Universidade do Minho

Departamento de Informática

TP1 : Protocolos da Camada de Transporte
Grupo 56

Cecília Soares (34900) Filipe Monteiro (a80229)

Leonardo Neri(80056)

1 de Março de 2019

Conteúdo

1 Questões e Respostas

1.1 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.

Comando usado	Protocolo de Aplicação	Protocolo de Transporte	Porta de Atendimento	Overhead[1] de Transporte (em bytes)
Ping	-	-	-	-[2]
Traceroute	-	-	-	-[2]
Telnet	TELNET	TCP	23	$\frac{20}{485} * 100 = 4\%$
ftp	FTP	TCP	21	$\frac{32}{48} * 100 = 67\%$
Tftp	TFTP	UDP	69	$\frac{8}{52} * 100 = 15,4\%$
Browser	HTTP	TCP	80	$\frac{32}{131} * 100 = 25\%$
nslookup	DNS	UDP	53	$\frac{8}{39} * 100 = 20\%$
ssh	SSH	TCP	22	$\frac{20}{61} * 100 = 33\%$
https	SSL	TCP	443	$\frac{32}{272} * 100 = 11,7\%$
ntp	NTP	UDP	123	$\frac{8}{64} * 100 = 12,5\%$

Tabela 1: Camada de Transporte

Para testar a conectividade, ao nível do IP, entre máquinas utilizamos o comando Ping (*Packet InterNet Groper*) direcionado a cisco.di.uminho.pt. Conforme podemos observar através da figura 1, o destino está alcançável e podemos saber qual o delay da resposta, o tempo de vida do pacote, entre outras informações da rede.

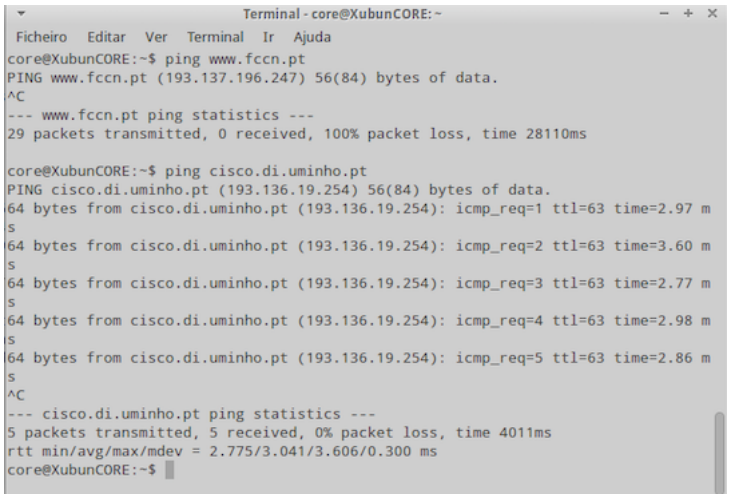

Figura 1: Ping

Através da análise do tráfego da rede após o uso do comando ping, concluimos que o mesmo corre diretamente na camada 3 do Modelo OSI e, por isso, não tem protocolo de aplicação, nem protocolo de transporte, e também não tem porta de atendimento, nem overhead de transporte (ver figura 2). O mesmo acontece com o comando de traceroute, conforme se pode verificar através da figura 3. Note-se que usamos a "flag-I para acionar o comando traceroute, pelo que o mesmo corre diretamente sobre a camada 3. Assim, ambos os referidos comandos (ping e traceroute) "testam"a rede ao nível da camada de rede e servem-se do *Internet Control Message Protocol* (ICMP) para diagnosticar o funcionamento dessa mesma rede.

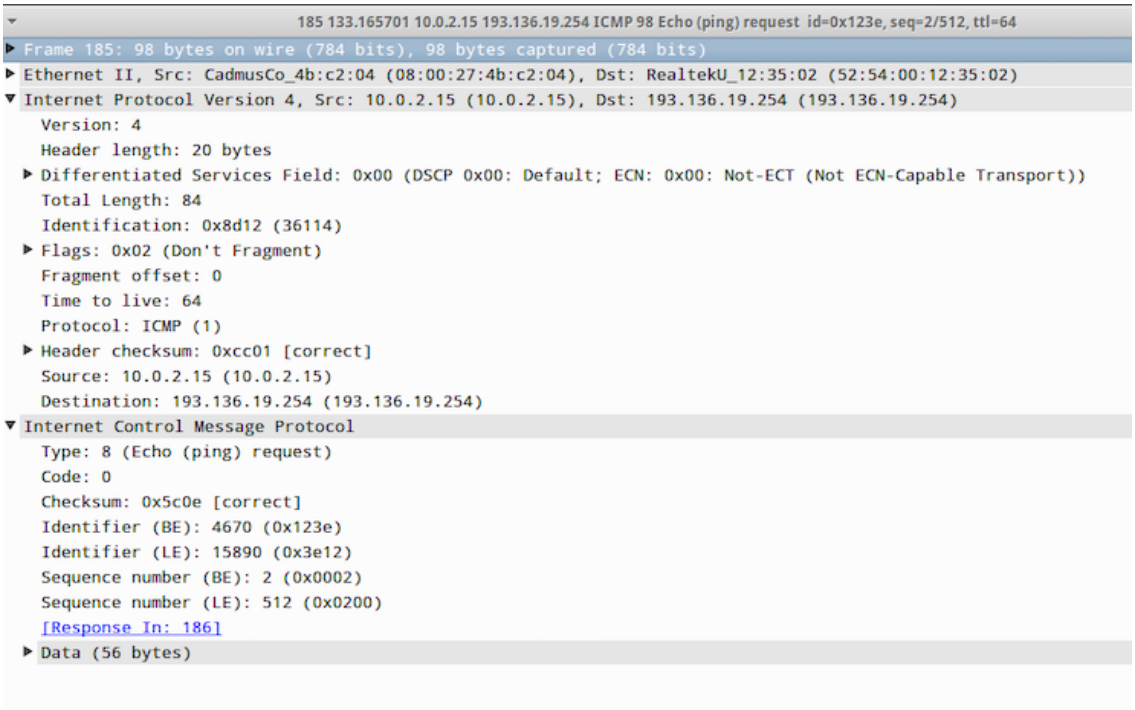

Figura 2: Análise do comando ping no wireshark

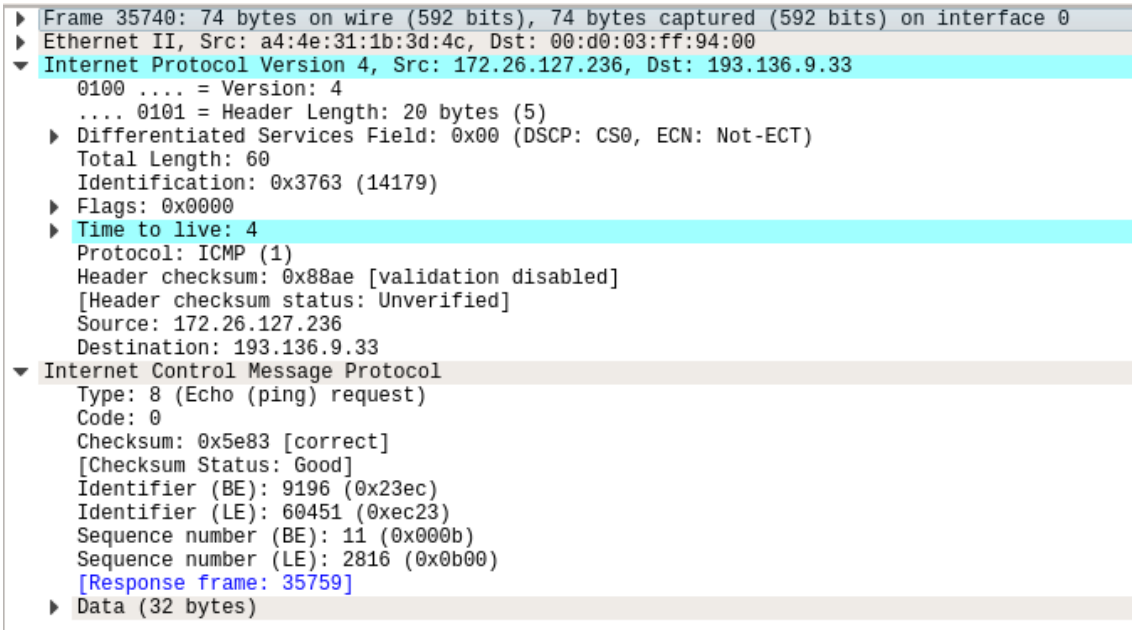

Figura 3: Traceroute

No que se refere ao comando Telnet, verificamos que o protocolo de aplicação é o protocolo TELNET, o protocolo de transporte é o *Transmission Control Protocol* (doravante TCP), a porta de atendimento é a 23 e o overhead é de 4% (ver figura 4).

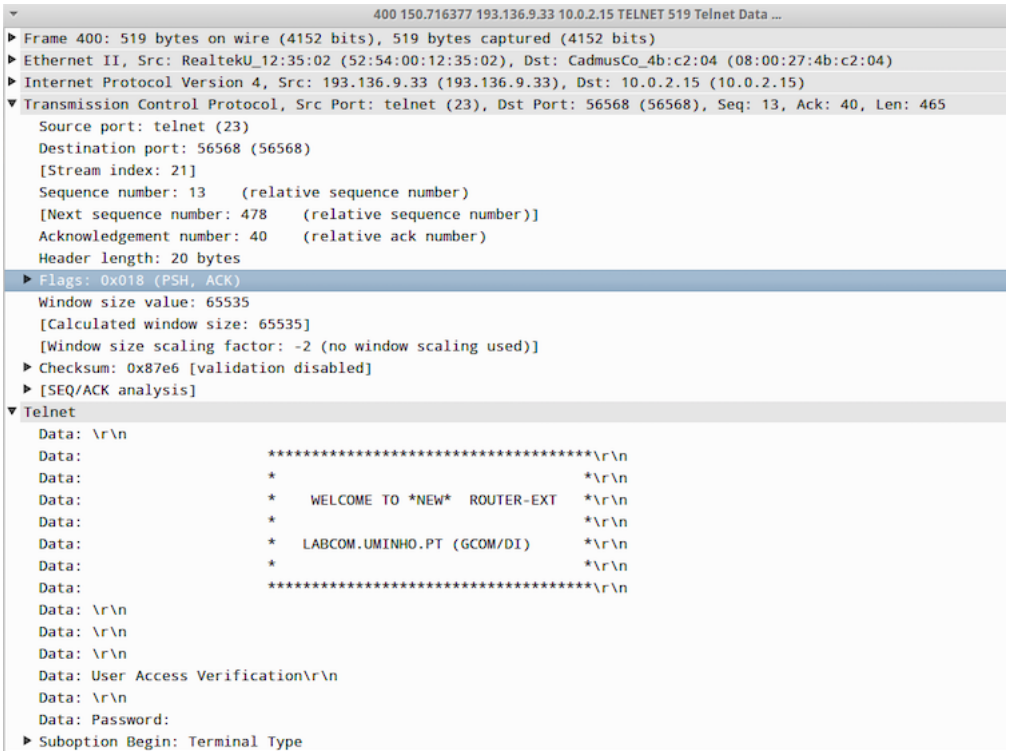

Figura 4: Análise do comando Telnet no wireshark

Quanto ao comando ftp, o protocolo de aplicação é o *File Transfer Protocol* (FTP), o protocolo

de transporte é o TCP, a porta de atendimento é a 21 e o overhead é de 67% (ver figura 5).

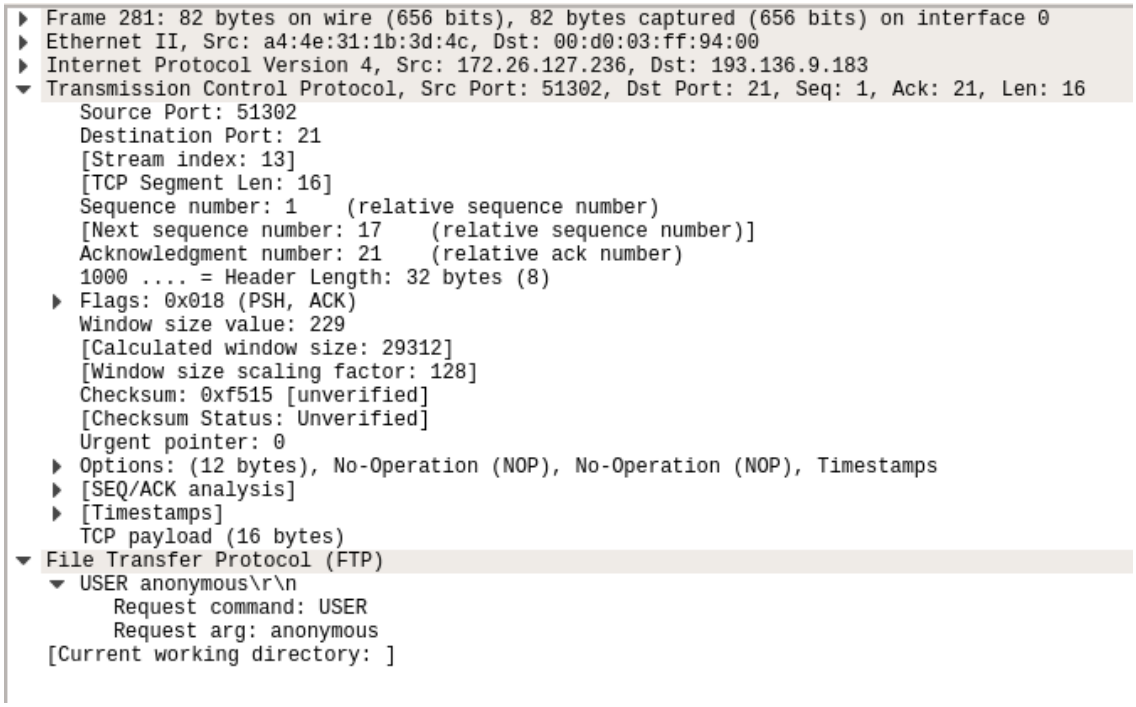

Figura 5: Análise do comando ftp no wireshark

O quinto comando que utilizamos foi o Tftp, sendo que este utiliza como protocolo de aplicação o *Trivial File Transfer Protocol* (TFTP), o seu protocolo de transporte é o *User Datagram Protocol* (doravante UDP), a porta de atendimento é a número 69 e o overhead da camada de transporte é de 15,4% (ver figura 6).

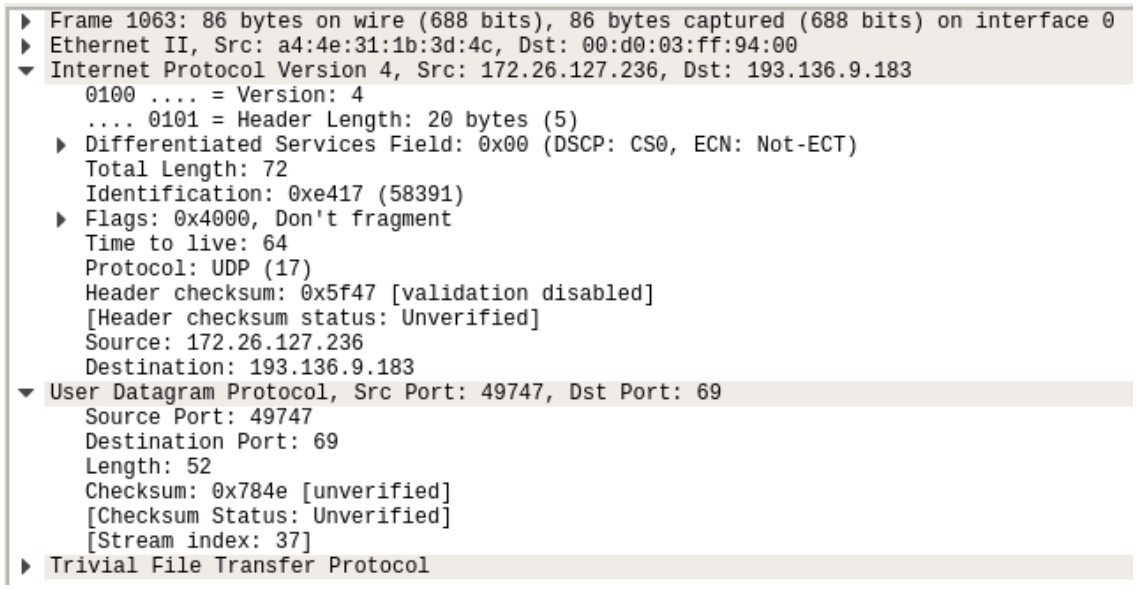

Figura 6: Análise do comando tftp no wireshark

De seguida utilizamos o comando http, cujo protocolo de aplicação utilizado é o *Hypertext*

Transfer Protocol (HTTP), o protocolo de transporte é o TCP, a porta de atendimento é a 80 e o overhead é de 25% (ver figura 7).

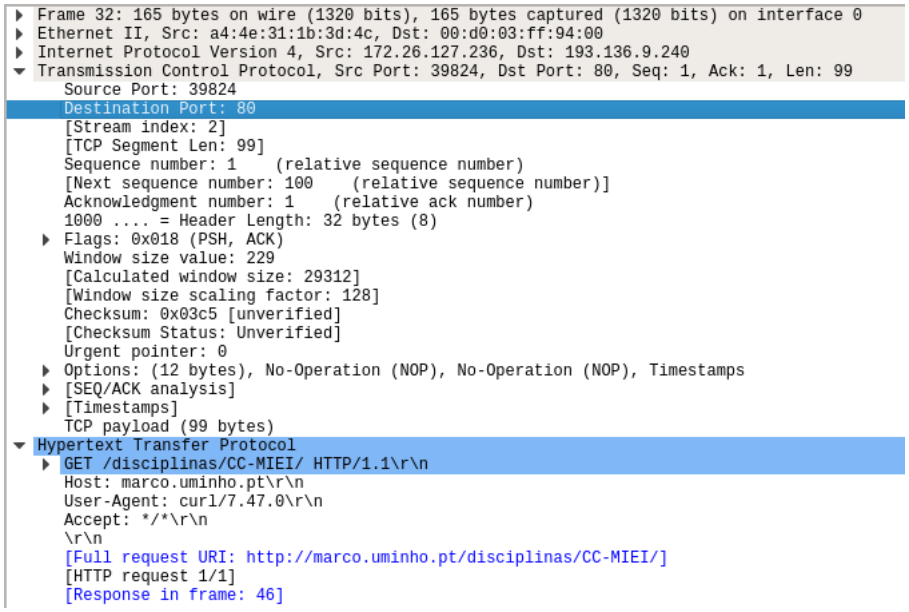

Figura 7: Análise do comando http no wireshark

Por sua vez, o comando nslookup utiliza o *Domain Name System* (DNS) como protocolo de aplicação, o UDP como protocolo de transporte, a porta de atendimento é a 53 e o overhead de transporte é de 33% (ver figura 8).

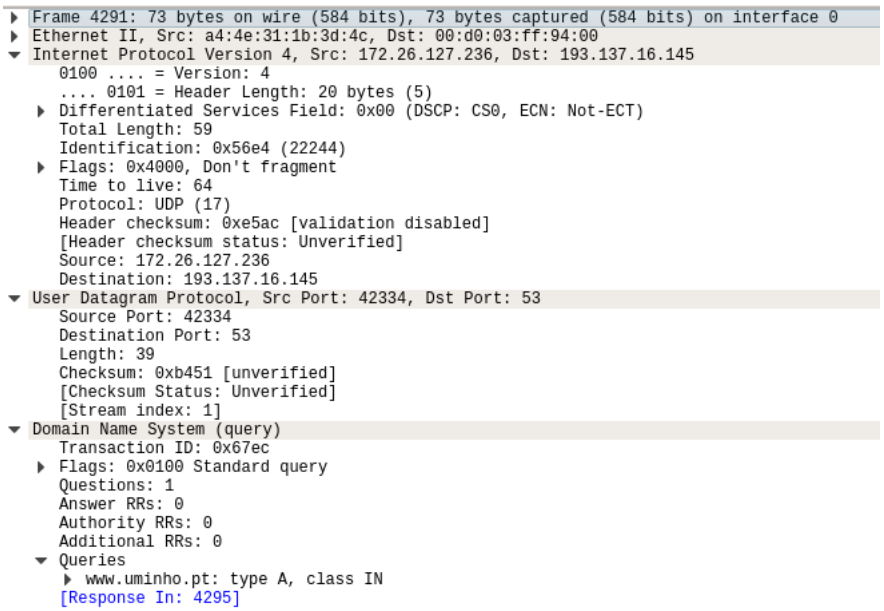

Figura 8: Análise do comando nslookup no wireshark

No que concerne ao comando ssh, o protocolo de aplicação é o *Secure Shell* (SSH), o protocolo de transporte é o TCP, a porta de atendimento é a 22 e o overhead de transporte é de 33% (ver figura 9).

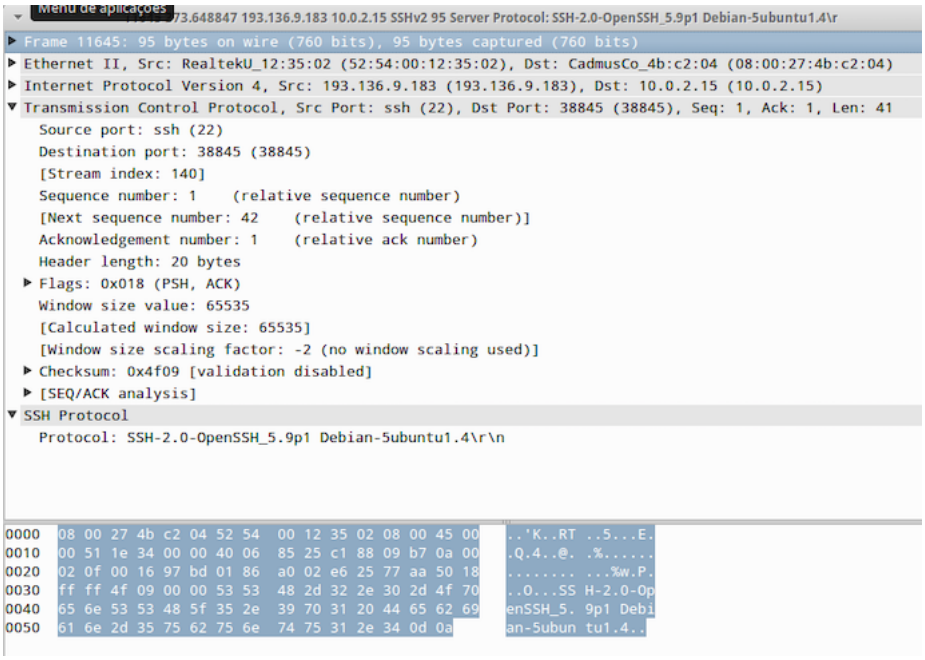

Figura 9: Análise do comando ssh no wireshark

Relativamente ao comando https, o protocolo de aplicação é o *Secure Sockets Layer* (SSL), o protocolo de transporte é o TCP, a porta de atendimento é a 443 e o overhead de transporte é de 11,7% (ver figura 10).

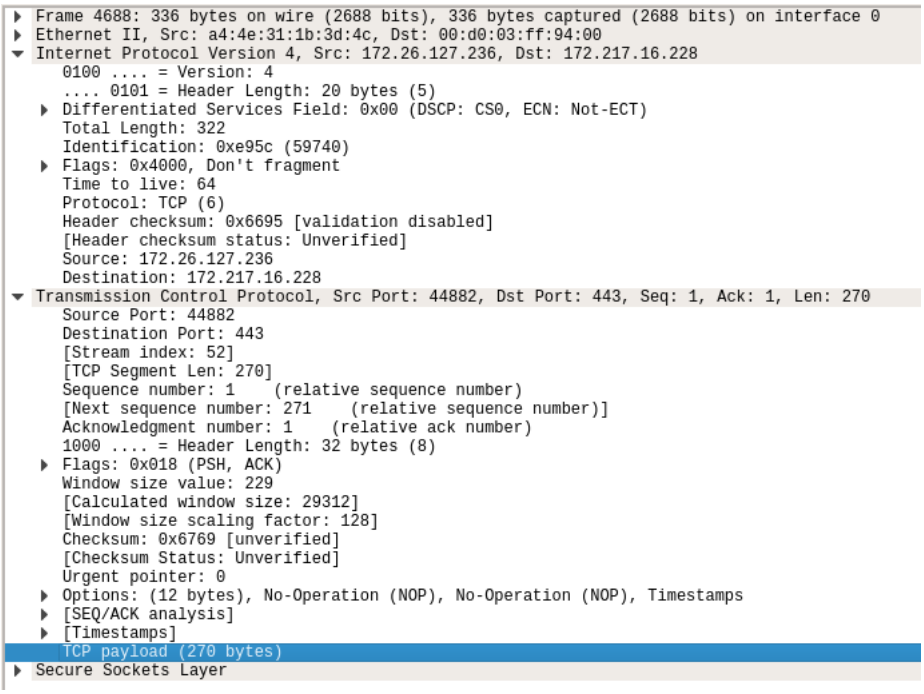

Figura 10: Análise do comando https no wireshark

Finalmente, executamos o comando ntpdate que utliza o *Network Time Protocol* (NTP) como protocolo de aplicação, o UDP como protocolo de transporte, a porta de atendimento é a 123 e o

overhead de transporte é de 12,5% (ver figura 11).

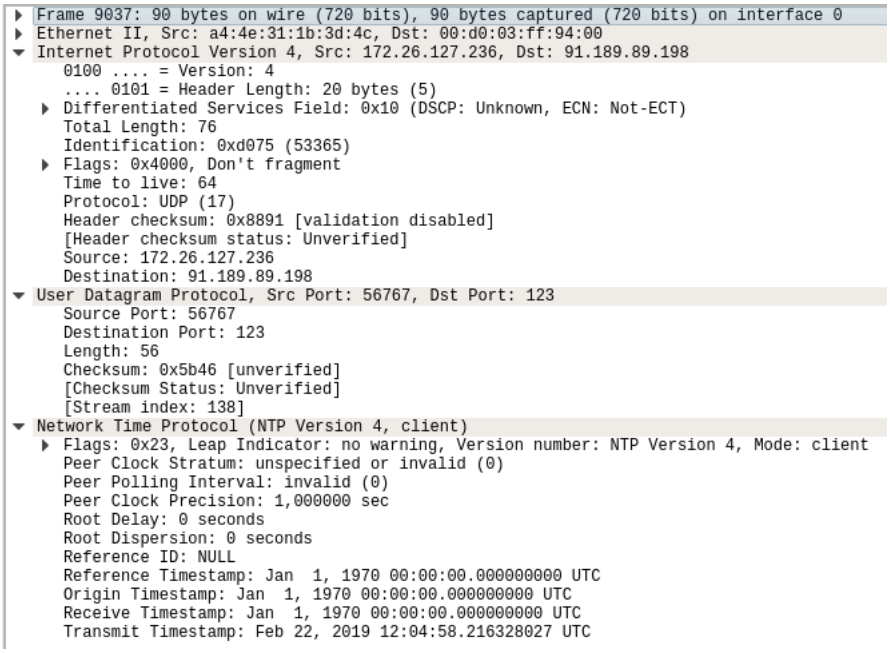

Figura 11: Análise do comando ntp no wireshark

1.2 Uma representação dum 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 e os números de sequência usados quer nos dados como nas confirmações. ***Nota: a transferência por FTP envolve mai sque uma conexão FTP, nomeadamente uma de controlo [ftp] e outra de dados [ftp-data]. Faça o diagrama apenas para a conexão de transferência de dados do ficheiro mais pequeno).***

Nas figuras 12 e 16 apresentamos um diagrama temporal das transferências do ***file1*** por **FTP** e **TFTP**, respetivamente. Os referidos diagramas foram elaborados tendo por base os dados recolhidos pelo programa wireshark.

Por um lado, a representação da transferência por **FTP** num diagrama temporal sustentou-se nas informações captadas nas figuras 13 (estabelecimento da conexão), 14 (dados transferidos) e 15 (encerramento da conexão).

Por outro lado, o nosso diagrama temporal da transferência por **TFTP** foi alicerçado nos resultados constantes da figura 17.

[1]O overhead é calculado através da divisão do header da mensagem pelo tamanho da mesma.
[2]Esta aplicação corre diretamente na camada de rede.

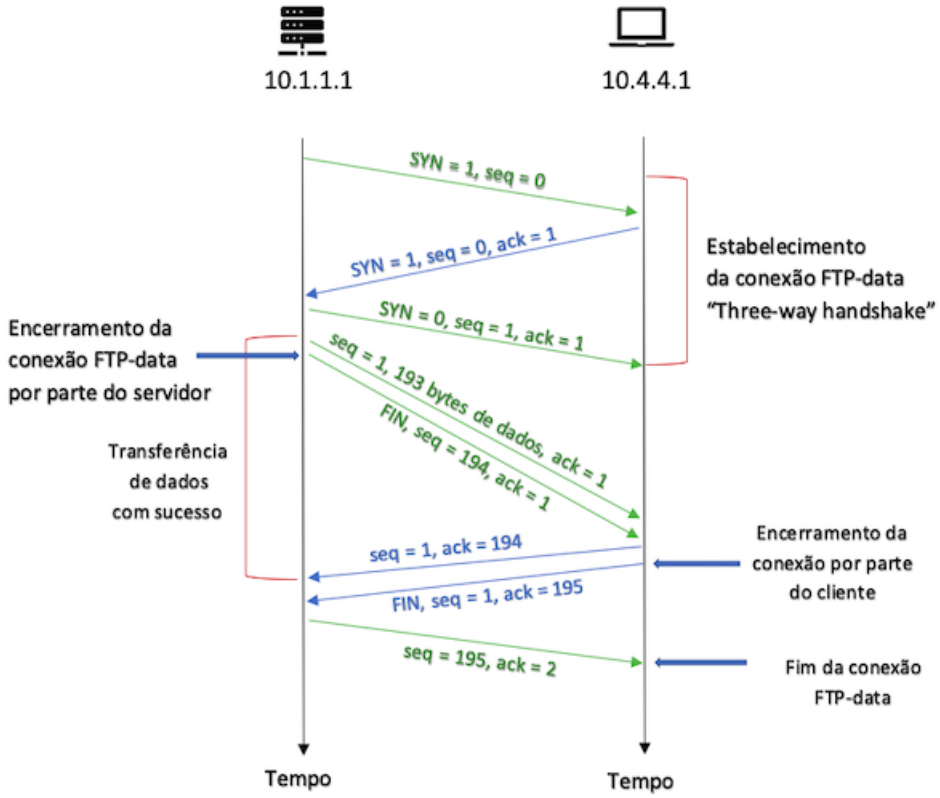

Figura 12: Diagrama da transferência do ficheiro por **FTP**

```
10.1.1.1    10.4.4.1    TCP    74 ftp-data > 43218 [SYN] Seq=0 Win=14600 Len=0 MSS=1460 SACK_PERM=1 TSval=11765781 TSecr=0 WS=16
10.4.4.1    10.1.1.1    TCP    74 43218 > ftp-data [SYN, ACK] Seq=0 Ack=1 Win=14480 Len=0 MSS=1460 SACK_PERM=1 TSval=11765782 TSecr=11765781 WS=16
10.1.1.1    10.4.4.1    TCP    66 ftp-data > 43218 [ACK] Seq=1 Ack=1 Win=14608 Len=0 TSval=11765783 TSecr=11765782
```

Figura 13: Estabelecimento da conexão

```
▶ Frame 790: 259 bytes on wire (2072 bits), 259 bytes captured (2072 bits)
▶ Ethernet II, Src: 00:00:00_aa:00:16 (00:00:00:aa:00:16), Dst: 00:00:00_aa:00:12 (00:00:00:aa:00:12)
▶ Internet Protocol Version 4, Src: 10.1.1.1 (10.1.1.1), Dst: 10.4.4.1 (10.4.4.1)
▼ Transmission Control Protocol, Src Port: ftp-data (20), Dst Port: 43218 (43218), Seq: 1, Ack: 1, Len: 193
    Source port: ftp-data (20)
    Destination port: 43218 (43218)
    [Stream index: 7]
    Sequence number: 1    (relative sequence number)
    [Next sequence number: 194    (relative sequence number)]
    Acknowledgement number: 1    (relative ack number)
    Header length: 32 bytes
  ▶ Flags: 0x018 (PSH, ACK)
    Window size value: 913
    [Calculated window size: 14608]
    [Window size scaling factor: 16]
  ▶ Checksum: 0x19ee [validation disabled]
  ▶ Options: (12 bytes)
  ▼ [SEQ/ACK analysis]
      [Bytes in flight: 193]
▶ FTP Data
```

Figura 14: Tranferência de dados

10.1.1.1	10.4.4.1	TCP	66 ftp-data > 43218 [FIN, ACK] Seq=194 Ack=1 Win=14608 Len=0 TSval=11765783 TSecr=11765782
10.4.4.1	10.1.1.1	TCP	66 43218 > ftp-data [ACK] Seq=1 Ack=194 Win=15552 Len=0 TSval=11765784 TSecr=11765783
10.4.4.1	10.1.1.1	TCP	66 43218 > ftp-data [FIN, ACK] Seq=1 Ack=195 Win=15552 Len=0 TSval=11765784 TSecr=11765783

Figura 15: Encerramento da conexão

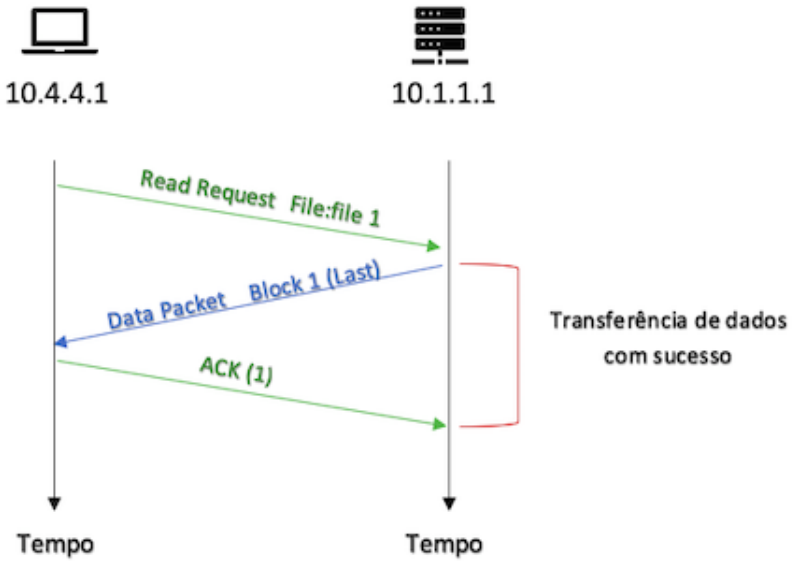

Figura 16: Diagrama da transferência do ficheiro por **TFTP**

23502	17144.4372	10.4.4.1	10.1.1.1	TFTP	56 Read Request, File: file1, Transfer type: octet
23503	17144.4375	10.1.1.1	10.4.4.1	TFTP	239 Data Packet, Block: 1 (last)
23504	17144.4382	10.4.4.1	10.1.1.1	TFTP	46 Acknowledgement, Block: 1

Figura 17: Diagrama da transferência do ficheiro por **TFTP**

1.3 Com base nas experiências realizadas, distinga e compare, sucintamente, as quatro aplicações de transferência de ficheiros que usou nos seguites pontos (i) uso da camada de transporte; (ii) eficiência na transferência; (iii) complexidade; (iv) segurança.

(i) Camada de Transporte

Com excepção do **TFTP** que usa o UDP como protocolo da camada de transporte, verificou-se que todas as outras aplicacões analisadas usam TCP nessa mesma camada.

(ii) Eficiência na transferência

Fazendo numa análise mais interna das aplicações, em especial dos protocolos que estas utilizam na camada 4 do Modelo OSI, a aplicação **TFTP** é a mais eficiente porque utiliza o protocolo UDP. Este protocolo é mais simples do que o protocolo TCP porque, por si só, não garante a entrega dos pacotes ao destinatário, nem necessita de procedimentos para o acesso ao sistema de comunicação, denominado "Three-way handshake"(SYN, SYN-ACK, ACK), e para o encerramento da conexão (FIN, ACK, FIN, ACK). Estas duas razões, aliadas ao facto da nossa topologia de rede ser pequena e sem grande tráfego, assim como de apenas ser necessário transferir um único pacote de dados,

fazem com que a comunicação seja mais rápida com a referida aplicação do que com as outras três.

Com efeito, esse facto foi corroborado pelas capturas do tráfego da rede, tendo sido observado que o **TFTP** demorou 0,9ms; o **HTTP** quase 8ms; o **FTP** demorou, aproximadamente, 12ms e o **SFTP** 16ms, a transferir o *file1*, o qual continha 192 bytes.

(iii) Complexidade

Quanto à complexidade, se a mesma for avaliada em função do "trabalho"que a aplicação tem na transferência dos pacotes, o **TFTP** é de todos o mais complexo pois tem de implementar algumas funcionalidades que o UDP não possui e que são intrínsecas a outras aplicações que utilizam o TCP como protocolo da camada de transporte, nomeadamente garantir uma ligação fiável entre o cliente e o servidor ou o controlo de congestionamento.

Por seu turno, se encararmos o conceito de complexidade numa perspetiva das funcionalidades de cada uma das aplicações concluimos que o **TFTP**, é o mais simples das aplicações em análise, não sendo recomendado para iterações complexas entre o servidor e o cliente, visto que não garante uma conexão fiável. Na verdade, esta aplicação, por ser tão simples e modesta ao nível funcional é usada somente para a transferência de pequenos ficheiros.

(iv) Segurança

No que se refere à segurança das aplicações, verificou-se que o **SFTP** é a aplicação que oferece mais garantias e segurança numa comunicação fiável porque possui melhores técnicas de encriptação dos dados em relação às demais. Esta aplicação usa o protocolo SSHv2 que se caracteriza por oferecer métodos de autenticação robustos que garantem uma comunicação entre o cliente e o servidor segura. Ademais, o referido protocolo não permite o acesso ao diretório raíz, ao contrário do **FTP**, o qual é, por isso, mais vulnerável. (ver figura 18)

Nos antípodas temos as aplicações **HTTP**, que não utiliza nenhum método de encriptação de dados, e **TFTP**, que também não tem mecanismos de autenticação, nem de encriptação de dados.

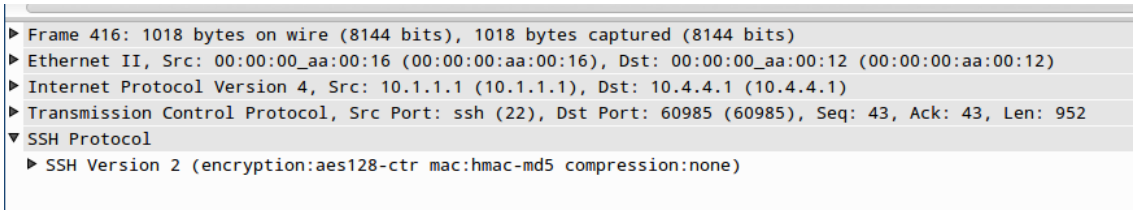

Figura 18: Pacote encriptado pelo SSHv2

1.4 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).

Para responder a esta pergunta foram comparadas duas aplicações baseadas em HTTP e TFTP, as quais utilizam como protocolo de transporte TCP e UDP, respectivamente. Primeiro o arquivo foi transferido através de um link não fiável, com perdas e depois através de uma rede com boa confiabilidade. A Tabela 2 assinala os tempos de transmissão obtidos em cada um destes cenários.

Como era esperado o TFTP, que utiliza UDP na camada de transporte, tem um desempenho muito melhor que a aplicação que usa TCP em uma rede fiável. Isso acontece porque o UDP não apresenta uma série de funcionalidades implementadas pelo TCP, como controle de fluxo, controle de congestão, controle de erros e alguns formalismos para estabelecer e encerrar uma

Aplicação	Rede não fiável (100Mbps, 5% perdidos e 15% duplicados)	Rede fiável (1Gbps)
HTTP	1.5669s	2.9432s
TFTP	67.5274s	170.743ms

Tabela 2: Tempos de transmissão das aplicações

conexão. Todas estas funcionalidades já implementadas pelo TCP podem, opcionalmente, ser implementadas pela aplicação que utiliza sockets UDP para transferência de dados.

Quando passamos para o cenário de uma rede não fiável, o HTTP tem um desempenho melhor que o TFTP, devido às funcionalidades já descritas anteriormentes implementadas pelo TCP.

Figura 19: Transferência de um arquivo via TFTP em uma rede não fiável

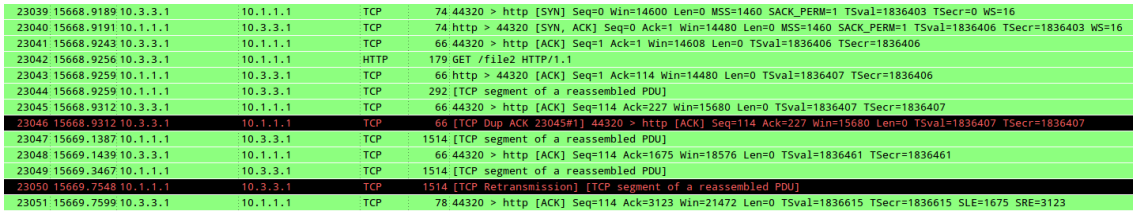

Figura 20: Transferência de um arquivo via HTTP em uma rede não fiável

2 Conclusões

Em conclusão, os protocolos da camada de transporte podem ser extremamente simples, não oferecendo nenhum serviço adicional às aplicações, tal como acontece com o protocolo UDP, que apenas garante as funções de desmultiplexagem e multiplexagem inerentes à comunicação em qualquer topologia de rede. Contudo, um protocolo de transporte pode oferecer uma variedade de garantias às aplicações, como uma comunicação fiável e a gestão do tráfego na rede. Exemplo paradigmático desse tipo de protocolo é o TCP que fornece uma série de funcionalidades às aplicações comparativamente com o UDP.

Na verdade, apesar do TCP poder ser menos eficiente e mais "pesado"do que o UDP, a sua principal vantagem prende-se com a fiabilidade na comunicação. Este protocolo é orientado à conexão, garantindo que a comunicação entre o cliente e o servidor é estabelecida através do procedimento *Three-way Handshake*, assegura que todos os pacotes são recebidos e ordenados antes de serem transferidos para a camada de rede superior e faz controlo de fluxo e controlo de

congestão, sendo, por isso, um protocolo de transporte muito mais complexo e mais "pesado"(só o seu header tem 20 bytes, ao passo que o header do UDP tem apenas 8) que o protocolo UDP.

Apesar de serem muito diferentes, estes dois protocolos de transporte, UDP e TCP, têm funções específicas, se o UDP é mais utilizado pelas aplicações quando o que está em causa é a rapidez e a simplicidade de uma comunicação, contudo, priveligiam o TCP quando necessitam de maior segurança e fiabilidade na comunicação.

O presente trabalho permitiu-nos comparar diferentes aplicações (TFTP, FTP, SFTP e FTP) e testar a eficiência, segurança e complexidade das mesmas no âmbito da transferência de ficheiros. Com base nos resultados obtidos no caso em concreto, a transferência de um ficheiro com dimensão reduzida (cerca de 190 bytes), concluímos que a aplicação TFTP foi a mais eficiente, mas a menos segura, ao passo que a aplicação SFTP é a mais segura mas a mais morosa, sendo 16 vezes mais lenta do que a TFTP. Ademais, convém ainda notar que a aplicação HTTP também não tem qualquer tipo de encriptação de dados, o que é compreensível, visto que a sua principal preocupação é a rapidez da transferência de dados.

Finalmente, analisamos a camada de transporte e duas aplicações (HTTP e TFTP) em dois cenários díspares, uma topologia de rede perfeita e outra onde 5% dos pacotes são perdidos e 15% duplicados. Face a estes duas situações, observamos que numa rede fiável, o TFTP tem melhor desempenho face ao HTTP, o que se compreende dado que este último usa TCP, que é um protocolo muito mais "pesado"e o primeiro utiliza o UDP, o qual, ao contrário do TCP, não tem preocupações com o controlo de fluxo ou congestionamento da rede, limitando-se a transferir pacotes. Por seu turno, no caso de uma rede não fiável, o TFTP teve um desempenho desastroso comparativamente com o HTTP, demorando quase 45 vezes mais tempo do que o HTTP, corroborando o que dissemos anteriormente relativamente ao UDP, ou seja, as suas parcas funcionalidades e o descaso face às características da rede podem ter um efeito perverso quando não está só em causa a velocidade na comunicação.