



Relatório de Aula Prática – Redes de Computadores

Título: Análise do protocolo HTTP

Aluno: Vitor Mayorca Camargo

Data: 28/08/2024

1. INTRODUÇÃO

A internet surgiu na década de 1960 como ARPANET, inicialmente usada para comunicação militar e acadêmica nos EUA (LEINER, *et al.* 2009). Em 1989, a criação da World Wide Web democratizou o acesso e compartilhamento de informações por meio de navegadores web, permitindo que pessoas comuns usassem a internet de forma fácil e rápida (MONTEIRO, 2001).

As informações transmitidas na internet seguem protocolos rigorosos, como o TCP e o IP, que surgiram da necessidade de estabelecer regras e padrões para unificar as diferentes redes e máquinas que formam a internet (CORONA, 2004). O protocolo TCP (Transmission Control Protocol) foi projetado para fornecer uma comunicação confiável entre dois computadores, reduzindo ou aumentando a taxa de transmissão de dados de forma dinâmica, com o fim de evitar perda de dados. (CAMPISTA, *et al.* 2010).

Já o protocolo IP (Internet Protocol) define a base para a transação, tráfego e reconhecimento de dados em uma rede. Ele é responsável por definir o "Endereço IP", que é um número único dado a cada máquina ou "host" na rede. Esse endereço é composto por quatro números separados por pontos (de 0 a 255), e permite que uma máquina seja identificada e se comunique com outras na rede (CORONA, *et al.* 2004). Os autores também explicam que o protocolo TCP/IP divide os dados em pequenas porções para transferência na Internet. O protocolo IP organiza o envio e recebimento desses pacotes, enquanto o protocolo TCP se encarrega de dividir, ordenar e garantir que o fluxo de dados ocorra corretamente.

Duas versões do protocolo IP são usadas como padrão na internet. O IPv4 especifica as capacidades e protocolos básicos que toda máquina deve seguir, e usa um endereço de 32 bits (ALI, 2012). Já o IPv6, mais avançado, possui uma maior segurança e melhor comunicação entre as redes, fornecendo um endereço de 128 bits (CHANDRA, KATHING, KUMAR. 2013).

Segundo Kurose e Ross (2021), páginas web são compostas por objetos, que são armazenados em um servidor, podendo ser acessados por meio de uma URL. Os objetos de uma página nada mais são do que os diferentes elementos que as compõem, como arquivos HTML, imagens, scripts e vídeos. Em 1997, o World Wide Web consortium (W3C) publicou as especificações do protocolo HTTP, que padronizava a navegação em páginas web de hipertexto (TANENBAUM, 2003).

O HTTP é uma combinação dos protocolos FTP e SMTP. Segundo Tanenbaum (2003), o protocolo HTTP especifica as mensagens que os clientes podem enviar aos servidores, e que respostas eles receberão. Cada interação usuário-servidor consiste de uma solicitação/requisição em ASCII, seguida por uma resposta seguindo o padrão RFC 822, com cabeçalho, corpo, e diversos tipos de mídias. Kurose e Ross (2021) apontam que o HTTP usa o protocolo TCP para o transporte de informações.

Tenenbaum (2003) explica os 8 métodos internos de solicitação HTTP:

- **GET:** Solicita a leitura de uma página da Web;
- **HEAD:** Solicita a leitura de um cabeçalho de página da Web;
- **PUT:** Solicita o armazenamento de uma página da Web ;
- **POST:** Acrescenta a um recurso (por exemplo, uma página da Web) ;
- **DELETE:** Remove a página da Web;
- **TRACE:** Ecoa a solicitação recebida;
- **CONNECT:** Reservado para uso futuro ;
- **OPTIONS:** Consulta certas opções;

A primeira linha de uma mensagem de requisição é chamada de linha de requisição, e tem 3 campos: o método interno de solicitação, a URL, e a versão do HTTP sendo usada. As linhas seguintes são chamadas de linhas de cabeçalho, que contêm informações adicionais sobre a requisição do cliente. Cada linha de cabeçalho tem um nome, dois pontos, e o valor correspondente. Exemplos são: Host, Cookie, User-Agent, Connection, e Accept-language. Por fim, solicitações de alguns tipos de métodos também terão um “corpo de entidade” após as linhas de cabeçalho, que vai armazenar diversos tipos de informações, como strings, arquivos json, ou arquivos binários (TENENBAUM. 2003; KUROSE, ROSS. 2021).

Já as respostas HTTP funcionam de forma similar, e possuem três seções: a linha inicial (linha de estado), linhas de cabeçalho, e o corpo da entidade. A linha de estado tem 3 campos: a versão do protocolo, o código de estado/status, e uma mensagem de estado correspondente. Existem 5 grupos de respostas para o código de status: 1xx: Informação; 2xx: Sucesso; 3xx: Redirecionamento; 4xx: Erro do cliente; 5xx: Erro do servidor. Um exemplo de código de status com mensagem de estado correspondente seria: 404 NOT FOUND.

A linha de cabeçalho da resposta é similar à da requisição, e podem ser, por exemplo: Server, Date, Location, Set-Cookie, Content-Type, e Last-Modified. Por fim, o corpo da entidade também irá transferir dados e informações extras ao cliente, e o tipo do dado geralmente é especificado pelo cabeçalho Content-Type.

Ao longo dos anos, diversas versões do protocolo HTTP foram desenvolvidas, com destaque às versões 0.9, 1.0, 1.1, 2.0, e 3.0.

O HTTP 0.9 foi a primeira versão do protocolo a ser implementada e publicada pela W3C, e a sua documentação. Essa versão é simples e minimalista, contendo apenas a request “GET”, e uma resposta em HTML. A conexão cliente-servidor é feita por meio do protocolo TCP-IP, e é terminada quando o documento HTML é totalmente transferido. O servidor não armazena informações sobre as requests (BERNERS-LEE, 1991).

O HTTP 1.0 foi publicado em 1996, e introduziu: novos métodos de request (POST, HEAD, OPTIONS); uma nova estrutura das respostas, que agora possuíam cabeçalhos com metadados; códigos de status que informam o resultado da requisição; e suporte a objetos além de arquivos HTML (BERNERS-LEE, FIELDING. 1996).

O HTTP 1.1 teve como principal vantagem o suporte a conexões persistentes com uma pipeline mais estruturada, capaz de reduzir a latência das solicitações e economizar processamento. Também foi implementado o paralelismo em conexões TCP, permitindo que o host mantenha várias conexões abertas de uma vez (SILVA, 2021).

Em 2009, a google desenvolveu um protocolo não-oficial experimental chamado SPDY, que se mostrou 55% mais rápido que o protocolo HTTP 1.1 (BELSHE, PEON. 2009). O protocolo HTTP/2 foi então publicado em 2015, tendo como principal inspiração o SPDY. Stenberg (2014), aponta que as principais vantagens desse protocolo foram: Multiplexação de fluxos em uma única conexão, priorização de fluxos, compressão de cabeçalho, *Server Push*, cancelamento de mensagens sem encerrar a conexão, e um maior controle do fluxo de conexão. Tudo isso diminuiu a latência de comunicação para a maior parte das redes (SILVA, 2021).

Por fim, foi publicado em 2020 o protocolo HTTP/3, baseado no protocolo QUIC, com uma menor latência na conexão de websites, melhor streaming de vídeos, e menor perda de pacotes (SILVA, 2021). Esse protocolo veio com uma melhor continuidade de conexões, reduziu o número de Round Trip Times (RTT), e trouxe melhorias de segurança. A menor latência é consequência do uso do protocolo UDP ao invés do TCP, o que resolveu o problema de bloqueio de fila do HTTP/2. O alto consumo de CPU, o bloqueio de tráfego UDP por parte de algumas organizações, e a variação de performance por arquitetura de processador são fatores que dificultam a implementação do modelo HTTP/3 (SILVA, 2021).

Normalmente, os pacotes de requisição e resposta dos protocolos HTTP são invisíveis ao usuário. Porém, eles podem ser visualizados usando ferramentas conhecidas como sniffers de rede. Essas ferramentas capturam e analisam o tráfego de rede, permitindo a visualização dos pacotes em tempo real. As duas ferramentas de sniffing de redes mais populares são o Wireshark e o Tcpdump (GOYAL, GOYAL. 2017).

2. OBJETIVOS

A atividade prática realizada no dia 15/08/2024 teve como objetivo entender o funcionamento dos protocolos HTTP por meio da visualização dos pacotes com o uso de um sniffer de rede. Isso foi feito por meio do download de páginas na web, e visualização e comparação dos pacotes com ferramentas como wireshark e h2load.

3. MATERIAL UTILIZADO

Os testes foram realizados numa máquina virtual, executando num notebook da marca DELL, modelo Vostro 3520, com um CPU Intel Core i7-1255U 1.70 GHz, 16Gb de memória RAM DDR4, placa de vídeo integrada Intel Iris Xe Graphics, placa de vídeo dedicada NVIDIA GeForce MX550, adaptador de rede modelo Intel(R) Wi-Fi 6 AX201 160MHz, unidade de disco SSD NVMe ADATA 512Gb, com o sistema operacional Windows 11 Home versão 23H2.

A máquina virtual foi criada com o software Oracle VM Virtualbox, versão 7.0.12. Dentro dela, foi executado o sistema operacional MX Linux versão 6.1.0-21-amd64, uma distro baseada em Debian. O ambiente virtual foi conectado à placa de rede do notebook por meio do modo Bridge. Mais especificações da máquina virtual estão explícitas na figura 1.

4.4 Documentos HTML com objetos:

Foi iniciada mais uma captura do wireshark, e o cache do navegador foi limpo mais uma vez. Após aplicar o filtro http no wireshark, o quarto e último arquivo foi aberto no navegador, por meio do link “<http://gaia.cs.umass.edu/ethereal-labs/HTTP-ethereal-file4.html>”. Os resultados foram capturados, e a captura interrompida para subsequente análise.

4.5 Interação HTTP/2 básica:

Mais uma vez o cache do navegador foi limpo, e mais uma captura foi iniciada no wireshark. Dessa vez, o filtro aplicado foi “http2”. Depois, foi executado, no terminal Linux, o seguinte comando: “curl --http2 -v nghttp2.org/robots.txt” e “curl --http2 -v nghttp2.org/humans.txt”. A captura foi finalizada, e os pacotes verificados.

4.6 Compressão do cabeçalho – HPACK:

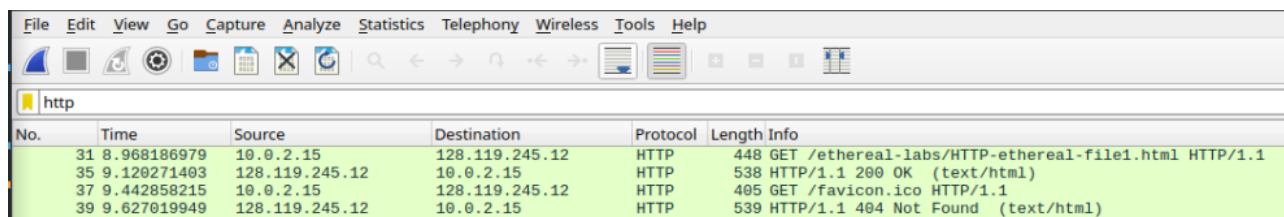
Inicialmente, foi instalado o pacote nghttp2-client. Depois, foi executado o comando “h2load <https://blog.cloudflare.com>”. Depois, o mesmo foi testado para 2, 3, 4, e 5 requisições, com o comando “h2load <https://blog.cloudflare.com> -n X | tail -6 | head -1”, onde X = 2, 3, 4, e 5. A taxa de compressão foi avaliada para todos os testes.

4.7 Comparação HTTP/1.1 e HTTP/2:

Por fim, foi acessado o website <http://www.http2demo.io/>, aonde podemos checar as diferenças de velocidade entre os protocolos HTTP/1.1 e HTTP/2. Os resultados do teste foram anotados.

5. RESULTADOS E DISCUSSÃO

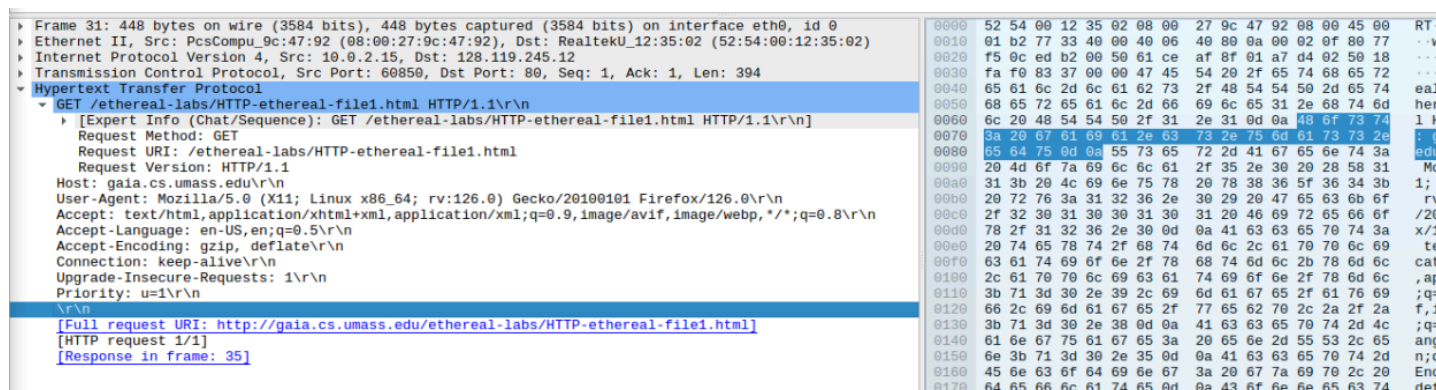
5.1 Interação HTTP GET/resposta básica:



No.	Time	Source	Destination	Protocol	Length	Info
31	8.968186979	10.0.2.15	128.119.245.12	HTTP	448	GET /ethereal-labs/HTTP-ethereal-file1.html HTTP/1.1
35	9.120271403	128.119.245.12	10.0.2.15	HTTP	538	HTTP/1.1 200 OK (text/html)
37	9.442858215	10.0.2.15	128.119.245.12	HTTP	405	GET /favicon.ico HTTP/1.1
39	9.627019949	128.119.245.12	10.0.2.15	HTTP	539	HTTP/1.1 404 Not Found (text/html)

Figura 2: Informações coletadas pelo sniffer de rede no arquivo 1.

Ao avaliar as informações coletadas pelo sniffer de rede (Figura 2), foi possível coletar diversas informações sobre o cliente e o servidor. Na seção info, é possível verificar que tanto o GET quanto a resposta foram feitas seguindo o protocolo HTTP versão 1.1. Ao selecionarmos o primeiro pacote GET, e clicarmos em Hypertext Transfer Protocol, podemos obter mais informações do cliente (Figura 3).



Frame 31: 448 bytes on wire (3584 bits), 448 bytes captured (3584 bits) on interface eth0, id 0	0000	52 54 00 12 35 02 08 00	27 9c 47 92 68 00 45 00	RT-
Ethernet II, Src: PcsCompu_9c:47:92 (08:00:27:9c:47:92), Dst: RealtekU_12:35:02 (52:54:00:12:35:02)	0010	01 b2 77 33 40 00 40 06	40 89 0a 00 02 0f 89 77	..w
Internet Protocol Version 4, Src: 10.0.2.15, Dst: 128.119.245.12	0020	f5 0c ed b2 00 50 61 ce	af 8f 01 a7 d4 02 50 18	...
Transmission Control Protocol, Src Port: 60850, Dst Port: 80, Seq: 1, Ack: 1, Len: 394	0030	fa f0 83 37 00 00 47 45	54 29 2f 65 74 68 65 72	...
Hypertext Transfer Protocol	0040	65 61 6c 2d 6c 61 62 73	2f 48 54 54 50 2d 65 74	eal
GET /ethereal-labs/HTTP-ethereal-file1.html HTTP/1.1	0050	68 65 72 65 61 6c 2d 66	69 6c 65 31 2e 68 74 6d	her
[Expert Info (Chat/Sequence): GET /ethereal-labs/HTTP-ethereal-file1.html HTTP/1.1]	0060	6c 20 48 54 54 50 2f 31	2e 31 0d 0a 48 6f 73 74	1 H
Request Method: GET	0070	3a 20 67 61 69 61 2e 63	73 2e 75 60 61 73 73 2e	! g
Request URI: /ethereal-labs/HTTP-ethereal-file1.html	0080	65 64 75 0d 0a 55 73 65	72 2d 41 67 65 6e 74 3a	edu
Request Version: HTTP/1.1	0090	20 4d 6f 7a 69 6c 6c 61	2f 35 2e 30 20 28 58 31	Mo
Host: gaia.cs.umass.edu	00a0	31 3b 20 4c 69 6e 75 78	20 78 38 36 5f 36 34 3b	1;
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:126.0) Gecko/20100101 Firefox/126.0	00b0	20 72 76 3a 31 32 36 2e	30 29 20 47 65 63 6b 6f	rv
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8	00c0	2f 32 30 31 30 30 31 30	31 29 46 69 72 65 6b 6f	/20
Accept-Language: en-US,en;q=0.5	00d0	78 2f 31 32 36 2e 30 0d	0a 41 63 63 65 70 74 3a	x/1
Accept-Encoding: gzip, deflate	00e0	20 74 65 78 74 2f 68 74	6d 6c 2c 61 70 70 6c 69	te
Connection: keep-alive	00f0	63 61 74 69 6f 6e 2f 78	68 74 6d 6c 2b 78 6d 6c	cat
Upgrade-Insecure-Requests: 1	0100	2c 61 70 70 6c 69 63 61	74 69 6f 6e 2f 78 6d 6c	;ap
Priority: u=1	0110	3b 71 3d 30 2e 39 2c 69	6d 61 67 65 2f 61 76 69	;q=
Full request URI: http://gaia.cs.umass.edu/ethereal-labs/HTTP-ethereal-file1.html	0120	66 2c 69 6d 61 67 65 2f	77 65 62 70 2c 2a 2f 2a	f,1
HTTP request 1/1	0130	3b 71 3d 30 2e 38 0d 0a	41 63 63 65 70 74 2d 4c	;q=
Response in frame: 35	0140	61 6e 67 75 61 67 65 3a	20 65 6e 2d 55 53 2c 65	ang
	0150	6e 3b 71 3d 30 2e 35 0d	0a 41 63 63 65 70 74 2d	n;q
	0160	45 6e 63 6f 64 69 6e 67	3a 29 67 7a 69 70 2c 20	Enc
	0170	64 65 66 6c 61 74 65 0d	0a 43 6f 6e 6e 65 63 74	def

Figura 3: Detalhes do protocolo HTTP na ferramenta Wireshark..

Podemos verificar que: o navegador aceita respostas nas linguagens “en” e “en-US”, o endereço de IP do cliente é “10.0.2.15”, e o do servidor é “128.119.245.12”. Ao selecionarmos o pacote da resposta no wireshark, obteremos mais informações: Vemos que a requisição foi um sucesso, pois o código resposta do servidor foi “200”, que significa OK; também percebemos que a última modificação do HTML ocorreu no dia 27 de agosto de 2024, às 05:59:02 GMT, 16 horas antes da requisição. Por fim, verificamos que foram retornados 126 bytes de conteúdo.

5.2 Interação HTTP CONDITIONAL GET/resposta:

Ao analisarmos a figura 4 podemos ver que, dessa vez, ocorreu a troca de mais 2 pacotes, provavelmente devido ao *refresh* que foi feito no navegador. Se observarmos o terceiro GET, podemos ver que o navegador adicionou mais duas linhas ao cabeçalho da nova requisição: “If-Modified_Since”. Verificamos que isso aconteceu apenas na requisição GET feita após o refresh da página. Além disso, a resposta desse GET pós-refresh foi diferente: o corpo da resposta veio vazio, e o código foi “304 - Not Modified”. Isso provavelmente foi implementado com o intuito de evitar uma transferência desnecessária de novos arquivo.

Por fim, vale mencionar o segundo GET. Ele tentou buscar um arquivo chamado “favicon.ico”, mas obteve a resposta “404 - Not Found”.

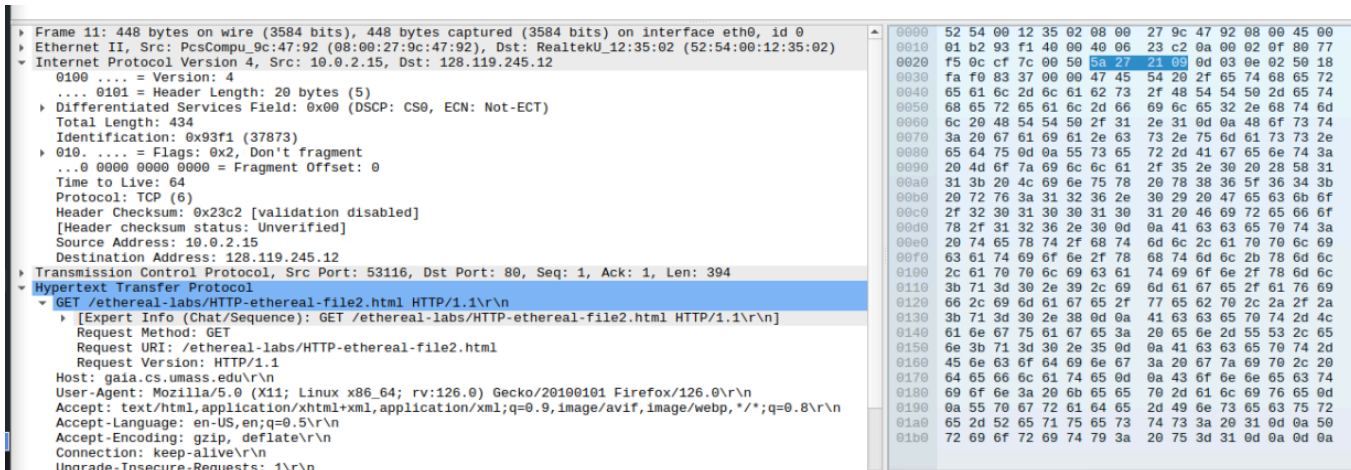


Figura 4: Informações coletadas pelo sniffer de rede no arquivo 2.

5.3 Obtendo documentos longos:

No.	Time	Source	Destination	Protocol	Length Info
138	82.015013333	10.0.2.15	128.119.245.12	HTTP	448 GET /ethereal-labs/HTTP-ethereal-file3.html HTTP/1.1
142	82.172990305	128.119.245.12	10.0.2.15	HTTP	1995 HTTP/1.1 200 OK (text/html)

Figura 5: Informações coletadas pelo sniffer de rede no arquivo 3.

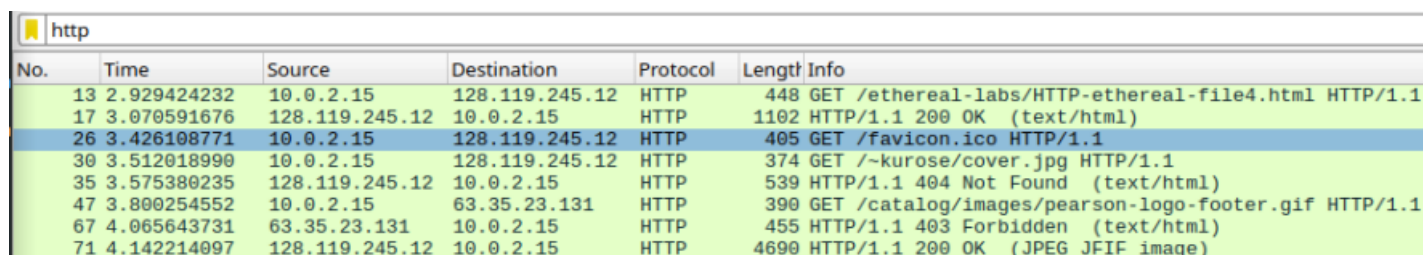
Inicialmente vemos que, quando apenas o filtro http está aplicado, podemos ver duas transmissões: o GET e o OK. Porém, o pacote GET tem o número 138, enquanto o OK tem o número 142 (Figura 5). Isso significa que há alguns outros pacotes entre eles. Isso ocorre pois o arquivo é grande, e foi dividido em vários pacotes menores pelo protocolo tcp. Se mudarmos o filtro pra http|tcp, podemos ver esses vários pacotes entre eles, conforme a figura 6.

No.	Time	Source	Destination	Protocol	Length Info
131	81.866291968	10.0.2.15	128.119.245.12	TCP	74 38538 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM TSval=2996742448 TSecr=...
132	81.867316274	10.0.2.15	128.119.245.12	TCP	74 38544 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM TSval=2996742449 TSecr=...
134	82.012913855	128.119.245.12	10.0.2.15	TCP	60 80 → 38544 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1460
135	82.013450260	10.0.2.15	128.119.245.12	TCP	54 38544 → 80 [ACK] Seq=1 Ack=1 Win=64240 Len=0
136	82.014894343	128.119.245.12	10.0.2.15	TCP	60 80 → 38538 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1460
137	82.014923139	10.0.2.15	128.119.245.12	TCP	54 38538 → 80 [ACK] Seq=1 Ack=1 Win=64240 Len=0
138	82.015013333	10.0.2.15	128.119.245.12	HTTP	448 GET /ethereal-labs/HTTP-ethereal-file3.html HTTP/1.1
139	82.015856812	128.119.245.12	10.0.2.15	TCP	60 80 → 38544 [ACK] Seq=1 Ack=395 Win=65535 Len=0
140	82.172130836	128.119.245.12	10.0.2.15	TCP	2974 80 → 38544 [ACK] Seq=1 Ack=395 Win=65535 Len=2920 [TCP segment of a reassembled P...
141	82.172310732	10.0.2.15	128.119.245.12	TCP	54 38544 → 80 [ACK] Seq=395 Ack=2921 Win=62780 Len=0
142	82.172990305	128.119.245.12	10.0.2.15	HTTP	1995 HTTP/1.1 200 OK (text/html)
143	82.173114365	10.0.2.15	128.119.245.12	TCP	54 38544 → 80 [ACK] Seq=395 Ack=4862 Win=62780 Len=0
144	82.319112875	10.0.2.15	128.119.245.12	HTTP	405 GET /favicon.ico HTTP/1.1
145	82.320088618	128.119.245.12	10.0.2.15	TCP	60 80 → 38538 [ACK] Seq=1 Ack=352 Win=65535 Len=0
152	82.363631278	10.0.2.15	34.149.100.209	TCP	74 60000 → 443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM TSval=4268134079 TSecr=...
153	82.387981642	34.149.100.209	10.0.2.15	TCP	60 443 → 60000 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1460

Figura 6: Segmentação dos pacotes do arquivo 3.

A request inicial da página web se inicia com um pacote [SYN], e termina com um pacote [FIN, ACK] – ambos no protocolo TCP. Também verificamos que houve apenas um GET que requisitou o documento HTML e, até receber a resposta “200 - OK”, houve uma troca de 3 pacotes TCP. Podemos ver que um dos pacotes TCP está marcado como “[TCP Segment of a reassembled PDU]”. Isso significa que ele contém uma das partes do pacote total, que fora segmentado por ser grande demais.

5.4 Documentos HTML com objetos:



A screenshot of a Wireshark network capture showing HTTP traffic. The filter is set to 'http'. The packet list shows several GET requests. The packet details pane for packet 26 shows the request for /favicon.ico.

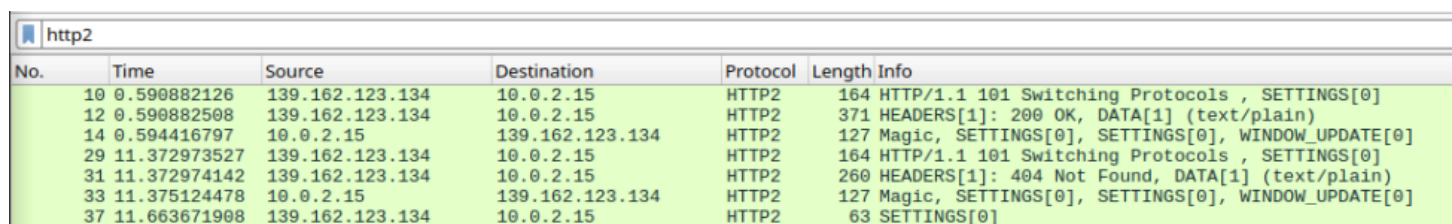
No.	Time	Source	Destination	Protocol	Length	Info
13	2.929424232	10.0.2.15	128.119.245.12	HTTP	448	GET /ethereal-labs/HTTP-ethereal-file4.html HTTP/1.1
17	3.070591676	128.119.245.12	10.0.2.15	HTTP	1102	HTTP/1.1 200 OK (text/html)
26	3.426108771	10.0.2.15	128.119.245.12	HTTP	405	GET /favicon.ico HTTP/1.1
30	3.512018990	10.0.2.15	128.119.245.12	HTTP	374	GET /~kurose/cover.jpg HTTP/1.1
35	3.575380235	128.119.245.12	10.0.2.15	HTTP	539	HTTP/1.1 404 Not Found (text/html)
47	3.800254552	10.0.2.15	63.35.23.131	HTTP	390	GET /catalog/images/pearson-logo-footer.gif HTTP/1.1
67	4.065643731	63.35.23.131	10.0.2.15	HTTP	455	HTTP/1.1 403 Forbidden (text/html)
71	4.142214097	128.119.245.12	10.0.2.15	HTTP	4690	HTTP/1.1 200 OK (JPEG JFIF image)

Figura 7: Informações coletadas pelo sniffer de rede no arquivo 4.

Desde já percebemos que esta página é um pouco mais complexa, sendo composta pelo HTML, uma imagem, um ícone, e um gif (Figura 7). Logo, foram necessárias 4 requisições GET, uma para cada tipo de objeto. Porém, percebemos que dois servidores diferentes foram acessados: os GETs do HTML, da imagem, e do ícone tiveram como destino o endereço de IP “128.119.245.12”, enquanto o GET do gif teve como destino o endereço “63.35.23.131”. Isso está explicado no corpo do documento HTML: o gif está armazenado no site de uma publicadora de livros.

Além disso, verificamos que os pacotes GET do ícone e do .jpg foram enviados um após o outro. Isso pode indicar que esse processo ocorreu de forma paralela, já que a resposta aconteceu após o envio dos GETs. Se fosse serial, a resposta do primeiro GET viria antes do envio do segundo GET.

5.5 Interação HTTP/2 básica:



A screenshot of a Wireshark network capture showing HTTP/2 traffic. The filter is set to 'http2'. The packet list shows several HTTP/2 frames. The packet details pane for packet 10 shows the SETTINGS frame.

No.	Time	Source	Destination	Protocol	Length	Info
10	0.590882126	139.162.123.134	10.0.2.15	HTTP2	164	HTTP/1.1 101 Switching Protocols , SETTINGS[0]
12	0.590882508	139.162.123.134	10.0.2.15	HTTP2	371	HEADERS[1]: 200 OK, DATA[1] (text/plain)
14	0.594416797	10.0.2.15	139.162.123.134	HTTP2	127	Magic, SETTINGS[0], SETTINGS[0], WINDOW_UPDATE[0]
29	11.372973527	139.162.123.134	10.0.2.15	HTTP2	164	HTTP/1.1 101 Switching Protocols , SETTINGS[0]
31	11.372974142	139.162.123.134	10.0.2.15	HTTP2	260	HEADERS[1]: 404 Not Found, DATA[1] (text/plain)
33	11.375124478	10.0.2.15	139.162.123.134	HTTP2	127	Magic, SETTINGS[0], SETTINGS[0], WINDOW_UPDATE[0]
37	11.663671908	139.162.123.134	10.0.2.15	HTTP2	63	SETTINGS[0]

Figura 8: Informações coletadas pelo sniffer de rede para o protocolo HTTP/2.

Inicialmente, percebe-se que o primeiro GET é feito em HTTP/1.1, mas ocorreu uma “troca de protocolos”, saindo do protocolo HTTP/1.1, e trocando para o protocolo HTTP/2 (Figura 8). Isso é identificado pelo código de status 101 - Switching Protocols. Após a troca para o protocolo HTTP/2, podemos ver que existem algumas pequenas diferenças na forma que o wireshark nos mostra os headers do pacote. Também verificamos que é um pouco mais difícil de localizar os pacotes de resposta, pois a seção *info* do wireshark parece mais poluída.

5.6 Compressão do cabeçalho – HPACK:

A primeira execução obteve uma taxa de compressão de 23.61%. Os resultados completos da execução podem ser visualizados na Figura 9, enquanto os testes seguintes são vistos na Figura 10.


```

$ h2load https://blog.cloudflare.comseeing this, you've downloaded the page correctly
starting benchmark...
spawning thread #0: 1 total client(s). 1 total requests
TLS Protocol: TLSv1.3
Cipher: TLS_AES_128_GCM_SHA256
Server Temp Key: X25519 253 bits
Application protocol: h2
progress: 100% done

finished in 749.19ms, 1.33 req/s, 124.51KB/s
requests: 1 total, 1 started, 1 done, 1 succeeded, 0 failed, 0 errored, 0 timeout
status codes: 1 2xx, 0 3xx, 0 4xx, 0 5xx
traffic: 93.28KB (95519) total, 686B (686) headers (space savings 23.61%), 92.43KB (94649) data

```

	min	max	mean	sd	+/-	sd
time for request:	645.41ms	645.41ms	645.41ms	0us	100.00%	
time for connect:	99.78ms	99.78ms	99.78ms	0us	100.00%	
time to 1st byte:	672.74ms	672.74ms	672.74ms	0us	100.00%	
req/s	1.34	1.34	1.34	0.00	100.00%	

Figura 9: Resultados da execução do h2load para uma requisição.

```

radajaa@roboroto:~$ h2load https://blog.cloudflare.com -n 2 | tail -6 | head -1
traffic: 186.29KB (190759) total, 1.11KB (1133) headers (space savings 36.63%), 184.86KB (189298) data
radajaa@roboroto:~$ h2load https://blog.cloudflare.com -n 3 | tail -6 | head -1
traffic: 279.97KB (286691) total, 2.18KB (2236) headers (space savings 35.82%), 277.29KB (283947) data
radajaa@roboroto:~$ h2load https://blog.cloudflare.com -n 4 | tail -6 | head -1
traffic: 372.94KB (381890) total, 2.62KB (2678) headers (space savings 38.86%), 369.72KB (378596) data
radajaa@roboroto:~$ h2load https://blog.cloudflare.com -n 5 | tail -6 | head -1
traffic: 465.34KB (476505) total, 2.47KB (2527) headers (space savings 43.54%), 462.15KB (473245) data

```

Figura 10: Resultados da execução do h2load para múltiplas requisições.

Com duas requisições, houve uma compressão de 36.63%. Com três, 35.82%. Com quatro, 38.86%. Com cinco, 43.54%.

5.7 Comparação HTTP/1.1 e HTTP/2:

Ao ser acessado, o website mostra uma figura composta por 200 imagens que são carregadas pelos protocolos HTTP/1.1 e HTTP/2, junto com um timer. Após a execução dos testes, verificou-se que, com o protocolo HTTP/1.1, as imagens demoraram 5.82 segundos para serem carregadas, enquanto que o protocolo HTTP/2 fez isso em 1.47 segundos, mais de 5 vezes mais rápido. Os resultados estão explícitos na Figura 11.

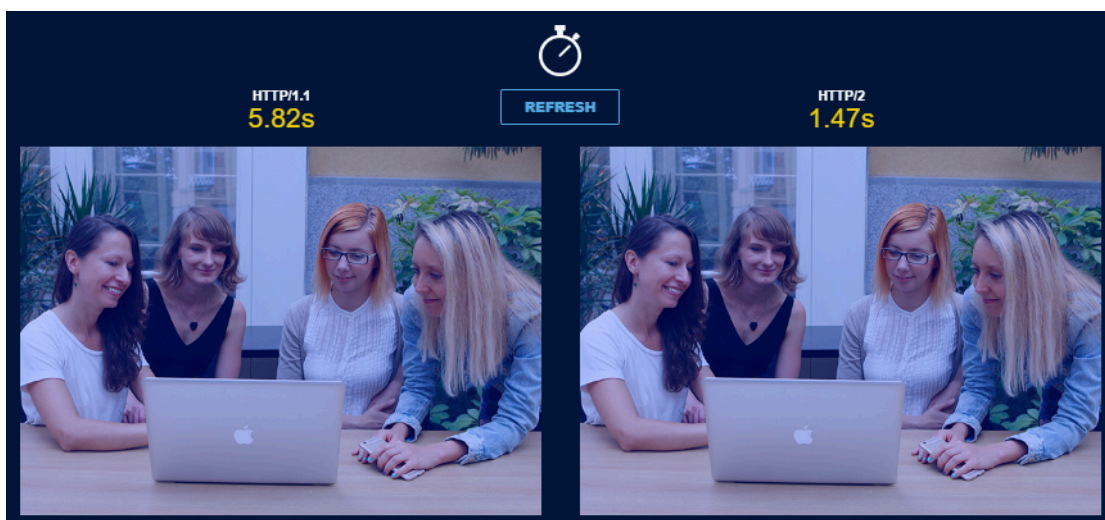


Figura 11: Comparação de velocidade entre os protocolos HTTP/1.1 e HTTP/2..

6. CONCLUSÕES

A aula prática teve como objetivo clarificar alguns dos conceitos básicos relacionados ao funcionamento dos protocolos HTTP, bem como permitir a familiarização com ferramentas sniffer de rede. Nesse sentido, conclui-se que o trabalho foi um sucesso, pois foi possível completar todas as atividades propostas, bem como obter um conhecimento básico sobre o funcionamento da ferramenta Wireshark.

O sniffer de rede permitiu uma ótima visualização da estruturação das *requests* e *responses* do protocolo HTTP, permitindo identificar bem os métodos internos, os cabeçalhos, os códigos de status, e o corpo das entidades. Também foi possível visualizar a segmentação de pacotes grandes por meio do protocolo TCP, e o paralelismo no download de imagens. Por fim, as interações com o protocolo HTTP/2 mostraram claramente as vantagens que este tem em relação ao protocolo HTTP/1.1, quando o assunto é velocidade de comunicação.

BIBLIOGRAFIA

ALI, Amer Nizar Abu. Comparison study between IPV4 & IPV6. **International Journal of Computer Science Issues (IJCSI)**, v. 9, n. 3, p. 314, 2012.

BELSHE, M.; PEON, R. A 2x Faster Web. 2009. <<https://blog.chromium.org/2009/11/2x-faster-web.html>>. Acesso em: 24/08/2024.

BERNERS-LEE, Tim. **The Original HTTP as defined in 1991**. World Wide Web Consortium. 1991. Disponível em: <<https://www.w3.org/Protocols/HTTP/AsImplemented.html>>. Acesso em 24 de agosto de 2024.

BERNERS-LEE, Tim; FIELDING, Roy; FRYSTYK, Henrik. **Hypertext transfer protocol--HTTP/1.0**. Network Working Group. 1996.

CAMPISTA, Miguel Elias M. et al. Interconexão de Redes na Internet do Futuro: Desafios e Soluções. **Minicursos do Simpósio Brasileiro de Redes de Computadores-SBRC**, v. 2010, p. 47-101, 2010.

CHANDRA, Deka Ganesh; KATHING, Margaret; KUMAR, Das Prashanta. A comparative study on IPv4 and IPv6. In: **2013 International Conference on Communication Systems and Network Technologies**. IEEE, 2013. p. 286-289.

CORONA, Adrián Estrada. *et al.* **Protocolos TCP/IP de internet**. 2004.

GOYAL, Piyush; GOYAL, Anurag. Comparative study of two most popular packet sniffing tools-Tcpdump and Wireshark. In: **2017 9th International Conference on Computational Intelligence and Communication Networks (CICN)**. IEEE, 2017. p. 77-81.

KUROSE, James F.; ROSS, Keith W. **Redes de Computadores e a Internet: Uma abordagem top-down**. Trad. 8 ed. Sao Paulo: Francisco Araújo da Costa, 2021.

LEINER, Barry M. et al. **A brief history of the Internet**. ACM SIGCOMM computer communication review, v. 39, n. 5, p. 22-31, 2009.

MONTEIRO, Luís. **A internet como meio de comunicação: possibilidades e limitações**. In: Congresso Brasileiro de Comunicação. sn, 2001.

SILVA, Antonio Vinicius Ferreira e. **UMA ANÁLISE COMPARATIVA DAS VERSÕES DO PROTOCOLO HTTP:EVOLUÇÃO E PONTOS QUE AMPLIEM O USO DO HTTP/3**. Unichristus. Trabalho de Conclusão de Curso. Fortaleza, Ceará. 2021

STENBERG, D. **HTTP2 explained**. [S.l.]: GitBook, 2014. <<https://http2-explained.haxx.se/en>>. Acesso em: 24/08/2024.

TANENBAUM, Andrew S. **Redes de Computadores**, 7ª Edição, Editora Campus, Rio de Janeiro – RJ, 2003.