



Universidade do Minho
Escola de Engenharia

Comunicações por Computador

Trabalho Prático 1

Grupo 07

2022/2023

Autores:

A91671 – João Manuel Novais da Silva

A91697 – Luís Filipe Fernandes Vilas

A91660 – Pedro António Pires Correia Leite Sequeira

Docente:

Bruno Alexandre Fernandes Dias

Braga, 13 de outubro de 2022

Índice

Parte B	3
Questão 1	3
Questão 2	4
Questão 3	6
Questão 4	8
Questão 5	9
Conclusão	12

Parte B

Questão 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.

```
vcmd
root@Portatil1:/tmp/pycore.41017/Portatil1.conf# ping 10.2.2.1 -c 20
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=1.68 ms
64 bytes from 10.2.2.1: icmp_seq=2 ttl=61 time=0.481 ms
64 bytes from 10.2.2.1: icmp_seq=3 ttl=61 time=0.350 ms
64 bytes from 10.2.2.1: icmp_seq=4 ttl=61 time=0.374 ms
64 bytes from 10.2.2.1: icmp_seq=5 ttl=61 time=0.464 ms
64 bytes from 10.2.2.1: icmp_seq=6 ttl=61 time=1.41 ms
64 bytes from 10.2.2.1: icmp_seq=7 ttl=61 time=0.326 ms
64 bytes from 10.2.2.1: icmp_seq=8 ttl=61 time=0.653 ms
64 bytes from 10.2.2.1: icmp_seq=9 ttl=61 time=0.339 ms
64 bytes from 10.2.2.1: icmp_seq=10 ttl=61 time=0.402 ms
64 bytes from 10.2.2.1: icmp_seq=11 ttl=61 time=0.407 ms
64 bytes from 10.2.2.1: icmp_seq=12 ttl=61 time=0.457 ms
64 bytes from 10.2.2.1: icmp_seq=13 ttl=61 time=0.335 ms
64 bytes from 10.2.2.1: icmp_seq=14 ttl=61 time=0.422 ms
64 bytes from 10.2.2.1: icmp_seq=15 ttl=61 time=0.468 ms
64 bytes from 10.2.2.1: icmp_seq=16 ttl=61 time=0.553 ms
64 bytes from 10.2.2.1: icmp_seq=17 ttl=61 time=0.382 ms
64 bytes from 10.2.2.1: icmp_seq=18 ttl=61 time=0.414 ms
64 bytes from 10.2.2.1: icmp_seq=19 ttl=61 time=0.478 ms
64 bytes from 10.2.2.1: icmp_seq=20 ttl=61 time=0.393 ms

--- 10.2.2.1 ping statistics ---
20 packets transmitted, 20 received, 0% packet loss, time 19399ms
rtt min/avg/max/mdev = 0.326/0.539/1.680/0.345 ms
```

Figura 1 - Portátil

Prestando atenção às Figura 1 e Figura 2, verifica – se que, enquanto no Portatil 1 não existem pacotes duplicados e 0% de package loss, no Grilo há 1 pacote duplicado. A maior parte das vezes, isto deve – se a problemas ao nível das infraestruturas de rede, algo que não acontece com o Portátil 1, pois na rede de Backbone dispõe de 1 Gbps de banda, contrariamente à rede de Backbone do Grilo, que será possivelmente outro tipo de tecnologias e apenas dispõe de uma conexão a 100 Mbps, com 5% Loss e 10% Duplicação (Figura 4).

```
vcmd
rtt min/avg/max/mdev = 5.345/6.180/8.913/1.369 ms
root@Grilo:/tmp/pycore.41017/Grilo.conf# ping 10.2.2.1 -c 20
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=5.30 ms
64 bytes from 10.2.2.1: icmp_seq=2 ttl=61 time=5.57 ms
64 bytes from 10.2.2.1: icmp_seq=3 ttl=61 time=6.07 ms
64 bytes from 10.2.2.1: icmp_seq=4 ttl=61 time=6.18 ms
64 bytes from 10.2.2.1: icmp_seq=4 ttl=61 time=6.18 ms (DUP!)
64 bytes from 10.2.2.1: icmp_seq=5 ttl=61 time=6.03 ms
64 bytes from 10.2.2.1: icmp_seq=6 ttl=61 time=7.81 ms
64 bytes from 10.2.2.1: icmp_seq=7 ttl=61 time=7.24 ms
64 bytes from 10.2.2.1: icmp_seq=8 ttl=61 time=5.28 ms
64 bytes from 10.2.2.1: icmp_seq=9 ttl=61 time=6.95 ms
64 bytes from 10.2.2.1: icmp_seq=10 ttl=61 time=5.23 ms
64 bytes from 10.2.2.1: icmp_seq=11 ttl=61 time=5.31 ms
64 bytes from 10.2.2.1: icmp_seq=12 ttl=61 time=5.87 ms
64 bytes from 10.2.2.1: icmp_seq=13 ttl=61 time=5.53 ms
64 bytes from 10.2.2.1: icmp_seq=14 ttl=61 time=6.01 ms
64 bytes from 10.2.2.1: icmp_seq=15 ttl=61 time=5.26 ms
64 bytes from 10.2.2.1: icmp_seq=16 ttl=61 time=5.78 ms
64 bytes from 10.2.2.1: icmp_seq=17 ttl=61 time=5.36 ms
64 bytes from 10.2.2.1: icmp_seq=18 ttl=61 time=5.61 ms
64 bytes from 10.2.2.1: icmp_seq=19 ttl=61 time=6.04 ms
64 bytes from 10.2.2.1: icmp_seq=20 ttl=61 time=5.33 ms

--- 10.2.2.1 ping statistics ---
20 packets transmitted, 20 received, +1 duplicates, 0% packet loss, time 19033ms
rtt min/avg/max/mdev = 5.234/5.901/7.811/0.679 ms
```

Figura 4 - Grilo

Link 1.0 Gbps, 100 microsec (green)

Figura 2 - Backbone Portátil

Link 100 Mbps, 5 milisec, loss=5%, dup=10% (cyan)

Figura 3 - Backbone Grilo

Questão 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.

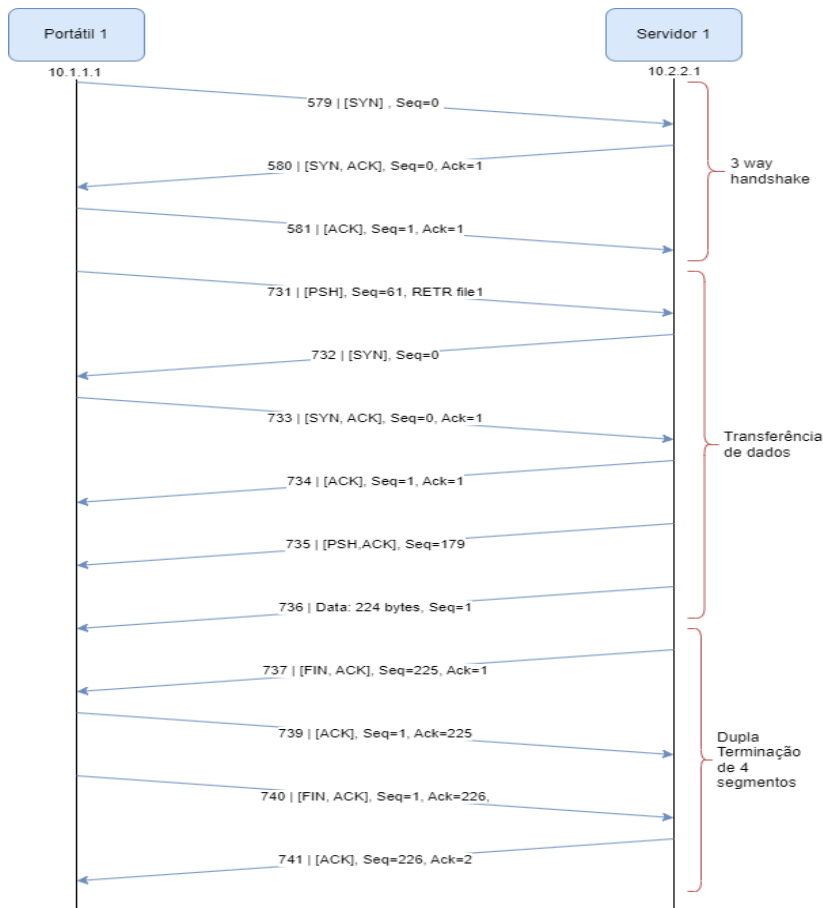


Figura 5 - TCP na transferência do ficheiro 1

Na transferência do *file1* por FTP realizada na A.3 houveram pedidos de início de conexão (figura 6) que consistiram num envio inicial de um segmento do tipo [SYN] com [Seq=0] do portátil 1 (10.1.1.1) para o servidor 1 (10.2.2.1), este respondeu com um segmento de tipo [ACK,SYN] e [Seq=0], e por fim um segmento de [ACK] e [Seq=1] do portátil 1. Sendo assim, óbvio que houve um *3 way handshake* para dar início á conexão entre o servidor 1 e o portátil 1 a partir do FTP uma vez que este recorre ao TPC como protocolo de transporte.

No que diz respeito ao início da transferência de dados, é enviado, pelo portátil 1, um segmento de tipo [PSH] com [Seq=61] nomeadamente **Request: RETR file1** para o servidor 1 (figura 7). Este é respondido, pelo servidor, com um segmento do tipo [SYN], assim como, a indicação do número de sequência inicial [Seq=0] e do tamanho de janela para a seguinte transferência. Com isto, o portátil 1 aloca o espaço de armazenamento

necessário e especifica o número de sequência [Seq=0] enviando um segmento do tipo [SYN,ACK]. Por fim, o servidor 1 responde com um segmento [ACK] com o número de sequência igual a 1 ([Seq=1]) e agora já é possível haver transferência de dados.

Analisando os segmentos capturados conseguimos identificar o processo de transferência do file1 entre o servidor e o portátil, visível no segmentos 735 com número de sequência [Seq=179] do tipo [PSH,ACK] e no segmento 736 com número de sequência [Seq=1] e do tipo [PSH,ACK] (figura 8).

Depois de concluída esta transferência o servidor dá sinal que quer terminar a conexão, sendo que já não tem mais informação para enviar, a partir do segmento 737 do tipo [FIN,ACK] e com [Seq=225]. O portátil responde com a verificação da receção do ficheiro através do segmento 738 do tipo [ACK] e [Seq=73], e também, seguidamente, repõe com o segmento 739 que é uma resposta ao pedido de término da conexão do tipo [ACK] e [Seq=1]. Além disto o portátil envia um segmento a indicar o fim da conexão do tipo [FIN,ACK] e [Seq=1]. Por fim o servidor responde com um [ACK] [Seq=226] na forma do segmento 741.

Com este processo de final de conexão podemos identificar uma dupla terminação de 4 segmentos característico do protocolo TCP.

579	384.2418...	10.1.1.1	10.2.2.1	TCP	74 43748 → 21 [SYN] Seq=0 Win=64240 Len=0 MSS...
580	384.2420...	10.2.2.1	10.1.1.1	TCP	74 21 → 43748 [SYN, ACK] Seq=0 Ack=1 Win=6516...
581	384.2423...	10.1.1.1	10.2.2.1	TCP	66 43748 → 21 [ACK] Seq=1 Ack=1 Win=64256 Len=...

Figura 6 - 3 way handshake no FTP.

731	454.07000...	10.1.1.1	10.2.2.1	FTP	78 Request: RETR file1
732	454.07021...	10.2.2.1	10.1.1.1	TCP	74 20 → 40383 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 ...
733	454.07034...	10.1.1.1	10.2.2.1	TCP	74 40383 → 20 [SYN, ACK] Seq=0 Ack=1 Win=65160 Len=...
734	454.07047...	10.2.2.1	10.1.1.1	TCP	66 20 → 40383 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSv...

Figura 7 - Início de transferência de dados.

735	454.07050...	10.2.2.1	10.1.1.1	FTP	130 Response: 150 Opening BINARY mode data connectio...
736	454.07055...	10.2.2.1	10.1.1.1	FTP-...	290 FTP Data: 224 bytes (PORT) (RETR file1)
737	454.07055...	10.2.2.1	10.1.1.1	TCP	66 20 → 40383 [FIN, ACK] Seq=225 Ack=1 Win=64256 Le...
738	454.07071...	10.1.1.1	10.2.2.1	TCP	66 43750 → 21 [ACK] Seq=73 Ack=243 Win=64256 Len=0 ...
739	454.07071...	10.1.1.1	10.2.2.1	TCP	66 40383 → 20 [ACK] Seq=1 Ack=225 Win=65024 Len=0 T...
740	454.07110...	10.1.1.1	10.2.2.1	TCP	66 40383 → 20 [FIN, ACK] Seq=1 Ack=226 Win=65024 Le...
741	454.07145...	10.2.2.1	10.1.1.1	TCP	66 20 → 40383 [ACK] Seq=226 Ack=2 Win=64256 Len=0 T...
742	454.07145...	10.2.2.1	10.1.1.1	FTP	90 Response: 226 Transfer complete.
743	454.07170...	10.1.1.1	10.2.2.1	TCP	66 43750 → 21 [ACK] Seq=73 Ack=267 Win=64256 Len=0 ...

Figura 8 - Processo de transferência de dados e fim da mesma conexão.

Questão 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.

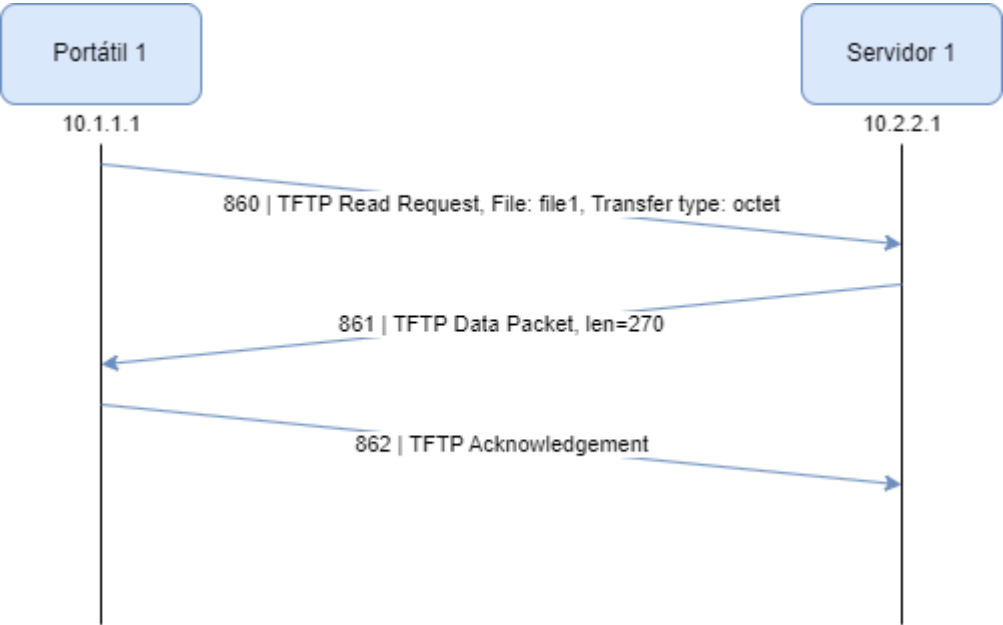


Figura 9 - Transferência de ficheiro por UDP

Time	Source	Destination	Protocol	Length	Info
2436.3806236...	10.1.1.1	10.2.2.1	TFTP	56	Read Request, File: file1, Transfer type: octet
2436.3818817...	10.2.2.1	10.1.1.1	TFTP	270	Data Packet, Block: 1 (last)
2436.3822877...	10.1.1.1	10.2.2.1	TFTP	46	Acknowledgement, Block: 1
2455.3617204...	10.4.4.1	10.2.2.1	TFTP	56	Read Request, File: file1, Transfer type: octet
2455.3623872...	10.2.2.1	10.4.4.1	TFTP	270	Data Packet, Block: 1 (last)
2455.3679497...	10.4.4.1	10.2.2.1	TFTP	46	Acknowledgement, Block: 1

```
4
+-----+
| Frame 2017: 56 bytes on wire (448 bits), 56 bytes captured (448 bits) on interface veth1.2.36, 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 |
+-----+
| User Datagram Protocol, Src Port: 51490, Dst Port: 69 |
+-----+
|   Source Port: 51490 |
|   Destination Port: 69 |
|   Length: 22 |
|   Checksum: 0xc3bc [unverified] |
|   [Checksum Status: Unverified] |
|   [Stream index: 2] |
|   [Timestamps] |
+-----+
| Trivial File Transfer Protocol |
+-----+
```

Figura 10 - Pedido do ficheiro através do TFTP

No caso da obtenção do file1 através do método TFTP (figura 9 e 10), como este utiliza o UDP para transporte dos dados, não existe nem fase de início de conexão nem de término da conexão. De igual forma, não existe número de sequência.

Existe apenas um pedido do cliente para transferência do file1, e posteriormente o envio do mesmo por parte do servidor. Quando o ficheiro é recebido, a aplicação “TFTP” envia um *acknowledgement*.

Questão 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.

Tabela 1 - Os dados de eficiência, complexidade e segurança são cotados de Baixa, Médio e Alta, explicando esse raciocínio em baixo.

	Camada de Transporte	Eficiência	Complexidade	Segurança
SFTP	TCP	baixa	alta	alta
FTP	TCP	médio	médio	médio
TFTP	UDP	alta	baixa	baixa
HTTP	TCP	média	média	baixa

O SFTP utiliza o TCP para transporte que permite correção de erros e prevenção de falhas de pacotes. De igual modo, os dados são encriptados o que revela maior complexidade subjacente a uma menor eficiência (tabela 1).

O FTP, também, o TCP para transporte dos dados, mas contrariamente ao SFTP estes não são encriptados, o que revela menor segurança e complexidade em relação ao SFTP. Em troca disto o FTP possui maior eficiência (tabela 1).

O TFTP é o único que utiliza UDP para transporte, apesar de a aplicação controlar o envio de *acknowledgement* para o destino. Desta forma, não existe nem início de conexão (*3 way handshake* do TCP) nem término da mesma (dupla transmissão de 4 segmentos do TCP). Esta aplicação também não utiliza qualquer encriptação dos dados o que, por sua vez, leva a uma baixa complexidade e baixa segurança, mas uma elevada eficiência (tabela 1).

Para terminar temos o HTTP que, não dispõe de medidas de segurança, mas utiliza o TCP para uma transmissão fiável dos dados ponto a ponto realizando início e término de conexão. Esta aplicação é menos eficiente do que o TFTP e mais complexa que a mesma (tabela 1).

Questão 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.

Tabela 2 - Tabela Informativa

Aplicação	Protocolo de Aplicação	Protocolo de Transporte	Porta de Atendimento	Overhead de Transporte
ping	-	-	-	-
traceroute	-	UDP/TCP	33445-33545	8 bytes
telnet	Telnet	TCP	23	20 bytes *
ftp	FTP	TCP	21	44 bytes
tftp	TFTP	UDP	69	8 bytes
wget	HTTP	TCP	80	44 bytes ***
nslookup	DNS	UDP	53	8 bytes *
ssh	SSH	TCP	22	44 bytes ***

* - não foi possível, devido a problemas no servidor, estabelecer a conexão, logo não nos foi possível obter fotos do wireshark.

*** - SYN (40 bytes) + SYN,ACK (40 bytes) + 3*ACK (32 bytes) + FIN (32 bytes) + TCP Segments (32 bytes)

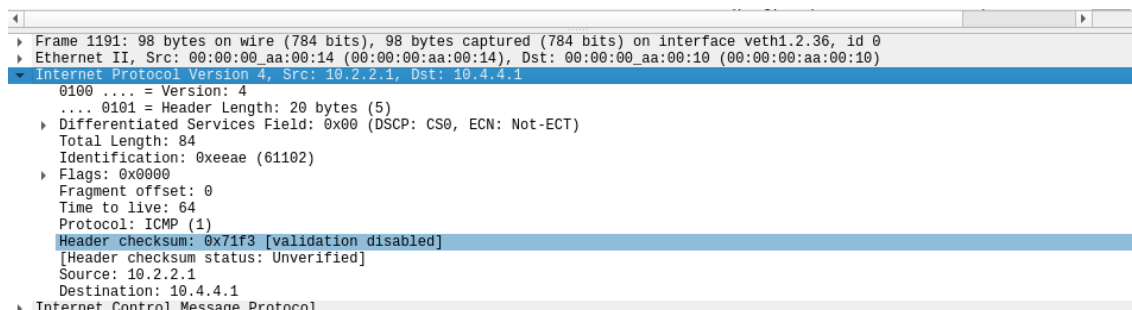


Figura 11 - Ping Wireshark

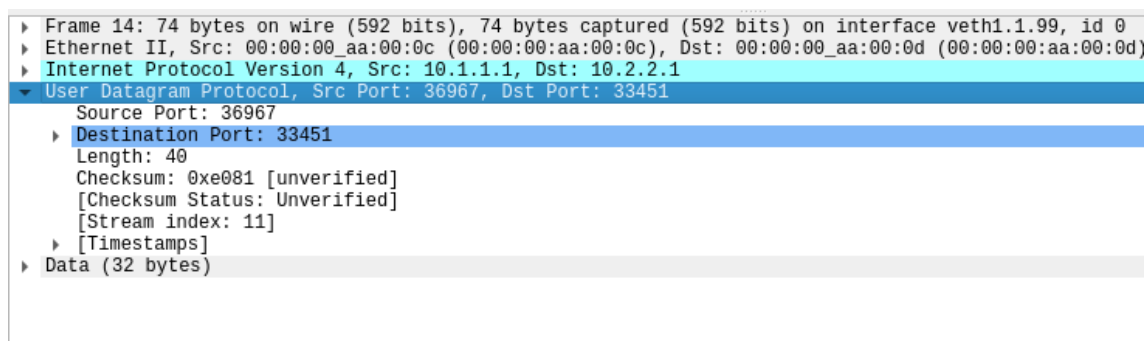


Figura 12 - UDP Wireshark no traceroute

916229...	10.1.1.1	10.2.2.1	TCP	66 36126 → 21 [ACK] Seq=1 ACK=1 Win=64256 Len=0 IsVal=15456/1386...
983859...	10.2.2.1	10.1.1.1	FTP	86 Response: 220 (vsFTPd 3.0.3)

<ul style="list-style-type: none"> Transmission Control Protocol, Src Port: 21, Dst Port: 36126, Seq: 1, Ack: 1, Len: 20 <ul style="list-style-type: none"> Source Port: 21 Destination Port: 36126 [Stream index: 2] [TCP Segment Len: 20] Sequence number: 1 (relative sequence number) Sequence number (raw): 3292264896 [Next sequence number: 21 (relative sequence number)] Acknowledgment number: 1 (relative ack number) Acknowledgment number (raw): 947399157 1000 = Header Length: 32 bytes (8) Flags: 0x018 (PSH, ACK) Window size value: 510 [Calculated window size: 65280] [Window size scaling factor: 128] Checksum: 0xfbf4 [unverified] [Checksum Status: Unverified] Urgent pointer: 0 Options: (12 bytes), No-Operation (NOP), No-Operation (NOP), Timestamps <ul style="list-style-type: none"> [SEQ/ACK analysis] [Timestamps] TCP payload (20 bytes) File Transfer Protocol (FTP)
--

Figura 13 - FTP (12 bytes Opções + 32 Header)

2436.3818817...	10.2.2.1	10.1.1.1	TFTP	270 Data Packet, Block: 1 (last)
2436.3822877...	10.1.1.1	10.2.2.1	TFTP	46 Acknowledgement, Block: 1
2455.3617284...	10.4.4.1	10.2.2.1	TFTP	56 Read Request, File: file1, Transfer type: octet
2455.3623872...	10.2.2.1	10.4.4.1	TFTP	270 Data Packet, Block: 1 (last)
2455.3679497...	10.4.4.1	10.2.2.1	TFTP	46 Acknowledgement, Block: 1

<ul style="list-style-type: none"> Frame 2018: 270 bytes on wire (2160 bits), 270 bytes captured (2160 bits) on interface veth1.2.36, id 0 Ethernet II, Src: 00:00:00_aa:00:14 (00:00:00:aa:00:14), Dst: 00:00:00_aa:00:10 (00:00:00:aa:00:10) Internet Protocol Version 4, Src: 10.2.2.1, Dst: 10.1.1.1 User Datagram Protocol, Src Port: 60673, Dst Port: 51490 <ul style="list-style-type: none"> Source Port: 60673 Destination Port: 51490 Length: 236 Checksum: 0xf16f [unverified] [Checksum Status: Unverified] [Stream index: 3] [Timestamps] Trivial File Transfer Protocol <ul style="list-style-type: none"> Opcode: Data Packet (3) [Source File: file1] Block: 1 [Full Block Number: 1] Data (224 bytes)

Figura 14 - TFTP UDP

2236.2527.1497960...	10.4.4.1	10.2.2.1	HTTP	206 GET /file1 HTTP/1.1
2238.2527.1507726...	10.2.2.1	10.4.4.1	HTTP	508 HTTP/1.1 200 OK (text/plain)
2245.2529.0754392...	10.4.4.1	10.2.2.1	HTTP	206 GET /file2 HTTP/1.1
2247.2529.0758818...	10.4.4.1	10.2.2.1	HTTP	206 [TCP Spurious Retransmission] GET /file2 HTTP/1.1

<ul style="list-style-type: none"> Frame 2236: 206 bytes on wire (1648 bits), 206 bytes captured (1648 bits) on interface veth1.2.36, 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: 10.4.4.1, Dst: 10.2.2.1 Transmission Control Protocol, Src Port: 35816, Dst Port: 80, Seq: 1, Ack: 1, Len: 140 <ul style="list-style-type: none"> Source Port: 35816 Destination Port: 80 [Stream index: 13] [TCP Segment Len: 140] Sequence number: 1 (relative sequence number) Sequence number (raw): 2283680986 [Next sequence number: 141 (relative sequence number)] Acknowledgment number: 1 (relative ack number) Acknowledgment number (raw): 2133936317 1000 = Header Length: 32 bytes (8) Flags: 0x018 (PSH, ACK) Window size value: 502 [Calculated window size: 64256] [Window size scaling factor: 128] Checksum: 0x2032 [unverified] [Checksum Status: Unverified] Urgent pointer: 0 Options: (12 bytes), No-Operation (NOP), No-Operation (NOP), Timestamps <ul style="list-style-type: none"> [SEQ/ACK analysis] [Timestamps]
--

Figura 15 - TCP no HTTP

204815...	10.1.1.1	10.2.2.1	SSHv2	107 Client: Protocol (SSH-2.0-OpenSSH_8.2p1 Ubuntu-4ubuntu0.3)	
217381...	10.2.2.1	10.1.1.1	TCP	66 22 → 54772 [ACK] Seq=1 Ack=42 Win=65152 Len=0 TSval=330432769...	
573687...	10.2.2.1	10.1.1.1	SSHv2	107 Server: Protocol (SSH-2.0-OpenSSH_8.2p1 Ubuntu-4ubuntu0.3)	
579690...	10.1.1.1	10.2.2.1	TCP	66 54772 → 22 [ACK] Seq=42 Ack=42 Win=64256 Len=0 TSval=15454676...	
581700...	10.1.1.1	10.2.2.1	TCP	1514 54772 → 22 [ACK] Seq=42 Ack=42 Win=64256 Len=1448 TSval=15454...	

▶ Frame 1259: 107 bytes on wire (856 bits), 107 bytes captured (856 bits) on interface veth1.2.36, 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
 ▶ Transmission Control Protocol, Src Port: 54772, Dst Port: 22, Seq: 1, Ack: 1, Len: 41

- Source Port: 54772
- Destination Port: 22
- [Stream index: 0]
- [TCP Segment Len: 41]
- Sequence number: 1 (relative sequence number)
- Sequence number (raw): 645314068
- [Next sequence number: 42 (relative sequence number)]
- Acknowledgment number: 1 (relative ack number)
- Acknowledgment number (raw): 1292758361
- 1000 ... = Header Length: 32 bytes (8)
- Flags: 0x018 (PSH, ACK)
- Window size value: 502
- [Calculated window size: 64256]
- [Window size scaling factor: 128]
- Checksum: 0x2b74 [unverified]
- [Checksum Status: Unverified]
- Urgent pointer: 0
- Options: (12 bytes), No-Operation (NOP), No-Operation (NOP), Timestamps
- [SEQ/ACK analysis]
- [Timestamps]...

Figura 16 - SSH com base em TCP

Conclusão

A realização deste trabalho ajudou-nos a consolidar vários conceitos alusivos às camadas de transporte e aplicação vistas nas aulas teóricas.

Estudamos a diferença entre os protocolos TPC (Transmission Control Protocol) e UDP (User Datagram Protocol) e as situações em que os mesmos são aplicados. A complexidade do TCP permite um melhor controlo de erros e de fluxo e permite a segurança na transmissão de dados. O UDP possui uma maior eficiência devido a ser menos complexo permitindo as aplicações que usam o mesmo ter um maior through put de dados apesar de ocorrerem mais erros e perda de informação.

Assim, ficamos a compreender melhor os casos em que estes protocolos são usados e as situações em que os devemos implementar.