

UNIVERSIDADE FEDERAL DE SANTA CATARINA
DEPARTAMENTO DE INFORMATICA E ESTATISTICA
DISCIPLINA: Redes De Computadores I
PROFESSOR: Carlos Becker Westphall
ACADÊMICO: Bruno Aurélio Rôzza de Moura Campos
MATRÍCULA: 14104255

TRABALHO PRÁTICO 03

Florianópolis, maio de 2016

1.Introdução

Proposta do professor:

Neste trabalho prático os alunos deverão utilizar o "wireshark" para identificar a ocorrência de conexões e transferências de dados, envolvendo as camadas de aplicação, transporte e rede. Nestas telas capturadas pelo "wireshark" os alunos deverão identificar os "pacotes" relacionados com criação da conexão, a transferência de dados e a liberação de conexões. Usar, preferencialmente, FTP, HTTP e/ou SNMP. Fazer o relatório descrevendo as atividades. O arquivo de dados analisado e obtido através do "wireshark" também deve ser entregue junto com o relatório.

Esse relatório tem como objetivo analisar o tráfego de uma rede para identificar a criação da conexão, a transferência de dados e a liberação de conexões, com base em um programa sniffer (wireshark). Este tipo de ferramenta registra os pacotes, que trafegam pelas redes, e suas informações. Como são muitos dados, o programa possibilita a filtragem por tipo de protocolos. Para uma maior facilidade foi filtrado os protocolos de interesse HTTP e TCP.

1.1 Dados utilizados

Para a realização deste trabalho, foi escolhido para analisar a URL <https://moodle.ufsc.br> (150.162.1.112). No arquivo de captura de pacotes foi obtido 545 pacotes sendo o navegador utilizado Mozilla Firefox.

2. Criação da conexão

A primeira atitude para se obter a comunicação é denominado processo cliente. O processo de aplicação cliente primeiramente informa à camada de transporte do cliente que ele quer estabelecer uma conexão com um processo servidor. Para isso é realizado a apresentação de 3 vias (**3 way handshake**).

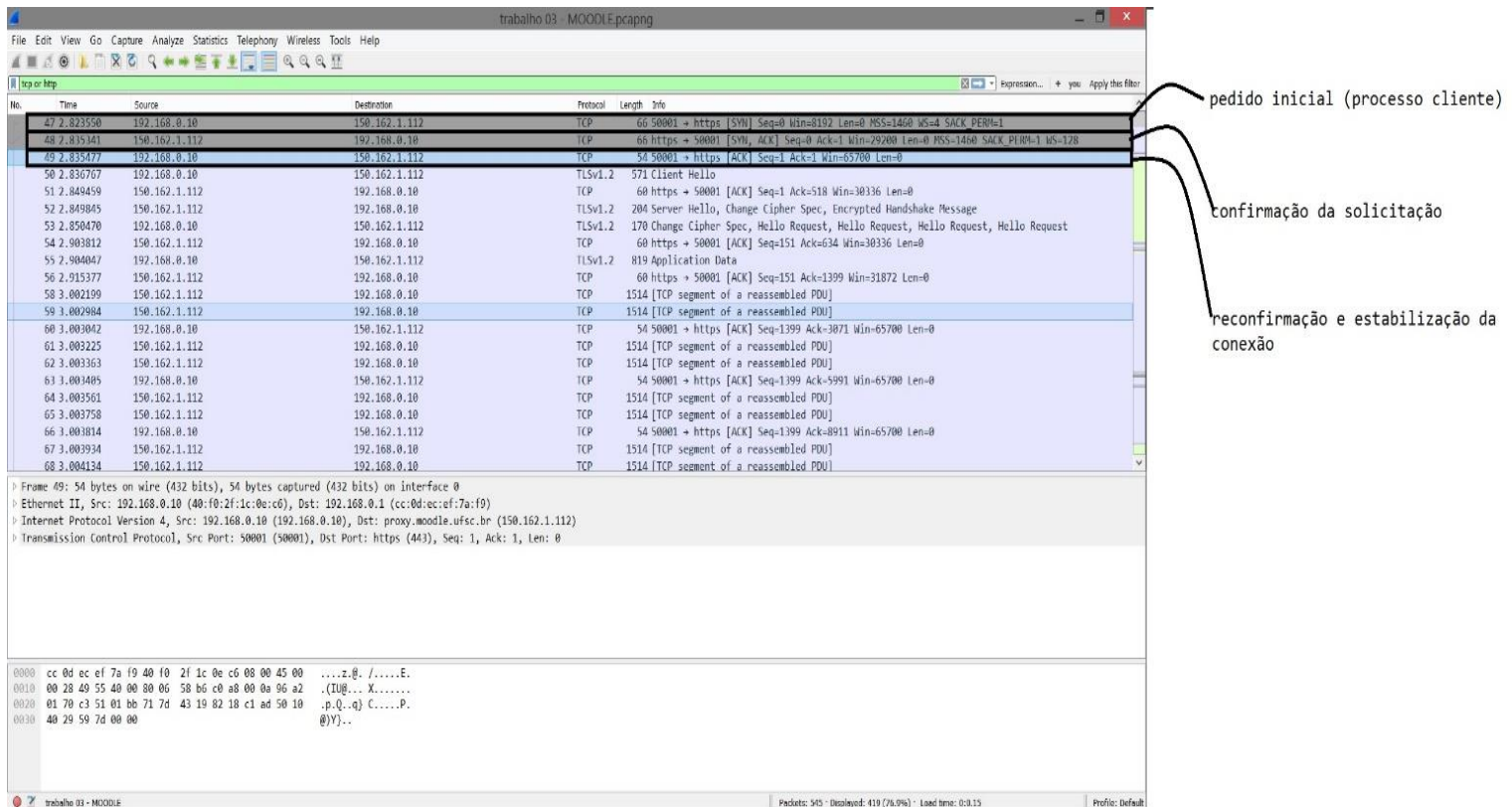
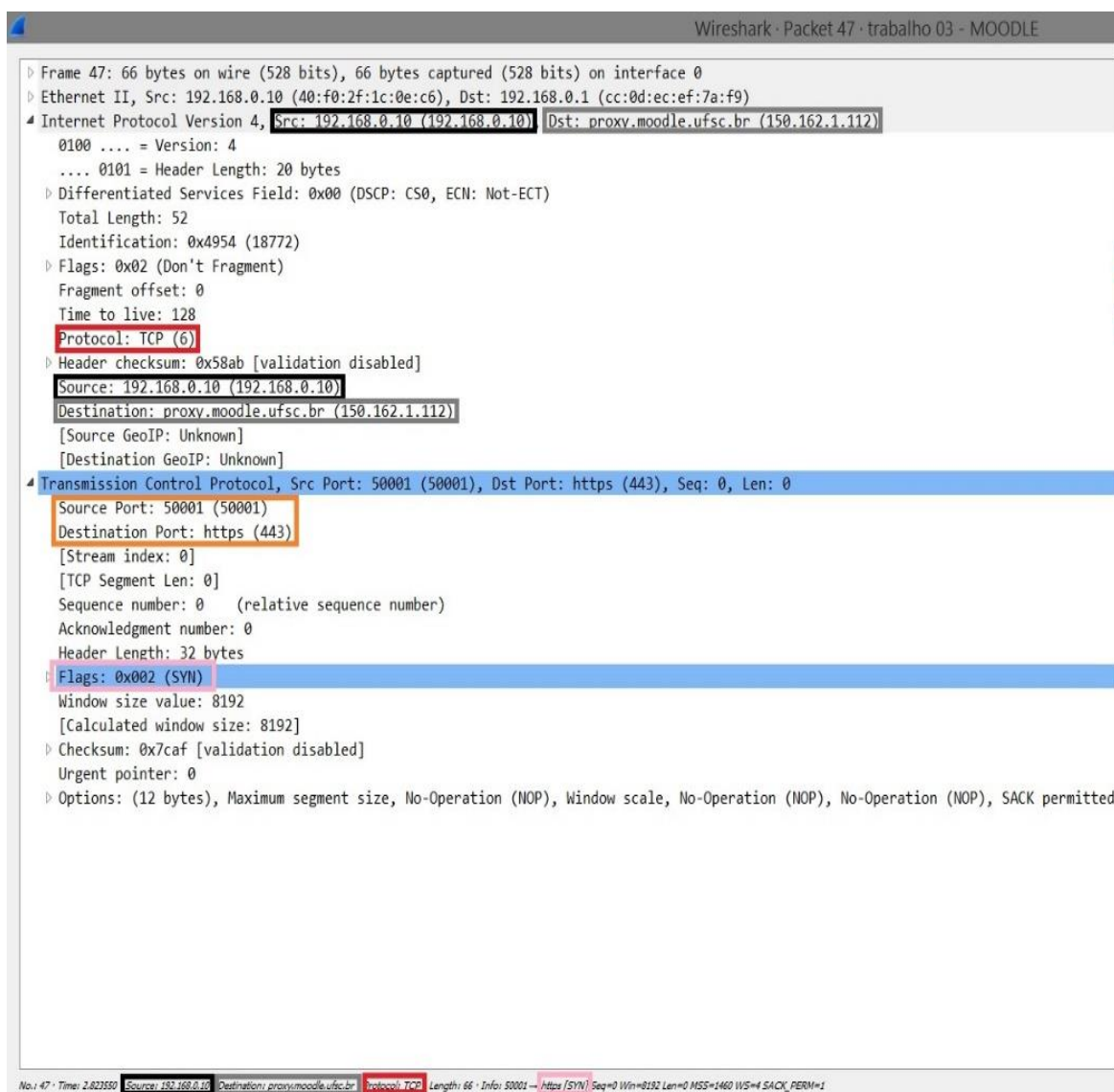


Figura 01. 3 way handshake

Etapa 01:

Primeiro é feito o pedido inicial com a flag SYN, com um valor aleatório.



LEGENDA:

	Origem do serviço (cliente)
	Destino do pacote (servidor)
	Protocolo usado
	Porta de origem e destino
	Valor da flag (SYN)

Figura 02. Início do processo de comunicação com a flag SYN

Etapas 02:

Se disponível o servidor responde com uma confirmação do pedido em um novo pacote contendo as flags SYN + ACK.

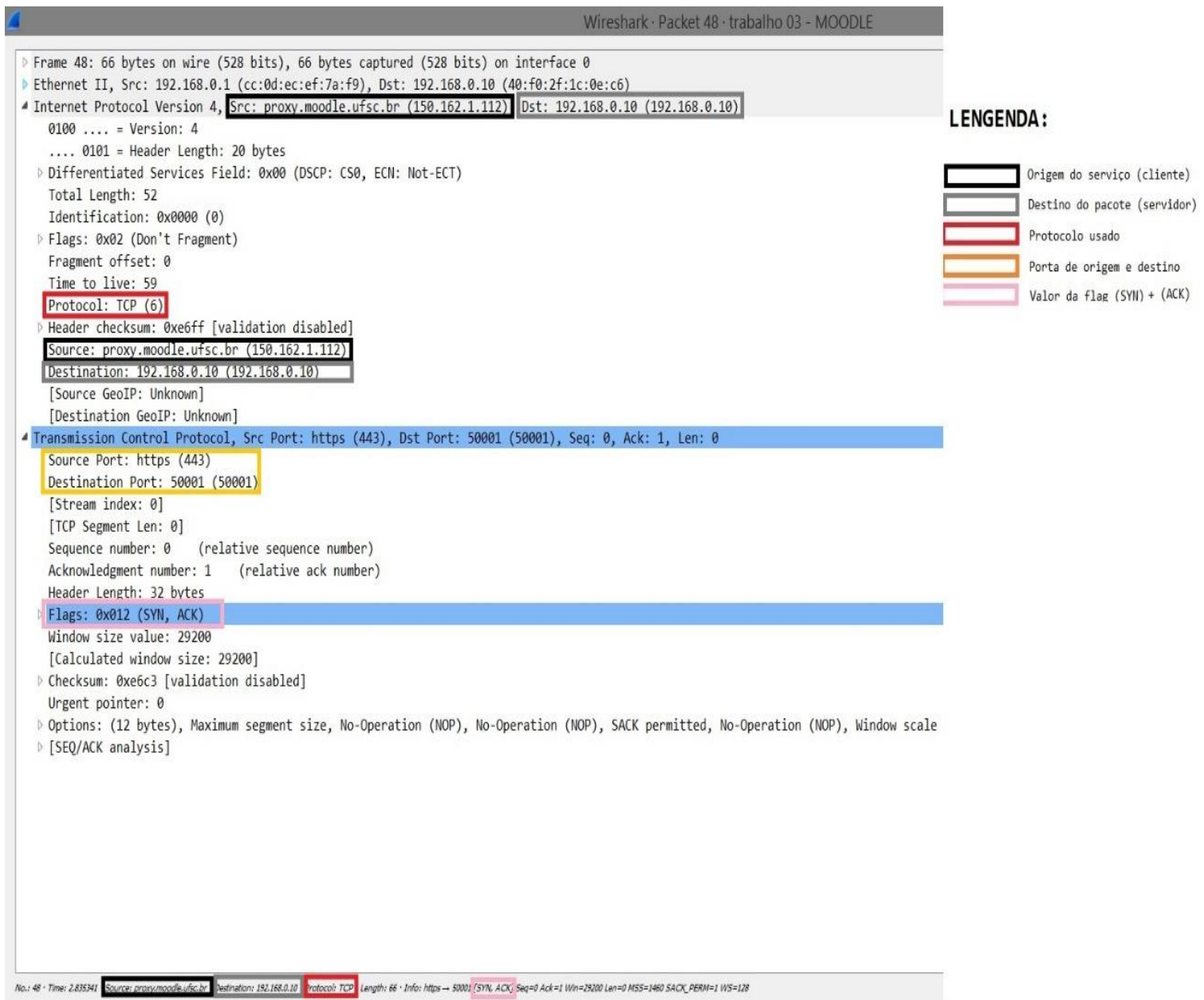


Figura 03. Resposta do servidor com as flags SYN + ACK

Etapa 03:

Para estabelecimento do processo cliente-servidor o cliente envia mais um pacote contendo um ACK do mesmo número de ACK recebido, que veio do servidor.

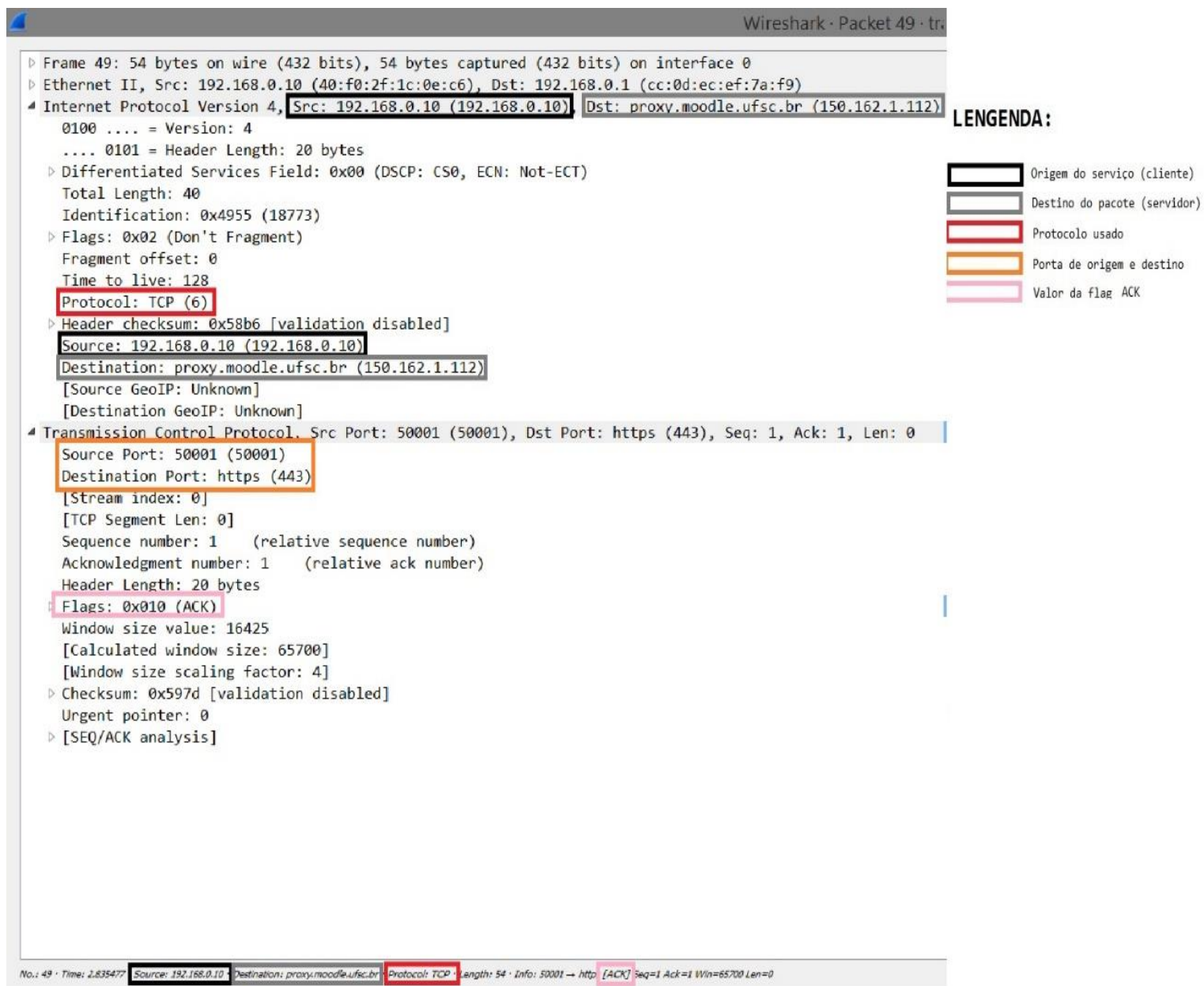


Figura 04. Estabelecimento de comunicação com a flag ACK

3. Transferência de Dados

Após o cliente ter executado uma primitiva `CONNECT`, especificando o endereço IP e a porta à qual deseja se conectar, a primitiva `CONNECT` envia um segmento TCP com o bit da flag `SYN` ativado e um bit da flag `ACK` desativado, e aguarda uma resposta. Para um maior entendimento sobre as flags, abaixo há um exemplo de como ela é configurada num pacote.

```
▲ Flags: 0x010 (ACK)
000. .... = Reserved: Not set
...0 .... = Nonce: Not set
.... 0... = Congestion Window Reduced (CWR): Not set
.... .0.. = ECN-Echo: Not set
.... ..0. = Urgent: Not set
.... ...1 = Acknowledgment: Set
.... ....0... = Push: Not set
.... .....0.. = Reset: Not set
.... ..... ..0. = Syn: Not set
.... ..... ...0 = Fin: Not set
[TCP Flags: *****A****]
```

Figura 05. Exemplo de ACK configurado dentro do pacote

Os programas que trabalham com redes estão fortemente ligados na camada TCP/IP, de aplicação. Com a requisição processada, essa camada chama a de Transporte através de uma porta específica, dependendo do protocolo usado, como HTTP, que sempre usa a 80. Assim já indica qual tipo do conteúdo e, no receptor, possibilita entregar os dados para o protocolo de aplicação correto.

A camada de Transporte é responsável no envio por dividir os dados em pacotes; e a receber por ordenar os pacotes. Sendo usado principalmente o TCP. O protocolo nomeado como Transmission Control Protocol encapsula nos pacotes informações de controle, resultando entre 20 a 24 bytes.

Na transferência de dados com o TCP, ao se enviar uma solicitação, ou seja, iniciar um processo cliente, obrigatoriamente haverá uma resposta com os seguintes campos:

Protocolo / versão do protocolo + código de requisição + significado do código + rubricas de resposta + corpo de resposta

The image shows a Wireshark packet capture analysis of an HTTP GET request. The left pane displays a list of packets, with packet 113 selected. The middle pane shows the packet details for the selected packet, highlighting the HTTP GET method. The right pane shows the raw packet data in hexadecimal and ASCII.

Packet List:

Time	Source	Destination
111 3.286308	paginas.ufsc.br	192.168.0.10
112 3.286452	192.168.0.10	paginas.ufsc.br
113 3.298758	192.168.0.10	paginas.ufsc.br
114 3.311042	paginas.ufsc.br	192.168.0.10
115 3.311696	paginas.ufsc.br	192.168.0.10
116 3.318406	192.168.0.10	paginas.ufsc.br
117 3.319183	192.168.0.10	paginas.ufsc.br
118 3.329930	paginas.ufsc.br	192.168.0.10
119 3.330040	192.168.0.10	paginas.ufsc.br
120 3.330145	paginas.ufsc.br	192.168.0.10
121 3.330197	192.168.0.10	paginas.ufsc.br
122 3.344418	192.168.0.10	paginas.ufsc.br
123 3.356833	paginas.ufsc.br	192.168.0.10
124 3.357770	paginas.ufsc.br	192.168.0.10
125 3.358323	paginas.ufsc.br	192.168.0.10
126 3.358404	192.168.0.10	paginas.ufsc.br
127 3.358689	paginas.ufsc.br	192.168.0.10
128 3.358933	paginas.ufsc.br	192.168.0.10
129 3.358983	192.168.0.10	paginas.ufsc.br
130 3.359211	paginas.ufsc.br	192.168.0.10
131 3.359424	paginas.ufsc.br	192.168.0.10

Packet Details:

- Frame 113: 257 bytes on wire (2056 bits), 257 bytes captured (2056 bits) on interface 0
- Ethernet II, Src: 192.168.0.10 (40:f0:2f:1c:0e:c6), Dst: 192.168.0.1 (cc:0d:ec:ef:7a:f9)
- Internet Protocol Version 4, Src: 192.168.0.10 (192.168.0.10), Dst: paginas.ufsc.br (150.162.2.2)
- Transmission Control Protocol, Src Port: 50005 (50005), Dst Port: http (80), Seq: 1, Ack: 1, Len: 257
 - Source Port: 50005 (50005)
 - Destination Port: http (80)
 - [Stream index: 2]
 - [TCP Segment Len: 203]
 - Sequence number: 1 (relative sequence number)
 - [Next sequence number: 204 (relative sequence number)]
 - Acknowledgment number: 1 (relative ack number)
 - Header Length: 20 bytes
 - Flags: 0x018 (PSH, ACK)
 - Window size value: 16425
 - [Calculated window size: 65700]
 - [Window size scaling factor: 4]
 - Checksum: 0x9de2 [validation disabled]
 - [Good Checksum: False]
 - [Bad Checksum: False]
 - Urgent pointer: 0
 - [SEQ/ACK analysis]
 - [iRTT: 0.012855000 seconds]
 - [Bytes in Flight: 203]
- Hypertext Transfer Protocol
 - GET / HTTP/1.1\r\n
 - Cookie: UserCulture=en-US\r\n
 - Host: www.ufsc.br\r\n
 - Accept: text/html, */*\r\n
 - Accept-Encoding: identity\r\n
 - User-Agent: Mozilla/5.0 (compatible; PRTG Network Monitor (www.paessler.com); Windows)\r\n\r\n
 - [Full request URI: http://www.ufsc.br/]
 - [HTTP request 1/1]
 - [Response in frame: 115]

Raw Data:

```

cc 0d ec ef 7a f9 40 f0 2f 1c 0e c6 08 00 45 00 2f 1c 0e c6 08 00 45 00
00 f3 78 9c 40 00 80 06 28 0a c0 a8 00 0a 96 a2 28 0a c0 a8 00 0a 96 a2
02 0a c3 55 00 50 6d 0d c5 2a a6 78 55 b1 50 18 c5 2a a6 78 55 b1 50 18
40 29 9d e2 00 00 47 45 54 20 2f 20 48 54 54 50 54 20 2f 20 48 54 54 50
2f 31 2e 31 0d 0a 43 6f 6f 6b 69 65 3a 20 55 73 6f 6f 6b 69 65 3a 20 55 73
65 72 43 75 6c 74 75 72 65 3d 65 6e 2d 55 53 0d 65 3d 65 6e 2d 55 53 0d
0a 48 6f 73 74 3a 20 77 77 77 2e 75 66 73 63 2e 77 77 2e 75 66 73 63 2e
    ....z.@. /.....E.
    ..x.@... (.....
    ...U.Pm. .*xU.P.
    @)....GE T / HTTP
    /1.1..Co okie: Us
    erCultur e=en-US.
    ..Host: w ww.ufsc.
  
```

Figura 06. Pacote HTTP com o método GET

No pacote acima há algumas informações que identificam o cliente no caminho mas o mais importante é o método da requisição. O servidor, por sua vez, identifica os cabeçalhos que lhe são convenientes e envia uma resposta. Neste exemplo, recebemos os cabeçalhos de resposta.

Método GET: É o método mais comum para o protocolo HTTP: solicita algum recurso como um arquivo ou um script CGI (qualquer dado que estiver identificado pelo URI) por meio do protocolo HTTP. O método GET é reconhecido

por todos os servidores. Quando uma solicitação obtém sucesso o código do HTTP é 200 caso contrário, se não existir, é 404.

4. Liberação de conexões

Para encerrar uma conexão, qualquer um dos lados pode enviar um segmento com o bit FIN o que significa que não há mais dados a ser transmitidos. Quando FIN é confirmado, esse sentido é desativado para novos dados.

De modo geral, são necessários **quatro segmentos TCP para encerrar uma conexão**, isto é, um FIN e um ACK para cada sentido.

Quando um dos extremos tem a iniciativa de finalizar a conexão, será enviado desse um pacote TCP com a flag FIN. Contando que não ocorra erros, o outro irá confirmar o pedido com o ACK e seguidamente irá enviar um FIN; sendo realmente interrompida a ligação quando o primeiro enviar um ACK.

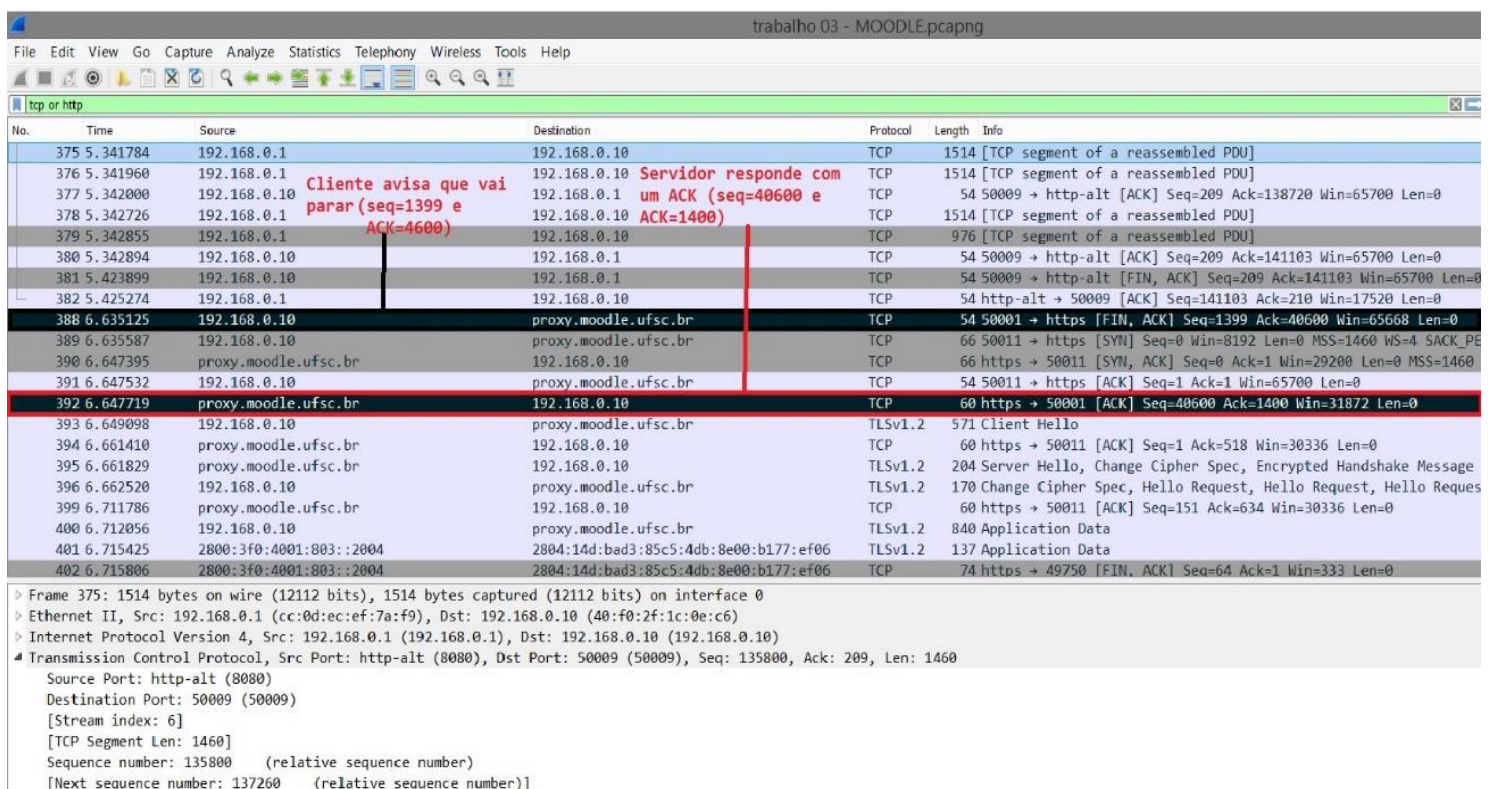


Figura 07. Dois primeiros pacotes para o encerramento da comunicação

Para maiores detalhes do encerramento de comunicação, veremos o pacote que contém a flag FIN.

Wireshark - Packet 388 - trabalho 03 - MOODLE

Internet Protocol Version 4, Src: 192.168.0.10 (192.168.0.10), Dst: proxy.moodle.ufsc.br (150.162.1.112)

Transmission Control Protocol, Src Port: 50001 (50001), Dst Port: https (443) Seq: 1399, Ack: 40600, Len: 0

Source Port: 50001 (50001)
Destination Port: https (443)
[Stream index: 0]
[TCP Segment Len: 0]
Sequence number: 1399 (relative sequence number)
Acknowledgment number: 40600 (relative ack number)
Header Length: 20 bytes

Flags: 0x011 (FIN, ACK)

000. = Reserved: Not set
...0. = Nonce: Not set
....0... = Congestion Window Reduced (CWR): Not set
....0.. = ECN-Echo: Not set
....0. = Urgent: Not set
....1... = Acknowledgment: Set
....0... = Push: Not set
....0.. = Reset: Not set
....0. = Syn: Not set
....1... = Fin: Set

[Expert Info (Chat/Sequence): Connection finish (FIN)]
[Connection finish (FIN)]
[Severity level: Chat]
[Group: Sequence]
[TCP Flags: *****A***F]
Window size value: 16417
[Calculated window size: 65668]
[Window size scaling factor: 4]
Checksum: 0xb576 [validation disabled]
[Good Checksum: False]
[Bad Checksum: False]
Urgent pointer: 0

LEGENDA:

Portas de origem e destino

Flags FIN + ACK

No.: 388 - Time: 6.635125 - Source: 192.168.0.10 - Destination: proxy.moodle.ufsc.br - Protocol: TCP - Length: 54 - Info: 50001 → https [FIN, ACK] Seq=1399 Ack=40600 Win=65668 Len=0

Figura 08. Portas de origem e destino são as mesmas utilizadas no 3 way handshake

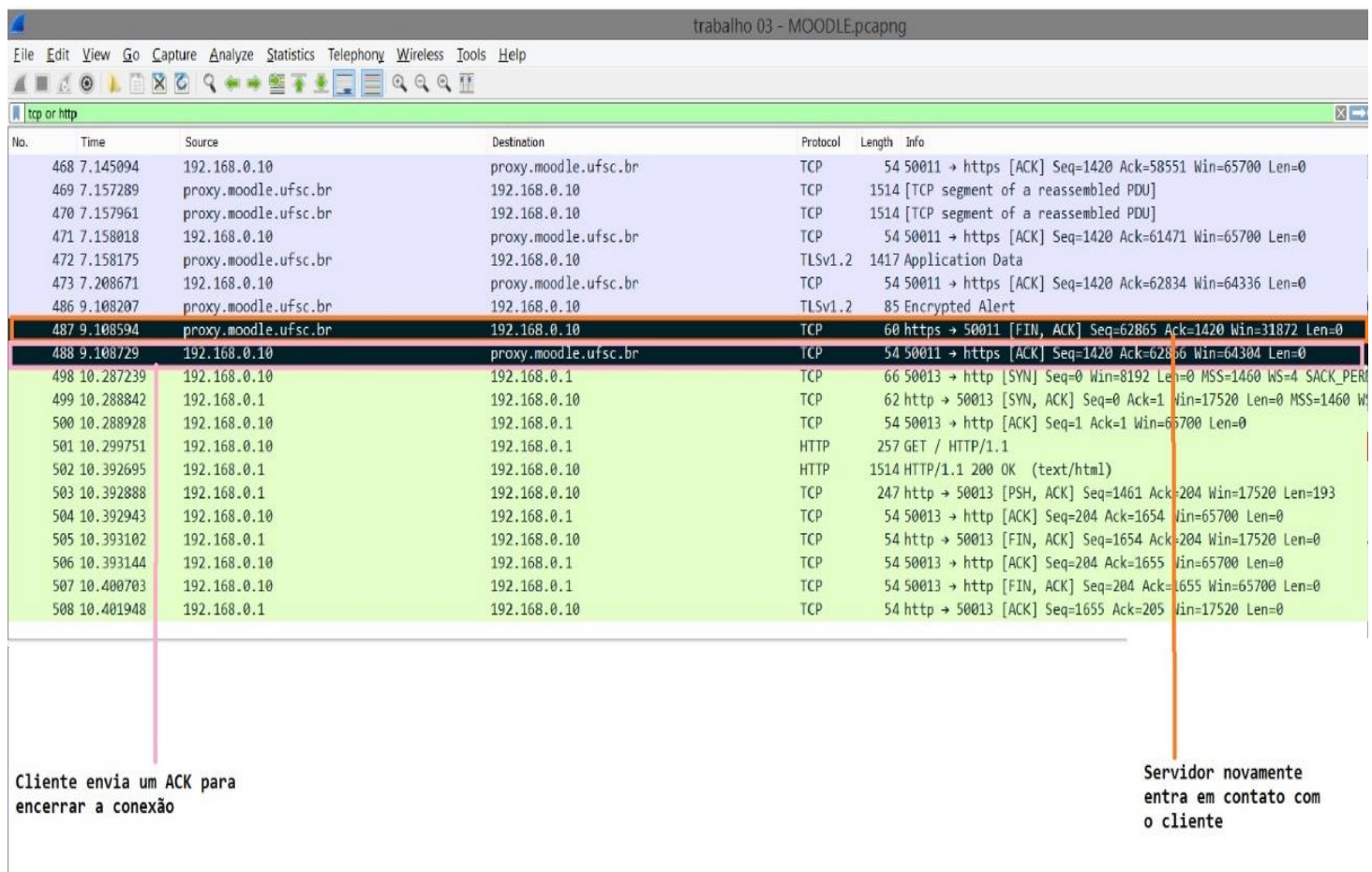


Figura 09. Dois últimos pacotes para o encerramento da comunicação

5. Bibliografia

- [1] https://www.oficinadanet.com.br/artigo/459/o_protocolo_http
- [2] <https://nandovieira.com.br/entendendo-um-pouco-mais-sobre-o-protocolo-http>
- [3] https://en.wikipedia.org/wiki/List_of_HTTP_status_codes
- [4] Kurose, Ross. (2013). *Redes de Computadores e a Internet*, Local: Pearson