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

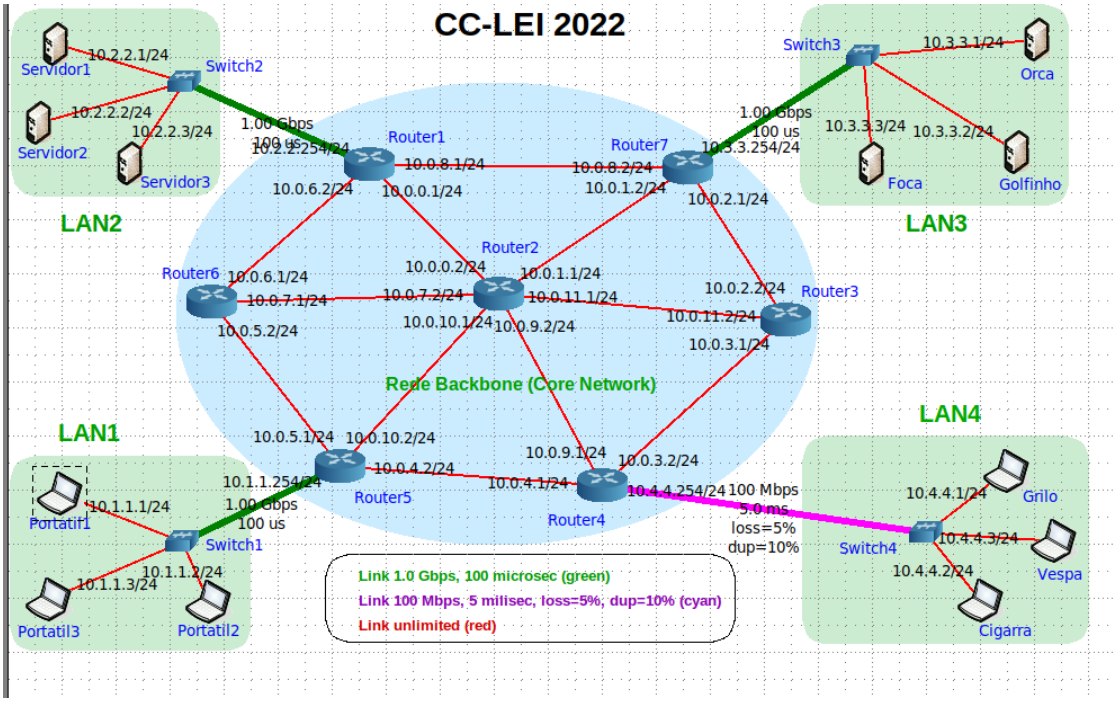

Figure 1: Topologia utilizada

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

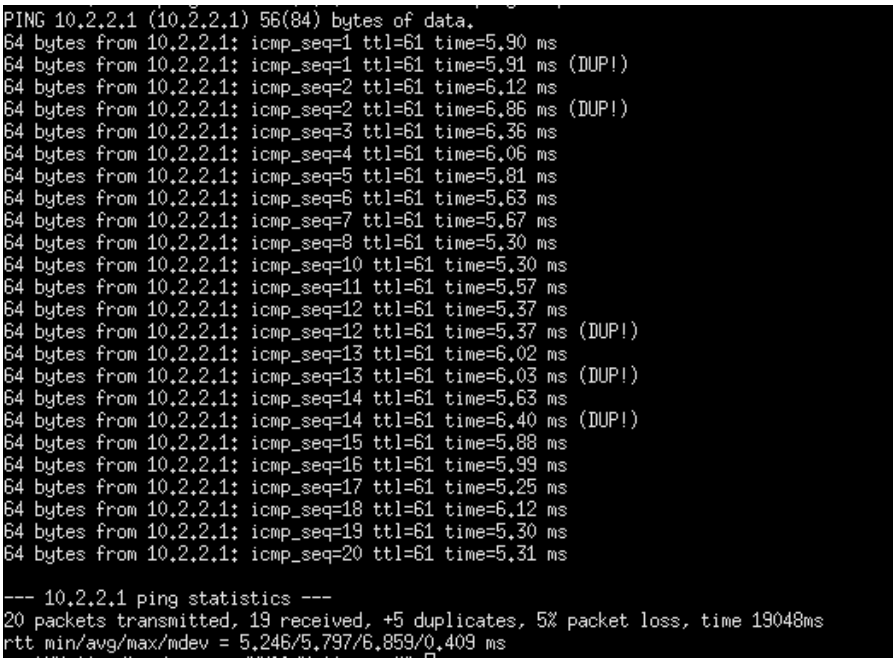

Figure 3: *Ping* Grilo -> Servidor1

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 cada de transporte. Caso se use UDP, a camada que trata dos problemas é a aplicacional. 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.

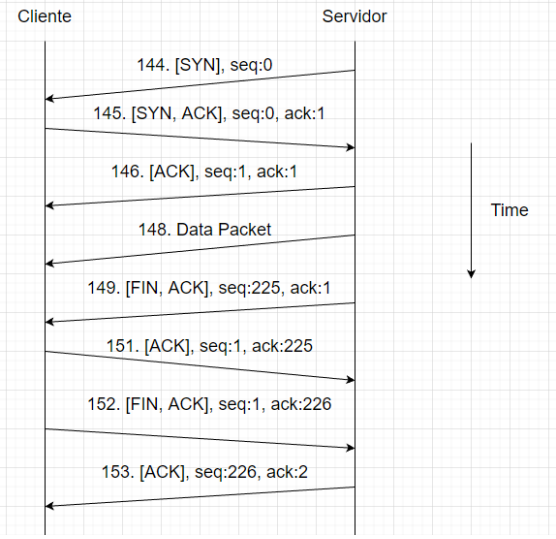

Figure 4: Diagrama Temporal FTP

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.

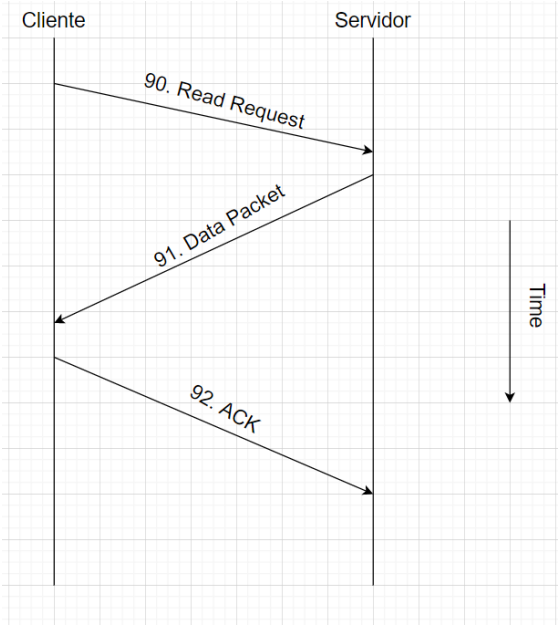

Figure 6: Diagrama Temporal TFTP

R: Inicialmente o cliente envia um *Read Request* ao servidor. Em seguida o sevidor 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 quatros aplicações de tranferência 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**	***Overhead* transportado em *bytes***
Ping	Ping	-	-	-
Traceroute	Traceroute	UDP	33446	20
Telnet	TELNET	TCP	23	20
Ftp	FTP	TCP	21	20
Tftp	TFTP	UDP	69	8
Wget	HTTP	TCP	80	20
Nslookup	DNS	UDP	53	8
Ssh	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*

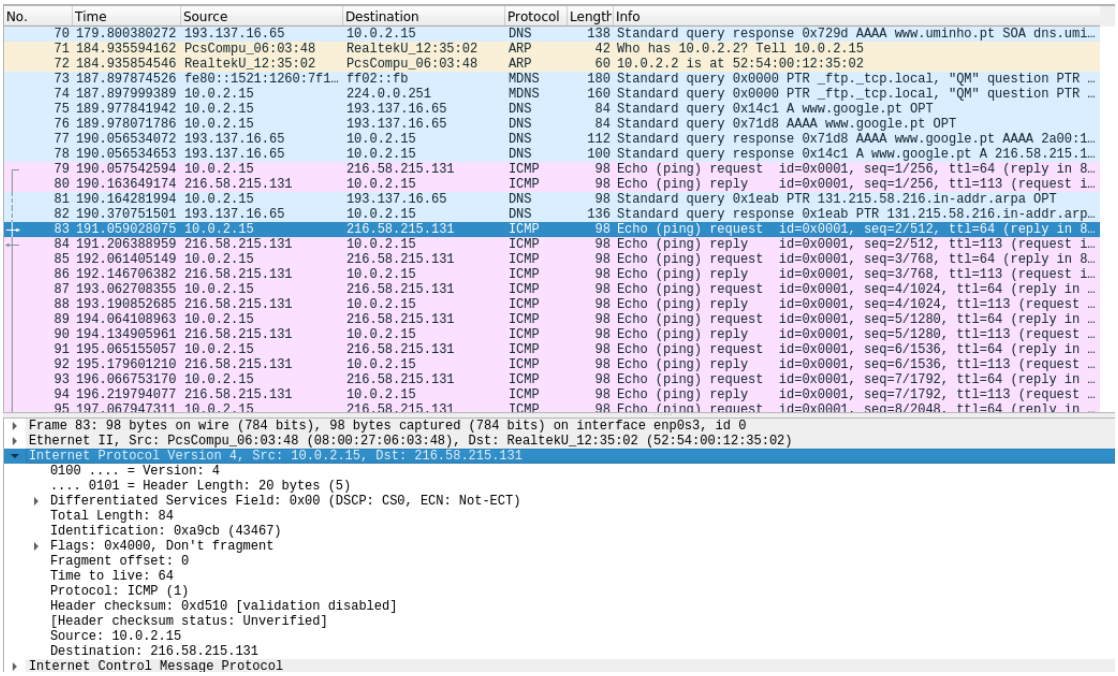

Figure 8: Captura *Ping*

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

1.5.2 *Traceroute*

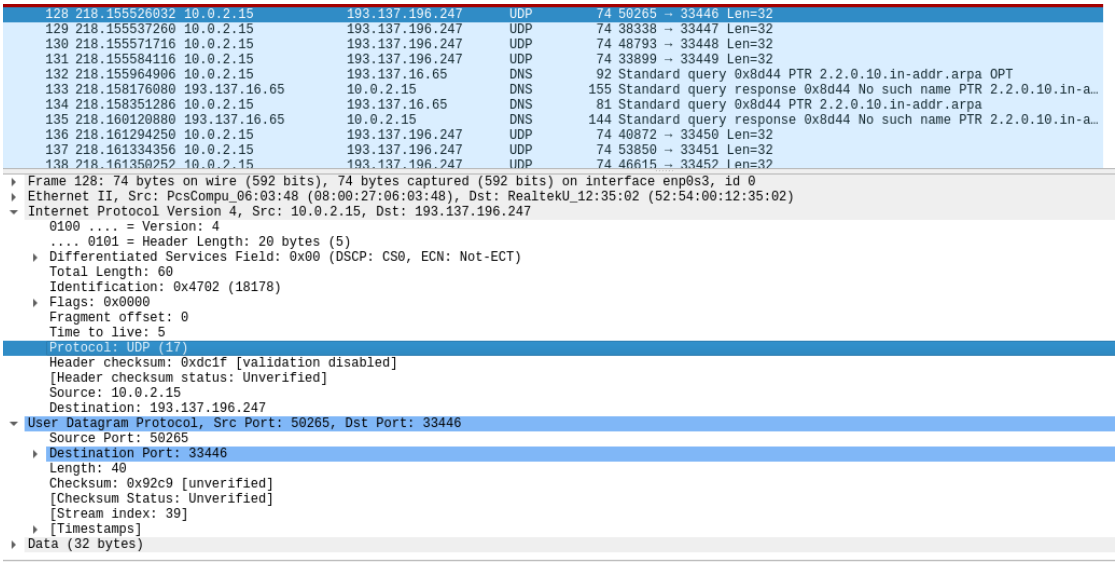

Figure 9: Captura *Traceroute*

1.5.3 *Telnet*

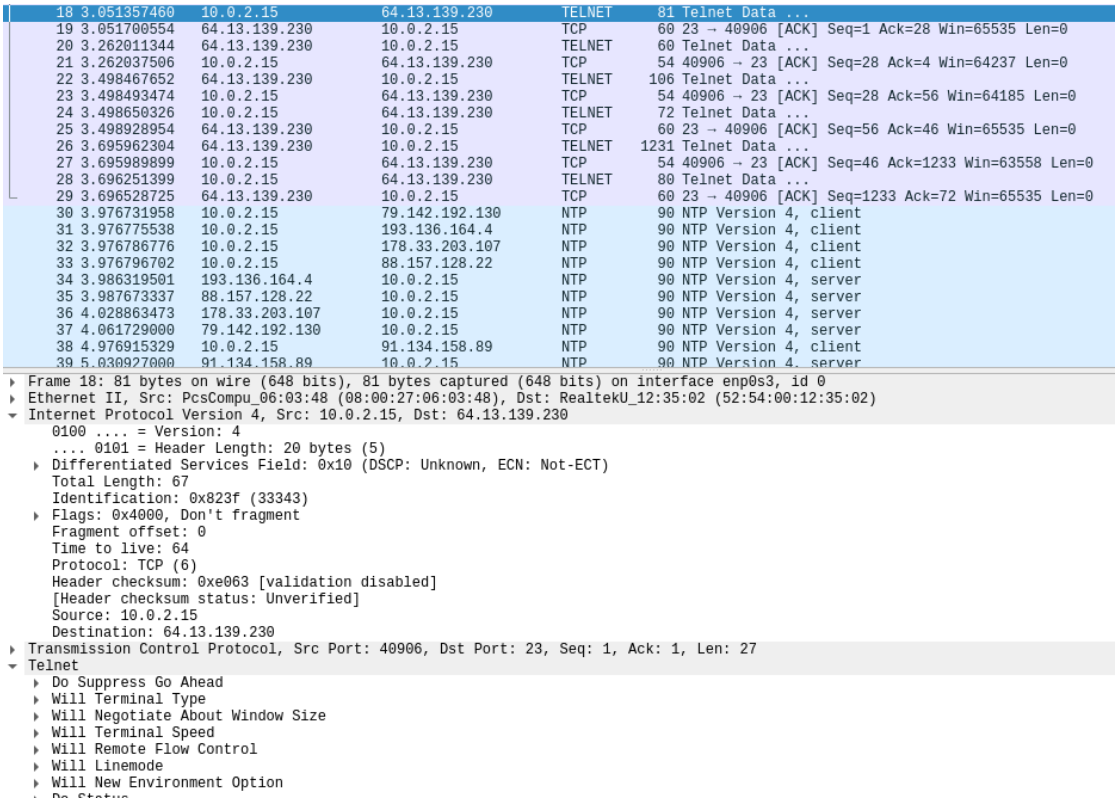

Figure 10: Captura *Telnet*

1.5.4 ***FTP***

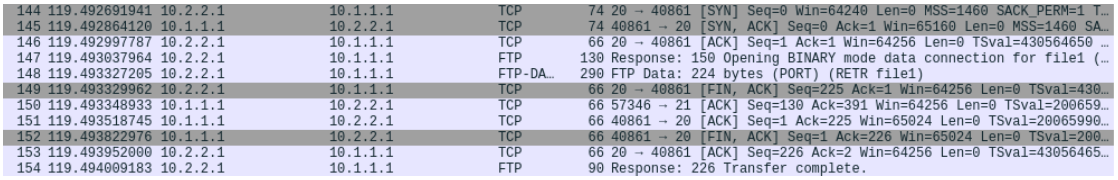

Figure 11: Captura FTP

1.5.5 *Tftp*

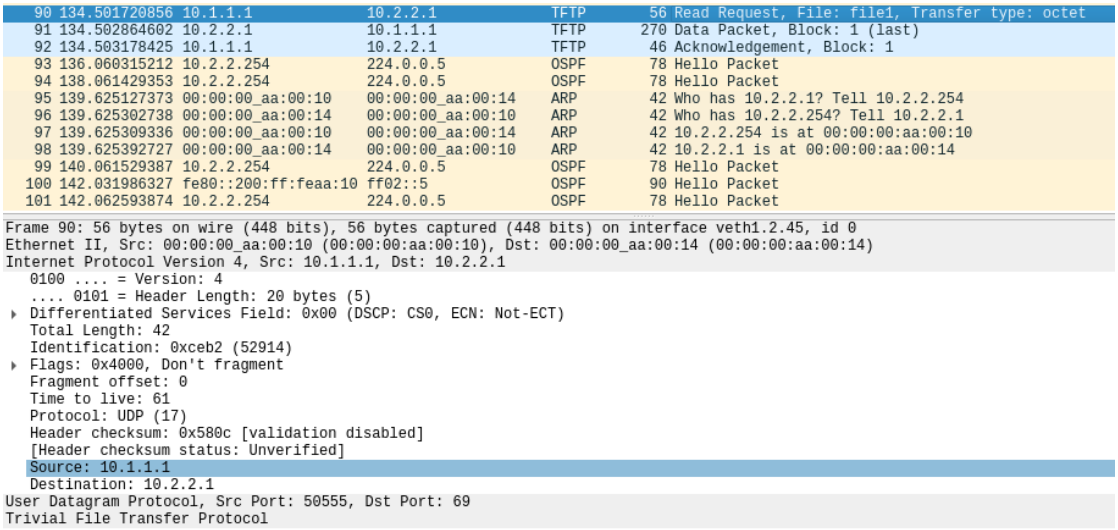

Figure 12: Captura FTP

1.5.6 *HTTP / Wget*

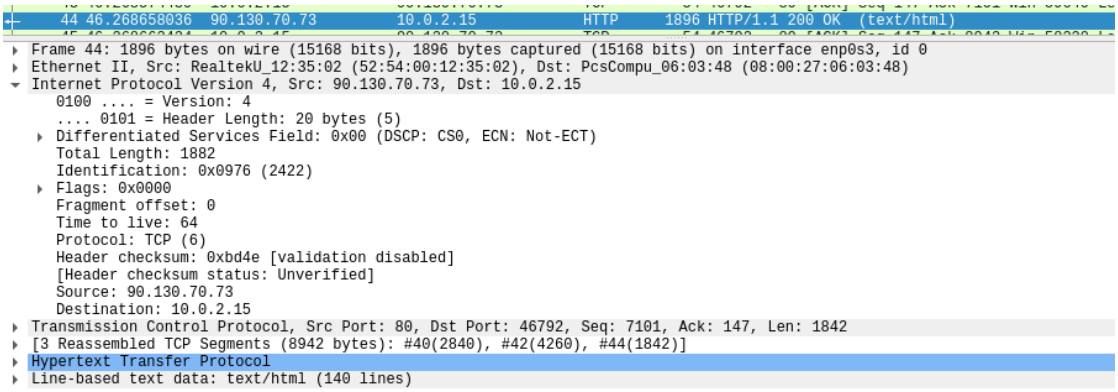

Figure 13: Captura HTTP

1.5.7 *Nslookup*

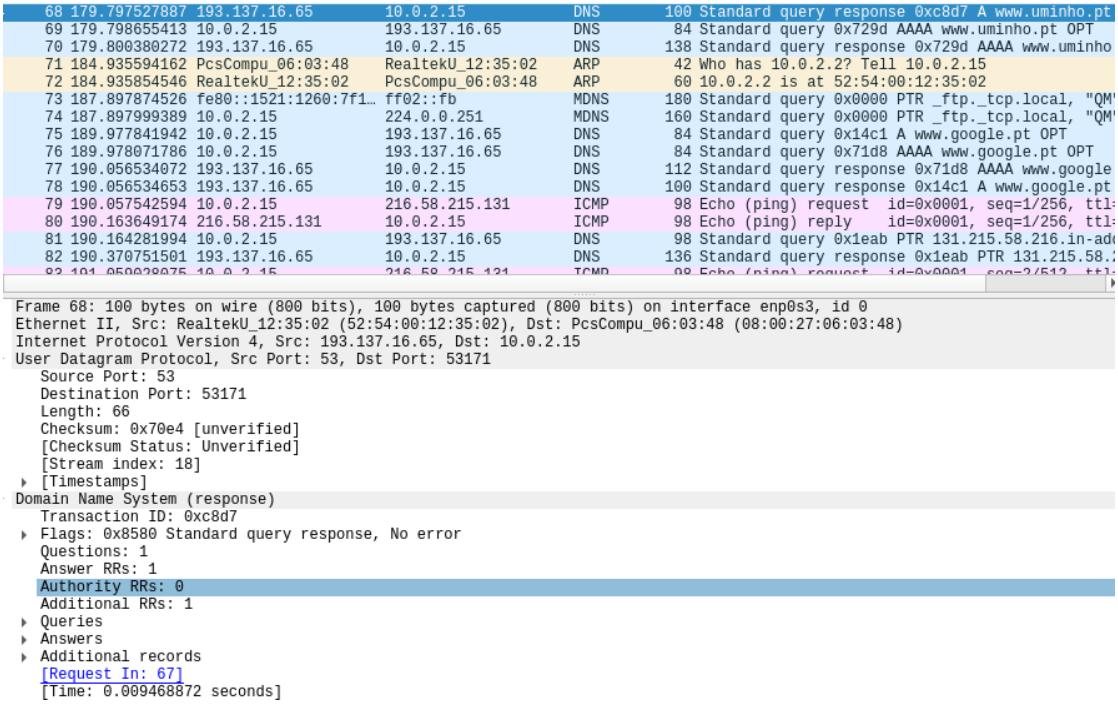

Figure 14: Captura Nslookup

1.5.8 *Ssh*

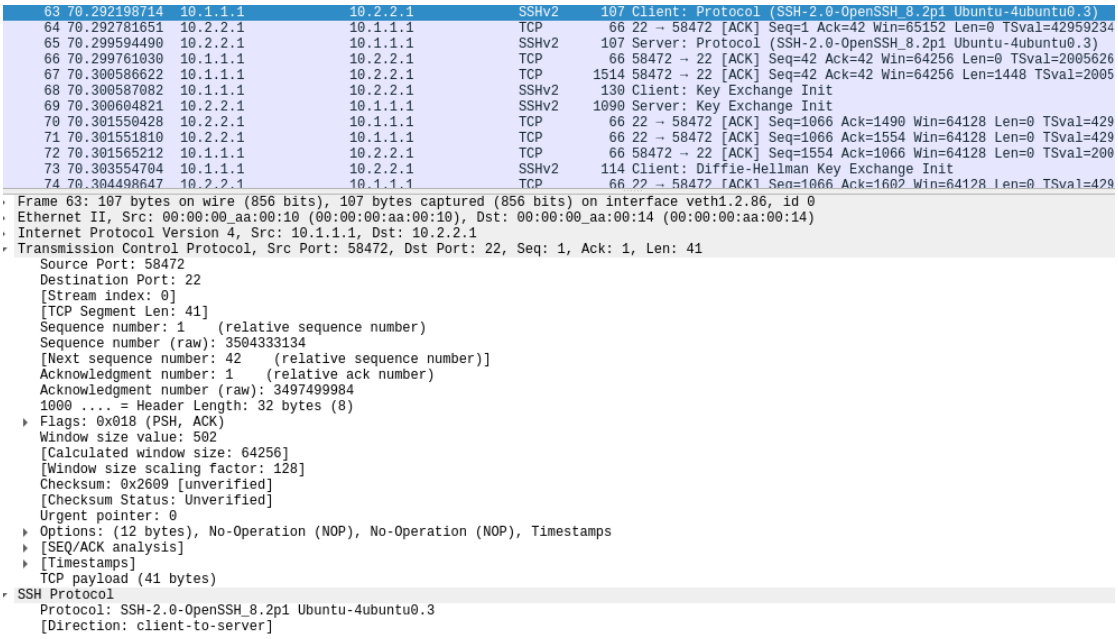

Figure 15: Captura SSH

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.