

Trabalho Prático Nº.1 – Protocolos da Camada de Transporte Comunicações por Computador

Grupo:

João Nuno Abreu

Hugo Matias

Data de entrega: 4th Março, 2020

N: A84802

N: A85370

Problem 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				
tracert				
telnet				
ftp				
tftp				
browser/http				
nslookup				
ssh				

Solution:

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	-	-	-	-
tracert	MDNS	UDP	5353	8
telnet	telnet	TCP	23	32
ftp	ftp	TCP	21	32
tftp	tftp	UDP	69	8
browser/http	http	TCP	80	32
nslookup	DNS	UDP	53	8
ssh	SSHv2	TCP	22	32

13	3.213852	172.26.111.177	172.217.18.110	ICMP	98	Echo (ping) request	id=0x9d04, seq=0/0, ttl=64 (reply in 14)
14	3.262213	172.217.18.110	172.26.111.177	ICMP	98	Echo (ping) reply	id=0x9d04, seq=0/0, ttl=48 (request in 13)
17	4.216067	172.26.111.177	172.217.18.110	ICMP	98	Echo (ping) request	id=0x9d04, seq=1/256, ttl=64 (reply in 18)
18	4.264999	172.217.18.110	172.26.111.177	ICMP	98	Echo (ping) reply	id=0x9d04, seq=1/256, ttl=48 (request in 17)
19	5.216808	172.26.111.177	172.217.18.110	ICMP	98	Echo (ping) request	id=0x9d04, seq=2/512, ttl=64 (reply in 20)
20	5.266396	172.217.18.110	172.26.111.177	ICMP	98	Echo (ping) reply	id=0x9d04, seq=2/512, ttl=48 (request in 19)
21	6.216939	172.26.111.177	172.217.18.110	ICMP	98	Echo (ping) request	id=0x9d04, seq=3/768, ttl=64 (reply in 22)
22	6.266481	172.217.18.110	172.26.111.177	ICMP	98	Echo (ping) reply	id=0x9d04, seq=3/768, ttl=48 (request in 21)
23	7.218196	172.26.111.177	172.217.18.110	ICMP	98	Echo (ping) request	id=0x9d04, seq=4/1024, ttl=64 (reply in 24)
24	7.268467	172.217.18.110	172.26.111.177	ICMP	98	Echo (ping) reply	id=0x9d04, seq=4/1024, ttl=48 (request in 23)
25	8.221988	172.26.111.177	172.217.18.110	ICMP	98	Echo (ping) request	id=0x9d04, seq=5/1280, ttl=64 (reply in 26)
26	8.271069	172.217.18.110	172.26.111.177	ICMP	98	Echo (ping) reply	id=0x9d04, seq=5/1280, ttl=48 (request in 25)
27	9.225350	172.26.111.177	172.217.18.110	ICMP	98	Echo (ping) request	id=0x9d04, seq=6/1536, ttl=64 (reply in 28)
28	9.274518	172.217.18.110	172.26.111.177	ICMP	98	Echo (ping) reply	id=0x9d04, seq=6/1536, ttl=48 (request in 27)

Figure 1: ping.

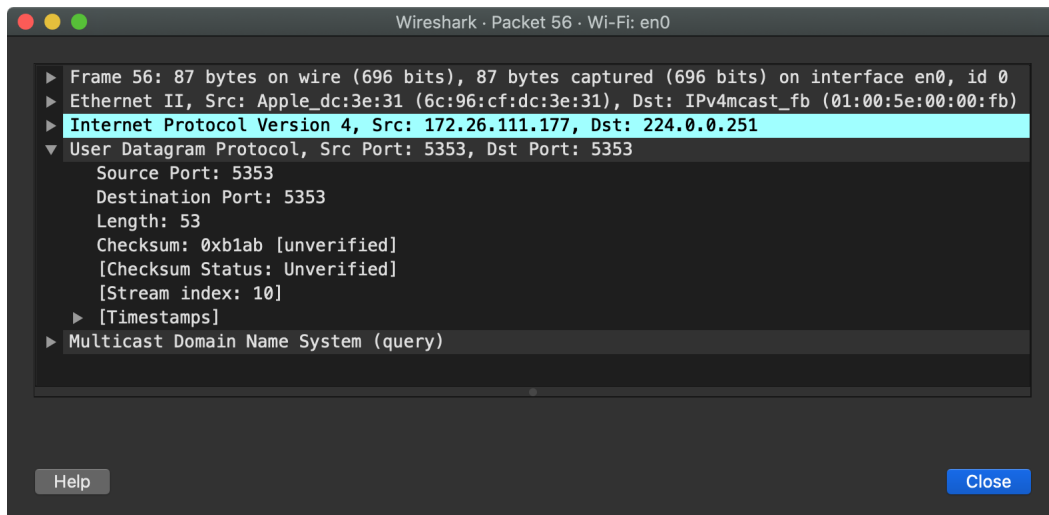


Figure 2: traceroute.

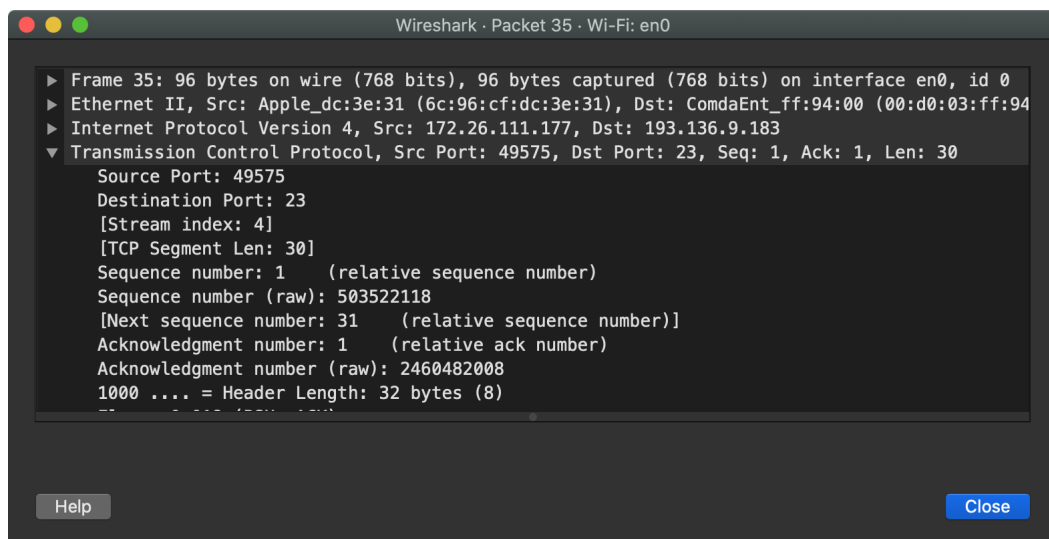


Figure 3: telnet.

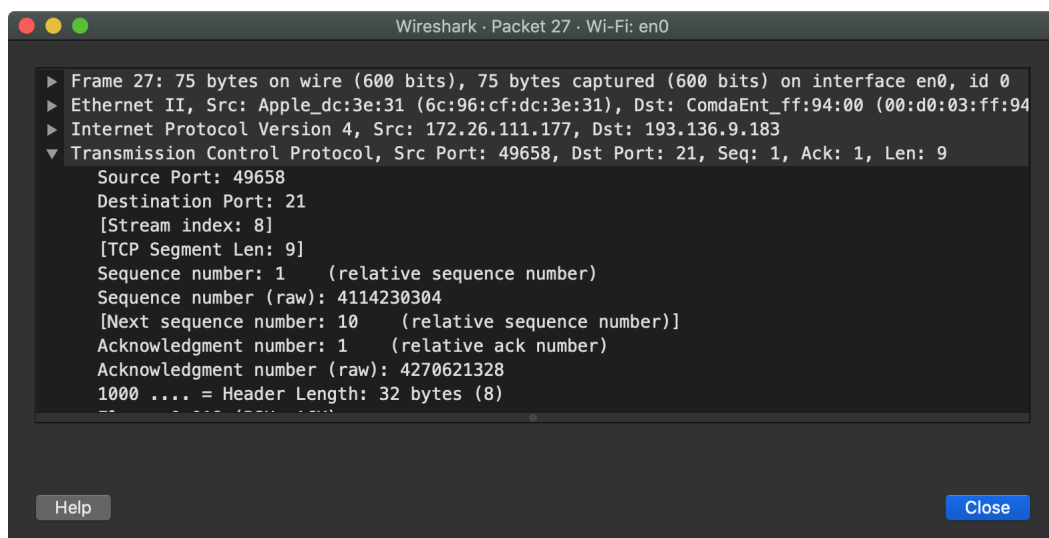


Figure 4: ftp.

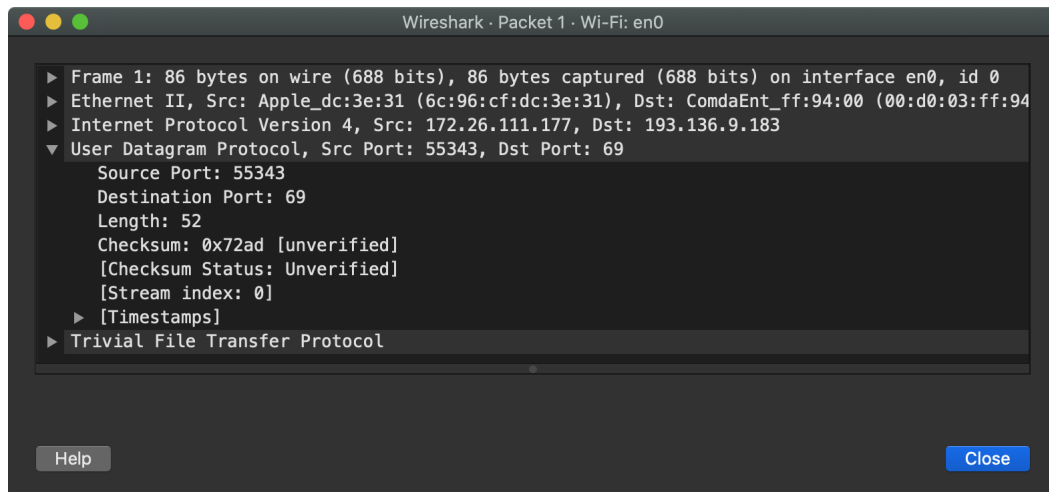


Figure 5: tftp.

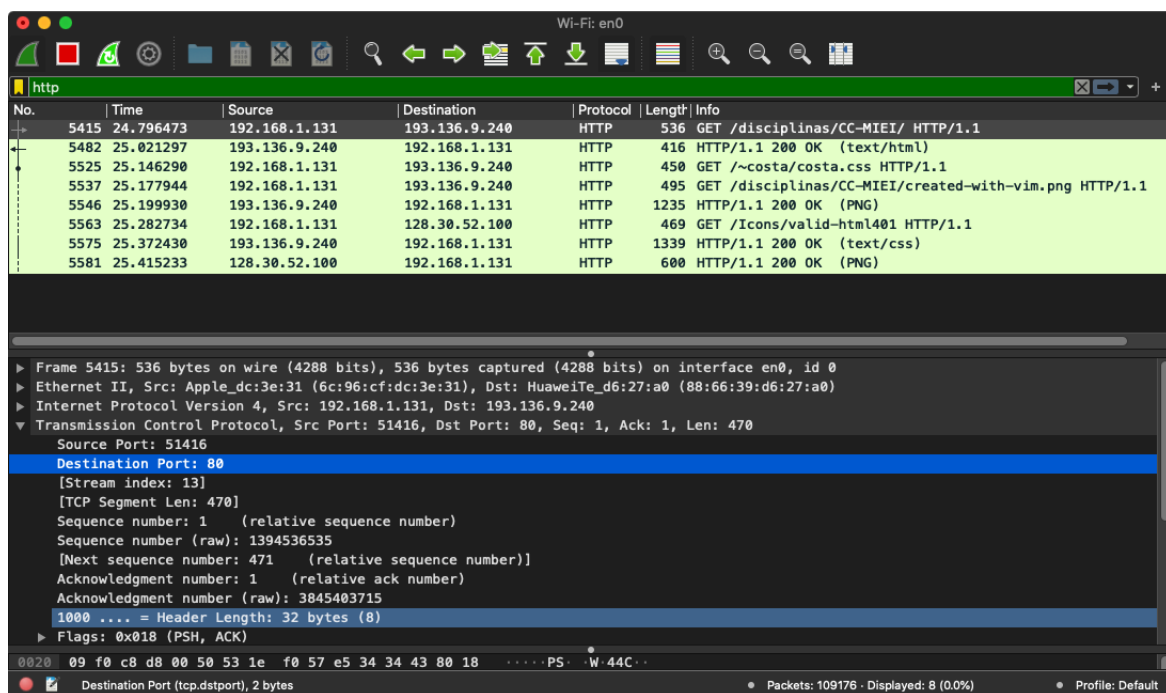


Figure 6: http.

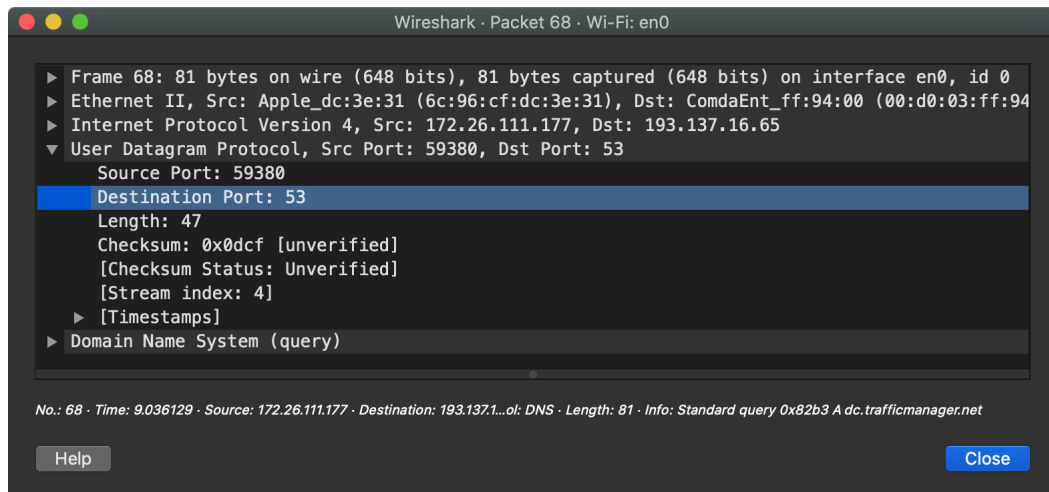


Figure 7: nslookup.

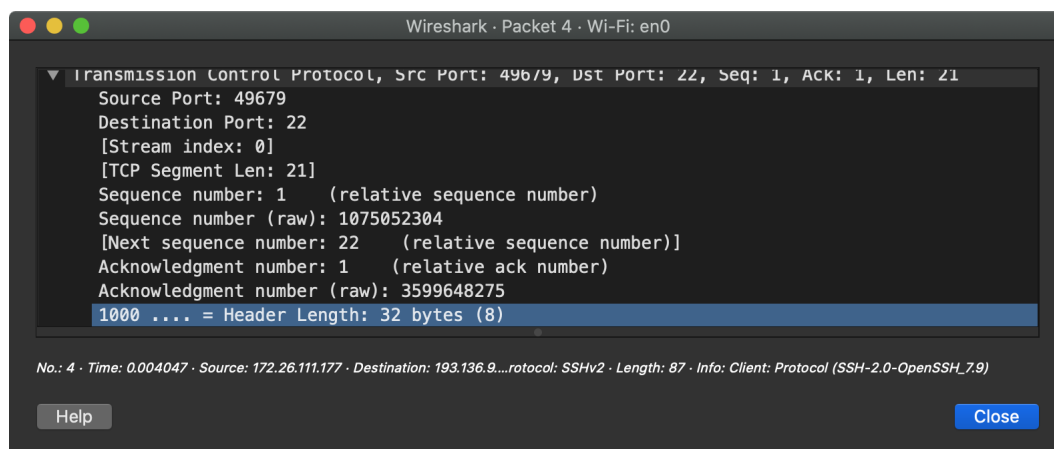


Figure 8: ssh.

Problem 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.

Solution:

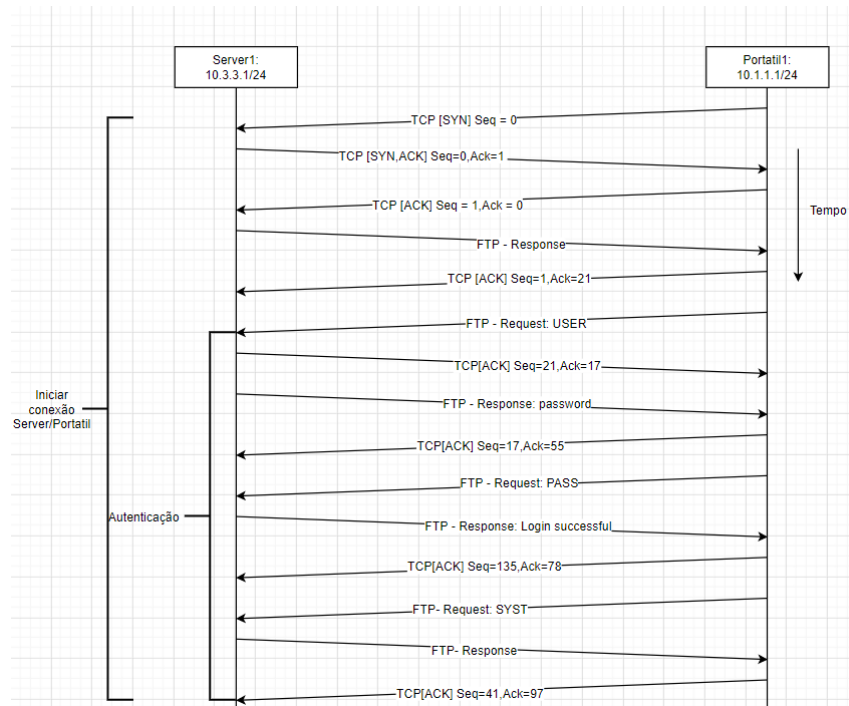


Figure 9: Inicio da conexão FTP.

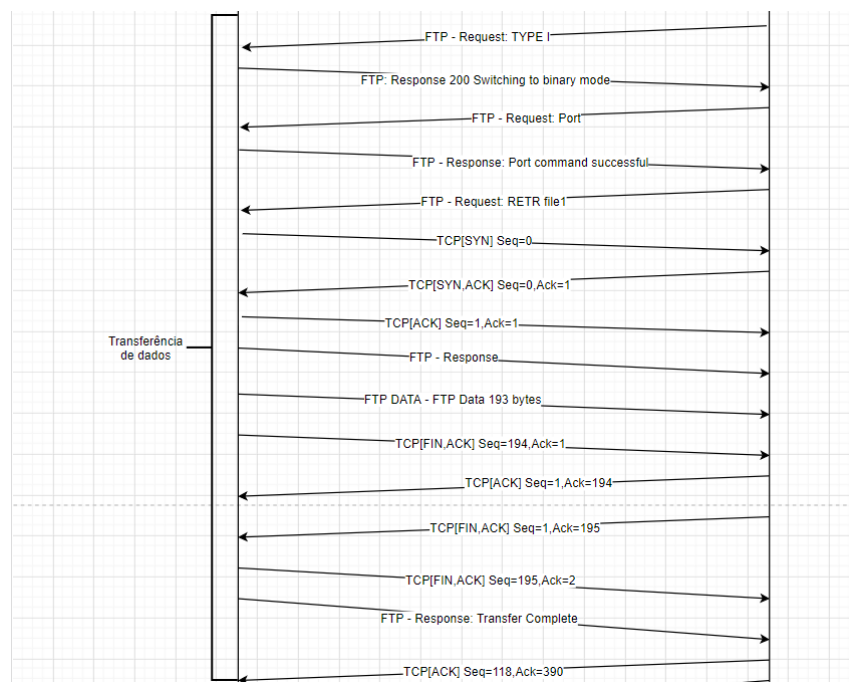


Figure 10: Transferência de dados FTP.

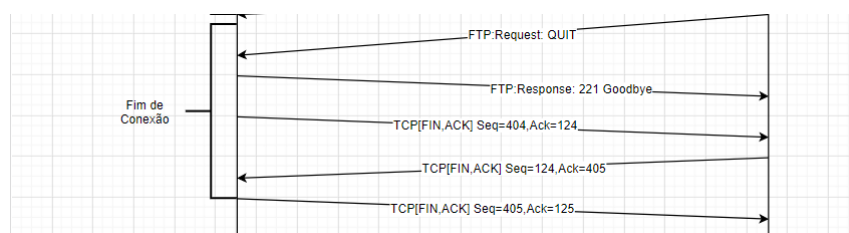


Figure 11: Fim da conexão FTP.

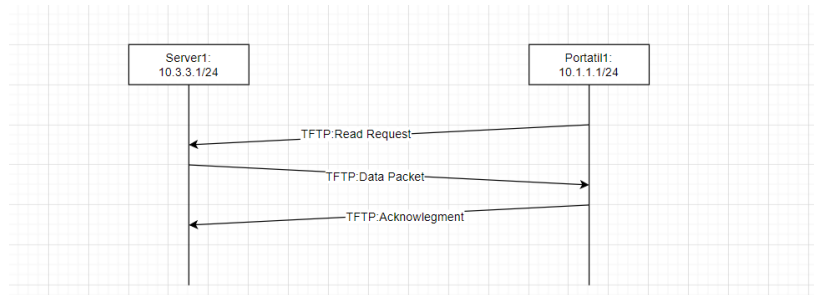


Figure 12: Transferência TFTP.

Problem 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;

Solution:

HTTP

- (i) TCP
- (ii) Destina-se a transferir arquivos criados com hypertext; significando uma página da Web, que são muito mais pequenos do que aqueles que pretenderíamos fazer download. Em relação ao FTP, é mais lento a transferir grandes ficheiros.
- (iii) O HTTP foi implementado para ser simples.
- (iv) Não é encriptada. Todos os navegadores e servidores do mundo falam HTTP, ou seja, um possível hacker conseguiria ler tudo o que se está a passar num browser, incluindo usernames e passwords escritas.

FTP

- (i) TCP
- (ii) Mais rápido para transferir ficheiros grandes que não usam hypertext e não exige que primeiro se renderize uma página da Web, ao contrário de HTTP
- (iii) Como não implementa nenhuma segurança adicional, é por isso uma aplicação de baixa complexidade.
- (iv) Não é um protocolo seguro, pois os dados transferidos não são encriptados nem é necessário autenticação.

SFTP

- (i) TCP
- (ii) Mais lento do que FTP pois usa SSH para a autenticação diminuindo a eficiência de transmissão.
- (iii) Apresenta maior complexidade devido ao uso do protocolo SSH.
- (iv) Adiciona uma camada de segurança em relação ao FTP. Os dados são encriptados usando uma shell segura (SSH), autenticando o user e o server.

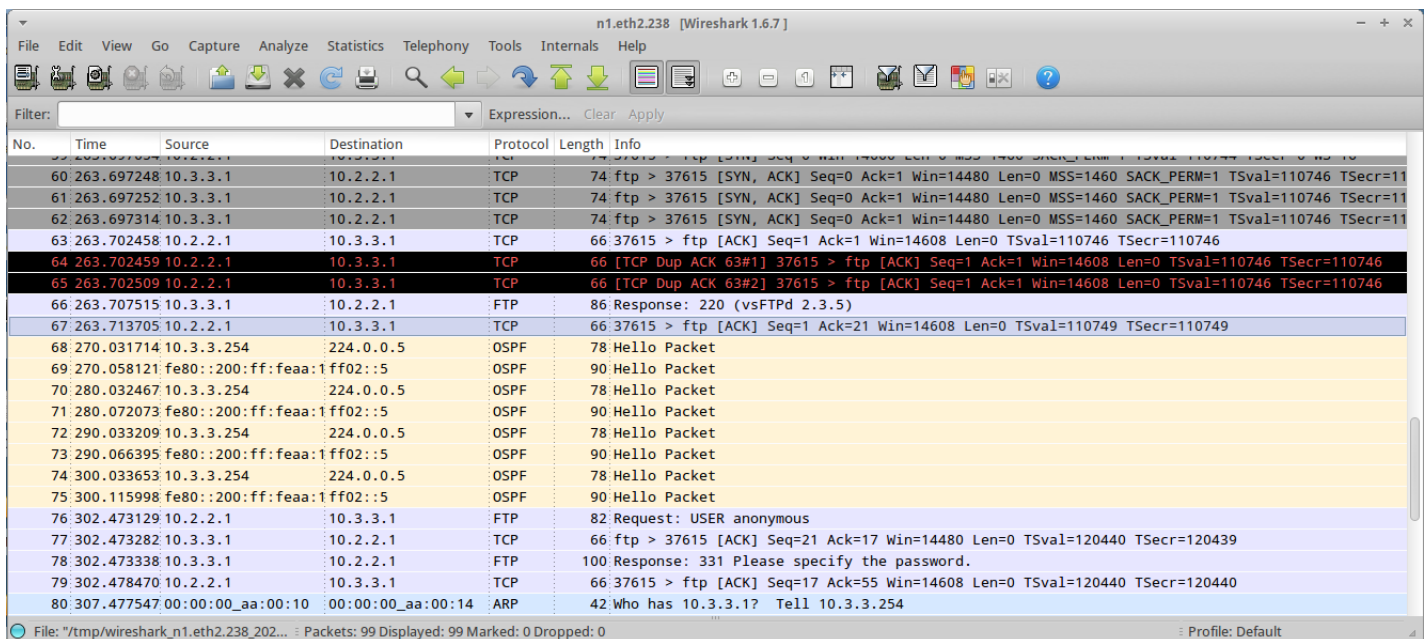
TFTP

- (i) UDP
- (ii) Como usa protocolo UDP, à primeira vista, TFTP pode parecer mais eficiente que as outras aplicações de transferência de ficheiros. No entanto, em caso de erros, UDP não tem os mesmos mecanismos de controlo e correção de erros que TCP tem, tornando, desta forma, a transferência menos eficiente.
- (iii) Assim como a FTP não implementa nenhuma segurança adicional, sendo por isso de baixa complexidade.
- (iv) Protocolo de transmissão mais simples que não usa Internet para transferir ficheiros, mas sim dentro de uma rede local. Tal como o FTP, não usa encriptação e autenticação, ou seja, não fornece segurança durante a transferência.

Problem 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).

Solution:



The image shows a Wireshark packet capture window titled "n1.eth2.238 [Wireshark 1.6.7]". The packet list table is as follows:

No.	Time	Source	Destination	Protocol	Length	Info
60	263.697248	10.3.3.1	10.2.2.1	TCP	74	ftp > 37615 [SYN, ACK] Seq=0 Ack=1 Win=14480 Len=0 MSS=1460 SACK_PERM=1 TSval=110746 TSecr=110746
61	263.697252	10.3.3.1	10.2.2.1	TCP	74	ftp > 37615 [SYN, ACK] Seq=0 Ack=1 Win=14480 Len=0 MSS=1460 SACK_PERM=1 TSval=110746 TSecr=110746
62	263.697314	10.3.3.1	10.2.2.1	TCP	74	ftp > 37615 [SYN, ACK] Seq=0 Ack=1 Win=14480 Len=0 MSS=1460 SACK_PERM=1 TSval=110746 TSecr=110746
63	263.702458	10.2.2.1	10.3.3.1	TCP	66	37615 > ftp [ACK] Seq=1 Ack=1 Win=14608 Len=0 TSval=110746 TSecr=110746
64	263.702459	10.2.2.1	10.3.3.1	TCP	66	[TCP Dup ACK 63#1] 37615 > ftp [ACK] Seq=1 Ack=1 Win=14608 Len=0 TSval=110746 TSecr=110746
65	263.702509	10.2.2.1	10.3.3.1	TCP	66	[TCP Dup ACK 63#2] 37615 > ftp [ACK] Seq=1 Ack=1 Win=14608 Len=0 TSval=110746 TSecr=110746
66	263.707515	10.3.3.1	10.2.2.1	FTP	86	Response: 220 (vsFTPd 2.3.5)
67	263.713705	10.2.2.1	10.3.3.1	TCP	66	37615 > ftp [ACK] Seq=1 Ack=21 Win=14608 Len=0 TSval=110749 TSecr=110749
68	270.031714	10.3.3.254	224.0.0.5	OSPF	78	Hello Packet
69	270.058121	fe80::200:ff:feaa:1ff02::5	224.0.0.5	OSPF	90	Hello Packet
70	280.032467	10.3.3.254	224.0.0.5	OSPF	78	Hello Packet
71	280.072073	fe80::200:ff:feaa:1ff02::5	224.0.0.5	OSPF	90	Hello Packet
72	290.033209	10.3.3.254	224.0.0.5	OSPF	78	Hello Packet
73	290.066395	fe80::200:ff:feaa:1ff02::5	224.0.0.5	OSPF	90	Hello Packet
74	300.033653	10.3.3.254	224.0.0.5	OSPF	78	Hello Packet
75	300.115998	fe80::200:ff:feaa:1ff02::5	224.0.0.5	OSPF	90	Hello Packet
76	302.473129	10.2.2.1	10.3.3.1	FTP	82	Request: USER anonymous
77	302.473282	10.3.3.1	10.2.2.1	TCP	66	ftp > 37615 [ACK] Seq=21 Ack=17 Win=14480 Len=0 TSval=120440 TSecr=120439
78	302.473338	10.3.3.1	10.2.2.1	FTP	100	Response: 331 Please specify the password.
79	302.478470	10.2.2.1	10.3.3.1	TCP	66	37615 > ftp [ACK] Seq=17 Ack=55 Win=14608 Len=0 TSval=120440 TSecr=120440
80	307.477547	00:00:00_aa:00:10	00:00:00_aa:00:14	ARP	42	Who has 10.3.3.1? Tell 10.3.3.254

At the bottom of the window, the status bar shows: "File: /tmp/wireshark_n1.eth2.238_202... : Packets: 99 Displayed: 99 Marked: 0 Dropped: 0" and "Profile: Default".

Figure 13: Duplicação de pacotes

The image shows two terminal windows side-by-side, comparing ping results. The left window is titled 'root@Portatil1: /tmp/pycore.42825/Portatil1.conf' and shows a successful ping to 10.3.3.1 with 7 packets transmitted, 7 received, and 0% packet loss. The right window is titled 'root@Alfa: /tmp/pycore.42825/Alfa.conf' and shows a failed ping to 10.3.3.1 with 8 packets transmitted, 6 received, and 25% packet loss.

```
root@Portatil1: /tmp/pycore.42825/Portatil1.conf
--2020-03-04 14:51:55-- http://10.3.3.1/file1
A conectar 10.3.3.1:80... conectado.
Pedido HTTP enviado, a aguardar resposta... 200 Ok
Tamanho: 193 [text/plain]
Saving to: 'file1.2'

100%[=====] 193      --.-K/s   em 0s

2020-03-04 14:51:55 (43.9 MB/s) - 'file1.2' saved [193/193]

root@Portatil1: /tmp/pycore.42825/Portatil1.conf# ping 10.3.3.1
PING 10.3.3.1 (10.3.3.1) 56(84) bytes of data:
64 bytes from 10.3.3.1: icmp_req=1 ttl=61 time=0.444 ms
64 bytes from 10.3.3.1: icmp_req=2 ttl=61 time=0.393 ms
64 bytes from 10.3.3.1: icmp_req=3 ttl=61 time=0.386 ms
64 bytes from 10.3.3.1: icmp_req=4 ttl=61 time=0.462 ms
64 bytes from 10.3.3.1: icmp_req=5 ttl=61 time=0.343 ms
64 bytes from 10.3.3.1: icmp_req=6 ttl=61 time=0.411 ms
64 bytes from 10.3.3.1: icmp_req=7 ttl=61 time=0.427 ms
^C
--- 10.3.3.1 ping statistics ---
7 packets transmitted, 7 received, 0% packet loss, time 5997ms
rtt min/avg/max/mdev = 0.343/0.409/0.462/0.041 ms
root@Portatil1: /tmp/pycore.42825/Portatil1.conf#

root@Alfa: /tmp/pycore.42825/Alfa.conf
root@Alfa: /tmp/pycore.42825/Alfa.conf# wget http://10.3.3.1/file1
--2020-03-04 14:52:00-- http://10.3.3.1/file1
A conectar 10.3.3.1:80... conectado.
Pedido HTTP enviado, a aguardar resposta... 200 Ok
Tamanho: 193 [text/plain]
Saving to: 'file1.2'

100%[=====] 193      --.-K/s   em 0s

2020-03-04 14:52:00 (42.1 MB/s) - 'file1.2' saved [193/193]

root@Alfa: /tmp/pycore.42825/Alfa.conf# ping 10.3.3.1
PING 10.3.3.1 (10.3.3.1) 56(84) bytes of data:
64 bytes from 10.3.3.1: icmp_req=2 ttl=61 time=5.37 ms
64 bytes from 10.3.3.1: icmp_req=3 ttl=61 time=6.85 ms
64 bytes from 10.3.3.1: icmp_req=4 ttl=61 time=7.77 ms
64 bytes from 10.3.3.1: icmp_req=6 ttl=61 time=5.41 ms
64 bytes from 10.3.3.1: icmp_req=7 ttl=61 time=6.44 ms
64 bytes from 10.3.3.1: icmp_req=8 ttl=61 time=5.28 ms
^C
--- 10.3.3.1 ping statistics ---
8 packets transmitted, 6 received, 25% packet loss, time 7017ms
rtt min/avg/max/mdev = 5.287/6.191/7.776/0.926 ms
root@Alfa: /tmp/pycore.42825/Alfa.conf#
```

Figure 14: Comparação de pings.

No desenvolvimento de uma aplicação fiável temos de ter saber qual o melhor protocolo de transporte que se enquadra no problema em questão.

TCP, que costuma ser a principal opção escolhida, trata-se de um protocolo de transporte fiável que garante que todos os pacotes são enviados e sem erros. Para cada pacote recebido é sempre enviado pelo menos um ACK em como esse pacote foi recebido com sucesso. É um processo complexo e exige que sejam trocados muitos pacotes de controlo entre ambas as partes. Numa rede de menor qualidade, são perdidos e corrompidos imensos factores causando mais sobrecarga na rede e atraso na app.

As alternativas não fiáveis, como o UDP, serão uma opção se a aplicação em desenvolvimento se garantir que os dados são recebidos corretamente. UDP, ao contrário de TCP, não sobrecarrega a rede com ACK's e com reenvios permitindo obter-se uma menor latência de ligação.

Problem 5

Conclusão

Solution:

Este trabalho permitiu entrar em contacto com protocolos de transporte e de aplicação de uma forma mais prática, o que permitiu consolidar melhor o objetivo destes. Permitiu também observar (através do Wireshark) como o servidor e o portátil reagem (respondiam) a packets com certos protocolos.

Assim foi possível aplicar conhecimentos prévios de outras cadeiras (lêia-se Redes de Computadores) neste trabalho de modo a aprofundar o nosso saber geral de redes.