
TP1: Protocolos da Camada de Transporte

TRABALHO REALIZADO POR:

BEATRIZ RIBEIRO MONTEIRO
GONÇALO MARTINS DOS SANTOS
JOÃO AFONSO ALVIM OLIVEIRA DIAS DE ALMEIDA



A95437
Beatriz Monteiro



A95354
Gonçalo Santos



A95191
João Alvim

PL2 GRUPO 1
DATA DE ENTREGA: 12-01-2022
COMUNICAÇÕES POR COMPUTADOR 22/23
UNIVERSIDADE DO MINHO

Índice

1	Parte B: Questões	1
1.1	Pergunta 1	1
1.2	Pergunta 2	2
1.3	Pergunta 3	3
1.4	Pergunta 4	4
1.4.1	Identificação da camada de transporte	4
1.4.2	Eficiência	5
1.4.3	Complexidade	5
1.4.4	Segurança	5
1.5	Pergunta 5	6
1.5.1	<i>Ping</i>	7
1.5.2	<i>Traceroute</i>	7
1.5.3	<i>Telnet</i>	8
1.5.4	<i>FTP</i>	8
1.5.5	<i>Tftp</i>	9
1.5.6	<i>HTTP/Wget</i>	9
1.5.7	<i>Nslookup</i>	10
1.5.8	<i>Ssh</i>	10
2	Conclusão	11

1 Parte B: Questões

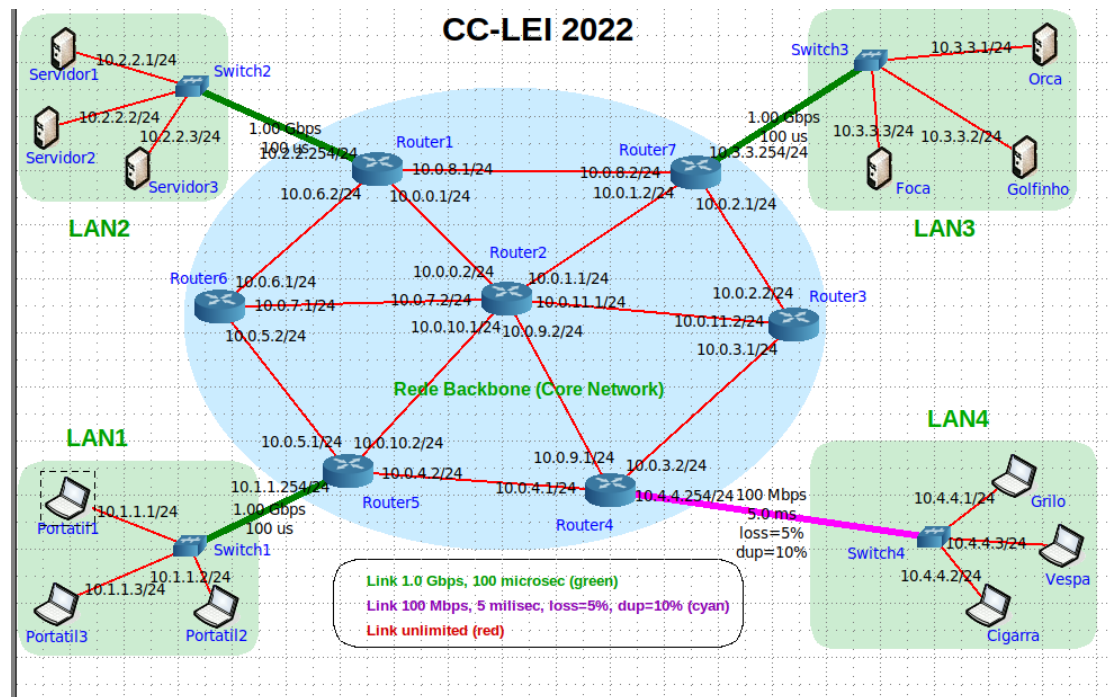


Figure 1: Topologia utilizada

1.1 Pergunta 1

De que forma as perdas e duplicações de pacotes afetaram o desempenho das aplicações? Que camada lidou com esses problemas: transporte ou aplicação? Responda com base nas experiências feitas e nos resultados observados.

```
WARNING: terminal is not fully functional
file-ping-output (press RETURN)PING 10.2.2.1 (10.2.2.1) 56(84) bytes of data.
64 bytes from 10.2.2.1: icmp_seq=1 ttl=61 time=0.524 ms
64 bytes from 10.2.2.1: icmp_seq=2 ttl=61 time=0.411 ms
64 bytes from 10.2.2.1: icmp_seq=3 ttl=61 time=0.364 ms
64 bytes from 10.2.2.1: icmp_seq=4 ttl=61 time=0.375 ms
64 bytes from 10.2.2.1: icmp_seq=5 ttl=61 time=0.375 ms
64 bytes from 10.2.2.1: icmp_seq=6 ttl=61 time=0.363 ms
64 bytes from 10.2.2.1: icmp_seq=7 ttl=61 time=0.363 ms
64 bytes from 10.2.2.1: icmp_seq=8 ttl=61 time=0.359 ms
64 bytes from 10.2.2.1: icmp_seq=9 ttl=61 time=0.492 ms
64 bytes from 10.2.2.1: icmp_seq=10 ttl=61 time=0.366 ms
64 bytes from 10.2.2.1: icmp_seq=11 ttl=61 time=0.354 ms
64 bytes from 10.2.2.1: icmp_seq=12 ttl=61 time=0.372 ms
64 bytes from 10.2.2.1: icmp_seq=13 ttl=61 time=0.366 ms
64 bytes from 10.2.2.1: icmp_seq=14 ttl=61 time=0.428 ms
64 bytes from 10.2.2.1: icmp_seq=15 ttl=61 time=0.449 ms
64 bytes from 10.2.2.1: icmp_seq=16 ttl=61 time=0.414 ms
64 bytes from 10.2.2.1: icmp_seq=17 ttl=61 time=0.356 ms
64 bytes from 10.2.2.1: icmp_seq=18 ttl=61 time=0.397 ms
64 bytes from 10.2.2.1: icmp_seq=19 ttl=61 time=0.376 ms
64 bytes from 10.2.2.1: icmp_seq=20 ttl=61 time=0.377 ms

--- 10.2.2.1 ping statistics ---
20 packets transmitted, 20 received, 0% packet loss, time 19453ms
rtt min/avg/max/mdev = 0.354/0.394/0.524/0.045 ms
```

Figure 2: Ping Portatil1 -> Servidor1

```

PING 10.2.2.1 (10.2.2.1) 56(84) bytes of data:
64 bytes from 10.2.2.1: icmp_seq=1 ttl=61 time=5.90 ms
64 bytes from 10.2.2.1: icmp_seq=1 ttl=61 time=5.91 ms (DUP!)
64 bytes from 10.2.2.1: icmp_seq=2 ttl=61 time=6.12 ms
64 bytes from 10.2.2.1: icmp_seq=2 ttl=61 time=6.86 ms (DUP!)
64 bytes from 10.2.2.1: icmp_seq=3 ttl=61 time=6.36 ms
64 bytes from 10.2.2.1: icmp_seq=4 ttl=61 time=6.06 ms
64 bytes from 10.2.2.1: icmp_seq=5 ttl=61 time=5.81 ms
64 bytes from 10.2.2.1: icmp_seq=6 ttl=61 time=5.63 ms
64 bytes from 10.2.2.1: icmp_seq=7 ttl=61 time=5.67 ms
64 bytes from 10.2.2.1: icmp_seq=8 ttl=61 time=5.30 ms
64 bytes from 10.2.2.1: icmp_seq=10 ttl=61 time=5.30 ms
64 bytes from 10.2.2.1: icmp_seq=11 ttl=61 time=5.57 ms
64 bytes from 10.2.2.1: icmp_seq=12 ttl=61 time=5.37 ms
64 bytes from 10.2.2.1: icmp_seq=12 ttl=61 time=5.37 ms (DUP!)
64 bytes from 10.2.2.1: icmp_seq=13 ttl=61 time=6.02 ms
64 bytes from 10.2.2.1: icmp_seq=13 ttl=61 time=6.03 ms (DUP!)
64 bytes from 10.2.2.1: icmp_seq=14 ttl=61 time=5.63 ms
64 bytes from 10.2.2.1: icmp_seq=14 ttl=61 time=6.40 ms (DUP!)
64 bytes from 10.2.2.1: icmp_seq=15 ttl=61 time=5.88 ms
64 bytes from 10.2.2.1: icmp_seq=16 ttl=61 time=5.99 ms
64 bytes from 10.2.2.1: icmp_seq=17 ttl=61 time=5.25 ms
64 bytes from 10.2.2.1: icmp_seq=18 ttl=61 time=6.12 ms
64 bytes from 10.2.2.1: icmp_seq=19 ttl=61 time=5.30 ms
64 bytes from 10.2.2.1: icmp_seq=20 ttl=61 time=5.31 ms

--- 10.2.2.1 ping statistics ---
20 packets transmitted, 19 received, +5 duplicates, 5% packet loss, time 19048ms
rtt min/avg/max/mdev = 5.246/5.797/6.859/0.409 ms

```

Figure 3: *Ping* Grilo -> Servidor1

R: Nos dois *pings* efetuados entre o **Portátil1** e o **Servidor1** e **Grilo** e o **Servidor1** apenas o segundo causou problemas.

Neste segundo *ping*, houve 5 pacotes duplicados que aparecem como (DUP!), estes pacotes são simplesmente ignorados pelas aplicações.

Os pacotes que foram perdidos, não chegam ao destino pelo que se pode observar na imagem que houve uma perda de 5% dos pacotes.

Caso se use TCP, a camada que trata dos problemas é a camada de transporte. Caso se use UDP, a camada que trata dos problemas é a camada de aplicação. Isto porque o protocolo TCP tem mecanismos de controlo de erros enquanto que UDP não tem.

1.2 Pergunta 2

Obtenha a partir do *Wireshark*, ou desenhe manualmente, um diagrama temporal para a transferência do ficheiro *file1* por FTP realizada em A.3. Foque-se apenas na transferência de dados [ftp-data] e não na conexão de controlo (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 tanto nos dados como nas confirmações.

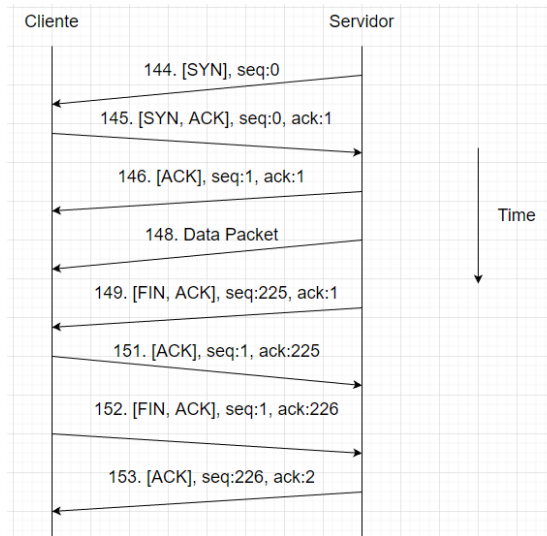


Figure 4: Diagrama Temporal FTP

R: O protocolo FTP (*file transfer protocol*) é um protocolo de transferência de dados que pode ser executado em 2 modos: ativo ou passivo, dependendo de quem inicia a conexão de dados entre o servidor e o cliente.

O cliente estabelece uma conexão TCP para a porta 21 do servidor, sendo a primeira conexão a ser criada e a segunda é com a porta 20 do servidor.

Todos os *browsers* são configurados, por *default*, para funcionar no modo passivo quando utilizados por clientes.

Entre os pacotes 144 a 146 estabelece-se a conexão entre o cliente e o servidor. O pacote 148 representa a fase de transferência de dados. Do pacote 149 até ao final (153) realiza-se a conclusão da conexão.

Os tipos de segmentos trocados foram segmentos TCP e um pacotes de dados (FTP-Data).

144	119.492691941	10.2.2.1	10.1.1.1	TCP	74	20 → 40861	[SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 T...
145	119.492864120	10.1.1.1	10.2.2.1	TCP	74	40861 → 20	[SYN, ACK] Seq=0 Ack=1 Win=65160 Len=0 MSS=1460 SA...
146	119.492997787	10.2.2.1	10.1.1.1	TCP	66	20 → 40861	[ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=430564650 ...
147	119.493037964	10.2.2.1	10.1.1.1	FTP	130		Response: 150 Opening BINARY mode data connection for file1 (...)
148	119.493327205	10.2.2.1	10.1.1.1	FTP-DA...	290		FTP Data: 224 bytes (PORT) (RETR file1)
149	119.493329962	10.2.2.1	10.1.1.1	TCP	66	20 → 40861	[FIN, ACK] Seq=225 Ack=1 Win=64256 Len=0 TSval=430...
150	119.493348933	10.1.1.1	10.2.2.1	TCP	66	57346 → 21	[ACK] Seq=130 Ack=391 Win=64256 Len=0 TSval=200659...
151	119.493518745	10.1.1.1	10.2.2.1	TCP	66	40861 → 20	[ACK] Seq=1 Ack=225 Win=65024 Len=0 TSval=20065990...
152	119.493822976	10.1.1.1	10.2.2.1	TCP	66	40861 → 20	[FIN, ACK] Seq=1 Ack=226 Win=65024 Len=0 TSval=200...
153	119.493952000	10.2.2.1	10.1.1.1	TCP	66	20 → 40861	[ACK] Seq=226 Ack=2 Win=64256 Len=0 TSval=43056465...
154	119.494009183	10.2.2.1	10.1.1.1	FTP	90		Response: 226 Transfer complete.

Figure 5: Segmentos correspondentes à conexão dos dados

1.3 Pergunta 3

Obtenha a partir do *Wireshark*, ou desenhe manualmente, um diagrama temporal para a transferência do ficheiro *file1* por TFTP realizada em A.4. 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 tanto nos dados como nas confirmações.

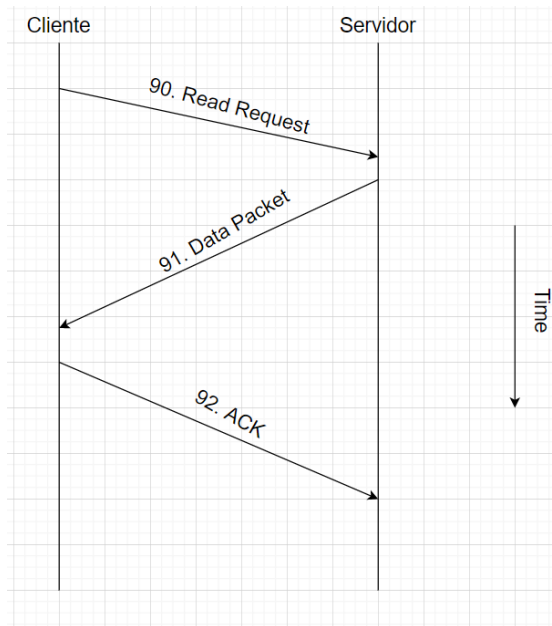


Figure 6: Diagrama Temporal TFTP

R: Inicialmente o cliente envia um *Read Request* ao servidor. Em seguida o servidor envia um *Data Packet* com os dados que pretende enviar, no caso destes serem maiores do que 512 *bytes*, são divididos em pacotes diferentes com este limite máximo. O cliente, assim que recebe os dados envia uma trama ACK para confirmar o sucesso da receção dos dados.

Como o TFTP usa UDP como protocolo de transporte e este protocolo não tem fase de estabelecimento nem de fim de conexão, não é possível identificar estas fases. Ainda não é possível identificar os números de sequência usados nos dados e nas confirmações porque pacotes UDP não têm no seu cabeçalho um número de sequência.

90	134.501720856	10.1.1.1	10.2.2.1	TFTP	56 Read Request, File: file1, Transfer type: octet
91	134.502864602	10.2.2.1	10.1.1.1	TFTP	270 Data Packet, Block: 1 (last)
92	134.503178425	10.1.1.1	10.2.2.1	TFTP	46 Acknowledgement, Block: 1

Figure 7: Segmentos referentes à transferência de dados

1.4 Pergunta 4

Compare sucintamente as quatro aplicações de transferência de ficheiros que usou, tendo em consideração os seguintes aspetos: (i) identificação da camada de transporte; (ii) eficiência; (iii) complexidade; (iv) segurança.

R: As quatro aplicações de transferência de ficheiros que utilizamos na elaboração deste trabalho foram SFTP(*SSH File Transfer Protocol*), FTP(*File Transfer Protocol*), TFTP(*Trivial File Transfer Protocol*) e HTTP(*HyperText Transfer Protocol*)

1.4.1 Identificação da camada de transporte

- **SFTP:** Utiliza o protocolo TCP;
- **FTP:** Utiliza o protocolo TCP - utiliza duas conexões TCP;
- **TFTP:** Utiliza o protocolo UDP - utiliza uma conexão (*stop and wait*);
- **HTTP:** Utiliza o protocolo TCP;

1.4.2 Eficiência

- **SFTP:** Semelhante à do FTP, no entanto é um pouco mais eficiente e segura porque os dados se encontram encriptados (usa *ssh* para o fazer).
- **FTP:** Bastante fiável na transferência de dados tendo um custo na sua eficiência. O uso do protocolo TCP garante que o segmento vai ser transmitido, no entanto perde eficiência visto que tem que esperar pelos *acknowledges* para continuar a transmissão. Contudo o FTP não utiliza encriptação dos dados permitindo que ocorram possíveis ataques.
- **TFTP:** O protocolo TFTP utiliza o protocolo UDP. O protocolo UDP não usa *acknowledge*, o que faz com que não seja possível confirmar se o pacote foi entregue com sucesso ou não levando à retransmissão dos pacotes várias vezes. No entanto, apesar de menos viável, se a transmissão for bem sucedida, é mais rápido do que o FTP. Ou seja, em geral o TFTP é mais rápido que o FTP mas o FTP é bem mais seguro.
- **HTTP:** O HTTP permite vários HTTP *requests* sejam enviados em simultâneo numa única ligação TCP sem necessidade de esperar pelas respetivas respostas, devido ao *pipelining*. Logo é bastante eficiente.

1.4.3 Complexidade

Em geral, aplicações de transferência de ficheiros que usem TCP são mais complexas que as que usam UDP pela própria natureza dos protocolos de transporte.

- **SFTP:** Este protocolo é considerado complexo uma vez que permite aceder, transferir e gerir dados.
- **FTP:** Como já foi referido anteriormente este é um processo bastante fiável tornando-se, assim, bastante complexo. Esta elevada complexidade deve-se ainda à capacidade de suportar vários pedidos para transferir dados em paralelo, cada pedido estabelece uma conexão de dados, desta forma é necessário existirem diferentes velocidades de transferência.
- **TFTP:** O TFTP é uma versão simplificada do FTP, para além de apresentar menos funcionalidades é mais leve e simples e baseia-se em UDP, sendo, deste modo, menos complexo quando comparado com o FTP.
- **HTTP:** O protocolo HTTP é considerado complexo, uma vez que é utilizado em sistemas distribuídos e permite escalabilidade e desacoplamento de sistemas, entre outras funcionalidades.

1.4.4 Segurança

- **SFTP:** O SFTP é o protocolo mais seguro dos 4 estamos a analisar. É capaz de fornecer um canal seguro sobre uma rede insegura numa arquitetura cliente-servidor. Este protocolo para além de utilizar o TCP possui ainda uma arquitetura por camadas que conferem encriptação e proteção da integridade dos dados.
- **FTP:** Apesar de possuir autenticação não tem encriptação dos dados, permitindo, desta forma, a qualquer pessoa intercetar os pacotes na rede e obter informações dos arquivos e credenciais.

- **TFTP:** Não fornece qualquer tipo de autenticação nem de proteção de dados, podemos concluir que tem um nível de segurança bastante baixo.
- **HTTP:** Tal como o FTP, recorre a autenticação mas não tem encriptação dos dados, tornando assim possível o roubo de informação confidencial ou até mesmo a alteração da informação que está a ser transferida.

1.5 Pergunta 5

Com base no trabalho realizado, construa uma tabela informativa identificando, para cada aplicação executada (*ping*, *traceroute*, *telnet*, *ftp*, *tftp*, *wget/lynx*, *nslookup*, *ssh*, etc.), qual o protocolo de aplicação, o protocolo de transporte, a porta de atendimento e o *overhead* de transporte.

R:

Comando Usado	Protocolo de Aplicação	Protocolo de Transporte	Porta de atendimento	<i>Overhead</i> transportado em <i>bytes</i>
<i>Ping</i>	Ping	-	-	-
<i>Traceroute</i>	Traceroute	UDP	33446	20
<i>Telnet</i>	TELNET	TCP	23	20
<i>Ftp</i>	FTP	TCP	21	20
<i>Tftp</i>	TFTP	UDP	69	8
<i>Wget</i>	HTTP	TCP	80	20
<i>Nslookup</i>	DNS	UDP	53	8
<i>Ssh</i>	SSHv2	TCP	22	20

Table 1: Tabela Informativa - Pergunta 5

O *overhead* de transporte é o tamanho do cabeçalho de transporte sendo 20 *bytes* para o protocolo TCP, e 8 *bytes* para o protocolo UDP.

Para complementar a tabela acima, seguem-se imagens relativas ao tráfego capturado no *Wireshark*.

1.5.1 Ping

No.	Time	Source	Destination	Protocol	Length	Info
70	179.800380272	193.137.16.65	10.0.2.15	DNS	138	Standard query response 0x729d AAAA www.uminho.pt SOA dns.umi...
71	184.93594162	PcsCompu_06:03:48	RealtekU_12:35:02	ARP	42	Who has 10.0.2.2? Tell 10.0.2.15
72	184.935854546	RealtekU_12:35:02	PcsCompu_06:03:48	ARP	60	10.0.2.2 is at 52:54:00:12:35:02
73	187.897874526	fe80::1521:1260:7f1...	ff02::fb	MDNS	188	Standard query 0x0000 PTR _ftp.tcp.local, "QM" question PTR ...
74	187.897999389	10.0.2.15	224.0.0.251	MDNS	160	Standard query 0x0000 PTR _ftp.tcp.local, "QM" question PTR ...
75	189.977841942	10.0.2.15	193.137.16.65	DNS	84	Standard query 0x14c1 A www.google.pt OPT
76	189.978071786	10.0.2.15	193.137.16.65	DNS	84	Standard query 0x71d8 AAAA www.google.pt OPT
77	190.056534072	193.137.16.65	10.0.2.15	DNS	112	Standard query response 0x71d8 AAAA www.google.pt AAAA 2a00:1...
78	190.056534653	193.137.16.65	10.0.2.15	DNS	100	Standard query response 0x14c1 A www.google.pt A 216.58.215.1...
79	190.057542594	10.0.2.15	216.58.215.131	ICMP	98	Echo (ping) request id=0x0001, seq=1/256, ttl=64 (reply in 8...
80	190.163649174	216.58.215.131	10.0.2.15	ICMP	98	Echo (ping) reply id=0x0001, seq=1/256, ttl=113 (request i...
81	190.164281994	10.0.2.15	193.137.16.65	DNS	98	Standard query 0x1eab PTR 131.215.58.216.in-addr.arpa OPT
82	190.370751501	193.137.16.65	10.0.2.15	DNS	136	Standard query response 0x1eab PTR 131.215.58.216.in-addr.arp...
83	191.059028075	10.0.2.15	216.58.215.131	ICMP	98	Echo (ping) request id=0x0001, seq=2/512, ttl=64 (reply in 8...
84	191.206388959	216.58.215.131	10.0.2.15	ICMP	98	Echo (ping) reply id=0x0001, seq=2/512, ttl=113 (request 1...
85	192.061405149	10.0.2.15	216.58.215.131	ICMP	98	Echo (ping) request id=0x0001, seq=3/768, ttl=64 (reply in 8...
86	192.146706302	216.58.215.131	10.0.2.15	ICMP	98	Echo (ping) reply id=0x0001, seq=3/768, ttl=113 (request 1...
87	193.062708355	10.0.2.15	216.58.215.131	ICMP	98	Echo (ping) request id=0x0001, seq=4/1024, ttl=64 (reply in ...
88	193.190852685	216.58.215.131	10.0.2.15	ICMP	98	Echo (ping) reply id=0x0001, seq=4/1024, ttl=113 (request ...
89	194.064108963	10.0.2.15	216.58.215.131	ICMP	98	Echo (ping) request id=0x0001, seq=5/1280, ttl=64 (reply in ...
90	194.134905961	216.58.215.131	10.0.2.15	ICMP	98	Echo (ping) reply id=0x0001, seq=5/1280, ttl=113 (request ...
91	195.065155057	10.0.2.15	216.58.215.131	ICMP	98	Echo (ping) request id=0x0001, seq=6/1536, ttl=64 (reply in ...
92	195.179601210	216.58.215.131	10.0.2.15	ICMP	98	Echo (ping) reply id=0x0001, seq=6/1536, ttl=113 (request ...
93	196.066753170	10.0.2.15	216.58.215.131	ICMP	98	Echo (ping) request id=0x0001, seq=7/1792, ttl=64 (reply in ...
94	196.219794077	216.58.215.131	10.0.2.15	ICMP	98	Echo (ping) reply id=0x0001, seq=7/1792, ttl=113 (request ...
95	197.067947311	10.0.2.15	216.58.215.131	ICMP	98	Echo (ping) request id=0x0001, seq=8/2048, ttl=64 (reply in ...
Frame 83: 98 bytes on wire (784 bits), 98 bytes captured (784 bits) on interface enp0s3, id 0						
Ethernet II, Src: PcsCompu_06:03:48 (08:00:27:06:03:48), Dst: RealtekU_12:35:02 (52:54:00:12:35:02)						
Internet Protocol Version 4, Src: 10.0.2.15, Dst: 216.58.215.131						
0100 = Version: 4						
.... 0101 = Header Length: 20 bytes (5)						
Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)						
Total Length: 84						
Identification: 0xa9cb (43467)						
Flags: 0x4000, Don't fragment						
Fragment offset: 0						
Time to live: 64						
Protocol: ICMP (1)						
Header checksum: 0xd510 [validation disabled]						
[Header checksum status: Unverified]						
Source: 10.0.2.15						
Destination: 216.58.215.131						
Internet Control Message Protocol						

Figure 8: Captura *Ping*

O *ping* trabalha diretamente com a camada de rede, não tendo, assim, nenhum protocolo aplicacional nem de transporte.

1.5.2 Traceroute

128	218.155526032	10.0.2.15	193.137.196.247	UDP	74	50265 → 33446 Len=32
129	218.155537260	10.0.2.15	193.137.196.247	UDP	74	38338 → 33447 Len=32
130	218.155571716	10.0.2.15	193.137.196.247	UDP	74	48793 → 33448 Len=32
131	218.155584116	10.0.2.15	193.137.196.247	UDP	74	33899 → 33449 Len=32
132	218.155964906	10.0.2.15	193.137.16.65	DNS	92	Standard query 0x8d44 PTR 2.2.0.10.in-addr.arpa OPT
133	218.158176080	193.137.16.65	10.0.2.15	DNS	155	Standard query response 0x8d44 No such name PTR 2.2.0.10.in-a...
134	218.158351286	10.0.2.15	193.137.16.65	DNS	81	Standard query 0x8d44 PTR 2.2.0.10.in-addr.arpa
135	218.160120880	193.137.16.65	10.0.2.15	DNS	144	Standard query response 0x8d44 No such name PTR 2.2.0.10.in-a...
136	218.161294250	10.0.2.15	193.137.196.247	UDP	74	40872 → 33450 Len=32
137	218.161334356	10.0.2.15	193.137.196.247	UDP	74	53850 → 33451 Len=32
138	218.161350252	10.0.2.15	193.137.196.247	UDP	74	46615 → 33452 Len=32
Frame 128: 74 bytes on wire (592 bits), 74 bytes captured (592 bits) on interface enp0s3, id 0						
Ethernet II, Src: PcsCompu_06:03:48 (08:00:27:06:03:48), Dst: RealtekU_12:35:02 (52:54:00:12:35:02)						
Internet Protocol Version 4, Src: 10.0.2.15, Dst: 193.137.196.247						
0100 = Version: 4						
.... 0101 = Header Length: 20 bytes (5)						
Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)						
Total Length: 60						
Identification: 0x4702 (18178)						
Flags: 0x0000						
Fragment offset: 0						
Time to live: 5						
Protocol: UDP (17)						
Header checksum: 0xdcd1 [validation disabled]						
[Header checksum status: Unverified]						
Source: 10.0.2.15						
Destination: 193.137.196.247						
User Datagram Protocol, Src Port: 50265, Dst Port: 33446						
Source Port: 50265						
Destination Port: 33446						
Length: 40						
Checksum: 0x92c9 [unverified]						
[Checksum Status: Unverified]						
[Stream index: 39]						
[Timestamps]						
Data (32 bytes)						

Figure 9: Captura *Traceroute*

1.5.3 Telnet

18	3.051357460	10.0.2.15	64.13.139.230	TELNET	81 Telnet Data ...
19	3.051700554	64.13.139.230	10.0.2.15	TCP	60 23 → 40906 [ACK] Seq=1 Ack=28 Win=65535 Len=0
20	3.262011344	64.13.139.230	10.0.2.15	TELNET	60 Telnet Data ...
21	3.262037506	10.0.2.15	64.13.139.230	TCP	54 40906 → 23 [ACK] Seq=28 Ack=4 Win=64237 Len=0
22	3.498467652	64.13.139.230	10.0.2.15	TELNET	106 Telnet Data ...
23	3.498493474	10.0.2.15	64.13.139.230	TCP	54 40906 → 23 [ACK] Seq=28 Ack=56 Win=64185 Len=0
24	3.498650326	10.0.2.15	64.13.139.230	TELNET	72 Telnet Data ...
25	3.498928954	64.13.139.230	10.0.2.15	TCP	60 23 → 40906 [ACK] Seq=56 Ack=46 Win=65535 Len=0
26	3.695962304	64.13.139.230	10.0.2.15	TELNET	1231 Telnet Data ...
27	3.695989899	10.0.2.15	64.13.139.230	TCP	54 40906 → 23 [ACK] Seq=46 Ack=1233 Win=63558 Len=0
28	3.696251399	10.0.2.15	64.13.139.230	TELNET	80 Telnet Data ...
29	3.696528725	64.13.139.230	10.0.2.15	TCP	60 23 → 40906 [ACK] Seq=1233 Ack=72 Win=65535 Len=0
30	3.976731958	10.0.2.15	79.142.192.130	NTP	90 NTP Version 4, client
31	3.976755538	10.0.2.15	193.136.164.4	NTP	90 NTP Version 4, client
32	3.976786776	10.0.2.15	178.33.203.107	NTP	90 NTP Version 4, client
33	3.976796702	10.0.2.15	88.157.128.22	NTP	90 NTP Version 4, client
34	3.986319501	193.136.164.4	10.0.2.15	NTP	90 NTP Version 4, server
35	3.987673337	88.157.128.22	10.0.2.15	NTP	90 NTP Version 4, server
36	4.028863473	178.33.203.107	10.0.2.15	NTP	90 NTP Version 4, server
37	4.061729000	79.142.192.130	10.0.2.15	NTP	90 NTP Version 4, server
38	4.976915329	10.0.2.15	91.134.158.89	NTP	90 NTP Version 4, client
39	5.030927000	91.134.158.89	10.0.2.15	NTP	90 NTP Version 4, server

▶ Frame 18: 81 bytes on wire (648 bits), 81 bytes captured (648 bits) on interface enp0s3, id 0
 ▶ Ethernet II, Src: PcsCompu_06:03:48 (08:00:27:06:03:48), Dst: RealtekU_12:35:02 (52:54:00:12:35:02)
 ▶ Internet Protocol Version 4, Src: 10.0.2.15, Dst: 64.13.139.230
 0100 = Version: 4
 0101 = Header Length: 20 bytes (5)
 ▶ Differentiated Services Field: 0x10 (DSCP: Unknown, ECN: Not-ECT)
 Total Length: 67
 Identification: 0x823f (33343)
 ▶ Flags: 0x4000, Don't fragment
 Fragment offset: 0
 Time to live: 64
 Protocol: TCP (6)
 Header checksum: 0xe063 [validation disabled]
 [Header checksum status: Unverified]
 Source: 10.0.2.15
 Destination: 64.13.139.230
 ▶ Transmission Control Protocol, Src Port: 40906, Dst Port: 23, Seq: 1, Ack: 1, Len: 27
 ▶ Telnet
 ▶ Do Suppress Go Ahead
 ▶ Will Terminal Type
 ▶ Will Negotiate About Window Size
 ▶ Will Terminal Speed
 ▶ Will Remote Flow Control
 ▶ Will Linemode
 ▶ Will New Environment Option
 Do Status

Figure 10: Captura Telnet

1.5.4 FTP

144	119.492691941	10.2.2.1	10.1.1.1	TCP	74 20 → 40861 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 T...
145	119.492864120	10.1.1.1	10.2.2.1	TCP	74 40861 → 20 [SYN, ACK] Seq=0 Ack=1 Win=65160 Len=0 MSS=1460 SA...
146	119.492997787	10.2.2.1	10.1.1.1	TCP	66 20 → 40861 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=430564650 ...
147	119.493037964	10.2.2.1	10.1.1.1	FTP	130 Response: 150 Opening BINARY mode data connection for file1 (...)
148	119.493327205	10.2.2.1	10.1.1.1	FTP-DA...	290 FTP Data: 224 bytes (PORT) (RETR file1)
149	119.493329962	10.2.2.1	10.1.1.1	TCP	66 20 → 40861 [FIN, ACK] Seq=225 Ack=1 Win=64256 Len=0 TSval=430...
150	119.493348933	10.1.1.1	10.2.2.1	TCP	66 57346 → 21 [ACK] Seq=130 Ack=391 Win=64256 Len=0 TSval=200659...
151	119.493518745	10.1.1.1	10.2.2.1	TCP	66 40861 → 20 [ACK] Seq=1 Ack=225 Win=65024 Len=0 TSval=20065990...
152	119.493822976	10.1.1.1	10.2.2.1	TCP	66 40861 → 20 [FIN, ACK] Seq=1 Ack=226 Win=65024 Len=0 TSval=200...
153	119.493952000	10.2.2.1	10.1.1.1	TCP	66 20 → 40861 [ACK] Seq=226 Ack=2 Win=64256 Len=0 TSval=43056465...
154	119.494009183	10.2.2.1	10.1.1.1	FTP	90 Response: 226 Transfer complete.

Figure 11: Captura FTP

1.5.5 Tftp

90	134.501720856	10.1.1.1	10.2.2.1	TFTP	56 Read Request, File: file1, Transfer type: octet
91	134.502864602	10.2.2.1	10.1.1.1	TFTP	270 Data Packet, Block: 1 (last)
92	134.503178425	10.1.1.1	10.2.2.1	TFTP	46 Acknowledgement, Block: 1
93	136.060315212	10.2.2.254	224.0.0.5	OSPF	78 Hello Packet
94	138.061429353	10.2.2.254	224.0.0.5	OSPF	78 Hello Packet
95	139.625127373	00:00:00_aa:00:10	00:00:00_aa:00:14	ARP	42 Who has 10.2.2.1? Tell 10.2.2.254
96	139.625302738	00:00:00_aa:00:14	00:00:00_aa:00:10	ARP	42 Who has 10.2.2.254? Tell 10.2.2.1
97	139.625309336	00:00:00_aa:00:10	00:00:00_aa:00:14	ARP	42 10.2.2.254 is at 00:00:00_aa:00:10
98	139.625392727	00:00:00_aa:00:14	00:00:00_aa:00:10	ARP	42 10.2.2.1 is at 00:00:00_aa:00:14
99	140.061529387	10.2.2.254	224.0.0.5	OSPF	78 Hello Packet
100	142.031986327	fe80::200:ff:feaa:10	ff02::5	OSPF	90 Hello Packet
101	142.062593874	10.2.2.254	224.0.0.5	OSPF	78 Hello Packet

Frame 90: 56 bytes on wire (448 bits), 56 bytes captured (448 bits) on interface veth1.2.45, id 0
 Ethernet II, Src: 00:00:00_aa:00:10 (00:00:00_aa:00:10), Dst: 00:00:00_aa:00:14 (00:00:00_aa:00:14)
 Internet Protocol Version 4, Src: 10.1.1.1, Dst: 10.2.2.1
 0100 = Version: 4
 0101 = Header Length: 20 bytes (5)
 ▶ Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
 Total Length: 42
 Identification: 0xceb2 (52914)
 ▶ Flags: 0x4000, Don't fragment
 Fragment offset: 0
 Time to live: 61
 Protocol: UDP (17)
 Header checksum: 0x580c [validation disabled]
 [Header checksum status: Unverified]
 Source: 10.1.1.1
 Destination: 10.2.2.1
 User Datagram Protocol, Src Port: 50555, Dst Port: 69
 Trivial File Transfer Protocol

Figure 12: Captura FTP

1.5.6 HTTP/Wget

44	46.268658036	90.130.70.73	10.0.2.15	HTTP	1896 HTTP/1.1 200 OK (text/html)
----	--------------	--------------	-----------	------	----------------------------------

Frame 44: 1896 bytes on wire (15168 bits), 1896 bytes captured (15168 bits) on interface enp0s3, id 0
 Ethernet II, Src: RealtekU_12:35:02 (52:54:00:12:35:02), Dst: PcsCompu_06:03:48 (08:00:27:06:03:48)
 Internet Protocol Version 4, Src: 90.130.70.73, Dst: 10.0.2.15
 0100 = Version: 4
 0101 = Header Length: 20 bytes (5)
 ▶ Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
 Total Length: 1882
 Identification: 0x0976 (2422)
 ▶ Flags: 0x0000
 Fragment offset: 0
 Time to live: 64
 Protocol: TCP (6)
 Header checksum: 0xbd4e [validation disabled]
 [Header checksum status: Unverified]
 Source: 90.130.70.73
 Destination: 10.0.2.15
 ▶ Transmission Control Protocol, Src Port: 80, Dst Port: 46792, Seq: 7101, Ack: 147, Len: 1842
 ▶ [3 Reassembled TCP Segments (8942 bytes): #40(2840), #42(4260), #44(1842)]
 ▶ Hypertext Transfer Protocol
 ▶ Line-based text data: text/html (140 lines)

Figure 13: Captura HTTP

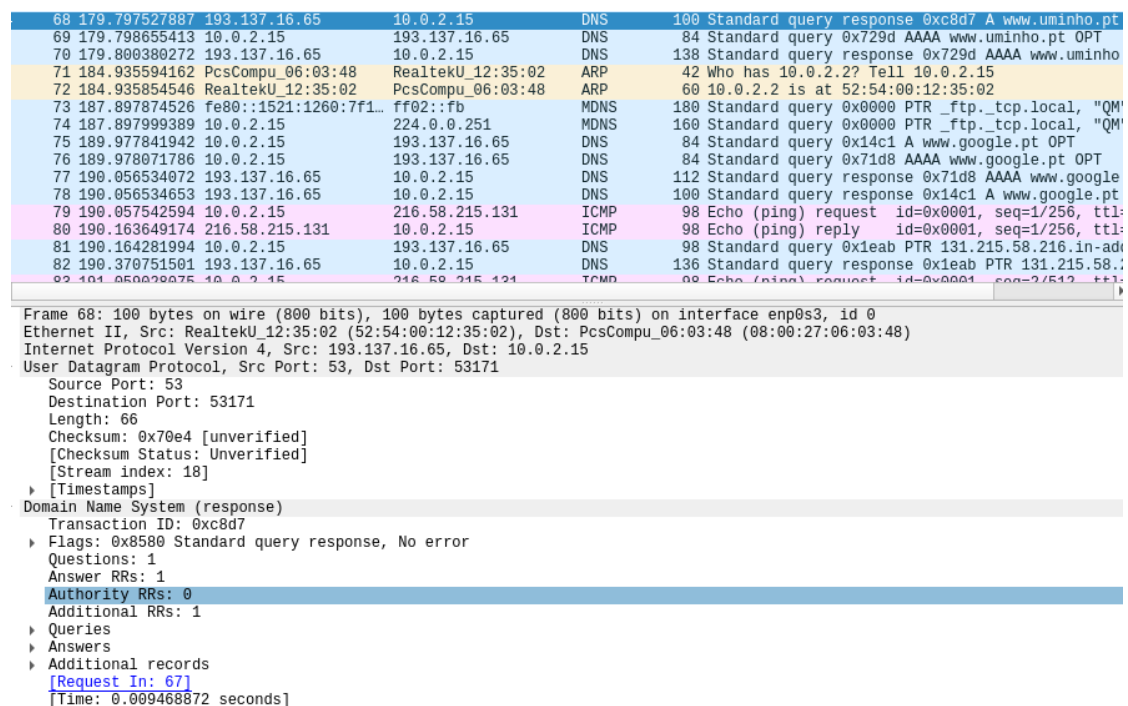


Figure 14: Captura Nslookup

1.5.8 *Ssh*

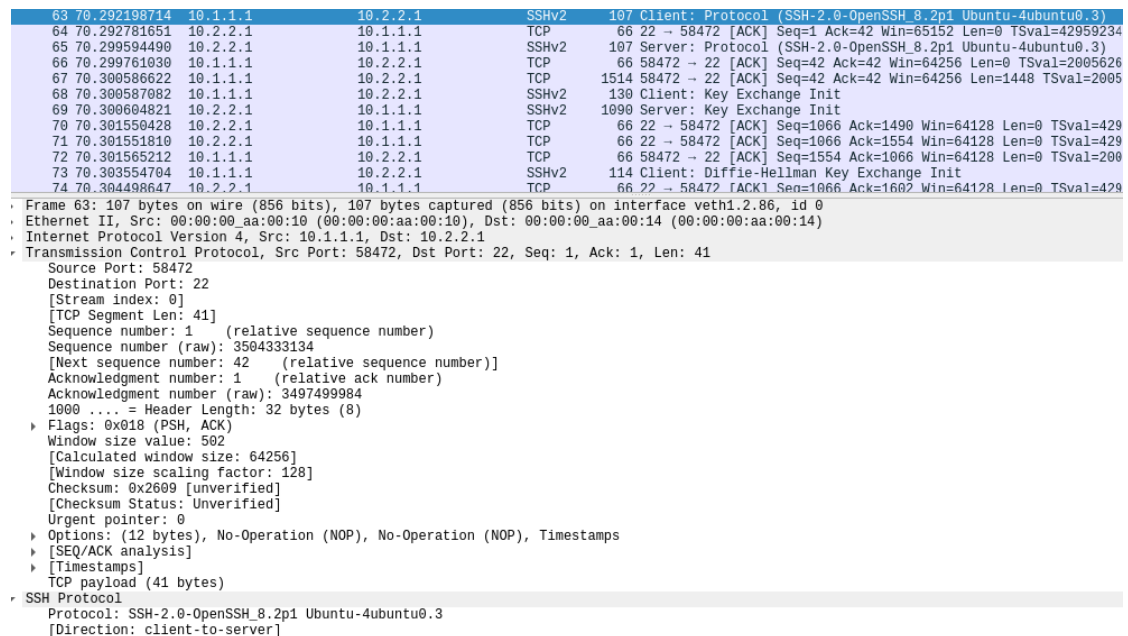


Figure 15: Captura SSH

2 Conclusão

Para a realização deste trabalho utilizamos a máquina virtual para ter acesso às ferramentas necessárias: o Core e o Wireshark.

Com a realização do trabalho, consolidamos bastantes conhecimentos sobre protocolos de transporte (TCP e UDP) e ainda sobre protocolos da camada aplicacional. Além disso, tivemos ainda a oportunidade de compará-los e identificar as suas vantagens e desvantagens.