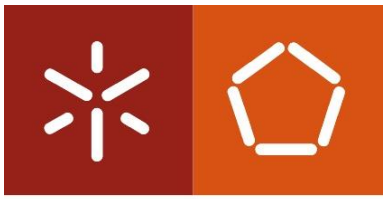


Comunicações por computador

Grupo 5 - PL1

16 de março de 2021

TP1: Protocolos da Camada de Transporte

Luís Martins (A89600)

Renata Teixeira (A89611)

Tiago Alves (A89554)

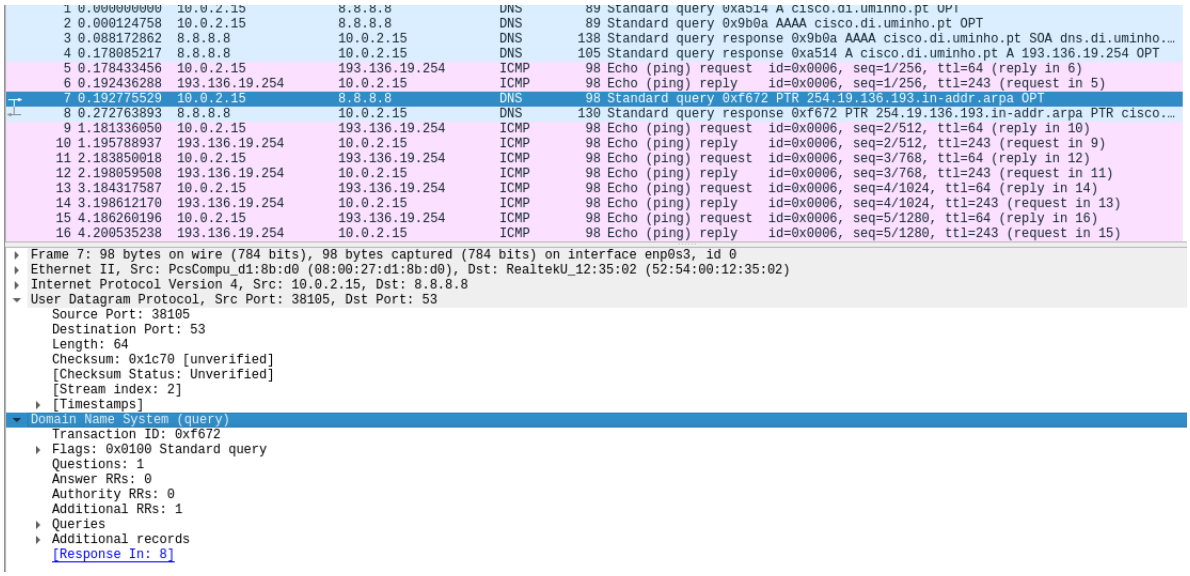
1 Perguntas

1.1 Pergunta 1

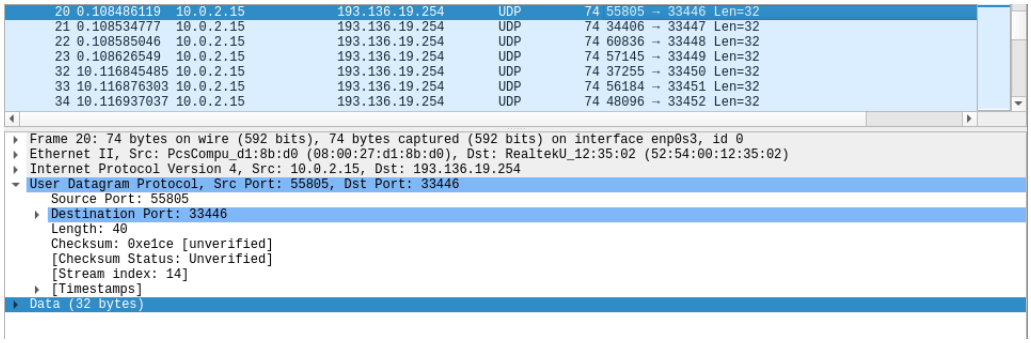
Inclua no relatório uma tabela em que identifique, para cada comando executado, qual o protocolo de aplicação, o protocolo de transporte, porta de atendimento e overhead de transporte, como ilustrado no exemplo seguinte:

Comando usado (se aplicável)	Protocolo de Aplicação (se aplicável)	Protocolo de Transporte (se aplicável)	Porta de Atendimento (se aplicável)	Overhead de Transporte em Bytes (se aplicável)
ping	DNS	UDP	53	8
traceroute	TRACEROUTE	UDP	33446	8
telnet	TELNET	TCP	23	20
ftp	FTP	TCP	21	20
tftp	TFTP	UDP	69	8
browser / http	HTTP	TCP	80	20
nslookup	DNS	UDP	53	8
ssh	SSH	TCP	22	20

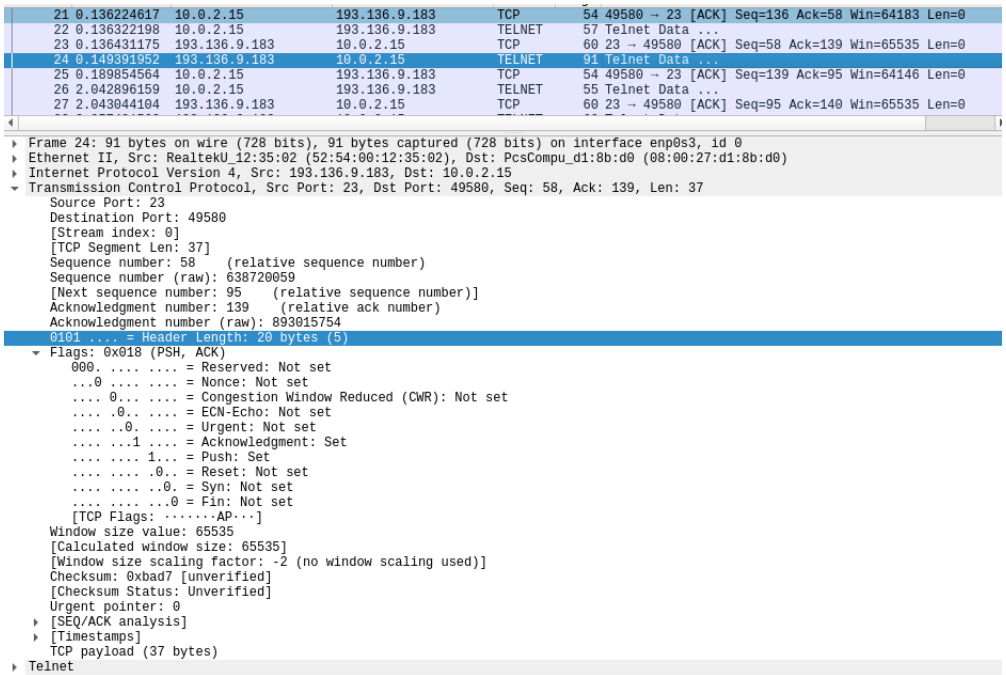
- **ping**: No print que se segue, podemos ver que se usa UDP (User Datagram Protocol), que o protocolo da aplicação é DNS e que a porta de atendimento é a 64. Visto que o protocolo de transporte é UDP, podemos concluir que o overhead tem 8 bytes de tamanho.



- **traceroute**: Tal como no comando *ping*, podemos ver, pelo print abaixo, que o comando *traceroute* também utiliza como Protocolo de transporte o UDP, tendo, por isso, o overhead tamanho 8. Também podemos ver pela figura que, para o pacote que escolhemos, o destination port é o 33446.

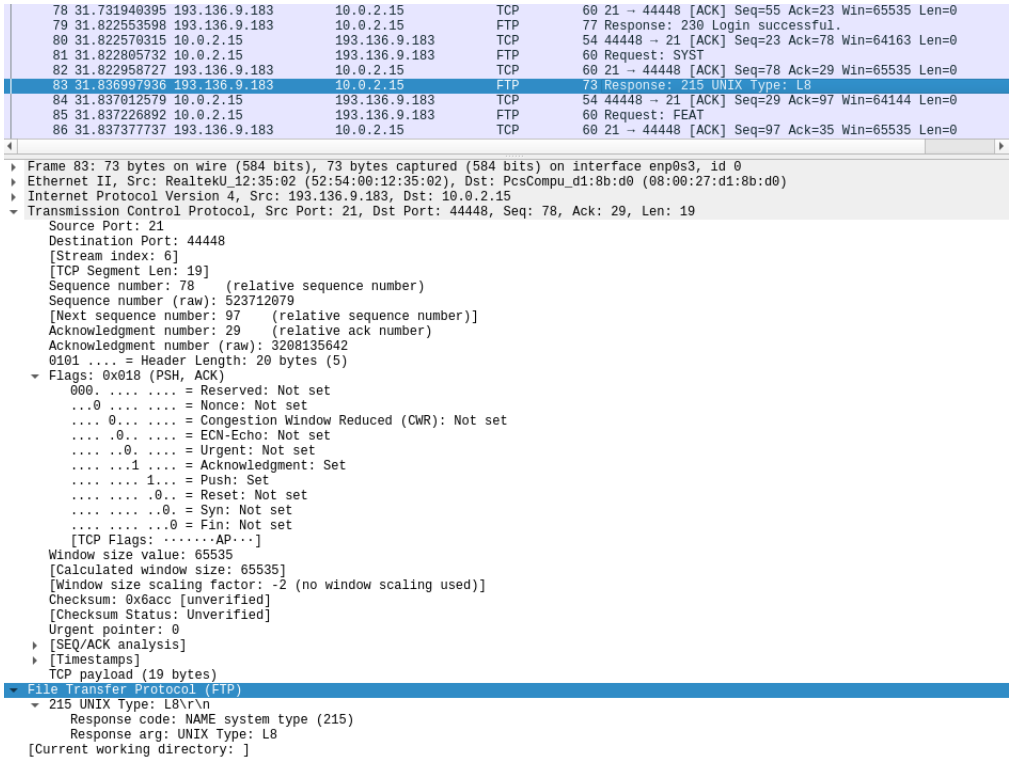


- **telnet**: Como podemos ver pelo print, o comando *telnet* tem como protocolo de transporte o TCP (Transmission Control Protocol), neste caso, a porta de atendimento é a 23 e que o *overhead*(header length) é 20 bytes.



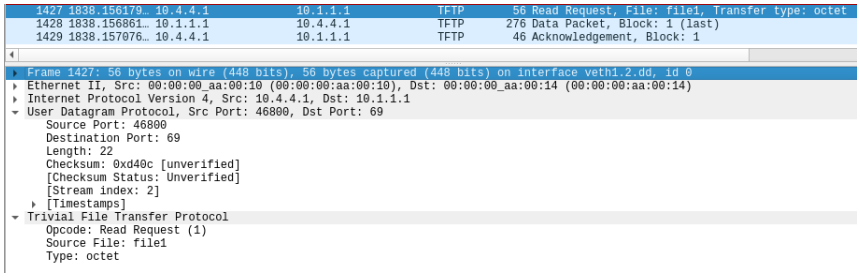
- **ftp**: Como podemos ver pelo print, o comando *ftp* tem como protocolo de transporte

o TCP (Transmission Control Protocol), neste caso a porta de atendimento é a 21 e que o *overhead*(header length) é 20 bytes.

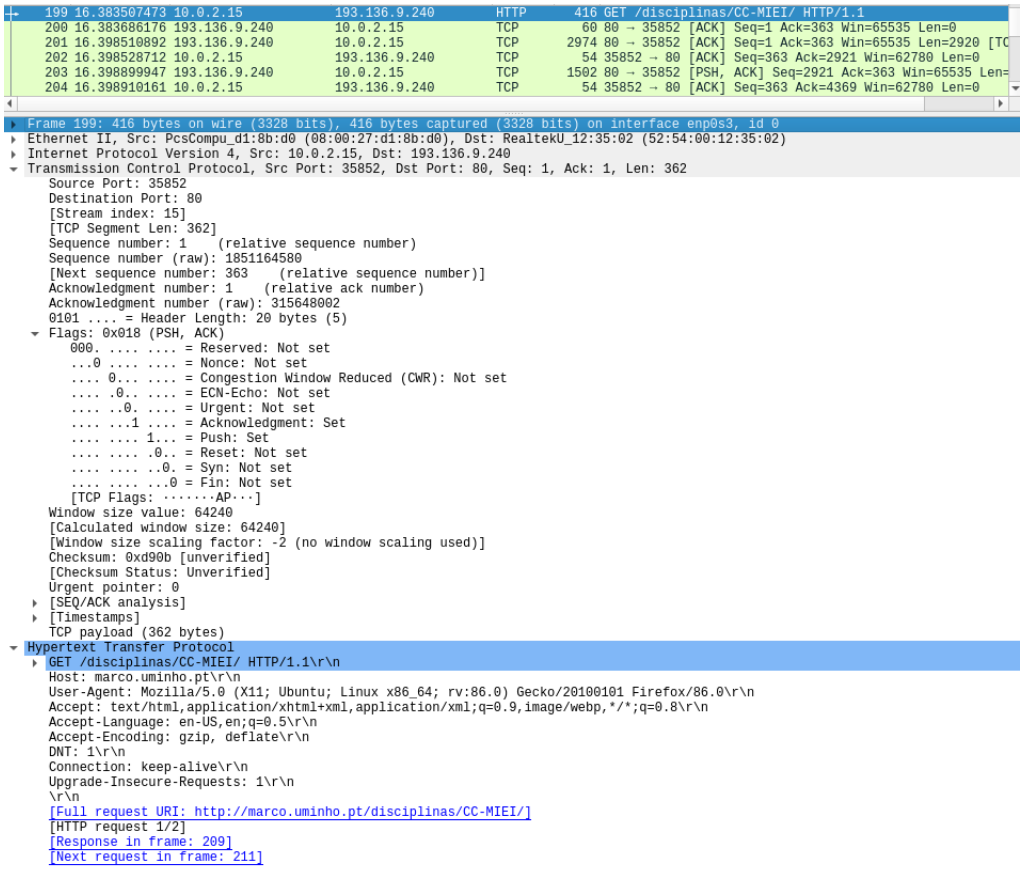


- **tftp**: Como podemos verificar na figura seguinte, o comando *tftp* tem como protocolo de transporte o UDP, por isso tem overhead de 8 bytes e também podemos ver que a porta de atendimento é a 69.

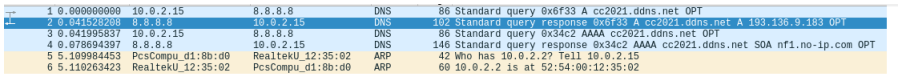
  Como estavamos a ter problemas com o uso deste protocolo na máquina virtual fornecida, decidimos, então, testar o *tftp* na topologia core.

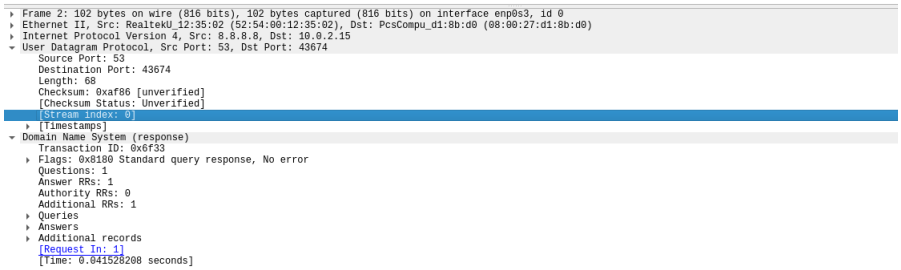


- **browser/http**: Através da seguinte figura, podemos observar que o comando *http* tem como protocolo de transporte o TCP. Neste caso, a porta de atendimento é a 80 e o overhead de transporte é de 20 bytes.



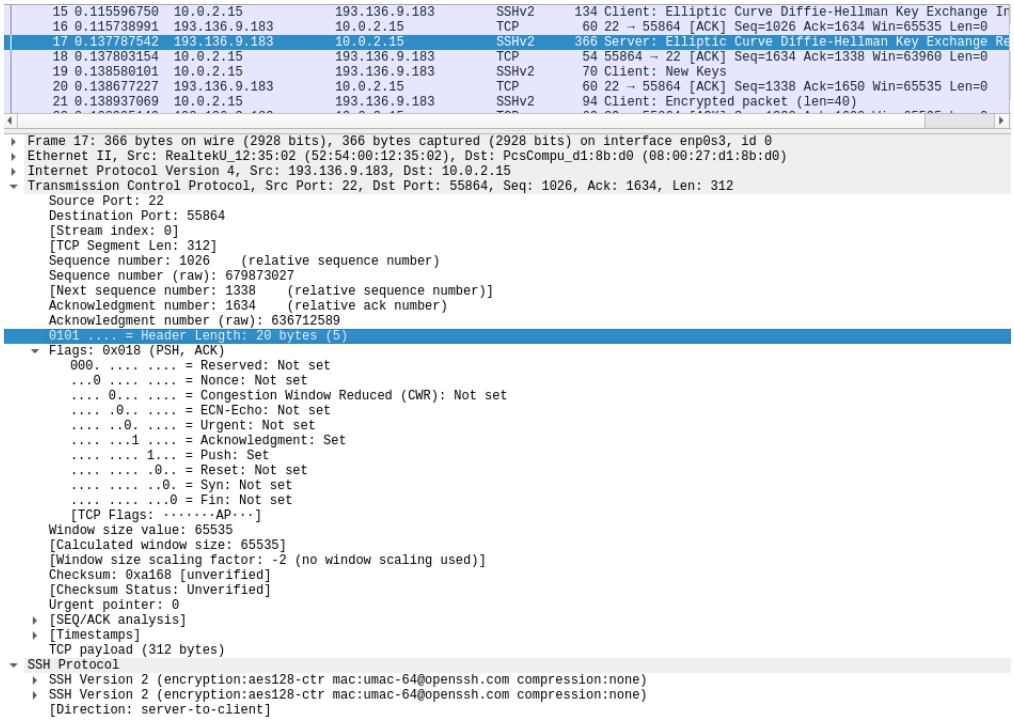
- **nslookup**: A partir da análise da figura seguinte, podemos verificar que o comando *nslookup* tem como protocolo de aplicação o DNS, como protocolo de transporte o UDP, ou seja 8 bytes de overhead de transporte. E a porta de atendimento é a 53.





- **ssh**: De acordo com a seguinte figura, observamos que o comando *ssh* tem como protocolo de transporte o TCP, tem 20 bytes de overhead de transporte e a porta de

atendimento é a 22.



Posto isto, consideramos ainda importante mencionar algumas diferenças entre o TCP e o UDP. Como já referimos anteriormente, o tamanho do cabeçalho do protocolo TCP é de 20 bytes enquanto o do UDP é de apenas 8 bytes. Para além disso, ao contrário do protocolo UDP, o protocolo TCP é orientado à conexão.

O protocolo TCP é capaz de fazer a deteção e correção de erros, fazendo com que os pacotes com erros sejam retransmitidos. Por outro lado, o protocolo UDP é capaz de detetar erros mas não os tenta corrigir, simplesmente descarta os pacotes que tenham erros.

No que toca à velocidade de transferência, o UDP é mais eficiente que o TCP, uma vez que não efetua correção de erros.

1.2 Pergunta 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.

- **FTP**

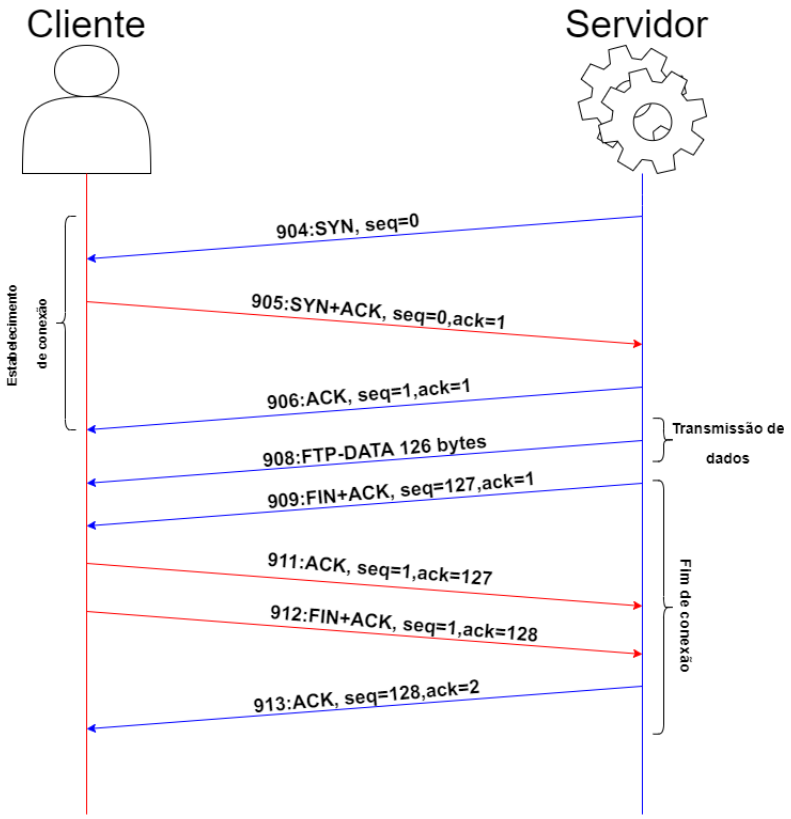


Figura 1: Diagrama Temporal FTP

Através da análise dos vários pacotes de dados, podemos concluir que o diagrama temporal da transferência seria o da seguinte figura. Mediante a sua análise, é possível afirmar que a conexão se inicia com um envio por parte do servidor de um pacote SYN, com número de sequência de valor 0. Seguidamente, o cliente responde com um segmento SYN + ACK, de forma a confirmar o sucesso da receção do segmento TCP anterior. Em resposta ao SYN anteriormente enviado pelo cliente, é encaminhada pelo servidor uma trama ACK, seguida dos dados pretendidos. O servidor envia, posteriormente, um segmento FIN + ACK que comunica a conclusão do envio dos dados. Em seguida, o cliente responde com uma trama ACK e com um pacote FIN + ACK. Por fim, o servidor informa o sucesso no recebimento do segmento TCP FIN, enviando uma trama ACK.

Na figura 2, podemos ver todos os segmentos representados no esquema representado na figura 1. Foram filtrados, no Wireshark, os segmentos correspondentes à conexão de dados.

904	1046.731124…	10.1.1.1	10.4.4.1	TCP	74 20 → 47685 [SYN] Seq=0 Win=64240 Len=0 MSS=1460
905	1046.731267…	10.4.4.1	10.1.1.1	TCP	74 47685 → 20 [SYN, ACK] Seq=0 Ack=1 Win=65160 Len
906	1046.731391…	10.1.1.1	10.4.4.1	TCP	66 20 → 47685 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TS
907	1046.731456…	10.1.1.1	10.4.4.1	FTP	105 Response: 150 Here comes the directory listing.
908	1046.731564…	10.1.1.1	10.4.4.1	FTP-DA…	192 FTP Data: 126 bytes (PORT) (LIST)
909	1046.731565…	10.1.1.1	10.4.4.1	TCP	66 20 → 47685 [FIN, ACK] Seq=127 Ack=1 Win=64256 L
910	1046.731589…	10.4.4.1	10.1.1.1	TCP	66 37104 → 21 [ACK] Seq=81 Ack=187 Win=64256 Len=0
911	1046.731887…	10.4.4.1	10.1.1.1	TCP	66 47685 → 20 [ACK] Seq=1 Ack=127 Win=65152 Len=0
912	1046.731887…	10.4.4.1	10.1.1.1	TCP	66 47685 → 20 [FIN, ACK] Seq=1 Ack=128 Win=65152 L
913	1046.732021…	10.1.1.1	10.4.4.1	TCP	66 20 → 47685 [ACK] Seq=128 Ack=2 Win=64256 Len=0
914	1046.732052…	10.1.1.1	10.4.4.1	FTP	90 Response: 226 Directory send OK.

Figura 2: Segmento de Wireshark da transferência FTP

- **TFTP**

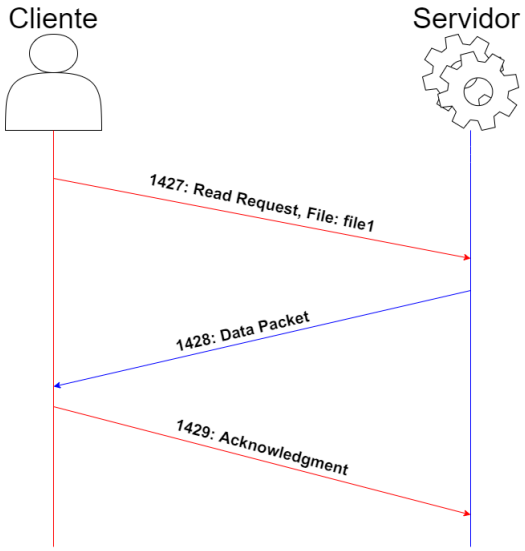


Figura 3: Diagrama Temporal TFTP

De acordo com a figura acima, podemos observar que, primeiramente, o cliente envia um Read Request ao servidor. De seguida, o servidor envia ao cliente um Data Packet com os dados que pretende enviar, no caso dos dados serem maiores do que 512 bytes, são enviados em vários pacotes distintos com esse tamanho máximo. Por último, o cliente, aquando da receção dos dados, envia uma trama ACK ao servidor, confirmando o sucesso da receção dos dados.

Podemos comprovar este comportamento, retirando do Wireshark a seguinte imagem:

1427	1838.156179…	10.4.4.1	10.1.1.1	TFTP	56 Read Request, File: file1, Transfer type: octet
1428	1838.156861…	10.1.1.1	10.4.4.1	TFTP	276 Data Packet, Block: 1 (last)
1429	1838.157076…	10.4.4.1	10.1.1.1	TFTP	46 Acknowledgement, Block: 1

1.3 Pergunta 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) eficiencia na transferência; (iii) complexidade; (iv) segurança;

(i) **Uso da camada de transporte**

- **SFTP**: Utiliza o protocolo TCP
- **FTP**: Utiliza o protocolo TCP
- **TFTP**: Utiliza o protocolo UDP
- **HTTP**: Utiliza o protocolo TCP

(ii) **Eficiência na transferência**

- **SFTP**: Idêntico ao FTP, mas os dados são encriptados.
- **FTP**: Visto que usa o protocolo TCP, é garantido, através do uso de *acknowledges*, que o segmento vai ser transmitido, porém há uma perda de eficiência, visto que é necessário esperar pelo *acknowledge* para continuar.
- **TFTP**: Devido ao uso do protocolo UDP por parte desta aplicação, esta torna-se menos viável. Este protocolo perde viabilidade pois não usa *acknowledgements*, não sendo, por essa razão, possível averiguar se o pacote foi entregue com sucesso, sendo, por vezes, necessário a retransmissão dos mesmos. No entanto, em caso de sucesso, este é mais rápido que o FTP.
- **HTTP**: permite que vários HTTP requests sejam enviados numa única ligação TPC sem que seja necessário esperar pelas respostas correspondentes.

(iii) **Complexidade**

- **SFTP**: Uma vez que o protocolo SFTP é muito fiável e possibilita o acesso, transferência e gestão de dados, e visto que estas funcionalidades apresentam custos elevados de processamento, este protocolo revela-se bastante complexo.
- **FTP**: Dado que o protocolo FTP é capaz de suportar múltiplos pedidos de transferência de dados concorrentemente em que realiza uma nova conexão para cada uma das transferências, têm de existir diferentes velocidades de transferência. A elevada frequência de novas conexões torna este protocolo bastante complexo.
- **TFTP**: Através do nome deste protocolo TFTP (Trivial File Transfer Protocol), podemos perceber que este é uma alternativa simplificada do protocolo anterior FTP(File Transfer Protocol). O protocolo TFTP, para além de ser mais simples, suporta muito menos funcionalidades do que a versão com maior complexidade. Para além disto, juntando também o facto deste protocolo ser baseado em UDP e não em TCP, permite-nos concluir que este protocolo não é muito complexo.

- **HTTP** Este protocolo tem sido usado pela WWW desde 1990. A primeira versão de HTTP, chamada HTTP/0.9, era um protocolo simples para a transferência de dados no formato de texto ASCII pela Internet Mesmo com mais complexidade introduzida no HTTP/2.0 por encapsular mensagens HTTP em quadros (frames), o HTTP foi projetado para ser simples e legível às pessoas. Sendo, neste momento, um protocolo de complexidade moderada.

(iv) **Segurança**

- **SFTP:** O SFTP (assemelhando-se ao protocolo FTPS) oferece uma proteção extra aos arquivos e alterações feitas na hospedagem. No entanto, o SFTP utiliza-se da tecnologia SSH (Secure Shell) para autenticar o contacto e estabelecer conexões seguras entre as máquinas. O SSH usa uma arquitetura em camadas, em termos de segurança a camada de transporte fornece encriptação de dados e autenticação do servidor e a camada de autenticação é responsável por manusear a autenticação do utilizador, logo, afirma-se que este protocolo consegue garantir uma elevada segurança.
- **FTP:** Este protocolo utiliza autenticação, não proporcionando encriptação de dados, tornando-o suscetível a ter bastantes falhas na segurança. Desta forma, e devido às transmissões não serem encriptadas, este protocolo é extremamente inseguro.
- **TFTP:** Protocolo que não fornece autenticação; como não protege os dados a serem transferidos, é considerado relativamente inseguro.
- **HTTP:** Protocolo da camada de aplicação que não é encriptado, tendo a informação representada em texto. Por esta razão, embora utilize autenticação, é vulnerável a adúlteros dos dados, não garantindo segurança a esse nível.

1.4 Pergunta 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).

Ao contrário da situação decorrida na LAN4 em que não houve qualquer problema a nível de ligações de rede, na LAN3, também alusiva à topologia dada no enunciado, repara-se que se sucedeu perda e duplicação de pacotes IP a níveis de Transporte e Aplicação. Durante as experiências feitas nesta LAN, foram usados dois protocolos de aplicação distintos - TFTP e FTP. Tendo em conta que, numa transferência de dados, o protocolo de transporte usado está relacionado com o protocolo de aplicação, conseguimos, a partir dos resultados obtidos, comparar o UDP e TCP, respetivamente.

Através da realização deste projeto e de acordo com o conteúdo estudado nas aulas teóricas, podemos afirmar que o protocolo TCP é capaz de detetar e corrigir possíveis erros, realizando, nestes casos, uma retransmissão do pacote.

De outra forma, o protocolo UDP, apesar de, igualmente, conseguir detetar erros, não tem capacidade de os corrigir, descartando os pacotes nessas situações. Sendo impossível a deteção das perdas de pacotes.

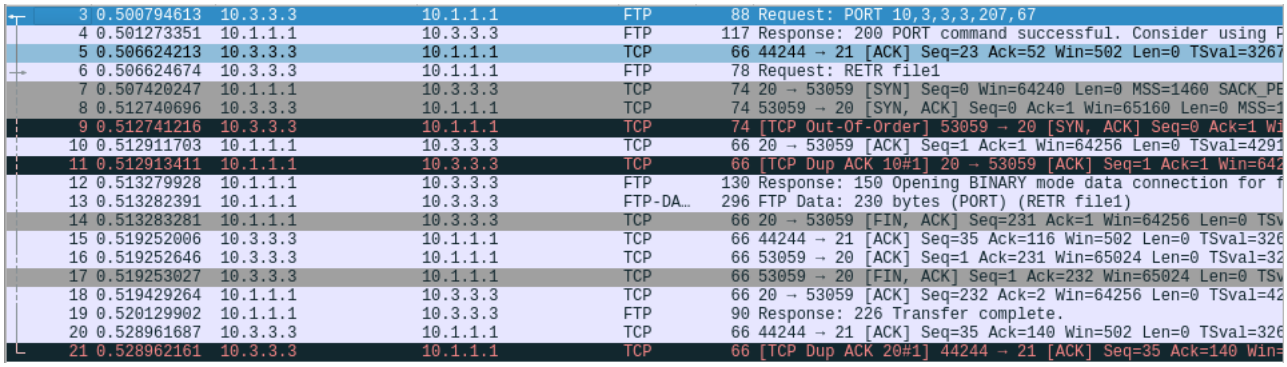


Figura 4: Computador Corvo a fazer download usando o FTP

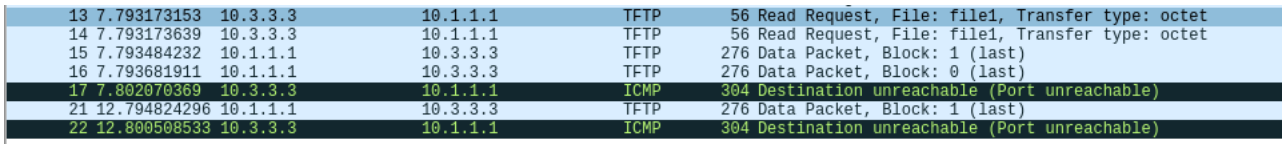


Figura 5: Computador Corvo a fazer download usando o TFTP

Nas figuras anteriores podemos ver o download do ficheiro "file1", usando num dos casos o protocolo FTP e no outro o protocolo TFTP. Visto que na LAN3 existe apenas 5% de package loss e 10% de package duplication, foi necessário fazer vários requests, uma vez que a maior parte deles foram feitos com sucesso. No entanto, após várias tentativas foram encontrados erros em algumas transmissões.

No caso do FTP verificamos que haviam segmentos de Acknowledgment que apareciam repetidos, para além disso haviam segmentos SYN + ACK que chegavam fora de ordem ao destino.

No caso do TFTP, por vezes o ficheiro não chegava ao seu destino final e o host demorava bastante tempo até notar que algo de errado tinha sucedido.

Por outro lado, nos casos em que não aconteciam erros, as transferências em que o protocolo FTP era usado eram mais lentas comparando com o uso do TFTP.

Deste modo, concluímos que, se a velocidade de transmissão for a prioridade, é recomendado o uso do protocolo TFTP; por outro lado, se a prioridade for o ficheiro efetivamente chegar ao seu destino, é aconselhado o uso do protocolo FTP, uma vez que garante a transferência de todos os dados.

2 Conclusão

Neste trabalho foram abordados e explorados diversos Protocolos de Camada de Aplicação e de Transporte. Com recurso à máquina virtual xubuncore e à ferramenta Wireshark, foram realizados vários testes de forma a contemplar e acompanhar o processo de transferência de dados, com o objetivo de se estudar os protocolos, e as vantagens e desvantagens da sua utilização.

Baseada nos resultados, é possível obter a conclusão de que, na eventualidade de uma transferência de dados, ambos os protocolos de transporte têm vantagens a serem tidas em conta, dependendo das prioridades do envio. No caso de se desejar uma transferência de alta fiabilidade, com a garantia de que os dados enviados sejam, de facto, recebidos, devemos optar por uma aplicação que recorra ao protocolo TCP para efetuar o transporte, consentindo com uma menor eficiência.

De outro modo, se uma maior velocidade de transferência for uma prioridade, negligenciando possíveis perdas de pacotes, o protocolo UDP seria o mais indicado.

Em geral, consideramos chegar a conclusões pertinentes e ter consolidado conhecimentos úteis para a Unidade Curricular de Comunicações por Computador, tendo cumprido os objetivos previstos.