



Universidade Federal de Viçosa – Campus  
UFV-Florestal

Ciência da Computação – Redes de Computadores

Professora: Thais Regina de Moura Braga Silva

Monitores: Aymê Faustino / Pedro Augusto

## **LAB - Laboratório Wireshark**

4705 - Gabriel Santos Ferreira de Pádua

5093 - Matheus Kauan Passos de Souza

5382 - Matheus Júnio da Silva

<b>Parte 1 - Ambientação do Wireshark:</b>	<b>3</b>
1. Descrição do software.....	3
2. Captura de teste.....	3
3. Cores de cada protocolo padrão no Wireshark.....	5
4. Captura automática de arquivos.....	6
<b>Parte 2 - Análise Do Protocolo HTTP:</b>	<b>8</b>
2.1) Qual versão do http está sendo usada?.....	10
2.2) Quais linguagens o navegador pode aceitar?.....	11
2.3) Qual endereço IP do seu computador e do servidor?.....	12
2.4) Qual aplicação está sendo utilizada no servidor?.....	13
Pacotes TCP:.....	14
2.5) Qual o No. que contém o segmento TCP SYN que caracteriza o início de uma conexão TCP entre o seu computador e o site que você visitou?.....	15
2.6) Qual é o tamanho da janela em bytes?.....	15
2.7) Identifique, abra e mostre o pacote do segmento TCP responsável pelo término da conexão. Qual é a flag que o TCP usa para encerrar a conexão? Mostre e diga qual é essa flag. ?.....	16
<b>Parte 3 - Análise do Protocolo TCP:</b>	<b>17</b>
Objetivo.....	17
Procedimentos Realizados.....	18
Three-Way Handshake (SYN, SYN-ACK, ACK).....	18
<b>Parte 4 - Análise do Protocolo DNS:</b>	<b>21</b>
Objetivo.....	21
Respostas às Questões:.....	21
4.1) Qual o endereço IP do servidor DNS que resolveu o nome do host?.....	21
<b>Parte 5 - Análise do Protocolo Ethernet:</b>	<b>23</b>
Endereços MAC meu e Site <a href="http://www.florestal.mg.gov.br">www.florestal.mg.gov.br</a> :.....	23
<b>Conclusão:</b>	<b>25</b>

# Parte 1 - Ambientação do Wireshark:

## 1. Descrição do software

O Wireshark é um analisador de pacotes de rede em tempo real. O software permite capturar e inspecionar os dados que trafegam pela interface de rede do computador. Alguns dos protocolos que são exibidos são HTTP, DNS, UDP e ICMP, entre outros.

A interface do Wireshark é composta por:

- Tela inicial com opções de entrada de tipo de conexão à ser capturada (Wifi, Ethernet)
- Uma barra superior com botões de controle de captura
- Uma janela de detalhes dos protocolos por pacote

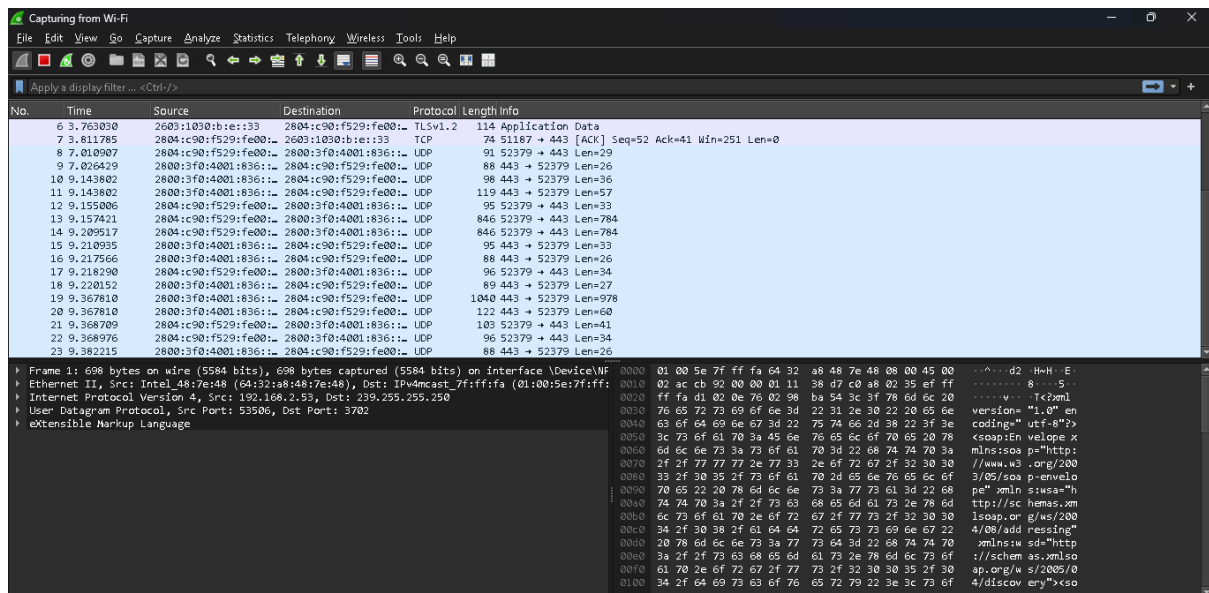


Figura 1 - Tela de captura de tráfego de rede

## 2. Captura de teste

Após a limpeza de DNS no terminal, foram acessados os sites:

- [www.prefeitura.pbh.gov.br](http://www.prefeitura.pbh.gov.br) ○ [www.florestal.mg.gov.br](http://www.florestal.mg.gov.br)

Foi criado um filtro para o site da Prefeitura de BH. Para isso, um comando de ping foi executado no terminal para identificar o IP correspondente do site.

```
C:\Users\mathe>ping www.prefeitura.pbh.gov.br

Disparando prefeitura.pbh.gov.br.cdn.gocache.net [170.82.174.5] com 32 bytes de dados:
Resposta de 170.82.174.5: bytes=32 tempo=11ms TTL=50
Resposta de 170.82.174.5: bytes=32 tempo=17ms TTL=50
Resposta de 170.82.174.5: bytes=32 tempo=10ms TTL=50
Resposta de 170.82.174.5: bytes=32 tempo=11ms TTL=50

Estatísticas do Ping para 170.82.174.5:
    Pacotes: Enviados = 4, Recebidos = 4, Perdidos = 0 (0% de perda),
    Aproximar um número redondo de vezes em milissegundos:
    Mínimo = 10ms, Máximo = 17ms, Média = 12ms
```

Figura 2 - Identificação do IP da rede do site da Prefeitura de BH

Com o IP identificado como **170.82.174.5**, foi criado o filtro utilizando o comando **ip.addr == 170.82.174.5** na barra de filtragem.

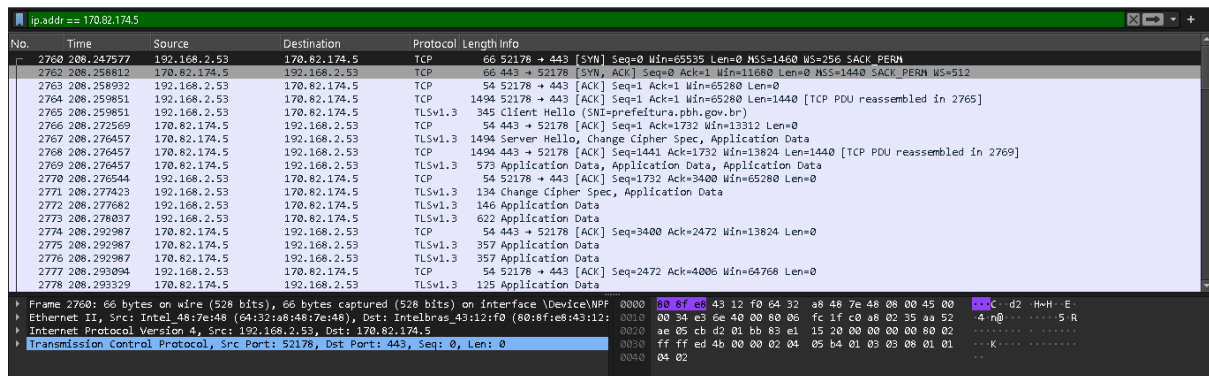


Figura 3 - Filtragem da Rede para o IP da Prefeitura de BH

Depois o filtro foi salvo no sistema como “Prefeitura BH”.

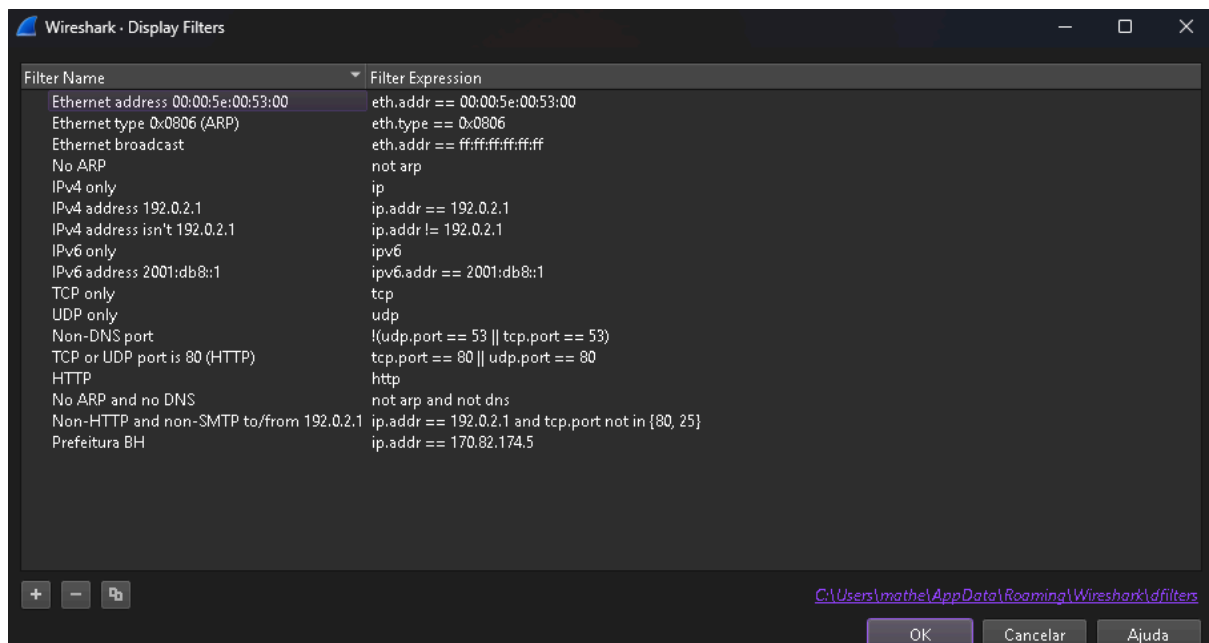


Figura 4 - IP salvo no sistema

### 3. Cores de cada protocolo padrão no Wireshark

O Wireshark possui algumas cores padronizadas para cada tipo de protocolo capturado na rede.

Como por exemplo:

- UDP: azul ciano
- TCP: cinza claro
- HTTP: verde claro
- ICMP: roxo claro

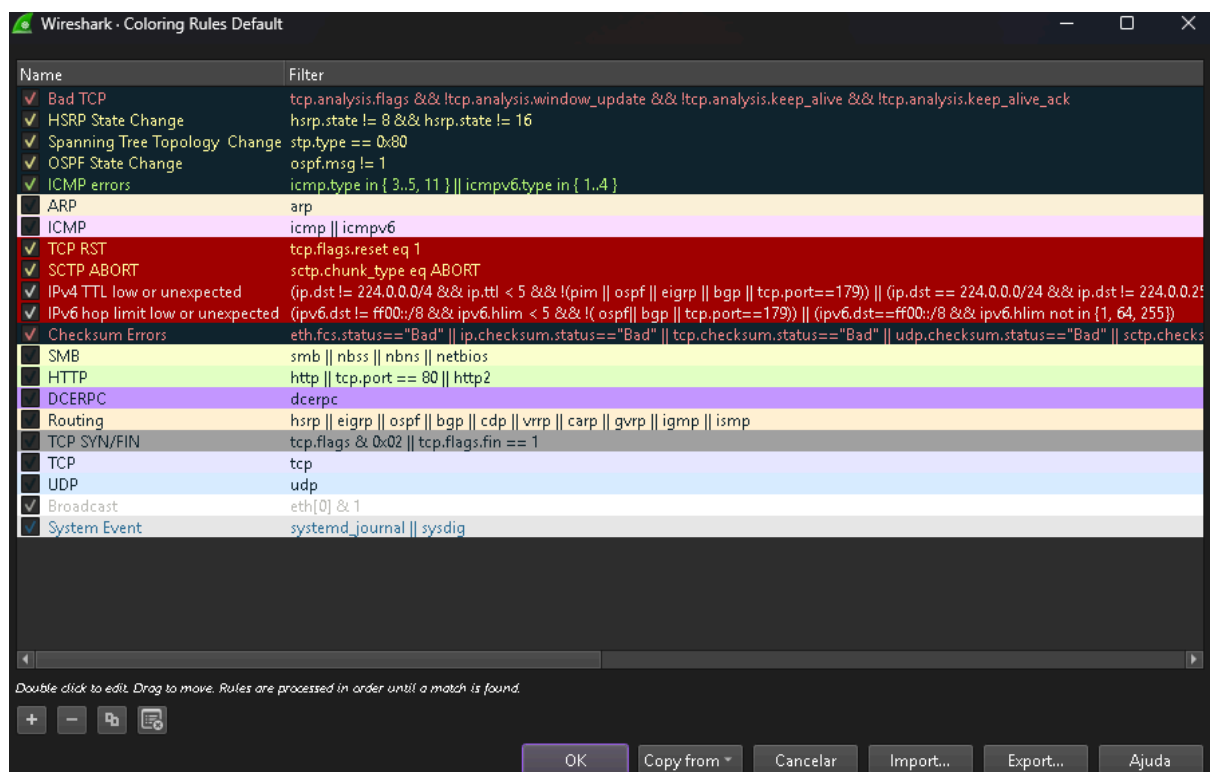


Figura 5 - Cores dos protocolos capturados no Wireshark

## 4. Captura automática de arquivos

### a) Criação de 10 arquivos a cada 20 segundos:

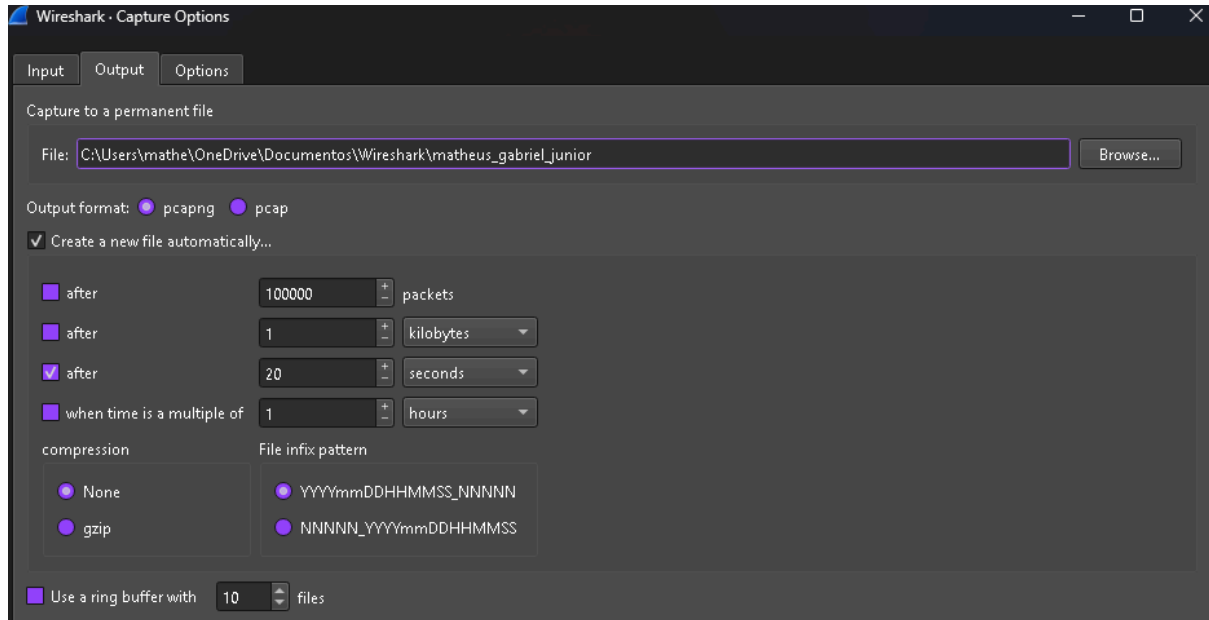


Figura 6 - Configuração do output para a parte (a)

**\* Obs: Também foi configurado o limite de 10 arquivos na aba "Options"**

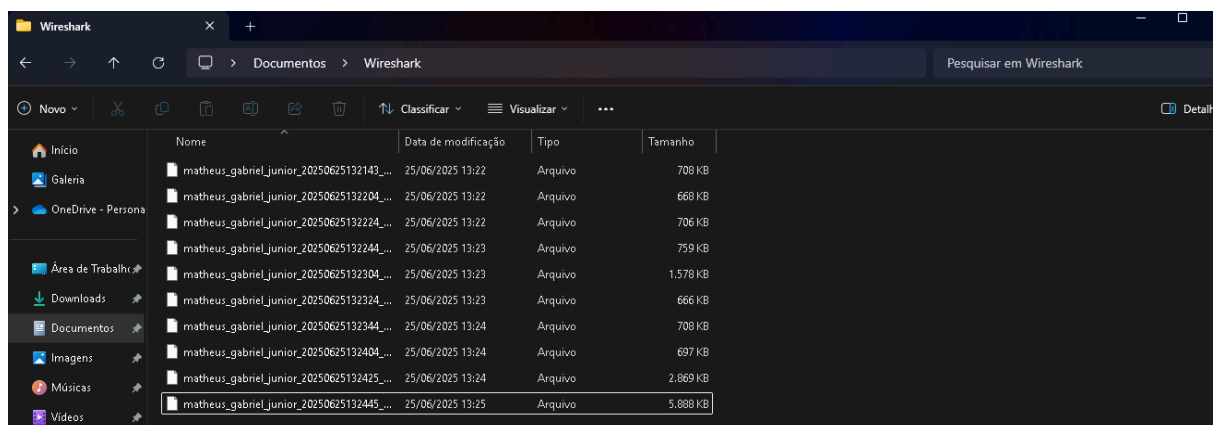
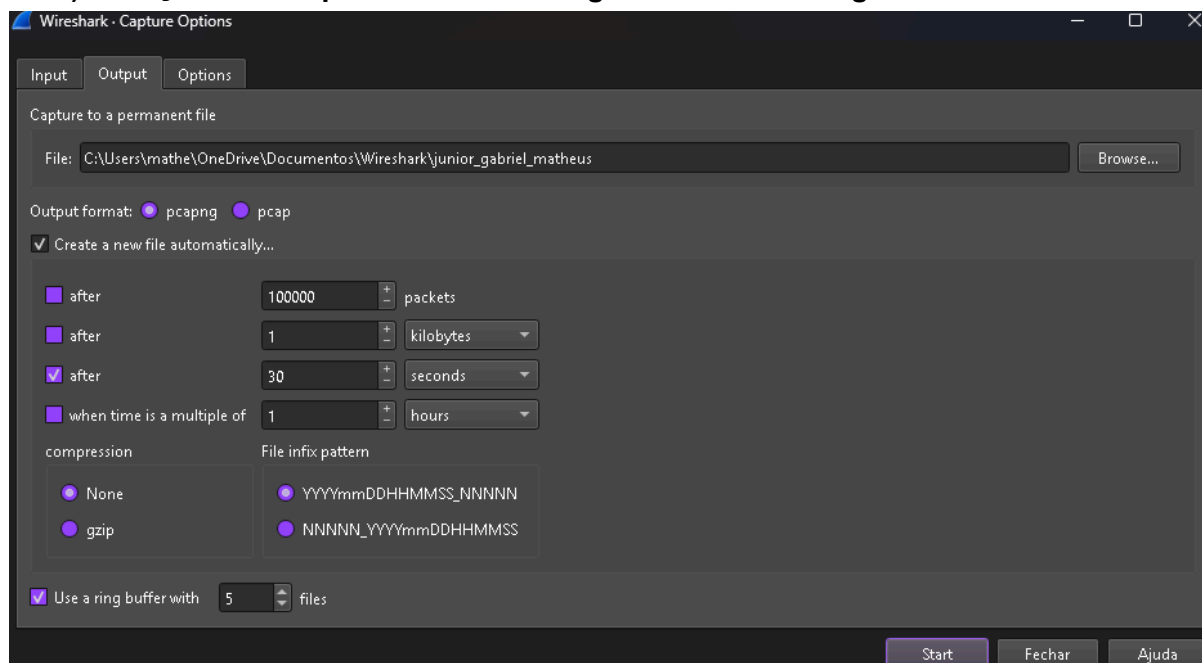


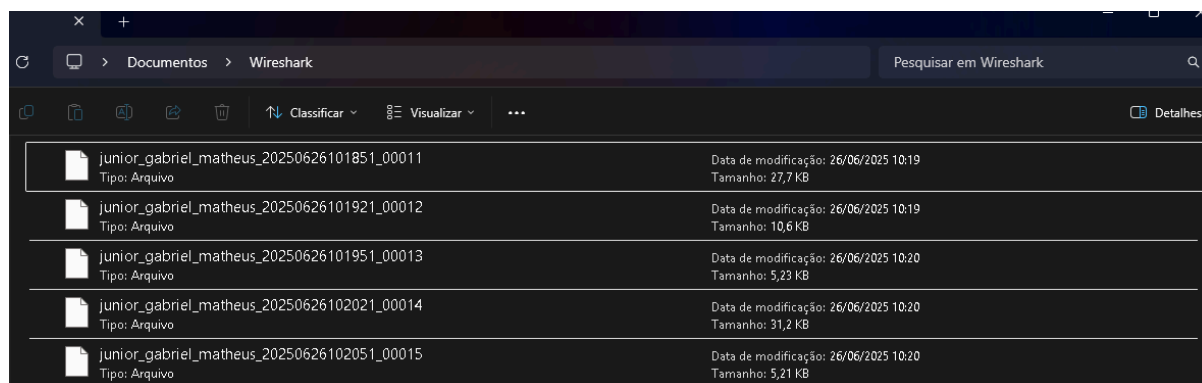
Figura 7 - Arquivos gerados para parte (a)

**b) Criação de 5 arquivos a cada 30 segundos usando ring buffer:**



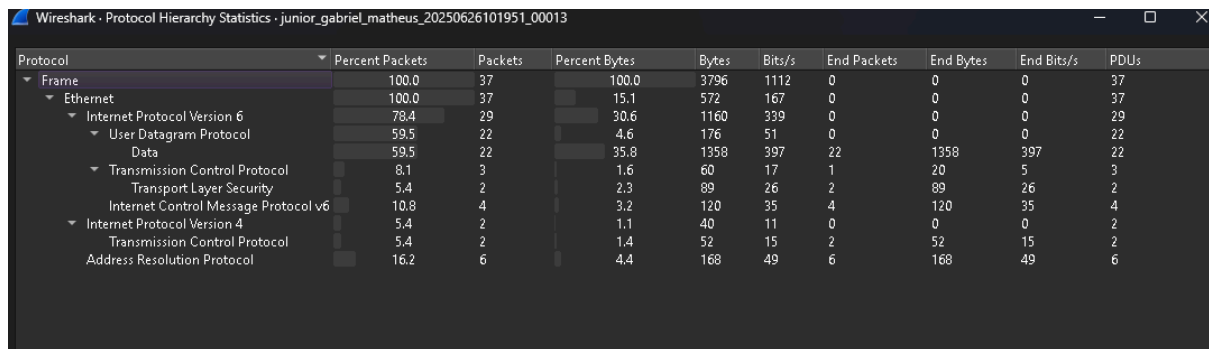
*Figura 8 - Configuração do output para a parte (b)*

Após a configuração da opção **Output**, foram gerados cinco arquivos automaticamente. Como foi utilizado o recurso de **ring buffer**, a cada 30 segundos um novo arquivo era criado. Após a geração dos cinco primeiros, os arquivos anteriores passaram a ser **sobrescritos**. A captura foi interrompida quando a sequência alcançou o arquivo de número 15.



*Figura 9 - Arquivos gerados para a parte (b)*

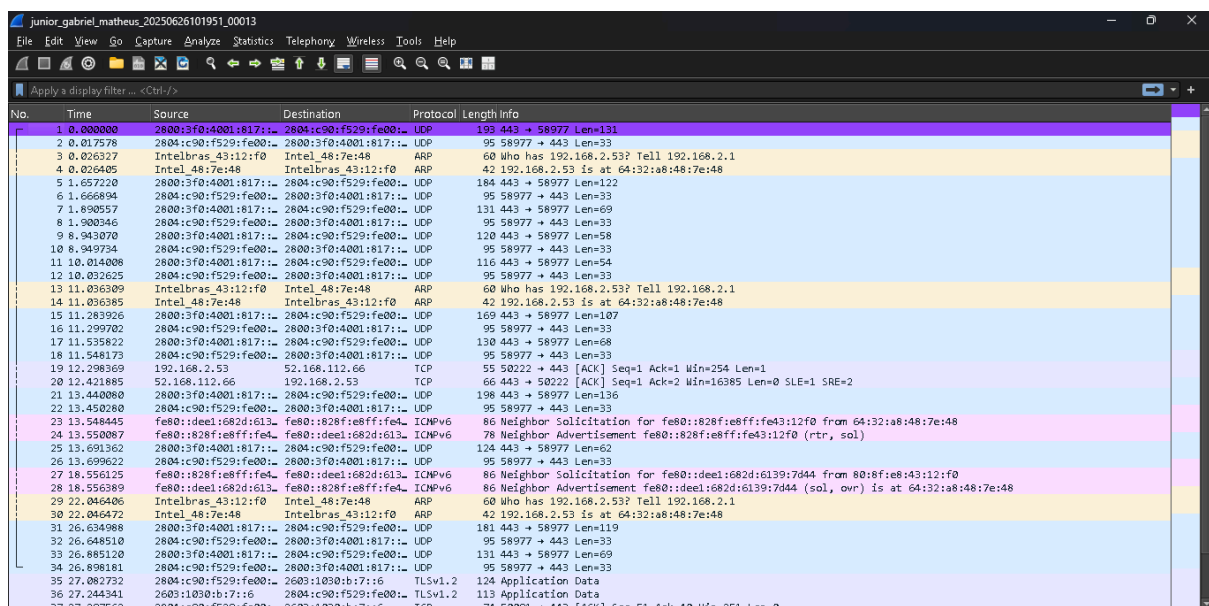
Para visualização da estatística dos protocolos hierarquicamente e a os protocolos listados foi utilizado o arquivo número 13.



Protocol	Percent Packets	Packets	Percent Bytes	Bytes	Bits/s	End Packets	End Bytes	End Bits/s	PDU's
Frame	100.0	37	100.0	3796	1112	0	0	0	37
Ethernet	100.0	37	15.1	572	167	0	0	0	37
Internet Protocol Version 6	78.4	29	30.6	1160	339	0	0	0	29
User Datagram Protocol	59.5	22	4.6	176	51	0	0	0	22
Data	59.5	22	35.8	1358	397	22	1358	397	22
Transmission Control Protocol	8.1	3	1.6	60	17	1	20	5	3
Transport Layer Security	5.4	2	2.3	89	26	2	89	26	2
Internet Control Message Protocol v6	10.8	4	3.2	120	35	4	120	35	4
Internet Protocol Version 4	5.4	2	1.1	40	11	0	0	0	2
Transmission Control Protocol	5.4	2	1.4	52	15	2	52	15	2
Address Resolution Protocol	16.2	6	4.4	168	49	6	168	49	6

Figura 10 - visualização da estatística de protocolos hierárquico

Os protocolos listados foram: UDP, ARP, TCP, ICMPv6 e TLSv1.2, como pode ser observado na imagem abaixo.



No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	2804:3f0:4001:817::	2804:c90:f529:fe00::	UDP	193	443 → 58977 Len=131
2	0.017578	2804:c90:f529:fe00::	2804:3f0:4001:817::	UDP	95	58977 → 443 Len=33
3	0.026327	Intelbras_43:12:f0	Intel_48:7e:48	ARP	60	Who has 192.168.2.53? Tell 192.168.2.1
4	0.026405	Intel_48:7e:48	Intelbras_43:12:f0	ARP	42	192.168.2.53 is at 64:32:a8:48:7e:48
5	1.657220	2804:3f0:4001:817::	2804:c90:f529:fe00::	UDP	184	443 → 58977 Len=122
6	1.668894	2804:c90:f529:fe00::	2804:3f0:4001:817::	UDP	95	58977 → 443 Len=33
7	1.890557	2804:3f0:4001:817::	2804:c90:f529:fe00::	UDP	131	443 → 58977 Len=69
8	1.900346	2804:c90:f529:fe00::	2804:3f0:4001:817::	UDP	95	58977 → 443 Len=33
9	8.943070	2804:3f0:4001:817::	2804:c90:f529:fe00::	UDP	120	443 → 58977 Len=58
10	8.949734	2804:c90:f529:fe00::	2804:3f0:4001:817::	UDP	95	58977 → 443 Len=33
11	10.014008	2804:3f0:4001:817::	2804:c90:f529:fe00::	UDP	116	443 → 58977 Len=54
12	10.032625	2804:c90:f529:fe00::	2804:3f0:4001:817::	UDP	95	58977 → 443 Len=33
13	11.036309	Intelbras_43:12:f0	Intel_48:7e:48	ARP	60	Who has 192.168.2.53? Tell 192.168.2.1
14	11.036385	Intel_48:7e:48	Intelbras_43:12:f0	ARP	42	192.168.2.53 is at 64:32:a8:48:7e:48
15	11.283926	2804:3f0:4001:817::	2804:c90:f529:fe00::	UDP	169	443 → 58977 Len=107
16	11.299702	2804:c90:f529:fe00::	2804:3f0:4001:817::	UDP	95	58977 → 443 Len=33
17	11.535922	2804:3f0:4001:817::	2804:c90:f529:fe00::	UDP	130	443 → 58977 Len=68
18	11.548173	2804:c90:f529:fe00::	2804:3f0:4001:817::	UDP	95	58977 → 443 Len=33
19	12.298369	192.168.2.53	52.168.112.66	TCP	55	50222 → 443 [ACK] Seq=1 Ack=1 Win=254 Len=1
20	12.421885	52.168.112.66	192.168.2.53	TCP	66	443 → 50222 [ACK] Seq=1 Ack=2 Win=16385 Len=0 SLE=1 SRE=2
21	13.440080	2804:3f0:4001:817::	2804:c90:f529:fe00::	UDP	198	443 → 58977 Len=136
22	13.450280	2804:c90:f529:fe00::	2804:3f0:4001:817::	UDP	95	58977 → 443 Len=33
23	13.541645	fe80::828f:e8ff:fe43:12f0	fe80::828f:e8ff:fe43:12f0	ICMPv6	86	Neighbor Solicitation for fe80::828f:e8ff:fe43:12f0 from 64:32:a8:48:7e:48
24	13.550087	fe80::828f:e8ff:fe43:12f0	fe80::828f:e8ff:fe43:12f0	ICMPv6	78	Neighbor Advertisement fe80::828f:e8ff:fe43:12f0 (rtt, sol)
25	13.691362	2804:3f0:4001:817::	2804:c90:f529:fe00::	UDP	124	443 → 58977 Len=62
26	13.699622	2804:c90:f529:fe00::	2804:3f0:4001:817::	UDP	95	58977 → 443 Len=33
27	18.556125	fe80::828f:e8ff:fe43:12f0	fe80::828f:e8ff:fe43:12f0	ICMPv6	86	Neighbor Solicitation for fe80::828f:e8ff:fe43:12f0 from 80:8f:e8:43:12:f0
28	18.556399	fe80::828f:e8ff:fe43:12f0	fe80::828f:e8ff:fe43:12f0	ICMPv6	86	Neighbor Advertisement fe80::828f:e8ff:fe43:12f0 (sol, ovr) is at 64:32:a8:48:7e:48
29	22.046406	Intelbras_43:12:f0	Intel_48:7e:48	ARP	60	Who has 192.168.2.53? Tell 192.168.2.1
30	22.046472	Intel_48:7e:48	Intelbras_43:12:f0	ARP	42	192.168.2.53 is at 64:32:a8:48:7e:48
31	26.634988	2804:3f0:4001:817::	2804:c90:f529:fe00::	UDP	181	443 → 58977 Len=119
32	26.648510	2804:c90:f529:fe00::	2804:3f0:4001:817::	UDP	95	58977 → 443 Len=33
33	26.885120	2804:3f0:4001:817::	2804:c90:f529:fe00::	UDP	131	443 → 58977 Len=69
34	26.898161	2804:c90:f529:fe00::	2804:3f0:4001:817::	UDP	95	58977 → 443 Len=33
35	27.082732	2804:c90:f529:fe00::	2603:1030:b7:16	TLSv1.2	124	Application Data
36	27.244341	2603:1030:b7:16	2804:c90:f529:fe00::	TLSv1.2	113	Application Data
37	27.297562	2804:c90:f529:fe00::	2603:1030:b7:16	TCP	74	50091 → 443 [ACK] Seq=51 Ack=40 Win=251 Len=0

Figura 11 - Lista de protocolos do arquivo

## Parte 2 - Análise Do Protocolo HTTP:

Objetivo: Analisar o comportamento básico do protocolo HTTP, observando o processo de requisição e resposta.

O site utilizado nesta análise foi: <http://www.icec.edu.br>, conforme definido para o Grupo 7

**Problema:** o link deixado na especificação era de um site que usava protocolo HTTPS e não HTTP, fizemos varias buscas e até achamos que o erro podia estar em nosso software (wireshark), mas por fim conseguimos entender que por se tratar de um site com protocolo de segurança, ele não seria visível como um HTTP normal/ simples, portanto fizemos com



um que estava disponível, esse foi um servidor do ubuntu , que testa conectividade de internet.

Mais tarde, foi disponibilizado pelos monitores um site que realmente usava HTTP e não HTTPS, portando algumas respostas abaixo foram feitas após o “término” do trabalho e apenas estão adicionando às perguntas informações.

Através da captura do pacote HTTP, podemos ter acesso a aba de Hypertext Transfer Protocol:

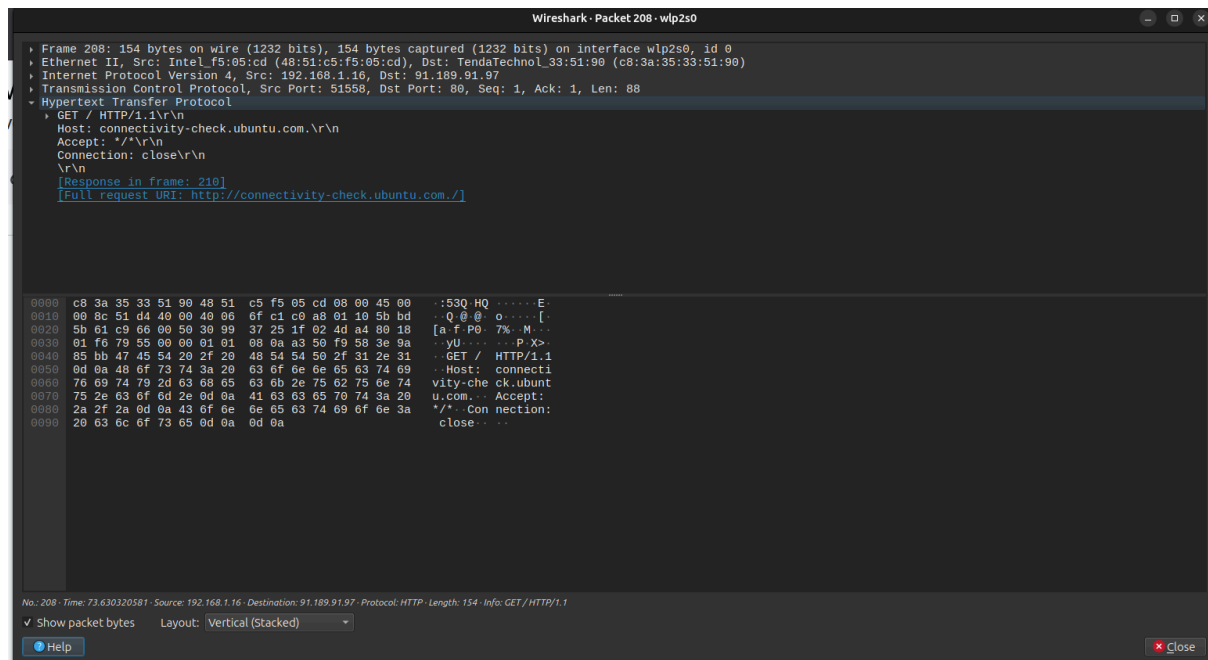


Figura 12 - visualização do Hypertext Transfer Protocol

Ping e IP do site disponibilizado posteriormente:

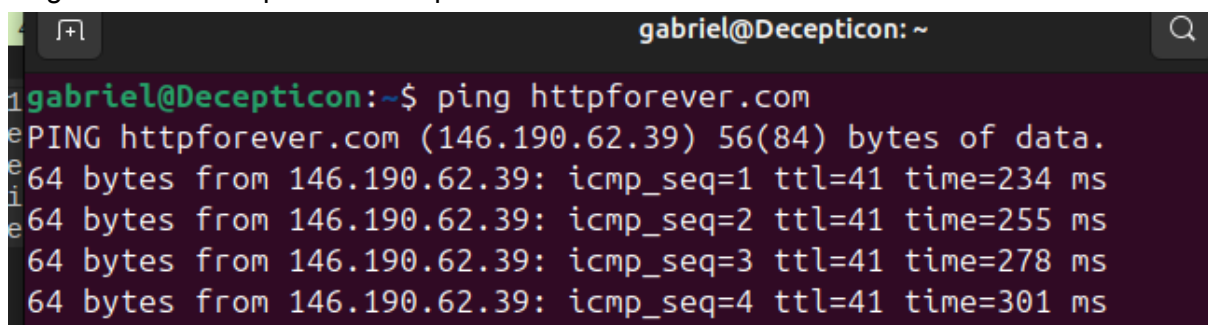


Figura 13 - Visualização do ping

No.	Time	Source	Destination	Protocol	Length	Info
49	4.207899880	23.223.205.175	192.168.1.16	HTTP	462	[TCP Previous segment not captured] Continuation
53	4.242661120	192.168.1.16	23.223.205.175	HTTP	333	GET / HTTP/1.1
90	4.265961802	23.223.205.175	192.168.1.16	HTTP	393	[TCP Previous segment not captured] Continuation
91	4.863761011	192.168.1.16	146.199.62.39	HTTP	459	GET / HTTP/1.1
133	5.111745867	146.199.62.39	192.168.1.16	HTTP	1377	[TCP Previous segment not captured] Continuation
140	5.207213169	192.168.1.16	146.199.62.39	HTTP	304	GET /js/init.min.js HTTP/1.1
163	5.413471975	192.168.1.16	146.199.62.39	HTTP	462	GET /css/style.min.css HTTP/1.1
191	5.507029945	192.168.1.16	146.199.62.39	HTTP	407	GET /css/style-wide.min.css HTTP/1.1
193	5.725695924	146.199.62.39	192.168.1.16	HTTP	2922	Continuation
210	5.726283220	146.199.62.39	192.168.1.16	HTTP	968	HTTP/1.1 200 OK (text/css)
281	6.203975163	192.168.1.16	146.199.62.39	HTTP	469	GET /css/images/banner.svg HTTP/1.1
282	6.203975970	192.168.1.16	146.199.62.39	HTTP	484	GET /css/images/header-major-on-light.svg HTTP/1.1
283	6.204080784	192.168.1.16	146.199.62.39	HTTP	483	GET /css/images/header-major-on-dark.svg HTTP/1.1
482	6.402504262	146.199.62.39	192.168.1.16	HTTP/XML	1377	Continuation
483	6.402505294	146.199.62.39	192.168.1.16	HTTP/XML	1333	HTTP/1.1 200 OK
485	6.402513649	146.199.62.39	192.168.1.16	HTTP/XML	1327	HTTP/1.1 200 OK
489	6.406637731	192.168.1.16	146.199.62.39	HTTP	462	GET /favicon.ico HTTP/1.1
492	6.648632202	146.199.62.39	192.168.1.16	HTTP	1494	[TCP Previous segment not captured] Continuation
493	6.648632823	146.199.62.39	192.168.1.16	HTTP	879	[TCP Previous segment not captured] Continuation
10935	183.696924607	192.168.1.16	185.125.199.97	HTTP	154	GET / HTTP/1.1
10937	184.696666594	185.125.199.97	192.168.1.16	HTTP	251	HTTP/1.1 204 No Content
14973	483.648037864	192.168.1.16	185.125.199.48	HTTP	154	GET / HTTP/1.1
14986	483.842063895	185.125.199.48	192.168.1.16	HTTP	255	HTTP/1.1 204 No Content
22820	783.666055757	192.168.1.16	91.189.91.98	HTTP	154	GET / HTTP/1.1
22821	783.864984962	91.189.91.98	192.168.1.16	HTTP	251	HTTP/1.1 204 No Content

Figura 14 - Tráfego da rede

```

> Frame 91: 459 bytes on wire (3672 bits), 459 bytes captured (3672 bits) on interface wlp2s0, id 0
> Ethernet II, Src: Intel_f5:05:cd (48:51:c5:f5:05:cd), Dst: TendaTechnol_33:51:90 (c8:3a:35:33:51:90)
> Internet Protocol Version 4, Src: 192.168.1.16, Dst: 146.199.62.39
> Transmission Control Protocol, Src Port: 42150, Dst Port: 80, Seq: 1, Ack: 1, Len: 393
> Hypertext Transfer Protocol
  > GET / HTTP/1.1\r\n
    Host: httpforever.com\r\n
    Connection: keep-alive\r\n
    Upgrade-Insecure-Requests: 1\r\n
    User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/137.0.0.0 Safari/537.36\r\n
    Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8\r\n
    Sec-GPC: 1\r\n
    Accept-Language: pt-BR,pt;q=0.9\r\n
    Accept-Encoding: gzip, deflate\r\n
    \r\n
    [Full request URI: http://httpforever.com/]

```

Figura 15 - Visualização do ypertext Transfer Protocol

## 2.1) Qual versão do http está sendo usada?

Para dizermos a versão que está sendo usada basta olharmos na frente do “GET”, neste exemplo fica claro ser um de pacote de versão HTTP/1.1

208	73.630320581	192.168.1.16	91.189.91.97	HTTP	154 GET / HTTP/1.1
210	73.772572458	91.189.91.97	192.168.1.16	HTTP	251 HTTP/1.1 204 No Content

Figura 16 - Versão do HTTP

A versão do HHTTP posteriormente disponibilizado também era HTTP/1.1, como podemos ver logo a frente do texto “GET” na linha 6.

```

> Frame 91: 459 bytes on wire (3672 bits), 459 bytes captured (3672 bits) on interface wlp2s0, id 0
> Ethernet II, Src: Intel_f5:05:cd (48:51:c5:f5:05:cd), Dst: TendaTechnol_33:51:90 (c8:3a:35:33:51:90)
> Internet Protocol Version 4, Src: 192.168.1.16, Dst: 146.199.62.39
> Transmission Control Protocol, Src Port: 42150, Dst Port: 80, Seq: 1, Ack: 1, Len: 393
> Hypertext Transfer Protocol
  > GET / HTTP/1.1\r\n
    Host: httpforever.com\r\n
    Connection: keep-alive\r\n
    Upgrade-Insecure-Requests: 1\r\n
    User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/137.0.0.0 Safari/537.36\r\n
    Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8\r\n
    Sec-GPC: 1\r\n
    Accept-Language: pt-BR,pt;q=0.9\r\n
    Accept-Encoding: gzip, deflate\r\n
    \r\n
    [Full request URI: http://httpforever.com/]

```

Figura 17 - Versão do HTTP (Hypertext)

## 2.2) Quais linguagens o navegador pode aceitar?

Para responder essa pergunta, se o pacote for HTTP há uma parte onde nos informam a linguagem definida, que seria logo a frente de “Accept:” nos campos de Hypertext Transfer Protocol. Algo do tipo deve aparecer :

Accept-Language: pt-BR,pt;q=0.9,en-US;q=0.8,en;q=0.7

Embora abaixo exemplifico com um pacote HTTP, onde mostra que “Accept” não “define” uma linguagem, mas na verdade está dizendo que aceita qualquer tipo de linguagem:

```

> Frame 208: 154 bytes on wire (1232 bits), 154 bytes captured (1232 bits) on interface wlp2s0, id 0
> Ethernet II, Src: Intel_f5:05:cd (48:51:c5:f5:05:cd), Dst: TendaTechnol_33:51:90 (c8:3a:35:33:51:90)
> Internet Protocol Version 4, Src: 192.168.1.16, Dst: 91.189.91.97
> Transmission Control Protocol, Src Port: 51558, Dst Port: 80, Seq: 1, Ack: 1, Len: 88
> Hypertext Transfer Protocol
  > GET / HTTP/1.1\r\n
    Host: connectivity-check.ubuntu.com.\r\n
    Accept: */*\r\n
    Connection: close\r\n
    \r\n
    [Response in frame: 210]
    [Full request URI: http://connectivity-check.ubuntu.com/]

```

Figura 18 - Linguagem do navegador

Accept : \*/\* indica que o navegador aceita qualquer tipo de conteúdo (por exemplo, HTML, JSON, texto simples etc.), mas não especifica a linguagem humana (como português, inglês etc.).

As linguagens aceitas pelo navegador em comunicação com o site disponibilizado posteriormente estão dispostas logo após as linhas que se iniciam com “Accept”:  
Elas indicam que é aceito:

- Textos do html , xml , imagens png :  
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,\*/\*;q=0.8
- linguagem humana - Brasileira: pt-BR,pt;q=0.9
- tipo de código, no caso Zip: gzip, deflate

```

> Frame 91: 459 bytes on wire (3672 bits), 459 bytes captured (3672 bits) on interface wlp2s0, id 0
> Ethernet II, Src: Intel_f5:05:cd (48:51:c5:f5:05:cd), Dst: TendaTechnol_33:51:90 (c8:3a:35:33:51:90)
> Internet Protocol Version 4, Src: 192.168.1.16, Dst: 146.190.62.39
> Transmission Control Protocol, Src Port: 42150, Dst Port: 80, Seq: 1, Ack: 1, Len: 393
> Hypertext Transfer Protocol
  > GET / HTTP/1.1\r\n
    Host: httpforever.com\r\n
    Connection: keep-alive\r\n
    Upgrade-Insecure-Requests: 1\r\n
    User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/137.0.0.0 Safari/537.36\r\n
    Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8\r\n
    Sec-GPC: 1\r\n
    Accept-Language: pt-BR,pt;q=0.9\r\n
    Accept-Encoding: gzip, deflate\r\n
    \r\n
    [Full request URI: http://httpforever.com/]

```

Figura 19 - Visualização do Hypertext Transfer Protocol

### 2.3) Qual endereço IP do seu computador e do servidor?

Neste exemplo que estou usando de HTTP, que é um server do ubuntu o endereço de ip deles é o 91.189.91.97, enquanto o meu é 192.168.1.16

Como era esperado, o meu apareceu primeiro destinando o servidor. Ordem essa normal, já que meu ip está solicitando algo ao servidor ubuntu.

Source	Destination	Protocol
192.168.1.16	91.189.91.97	HTTP
91.189.91.97	192.168.1.16	HTTP

Figura 20 - Visualização do IP de origem e servidor

O site deixado na especificação do tp tem o ip definido por 200.174.103.36.

```
gabriel@Decepticon:~$ ping www.icec.edu.br
PING www.icec.edu.br (200.174.103.36) 56(84) bytes of data.
```

Figura 21 - IP do site especificado

No site disponibilizado posteriormente, podemos ver o IP tanto do site quanto de minha máquina pela visão geral (source → Quem enviou e Destination → para quem vai ) ou pela informação no pacote (imagem preta - penúltima e antepenúltima linha)

- meu IP: 192.168.1.16
- IP do site: 146.190.62.39

```
Internet Protocol Version 4, Src: 192.168.1.16, Dst: 146.190.62.39
  0100 .... = Version: 4
  .... 0101 = Header Length: 20 bytes (5)
  Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
    Total Length: 445
    Identification: 0xbc0f (48143)
  010. .... = Flags: 0x2, Don't fragment
    ...0 0000 0000 0000 = Fragment Offset: 0
    Time to Live: 64
    Protocol: TCP (6)
    Header Checksum: 0xea8d [validation disabled]
    [Header checksum status: Unverified]
    Source Address: 192.168.1.16
    Destination Address: 146.190.62.39
    [Stream index: 2]
```

Figura 22 - informação específica

```
gabriel@Decepticon:~$ ping httpforever.com
PING httpforever.com (146.190.62.39) 56(84) bytes of data.
64 bytes from 146.190.62.39: icmp_seq=1 ttl=41 time=234 ms
64 bytes from 146.190.62.39: icmp_seq=2 ttl=41 time=255 ms
```

Figura 23 - ping e IP do site da Especificação por terminal

Source	Destination	Protocol
146.190.62.39	192.168.1.16	HTTP
192.168.1.16	146.190.62.39	HTTP
192.168.1.16	146.190.62.39	HTTP
192.168.1.16	146.190.62.39	HTTP
146.190.62.39	192.168.1.16	HTTP
146.190.62.39	192.168.1.16	HTTP
192.168.1.16	146.190.62.39	HTTP

Figura 24 - visão geral dos IP de envio e recebimento no wireshark

## 2.4) Qual aplicação está sendo utilizada no servidor?

Podemos ver que a aplicação feita ao servidor é de request (pedido ou “perguntas”) ao servidor, onde ele apenas fornece algo pra mim.

```

Frame 210: 251 bytes on wire (2008 bits), 251 bytes captured (2008 bits) on interface wlp2s0, id 0
Ethernet II, Src: TendaTechnol_33:51:90 (c8:3a:35:33:51:90), Dst: Intel_f5:05:cd (48:51:c5:f5:05:cd)
Internet Protocol Version 4, Src: 91.189.91.97, Dst: 192.168.1.16
Transmission Control Protocol, Src Port: 80, Dst Port: 51558, Seq: 1, Ack: 89, Len: 185
Hypertext Transfer Protocol
  HTTP/1.1 204 No Content\r\n
    server: nginx/1.18.0 (Ubuntu)\r\n
    date: Thu, 26 Jun 2025 11:56:58 GMT\r\n
    x-cache-status: from content-cache/1\r\n
    x-networkmanager-status: online\r\n
    connection: close\r\n
    \r\n
    [Request in frame: 208]
    [Time since request: 0.142251877 seconds]
    [Request URI: /]
    [Full request URI: http://connectivity-check.ubuntu.com./]

```

Figura 25 - Aplicação utilizada no servidor (1)

Essa URL ( <http://connectivity-check.ubuntu.com./>) indica que o serviço acessado faz parte do Ubuntu Connectivity Check, que é um recurso do Ubuntu para verificar conexão com a Internet.

```

[Request in frame: 208]
[Time since request: 0.142251877 seconds]
[Request URI: /]
[Full request URI: http://connectivity-check.ubuntu.com./]

```

Figura 26 - Aplicação utilizada no servidor (2)

Logo, a aplicação no servidor é:  
connectivity-check.ubuntu.com, usada pelo Ubuntu para testes de conectividade.

Para o site posteriormente passado, a finalidade com o servidor se mostrou igual. Apenas request eram feitas, como mostra a última linha:

```
▶ Frame 91: 459 bytes on wire (3672 bits), 459 bytes captured (3672 bits) on interface wlp2s0, id
▶ Ethernet II, Src: Intel_f5:05:cd (48:51:c5:f5:05:cd), Dst: TendaTechnol_33:51:90 (c8:3a:35:33:5
▶ Internet Protocol Version 4, Src: 192.168.1.16, Dst: 146.190.62.39
▶ Transmission Control Protocol, Src Port: 42150, Dst Port: 80, Seq: 1, Ack: 1, Len: 393
▼ Hypertext Transfer Protocol
  ▶ GET / HTTP/1.1\r\n
    Host: httpforever.com\r\n
    Connection: keep-alive\r\n
    Upgrade-Insecure-Requests: 1\r\n
    User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/137
    Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apn
    Sec-GPC: 1\r\n
    Accept-Language: pt-BR,pt;q=0.9\r\n
    Accept-Encoding: gzip, deflate\r\n
    \r\n
    [Full request URI: http://httpforever.com/]
```

Figura 27 - como estava sendo utilizada as chamadas/ conexões com o servidor

## Pacotes TCP:

Acima, respondemos usando os pacotes do protocolo HTTP. Para os pacotes de TCP, é possível usar o ip de nossas requisições para achar o site especificado no Trabalho prático, portanto assim será. Abaixo não há filtro por TCP e HTTP , e sim pelo IP de nosso site alvo.

Para saber o IP do site, fiz uma requisição de ping via terminal. Com a saída filtrei pelo ip no wireshark

```
gabriel@Decepticon:~$ ping www.icec.edu.br
PING www.icec.edu.br (200.174.103.36) 56(84) bytes of data.
```

Figura 28 - Visualização do IP do site ICEC através do ping

ip.addr == 200.174.103.36			
No.	Time	Source	Destination

Figura 29 - Filtro de IP

Após isso, usei as próprias colunas (metadados) do wireshark para ver apenas o protocolos de TPC.

## 2.5) Qual o No. que contém o segmento TCP SYN que caracteriza o início de uma conexão TCP entre o seu computador e o site que você visitou?

Abaixo vemos vários pacotes transacionados, onde os 2 cinzas presentes (abaixo do vermelho) são os pacotes com o bit SYN ligados.

No.	Time	Source	Destination	Protocol	Length	Info
44150	3772.224229206	192.168.1.16	200.174.103.36	ICMP	98	Echo (ping) request id=0x1f1d, seq=212/54272, ttl=64 (no response found!)
44151	3773.248229840	192.168.1.16	200.174.103.36	ICMP	98	Echo (ping) request id=0x1f1d, seq=213/54528, ttl=64 (no response found!)
44154	3774.272387487	192.168.1.16	200.174.103.36	ICMP	98	Echo (ping) request id=0x1f1d, seq=214/54784, ttl=64 (no response found!)
44155	3775.296242003	192.168.1.16	200.174.103.36	ICMP	98	Echo (ping) request id=0x1f1d, seq=215/55040, ttl=64 (no response found!)
44156	3776.320231394	192.168.1.16	200.174.103.36	ICMP	98	Echo (ping) request id=0x1f1d, seq=216/55296, ttl=64 (no response found!)
44157	3777.343948030	192.168.1.16	200.174.103.36	ICMP	98	Echo (ping) request id=0x1f1d, seq=217/55552, ttl=64 (no response found!)
44158	3778.367963584	192.168.1.16	200.174.103.36	ICMP	98	Echo (ping) request id=0x1f1d, seq=218/55808, ttl=64 (no response found!)
44159	3779.392160943	192.168.1.16	200.174.103.36	ICMP	98	Echo (ping) request id=0x1f1d, seq=219/56064, ttl=64 (no response found!)
127	7.231931355	192.168.1.16	200.174.103.36	TCP	66	35402 → 443 [ACK] Seq=1 Ack=1 Win=433 Len=0 TSval=1364617154 TSecr=258949476
13	7.251318977	200.174.103.36	192.168.1.16	TCP	66	[TCP ACKed unseen segment] 443 → 35402 [ACK] Seq=1 Ack=2 Win=256 Len=0 TSval=258954382 TSecr=1364523130
90	33.394819693	200.174.103.36	192.168.1.16	TCP	54	443 → 35402 [RST, ACK] Seq=1 Ack=2 Win=0 Len=0
115	48.967614857	192.168.1.16	200.174.103.36	TCP	74	49050 → 443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM TSval=1364658889 TSecr=0 WS=128
117	48.994861606	200.174.103.36	192.168.1.16	TCP	74	443 → 49050 [SYN, ACK] Seq=0 Ack=1 Win=8192 Len=0 MSS=1440 WS=256 SACK_PERM TSval=258958557 TSecr=1364658889
118	48.994908080	192.168.1.16	200.174.103.36	TCP	66	49050 → 443 [ACK] Seq=1 Ack=1 Win=8192 Len=0 MSS=1440 WS=256 SACK_PERM TSval=258958557 TSecr=1364658889
120	49.013000294	200.174.103.36	192.168.1.16	TCP	70	[TCP Dup ACK 117] 443 → 49050 [ACK] Seq=1 Ack=1 Win=8192 Len=0 MSS=1440 WS=256 SACK_PERM TSval=258958559 TSecr=1364658891 SLE=1429 SRE=1022
121	49.013881271	200.174.103.36	192.168.1.16	TCP	66	443 → 49050 [ACK] Seq=1 Ack=122 Win=65536 Len=0 TSval=258958559 TSecr=1364658891

Figura 30 - Segmento TCP SYN (1)

Protocol	Length	Info
ICMP	98	Echo (ping) request id=0x1f1d, seq=212/54272, ttl=64 (no response found!)
ICMP	98	Echo (ping) request id=0x1f1d, seq=213/54528, ttl=64 (no response found!)
ICMP	98	Echo (ping) request id=0x1f1d, seq=214/54784, ttl=64 (no response found!)
ICMP	98	Echo (ping) request id=0x1f1d, seq=215/55040, ttl=64 (no response found!)
ICMP	98	Echo (ping) request id=0x1f1d, seq=216/55296, ttl=64 (no response found!)
ICMP	98	Echo (ping) request id=0x1f1d, seq=217/55552, ttl=64 (no response found!)
ICMP	98	Echo (ping) request id=0x1f1d, seq=218/55808, ttl=64 (no response found!)
ICMP	98	Echo (ping) request id=0x1f1d, seq=219/56064, ttl=64 (no response found!)
TCP	66	35402 → 443 [ACK] Seq=1 Ack=1 Win=433 Len=0 TSval=1364617154 TSecr=258949476
TCP	66	[TCP ACKed unseen segment] 443 → 35402 [ACK] Seq=1 Ack=2 Win=256 Len=0 TSval=258954382 TSecr=1364523130
TCP	54	443 → 35402 [RST, ACK] Seq=1 Ack=2 Win=0 Len=0
TCP	74	49050 → 443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM TSval=1364658889 TSecr=0 WS=128
TCP	74	443 → 49050 [SYN, ACK] Seq=0 Ack=1 Win=8192 Len=0 MSS=1440 WS=256 SACK_PERM TSval=258958557 TSecr=1364658889

Figura 31 - Segmento TCP SYN (2)

O número do Nó. deles são 115 e 117, como mostra a figura abaixo. Deve haver um cuidado na observação, já que ao lado, separado por singelos espaços vemos o número do time.

No.	Time
44150	3772.224229206
44151	3773.248229840
44154	3774.272387487
44155	3775.296242003
44156	3776.320231394
44157	3777.343948030
44158	3778.367963584
44159	3779.392160943
127	7.231931355
13	7.251318977
90	33.394819693
115	48.967614857
117	48.994861606

Figura 32 - Número do Nó

## 2.6) Qual é o tamanho da janela em bytes?

O tamanho deles em Byte é de 74 B.



ICMP	98 Echo (ping) request id=0x1f1d, seq=219/56064, ttl=64 (no response found!)
TCP	66 35402 → 443 [ACK] Seq=1 Ack=1 Win=433 Len=0 TSval=1364617154 TSecr=258949476
TCP	66 [TCP ACKed unseen segment] 443 → 35402 [ACK] Seq=1 Ack=2 Win=256 Len=0 TSval=258954382 TSecr=1364523130
TCP	54 443 → 35402 [RST, ACK] Seq=1 Ack=2 Win=0 Len=0
TCP	74 49050 → 443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM TSval=1364658889 TSecr=0 WS=128
TCP	74 443 → 49050 [SYN, ACK] Seq=0 Ack=1 Win=8192 Len=0 MSS=1440 WS=256 SACK_PERM TSval=258958557 TSecr=1364658889

Figura 33 - Tamanho da janela em bytes

Abaixo mostro também a caixa de informações de um deles, onde na primeira linha vemos o tamanho em bytes e na linha demarcada de azul vemos o bit SYN ligado:

```

> Frame 115: 74 bytes on wire (592 bits), 74 bytes captured (592 bits) on interface wlp2s0, id 0
> Ethernet II, Src: Intel_f5:05:cd (48:51:c5:f5:05:cd), Dst: TendaTechnol_33:51:90 (c8:3a:35:33:51:90)
> Internet Protocol Version 4, Src: 192.168.1.16, Dst: 200.174.103.36
> Transmission Control Protocol, Src Port: 49050, Dst Port: 443, Seq: 0, Len: 0
  Source Port: 49050
  Destination Port: 443
  [Stream index: 8]
  [Stream Packet Number: 1]
  [Conversation completeness: Complete, WITH_DATA (47)]
  [TCP Segment Len: 0]
  Sequence Number: 0 (relative sequence number)
  Sequence Number (raw): 3673764011
  [Next Sequence Number: 1 (relative sequence number)]
  Acknowledgment Number: 0
  Acknowledgment number (raw): 0
  1010 .... = Header Length: 40 bytes (10)
  Flags: 0x0002 (SYN)
  Window: 64240
  [Calculated window size: 64240]
  Checksum: 0xf1b9 [unverified]
  [Checksum Status: Unverified]
  Urgent Pointer: 0
  Options: (20 bytes), Maximum segment size, SACK permitted, Timestamps, No-Operation (NOP), Window sc...
  [Timestamps]

```

Figura 34 - Informações adicionais

2.7) Identifique, abra e mostre o pacote do segmento TCP responsável pelo término da conexão. Qual é a flag que o TCP usa para encerrar a conexão? Mostre e diga qual é essa flag.?

Para que o término da conexão acontecesse, eu primeiro devia ter que forçar isso, portanto foi preciso fechar a guia onde estava o site aberto. Com isso era esperado que a conexão se fechasse através da ativação do bit FIN.

Isto é Exatamente o que é mostrado nas últimas linhas das imagens abaixo:

No.	Time	Source	Destination	Protocol	Length	Info
60	45.004125840	192.168.1.16	200.174.103.36	TCP	78	[TCP Dup ACK 62w1] 45018 → 443 [ACK] Seq=1758 Ack=2857 Win=61440 Len=0 TSval=1371037484 TSecr=259056403 SLE=1429 SRE=2857
70	45.004148120	192.168.1.16	200.174.103.36	TCP	66	45018 → 443 [ACK] Seq=1758 Ack=2857 Win=61440 Len=0 TSval=1371037484 TSecr=259056403
71	45.004157758	192.168.1.16	200.174.103.36	TCP	66	45018 → 443 [ACK] Seq=1758 Ack=4901 Win=59392 Len=0 TSval=1371037484 TSecr=259056403
75	45.047506061	200.174.103.36	192.168.1.16	TCP	8654	443 → 45018 [ACK] Seq=5936 Ack=2633 Win=64768 Len=8568 TSval=259056408 TSecr=1371637507 [TCP PDU reassembled in 77]
76	45.048059941	192.168.1.16	200.174.103.36	TCP	66	45018 → 443 [ACK] Seq=2633 Ack=13604 Win=50944 Len=0 TSval=1371037528 TSecr=259056408
81	45.089223613	192.168.1.16	200.174.103.36	TCP	66	45018 → 443 [ACK] Seq=2633 Ack=14833 Win=49792 Len=0 TSval=1371037578 TSecr=259056408
107	45.389292331	200.174.103.36	192.168.1.16	TCP	7286	443 → 45018 [ACK] Seq=14833 Ack=3278 Win=65536 Len=7148 TSval=259056442 TSecr=1371637847 [TCP PDU reassembled in 111]
108	45.389293309	200.174.103.36	192.168.1.16	TCP	1295	[TCP Previous segment not captured] 443 → 45018 [PSH, ACK] Seq=23401 Ack=3278 Win=65536 Len=1229 TSval=259056442 TSecr=1371637847
109	45.389299908	192.168.1.16	200.174.103.36	TCP	66	45018 → 443 [ACK] Seq=3278 Ack=2193 Win=42752 Len=0 TSval=1371037870 TSecr=259056442
110	45.389302908	192.168.1.16	200.174.103.36	TCP	66	45018 → 443 [ACK] Seq=3278 Ack=2193 Win=42752 Len=0 TSval=1371037870 TSecr=259056442
111	45.389307548	200.174.103.36	192.168.1.16	TCP	1494	[TCP Out-Of-Order] 443 → 45018 [ACK] Seq=3278 Ack=3278 Win=65536 Len=1428 TSval=259056442 TSecr=1371637847
112	45.389315650	192.168.1.16	200.174.103.36	TCP	66	45018 → 443 [ACK] Seq=3278 Ack=24630 Win=40192 Len=0 TSval=1371037870 TSecr=259056442
63	45.083036183	192.168.1.16	200.174.103.36	TLSv1.2	1823	Client Hello (SSLwww.ietf.org)
66	45.084108100	200.174.103.36	192.168.1.16	TLSv1.2	1494	[TCP Previous segment not captured] , Ignored Unknown Record
68	45.084109980	200.174.103.36	192.168.1.16	TLSv1.2	2170	Ignored Unknown Record
72	45.086361431	192.168.1.16	200.174.103.36	TLSv1.2	216	Client Key Exchange, Change Cipher Spec, Encrypted Handshake Message
73	45.025979226	200.174.103.36	192.168.1.16	TLSv1.2	141	Change Cipher Spec, Encrypted Handshake Message
74	45.026290654	192.168.1.16	200.174.103.36	TLSv1.2	791	Application Data
77	45.048203064	200.174.103.36	192.168.1.16	TLSv1.2	1295	Application Data
106	45.366312766	192.168.1.16	200.174.103.36	TLSv1.2	711	Application Data
108	45.393232538	192.168.1.16	200.174.103.36	TCP	66	[TCP Keep-Alive] 45018 → 443 [ACK] Seq=3277 Ack=24630 Win=40192 Len=0 TSval=1371038074 TSecr=259056442
109	45.413156078	200.174.103.36	192.168.1.16	TCP	66	[TCP Keep-Alive ACK] 443 → 45018 [ACK] Seq=24630 Ack=3278 Win=65536 Len=0 TSval=259060944 TSecr=1371637870
257	123.752369651	192.168.1.16	200.174.103.36	TCP	66	45018 → 443 [FIN, ACK] Seq=3278 Ack=24630 Win=40192 Len=0 TSval=1371716233 TSecr=259060944
263	123.75237321	200.174.103.36	192.168.1.16	TCP	66	443 → 45018 [FIN, ACK] Seq=24630 Ack=3278 Win=65536 Len=0 TSval=2590664208 TSecr=1371716233

Figura 35 - Tráfego da rede



Protocol	Length	Info
TCP	78	[TCP Dup ACK 62#1] 45618 → 443 [ACK] Seq=1758 Ack=1 Win=64256 Len=0 TSval=1371637484 TSecr=2
TCP	66	45618 → 443 [ACK] Seq=1758 Ack=2857 Win=61440 Len=0 TSval=1371637484 TSecr=259656403
TCP	66	45618 → 443 [ACK] Seq=1758 Ack=4961 Win=59392 Len=0 TSval=1371637484 TSecr=259656403
TCP	8634	443 → 45618 [ACK] Seq=5036 Ack=2633 Win=64768 Len=8568 TSval=259656408 TSecr=1371637507 [TCP
TCP	66	45618 → 443 [ACK] Seq=2633 Ack=13604 Win=50944 Len=0 TSval=1371637528 TSecr=259656408
TCP	66	45618 → 443 [ACK] Seq=2633 Ack=14833 Win=49792 Len=0 TSval=1371637570 TSecr=259656408
TCP	7206	443 → 45618 [ACK] Seq=14833 Ack=3278 Win=65536 Len=7140 TSval=259656442 TSecr=1371637847 [TC
TCP	1295	[TCP Previous segment not captured] 443 → 45618 [PSH, ACK] Seq=23401 Ack=3278 Win=65536 Len=
TCP	66	45618 → 443 [ACK] Seq=3278 Ack=21973 Win=42752 Len=0 TSval=1371637870 TSecr=259656442
TCP	78	[TCP Dup ACK 109#1] 45618 → 443 [ACK] Seq=3278 Ack=21973 Win=42752 Len=0 TSval=1371637870 TS
TCP	1494	[TCP Out-Of-Order] 443 → 45618 [ACK] Seq=21973 Ack=3278 Win=65536 Len=1428 TSval=259656442 T
TCP	66	45618 → 443 [ACK] Seq=3278 Ack=24630 Win=40192 Len=0 TSval=1371637870 TSecr=259656442
TLSv1.2	1823	Client Hello (SNI=www.icec.edu.br)
TLSv1.2	1494	[TCP Previous segment not captured] , Ignored Unknown Record
TLSv1.2	2170	Ignored Unknown Record
TLSv1.2	216	Client Key Exchange, Change Cipher Spec, Encrypted Handshake Message
TLSv1.2	141	Change Cipher Spec, Encrypted Handshake Message
TLSv1.2	791	Application Data
TLSv1.2	1295	Application Data
TLSv1.2	711	Application Data
TCP	66	[TCP Keep-Alive] 45618 → 443 [ACK] Seq=3277 Ack=24630 Win=40192 Len=0 TSval=1371682874 TSecr=
TCP	66	[TCP Keep-Alive ACK] 443 → 45618 [ACK] Seq=24630 Ack=3278 Win=65536 Len=0 TSval=259660944 TS
TCP	66	45618 → 443 [FIN, ACK] Seq=3278 Ack=24630 Win=40192 Len=0 TSval=1371716233 TSecr=259660944
TCP	66	443 → 45618 [FIN, ACK] Seq=24630 Ack=3279 Win=65536 Len=0 TSval=259664280 TSecr=1371716233

Figura 36 - Pacote do segmento TCP

## Parte 3 - Análise do Protocolo TCP:

Primeiro foi realizado a liberação do Cache do DNS:

```
C:\Windows\System32>ipconfig /flushdns

Configuração de IP do Windows

Liberação do Cache do DNS Resolver bem-sucedida.

C:\Windows\System32>
```

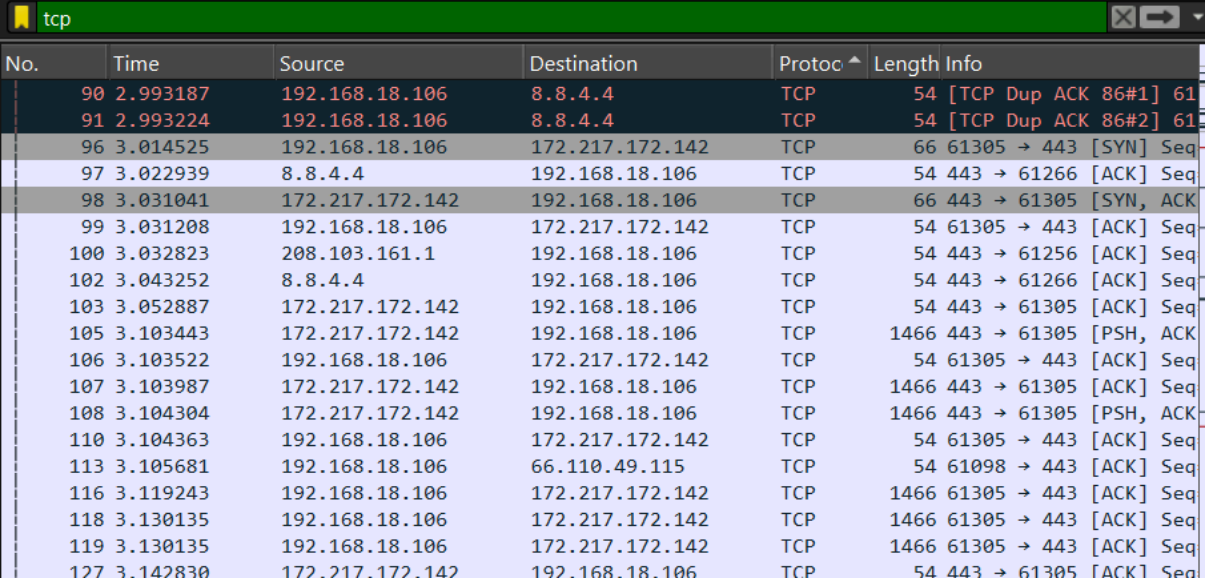
Figura 37 - Liberação do cache DNS

### Objetivo

Observar a interação entre o cliente e o servidor através do protocolo TCP, analisando o comportamento da conexão TCP em detalhes.

## Procedimentos Realizados

Para esta parte, utilizamos a captura já realizada anteriormente do site <http://www.icec.edu.br>. O protocolo TCP é responsável por estabelecer uma conexão confiável entre o cliente e o servidor, garantindo que os dados sejam entregues de forma ordenada e sem erros.



No.	Time	Source	Destination	Protoc	Length	Info
90	2.993187	192.168.18.106	8.8.4.4	TCP	54	[TCP Dup ACK 86#1] 61
91	2.993224	192.168.18.106	8.8.4.4	TCP	54	[TCP Dup ACK 86#2] 61
96	3.014525	192.168.18.106	172.217.172.142	TCP	66	61305 → 443 [SYN] Seq
97	3.022939	8.8.4.4	192.168.18.106	TCP	54	443 → 61266 [ACK] Seq
98	3.031041	172.217.172.142	192.168.18.106	TCP	66	443 → 61305 [SYN, ACK
99	3.031208	192.168.18.106	172.217.172.142	TCP	54	61305 → 443 [ACK] Seq
100	3.032823	208.103.161.1	192.168.18.106	TCP	54	443 → 61256 [ACK] Seq
102	3.043252	8.8.4.4	192.168.18.106	TCP	54	443 → 61266 [ACK] Seq
103	3.052887	172.217.172.142	192.168.18.106	TCP	54	443 → 61305 [ACK] Seq
105	3.103443	172.217.172.142	192.168.18.106	TCP	1466	443 → 61305 [PSH, ACK
106	3.103522	192.168.18.106	172.217.172.142	TCP	54	61305 → 443 [ACK] Seq
107	3.103987	172.217.172.142	192.168.18.106	TCP	1466	443 → 61305 [ACK] Seq
108	3.104304	172.217.172.142	192.168.18.106	TCP	1466	443 → 61305 [PSH, ACK
110	3.104363	192.168.18.106	172.217.172.142	TCP	54	61305 → 443 [ACK] Seq
113	3.105681	192.168.18.106	66.110.49.115	TCP	54	61098 → 443 [ACK] Seq
116	3.119243	192.168.18.106	172.217.172.142	TCP	1466	61305 → 443 [ACK] Seq
118	3.130135	192.168.18.106	172.217.172.142	TCP	1466	61305 → 443 [ACK] Seq
119	3.130135	192.168.18.106	172.217.172.142	TCP	1466	61305 → 443 [ACK] Seq
127	3.142830	172.217.172.142	192.168.18.106	TCP	54	443 → 61305 [ACK] Seq

Figura 38: Tela do Wireshark com filtro TCP aplicado, mostrando a lista de pacotes filtrados

## Three-Way Handshake (SYN, SYN-ACK, ACK)

Foi encontrado o Three-Way Handshake completo do protocolo TCP na troca de pacotes entre o cliente com IP 52.104.123.39 e o servidor 192.168.18.106, utilizando a porta de origem 61304 no cliente e a porta 443 no servidor. O processo se inicia com o envio de um pacote SYN pelo cliente, seguido pela resposta SYN, ACK do servidor, e finaliza com o envio de um pacote ACK pelo cliente, completando assim o estabelecimento da conexão TCP.

192.168.18.106	52.104.123.39	TCP	66	61304 → 443 [SYN]	Seq=0 Win=65535 Len=0 MSS=1460
192.168.18.106	200.196.224.15	TCP	54	61263 → 443 [ACK]	Seq=774 Ack=374 Win=254 Len=0
172.172.255.218	192.168.18.106	TCP	54	443 → 50612 [ACK]	Seq=474 Ack=79 Win=6950 Len=0
52.104.123.39	192.168.18.106	TCP	66	443 → 61304 [SYN, ACK]	Seq=0 Ack=1 Win=65535 Len=0
192.168.18.106	52.104.123.39	TCP	54	61304 → 443 [ACK]	Seq=1 Ack=1 Win=65280 Len=0

Figura 39: Momento onde há o estabelecimento da conexão TCP via Three-Way Handshake

## Respostas às Questões:

**3.1) Qual é o número da porta TCP usada pelo seu computador cliente (source) para requisitar o arquivo html para o site de seu grupo? E qual o número de porta TCP que o servidor está usando para receber e enviar essas respostas?**

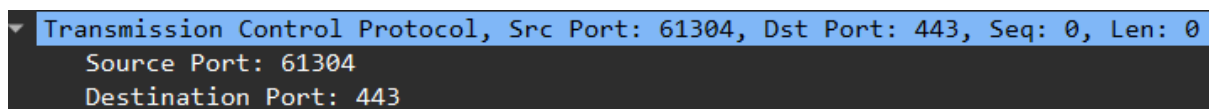
Foi selecionado o seguinte pacote TCP:



192.168.18.106 52.104.123.39 TCP 66 61304 → 443 [SYN] Seq=0 Win=65535 Len=0 MSS

Figura 40: pacote TCP selecionado

Portas source e destination:



Transmission Control Protocol, Src Port: 61304, Dst Port: 443, Seq: 0, Len: 0  
Source Port: 61304  
Destination Port: 443

Figura 41: Detalhe de um pacote TCP mostrando as portas source e destination

## Resposta encontrada:

- Porta do computador cliente (source): **61304**
- Porta do servidor (destination): **443**

**Explicação:** O servidor web está utilizando a porta 443, que é a porta padrão para **HTTPS** (HTTP Seguro com criptografia TLS/SSL), não HTTP comum. Isso indica que a comunicação com o site [www.icec.edu.br](http://www.icec.edu.br) está sendo feita de forma criptografada. O cliente utiliza uma porta aleatória alta (61304) que é atribuída automaticamente pelo sistema operacional para estabelecer a conexão.

**3.2) Mostre e comente o pacote que contém o segmento TCP SYN que caracteriza o início de uma conexão TCP entre o computador cliente e o site visitado.**

Pacote que contém o segmento TCP SYN que caracteriza o início de uma conexão TCP:



192.168.18.106 52.104.123.39 TCP 66 61304 → 443 [SYN] Seq=0 Win=65535 Len=0 MSS

Figura 42: estabelecimento inicial de conexão - three way handshake

O primeiro passo do processo conhecido como Three-Way Handshake é justamente o envio de um segmento TCP com a flag SYN ativada, que representa a solicitação de estabelecimento de conexão. Esse pacote é enviado pelo cliente e não contém ainda dados da aplicação, mas serve como um pedido de abertura de canal de comunicação.

Na imagem abaixo, vemos claramente o pacote SYN com a flag 0x002 marcada, o que indica que o campo SYN está ativado. Além disso, o próprio Wireshark destaca que se trata de um "Connection establish request (SYN)" para a porta do servidor 443. Isso confirma que

o pacote capturado é o responsável por iniciar a conexão TCP com o site acessado, caracterizando a primeira etapa do Three-Way Handshake.

```
Flags: 0x002 (SYN)
 000. .... = Reserved: Not set
...0 .... = Accurate ECN: Not set
.... 0... = Congestion Window Reduced: Not set
.... .0.. = ECN-Echo: Not set
.... ..0. = Urgent: Not set
.... ...0 = Acknowledgment: Not set
.... .... 0... = Push: Not set
.... .... .0.. = Reset: Not set
.... .... ..1. = Syn: Set
  [Expert Info (Chat/Sequence): Connection establish request (SYN): server port 443]
.... .... ...0 = Fin: Not set
[TCP Flags: .....S.]
Window: 65535
[Calculated window size: 65535]
Checksum: 0xc918 [unverified]
[Checksum Status: Unverified]
Urgent Pointer: 0
```

Figura 43: Pacote com a flag SYN ativada, correspondente ao início da conexão TCP

### 3.3) Mostre e comente o campo do segmento TCP que contém o tamanho da janela utilizada pelo TCP para o controle de fluxo. Qual é o tamanho da janela em bytes?

Após identificar o pacote SYN responsável por iniciar a conexão TCP, é possível analisar as informações adicionais contidas nesse mesmo segmento. Um dos campos importantes é o "Window Size", que representa o tamanho da janela TCP, utilizada no controle de fluxo. Esse valor indica quantos bytes o remetente está disposto a receber antes de precisar enviar um novo reconhecimento (ACK).

```
Window: 65535
[Calculated window size: 65535]
```

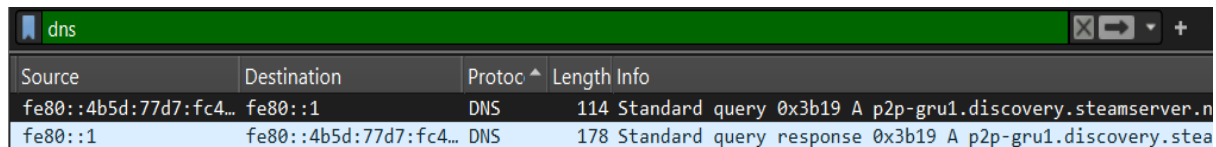
Figura 44: Tamanho da janela TCP em bytes, especificado no segmento SYN

No caso capturado, o campo mostra que o tamanho da janela é de 65535 bytes, o que significa que o computador que enviou esse pacote está apto a receber até 65535 bytes de dados antes de solicitar uma nova confirmação. Esse valor é relativamente alto, o que sugere uma configuração para comunicação eficiente, com maior volume de dados sendo trocado sem interrupções frequentes para confirmações.

## Parte 4 - Análise do Protocolo DNS

### Objetivo

Observar a consulta e resposta DNS para a resolução de nomes de domínio, entendendo como o sistema converte nomes legíveis (como [www.icec.edu.br](http://www.icec.edu.br)) em endereços IP.



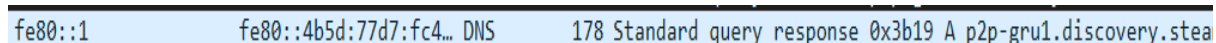
Source	Destination	Protoc	Length	Info
fe80::4b5d:77d7:fc43:ad99	fe80::1	DNS	114	Standard query 0x3b19 A p2p-gru1.discovery.steamserver.n
fe80::1	fe80::4b5d:77d7:fc43:ad99	DNS	178	Standard query response 0x3b19 A p2p-gru1.discovery.stea

Figura 45: Wireshark com filtro DNS aplicado, mostrando pacotes de consulta e resposta DNS

### Respostas às Questões:

#### 4.1) Qual o endereço IP do servidor DNS que resolveu o nome do host?

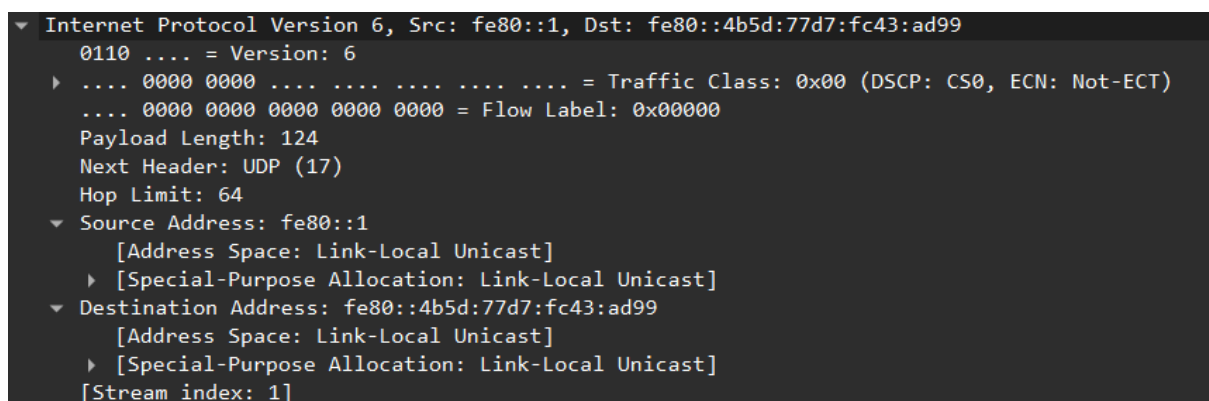
Esses pacotes geralmente aparecem com a descrição “Standard query response”. Ao selecionar um desses pacotes e expandir a seção “Internet Protocol”, é possível visualizar o endereço IP de origem da resposta, que corresponde ao servidor DNS que efetivamente resolveu o nome do host.



fe80::1	fe80::4b5d:77d7:fc43:ad99	DNS	178	Standard query response 0x3b19 A p2p-gru1.discovery.stea
---------	---------------------------	-----	-----	--

Figura 46: pacote com a descrição “Standard query response” analisado, em que foi possível, através dele, identificar o IP do servidor DNS que resolveu o nome do host.

Na imagem abaixo, observamos que o campo Source (Origem) mostra o endereço **fe80::1**, indicando que esse foi o IP do servidor DNS responsável por fornecer a resposta à consulta realizada. Já o campo Destination mostra o endereço do nosso computador, o cliente que fez a solicitação.



Internet Protocol Version 6, Src: fe80::1, Dst: fe80::4b5d:77d7:fc43:ad99				
0110 .... = Version: 6				
.... 0000 0000 .... = Traffic Class: 0x00 (DSCP: CS0, ECN: Not-ECT)				
.... 0000 0000 0000 0000 = Flow Label: 0x000000				
Payload Length: 124				
Next Header: UDP (17)				
Hop Limit: 64				
Source Address: fe80::1				
[Address Space: Link-Local Unicast]				
[Special-Purpose Allocation: Link-Local Unicast]				
Destination Address: fe80::4b5d:77d7:fc43:ad99				
[Address Space: Link-Local Unicast]				
[Special-Purpose Allocation: Link-Local Unicast]				
[Stream index: 1]				

Figura 47: IP de origem do pacote de resposta DNS (**fe80::1**), correspondente ao servidor DNS que resolveu o nome do host.

É importante destacar uma diferença sutil, mas essencial: se estivéssemos analisando um pacote de requisição DNS, o endereço IP do servidor DNS apareceria no campo Destination, já que estaríamos enviando a pergunta para ele. No entanto, como a pergunta da atividade pede qual servidor resolveu o nome do host, o correto é olhar um pacote de resposta, onde o servidor DNS é quem está enviando os dados de volta e, portanto, aparece no campo Source. Essa distinção garante que estamos realmente identificando o servidor que respondeu, e não apenas aquele que foi consultado.

#### 4.2) Qual a porta destino utilizada para a consulta DNS?

```
fe80::4b5d:77d7:fc4... fe80::1      DNS      114 Standard query 0x3b19 A p2p-gru1.discovery.steamserver.n
```

Figura 48: Standard query

```
▼ User Datagram Protocol, Src Port: 52096, Dst Port: 53
  Source Port: 52096
  Destination Port: 53
  Length: 60
  Checksum: 0xe57d [unverified]
  [Checksum Status: Unverified]
  [Stream index: 2]
  [Stream Packet Number: 1]
  ▶ [Timestamps]
  UDP payload (52 bytes)
```

Figura 49: Detalhe do pacote DNS mostrando a porta de destino UDP

Porta destino DNS: **53**

#### 4.3) Qual o protocolo de transporte utilizado na consulta DNS?

Na janela de detalhes de qualquer pacote DNS foi verificado se aparecia "User Datagram Protocol (UDP)" ou "Transmission Control Protocol (TCP)", a partir dessa verificação concluímos que o protocolo de transporte utilizado na consulta DNS foi o **UDP**.

```
▼ User Datagram Protocol, Src Port: 52096, Dst Port: 53
```

Figura 50: Estrutura do pacote DNS mostrando o protocolo de transporte utilizado

```
▼ User Datagram Protocol, Src Port: 52096, Dst Port: 53
  Source Port: 52096
  Destination Port: 53
  Length: 60
  Checksum: 0xe57d [unverified]
  [Checksum Status: Unverified]
  [Stream index: 2]
  [Stream Packet Number: 1]
  ▶ [Timestamps]
  UDP payload (52 bytes)
```

Figura 51: Estrutura do pacote DNS mostrando o protocolo de transporte utilizado

Protocolo de transporte: **UDP**

Explicação: O DNS tradicionalmente utiliza UDP na porta 53 para consultas simples, pois é mais rápido e eficiente. TCP é usado apenas em casos especiais, como transferências de zona ou quando a resposta é muito grande.

## Parte 5 - Análise do Protocolo Ethernet

Objetivo, nessa parte do trabalho analisaremos os endereços mac envolvidos nas transações de pacotes.

**Endereços MAC meu e Site [www.florestal.mg.gov.br](http://www.florestal.mg.gov.br):**

Primeiramente, pegamos o endereço IP do site do gov florestal. Acessamos o site, e no wireshark filtramos pelo IP.

```
gabriel@Decepticon:~$ ping www.florestal.mg.gov.br
PING www.florestal.mg.gov.br (172.67.198.105) 56(84) bytes of data.
64 bytes from 172.67.198.105: icmp_seq=1 ttl=57 time=16.9 ms
64 bytes from 172.67.198.105: icmp_seq=2 ttl=57 time=23.5 ms
64 bytes from 172.67.198.105: icmp_seq=3 ttl=57 time=23.6 ms
64 bytes from 172.67.198.105: icmp_seq=4 ttl=57 time=23.6 ms
64 bytes from 172.67.198.105: icmp_seq=5 ttl=57 time=23.8 ms
^C
```

Figura 52 - Ping de IP para o site de Florestal



ip.addr == 172.67.198.105						
No.	Time	Source	Destination	Protocol	Length	Info
28526	1961.465924950	192.168.1.16	172.67.198.105	QUIC	1292	Protected Payload
28527	1961.465942131	192.168.1.16	172.67.198.105	QUIC	1226	Protected Payload
28531	1961.514222865	172.67.198.105	192.168.1.16	QUIC	66	Protected Payload
28533	1961.514223843	172.67.198.105	192.168.1.16	QUIC	66	Protected Payload
28534	1961.514224751	172.67.198.105	192.168.1.16	QUIC	66	Protected Payload
28536	1961.514225729	172.67.198.105	192.168.1.16	QUIC	66	Protected Payload
28538	1961.514227126	172.67.198.105	192.168.1.16	QUIC	66	Protected Payload
28539	1961.514228103	172.67.198.105	192.168.1.16	QUIC	66	Protected Payload
28540	1961.514229011	172.67.198.105	192.168.1.16	QUIC	1130	Protected Payload
28542	1961.519725468	192.168.1.16	172.67.198.105	QUIC	87	Protected Payload
28544	1961.555839882	172.67.198.105	192.168.1.16	QUIC	64	Protected Payload
28545	1961.560180811	192.168.1.16	172.67.198.105	QUIC	87	Protected Payload
19544	1954.939973396	192.168.1.16	172.67.198.105	TCP	74	46492 → 443 [SYN]
19545	1954.961539614	172.67.198.105	192.168.1.16	TCP	74	443 → 46492 [SYN]
19547	1954.961576281	192.168.1.16	172.67.198.105	TCP	66	46492 → 443 [ACK]
19549	1954.979634221	172.67.198.105	192.168.1.16	TCP	66	443 → 46492 [ACK]
19551	1954.984297398	192.168.1.16	172.67.198.105	TCP	66	46492 → 443 [ACK]
19555	1955.005434720	172.67.198.105	192.168.1.16	TCP	66	443 → 46492 [ACK]
19558	1955.023310373	172.67.198.105	192.168.1.16	TCP	66	443 → 46492 [ACK]
19567	1955.027174843	192.168.1.16	172.67.198.105	TCP	66	46492 → 443 [ACK]
19585	1955.058340357	192.168.1.16	172.67.198.105	TCP	66	46492 → 443 [ACK]
19548	1954.961889660	192.168.1.16	172.67.198.105	TLSv1.3	1863	Client Hello (S)
19550	1954.984280776	172.67.198.105	192.168.1.16	TLSv1.3	3537	Server Hello, C
19552	1954.988123805	192.168.1.16	172.67.198.105	TLSv1.3	130	Change Cipher S
19553	1954.988351558	192.168.1.16	172.67.198.105	TLSv1.3	158	Application Dat
Frame 19549: 66 bytes on wire (528 bits), 66 bytes captured (528 bits) on interface wlp2s0, id 0						
Ethernet II, Src: TendaTechnol_33:51:90 (c8:3a:35:33:51:90), Dst: Intel_f5:05:cd (48:51:c5:f5:05:cd)						
Destination: Intel_f5:05:cd (48:51:c5:f5:05:cd)						
Source: TendaTechnol_33:51:90 (c8:3a:35:33:51:90)						
Type: IPv4 (0x0800)						
[Stream index: 0]						
Internet Protocol Version 4, Src: 172.67.198.105, Dst: 192.168.1.16						
Transmission Control Protocol, Src Port: 443, Dst Port: 46492, Seq: 1, Ack: 1798, Len: 0						

Figura 53 - Tráfego de rede pelo site de Florestal

- Endereço MAC do meu computador: 48:51:c5:f5:05:cd
- Endereço MAC do computador de destino: c8:3a:35:33:51:90



# Conclusão

Através da realização desta experiência prática simulada no Wireshark, foi possível obter uma compreensão mais aprofundada do funcionamento do tráfego de rede em um dispositivo e dos principais protocolos envolvidos, como HTTP, DNS, TCP e UDP. O software demonstrou ser eficaz e completo, permitindo a visualização em tempo real da comunicação entre dois IPs distintos — de origem e destino — incluindo a identificação do servidor.

Com isso, foi possível:

- Identificar e analisar os pacotes capturados durante a navegação;
- Observar a estrutura e os campos relevantes das requisições e respostas HTTP;
- Compreender o processo de estabelecimento e encerramento de conexões TCP por meio dos sinais de controle;;
- Visualizar os endereços MAC e IP envolvidos nas comunicações.