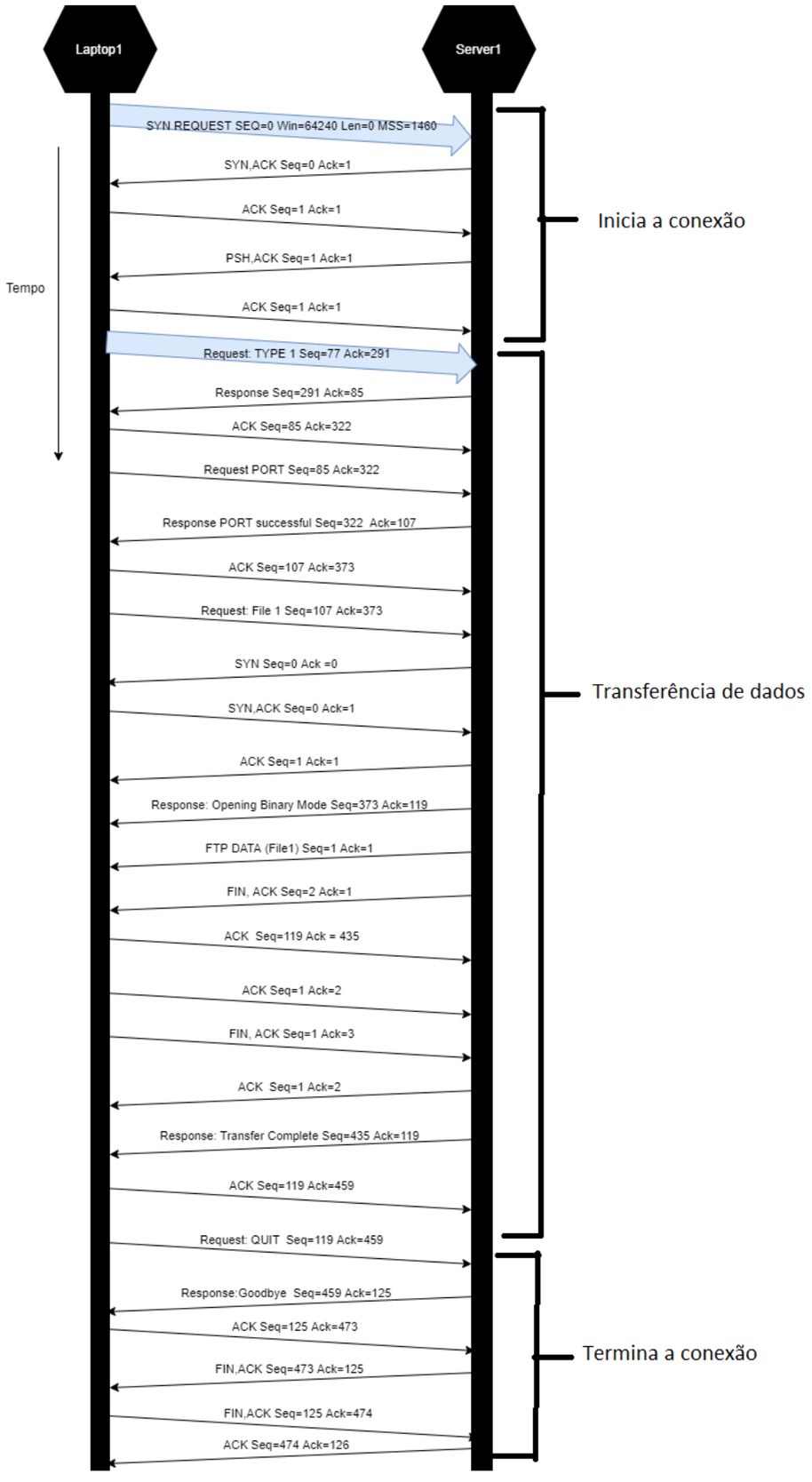


Figura 9: Diagrama Temporal da transferência do ficheiro file1 por FTP (Protocolo TCP)

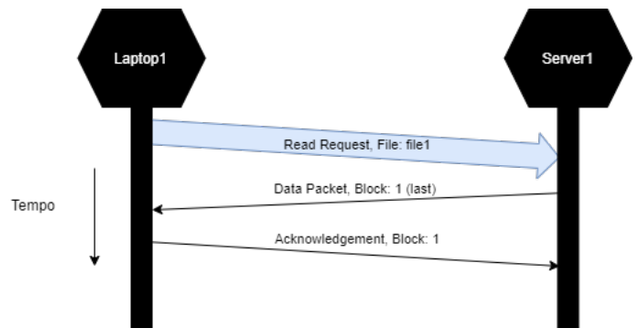


Figura 10: Diagrama Temporal da transferência do ficheiro file1 por TFTP (Protocolo UDP)

3. Com base nas experiências realizadas, distinga e compare sucintamente as quatro aplicações de transferência de ficheiros que usou nos seguintes pontos a (i) uso da camada de transporte; (ii) eficiência na transferência; (iii) complexidade; (iv) segurança;

FTP

A aplicação *FTP* (*File Transfer Protocol*) faz recurso ao protocolo de transporte *TCP* (por norma nas portas 20 e 21) para efetuar as trocas de informação. Desta forma antes que sejam efectuadas transferências é primeiro estabelecida uma conexão na porta 21, onde são adicionalmente confirmadas as credenciais do utilizador e transmitidos os seus pedidos.

Para se realizar uma transmissão de dados o cliente da aplicação começa por fazer um pedido de controlo *FTP* para indicar ao servidor que pretende fazer uma transferência de um dado tipo de dados. Posteriormente o servidor envia uma mensagem de resposta ao cliente que prossegue com um pedido para abrir um segundo canal de transmissão *TCP* (na porta 20) para efetuar a transferência. Caso tenha disponibilidade o servidor responde ao cliente com a porta do novo canal que transmite por fim o identificador do ficheiro que quer transferir. De seguida o servidor inicia a transmissão do ficheiro, começando por enviar no segundo canal uma mensagem a indicar tipo de dados especificado na fase de *request* seguido de uma série de mensagens *FTP-DATA* contendo o ficheiro em si. Por fim quando a transmissão acaba, o transmissor termina a conexão no segundo canal *TCP* e assim que esta operação for confirmada envia uma mensagem a confirmar o fim da transmissão.

Desta forma, podemos concluir que este método de transferência é bastante complexo e permite uma transmissão diversificada de dados e que devido a usar um segundo canal o utilizador pode continuar a trabalhar no sistema remoto enquanto ocorrem trocas de ficheiros. Porém ao implicar uma grande troca de mensagens entre o servidor e cliente (quer pelas mensagens de controlo *TCP* quer pela abertura de um novo canal *TCP*), é muito ineficiente pode provocar atrasos significativos durante as transferências principalmente se as conexões estiverem sujeitas a perdas de informação onde seja necessário fazer várias retransmissões.Por outro lado, como nenhuma da informação é encriptada ou protegida os intervenientes estão sujeitos a ataques de *middleman* que pode ser bastante perigoso visto que existem várias trocas de credenciais durante a conexão.

SFTP

O protocolo *SFTP* (*SSH File Transfer Protocol*) é um protocolo de transferência de ficheiros seguro que se baseia no protocolo de aplicação *SSH* e no protocolo de transporte *TCP* (por norma com recurso à porta 22).

Baseia-se bastante no processo do protocolo *FTP*, no entanto devido ao uso de encriptação *SSH*, há uma maior segurança comparativamente ao mesmo. Por outro lado a troca de informação é feita apenas em um canal que diminui bastante a quantidade de mensagens envolvidas tornando-o mais eficiente do que o *FTP* tendo a consequência de impedir que o utilizador aceda ao sistema remoto durante transferências. Outra melhoria existente é o facto de o cliente *ssh* armazenar alguma informação sobre o servidor, não sendo assim necessário comunicar com o servidor para todas as primitivas do sistema. Um exemplo disto é o comando *pwd* (*Print Working Directory*), que embora no *FTP* implicava comunicação com o servidor, com *SFTP*, o cliente nunca consulta o servidor.

TFTP

A aplicação *TFTP* (*Trivial File Transfer Protocol*) utiliza o protocolo de transporte *UDP* (por norma na porta 69) para efetuar as trocas de informação. As transferências baseadas em *TFTP* começam sempre com um pedido de leitura ou escrita que também tem a finalidade de pedir uma conexão. Assim que o servidor autorizar o pedido, a conexão fica aberta. Não utiliza nenhum mecanismo de segurança relativamente à autenticação do utilizador ou á encriptação de mensagens, pelo que se revela um protocolo pouco seguro.

Dado que utiliza o protocolo *UDP*, não existe uma grande complexidade. Quando um cliente pretende fazer uma transferência envia uma mensagem com o pedido. De seguida o servidor responde com o ficheiro que quando recebido pela aplicação do cliente envia uma mensagem de confirmação. Desta forma a maioria dos erros fecham a conexão e são sinalizados emitindo um pacote de erro, sendo que este pacote não é reconhecido nem retransmitido.

Também, devido ao *UDP*, tem uma elevada eficiência por causa de ter o overhead mais pequeno de todas estas aplicações porém não oferece uma solução viável para transferência de dados devido ás possíveis falhas.

HTTP (WGET)

O protocolo de aplicação HTTP (Hypertext Transfer Protocol) é utilizado para a comunicação de dados pela porta *TCP default* 80 (apesar de outras portas poderem ser configuradas para serem utilizadas). Para estabelecer uma conexão, o cliente começa por enviar um pedido de conexão (SYN) ao Servidor que, por sua vez, responde com um Acknowledge (SYN, ACK). Por último, o cliente responde com um acknowledge, confirmando que a conexão foi estabelecida (processo TCP 3-Way Handshake).

Para pedir uma transferência de um ficheiro ao servidor, o cliente começa por lhe enviar um pedido (GET) do mesmo. De seguida, o servidor responde com um acknowledge e depois envia os dados do ficheiro pretendido.

Para terminar a conexão, o cliente envia essa intenção ao servidor utilizando para o efeito uma mensagem FIN, ACK, ao qual o servidor responde com um Acknowledge a confirmar o término da conexão.

Concluindo, o protocolo HTTP é uma solução simples para a transferência de ficheiros uma vez que o consegue fazer com um número baixo de mensagens, porém não é a solução mais segura visto que e o facto de ser necessário abrir uma nova conexão *TCP* por ficheiro transmito pode atrasos significativos caso se esteja a transmitir grandes volumes de ficheiros. Estes problemas no entanto foram resolvidos no protocolo *HTTPS* (*Hypertext Transfer Protocol Secure*), que não só adiciona uma camada de encriptação SSL/TLS sobre os dados mas também permite a transferência de vários ficheiros pela mesma conexão *TCP*.

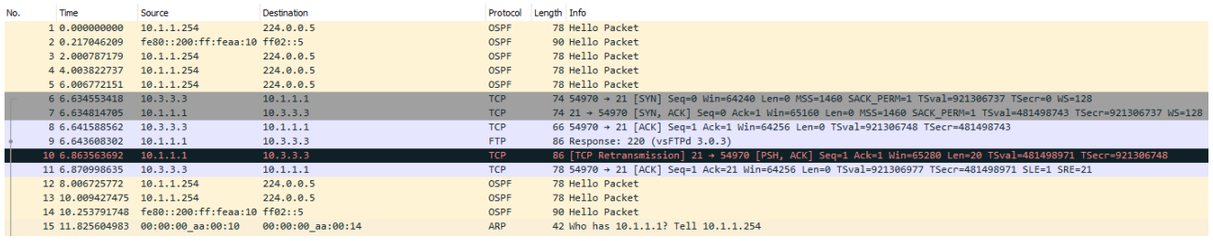
4. As características das ligações de rede têm uma enorme influência nos níveis de Transporte e de Aplicação. Discuta, relacionando a resposta com as experiências realizadas, as influências das situações de perda ou duplicação de pacotes IP no desempenho global de Aplicações fiáveis (se possível, relacionando 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 conetividade ao Backbone através de um link que funciona com perdas, atrasos e duplicações, que é o link entre o swicth SwitchLan3 e o router Router4. Nos testes podem mesmo ajustar esses parâmetros.

Os protocolos de transporte TCP e UDP são os mais usados e o processo de escolha de qual deles se usa depende do objetivo pretendido.

O TCP é um protocolo fiável que utiliza mecanismos de deteção de erros e de recuperação, para garantir que caso haja um erro no transporte, os pacotes continuam a ser enviados na ordem correta e sem erros. No entanto, dados todos estes mecanismos existentes no protocolo *TCP* este está dependente da qualidade da rede. Caso a rede seja de fraca qualidade, há uma muito maior probabilidade de perda de pacotes, o que fará com que haja um tráfego muito maior devido à necessidade de reenviarem esses pacotes. Podemos observar um pedido de retransmissão resultante de um erro de envio na captura realizada para a transferência de ficheiro usando *FTP* a partir do computador Corvo (fig. 11).

Por sua vez o protocolo UDP é quase uma antítese do protocolo *TCP*, já que é um protocolo não fiável e não se preocupa em reenviar pacotes que tenham sido corrompidos, deixando essa tarefa para as aplicações em si. Esta menor complexidade faz com que este protocolo seja muito favorável para redes com menor qualidade ou de *streaming*, visto que, ao contrário do protocolo fiável *TCP*, não provoca a sobrecarga da rede. Devido a esta falta de viabilidade podem aparecer situações de perda de dados como é o caso da transmissão *TFTP* da figura 12, em que falta um pacote *TFTP* de *Acknowledgment* que pode ter sido causada pela perda do pacote de *Data* ou pela perda do *Acknowledgment* entre a *Backbone* da topologia e a interface *Corvo*.



No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	10.1.1.254	224.0.0.5	OSPF	78	Hello Packet
2	0.217046209	fe80::200:ff:feaa:10	ff02::5	OSPF	90	Hello Packet
3	2.000787179	10.1.1.254	224.0.0.5	OSPF	78	Hello Packet
4	4.003822737	10.1.1.254	224.0.0.5	OSPF	78	Hello Packet
5	6.006772151	10.1.1.254	224.0.0.5	OSPF	78	Hello Packet
6	6.634553418	10.3.3.3	10.1.1.1	TCP	74	54970 → 21 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=921306737 TSecr=0 WS=128
7	6.634814705	10.1.1.1	10.3.3.3	TCP	74	21 → 54970 [SYN, ACK] Seq=0 Ack=1 Win=65160 Len=0 MSS=1460 SACK_PERM=1 TSval=481498743 TSecr=921306737 WS=128
8	6.641588562	10.3.3.3	10.1.1.1	TCP	66	54970 → 21 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=921306748 TSecr=481498743
9	6.643608302	10.1.1.1	10.3.3.3	FTP	86	Response: 220 (vsFTPd 3.0.3)
10	6.863563692	10.1.1.1	10.3.3.3	TCP	86	[TCP Retransmission] 21 → 54970 [PSH, ACK] Seq=1 Ack=1 Win=65280 Len=20 TSval=481498971 TSecr=921306748
11	6.870998635	10.3.3.3	10.1.1.1	TCP	78	54970 → 21 [ACK] Seq=1 Ack=21 Win=64256 Len=0 TSval=921306977 TSecr=481498971 SLE=1 SRE=21
12	8.006725772	10.1.1.254	224.0.0.5	OSPF	78	Hello Packet
13	10.009427475	10.1.1.254	224.0.0.5	OSPF	78	Hello Packet
14	10.253791748	fe80::200:ff:feaa:10	ff02::5	OSPF	90	Hello Packet
15	11.825604983	00:00:00_aa:00:10	00:00:00_aa:00:14	ARP	42	Who has 10.1.1.1? Tell 10.1.1.254

Figura 11: Exemplo de um pedido de retransmissão (Protocolo TCP)

No.	Time	Source	Destination	Protocol	Length	Info
166	116.165164202	10.1.1.254	224.0.0.5	OSPF	78	Hello Packet
167	118.171352745	10.1.1.254	224.0.0.5	OSPF	78	Hello Packet
168	119.095806027	10.3.3.3	10.1.1.1	TFTP	56	Read Request, File: file1, Transfer type: octet
169	119.098671612	10.1.1.1	10.3.3.3	TFTP	47	Data Packet, Block: 1 (last)
170	119.591156377	00:00:00_aa:00:10	00:00:00_aa:00:14	ARP	42	Who has 10.1.1.1? Tell 10.1.1.254
171	119.591484299	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
172	120.171623761	10.1.1.254	224.0.0.5	OSPF	78	Hello Packet
173	121.968557479	fe80::200:ff:feaa:10	ff02::5	OSPF	90	Hello Packet

Figura 12: Exemplo de um pacote TFTP de Acknowledgment em falta (Protocolo UCP)

Conclusão

Com este trabalho pudemos colocar em prática os conhecimentos adquiridos nas aulas teóricas da Unidade Curricular, fazendo com que tenhamos consolidado muito melhor as temáticas lecionadas, nomeadamente os diferentes tipos de protocolos de transporte e de aplicação. Para observarmos estes protocolos, utilizamos as ferramentas *CORE* e *Wireshark*.

Na verdade,ao observar a captura de tráfego no *Wireshark* torna bastante evidente as diferenças entre os vários protocolos estudados nesta fase. Ser capaz de ver diretamente o impacto dos diferentes protocolos no tempo de resposta do pedido de transferência de um ficheiro e verificar a retransmissão de pacotes a ocorrer no protocolo de transporte *TCP* quando ocorrem erros, ajudou-nos a entender a necessidade de existirem vários protocolos de aplicação tais como *FTP*, *SFTP*, *TFTP* e *HTTP* e protocolos de transporte como *UDP* e *TCP*.