

Lucas Bezerra Storino

HTTP

Respostas HTTP referentes ao site

<http://gaia.cs.umass.edu/wireshark-labs/HTTP-wireshark-file2.html>

```
▼ Hypertext Transfer Protocol
  > HTTP/1.1 200 OK\r\n
    Date: Fri, 27 Aug 2021 19:51:22 GMT\r\n
    Server: Apache/2.4.6 (CentOS) OpenSSL/1.0.2k-fips PHP/7.4.22 mod_perl/2.0.11 Perl/v5.16.3\r\n
    Last-Modified: Fri, 27 Aug 2021 05:59:01 GMT\r\n
    ETag: "173-5ca842e1a8b56"\r\n
    Accept-Ranges: bytes\r\n
  > Content-Length: 371\r\n
    Keep-Alive: timeout=5, max=100\r\n
    Connection: Keep-Alive\r\n
    Content-Type: text/html; charset=UTF-8\r\n
    \r\n
    [HTTP response 1/1]
    [Time since request: 0.197750000 seconds]
    [Request in frame: 1320]
    [Request URI: http://gaia.cs.umass.edu/wireshark-labs/HTTP-wireshark-file2.html]
    File Data: 371 bytes
  ▼ Line-based text data: text/html (10 lines)
    \n
    <html>\n
    \n
    Congratulations again! Now you've downloaded the file lab2-2.html. <br>\n
    This file's last modification date will not change. <p>\n
    Thus if you download this multiple times on your browser, a complete copy <br>\n
    will only be sent once by the server due to the inclusion of the IN-MODIFIED-SINCE<br>\n
    field in your browser's HTTP GET request to the server.\n
    \n
    </html>\n
```

Na primeira resposta, conteúdo é transmitido explicitamente e mostrado em line-based text data: text/html

```
▼ Hypertext Transfer Protocol
  > HTTP/1.1 304 Not Modified\r\n
    Date: Fri, 27 Aug 2021 19:51:29 GMT\r\n
    Server: Apache/2.4.6 (CentOS) OpenSSL/1.0.2k-fips PHP/7.4.22 mod_perl/2.0.11 Perl/v5.16.3\r\n
    Connection: Keep-Alive\r\n
    Keep-Alive: timeout=5, max=100\r\n
    ETag: "173-5ca842e1a8b56"\r\n
    \r\n
    [HTTP response 1/1]
    [Time since request: 0.201907000 seconds]
    [Request in frame: 3355]
    [Request URI: http://gaia.cs.umass.edu/wireshark-labs/HTTP-wireshark-file2.html]
```

Na segunda resposta, o arquivo não foi modificado (304 Not Modified), logo a resposta HTTP não retornou o conteúdo, graças ao cache.

Respostas HTTP referentes ao site

http://gaia.cs.umass.edu/wireshark-labs/protected_pages/HTTP-wireshark-file5.html

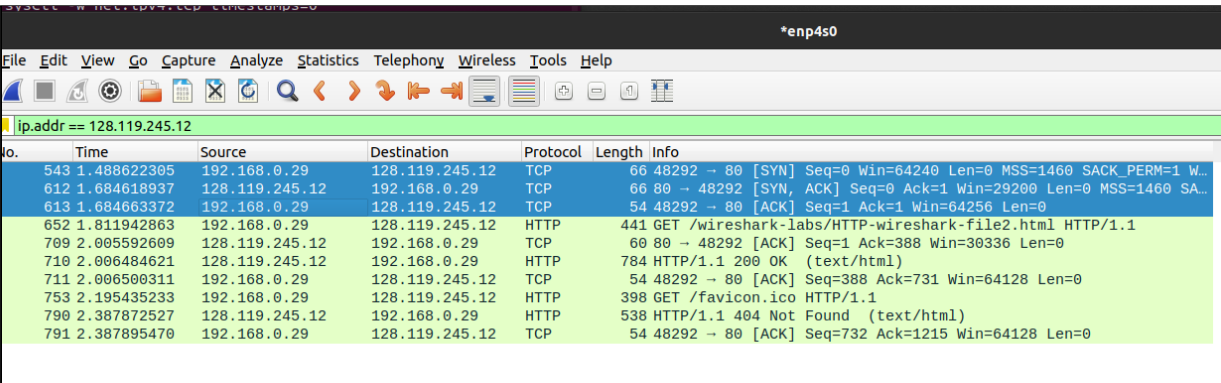
Primeiramente, recebemos uma mensagem 401 (Unauthorized). Depois de passar os dados pedidos, a mensagem de status é 200 OK (ou 304 Not Modified se não der flush na cache) e aparece uma nova linha *Authorization: Basic* contendo as credenciais de usuário.

```
▼ Hypertext Transfer Protocol
  > GET /wireshark-labs/protected_pages/HTTP-wireshark-file5.html HTTP/1.1\r\n
    Host: gaia.cs.umass.edu\r\n
    User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:91.0) Gecko/20100101 Firefox/91.0\r\n
    Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8\r\n
    Accept-Language: pt-BR,pt;q=0.8,en-US;q=0.5,en;q=0.3\r\n
    Accept-Encoding: gzip, deflate\r\n
  ▼ Authorization: Basic d2lyZXNoYXJrLXN0dWRlbnRzOm5ldHdvcmcs=\r\n
    Credentials: wireshark-students:network
    Connection: keep-alive\r\n
    Upgrade-Insecure-Requests: 1\r\n
    \r\n
    [Full request URI: http://gaia.cs.umass.edu/wireshark-labs/protected_pages/HTTP-wireshark-file5.html]
    [HTTP request 1/2]
    [Response in frame: 144]
```

TCP

Percebemos no print a seguir o então Three-way-handshake com [SYN], [SYN, ACK] e finalmente [SYN] ao acessar o site

<http://gaia.cs.umass.edu/wireshark-labs/HTTP-wireshark-file2.html>



The image shows a Wireshark packet capture window titled '*enp4s0'. The packet list shows several packets. The first three packets (543, 612, 613) represent the TCP three-way handshake. Packet 543 is a SYN from 192.168.0.29 to 128.119.245.12. Packet 612 is a SYN, ACK from 128.119.245.12 to 192.168.0.29. Packet 613 is an ACK from 192.168.0.29 to 128.119.245.12. Subsequent packets (652, 709, 710, 711, 753, 790, 791) are HTTP requests and responses.

No.	Time	Source	Destination	Protocol	Length	Info
543	1.488622305	192.168.0.29	128.119.245.12	TCP	66	48292 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 W...
612	1.684618937	128.119.245.12	192.168.0.29	TCP	66	80 → 48292 [SYN, ACK] Seq=0 Ack=1 Win=29200 Len=0 MSS=1460 SA...
613	1.684663372	192.168.0.29	128.119.245.12	TCP	54	48292 → 80 [ACK] Seq=1 Ack=1 Win=64256 Len=0
652	1.811942863	192.168.0.29	128.119.245.12	HTTP	441	GET /wireshark-labs/HTTP-wireshark-file2.html HTTP/1.1
709	2.005592609	128.119.245.12	192.168.0.29	TCP	60	80 → 48292 [ACK] Seq=1 Ack=388 Win=30336 Len=0
710	2.006484621	128.119.245.12	192.168.0.29	HTTP	784	HTTP/1.1 200 OK (text/html)
711	2.006500311	192.168.0.29	128.119.245.12	TCP	54	48292 → 80 [ACK] Seq=388 Ack=731 Win=64128 Len=0
753	2.195435233	192.168.0.29	128.119.245.12	HTTP	398	GET /favicon.ico HTTP/1.1
790	2.387872527	128.119.245.12	192.168.0.29	HTTP	538	HTTP/1.1 404 Not Found (text/html)
791	2.387895470	192.168.0.29	128.119.245.12	TCP	54	48292 → 80 [ACK] Seq=732 Ack=1215 Win=64128 Len=0

DNS

Rodando o comando `nslookup -type=A` para o endereço www.mit.edu, percebemos no package manager que a porta destino da mensagem DNS de requerimento é a mesma porta fonte da mensagem de resposta, que é a 53.

4	1.081704	2804:14d:5c31:560c:99ab:f82d:97ff:d0	2804:14d:5c31:560c:99ab:f82d:97ff:d0
5	1.120776	2804:14d:1:0:181:213:132:4	2804:14d:1:0:181:213:132:4
6	1.121584	2804:14d:5c31:560c:99ab:f82d:97ff:d0	2804:14d:5c31:560c:99ab:f82d:97ff:d0
7	1.132959	2804:14d:1:0:181:213:132:4	2804:14d:1:0:181:213:132:4

Hop Limit: 64	Hop Limit: 54
Source Address: 2804:14d:5c31:560c:99ab:f82d:97ff:d0	Source Address: 2804:14d:1:0:181:213:132:4
Destination Address: 2804:14d:1:0:181:213:132:4	Destination Address: 2804:14d:1:0:181:213:132:4
▼ User Datagram Protocol, Src Port: 56592	▼ User Datagram Protocol, Src Port: 53
Source Port: 56592	Source Port: 53
Destination Port: 53	Destination Port: 56592
Length: 98	Length: 158

O IP destino é o mesmo IP do DNS (2804:14d:1:0:181:213:132:4)

4	1.081704	2804:14d:5c31:560c:99ab:f82d:97ff:d0	2804:14d:5c31:560c:99ab:f82d:97ff:d0
5	1.120776	2804:14d:1:0:181:213:132:4	2804:14d:1:0:181:213:132:4
6	1.121584	2804:14d:5c31:560c:99ab:f82d:97ff:d0	2804:14d:5c31:560c:99ab:f82d:97ff:d0
7	1.132959	2804:14d:1:0:181:213:132:4	2804:14d:1:0:181:213:132:4

0110 = Version: 6
> 0000 0000 = Traffic Class
.... 0111 1011 1111 0010 1111 = Flow Label
Payload Length: 98
Next Header: UDP (17)
Hop Limit: 64
Source Address: 2804:14d:5c31:560c:99ab:f82d:97ff:d0
Destination Address: 2804:14d:1:0:181:213:132:4

```
Servidores DNS. . . . . : 2804:14d:1:0:181:213:132:4
                        2804:14d:1:0:181:213:132:5
                        181.213.132.4
                        181.213.132.5
```

Além disso, temos 3 Respostas vindas da mensagem de resposta do DNS. Cada uma com as seguintes informações: nome do host, tipo do endereço, classe, tempo, tamanho do dado e o endereço IP.

```

Answers
  www.mit.edu: type CNAME, class IN, cname www.mit.edu.edgekey.net
    Name: www.mit.edu
    Type: CNAME (Canonical NAME for an alias) (5)
    Class: IN (0x0001)
    Time to live: 1800 (30 minutes)
    Data length: 25
    CNAME: www.mit.edu.edgekey.net
  www.mit.edu.edgekey.net: type CNAME, class IN, cname e9566.dscb.akamaiedge.net
    Name: www.mit.edu.edgekey.net
    Type: CNAME (Canonical NAME for an alias) (5)
    Class: IN (0x0001)
    Time to live: 60 (1 minute)
    Data length: 24
    CNAME: e9566.dscb.akamaiedge.net
  e9566.dscb.akamaiedge.net: type A, class IN, addr 104.109.38.5
    Name: e9566.dscb.akamaiedge.net
    Type: A (Host Address) (1)
    Class: IN (0x0001)
    Time to live: 20 (20 seconds)
    Data length: 4
    Address: 104.109.38.5

```

Agora com o `-type=NS`:

Percebemos que novamente o IP destino é o mesmo IP do DNS
(2804:14d:1:0:181:213:132:4)

56	0.595320	2804:14d:5c31:560c:99ab:f82d:97ff:d068	2804:14d:...	DNS	152	Standard query 0x0001 PTR 4.0.0.0
96	0.935095	2804:14d:1:0:181:213:132:4	2804:14d:...	DNS	212	Standard query response 0x0001 No
97	0.936059	2804:14d:5c31:560c:99ab:f82d:97ff:d068	2804:14d:...	DNS	87	Standard query 0x0002 NS mit.edu
99	0.951486	2804:14d:1:0:181:213:132:4	2804:14d:...	DNS	422	Standard query response 0x0002 NS
127	1.182314	2804:14d:5c31:560c:99ab:f82d:97ff:d068	2800:3f0:...	TLSv1.2	113	Application Data
128	1.191731	2800:3f0:4004:803::2001	2804:14d:...	TLSv1.2	113	Application Data


```

> Frame 97: 87 bytes on wire (696 bits), 87 bytes captured (696 bits) on interface \Device\NPF_{AA624684-28A6-4C1F-A768}
> Ethernet II, Src: BiostarM_14:4c:ae (f4:b5:20:14:4c:ae), Dst: Kaonmedi_7b:ca:cd (98:77:e7:7b:ca:cd)
> Internet Protocol Version 6, Src: 2804:14d:5c31:560c:99ab:f82d:97ff:d068, Dst: 2804:14d:1:0:181:213:132:4
  0110 .... = Version: 6
  > .... 0000 0000 .... = Traffic Class: 0x00 (DSCP: CS0, ECN: Not-ECT)
  .... 1001 0110 0011 1000 0001 = Flow Label: 0x96381
  Payload Length: 33
  Next Header: UDP (17)
  Hop Limit: 64
  Source Address: 2804:14d:5c31:560c:99ab:f82d:97ff:d068
  Destination Address: 2804:14d:1:0:181:213:132:4
> User Datagram Protocol, Src Port: 54757, Dst Port: 53

```



```

Servidores DNS. . . . . : 2804:14d:1:0:181:213:132:4
                        2804:14d:1:0:181:213:132:5
                        181.213.132.4
                        181.213.132.5

```

Temos que o tipo de mensagem de query do DNS é agora do tipo “NS”, sem respostas. E a mensagem de resposta apresenta os seguintes nameservers do MIT:

```
▼ Answers
> mit.edu: type NS, class IN, ns asia1.akam.net
> mit.edu: type NS, class IN, ns ns1-173.akam.net
> mit.edu: type NS, class IN, ns ns1-37.akam.net
> mit.edu: type NS, class IN, ns eur5.akam.net
> mit.edu: type NS, class IN, ns use5.akam.net
> mit.edu: type NS, class IN, ns use2.akam.net
> mit.edu: type NS, class IN, ns asia2.akam.net
> mit.edu: type NS, class IN, ns usw2.akam.net
▼ Additional records
```

Suponha que você recebeu a faixa de endereços 146.164.70.0/23. A quantas máquinas você pode atribuir endereços IP públicos, com esta faixa? Quantas sub-redes IP você pode criar, no máximo? Neste caso, quantas máquinas no máximo poderiam receber endereços IP públicos? Justifique as respostas.

Podemos atribuir 510 IPs públicos com essa faixa. uma vez que temos $32 - 23 = 9$ bits disponíveis para variar a faixa, o que dá $2^9 = 512$ combinações, e desconsideramos o endereço de host e broadcast ($512 - 2 = 510$).

Diga quais são as camadas do modelo TCP/IP, uma descrição e um protocolo.

- **Aplicação:** contém todos os protocolos para um serviço específico de comunicação de dados em um nível de processo-a-processo. Um protocolo seria o HTTP.
- **Transporte:** controla a comunicação host-a-host. Um protocolo seria o TCP.
- **Rede:** responsável pelas conexões entre as redes locais, estabelecendo assim a interconexão. Um protocolo seria o IP.
- **Enlace:** método usado para passar quadros da camada de rede de um dispositivo para a camada de rede de outro. Esse processo pode ser controlado tanto em software (device driver) para a placa de rede quanto em firmware ou chipsets especializados. Um protocolo seria o Ethernet.

Para os endereços abaixo, diga a rede, host e broadcast:

- **IP:177.32.168.223 Máscara: 255.255.255.248**
- **IP:7.26.0.64 Máscara: /26**

177.32.168.217 - 177.32.168.222

host: 177.32.168.216

broadcast: 177.32.168.223

7.26.0.65 - 7.26.0.190

host: 7.26.0.64

broadcast: 7.26.0.191