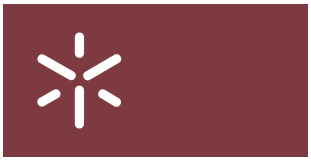


Universidade do Minho
Licenciatura em Engenharia Informática

Comunicações por Computador

Trabalho Prático nº1
PL3 - GRUPO 9
Protocolos da Camada de Transporte

Henrique Pereira (a100831) Maya Gomes (a100822) José Faria(a95255)

October 3, 2023

Conteúdos

1 Introdução

Neste exercício pretende-se transferir o mesmo ficheiro usando 4 serviços diferentes: SFTP, FTP, TFTP e HTTP, capturando todos os pacotes trocados durante a transferência com o Wireshark.

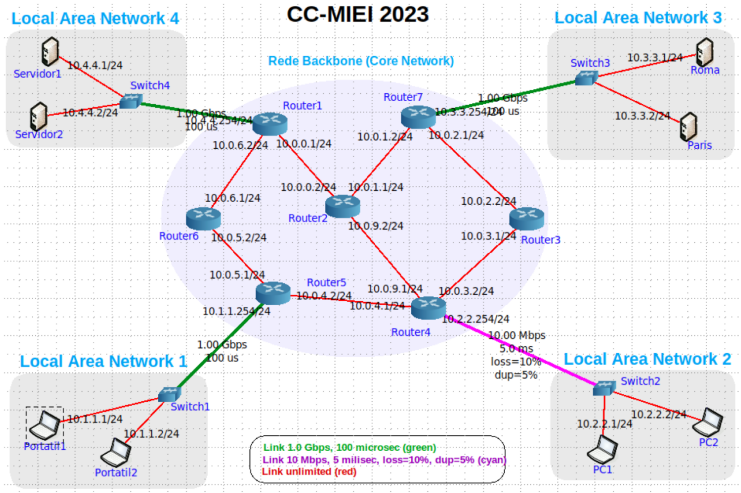


Figure 1: Topologia de Rede (backbone, acesso e local).

2 Questões e Respostas

2.1 QUESTÕES (Parte I)

2.1.1 Questão 1:

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

Os pacotes são conjuntos de dados que ocasionalmente não conseguem alcançar o seu destino, e às vezes podem-se perder durante o processo de transferência. Essa perda pode afetar naturalmente várias aplicações, mas tem um impacto mais significativo naquelas que dependem de transferências de dados em tempo real. No entanto, perdas ocasionais geralmente não afetam muito o desempenho geral da rede, mas a ocorrência frequente de perda de pacotes representa efetivamente um problema. Essas perdas podem ser causadas por uma série de razões, como aumento no tráfego, erros de software, problemas de hardware e software de rede desatualizados, bem como violações de segurança, como ataques e ameaças à integridade da rede.

A camada que lida com as perdas e duplicações é a camada de transporte, com o seu protocolo confiável TCP. Os resultados e efeitos destas perdas variam consoante o protocolo em questão. Desta forma, são os protocolos TCP (Protocolo de Controlo de Transmissão) e UDP (Protocolo de Datagrama de Usuário) que procuram resolver este problema. O efeito dessas perdas e/ou duplicações são os seguintes:

• Um protocolo de transmissão (TCP) é responsável por resolver estas perdas. O TCP é um protocolo orientado à conexão e é encarregado de gerenciar o controle de fluxo da comunicação. Ele avalia, identifica e retransmite os pacotes (controlo de erros), quando ocorrem problemas específicos, garantindo assim a correção de erros durante a transmissão.

• Um protocolo de Usuário (UDP), não possui recursos para corrigir erros na transmissão, limitando-se a detetá-los. Assim, ele descarta diretamente o pacote em vez de o reenviar. Desta forma, os protocolos superiores terão de ser responsáveis por reenviar os pacotes para que não se percam. Além disso, no UDP, não ocorre segmentação dos dados em pacotes menores, e não são enviados ACKs (confirmações de recebimento) para confirmar que os pacotes foram entregues com sucesso.

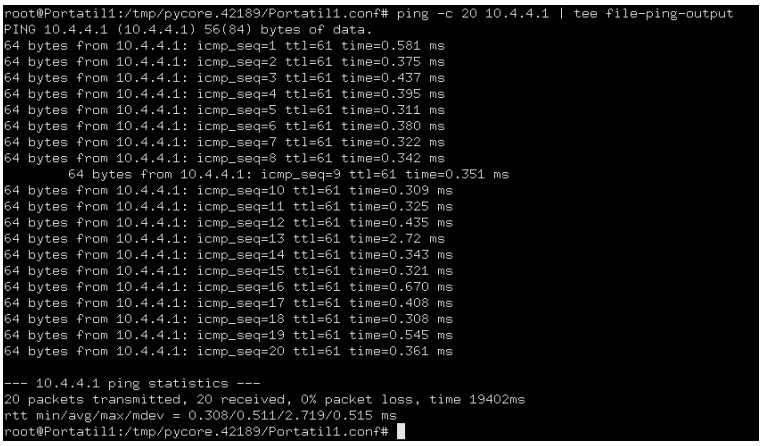


Figure 2: Comando PING Portatil1 - conectividade

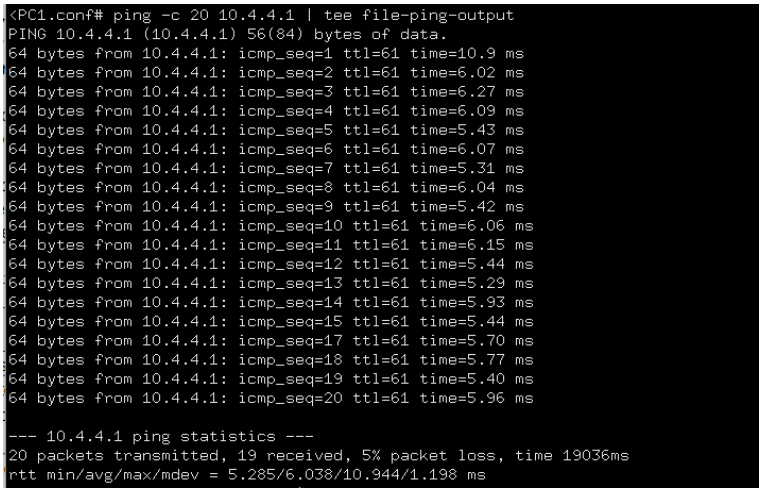


Figure 3: Comando PING PC1 - conectividade

No que toca à duplicação de pacotes podemos observar que entre o Portátil da LAN 1 e o PC1 da LAN 2 o PING é diferente sendo muito menor no Portátil1.

Notamos com o comando PING ocorre perda de pacotes, pois enquanto que no Portatil1 são transmitidos e recebidos 20 pacotes, no PC1 são transmitidos 20 pacotes mas somente 19 são recebidos.

Figure 4: ftp PC1

Figure 5: Wireshark ftp file1 no PC1

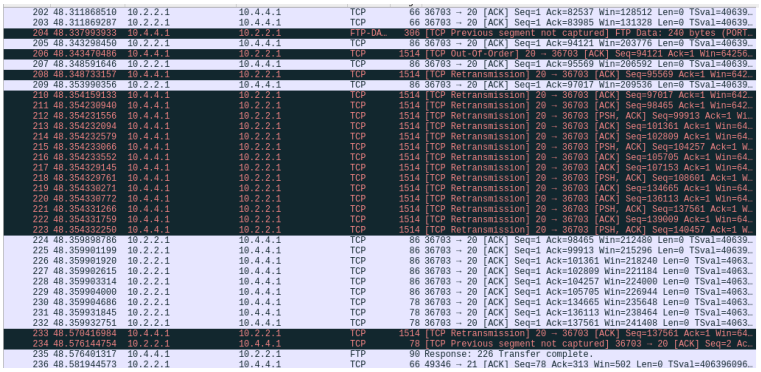
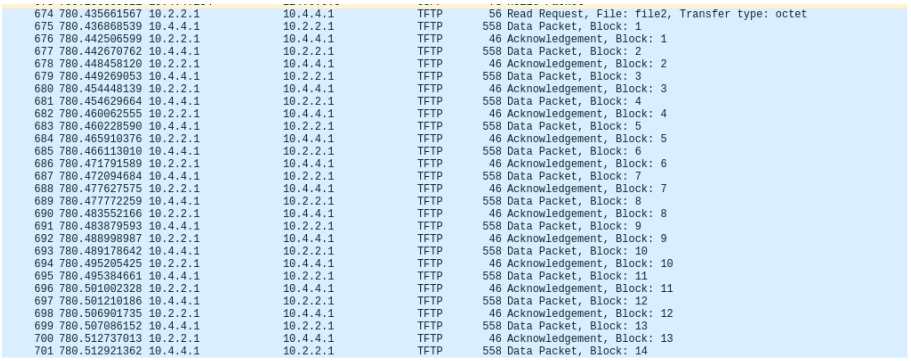
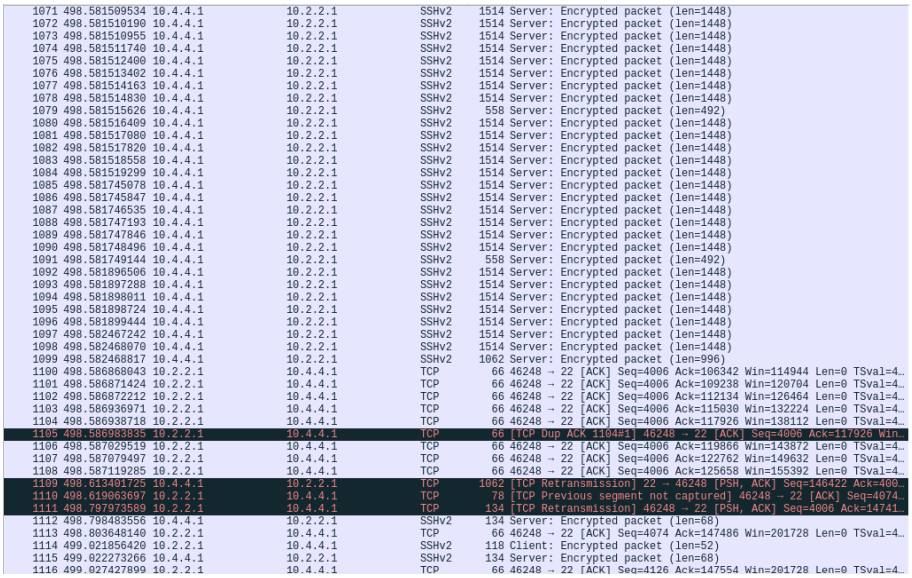


Figure 6: Wireshark file2 no PC1



674	780.435661567	10.2.2.1	10.4.4.1	TFTP	56 Read Request, File: file2, Transfer type: octet
675	780.436868539	10.4.4.1	10.2.2.1	TFTP	558 Data Packet, Block: 1
676	780.442506599	10.2.2.1	10.4.4.1	TFTP	46 Acknowledgement, Block: 1
677	780.442670762	10.4.4.1	10.2.2.1	TFTP	558 Data Packet, Block: 2
678	780.448458120	10.2.2.1	10.4.4.1	TFTP	46 Acknowledgement, Block: 2
679	780.449269053	10.4.4.1	10.2.2.1	TFTP	558 Data Packet, Block: 3
680	780.454448139	10.2.2.1	10.4.4.1	TFTP	46 Acknowledgement, Block: 3
681	780.454629664	10.4.4.1	10.2.2.1	TFTP	558 Data Packet, Block: 4
682	780.460062555	10.2.2.1	10.4.4.1	TFTP	46 Acknowledgement, Block: 4
683	780.460228590	10.4.4.1	10.2.2.1	TFTP	558 Data Packet, Block: 5
684	780.465910376	10.2.2.1	10.4.4.1	TFTP	46 Acknowledgement, Block: 5
685	780.466113010	10.4.4.1	10.2.2.1	TFTP	558 Data Packet, Block: 6
686	780.471791589	10.2.2.1	10.4.4.1	TFTP	46 Acknowledgement, Block: 6
687	780.472094684	10.4.4.1	10.2.2.1	TFTP	558 Data Packet, Block: 7
688	780.477627575	10.2.2.1	10.4.4.1	TFTP	46 Acknowledgement, Block: 7
689	780.477772259	10.4.4.1	10.2.2.1	TFTP	558 Data Packet, Block: 8
690	780.483552166	10.2.2.1	10.4.4.1	TFTP	46 Acknowledgement, Block: 8
691	780.483879593	10.4.4.1	10.2.2.1	TFTP	558 Data Packet, Block: 9
692	780.488998987	10.2.2.1	10.4.4.1	TFTP	46 Acknowledgement, Block: 9
693	780.489178642	10.4.4.1	10.2.2.1	TFTP	558 Data Packet, Block: 10
694	780.495205425	10.2.2.1	10.4.4.1	TFTP	46 Acknowledgement, Block: 10
695	780.495384661	10.4.4.1	10.2.2.1	TFTP	558 Data Packet, Block: 11
696	780.501002328	10.2.2.1	10.4.4.1	TFTP	46 Acknowledgement, Block: 11
697	780.501210186	10.4.4.1	10.2.2.1	TFTP	558 Data Packet, Block: 12
698	780.506901735	10.2.2.1	10.4.4.1	TFTP	46 Acknowledgement, Block: 12
699	780.507086152	10.4.4.1	10.2.2.1	TFTP	558 Data Packet, Block: 13
700	780.512737013	10.2.2.1	10.4.4.1	TFTP	46 Acknowledgement, Block: 13
701	780.512921362	10.4.4.1	10.2.2.1	TFTP	558 Data Packet, Block: 14

Figure 7: Wireshark tftp file2 no PC1



1071	498.581509534	10.4.4.1	10.2.2.1	SSHv2	1514 Server: Encrypted packet (len=1448)
1072	498.581510190	10.4.4.1	10.2.2.1	SSHv2	1514 Server: Encrypted packet (len=1448)
1073	498.581510955	10.4.4.1	10.2.2.1	SSHv2	1514 Server: Encrypted packet (len=1448)
1074	498.581511740	10.4.4.1	10.2.2.1	SSHv2	1514 Server: Encrypted packet (len=1448)
1075	498.581512400	10.4.4.1	10.2.2.1	SSHv2	1514 Server: Encrypted packet (len=1448)
1076	498.581513402	10.4.4.1	10.2.2.1	SSHv2	1514 Server: Encrypted packet (len=1448)
1077	498.581514163	10.4.4.1	10.2.2.1	SSHv2	1514 Server: Encrypted packet (len=1448)
1078	498.581514830	10.4.4.1	10.2.2.1	SSHv2	1514 Server: Encrypted packet (len=1448)
1079	498.581515626	10.4.4.1	10.2.2.1	SSHv2	558 Server: Encrypted packet (len=492)
1080	498.581516409	10.4.4.1	10.2.2.1	SSHv2	1514 Server: Encrypted packet (len=1448)
1081	498.581517080	10.4.4.1	10.2.2.1	SSHv2	1514 Server: Encrypted packet (len=1448)
1082	498.581517820	10.4.4.1	10.2.2.1	SSHv2	1514 Server: Encrypted packet (len=1448)
1083	498.581518558	10.4.4.1	10.2.2.1	SSHv2	1514 Server: Encrypted packet (len=1448)
1084	498.581519299	10.4.4.1	10.2.2.1	SSHv2	1514 Server: Encrypted packet (len=1448)
1085	498.581745078	10.4.4.1	10.2.2.1	SSHv2	1514 Server: Encrypted packet (len=1448)
1086	498.581745847	10.4.4.1	10.2.2.1	SSHv2	1514 Server: Encrypted packet (len=1448)
1087	498.581746535	10.4.4.1	10.2.2.1	SSHv2	1514 Server: Encrypted packet (len=1448)
1088	498.581747193	10.4.4.1	10.2.2.1	SSHv2	1514 Server: Encrypted packet (len=1448)
1089	498.581747846	10.4.4.1	10.2.2.1	SSHv2	1514 Server: Encrypted packet (len=1448)
1090	498.581748496	10.4.4.1	10.2.2.1	SSHv2	1514 Server: Encrypted packet (len=1448)
1091	498.581749144	10.4.4.1	10.2.2.1	SSHv2	558 Server: Encrypted packet (len=492)
1092	498.581896506	10.4.4.1	10.2.2.1	SSHv2	1514 Server: Encrypted packet (len=1448)
1093	498.581897288	10.4.4.1	10.2.2.1	SSHv2	1514 Server: Encrypted packet (len=1448)
1094	498.581898011	10.4.4.1	10.2.2.1	SSHv2	1514 Server: Encrypted packet (len=1448)
1095	498.581898724	10.4.4.1	10.2.2.1	SSHv2	1514 Server: Encrypted packet (len=1448)
1096	498.581899444	10.4.4.1	10.2.2.1	SSHv2	1514 Server: Encrypted packet (len=1448)
1097	498.582467242	10.4.4.1	10.2.2.1	SSHv2	1514 Server: Encrypted packet (len=1448)
1098	498.582468070	10.4.4.1	10.2.2.1	SSHv2	1514 Server: Encrypted packet (len=1448)
1099	498.582468817	10.4.4.1	10.2.2.1	SSHv2	1062 Server: Encrypted packet (len=996)
1100	498.586868043	10.2.2.1	10.4.4.1	TCP	66 46248 → 22 [ACK] Seq=4006 Ack=106342 Win=114944 Len=0 TSval=4…
1101	498.586871424	10.2.2.1	10.4.4.1	TCP	66 46248 → 22 [ACK] Seq=4006 Ack=109238 Win=120704 Len=0 TSval=4…
1102	498.586872212	10.2.2.1	10.4.4.1	TCP	66 46248 → 22 [ACK] Seq=4006 Ack=112134 Win=126464 Len=0 TSval=4…
1103	498.586936971	10.2.2.1	10.4.4.1	TCP	66 46248 → 22 [ACK] Seq=4006 Ack=115030 Win=132224 Len=0 TSval=4…
1104	498.586938718	10.2.2.1	10.4.4.1	TCP	66 46248 → 22 [ACK] Seq=4006 Ack=117926 Win=138112 Len=0 TSval=4…
1105	498.586983835	10.2.2.1	10.4.4.1	TCP	66 [TCP Dup ACK 1104#1] 46248 → 22 [ACK] Seq=4006 Ack=117926 Win…
1106	498.587029519	10.2.2.1	10.4.4.1	TCP	66 46248 → 22 [ACK] Seq=4006 Ack=119866 Win=143872 Len=0 TSval=4…
1107	498.587079497	10.2.2.1	10.4.4.1	TCP	66 46248 → 22 [ACK] Seq=4006 Ack=122762 Win=149632 Len=0 TSval=4…
1108	498.587119285	10.2.2.1	10.4.4.1	TCP	66 46248 → 22 [ACK] Seq=4006 Ack=125658 Win=155392 Len=0 TSval=4…
1109	498.613401725	10.4.4.1	10.2.2.1	TCP	1062 [TCP Retransmission] 22 → 46248 [PSH, ACK] Seq=146422 Ack=400…
1110	498.619063697	10.2.2.1	10.4.4.1	TCP	78 [TCP Previous segment not captured] 46248 → 22 [ACK] Seq=4074…
1111	498.797973589	10.2.2.1	10.4.4.1	TCP	134 [TCP Retransmission] 46248 → 22 [PSH, ACK] Seq=4006 Ack=14741…
1112	498.798483556	10.4.4.1	10.2.2.1	SSHv2	134 Server: Encrypted packet (len=68)
1113	498.803648140	10.2.2.1	10.4.4.1	TCP	66 46248 → 22 [ACK] Seq=4074 Ack=147486 Win=201728 Len=0 TSval=4…
1114	499.021856420	10.2.2.1	10.4.4.1	SSHv2	118 Client: Encrypted packet (len=52)
1115	499.022273266	10.4.4.1	10.2.2.1	SSHv2	134 Server: Encrypted packet (len=68)
1116	499.027427899	10.2.2.1	10.4.4.1	TCP	66 46248 → 22 [ACK] Seq=4126 Ack=147554 Win=201728 Len=0 TSval=4…

Figure 8: Wireshark sftp file2 no PC1

Olhando, para a figura acima, podemos notar uma duplicação de pacote e vários pacotes *Out-of-Order*. Assim, temos o mesmo pacote a ser enviado por múltiplos caminhos: é dado o SYN (*Synchronize*) da porta 46248 para a porta 22 repetidamente e acontece o mesmo nos ACKs. Desta forma, há ACKs duplicados significando a perda de pacotes e tornando a comunicação menos eficiente. Tal como referimos anteriormente, reparamos que na realização do comando PING vizualisamos esta tal perda de pacotes.

```
ftp> get file1
local: file1 remote: file1
200 PORT command successful. Consider using PASV.
150 Opening BINARY mode data connection for file1 (224 bytes).
226 Transfer complete.
224 bytes received in 0.00 secs (465.4255 kB/s)
ftp>
```

Figure 9: ftp Portatil1

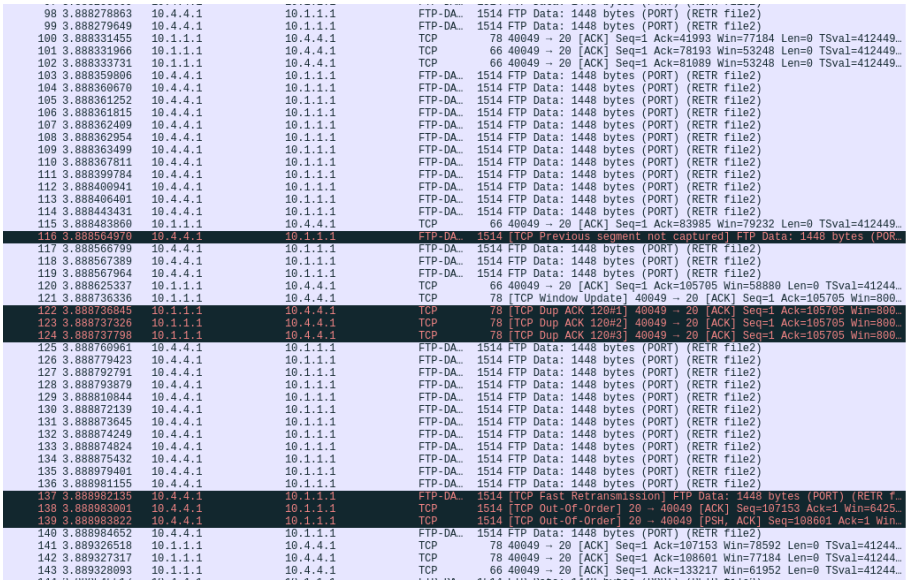


Figure 10: Wireshark ftp file2 no Portatil1

2.1.2 Questão 2:

Obtenha a partir do wireshark, ou desenhe manualmente, um diagrama temporal para a transferência de file1 por FTP. Foque-se apenas na transferência de dados [ftp-data] e não na conexão de controlo, pois 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 quer nos dados como nas confirmações.

O FTP emprega dois números de porta TCP amplamente reconhecidos: a porta 20 e a porta 21. A primeira porta é utilizada para a transferência de dados, como o envio de arquivos, enquanto que a segunda porta é destinada ao controlo da conexão, ao atuar como a porta de comando.

Para que uma transferência usando FTP ocorra entre o servidor e cliente, numa primeira fase, o cliente deve pedir acesso ao servidor. Assim, esse autentica-se com um user e uma password para efeitos de segurança. De seguida, inicia-se a fase inicial de conexão. O cliente envia um TCP SYN ao servidor, que lhe responde com um TCP SYN/ACK. De seguida, o cliente envia um TCP ACK. Fica a conexão iniciada e o ficheiro pode ser transmitido. Quando a transferência do ficheiro termina, é enviada uma mensagem de "transferência completa" para o cliente. Inicia-se a fase final da conexão, isto é, o servidor utiliza um TCP FIN para indicar que o ficheiro foi transferido e para informar que a conexão vai terminar. Finalmente, o cliente responde com um TCP ACK. O servidor termina a sua conexão, no entanto, o cliente deve fazer o mesmo, sendo que é enviado um TCP FIN, ao qual o servidor responde com um TCP ACK. Termina assim a ligação.

A figura 5 (da questão 1) ilustra as diferentes fases do processo, tanto em relação ao pedido de autenticação ao servidor, como à transferência completa.

Para melhor visualização das diferentes fases, tanto em relação ao pedido de autenticação ao servidor, como à transferência completa. A imagem abaixo já tem aplicado o filtro **tcp.port == 20**, já que o solicitado foi focar na transferência de dados.

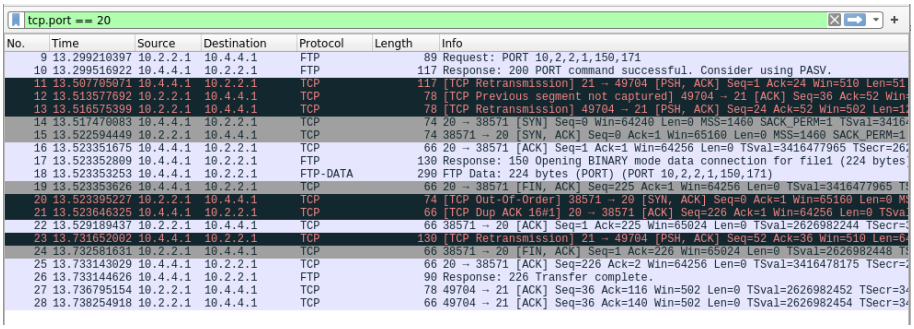


Figure 11: Tráfego com aplicação do filtro à porta 20

Na figura acima conseguimos observar um exemplo de uma retransmissão de um pacote que foi detetada. Como o pacote deve ser reenviado por se estar a utilizar o protocolo TCP, as perdas e duplicações acabam por serem facilmente detetadas.

Numa lógica de servidor-cliente, o que aconteceu foi que o servidor (10.4.4.1) enviou dados para o respetivo cliente (10.2.2.1), mas o cliente não deu uma resposta, ACK, no tempo indicado de retransmissão (Retransmission Timer for Server). Com isso, o servidor pensa que o pacote não chegou ao destino, reenviando-o.

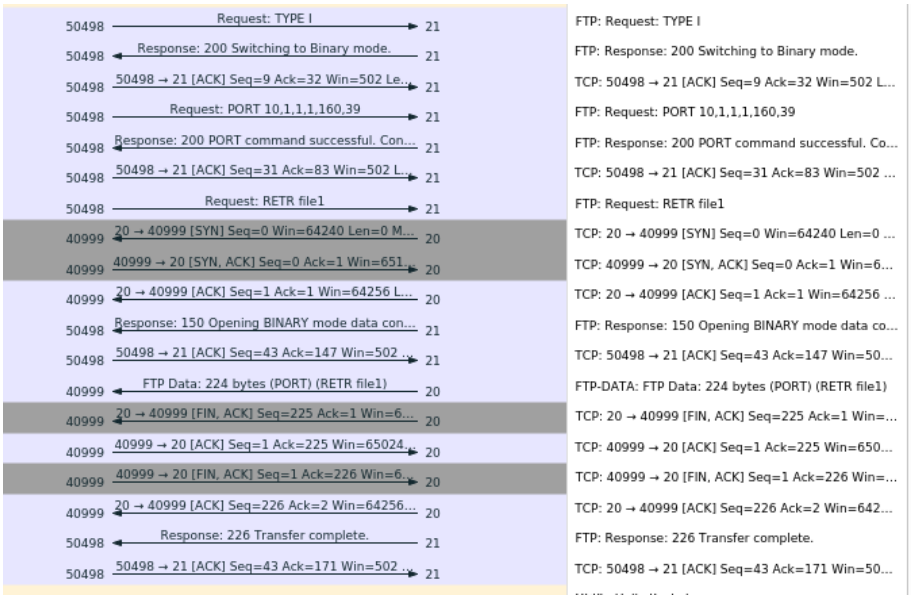


Figure 12: Diagrama temporal da transferência no Portátil1

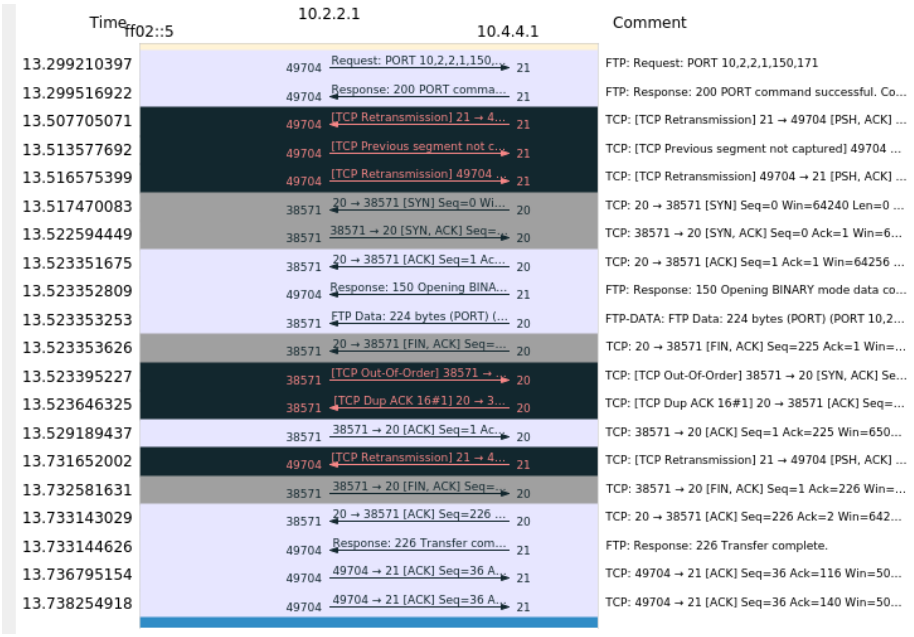


Figure 13: Diagrama temporal da transferência no PC1

2.1.3 Questão 3:

Obtenha a partir do wireshark, ou desenhe manualmente, um diagrama temporal para a transferência de file1 por TFTP. 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 quer nos dados como nas confirmações.

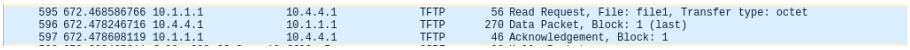


Figure 14: Wireshark tftp da transferência no Portátil1

É importante referir que o protocolo UDP não tem número de segmento, nem ACKs, pois estes só existem a nível aplicacional. Na figura acima notamos que, numa primeira fase, o cliente envia um pedido de leitura (*Read Request*) ao servidor, informando-o que pretende transferir o ficheiro. De seguida, o servidor dá inicio à transferência, enviando o *Data Packet* especificado. Finalmente, o cliente envia um ACK, para informar o servidor que o *packet* anterior foi recebido com sucesso.

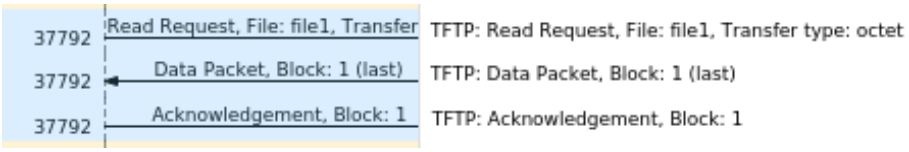


Figure 15: Diagrama temporal da transferência no Portátil1

2.1.4 Questão 4:

Compare sucintamente as quatro aplicações de transferência de ficheiros que usou nos seguintes pontos (i) uso da camada de transporte; (ii) eficiência; (iii) complexidade; (iv) segurança;

- SFTP : A camada de transporte utilizada é o TCP. Em relação à eficiência e transferência de dados, podemos dizer que é semelhante ao FTP, porém os dados são encriptados, tornando-o mais seguro. Quanto à complexidade, é uma opção fiável gestão de acessos, transferência e gestão de dados. Sendo bastante complexo. Em termos de segurança, utiliza o SSH, garantindo segurança por meio de uma camada de transporte, autenticação e conexão. A camada de transporte é executada com o auxílio do protocolo TCP/IP, fornecendo encriptação, autenticação do servidor e proteção da integridade dos dados. A camada de autenticação é responsável por lidar com a autenticação dos clientes.

- FTP : A camada de transporte utilizada é o TCP. Em relação à eficiência e transferência, é importante observar que os dados não são encriptados, o que torna a transmissão vulnerável à captura por terceiros. No entanto, o protocolo TCP é utilizado para assegurar a transmissão correta dos dados. Quanto à complexidade, esta camada permite a transferência de ficheiros em paralelo. Criando uma conexão, resultando em várias velocidades de transferência, tornando-o num sistema bastante complexo. Por fim, em relação à segurança, essa camada não oferece mecanismos de autenticação, o que significa que qualquer pessoa pode capturar pacotes de dados sem restrições.

- TFTP : A camada de transporte utiliza o UDP. Em relação à eficiência e transferência, podemos dizer que é menos confiável, uma vez que utiliza o protocolo UDP, que não envia mensagens de confirmação (*Acknowledgement*) para garantir que os dados foram entregues com sucesso. Quanto à complexidade, devido ao uso do protocolo UDP, a camada é muito simplificada e oferece menos funcionalidades em comparação com outros protocolos. Por fim, em termos de segurança, segue uma abordagem semelhante ao FTP, pois representa uma versão mais simples e não inclui mecanismos avançados de segurança.

- HTTP : A camada de transporte utiliza o protocolo UDP. Em relação à eficiência e transferência, é possível transmitir vários pacotes simultaneamente, o que pode aumentar significativamente a taxa de transferência de dados. É usado o protocolo TCP, o que conduz a um aumento do débito de informação e da fiabilidade da mesma. Sobre a complexidade, é garantida confiança, escalabilidade, e são suportadas as redes heterogéneas e não confiáveis. Por fim, relativamente a segurança, a informação está guardada de uma forma não encriptada, sendo mais fácil de ler e manipular por terceiros.

2.2 Uso da camada de transporte por parte das aplicações (Parte II)

2.2.1 Questão 1:

Com base no trabalho realizado, tanto na parte I como na parte II, identifique para cada aplicação executada, qual o protocolo de aplicação, o protocolo de transporte, porta de atendimento e overhead de transporte.

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)
wget, lynx ou via browser	DNS	TCP	80	11,05
ssh, sftp	SSH/SFTP	TCP	22	32,78
ftp	FTP	TCP	21	76,92
Tftp	TFTP	UDP	69	50
Telnet	Telnet	TCP	23	42,55
nslookup ou dig	DNS	UDP	53	16
Ping	*	*	*	*
Traceroute	DNS	UDP	53	15,38

* Como o PING utiliza o protocolo ICMP, não serão retiradas informações.

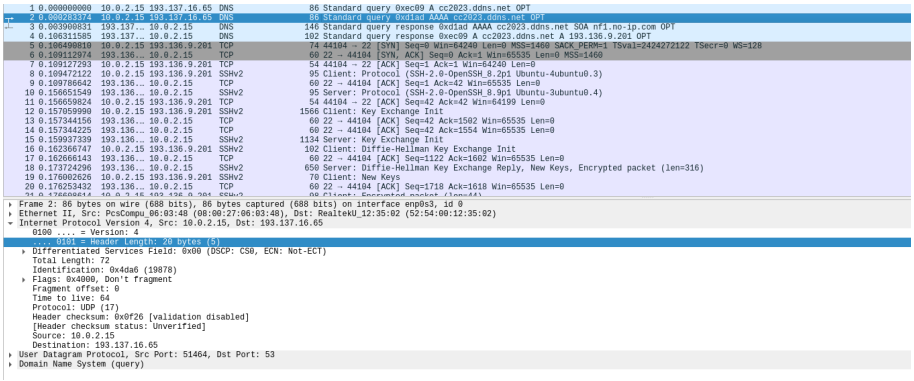


Figure 16: Informação sobre o comando Traceroute

Traceroute: Observando a figura, em *User Datagram Protocol*, reparamos que a porta de atendimento, ou seja, porta de destino é a 53 e que o *Total Lenght* é 72. Assim sendo, o *overhead* é dado por 8/(72 - 20) * 100 = 15,38.

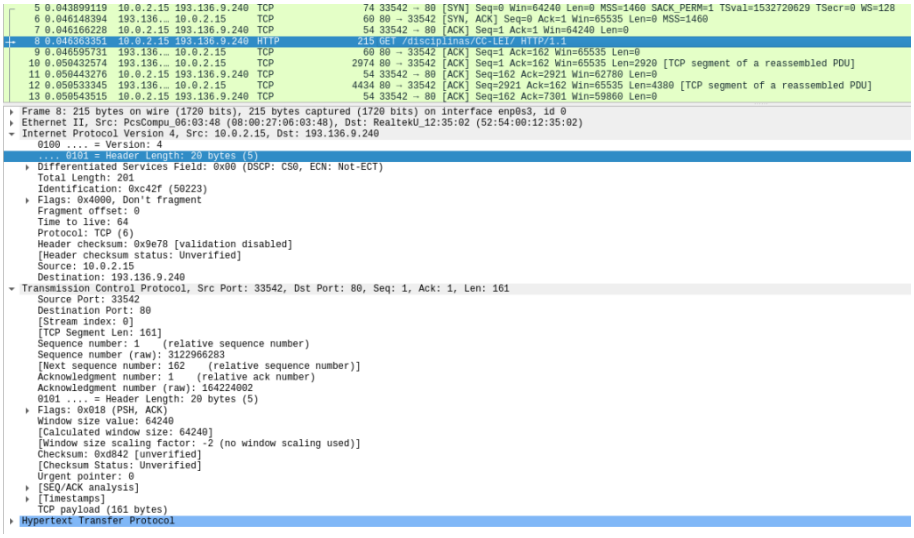


Figure 17: Informação sobre o comando wget

Browser/HTTP: Observando a figura, em *User Datagram Protocol*, reparamos que a porta de atendimento, ou seja, porta de destino é a 80 e que o *Total Lenght* é 201. Assim sendo, o *overhead* é dado por 20/(201 - 20) * 100 = 11,05

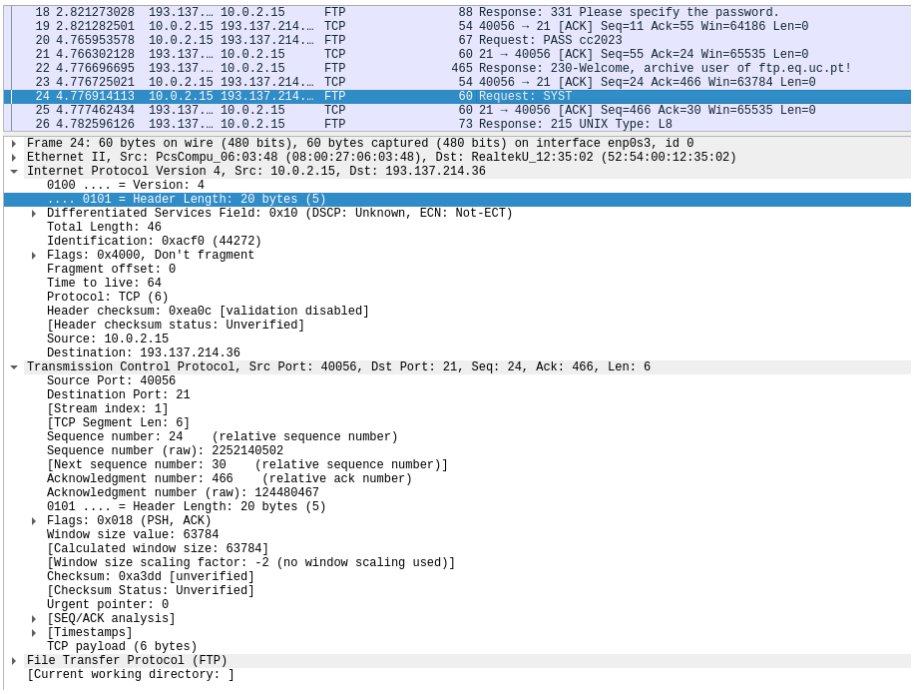


Figure 18: Informação sobre o comando FTP

FTP: Observando a figura, em *Transmission Control Protocol*, reparamos que a porta de atendimento, ou seja, porta de destino é a 21 e que o Total *Lenght* é 46. A porta de origem é a 21, que é normalmente a utilizada por este protocolo. Assim sendo, o *overhead* é dado por 20/(46 - 20) * 100 = 76,92.

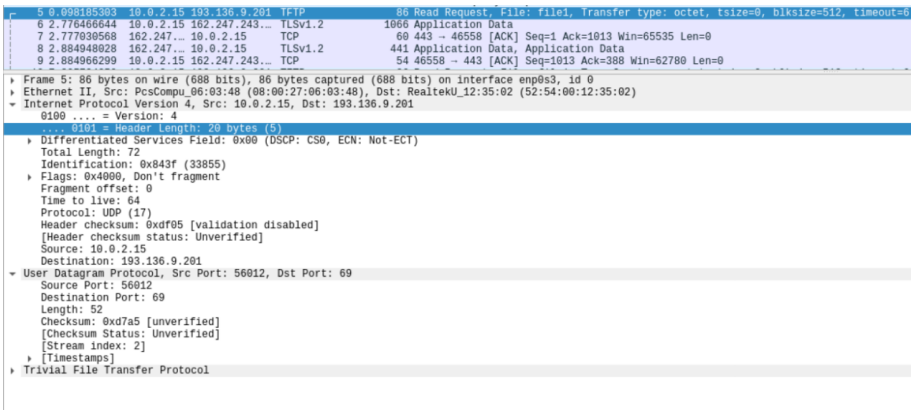


Figure 19: Informação sobre o comando TFTP

TFTP: Observando a figura, em *Transmission Control Protocol*, reparamos que a porta de atendimento, ou seja, porta de destino é a 69 e que o Total *Lenght* é 72. Assim sendo, *overhead* é dado por 8/(72 - 20) * 100 = 50.

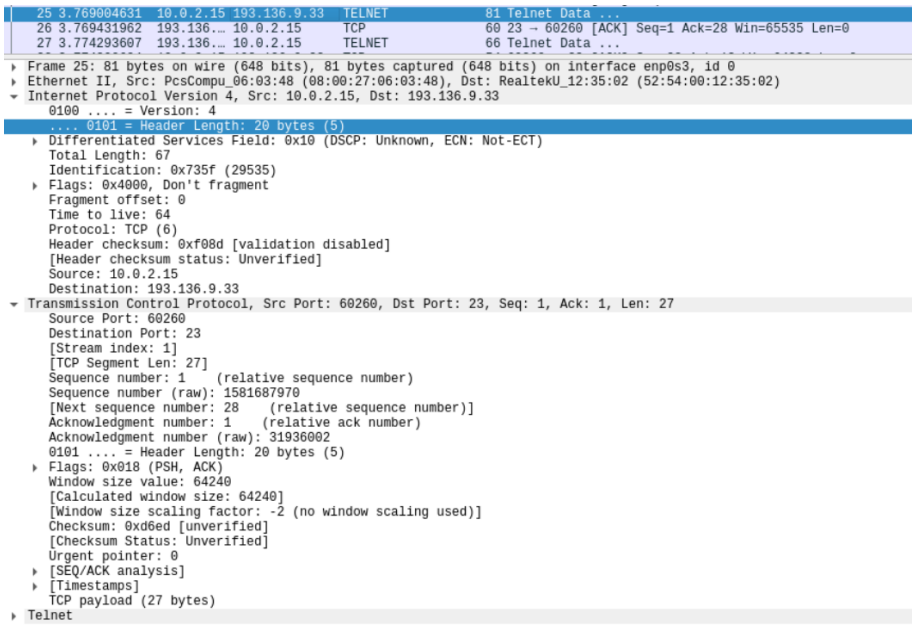


Figure 20: Informação sobre o comando Telnet

Telnet: Observando a figura, em *Transmission Control Protocol*, reparamos que a porta de atendimento, ou seja, porta de destino é a 23 e que o Total *Lenght* é 67. Assim sendo, o *overhead* é dado por 20/(67 - 20) * 100 = 42,55.

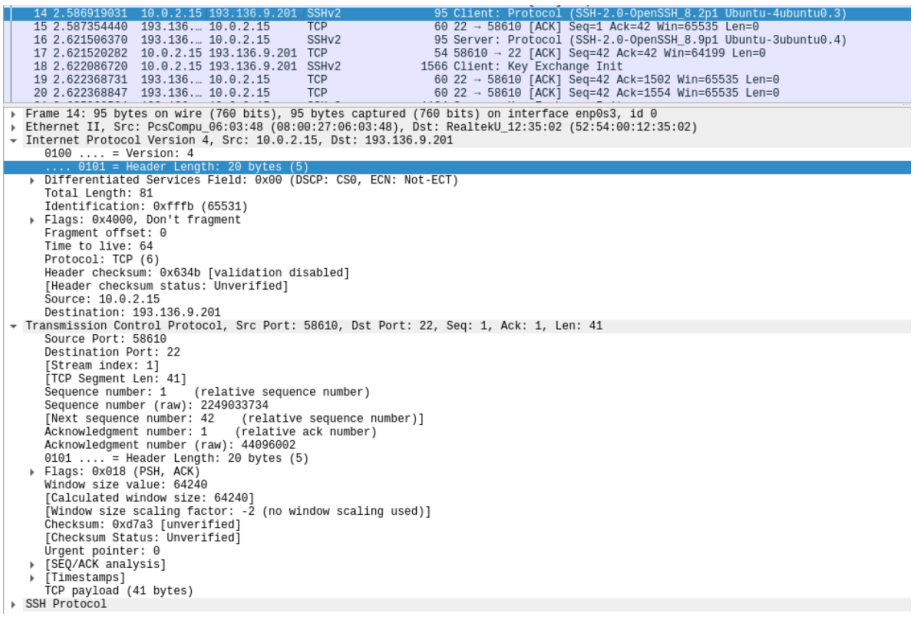


Figure 21: Informação sobre o comando SSH

SSH: Observando a figura, em *Transmission Control Protocol*, reparamos que a porta de atendimento, ou seja, porta de destino é a 22 e que o Total *Lenght* é 81. Assim sendo, o *overhead* é dado por 20/(81 - 20) * 100 = 32,78.

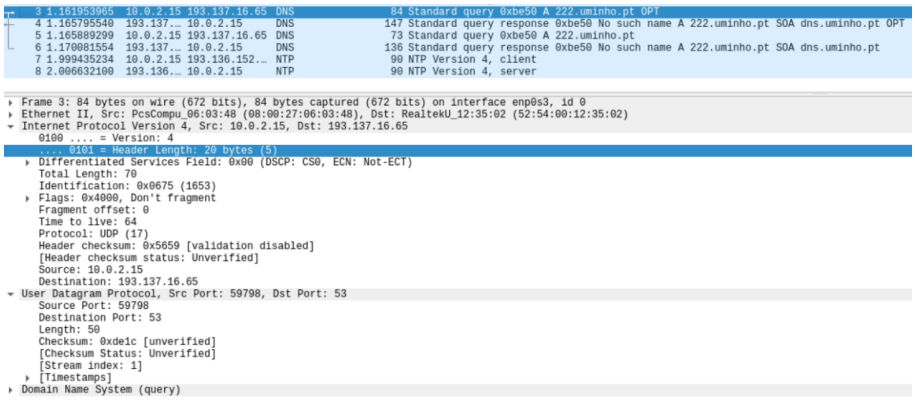


Figure 22: Informação sobre o comando SSH

Nslookup: Observando a figura, em *Transmission Control Protocol*, reparamos que a porta de atendimento, ou seja, porta de destino é a 53 e que o Total *Lenght* é 70. Assim sendo, o *overhead* é dado por 8/(70 - 20) * 100 = 16.