

Comunicações por Computador

Trabalho Prático 1: Protocolos da Camada de Transporte

Ana Rita Peixoto, Leonardo Marreiros, and Luís Pinto

University of Minho, Department of Informatics, 4710-057 Braga, Portugal
e-mail: {a89612,a89537,a89506}@alunos.uminho.pt

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	Protocolo de transporte	Porta de atendimento	Overhead de transporte em bytes
Ping	-	-	-	-
tracert	TRACEROUTE	UDP	33446	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

41	11.695607585	10.0.2.15	216.58.215.142	ICMP	>	Frame 44: 100 bytes on wire (800 bits), 100 bytes captured (800 bits)
42	11.713705981	216.58.215.142	10.0.2.15	ICMP	>	Linux cooked capture v1
43	12.697592506	10.0.2.15	216.58.215.142	ICMP	>	Internet Protocol Version 4, Src: 216.58.215.142, Dst: 10.0.2.15
44	12.715121334	216.58.215.142	10.0.2.15	ICMP	>	0100 = Version: 4
45	13.699355072	10.0.2.15	216.58.215.142	ICMP	> 0101 = Header Length: 20 bytes (5)
46	13.716240184	216.58.215.142	10.0.2.15	ICMP	>	Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
47	14.701383941	10.0.2.15	216.58.215.142	ICMP	>	Total Length: 84
48	14.718332256	216.58.215.142	10.0.2.15	ICMP	>	Identification: 0xb9e6 (2534)
49	15.702225809	10.0.2.15	216.58.215.142	ICMP	>	Flags: 0x00
50	15.719138536	216.58.215.142	10.0.2.15	ICMP	>	Fragment Offset: 0
51	16.704999227	10.0.2.15	216.58.215.142	ICMP	>	Time to Live: 59
52	16.722781170	216.58.215.142	10.0.2.15	ICMP	>	Protocol: ICMP (1)
53	17.706672951	10.0.2.15	216.58.215.142	ICMP	>	Header Checksum: 0xb9eb [validation disabled]
54	17.723777725	216.58.215.142	10.0.2.15	ICMP	>	[Header checksum status: Unverified]
55	18.708763545	10.0.2.15	216.58.215.142	ICMP	>	Source Address: 216.58.215.142
56	18.726635093	216.58.215.142	10.0.2.15	ICMP	>	Destination Address: 10.0.2.15
57	19.709730840	10.0.2.15	216.58.215.142	ICMP	>	Internet Control Message Protocol

Fig. 1: Ping

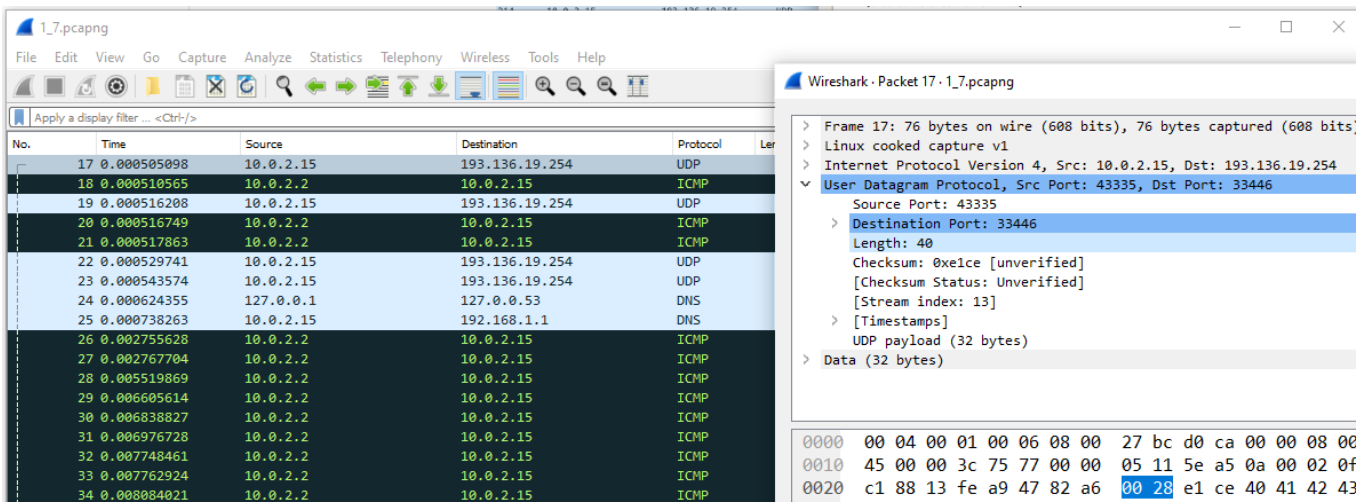


Fig. 2: TRACERoute

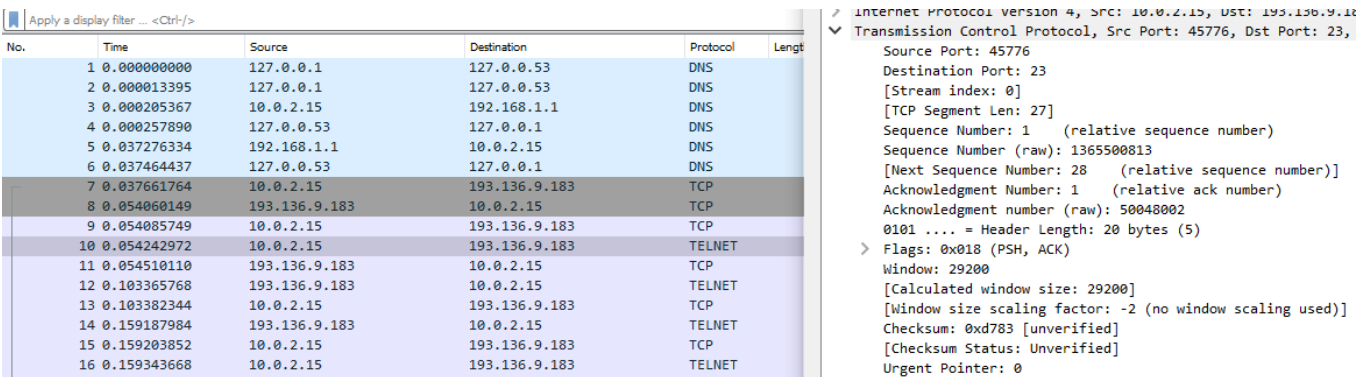


Fig. 3: TELNET

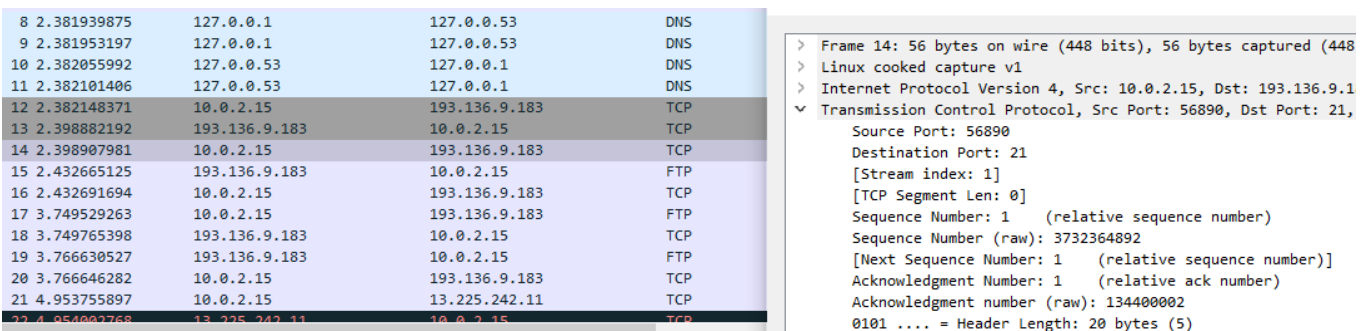


Fig. 4: FTP

No.	Time	Source	Destination	Protocol	Length
1	0.000000000	127.0.0.1	127.0.0.53	DNS	
2	0.000013184	127.0.0.1	127.0.0.53	DNS	
3	0.000157962	10.0.2.15	192.168.1.1	DNS	
4	0.000216591	127.0.0.53	127.0.0.1	DNS	
5	0.051395808	192.168.1.1	10.0.2.15	DNS	
6	0.051562663	127.0.0.53	127.0.0.1	DNS	
7	0.062226650	10.0.2.15	193.136.9.183	TFTP	
8	7.314336965	10.0.2.15	193.136.9.183	TFTP	
9	12.406550264	PcsCompu_bc:d0:ca		ARP	
10	12.406750213	RealtekU_12:35:02		ARP	
11	14.322994409	10.0.2.15	193.136.9.183	TFTP	
12	21.505246269	10.0.2.15	193.136.9.183	TFTP	
13	28.533155794	10.0.2.15	193.136.9.183	TFTP	

Frame 7: 88 bytes on wire (704 bits), 88 bytes captured (704 bits) on interface eth0

Linux cooked capture v1

Internet Protocol Version 4, Src: 10.0.2.15, Dst: 193.136.9.183

User Datagram Protocol, Src Port: 53600, Dst Port: 69

Source Port: 53600

Destination Port: 69

Length: 52

Checksum: 0xd793 [unverified]

[Checksum Status: Unverified]

[Stream index: 2]

[Timestamps]

UDP payload (44 bytes)

Trivial File Transfer Protocol

Fig. 5: TFTP

Time	Source	Destination	Protocol	Length	Info
1	0.000000000	10.0.2.15	TCP	56	46962 → 80 [ACK]
2	0.000094876	10.0.2.15	TCP	56	46960 → 80 [ACK]
3	0.000120006	10.0.2.15	TCP	56	46958 → 80 [ACK]
4	0.000475618	216.58.211.227	TCP	62	[TCP ACKed uns]
5	0.000483949	216.58.211.227	TCP	62	[TCP ACKed uns]
6	0.000485260	216.58.211.227	TCP	62	[TCP ACKed uns]
7	0.727303196	127.0.0.1	DNS	97	Standard query
8	0.727322777	127.0.0.1	DNS	97	Standard query
9	0.727531883	10.0.2.15	HTTP	352	GET /success.t
10	0.727598918	10.0.2.15	DNS	86	Standard query
11	0.727650784	10.0.2.15	DNS	86	Standard query
12	0.728078101	34.107.221.82	TCP	62	80 → 32836 [ACK]
13	0.732142745	192.168.1.254	DNS	102	Standard query
14	0.732324863	127.0.0.53	DNS	113	Standard query
15	0.746019961	192.168.1.254	DNS	209	Standard query
16	0.746293400	127.0.0.53	DNS	220	Standard query
17	0.755377170	34.107.221.82	HTTP	276	HTTP/1.1 200 OK
18	0.755401446	10.0.2.15	TCP	56	32836 → 80 [ACK]
19	0.756572990	127.0.0.1	DNS	84	Standard query
20	0.756589091	127.0.0.1	DNS	84	Standard query

Internet Protocol Version 4, Src: 10.0.2.15, Dst: 34.107.221.82

Transmission Control Protocol, Src Port: 32836, Dst Port: 80

Source Port: 32836

Destination Port: 80

[Stream index: 3]

[TCP Segment Len: 296]

Sequence number: 1 (relative sequence number)

[Next sequence number: 297 (relative sequence number)]

Acknowledgment number: 1 (relative ack number)

0101 = Header Length: 20 bytes (5)

Flags: 0x018 (PSH, ACK)

Window size value: 30016

[Calculated window size: 30016]

[Window size scaling factor: -1 (unknown)]

Checksum: 0x0d0f [unverified]

[Checksum Status: Unverified]

Urgent pointer: 0

[SEQ/ACK analysis]

[Timestamps]

TCP payload (296 bytes)

Hypertext Transfer Protocol

Fig. 6: HTTP

Time	Source	Destination	Protocol	Length	Info
7	1.842687937	193.136.9.183	TCP		
8	1.842707444	10.0.2.15	TCP		
9	7.566282863	127.0.0.1	UDP		
10	7.566305674	:::1	UDP		
11	7.566328917	127.0.0.1	DNS		
12	7.566521467	127.0.0.53	DNS		
13	7.566718320	127.0.0.1	DNS		
14	7.566846544	10.0.2.15	DNS		
15	7.582407382	192.168.1.1	DNS		
16	7.582596549	127.0.0.53	DNS		

Frame 11: 75 bytes on wire (600 bits), 75 bytes captured (600 bits) on interface eth0

Linux cooked capture v1

Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1

User Datagram Protocol, Src Port: 58848, Dst Port: 53

Source Port: 58848

Destination Port: 53

Length: 39

Checksum: 0xfe6e [unverified]

[Checksum Status: Unverified]

[Stream index: 2]

[Timestamps]

UDP payload (31 bytes)

Domain Name System (query)

Fig. 7: NSLOOKUP

No.	Time	Source	Destination	Protocol	Length
1	0.000000000	127.0.0.1	127.0.0.53	DNS	
2	0.000013016	127.0.0.1	127.0.0.53	DNS	
3	0.000200254	10.0.2.15	192.168.1.1	DNS	
4	0.000255922	127.0.0.53	127.0.0.1	DNS	
5	0.047564977	192.168.1.1	10.0.2.15	DNS	
6	0.047750824	127.0.0.53	127.0.0.1	DNS	
7	0.047877171	10.0.2.15	193.136.9.183	TCP	
8	0.064785481	193.136.9.183	10.0.2.15	TCP	
9	0.064831207	10.0.2.15	193.136.9.183	TCP	
10	0.065136499	10.0.2.15	193.136.9.183	SSHv2	
11	0.065310272	193.136.9.183	10.0.2.15	TCP	
12	0.118170614	193.136.9.183	10.0.2.15	SSHv2	
13	0.118186210	10.0.2.15	193.136.9.183	TCP	
14	0.118528780	10.0.2.15	193.136.9.183	SSHv2	

Frame 10: 87 bytes on wire (696 bits), 87 bytes captured (696 bits) on interface eth0

Linux cooked capture v1

Internet Protocol Version 4, Src: 10.0.2.15, Dst: 193.136.9.183

Transmission Control Protocol, Src Port: 41644, Dst Port: 22

Source Port: 41644

Destination Port: 22

[Stream index: 0]

[TCP Segment Len: 41]

Sequence Number: 1 (relative sequence number)

Sequence Number (raw): 273885562

[Next Sequence Number: 42 (relative sequence number)]

Acknowledgment Number: 1 (relative ack number)

Acknowledgment Number (raw): 86400002

0101 = Header Length: 20 bytes (5)

Flags: 0x018 (PSH, ACK)

Window: 29200

[Calculated window size: 29200]

Fig. 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 3					
No.	Time	Source	Destination	Protocol	Length Info
718	2011.1109421	10.1.1.1	10.4.4.1	TCP	74 20 → 54855 [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TSval=2321818725 TSecr=0 WS=128
719	2011.1128244	10.4.4.1	10.1.1.1	TCP	74 54855 → 20 [SYN, ACK] Seq=0 Ack=1 Win=28960 Len=0 MSS=1460 SACK_PERM=1 TSval=329056797 TSecr=2321818725 WS=128
720	2011.1129929	10.1.1.1	10.4.4.1	TCP	66 20 → 54855 [ACK] Seq=1 Ack=1 Win=29312 Len=0 TSval=2321818730 TSecr=329056797
722	2011.1135096	10.1.1.1	10.4.4.1	FTP-DA..	192 FTP Data: 126 bytes (PORT) (LIST)
723	2011.1135131	10.1.1.1	10.4.4.1	TCP	66 20 → 54855 [FIN, ACK] Seq=127 Ack=1 Win=29312 Len=0 TSval=2321818730 TSecr=329056797
724	2011.1137546	10.4.4.1	10.1.1.1	TCP	66 54855 → 20 [ACK] Seq=1 Ack=127 Win=29056 Len=0 TSval=329056799 TSecr=2321818730
725	2011.1137553	10.4.4.1	10.1.1.1	TCP	66 54855 → 20 [FIN, ACK] Seq=1 Ack=128 Win=29056 Len=0 TSval=329056799 TSecr=2321818730
726	2011.1139559	10.1.1.1	10.4.4.1	TCP	66 20 → 54855 [ACK] Seq=128 Ack=2 Win=29312 Len=0 TSval=2321818730 TSecr=329056799

Fig. 9: FTP

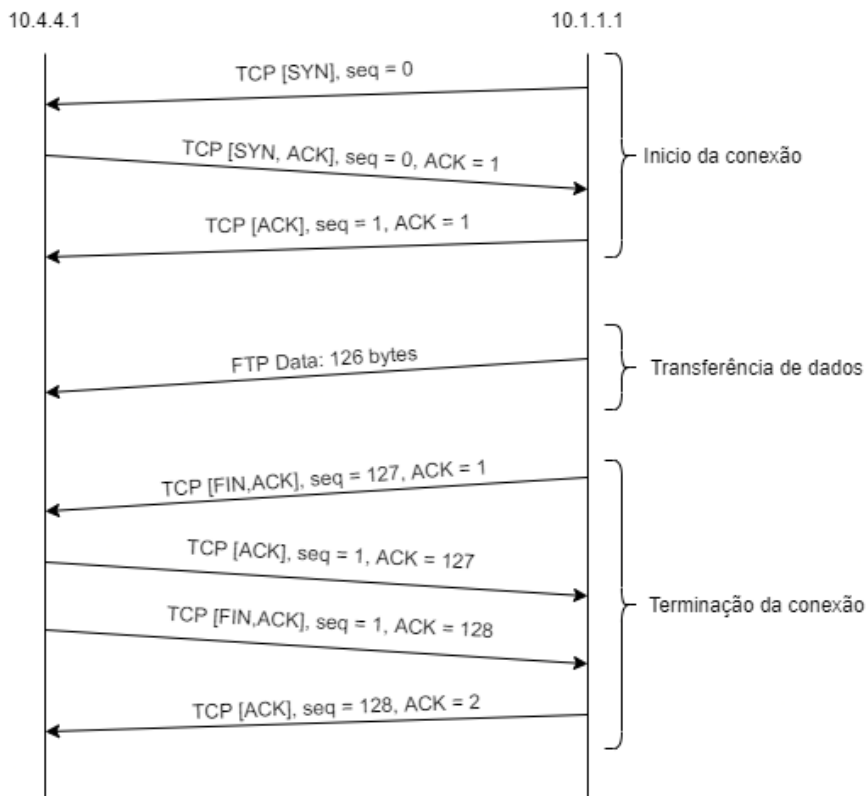


Fig. 10: Diagrama temporal das transferências do file1 por FTP

25	107.011916037	10.4.4.1	10.1.1.1	TFTP	56 Read Request, File: file1, Transfer type: octet
26	107.013641685	10.1.1.1	10.4.4.1	TFTP	46 Data Packet, Block: 1 (last)
27	107.013834105	10.4.4.1	10.1.1.1	TFTP	46 Acknowledgement, Block: 1

Fig. 11: TFTP

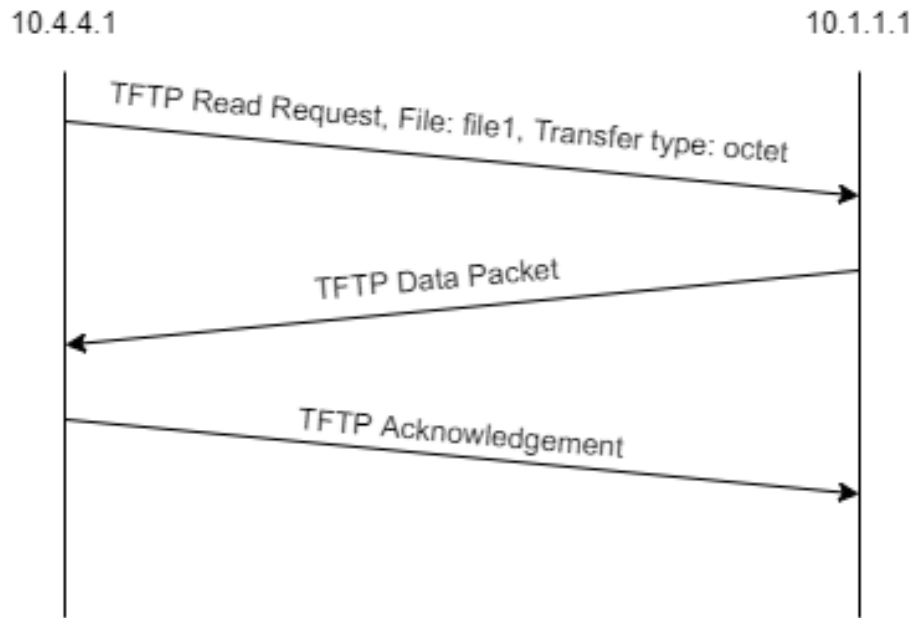


Fig. 12: Diagrama temporal das transferências do file1 por TFTP

Questão 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;

SFTP - Utiliza o protocolo TCP na camada de transporte. É um protocolo eficiente, na medida em que é possível transferir rapidamente vários ficheiros. Trata-se de uma versão segura do protocolo FTP, recorrendo a técnicas de encriptação e requer autenticação entre utilizador e servidor. Corre por cima do protocolo SSH, que fornece segurança e autenticação.

FTP - Utiliza o protocolo TCP na camada de transporte. Trata-se de um protocolo eficiente na transferência de grandes ficheiros, tanto para o uso do CPU do host como no uso da largura de banda da rede. Embora seja o protocolo de transferência de ficheiros mais comum em TCP/IP, é também o mais complexo e difícil de programar. Não é um protocolo seguro dado que não utiliza encriptação. Desta forma, os dados que os utilizadores usam para autenticação ficam vulneráveis e suscetíveis a ataques.

TFTP - Utiliza o protocolo UDP na camada de transporte. Desenhado para ser um mecanismo simples e leve para trocas de ficheiros, usa capacidade mínima e tem menor overhead tornando-se assim mais eficiente. Em termos de complexidade, é uma alternativa mais simples do protocolo FTP e é igualmente inseguro. Não utiliza encriptação e permite que os dados da autenticação e os próprios dados transferidos circulem de forma clara pela rede.

HTTP - Utiliza o protocolo TCP na camada de transporte. É mais eficiente a transferir ficheiros de pequenas dimensões, ao contrário do protocolo FTP. Não utiliza métodos de encriptação nem há garantias de privacidade (qualquer pessoa pode ver/alterar o conteúdo). Trata-se de um protocolo pouco complexo.

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).

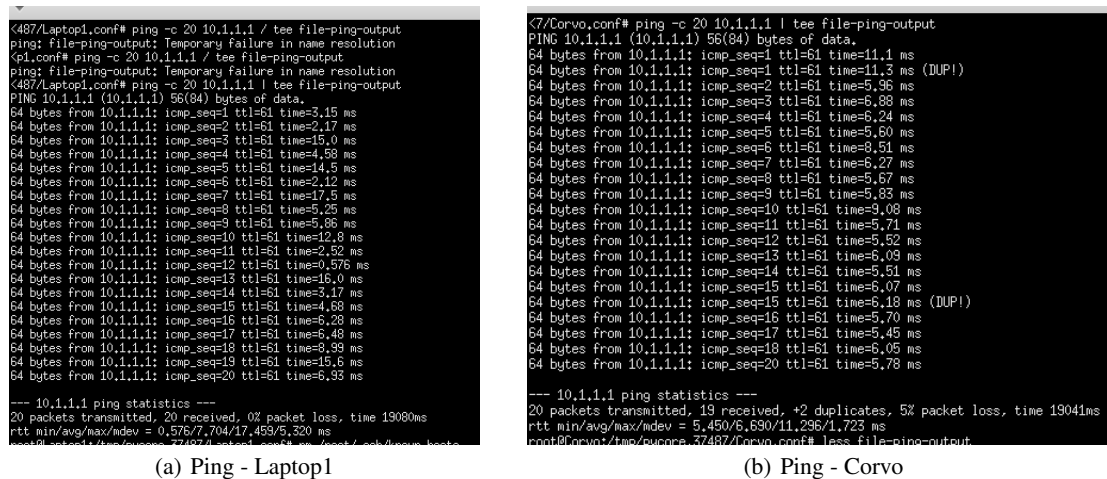


Fig. 13: Resultado do comando ping ao Server1 em diferentes clientes

Link 1.0 Gbps, 100 microsec (green)
 Link 100 Mbps, 5 milisec, loss=5%; dup=10% (cyan)

Fig. 14: Ligações

Observando as figuras Fig.13(a) e Fig.13(b) podemos verificar que, enquanto que no *Laptop1* houve 0% *packet loss* e nenhum pacote duplicado, no *Corvo* houve 5% *packet loss* e 2 pacotes duplicados (10%). A maior parte das vezes, isto acontece devido a problemas ao nível da ligação de rede, ora, como podemos verificar na topologia usada (Fig. 14), o *Laptop1* está conetado à rede *Backbone* por uma ligação de 1.0 Gbps enquanto que o *Corvo* está conetado por uma ligação de 100Mbps com 5%*loss* e 10%*dup* que vão exatamente ao encontro dos valores encontrados na Fig.13(b).

Quanto aos protocolos de transporte existem dois principais: TCP e UDP que tratam deste problema. O TCP é um protocolo orientado à conexão (*connection-oriented protocol*), envia *Delivery Acknowledgements*, re-transmissão no caso de erros, deteção fácil de erros... Os pacotes são enviados na ordem certa, usam um *handshake protocol* como *SYN*, *SYN-ACK*, *ACK*. Além disso, TCP é de confiança pois garante a entrega de informação no destino. No entanto, este protocolo atrasa a transmissão quando a rede está congestionada e a sua substituição por outro protocolo não é fácil.

Em contrapartida, o UDP é um protocolo sem conexão (*connectionless protocol*), é suportado em aplicações com uso intensivo de largura de banda que toleram a perda de pacotes, tem menos delay que o TCP, os pacotes não são enviados numa ordem fixa, cada pacote é independente, existe verificação de erros mas pacotes errados são descartados. A entrega de informação não é garantida no protocolo UDP. Como desvantagens, no UDP um pacote pode não ser entregue ou entregue fora de ordem e pacotes não são retransmitidos em caso de colisões.

Conclusão

Com este trabalho prático conseguimos consolidar os temas abordados nas aulas teóricas relativamente à camada de transporte. Em relação aos protocolos de transporte, foi possível ficar a conhecer mais acerca do funcionamento do TCP e UDP. Além disso, conseguimos aprofundar os conhecimentos e clarificar conceitos em relação aos serviços de transferência de ficheiros, tais como SFTP, FTP, TFTP e HTTP. Por fim, foi também explorada e testada a conectividade e características das ligações.