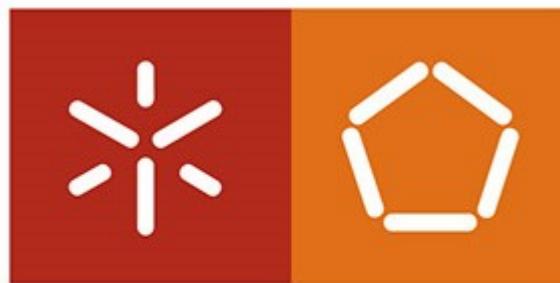


Comunicações por Computador - TP1 - GR68

Duarte Afonso Freitas Ribeiro :: A100764
Afonso Xavier Cardoso Marques :: A94940
Pedro Duarte Nobre Viana :: A100701

Outubro 2023



Universidade do Minho
Escola de Engenharia

1 Introdução

O presente relatório foi desenvolvido no âmbito da unidade curricular de Comunicações por Computador como proposta de resolução ao Trabalho Prático 1. Os leitores encontram no documento as conclusões que o grupo retirou face aos resultados empíricos do trabalho experimental proposto pela equipa docente.

Contents

1	Introdução	2
2	Parte Experimental - Questões e Respostas	4
2.1	Ping	4
2.2	SFTP	5
2.3	FTP	8
2.4	TFTP	11
2.5	HTTP	13
2.6	Parte I: Instalação, configuração e utilização de serviços de trans- ferência de ficheiros	16
2.7	Parte II: Uso da camada de transporte por parte das aplicações .	22
3	Comentários Finais	24
4	Anexos	25
4.1	Imagens Parte II:	25

2 Parte Experimental - Questões e Respostas

2.1 Ping

Figure 1: Resultado da execução do comando

Figure 2: Tráfego de pacotes durante a execução no PC

Figure 3: Tráfego de pacotes durante a execução no Portátil

Enquanto o tráfego proveniente do portátil funcionou como esperado, nem todos os pings do PC chegaram ao seu destino, como indicado pelo mesmo no terminal, sendo tal apoiado pelo Wireshark, que regista falhas nos mesmos pacotes que o PC (10, 12) e também regista o pacote 6 como duplicado, que o Wireshark de facto deteta duas vezes.

2.2 SFTP

SFTP é um protocolo que permite acessar, transferir e manipular ficheiros remotamente, fazendo uso de SSH para o transporte (via protocolo TCP) e encriptação de pacotes, fazendo uso da porta 22 para enviar e receber pacotes.

O início do tráfego é possível observar o estabelecimento da conexão TCP (cliente envia *SYN* para servidor, servidor envia *SYN,ACK*). De seguida decidem a versão do protocolo SSH a utilizar, o cliente autentica-se e há uma troca de chaves para encriptar o tráfego. A partir daí, pode haver transferência de dados segura, por via dos pacotes marcados com *Encrypted Packet*. Quando a transferência de dados finaliza, há troca de pacotes *FIN,ACK* de modo a terminar a conexão entre servidor e cliente.

No portátil este procedimento funcionou sem problemas, mas devido à instabilidade da conexão do PC houve pacotes perdidos (*TCP previous segment not captured*), e alguns duplicados (*TCP Dup ACK*). Sendo este protocolo baseado em TCP, a recuperação de pacotes dados como perdidos é a retransmissão dos mesmos, enquanto os pacotes duplicados são simplesmente descartados.

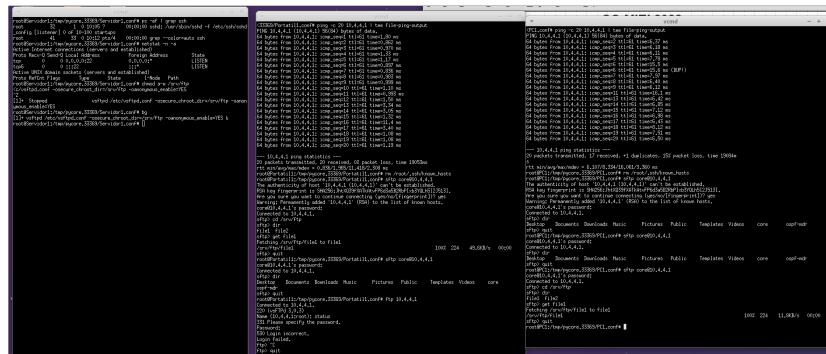


Figure 4: Execução dos comandos nos dispositivos

879	799	405604093	10.2.2.4	10.4.4.4	TCP	74	43964	22	[SYN]	Seq=0	Ack=64240	Len=0	MSS=1460	SACK PERM=1	Tsval=1183577260		
881	798	412513718	10.2.2.1	10.4.4.4	TCP	66	43964	22	[ACK]	Seq=1	Ack=64256	Len=0	Win=65160	Len=0	MSS=1460	SA	107 Client: Protocol [SSH-2.0-OpenSSH_8.2p1 Ubuntu-4ubuntu0.3]
882	798	412514569	10.2.2.1	10.4.4.4	TCP	66	43964	22	[ACK]	Seq=1	Ack=64256	Len=0	Win=65160	Len=0	TSval=1183577260		
883	798	405604093	10.2.2.1	10.4.4.4	TCP	66	43964	22	[ACK]	Seq=1	Ack=64256	Len=0	Win=65152	Len=0	TSval=783361526...		
884	798	412517182	10.2.2.1	10.4.4.4	TCP	66	43964	22	[ACK]	Seq=1	Ack=64256	Len=0	Win=65152	Len=0	TSval=783361526...		
885	798	433176492	10.2.2.1	10.4.4.4	TCP	66	43964	22	[ACK]	Seq=42	Ack=64256	Len=0	Win=64256	Len=0	TSval=11835772...		
886	798	433567256	10.2.2.1	10.4.4.4	SSHv2	1099	Server:	Key Exchange	Init								
887	798	437569157	10.2.2.1	10.4.4.4	TCP	1514	43964	22	[ACK]	Seq=42	Ack=64256	Len=0	Win=64256	Len=0	TSval=11835...		
888	798	438218885	10.2.2.1	10.4.4.4	TCP	66	22	-	43964	[ACK]	Seq=1066	Ack=1499	Win=64128	Len=0	TSval=1183...		
889	798	438218885	10.2.2.1	10.4.4.4	TCP	66	22	-	43964	[ACK]	Seq=1066	Ack=1499	Win=64128	Len=0	TSval=1183...		
890	798	438218885	10.2.2.1	10.4.4.4	SSHv2	114	Client:	Difflie-Hellman Key Exchange	Init								
891	798	057652804	10.2.2.1	10.4.4.4	TCP	78	[TCP Dup ACK 88881]	22	-	43964	[ACK]	Seq=1066	Ack=1499	Win=64...			
893	798	873709486	10.2.2.1	10.4.4.4	TCP	130	[TCP Retransmission]	43964	-	22	[PSH]	ACK	Seq=1498	Ack=1066			
894	798	405604093	10.2.2.1	10.4.4.4	TCP	66	22	-	43964	[ACK]	Seq=1066	Ack=1662	Win=64128	Len=0	TSval=1183...		
895	798	405604093	10.2.2.1	10.4.4.4	SSHv2	1182	Server:	Difflie-Hellman Key Exchange	Reply	New Keys,	Encrypte...						
896	798	909086583	10.2.2.1	10.4.4.4	TCP	66	43964	22	[ACK]	Seq=1602	Ack=2182	Win=64128	Len=0	TSval=1183...			
898	798	997412579	10.2.2.1	10.4.4.4	TCP	110	43964	22	[PSH]	ACK	Seq=1618	Ack=2182	Win=64128	Len=0	TSval=783...		
900	798	997412579	10.2.2.1	10.4.4.4	TCP	66	43964	22	[PSH]	ACK	Seq=1618	Ack=2182	Win=64128	Len=0	TSval=783...		
901	798	997412579	10.2.2.1	10.4.4.4	TCP	66	43964	22	[PSH]	ACK	Seq=1618	Ack=2182	Win=64128	Len=0	TSval=783...		
902	798	998427471	10.2.2.1	10.4.4.4	SSHv2	118	Server:	Encrypted packet	(len=44)								
903	798	915235411	10.2.2.1	10.4.4.4	TCP	66	43964	22	[ACK]	Seq=1662	Ack=2226	Win=64128	Len=0	TSval=1183...			
904	798	915236583	10.2.2.1	10.4.4.4	TCP	126	43964	22	[PSH]	ACK	Seq=1662	Ack=2226	Win=64128	Len=0	TSva...		
905	798	915961569	10.2.2.1	10.4.4.4	TCP	66	22	-	43964	[ACK]	Seq=1722	Ack=1722	Win=64128	Len=0	TSval=1183...		
906	798	922157893	10.2.2.1	10.4.4.4	SSHv2	118	Server:	Encrypted packet	(len=44)								
907	798	974382796	10.2.2.1	10.4.4.4	TCP	66	43964	22	[ACK]	Seq=1722	Ack=2279	Win=64128	Len=0	TSval=1183...			
912	798	975366409	10.2.2.1	10.4.4.4	TCP	150	43964	22	[PSH]	ACK	Seq=1722	Ack=2279	Win=64128	Len=0	TSva...		
913	798	973986556	10.2.2.1	10.4.4.4	SSHv2	94	Server:	Encrypted packet	(len=28)								
914	798	988137774	10.2.2.1	10.4.4.4	TCP	104	43964	22	[ACK]	Seq=1806	Ack=2389	Win=64128	Len=0	TSval=1183...			
915	798	988137774	10.2.2.1	10.4.4.4	TCP	104	43964	22	[ACK]	Seq=1806	Ack=2389	Win=64128	Len=0	TSval=1183...			
916	798	024683072	10.2.2.1	10.4.4.4	TCP	66	22	-	43964	[ACK]	Seq=2306	Ack=1918	Win=64128	Len=0	TSval=1183...		
917	798	245556227	10.2.2.1	10.4.4.4	SSHv2	534	Server:	Encrypted packet	(len=468)								
918	798	25225111	10.2.2.1	10.4.4.4	TCP	66	43964	22	[ACK]	Seq=1918	Ack=2774	Win=64128	Len=0	TSval=1183...			
919	798	252653950	10.2.2.1	10.4.4.4	SSHv2	110	Server:	Encrypted packet	(len=44)								
920	798	259711988	10.2.2.1	10.4.4.4	TCP	66	43964	22	[ACK]	Seq=1918	Ack=2819	Win=64128	Len=0	TSval=1183...			

Figure 5: Tráfego de pacotes durante a execução no PC

944	794	891404752	10.2.2.1	10.4.4.4	SSHv2	118	Server:	Encrypted packet	(len=52)							
945	794	894736552	10.2.2.1	10.4.4.4	TCP	66	43964	22	[ACK]	Seq=2950	Ack=3398	Win=64128	Len=0	TSval=1183...		
946	794	89958654	10.2.2.1	10.4.4.4	TCP	118	43964	22	[PSH]	ACK	Seq=3398	Win=64128	Len=52	Tsva...		
947	794	902292094	10.2.2.1	10.4.4.4	SSHv2	526	Server:	Encrypted packet	(len=468)							
948	794	99762850	10.2.2.1	10.4.4.4	TCP	118	43964	22	[PSH]	ACK	Seq=3002	Ack=3858	Win=64128	Len=52	Tsva...	
949	794	99762850	10.2.2.1	10.4.4.4	SSHv2	134	Server:	Encrypted packet	(len=468)							
950	794	918177727	10.2.2.1	10.4.4.4	TCP	66	43964	22	[PSH]	ACK	Seq=3054	Ack=3926	Win=64128	Len=52	Tsva...	
951	794	924081967	10.2.2.1	10.4.4.4	SSHv2	134	Server:	Encrypted packet	(len=68)							
952	794	973409919	10.2.2.1	10.4.4.4	TCP	66	43964	22	[ACK]	Seq=3106	Ack=3994	Win=64128	Len=0	TSval=1183...		
953	794	973410871	10.2.2.1	10.4.4.4	TCP	66	[TCP Dup ACK 952#1]	43964	22	[ACK]	Seq=3106	Ack=3994	Win=64...			
954	794	988137774	10.2.2.1	10.4.4.4	TCP	154	43964	22	[PSH]	ACK	Seq=3106	Ack=4079	Win=64128	Len=0	TSval=1183...	
955	794	988137774	10.2.2.1	10.4.4.4	SSHv2	142	Server:	Encrypted packet	(len=44)							
956	794	989427471	10.2.2.1	10.4.4.4	TCP	66	43964	22	[ACK]	Seq=3174	Ack=4079	Win=64128	Len=0	TSval=1183...		
957	794	991523988	10.2.2.1	10.4.4.4	TCP	134	43964	22	[PSH]	ACK	Seq=3174	Ack=4079	Win=64128	Len=0	TSval=1183...	
958	794	991523988	10.2.2.1	10.4.4.4	SSHv2	142	Server:	Encrypted packet	(len=76)							
959	794	991523988	10.2.2.1	10.4.4.4	TCP	66	43964	22	[ACK]	Seq=3424	Ack=4146	Win=64128	Len=0	TSval=1183...		
960	794	991523988	10.2.2.1	10.4.4.4	SSHv2	142	Server:	Encrypted packet	(len=52)							
961	794	991720676	10.2.2.1	10.4.4.4	TCP	134	43964	22	[PSH]	ACK	Seq=3474	Ack=4079	Win=64128	Len=0	TSval=1183...	
962	794	991720676	10.2.2.1	10.4.4.4	SSHv2	142	Server:	Encrypted packet	(len=76)							
963	794	991720676	10.2.2.1	10.4.4.4	TCP	66	43964	22	[ACK]	Seq=3474	Ack=4146	Win=64128	Len=0	TSval=1183...		
964	794	991720676	10.2.2.1	10.4.4.4	SSHv2	118	Server:	Encrypted packet	(len=52)							
965	794	993812736	10.2.2.1	10.4.4.4	TCP	66	43964	22	[PSH]	ACK	Seq=3318	Ack=4198	Win=64128	Len=0	TSval=1183...	
966	794	994427888	10.2.2.1	10.4.4.4	TCP	134	43964	22	[PSH]	ACK	Seq=3318	Ack=4198	Win=64128	Len=0	TSval=1183...	
967	794	994427888	10.2.2.1	10.4.4.4	SSHv2	342	Server:	Encrypted packet	(len=76)							
968	794	994427888	10.2.2.1	10.4.4.4	TCP	66	43964	22	[PSH]	ACK	Seq=3474	Ack=4474	Win=64128	Len=0	TSval=1183...	
969	794	994427888	10.2.2.1	10.4.4.4	SSHv2	134	Server:	Encrypted packet	(len=68)							
970	794	994427888	10.2.2.1	10.4.4.4	TCP	118	43964	22	[PSH]	ACK	Seq=3454	Ack=4542	Win=64128	Len=0	TSval=1183...	
971	798	405604093	10.2.2.1	10.4.4.4	SSHv2	134	Server:	Encrypted packet	(len=68)							
972	798	405604093	10.2.2.1	10.4.4.4	TCP	66	43964	22	[ACK]	Seq=3496	Ack=4540	Win=64128	Len=0	TSval=1183...		
973	798	405604093	10.2.2.1	10.4.4.4	SSHv2	109	Server:	Encrypted packet	(len=124)							
978	803	109864987	10.2.2.1	10.4.4.4	TCP	66	43964	22	[PSH]	ACK	Seq=3542	Ack=4734	Win=64128	Len=0	TSval=1183...	
979	803	109885818	10.2.2.1	10.4.4.4	TCP	102	43964	22	[PSH]	ACK	Seq=3542	Ack=4738	Win=64128	Len=0	TSval=1183...	
980	803	109884416	10.2.2.1	10.4.4.4	TCP	102	43964	22	[PSH]	ACK	Seq=3578	Ack=4738	Win=64128	Len=0	TSval=1183...	
981	803	109515238	10.2.2.1	10.4.4.4	TCP	66	22	-	43964	[ACK]	Seq=4734	Ack=3639	Win=64128	Len=0	TSval=1183...	
982	803	109515238	10.2.2.1	10.4.4.4	TCP	66	22	-	43964	[ACK]	Seq=4734	Ack=3639	Win=64128	Len=0	TSval=1183...	
983	803	114643231	10.2.2.1	10.4.4.4	TCP	66	22	-	43964	[FIN, ACK]	Seq=4734	Ack=3639	Win=64128	Len=0	TSval=1183...	
984	803	126515220	10.2.2.1	10.4.4.4	TCP	66	43964	22	[ACK]	Seq=3639	Ack=4735	Win=64128	Len=0	TSval=1183...		

Figure 7: Tráfego de pacotes durante a execução no Portátil

319	264.129240781	10.4.4.1	10.1.1.1	SSHv2	124 Client: Encrypted packet (Len=68)
320	264.138375678	10.1.1.1	10.4.4.1	TCP	66 51382 - 22 [ACK] Seq=3174 Ack=4070 Win=64128 Len=0 Tsva1=3980..
321	264.131720873	10.4.4.1	10.1.1.1	SSHv2	134 Client: Encrypted packet (Len=68)
322	264.131720873	10.4.4.1	10.1.1.1	SSHv2	142 Server: Encrypted packet (Len=76)
323	264.132566279	10.1.1.1	10.4.4.1	TCP	66 51382 - 22 [ACK] Seq=3146 Ack=4146 Win=64128 Len=0 Tsva1=3980..
324	264.132568895	10.1.1.1	10.4.4.1	SSHv2	142 Client: Encrypted packet (Len=76)
325	264.134012527	10.1.1.1	10.4.4.1	SSHv2	134 Client: Encrypted packet (Len=68)
326	264.134012527	10.1.1.1	10.4.4.1	SSHv2	134 Client: Encrypted packet (Len=68)
327	264.134751489	10.4.4.1	10.1.1.1	SSHv2	342 Server: Encrypted packet (Len=276)
328	264.137423292	10.4.4.1	10.1.1.1	SSHv2	134 Client: Encrypted packet (Len=68)
329	264.137423292	10.4.4.1	10.1.1.1	SSHv2	134 Client: Encrypted packet (Len=68)
330	264.138621473	10.1.1.1	10.4.4.1	SSHv2	118 Client: Encrypted packet (Len=52)
331	264.139337445	10.4.4.1	10.1.1.1	SSHv2	134 Server: Encrypted packet (Len=68)
332	264.139337445	10.4.4.1	10.1.1.1	TCP	66 51382 - 22 [ACK] Seq=3596 Ack=4610 Win=64128 Len=0 Tsva1=3980..
333	264.239911747	10.4.4.254	224.0.0.5	OSPF	78 Hello Packet
334	265.791959541	fe80::200:ff:fea:10	f002:5	OSPF	90 Hello Packet
335	265.242505274	10.4.4.254	224.0.0.5	OSPF	78 Hello Packet
336	265.255526142	10.4.4.1	10.1.1.1	SSHv2	180 Client: Encrypted packet (Len=36)
337	265.255526142	10.4.4.1	10.1.1.1	SSHv2	199 Server: Encrypted packet (Len=124)
338	266.256611943	10.1.1.1	10.4.4.1	TCP	66 51382 - 22 [ACK] Seq=3542 Ack=4734 Win=64128 Len=0 Tsva1=3980..
339	266.256611943	10.1.1.1	10.4.4.1	SSHv2	192 Client: Encrypted packet (Len=68)
340	266.256624427	10.1.1.1	10.4.4.1	SSHv2	126 Client: Encrypted packet (Len=68)
341	266.257126698	10.1.1.1	10.4.4.1	TCP	66 51382 - 22 [FIN, ACK] Seq=3638 Ack=4734 Win=64128 Len=0 Tsva1=3877..
342	266.257523164	10.4.4.1	10.1.1.1	TCP	66 22 - 51382 [ACK] Seq=4734 Ack=3638 Win=64128 Len=0 Tsva1=3877..
343	266.257523164	10.4.4.1	10.1.1.1	TCP	66 22 - 51382 [ACK] Seq=4734 Ack=3638 Win=64128 Len=0 Tsva1=3864..
344	266.261193918	10.1.1.1	10.4.4.1	TCP	66 51382 - 22 [ACK] Seq=3639 Ack=4735 Win=64128 Len=0 Tsva1=3980..

Figure 8: Tráfego de pacotes durante a execução no Portátil

2.3 FTP

O protocolo FTP, ao contrário do que o nome indica, não é a base do protocolo SFTP, visto que este último foi construído de raiz com a tecnologia SSH. Apesar disto, ambos utilizam o protocolo TCP para o transporte de pacotes, mas o FTP utiliza duas portas em vez de apenas uma. A porta 21 do servidor, a primeira a receber um pedido, é chamada porta de controlo e de seguida abre a sua porta 20 para a transmissão de dados entre si e o cliente.

É possível observar isto nos primeiros pacotes transmitidos. O cliente envia ao servidor uma request, à qual o servidor responde com o código 200 (OK). De seguida, o cliente pede ao servidor o ficheiro em questão (*RETR file1*). Nesse momento uma conexão TCP é estabelecida (*SYN* é enviado ao cliente, e este envia *SYN,ACK* de volta). Depois de estabelecida a ligação TCP, o Servidor estabelece o tipo de ligação de dados (neste caso binário), e envia o ficheiro (*FTP Data*). Finalmente a ligação de dados é terminada (*FIN, ACK*), no entanto a ligação de controlo continua aberta, por onde o servidor envia um pacote com código 226, que significa transferência completa. O cliente enviou então um pacote *QUIT*, ao qual o servidor responde com o código 221 (goodbye). Desse modo, a ligação de controlo é também terminada, do mesmo modo que a de dados, pois ambas são feitas no protocolo TCP.

Sendo um protocolo baseado em TCP, as perdas e duplicações de pacotes observadas no PC mas não no portátil são lidadas da mesma forma que no protocolo SFTP, descrito acima.

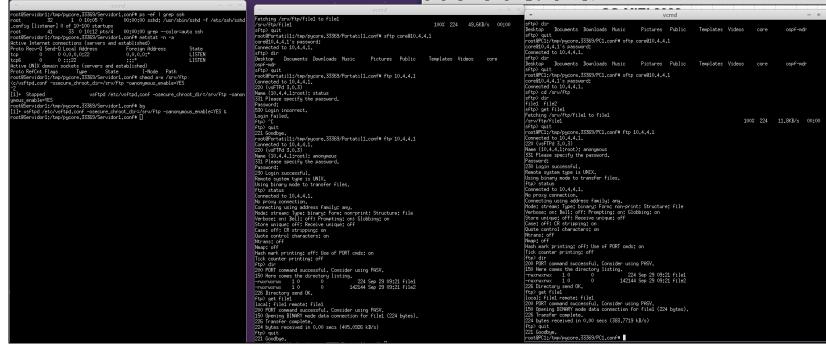


Figure 9: Execução dos comandos nos dispositivos

1319 1191.9788725..	10.2.2.1	18.4.4.1	FTP	5 Request: PORT 10,2,2,1,156,5
1320 1191.9751146..	10.4.4.1	10.2.2.1	FTP	117 Response: 200 PORT command successful. Consider using PASV.
1321 1191.9814458..	10.2.2.1	18.4.4.1	TCP	66 37186 - 21 [ACK] Seq=35 Ack=148 Win=64256 Len=0 Tsva1=1183988..
1322 1191.9814468..	10.2.2.1	10.4.4.1	FTP	72 Request: LIST
1323 1191.9821489..	10.4.4.1	10.2.2.1	TCP	74 29 - 38465 [SYN] Seq=0 Win=64256 Len=0 MSS=1460 SACK_PERM=1 T..
1324 1191.9821493..	10.4.4.1	10.2.2.1	TCP	74 29 - 38465 [SYN] Seq=0 Win=64256 Len=0 MSS=1460 SACK_PERM=1 T..
1325 1191.9881804..	10.4.4.1	10.2.2.1	TCP	66 29 - 38465 [ACK] Seq=1 Ack=1 Win=64256 Len=0 Tsva1=78372932 ..
1326 1191.9888671..	10.4.4.1	10.2.2.1	FTP	105 Response: 159 Here comes the directory listing.
1327 1191.9888992..	10.4.4.1	10.2.2.1	FTP-DA..	192 FTP Data: 126 bytes (PORT) (LIST)
1328 1191.9894462..	10.4.4.1	10.2.2.1	TCP	66 29 - 38465 [FIN, ACK] Seq=128 Ack=1 Win=64256 Len=0 Tsva1=783..
1329 1191.9894463..	10.2.2.1	18.4.4.1	TCP	65 37186 - 21 [ACK] Seq=61 Ack=1 Win=64256 Len=0 Tsva1=1183988..
1330 1191.9894468..	10.2.2.1	18.4.4.1	TCP	66 37186 - 21 [ACK] Seq=62 Ack=127 Win=65168 Len=0 Tsva1=1183986..
1331 1192.2056827..	10.4.4.1	10.2.2.1	TCP	66 [TCP Retransmission] 20 - 38465 [FIN, ACK] Seq=127 Ack=128 Win=1460 SA..
1332 1192.2148709..	10.2.2.1	18.4.4.1	TCP	78 [TCP Previous segment not captured] 38465 - 20 [ACK] Seq=2 Ac..
1333 1192.2148716..	10.2.2.1	18.4.4.1	TCP	78 [TCP Dup ACK 1332?1] 38465 - 20 [ACK] Seq=2 Ack=2 Win=65152..
1334 1192.2167183..	10.4.4.1	10.2.2.1	FTP	99 Response: 200 Directory send OK.
1335 1192.2167184..	10.4.4.1	10.2.2.1	TCP	66 29 - 38465 [FIN, ACK] Seq=211 Ack=211 Win=64256 Len=0 Tsva1=1183988..
1336 1192.437214..	10.2.2.1	18.4.4.1	TCP	66 [TCP Retransmission] 38465 - 20 [FIN, ACK] Seq=128 Ack=128 Win=..
1337 1192.4228616..	10.4.4.1	10.2.2.1	TCP	66 29 - 38465 [ACK] Seq=128 Ack=2 Win=64256 Len=0 Tsva1=78377336..
1338 1193.3252286..	10.4.4.254	224.0.9.5	OSPF	78 Hello Packet
1339 1194.1893288..	10.2.2.1	10.4.4.1	FTP	74 Request: TYPE
1340 1194.1893289..	10.2.2.1	10.4.4.1	FTP	97 Response: 200 Switching to Binary mode.
1341 1194.1963309..	10.2.2.1	10.4.4.1	TCP	66 37186 - 21 [ACK] Seq=69 Ack=242 Win=64256 Len=0 Tsva1=1183990..
1342 1194.1963317..	10.2.2.1	10.4.4.1	FTP	88 Request: PORT 10,2,2,1,188,71
1343 1194.1977923..	10.4.4.1	10.2.2.1	FTP	117 Response: 200 Directory send OK.
1344 1194.2043955..	10.2.2.1	10.4.4.1	FTP	78 Request: RETR file1
1345 1194.2123941..	10.4.4.1	10.2.2.1	TCP	74 29 - 38465 [SYN] Seq=0 Win=64256 Len=0 MSS=1460 SACK_PERM=1 T..
1346 1194.2123942..	10.4.4.1	10.2.2.1	TCP	74 29 - 38465 [SYN] Seq=0 Win=64256 Len=0 MSS=1460 SACK_PERM=1 T..
1347 1194.2138146..	10.4.4.1	10.2.2.1	TCP	66 29 - 48199 [ACK] Seq=1 Ack=1 Win=64256 Len=0 Tsva1=783775157..
1348 1194.2148888..	10.4.4.1	10.2.2.1	FTP	130 Response: 159 Opening BINARY mode data connection for file1 (-..
1349 1194.2149164..	10.4.4.1	10.2.2.1	FTP-DA..	298 FTP Data: 224 bytes (PORT) (RETR file1)
1350 1194.2151959..	10.4.4.1	10.2.2.1	TCP	66 29 - 48199 [FIN, ACK] Seq=233 Ack=233 Win=0 Tsva1=783775157..
1351 1194.2152054..	10.2.2.1	10.4.4.1	TCP	78 [TCP Previous segment not captured] 48199 - 20 [ACK] Seq=233 Ack=233 Win=..
1352 1194.2470673..	10.2.2.1	10.4.4.1	FTP	98 Response: 159 Here comes the directory listing.
1353 1194.2484982..	10.4.4.1	10.2.2.1	FTP	99 Response: 226 Transfer complete.
1354 1194.2543335..	10.2.2.1	10.4.4.1	TCP	66 37186 - 21 [ACK] Seq=103 Ack=381 Win=64256 Len=0 Tsva1=118399..

Figure 10: Tráfego de pacotes durante a execução no PC

1351 1194.2414024..	10.4.4.1	10.2.2.1	TCP	66 [TCP Retransmission] 20 - 48199 [FIN, ACK] Seq=225 Ack=1 Win=..
1352 1194.2470673..	10.2.2.1	10.4.4.1	FTP	78 [TCP Previous segment not captured] 48199 - 20 [ACK] Seq=2 Ac..
1353 1194.2470674..	10.2.2.1	10.4.4.1	FTP	99 Response: 226 Transfer complete.
1354 1194.437221..	10.2.2.1	10.4.4.1	TCP	66 [TCP Retransmission] 38465 - 20 [FIN, ACK] Seq=128 Ack=128 Win=..
1355 1194.437221..	10.2.2.1	10.4.4.1	TCP	66 [TCP Retransmission] 38465 - 20 [FIN, ACK] Seq=128 Ack=128 Win=..
1356 1195.3275266..	10.4.4.254	224.0.9.5	OSPF	78 Hello Packet
1357 1195.6222693..	10.2.2.1	10.4.4.1	TCP	72 Request: QUIT
1358 1195.6222694..	10.4.4.1	10.2.2.1	FTP	88 Response: 226 Transfer complete.
1360 1195.6295168..	10.2.2.1	10.4.4.1	TCP	66 29 - 38465 [FIN, ACK] Seq=305 Ack=109 Win=65280 Len=0 Tsva1=7..
1361 1195.6295168..	10.2.2.1	10.4.4.1	TCP	66 37186 - 21 [ACK] Seq=109 Ack=306 Win=64256 Len=0 Tsva1=7..
1362 1195.6295168..	10.4.4.1	10.2.2.1	TCP	66 21 - 37186 [ACK] Seq=398 Ack=110 Win=65280 Len=0 Tsva1=783776..

Figure 11: Tráfego de pacotes durante a execução no PC

1373 1046.1297405..	10.1.1.1	10.2.2.1	FTP	89 Request: PORT 10,1,1,1,232,189
1374 1046.1397648..	10.4.4.1	10.4.4.1	TCP	117 Response: 200 PORT command successful. Consider using PASV.
1375 1046.1317551..	10.1.1.1	10.4.4.1	TCP	66 40914 - 21 [ACK] Seq=148 Win=64256 Len=0 Tsva1=3981676..
1376 1046.1317552..	10.1.1.1	10.4.4.1	TCP	72 Request: LIST
1377 1046.1324341..	10.4.4.1	10.1.1.1	TCP	74 29 - 55851 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 T..
1378 1046.1324322..	10.1.1.1	10.4.4.1	TCP	74 50851 - 20 [ACK] Seq=1 Ack=1 Win=64256 Len=0 MSS=1460 SA..
1379 1046.1324323..	10.1.1.1	10.4.4.1	TCP	66 50851 - 20 [ACK] Seq=1 Ack=1 Win=64256 Len=0 MSS=1460 SA..
1380 1046.1384263..	10.1.1.1	10.4.4.1	TCP	66 50851 - 20 [ACK] Seq=1 Ack=1 Win=64256 Len=0 MSS=1460 SA..
1381 1046.1384263..	10.1.1.1	10.4.4.1	TCP	108 Response: 159 Here comes the directory listing.
1382 1046.1347144..	10.4.4.1	10.1.1.1	FTP-DA..	192 FTP Data: 126 bytes (PORT) (LIST)
1383 1046.1347144..	10.1.1.1	10.4.4.1	TCP	109 Response: 226 Transfer complete.
1384 1046.1347691..	10.1.1.1	10.4.4.1	TCP	66 40914 - 21 [ACK] Seq=137 Win=64256 Len=0 Tsva1=3981676..
1385 1046.1352879..	10.1.1.1	10.4.4.1	TCP	66 50951 - 20 [ACK] Seq=1 Ack=2 Win=65152 Len=0 Tsva1=3981676..
1386 1046.1398463..	10.1.1.1	10.4.4.1	TCP	66 50951 - 20 [ACK] Seq=1 Ack=2 Win=65152 Len=0 Tsva1=3981676..
1387 1046.1378837..	10.4.4.1	10.1.1.1	FTP	98 Response: 226 Transfer complete.
1388 1046.1384263..	10.1.1.1	10.4.4.1	TCP	66 40914 - 21 [ACK] Seq=137 Win=64256 Len=0 Tsva1=3981676..
1389 1046.1384263..	10.1.1.1	10.4.4.1	TCP	99 Response: 226 Transfer complete.
1390 1046.1384267..	10.4.4.1	10.1.1.1	OSPF	99 Request: RETR file1
1391 1046.1744089..	10.4.4.254	224.0.9.5	OSPF	75 Hello Packet
1392 1049.5197405..	10.1.1.1	10.4.4.1	FTP	91 Request: PORT 10,1,1,1,232,189
1393 1049.5197405..	10.1.1.1	10.4.4.1	FTP	117 Response: 200 Switching to Binary mode.
1394 1049.5216966..	10.1.1.1	10.4.4.1	TCP	66 40914 - 21 [ACK] Seq=71 Ack=242 Win=64256 Len=0 Tsva1=3981679..
1395 1049.5220542..	10.1.1.1	10.4.4.1	FTP	98 Request: PORT 10,1,1,1,232,189
1396 1049.5220542..	10.1.1.1	10.4.4.1	TCP	117 Response: 200 Opening BINARY mode data connection for file1 (-..
1397 1049.5284087..	10.1.1.1	10.4.4.1	TCP	66 40914 - 21 [ACK] Seq=94 Ack=299 Win=64256 Len=0 Tsva1=3981679..
1398 1049.5258949..	10.1.1.1	10.4.4.1	TCP	75 Request: RETR file1
1399 1049.5265661..	10.4.4.1	10.1.1.1	TCP	74 29 - 57771 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 T..
1400 1049.5265661..	10.1.1.1	10.4.4.1	TCP	66 50771 - 20 [ACK] Seq=1 Ack=1 Win=65168 Len=0 MSS=1460 SA..
1401 1049.5275426..	10.4.4.1	10.1.1.1	TCP	66 29 - 57771 [ACK] Seq=1 Ack=1 Win=65168 Len=0 MSS=1460 SA..
1402 1049.5281903..	10.4.4.1	10.1.1.1	FTP	138 Response: 159 Opening BINARY mode data connection for file1 (-..
1403 1049.5281903..	10.1.1.1	10.4.4.1	TCP	66 50771 - 20 [ACK] Seq=1 Ack=1 Win=65168 Len=0 MSS=1460 SA..
1404 1049.5291684..	10.1.1.1	10.4.4.1	TCP	66 40914 - 21 [ACK] Seq=357 Win=64256 Len=0 Tsva1=398167..
1405 1049.5291925..	10.4.4.1	10.1.1.1	TCP	66 29 - 57771 [FIN, ACK] Seq=225 Ack=201 Win=64256 Len=0 Tsva1=387..
1406 1049.5292040..	10.1.1.1	10.4.4.1	TCP	66 57771 - 20 [ACK] Seq=1 Ack=235 Win=64256 Len=0 Tsva1=3981679..
1407 1049.5292040..	10.1.1.1	10.4.4.1	TCP	66 29 - 57771 [ACK] Seq=226 Win=64256 Len=0 Tsva1=3981679..
1408 1049.5315839..	10.4.4.1	10.1.1.1	TCP	66 29 - 57771 [ACK] Seq=226 Win=64256 Len=0 Tsva1=3873559..

Figure 12: Tráfego de pacotes durante a execução no Portátil

Sequence Number	Source IP	Destination IP	Source Port	Destination Port	Type	Timestamp	Flags	Sequence Number	Acknowledgment Number	Window Size	Length	TSeqVal
1295 1049.5291925...	19.4.4.1	10.1.1.1	10.1.1.1	10.4.4.1	TCP	66 29 - 57771	[ACK]	3982	0	0	0	389107...
1296 1049.5292080...	10.1.1.1	10.1.1.1	10.4.4.1	10.4.4.1	TCP	66 5771 - 20	[ACK]	3982	1	64256	Len=0	TSeqVal=307...
1297 1049.5315838...	10.1.1.1	10.1.1.1	10.4.4.1	10.4.4.1	TCP	66 5771 - 21	[ACK]	3982	2	64256	Len=0	TSeqVal=308...
1298 1049.5315839...	10.4.4.1	10.1.1.1	10.1.1.1	10.4.4.1	TCP	66 20 - 57771	[FIN, ACK]	225	Ack=1	Win=64256	Len=0	TSeqVal=307...
1299 1049.5320440...	10.4.4.1	10.1.1.1	10.1.1.1	10.4.4.1	FTP	99 Response: 226	Transfer complete.					
1300 1051.1756212...	10.4.4.254	224.0.0.5	224.0.0.5	224.0.0.5	OSPF	78 Hello Packet						
1301 1051.1756212...	10.4.4.254	224.0.0.5	224.0.0.5	224.0.0.5	OSPF	78 Hello Packet						
1302 1053.3739700...	10.4.4.1	10.1.1.1	10.1.1.1	10.4.4.1	FTP	77 Goodbye.						
1303 1053.3739700...	10.4.4.1	10.1.1.1	10.1.1.1	10.4.4.1	FTP	80 Response: 221	Goodbye.					
1304 1053.3746841...	10.4.4.1	10.1.1.1	10.1.1.1	10.4.4.1	TCP	66 21 - 46914	[FIN, ACK]	395	Ack=112	Win=65280	Len=0	TSeqVal=3...
1305 1053.3746841...	10.1.1.1	10.1.1.1	10.4.4.1	10.4.4.1	TCP	66 46914 - 21	[ACK]	395	1	64256	Len=0	TSeqVal=3...
1306 1053.3746841...	10.1.1.1	10.1.1.1	10.4.4.1	10.4.4.1	TCP	66 46914 - 22	[ACK]	395	2	64256	Len=0	TSeqVal=3...
1307 1053.3778331...	10.4.4.1	10.1.1.1	10.1.1.1	10.4.4.1	TCP	66 21 - 46914	[ACK]	395	113	65280	Len=0	TSeqVal=387836...
1308 40CF...4364420...	10.4.4.1

Figure 13: Tráfego de pacotes durante a execução no Portátil

2.4 TFTP

Ao contrário dos protocolos anteriores, TFTP utiliza UDP na sua camada de transporte, algo que lhe confere menor overhead, pois não existe uma ligação formal, apenas troca de pacotes "soltos" entre máquinas.

A ligação TFTP é iniciada quando o cliente requisita ao servidor a leitura/escrita de um ficheiro. Neste caso houve uma *read request* para o *file1*. Ao receber esta informação, o servidor envia para o cliente o *data packet* com os dados do ficheiro. Como o ficheiro era pequeno suficiente para caber num único pacote (menor de 512 bytes), este pacote vem com uma flag a sinalizar que este é o último pacote de dados (*last*). A cada pacote de dados recebido (neste caso apenas um), o cliente envia um *acknowledgement* de o ter recebido ao servidor, e a "ligação" entre os dois termina.

Neste caso não observamos perda nem duplicação de pacotes em nenhuma das ligações, provavelmente pois o número de pacotes enviados e recebidos não foram suficientes para causar erros na conexão instável do PC1. No entanto, se houvesse perda de algum dos pacotes, após um certo período de tempo sem obter uma resposta da outra máquina (dados ou ACK), o dispositivo retransmitiria a sua mensagem de novo. Se o cliente recebesse um pacote duplicado, este iria reenviar um ack do pacote do ficheiro com maior *block* (número de pacote) que recebeu.

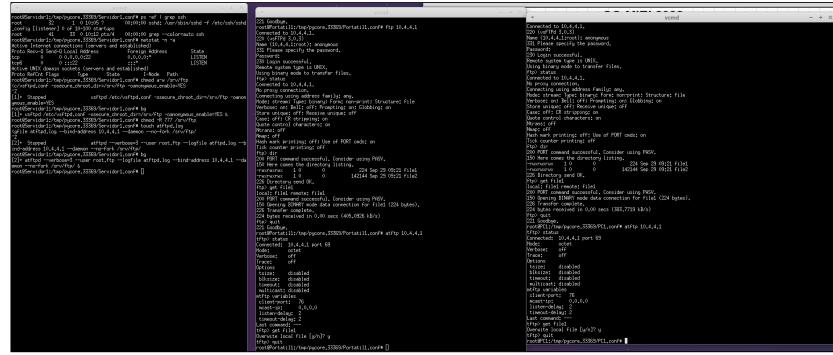


Figure 14: Execução dos comandos nos dispositivos

1623 1617.5911244.. 10.2.2.1	10.4.4.1	TFTP	56 Read Request, File: file1, Transfer type: octet
1624 1617.5922953.. 10.4.4.1	10.2.2.1	TFTP	270 Data Packet, Block: 1 (last)
1625 1617.5989309.. 10.2.2.1	10.4.4.1	TFTP	46 Acknowledgement, Block: 1

Figure 15: Tráfego de pacotes durante a execução no PC

1583 1563.7263624..	10.1.1.1	10.4.4.1	TFTP	56 Read Request, File: file1, Transfer type: octet
1584 1563.7361436..	10.4.4.1	10.1.1.1	TFTP	270 Data Packet, Block: 1 (last)
1585 1563.7368925..	10.1.1.1	10.4.4.1	TFTP	46 Acknowledgement, Block: 1

Figure 16: Tráfego de pacotes durante a execução no Portátil

2.5 HTTP

O protocolo HTTP, de forma semelhante a outros protocolos falados acima, utiliza a tecnologia de transporte TCP. Deste modo, a forma com que o cliente e o servidor criam e terminam uma ligação é semelhante aos outros protocolos, pois todas estas aplicações "cederam" o controlo sobre o estabelecimento de ligações à mesma tecnologia de transporte.

Depois da conexão ser estabelecida, o cliente pede ao servidor os ficheiros que precisa (*GET file1/2*). Após receber esse pedido, o servidor envia pacotes por TCP com os dados (*TCP segment of a reassembled PDU*), aos quais o cliente responde com um ACK. Após a transferência de dados terminar, o servidor envia ao cliente um pacote de controlo com código 200 (OK), a sinalizar que terminou a transferência do ficheiro. Após isso, a conexão é terminada como referido acima.

Usando este protocolo a camada de transporte TCP, a perda de pacotes observada no PC é tratada de forma semelhante a outros protocolos com esta tecnologia. Quando há perda de um pacote ACK, o servidor assume que o pacote que enviou não foi recebido pelo cliente e reenvia-o. Se um pacote de dados for perdido, o cliente enviará ACKs do pacote anterior a esse, de modo a sinalizar que ainda precisa do que foi perdido. Quando existe duplicação de pacotes, estes apenas são descartados.

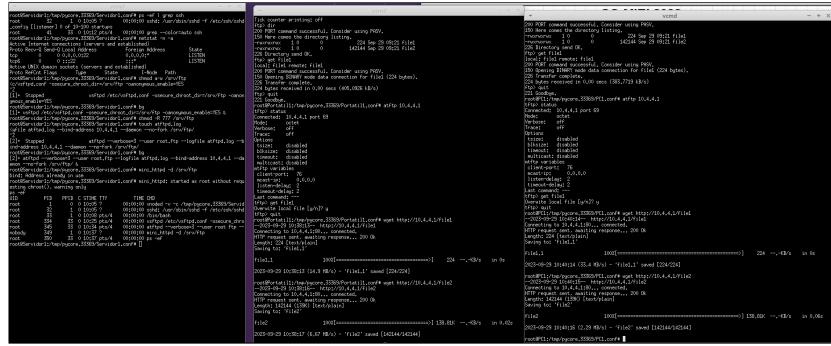


Figure 17: Execução dos comandos nos dispositivos

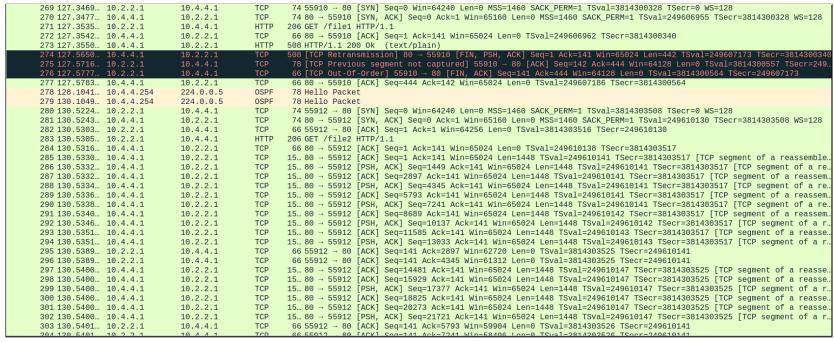


Figure 18: Tráfego de pacotes durante a execução no PC

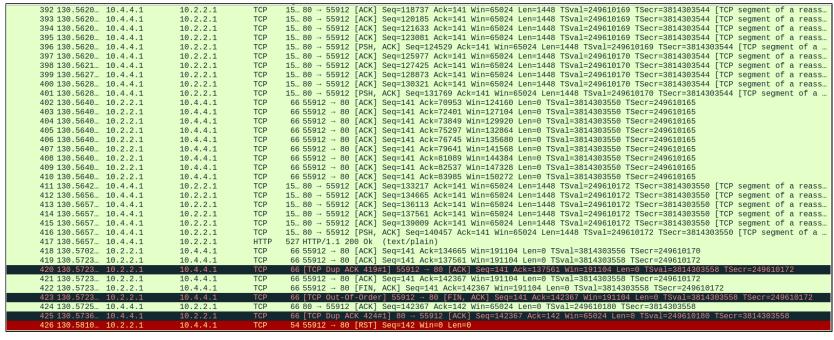


Figure 19: Tráfego de pacotes durante a execução no PC

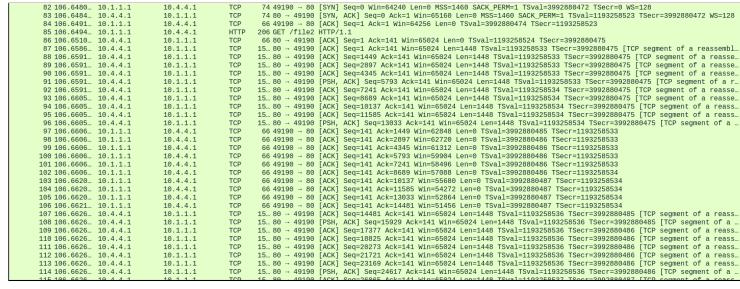


Figure 20: Tráfego de pacotes durante a execução no Portátil

Figure 21: Tráfego de pacotes durante a execução no Portátil

Para o trabalho experimental, a equipa docente disponibilizou uma topologia de rede que se baseia em oito hosts, quatro switchs e sete routers.

Os resultados são levados a cabo pela interação entre três hosts: **Servidor1**, **Portátil1**, **PC1**.

A baixo colocamos uma figura com a topologia e os destaques supramencionados.

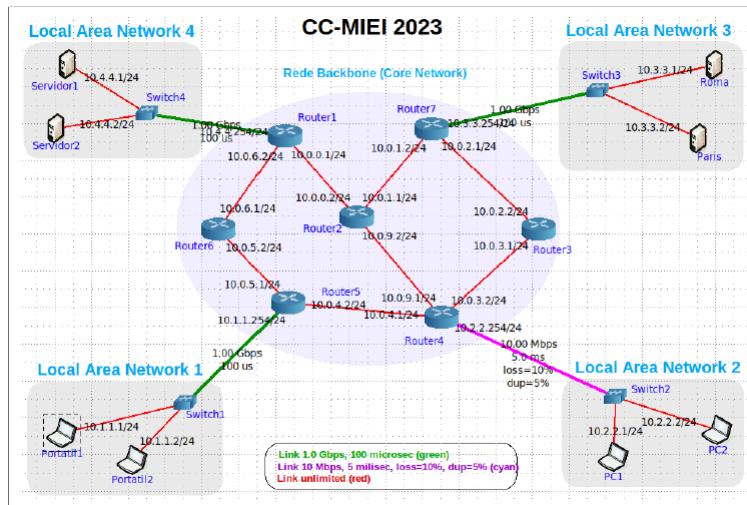


Figure 22: Topologia e os seus destaques

2.6 Parte I: Instalação, configuração e utilização de serviços de transferência de ficheiros

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.

Relativamente ao protocolo SFTP, o mesmo utiliza o TCP como protocolo de transporte. Ou seja, quando um pacote é enviado, o remetente inicia um temporizador. Caso esse temporizador expire antes de ser recebido um pacote de confirmação (ACK) do destinatário, confirmando que o pacote foi recebido, o remetente volta a transmitir o pacote. Às vezes, os dois pacotes podem acabar por chegar ao destino, originando uma duplicação. Neste caso, o pacote adicional

é simplesmente descartado e é enviado um pacote de confirmação apenas do pacote correto.

885 780.433176492 10.2.2.1	10.4.4.1	TCP	66 43964 - 22 [ACK] Seq=42 Ack=42 Win=64256 Len=0 Tsval=11835772..
886 780.433597258 10.2.2.1	10.4.4.1	SSHv2	10893 Server - 22 [ACK] Seq=42 Ack=42 Win=64256 Len=1448 Tsval=11835..
887 780.437569157 10.2.2.1	10.4.4.1	TCP	1514 43964 - 22 [ACK] Seq=42 Ack=42 Win=64256 Len=1448 Tsval=11835..
888 780.438218851 10.4.4.1	10.2.2.1	TCP	66 22 - 43964 [ACK] Seq=1066 Ack=1490 Win=64128 Len=0 Tsval=7833..
889 780.439702104 10.2.2.1	10.4.4.1	TCP	66 [TCP Previous segment not captured] 43964 - 22 [ACK] Seq=1554..
890 780.457267289 10.2.2.1	10.4.4.1	SSHv2	114 Client: Diffie-Hellman Key Exchange Init..
891 780.457652864 10.4.4.1	10.2.2.1	TCP	78 [TCP Dup ACK 89881] 22 - 43964 [ACK] Seq=1066 Ack=1490 Win=64..
893 780.873709442 10.2.2.1	10.4.4.1	TCP	138 [TCP Retransmission] 43964 - 22 [PSH, ACK] Seq=1490 Ack=1066..
894 780.87956874 10.4.4.1	10.2.2.1	TCP	66 22 - 43964 [ACK] Seq=1066 Ack=1692 Win=64128 Len=0 Tsval=7833..
895 780.898593797 10.4.4.1	10.2.2.1	SSHv2	1182 Server: Diffie-Hellman Key Exchange Reply: New Keys, Encrypt..
897 780.895816323 10.2.2.1	10.4.4.1	TCP	66 43964 - 22 [ACK] Seq=1062 Ack=2182 Win=64128 Len=0 Tsval=1183..
900 780.0000006502 10.2.2.1	10.4.4.1	SSHv2	97 Client: New Keys..

Figure 23: Perda e duplicação de pacotes em SFTP

Conforme observámos nas experiências que realizámos, no caso do PC, o mesmo envia um pacote com Seq=42 e Ack=42. Ou seja, espera que o próximo pacote recebido tenha o número de sequência 42. No entanto, o servidor responde com um pacote com Seq=1066 e Ack=1490. O cliente, como não obtém o número de sequência que esperava, envia um pacote ao servidor a indicar que poderão estar a faltar pacotes. O servidor volta a enviar o mesmo pacote (retransmissão) uma vez que não obteve a confirmação da receção do mesmo. O PC responde então com o número de sequência pedido (1490), confirmando a receção e ignorando o pacote enviado duas vezes. A partir daí, o tráfego volta a fluir normalmente. Muito provavelmente, houve uma reorganização da rede, o que causou a confusão relativa aos números de sequência.

No protocolo FTP também é utilizado o protocolo TCP na camada de transporte, ou seja, os procedimentos em caso de duplicação e perda de pacotes são semelhantes ao SFTP.

1328 1191.9894462 10.4.4.1	10.2.2.1	TCP	66 20 - 38405 [FIN, ACK] Seq=127 Ack=1 Win=64256 Len=0 Tsval=783..
1329 1191.9943383 10.2.2.1	10.4.4.1	TCP	66 37186 - 21 [ACK] Seq=61 Ack=187 Win=64256 Len=0 Tsval=1183988..
1330 1191.9943392 10.2.2.1	10.4.4.1	TCP	66 38405 - 20 [ACK] Seq=1 Ack=127 Win=65152 Len=0 Tsval=1183988..
1331 1192.2956827 10.4.4.1	10.2.2.1	TCP	66 [TCP Retransmission] 20 - 38405 [FIN, ACK] Seq=127 Ack=1 Win=..
1332 1192.2148708 10.2.2.1	10.4.4.1	TCP	78 [TCP Previous segment not captured] 38405 - 20 [ACK] Seq=2 Ac..
1333 1192.2148716 10.2.2.1	10.4.4.1	TCP	78 [TCP Dup ACK 1332#1] 38405 - 20 [ACK] Seq=2 Ack=128 Win=65152..
1334 1192.2167188 10.4.4.1	10.2.2.1	FTP	98 Response: 226 Directory send OK..
1335 1192.2223438 10.2.2.1	10.4.4.1	TCP	66 37186 - 21 [ACK] Seq=61 Ack=211 Win=64256 Len=0 Tsval=1183988..
1336 1192.4224174 10.2.2.1	10.4.4.1	TCP	66 [TCP Retransmission] 38405 - 20 [FIN, ACK] Seq=1 Ack=128 Win=..
1337 1192.4228616 10.4.4.1	10.2.2.1	TCP	66 20 - 38405 [ACK] Seq=128 Ack=2 Win=64256 Len=0 Tsval=78377336..
1338 1193.3252280 10.4.4.254	224.0.0.5	OSPF	78 Hello Packet..
1339 1194.1893288 10.2.2.1	10.4.4.1	FTP	74 Request: TYPE I..

Figure 24: Perda e duplicação de pacotes em FTP

Observando a retransmissão e duplicação de pacotes que ocorreu no teste com FTP, concluímos que de facto o processo é igual ao que ocorre no protocolo SFTP.

No protocolo TFTP não ocorreram duplicações ou perdas de pacotes, no entanto caso existissem não haveria qualquer plano de confirmação ou retransmissão dos pacotes associada ao protocolo UDP. Apesar disso, verificámos o envio de pacotes de confirmação ACK o que sugere que foi implementado um mecanismo de confirmação de pacotes ao nível da camada de aplicação.

Já no protocolo HTTP, a perda e duplicação de pacotes é tratada de forma semelhante aos restantes protocolos que utilizam TCP, retransmitindo os pacotes perdidos e ignorando os duplicados. Sendo portanto a camada de transporte a lidar com estes problemas.

269 127.3469.. 10.2.2.1	10.4.4.1	TCP	74 55910 - 80 [SYN] Seq=0 Win=64248 Len=0 MSS=1460 SACK_PERM=1 Tsvval=3814300328 Tsecr=0 WS=128	
270 127.3477.. 10.4.4.1	10.2.2.1	TCP	74 80 - 55910 [SYN, ACK] Seq=0 Ack=1 Win=65160 Len=0 MSS=1460 SACK_PERM=1 Tsvval=249606955 Tsecr=3814300328 WS=128	
271 127.3535.. 10.2.2.1	10.4.4.1	HTTP	206 GET /file1 HTTP/1.1	
272 127.3542.. 10.4.4.1	10.2.2.1	TCP	68 55910 - 55918 [ACK] Seq=1 Ack=141 Win=65024 Len=0 Tsvval=249606962 Tsecr=3814300349	
273 127.3556.. 10.4.4.1	10.2.2.1	HTTP	508 HTTP/1.1 200 OK (text/plain)	
274 127.3563.. 10.2.2.1	10.4.4.1	TCP	56 55910 - 55924 [TCP Previous segment not captured] [FIN, PSH, ACK] Seq=3 Ack=141 Win=65024 Len=0 Tsvval=3814300349 Tsecr=3814300349	
275 127.5716.. 10.2.2.1	10.4.4.1	TCP	78 [TCP Previous segment not captured] 55910 - 80 [ACK] Seq=142 Ack=444 Win=64128 Len=0 Tsvval=3814300557 Tsecr=249..	
276 127.5777.. 10.2.2.1	10.4.4.1	TCP	66 [TCP Out-Of-Order] 55910 - 80 [FIN, ACK] Seq=141 Ack=444 Win=64128 Len=0 Tsvval=3814300564 Tsecr=249667173	
277 127.5783.. 10.4.4.1	10.2.2.1	TCP	66 80 - 55910 [ACK] Seq=444 Ack=142 Win=65024 Len=0 Tsvval=249607186 Tsecr=3814300564	
278 128.1841.. 10.4.4.254	224.0.0.5	OSPF	78 Hello Packet	
279 130.1849.. 10.4.4.254	224.0.0.5	OSPF	78 Hello Packet	
280 130.5224.. 10.2.2.1	10.4.4.1	TCP	74 55912 - 55918 [SYN] Seq=0 Win=64248 Len=0 MSS=1460 SACK_PERM=1 Tsvval=3814300508 Tsecr=0 WS=128	
281 130.5231.. 10.4.4.1	10.2.2.1	TCP	74 80 - 55912 [SYN, ACK] Seq=0 Ack=1 Win=65160 Len=0 MSS=1460 SACK_PERM=1 Tsvval=249610138 Tsecr=3814300508 WS=128	
282 130.5303.. 10.2.2.1	10.4.4.1	TCP	66 55912 - 80 [ACK] Seq=1 Ack=1 Win=64256 Len=0 Tsvval=38143003516 Tsecr=249610138	
283 130.5305.. 10.2.2.1	10.4.4.1	HTTP	206 GET /file2 HTTP/1.1	

Figure 25: Perda e duplicação de pacotes em HTTP

Durante a experiência verificamos que os procedimentos em caso de duplicação e perda de pacotes são os mesmos que nos protocolos anteriores que também usam TCP.

Concluímos que a camada responsável por lidar com possíveis perdas e duplicações é a camada de aplicação no caso do protocolo TFTP, e a camada de transporte nas restantes aplicações.

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 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.

Nesta primeira imagem podemos ver toda a troca de pacotes, com protocolo FTP ao nível da aplicação, desde a autenticação até ao fim da tranferência. Encontram-se realçados dois pacotes com FTP-DATA: um que corresponde ao pedido de listagem de diretórias e outro com o conteúdo do *file1*. Todos os pacotes utilizam o protocolo TCP ao nível da camada de transporte.

Ipt or ftp-data						
No.	Time	Source	Destination	Protocol	Length	Info
65	99.990154964	10.4.4.1	10.1.1.1	FTP	86	Response: 220 (vsFTPD 3.0.3)
69	95.654200294	10.1.1.1	10.4.4.1	FTP	82	Request: USER anonymous
73	95.655020070	10.1.1.1	10.4.4.1	FTP	189	Response: 331 Please specify the password.
74	96.908360628	10.1.1.1	10.4.4.1	FTP	74	Request: PASS a
76	96.913729643	10.4.4.1	10.1.1.1	FTP	89	Response: 230 Login successful.
78	96.914154418	10.1.1.1	10.4.4.1	FTP	72	Request: SYST
80	96.914609357	10.4.4.1	10.1.1.1	FTP	85	Response: 215 UNIX Type: L8
86	96.914609357	10.4.4.1	10.1.1.1	FTP	71	Request: PWD
87	102.844418273	10.4.4.1	10.1.1.1	FTP	108	Response: 257 "/" is the current directory
90	104.800979344	10.1.1.1	10.4.4.1	FTP	89	Request: PORT 10.1.1.1,224,241
91	104.801844571	10.4.4.1	10.1.1.1	FTP	117	Response: 200 PORT command successful. Consider using PASV.
93	104.802610425	10.1.1.1	10.4.4.1	FTP	72	Request: LIST
97	104.8089273056	10.1.1.1	10.1.1.1	FTP	105	Response: 150 Here comes the directory listing.
98	104.809805768	10.1.1.1	10.1.1.1	FTP	103	FTP Data: 126 bytes (PORT) (LIST)
104	104.811561736	10.4.4.1	10.1.1.1	FTP	90	Response: 226 Transfer complete.
134	118.162804952	10.1.1.1	10.4.4.1	FTP	72	Request: QUIT
135	118.163513833	10.4.4.1	10.1.1.1	FTP	88	Response: 221 Goodbye.

Figure 26: Pacotes com FTP durante a transferência de file1

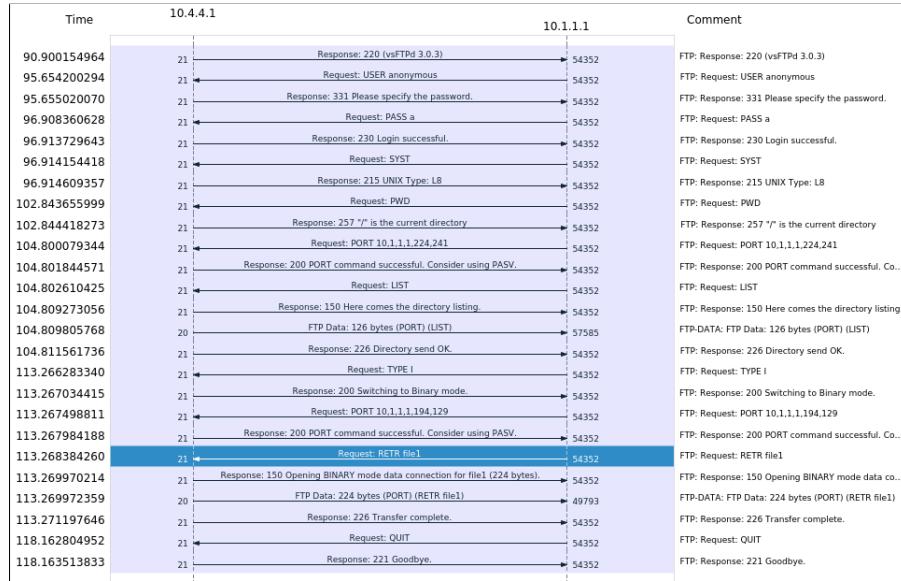


Figure 27: Diagrama temporal de FTP

Ignorando o processo de autenticação do FTP e sucessivos pedidos de LIST e PWD, é possível verificar que o processo de transferência ocorre com um pedido de "RETR file1" (*retrieve file1*) do Portátil1 para o Servidor1. O Servidor1 reconhece o pedido e responde dando inicio a uma conexão para um processo de transferência. De seguida ocorre a transferência de dados dentro de um pacote FTP-DATA onde o Servidor1 envia o ficheiro pedido em blocos. Quando termina, envia um pacote a indicar que a transferência acabou e de seguida o Portatill envia um pacote a indicar o fim da conexão.

95 104.890601026 10.4.4.1	10.1.1.1	TCP	66	20 57585 [ACK] Seq=1 Vim=4956 Len=0 TStamp=4197922397...
97 104.899273856 10.4.4.1	10.1.1.1	FTP	105	Response: 156 Here comes the directory listing.
98 104.899805768 10.4.4.1	10.1.1.1	FTP-DATA	192	FTP Data: 126 bytes (PORT) (LIST)

Figure 28: Pacotes de dados e confirmação

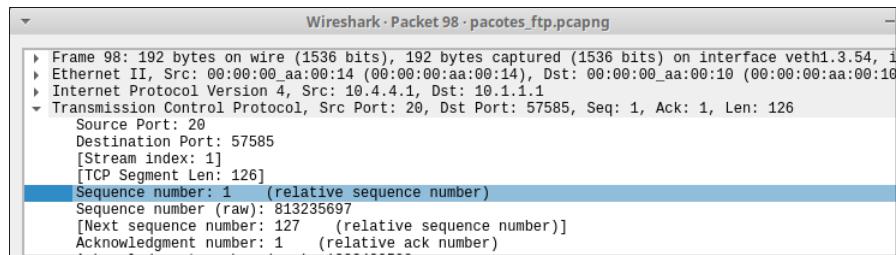


Figure 29: Número de sequência do pacote de dados

No pacote de dados FTP-DATA de *file1*, o número de sequência indicado é igual a 1. A nível da confirmação (pacote TCP "ACK") que ocorre antes da transferência, o número de sequência indicado é também 1.

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.

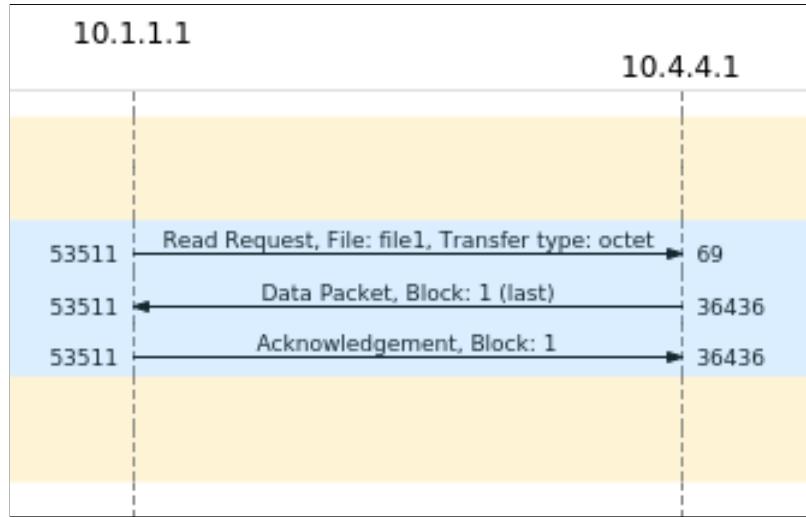


Figure 30: Diagrama temporal

Ocorre uma primeira fase de conexão onde o portátil envia um pedido de leitura para o servidor, de seguida ocorre uma transferência de dados onde o servidor envia o file pedido em blocos, o portátil confirma a receção de cada bloco com um pacote ACK, neste caso é apenas enviado 1 bloco de dados encerrando assim a conexão.

Identificamos 2 tipos de segmentos: pacotes RRQ e ACK de portátil para servidor e pacote dos dados solicitados do servidor para o portátil.

O número de sequência permite ao remetente e ao destinatário saber quais pacotes foram enviados e quais foram recebidos, controlando possíveis duplicações e perdas. Neste caso, tanto no pacote de dados quanto no pacote de ACK, o número de sequência indicado é 1, o que significa que o pacote foi enviado com sucesso.

4 2.597666... 10.4.4.1	10.1.1.1	TFTP 270 Data Packet, Block: 1 (last)
5 2.597883... 10.1.1.1	10.4.4.1	TFTP 46 Acknowledgement, Block: 1

Figure 31: Números de Sequência

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;

Comparando as quatro aplicações quanto à camada de transporte, o protocolo TFTP utiliza UDP enquanto os restantes utilizam o protocolo TCP.

Relativamente à eficiência, o protocolo TFTP, por usar o UDP na camada de transporte, é o mais eficiente. O protocolo UDP é o que apresenta menor overhead e o mais rápido.

Quanto à complexidade, percebemos que o protocolo menos complexo é o TFTP uma vez que é o que funciona com menos pacotes extra para realizar a transmissão do file1. O mais complexo é o SFTP principalmente pela encriptação dos dados transmitidos. Entre HTTP e FTP podemos afirmar que o FTP é ligeiramente mais complexo, uma vez que durante a transmissão do file1 foram feitos mais pedidos e foram transmitidos mais pacotes.

Comparando finalmente a segurança, o SFTP é o protocolo mais seguro, uma vez que todos os dados transmitidos são encriptados. No entanto, poderá ser implementada uma versão dos restantes protocolos com uso de encriptação de modo a aumentar a segurança.

2.7 Parte II: Uso da camada de transporte por parte das aplicações

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.

Tendo em conta que:

$$\text{Overhead} = \text{Tamanho Total do Pacote} - \text{Payload(Dados Úteis)}$$

App	App layer	Transport layer	Porta	Overhead
wget, lynx, via browser	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 ou dig	DNS	UDP	53	8
Ping	Não aplicável	Não aplicável	Não aplicável	Não aplicável
Traceroute	Não aplicável	UDP	Não aplicável	8

Nos anexos estão expostas as imagens dos pacotes que foram usados

3 Comentários Finais

Em modo de conclusão, neste trabalho prático aprofundámos o nosso conhecimento relativamente aos vários protocolos ao nível da camada de transporte e de aplicação. Deste modo, foi possível comparar o funcionamento do TCP e UDP e quais os protocolos de aplicação que os aplicam ao nível do transporte de dados.

4 Anexos

4.1 Imagens Parte II:

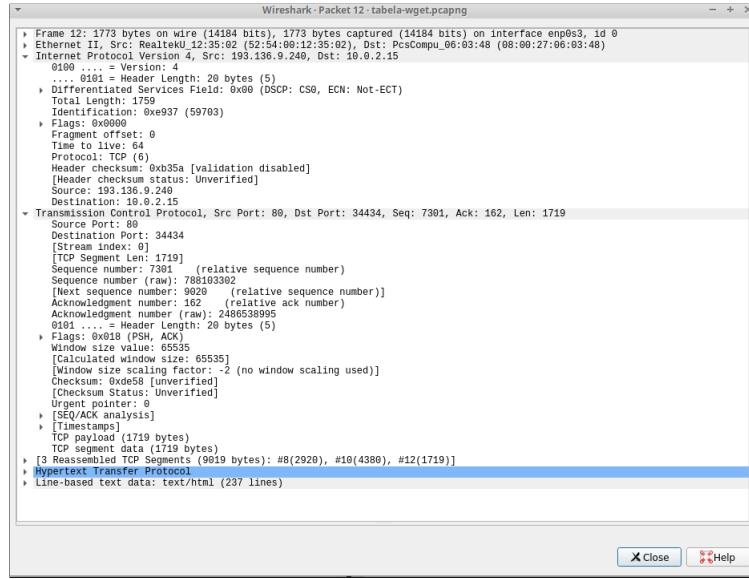


Figure 32: Pacote com wget

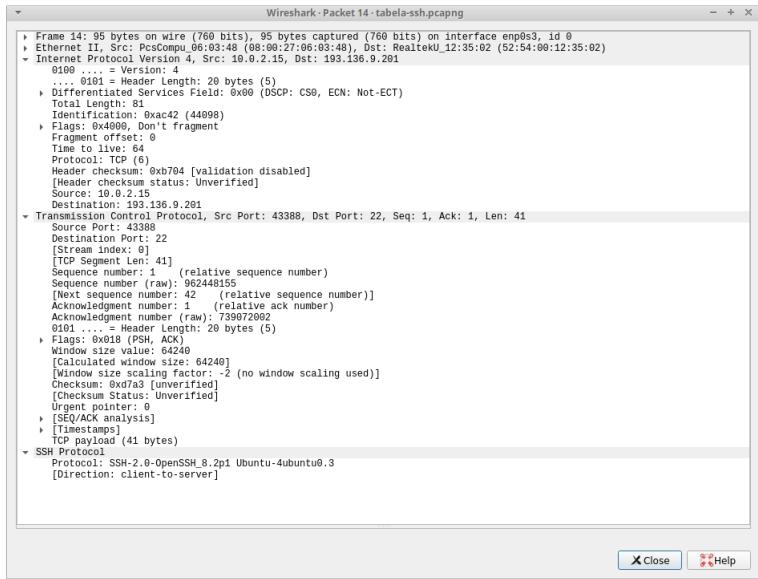


Figure 33: Pacote com ssh

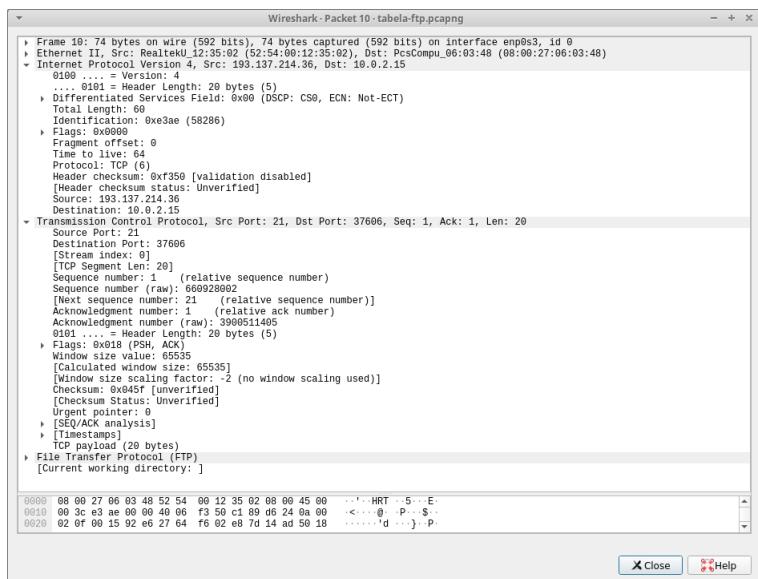


Figure 34: Pacote com ftp

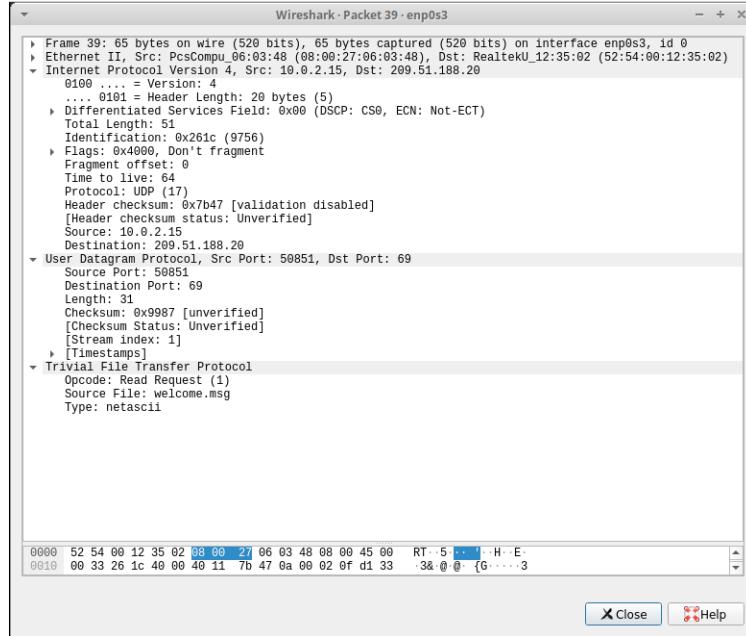


Figure 35: Pacote com tftp

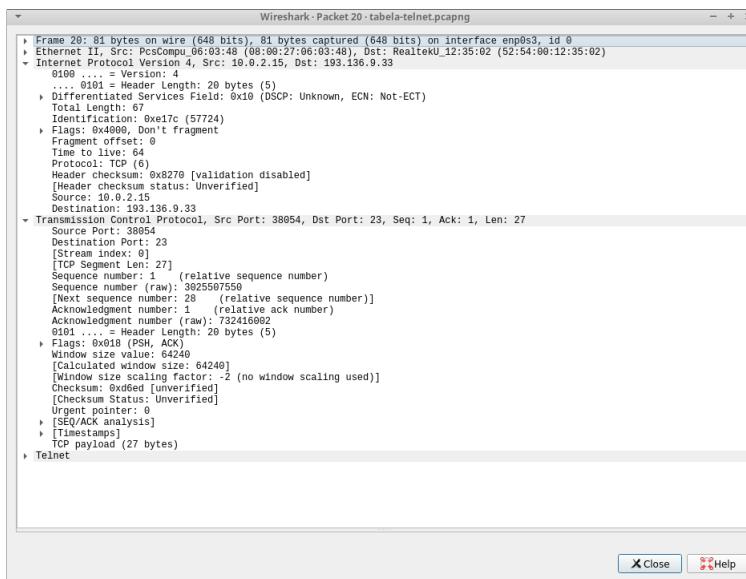


Figure 36: Pacote com telnet

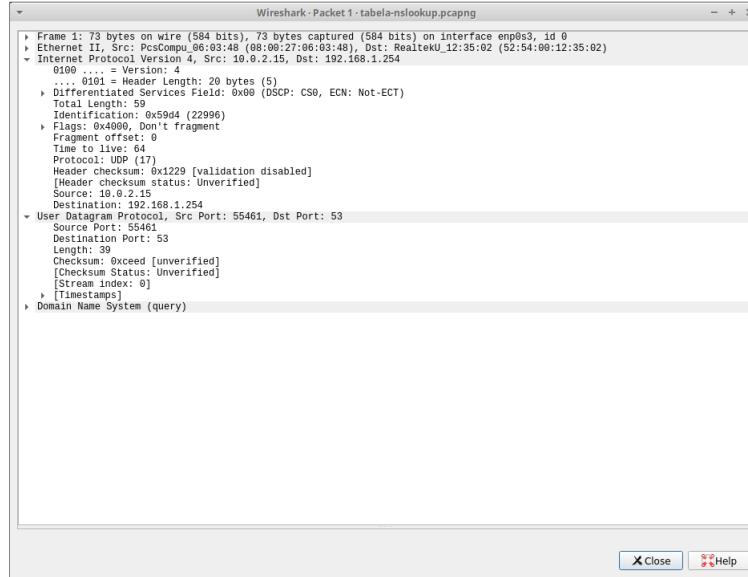


Figure 37: Pacote com nslookup

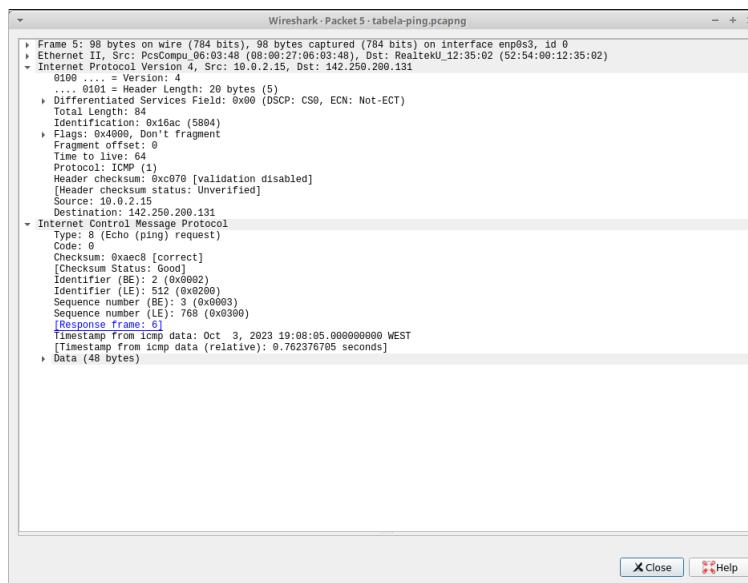


Figure 38: Pacote com ping