PTC 3360

2. Introdução às camadas superiores
2.1 A camada de aplicação

(Kurose, Seções 2.1 e 2.2)

Agosto 2025

2. Introdução às camadas superiores

- Neste capítulo do curso, será feita uma introdução a aspectos das 3 camadas superiores da pilha de protocolos Internet (Aplicação, Transporte e Rede).

- A abordagem utilizada é a *top-down*, começando da camada de aplicação, mais próxima dos usuários, em direção à camada física.

- Esses assuntos são detalhados em cursos posteriores de graduação e pós-graduação.

Conteúdo

2.1 A camada de aplicação

2.2 Princípios da transferência confiável de dados (recorte da camada de transporte)

2.3 Camada de rede

Alguns *aplicativos* de rede

- *E-mail*
- *World Wide Web (WWW)*
- Whatsapp
- *Waze, Google Maps, Apple Plans*
- BitTorrent
- Jogos em rede multiusuários
- *Streaming* de vídeo armazenado (YouTube, Disney+, Netflix)
- Voz-sobre-IP
- Videoconferência em tempo real (Zoom, Google Meet)
- Redes sociais
- Assistentes AI (Gemini, Chat GPT, Copilot, ...)

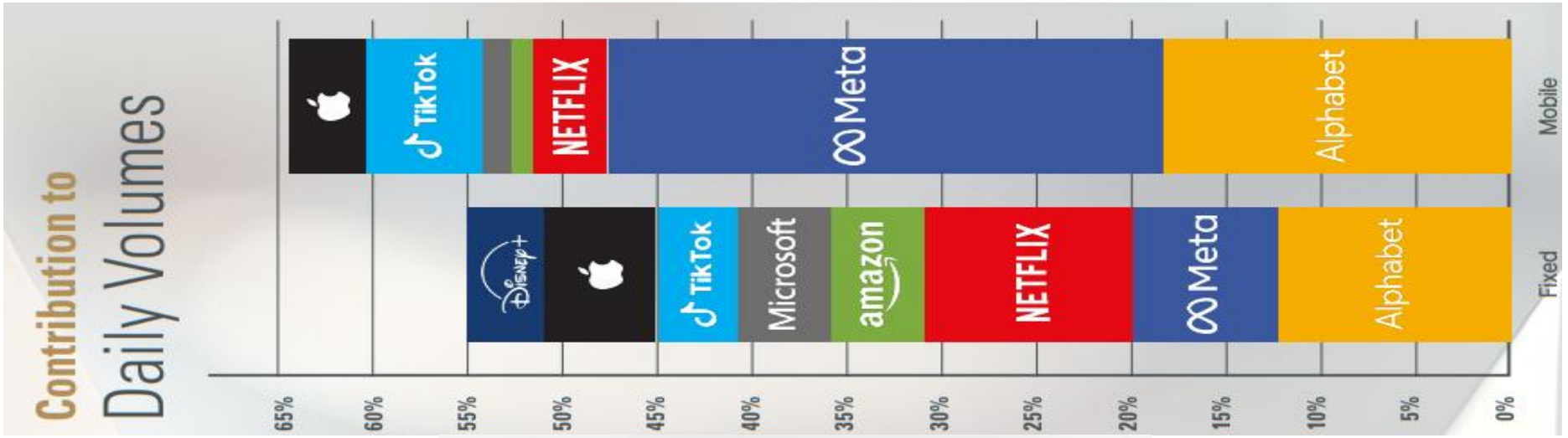
Tráfego Global – 2025 – Fixo

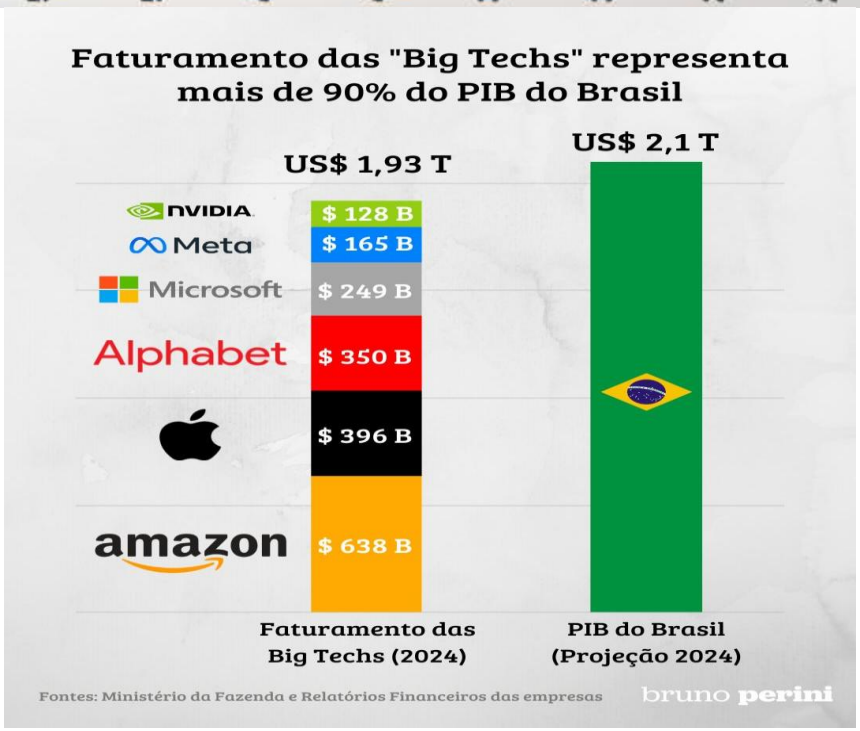
App Category/Name	% Users	YoY	Volume	YoY
Video				
YouTube	88%	0%	1.5 GB	21%
Netflix	66%	1%	1.6 GB	15%
Amazon Prime	49%	7%	644 MB	51%
Disney+	32%	4%	635 MB	17%
Generic Video	20%	-15%	1MB	-12%
Social Media				
Facebook	90%	0%	577 MB	-3%
Tik Tok	64%	2%	490 MB	7%
LinkedIn	64%	1%	3 MB	2%
Instagram	63%	-5%	189 MB	17%
Pinterest	62%	-1%	6 MB	1%
Device Gaming				
Steam	78%	0%	279 MB	20%
EA Game	36%	-1%	1 MB	29%
Unity engine	36%	8%	7 MB	57%
Xbox Live	17%	-14%	380 MB	-13%
Minecraft	12%	31%	2 MB	34%
Television				
Apple TV+	25%	2%	62 MB	13%
Tubi TV	17%	0%	77 MB	25%
Pluto TV	13%	9%	49 MB	-5%
Roku Channel	6%	6%	801 KB	16%
Plex	5%	10%	7 MB	24%
Audio				
Spotify	60%	5%	36 MB	13%
Apple Music	49%	2%	18 MB	5%
Podcast Services	15%	-9%	10 MB	-43%
Amazon Music	9%	2%	18 MB	-2%
Shazam	6%	16%	41 KB	54%
Cloud Gaming				
Playstation Now	3%	-5%	849 KB	45%
Tencent	3%	97%	15 KB	26%
GeForce Now	1%	27%	2 MB	27%
YouTube Playables	1%	36%	79 KB	187%
Playkey	1%	15%	35 KB	8%
Communication				
FB Messenger	80%	1%	31 MB	34%
WhatsApp	80%	1%	15 MB	20%
Google Messaging	46%	1%	540 KB	-26%
FaceTime	28%	-3%	109 MB	25%
iMessage	19%	14%	647 KB	6%
Conferencing				
Google Meet	54%	45%	5 MB	12%
Microsoft Teams	54%	-2%	48 MB	-16%
Skype	9%	-2%	983 KB	-21%
Zoom	8%	33%	13 MB	-5%
Zoho	4%	12%	214 KB	49%
IoT				
Apple Siri	56%	-5%	237 KB	-21%
Bixby	37%	4%	101 KB	13%
Amazon Alexa	31%	14%	2 MB	-45%
Google Home	17%	14%	78 KB	18%
SmartThings	13%	13%	203 KB	21%

Tráfego Global – 2025 - Móvel

App Category/Name	% Users	YoY	Volume	YoY
Video				
YouTube	72%	-3%	89 MB	7%
XVideos	9%	0%	10 MB	38%
Netflix	8%	10%	20 MB	22%
Generic Video	3%	-29%	6 KB	-134%
Amazon Prime	2%	22%	3 MB	11%
Social Media				
Facebook	77%	-3%	137 MB	-22%
Instagram	47%	1%	45 MB	-12%
Tik Tok	37%	2%	35 MB	5%
Snapchat	29%	3%	16 MB	34%
X (Twitter)	28%	-5%	2 MB	-1%
Device Gaming				
Unity engine	20.5%	5%	1 MB	27%
EA Game	7.1%	-41%	153 KB	-11%
Generic Gaming	1.3%	-8%	466 KB	-7%
Supercell	1.2%	-6%	35 KB	-30%
ROBLOX	1.0%	14%	3 MB	23%
Television				
Apple TV+	1.9%	-1%	190 KB	43%
Plex	0.1%	0%	27 KB	-10%
Sky	0.1%	37%	5 KB	-28%
Pluto TV	0.0%	0%	7 KB	24%
Roku	0.0%	0%	1 KB	40%
Audio				
Apple Music	12.1%	13%	704 KB	-4%
Spotify	10.1%	5%	2 MB	16%
SoundCloud	1.2%	7%	185 KB	29%
YouTube Music	1.0%	-2%	5 KB	28%
Shazam	0.9%	-1%	3 KB	99%
Cloud Gaming				
Tencent	2.3%	7%	6 KB	-11%
Garena+	1.5%	21%	258 KB	18%
Playkey	0.3%	-11%	9 KB	-4%
Playstation Now	0.2%	-7%	1 KB	15%
YouTube Playables	0.1%	63%	3 KB	158%
Communication				
WhatsApp	80.0%	0%	43 MB	19%
Google Messaging	67.2%	-1%	171 KB	-11%
FB Messenger	64.4%	-4%	1 MB	-10%
Telegram	12.8%	12%	7 MB	4%
FaceTime	5.6%	-2%	866 KB	6%
Conferencing				
Microsoft Teams	27.0%	-4%	999 KB	-30%
Google Meet	26.9%	34%	376 KB	16%
Zoom	1.0%	2%	579 KB	-5%
Skype	0.6%	-6%	25 KB	-34%
Zoho	0.4%	0%	8 KB	4%
IoT				
Apple Siri	23.6%	6%	43 KB	11%
Bixby	18.5%	-3%	26 KB	6%
SmartThings	3.3%	31%	8 KB	71%
Google Home	3.1%	22%	7 KB	36%
Amazon Alexa	0.7%	-6%	17 KB	-34%

Os “donos da Internet”





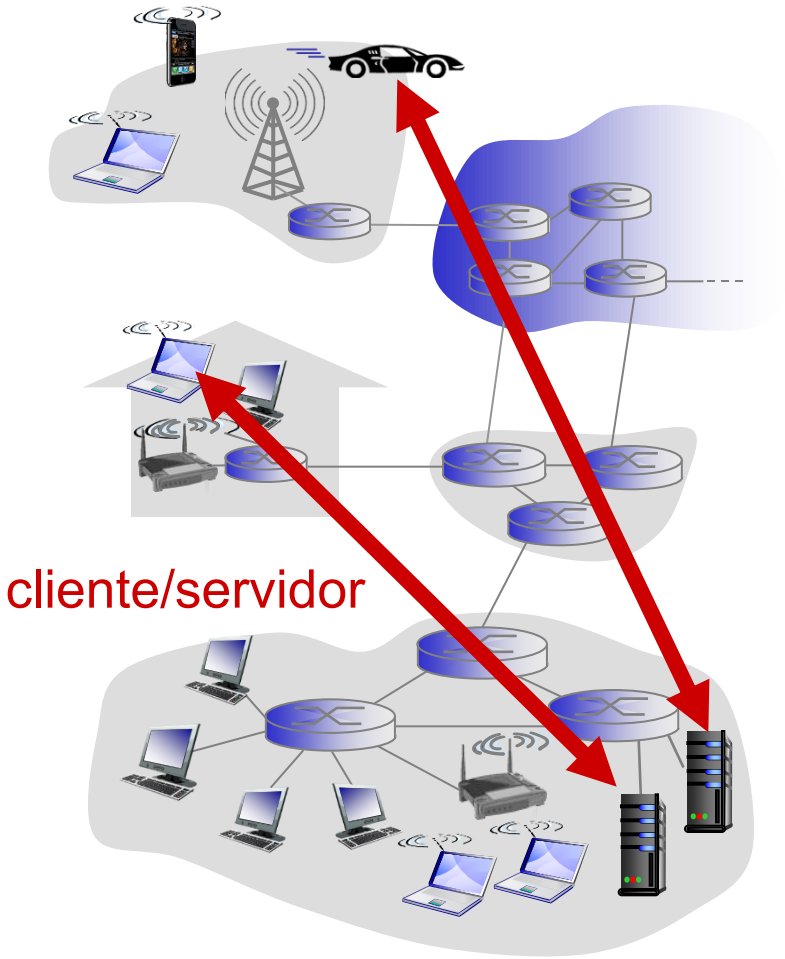
Enquanto isso, na China...

Arquiteturas de aplicativos

Possíveis estruturas de aplicativos:

- A. Cliente-servidor
- *B. Peer-to-peer* (P2P)

A. Arquitetura cliente-servidor



Servidor:

- em *host* sempre ligado
- tem endereço IP permanente
- *podem estar em data centers*, pensando em escala (servidor virtual)

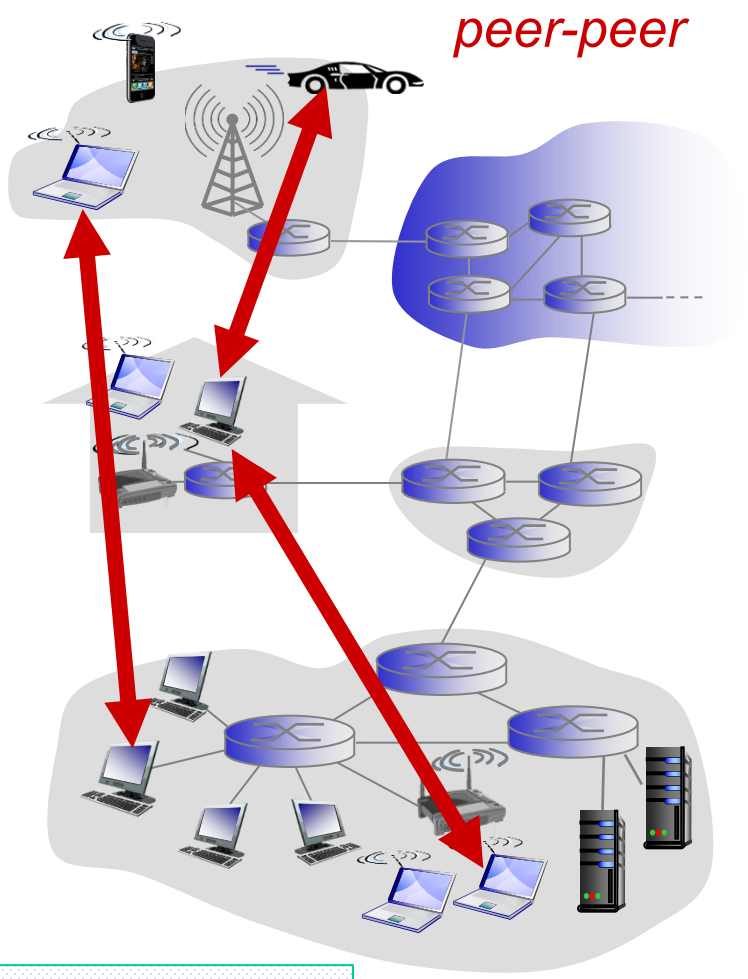
Clientes:

- comunicam-se com servidor e não diretamente entre si
- podem se conectar de forma intermitente
- podem ter endereços IP dinâmicos

Web, E-mail, *Streaming* de vídeo, ...

B. Arquitetura P2P

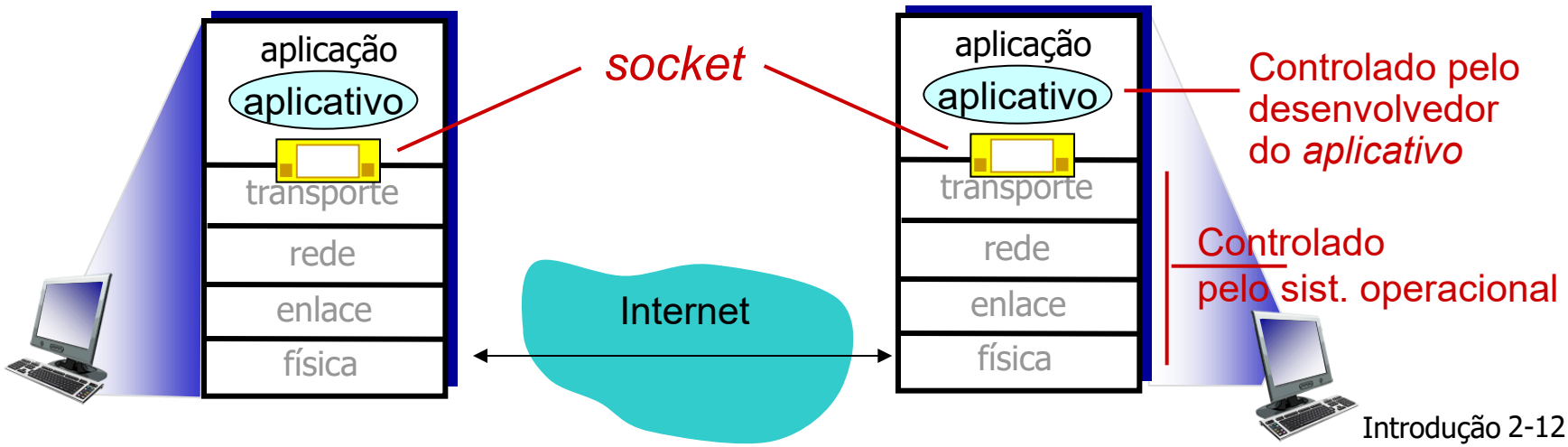
- Dependência mínima de servidores dedicados
- Comunicação direta ente sistemas finais (*peers*)
- *Peers* requerem serviço de outros *peers* – provêm serviço para outros *peers* em retorno
  - *Autoescalável* – novos *peers* trazem novas demandas de serviço, mas também fornecem serviço
- *Peers* conectam-se intermitente-mente e mudam endereço IP
  - gerenciamento mais complexo
- Exemplo: BitTorrent, Bitcoin.



Discussão:
Whatsapp mensagens de texto, fotos e arquivos é cliente-servidor ou P2P? O que motiva essa escolha de arquitetura?

Sockets

- Aplicativo envia/recebe mensagens de/para uma interface de *software* (API -*Application Programming Interface*) : *socket*
- Aplicativo transmissor envia mensagem pela porta
- Ele confia na infraestrutura de transporte do outro lado da porta para entregar mensagem ao *socket* no aplicativo receptor
- Aplicativo receptor lê dados do *socket*



Endereçamento de processos

- Para receber mensagens, processo precisa ter *identificador*
- Dispositivo *host tem endereço* IP *de* 32 bits único (IPv4)
- Endereço IP do *host* em que o processo roda é suficiente para identificar o processo?
  - *Resposta:* Não! *Muitos* processos podem estar rodando no mesmo *host*!

- *Identificador* inclui tanto endereço IP (32 bits) quanto número da porta associado com processo no *host*.
- Exemplos de números de porta:
  - Servidor *web* (HTTP): 80
  - Servidor *web* (HTTPS): 443
  - Servidor de *e-mail (SMTP)*: 25
  - Gerenciados pela *Internet Assigned Numbers Authority* (IANA)
  - http://www.iana.org
- Para enviar mensagem HTTP para servidor *web* www.usp.br:
  - Endereço IP: 200.144.248.41
  - Número de porta: 80
- Mais em breve…

Protocolo da camada de aplicação define

- Tipos de mensagens trocadas
  - Por exemplo, requisição, resposta
- Sintaxe da mensagem
  - Campos da mensagem e como eles são delineados
- Semântica das mensagens
  - Significado das informações dos campos
- Regras para quando e como aplicativos enviam e respondem mensagens

Protocolos abertos:

- Definidos em RFCs
- Permitem interoperabilidade
- Exemplos:
  - HTTP [RFC 9110]
  - SMTP [RFC 5321]

Protocolos proprietários:

- Por exemplo, Microsoft Teams.

Exemplo de Aplicação: Web e HTTP 1.1

Primeiro, alguns conceitos básicos…

- Uma *página web* consiste de *objetos (arquivos).*
- Um objeto pode ser um arquivo HTML, uma imagem JPEG, um arquivo de áudio, um vídeo, etc.
- Uma página *web* consiste de *"programa" HTML base* que inclui *diversos objetos referenciados.*
- Cada objeto é endereçável por uma *URL (Uniform Resource Locator)*, por exemplo,

`www.lcs.poli.usp.br/~marcio/index_arquivos/image002.jpg`

nome do *host* | local do objeto

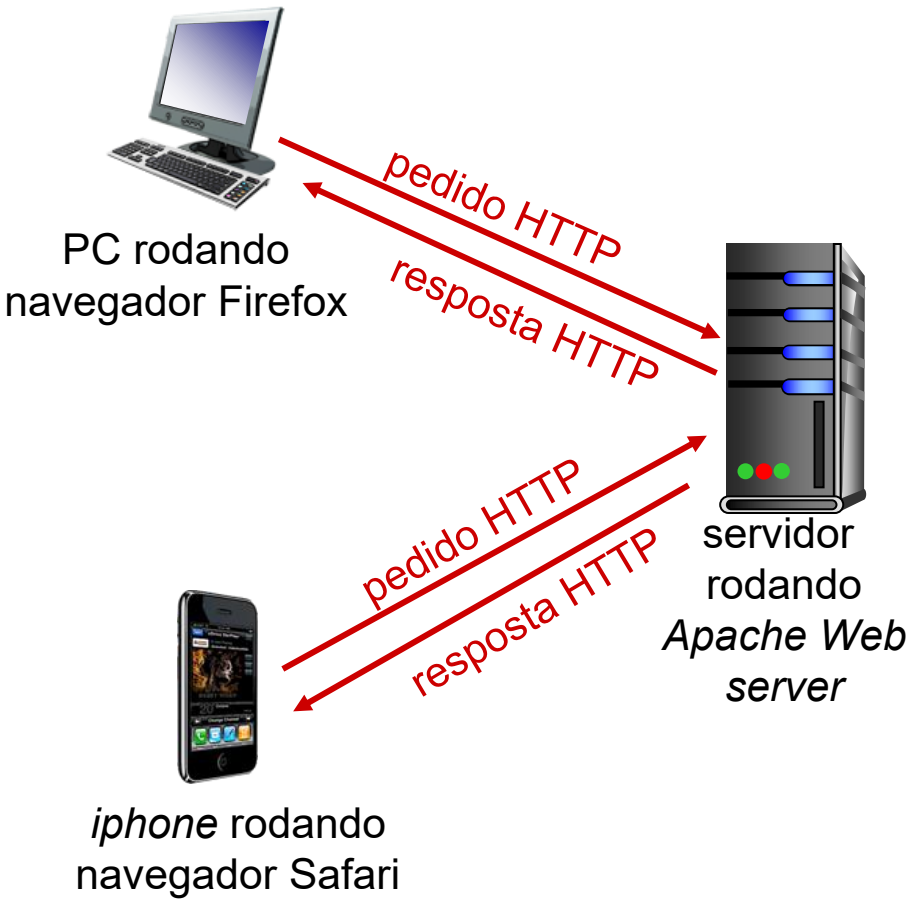
Visão geral do HTTP

HTTP: *HyperText Transfer Protocol*

- HTTP 1.0 (RFC 1945 - 1996)
- HTTP 1.1 (RFC 2068 –1997)
- HTTP/2 (RFC 7540 - 2015)
- HTTP/3 (RFC 9114 – junho/2022)

- Protocolo associado à aplicação *World Wide Web*

- Modelo cliente/servidor
  - *Cliente:* navegador que pede, recebe e apresenta objetos *Web* (Microsoft Edge, Firefox, Chrome)
  - *Servidor:* servidor *Web* envia objetos em resposta a requisições (Apache, Microsoft Internet Information Server)



Visão geral do HTTP 1.1 (continuação)

Usa TCP como protocolo da camada de transporte:

- Cliente inicia conexão TCP (cria *socket*) para o servidor, porta 80
- Servidor aceita conexão TCP do cliente
- Mensagens HTTP trocadas entre navegador (cliente HTTP) e servidor *Web* (servidor HTTP)
- Conexão TCP fechada

HTTP é "sem memória"

- Servidor não mantém informação sobre pedidos anteriores do cliente

nota

Protocolos que mantêm "memória" são complexos!

- História passada (estado) precisa ser mantido
- Se cliente/servidor cai, suas visões do "estado" podem ser inconsistentes e precisam ser reconciliadas

Tipos de Conexões HTTP 1.1

A. HTTP não persistente

- No máximo um objeto enviado sobre uma conexão TCP
  - conexão então fechada
- Fazer *download* de múltiplos objetos requer múltiplas conexões

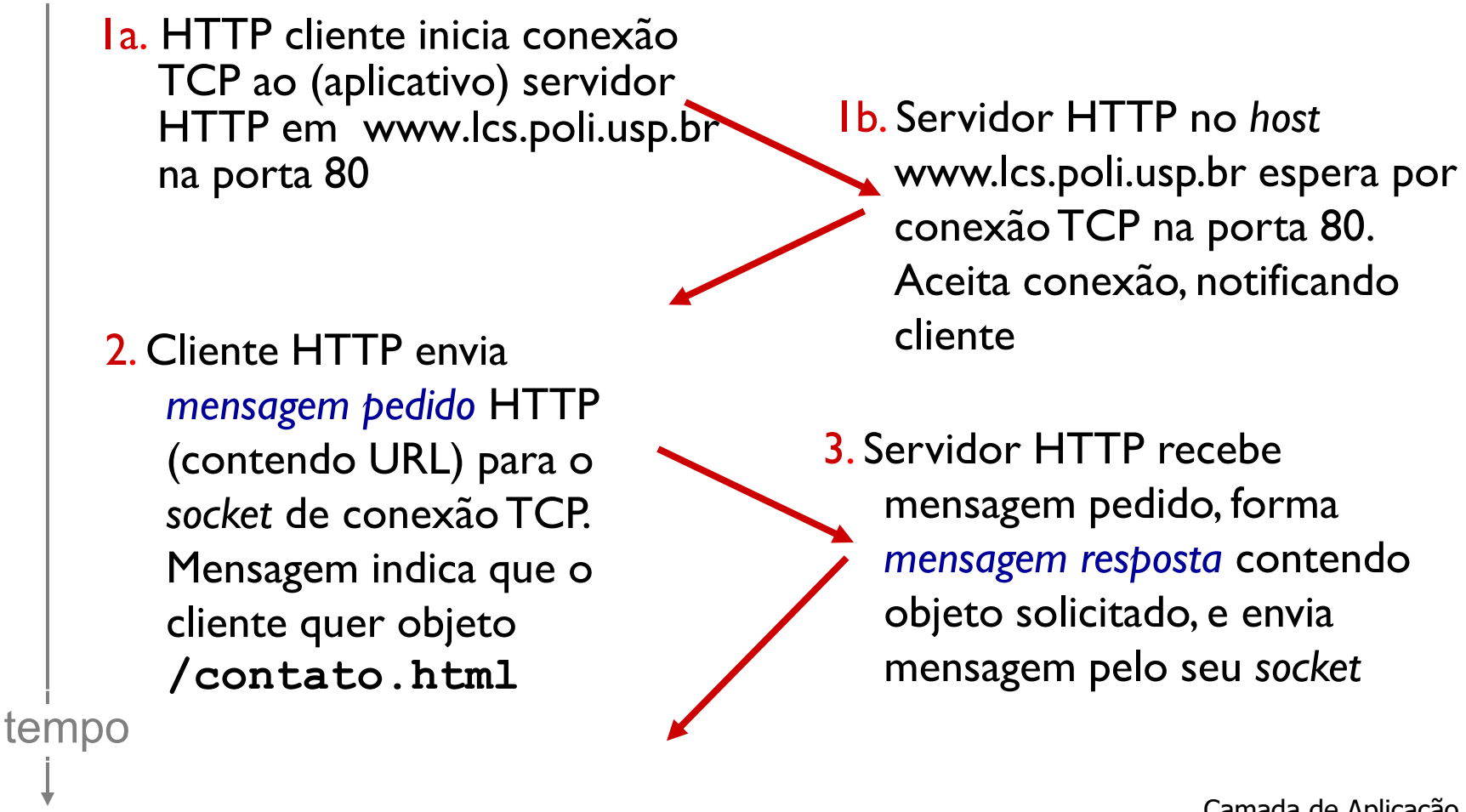
B. HTTP persistente

- Múltiplos objetos podem ser enviados sobre única conexão TCP entre cliente, servidor
- Padrão para HTTP/1.1

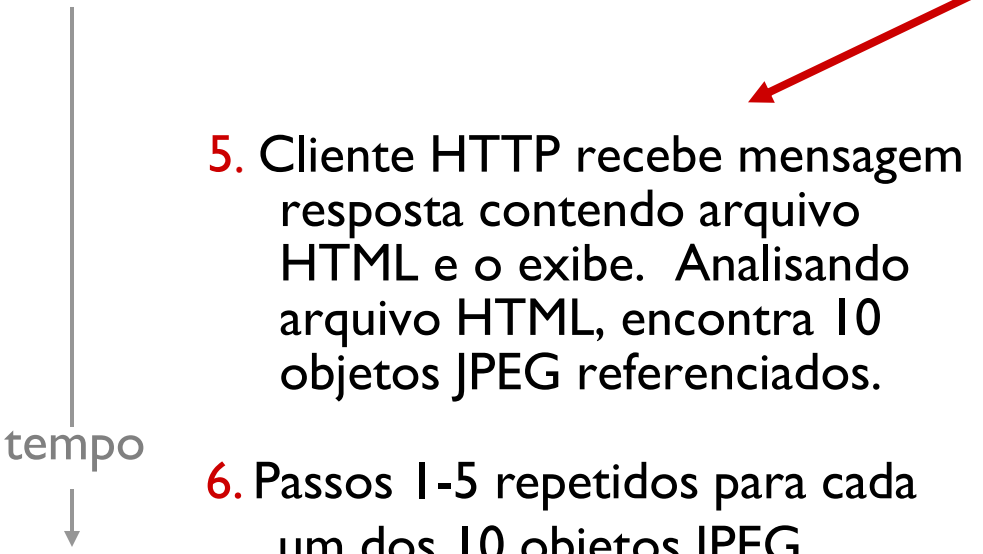
A. HTTP não persistente

Suponha que usuário digita URL:
http://www.lcs.poli.usp.br/contato.html

(contém texto e referências a 10 imagens jpeg)



A. HTTP não persistente (cont.)



4. Servidor HTTP fecha conexão TCP.

5. Cliente HTTP recebe mensagem resposta contendo arquivo HTML e o exibe. Analisando arquivo HTML, encontra 10 objetos JPEG referenciados.

6. Passos 1-5 repetidos para cada um dos 10 objetos JPEG

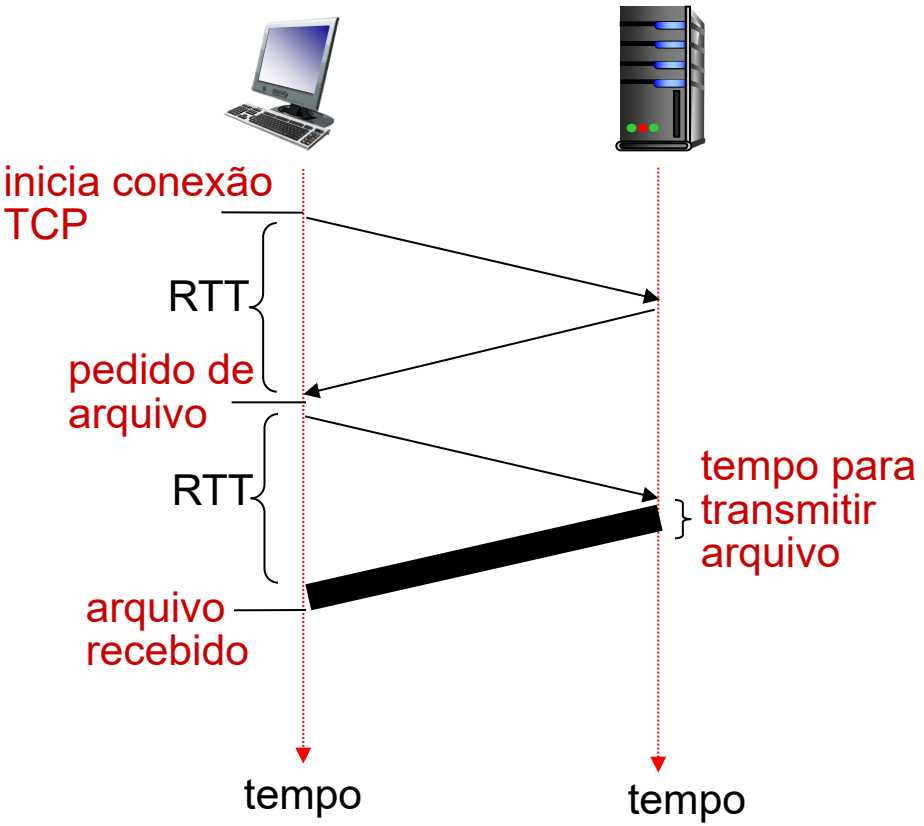
A. HTTP não persistente: tempo de resposta

RTT (*Round-Trip Time*) : tempo para pequeno pacote viajar do cliente ao servidor e voltar

tempo de resposta HTTP:

- 1 RTT para iniciar conexão TCP
- 1 RTT para pedido HTTP e primeiros bytes da resposta HTTP retornar
- Tempo de transmissão do arquivo
- Tempo de resposta para HTTP não persistente =

  2RTT+ tempo de transmissão do arquivo



B. HTTP Persistente

Problemas do HTTP não persistente :

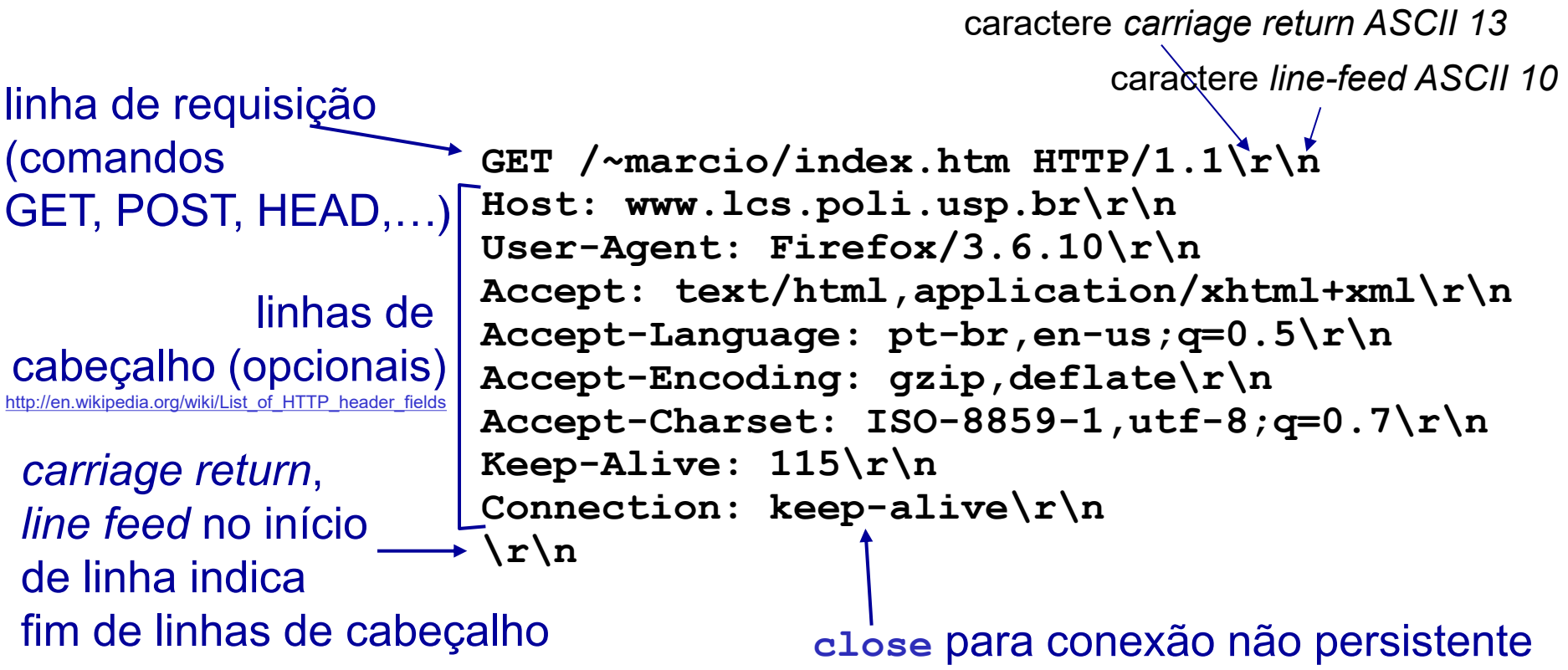
- Requer 2 RTTs por objeto, aumentando a latência do sistema

HTTP persistente:

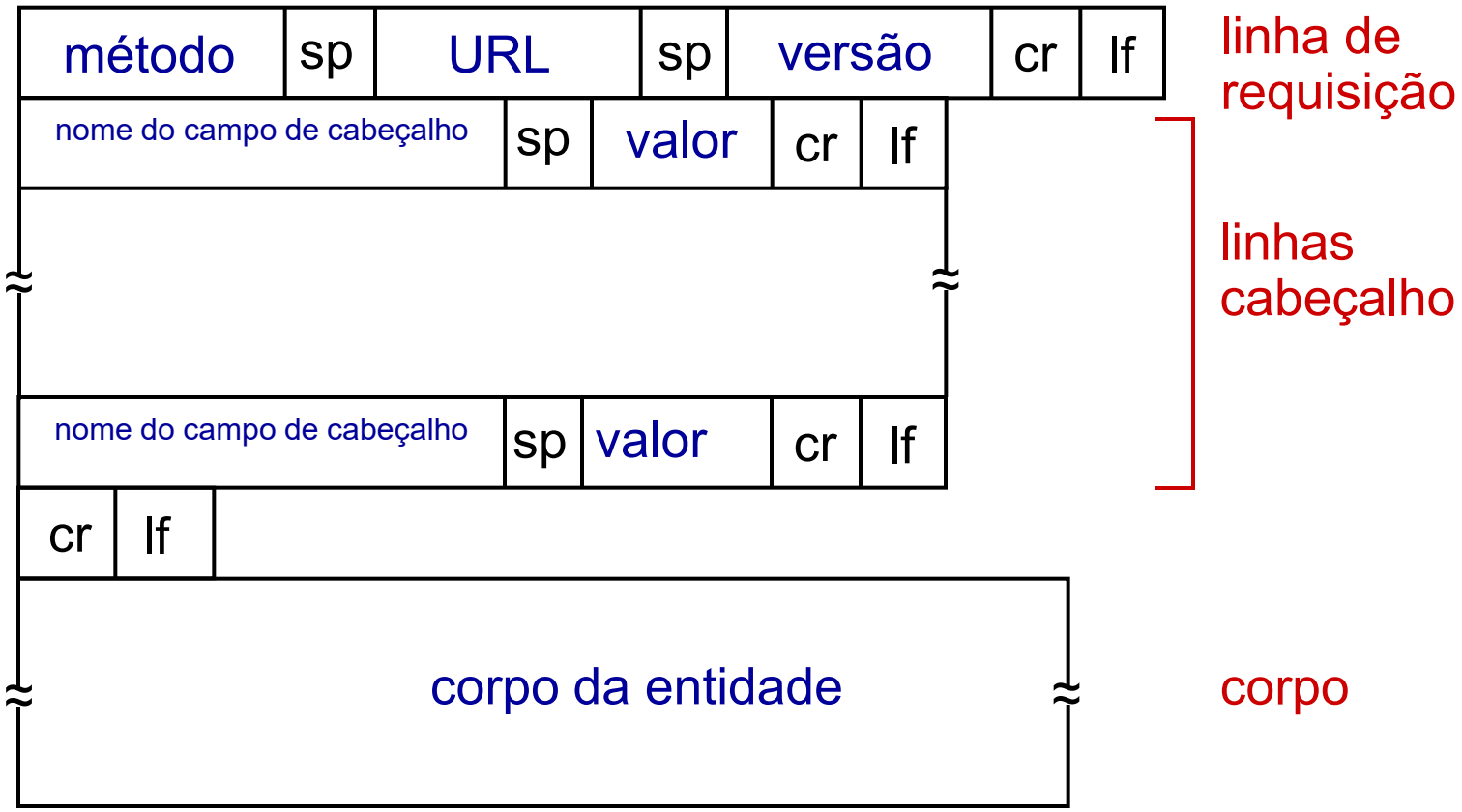
- Servidor deixa conexão aberta depois de enviar resposta
- Mensagens HTTP subsequentes entre mesmo cliente/servidor enviadas sobre a conexão aberta
- Cliente envia pedido assim que encontra objeto referenciado
- Perto de 1 RTT para todos os objetos referenciados

Mensagem pedido HTTP 1.1

- 2 tipos de mensagens HTTP: *pedido (request), resposta*
- Mensagem pedido HTTP:
  - ASCII (formato que permite leitura por humanos)



Mensagem pedido HTTP 1.1: formato geral



Obs.: sp=caracter de espaço; cr=carriage return; lf=line feed

Mensagem resposta HTTP 1.1

linha de estado (código e frase de estado do protocolo)

linhas de cabeçalho

dados, e.g., arquivo HTML requisitado

```
HTTP/1.1 200 OK\r\n
Date: Tue, 25 Feb 2014 18:24:20 GMT\r\n
Server: Apache/2.0.52 (CentOS)\r\n
Last-Modified: Tue, 18 Feb 2014 17:00:02
   GMT\r\n
ETag: "17dc6-a5c-bf716880"\r\n
Accept-Ranges: bytes\r\n
Content-Length: 2652\r\n
Keep-Alive: timeout=10, max=100\r\n
Connection: Keep-Alive\r\n
Content-Type: text/html; charset=ISO-8859-
   1\r\n
\r\n
data data data data data ...
```

Códigos de estado da resposta HTTP

- Código de estado aparece na 1a linha da mensagem resposta servidor-cliente
- Alguns códigos exemplos:

200 OK

- Atendido com sucesso, objeto pedido mais para frente na msg

301 Moved Permanently

- Objeto pedido foi movido, nova localização especificada mais a frente nessa msg (Location:)

400 Bad Request

- Mensagem pedido não entendida pelo servidor

404 Not Found

- Documento pedido não encontrado nesse servidor

505 HTTP Version Not Supported

Experimentando o HTTP 1.1 (lado cliente)

Usando o Wireshark

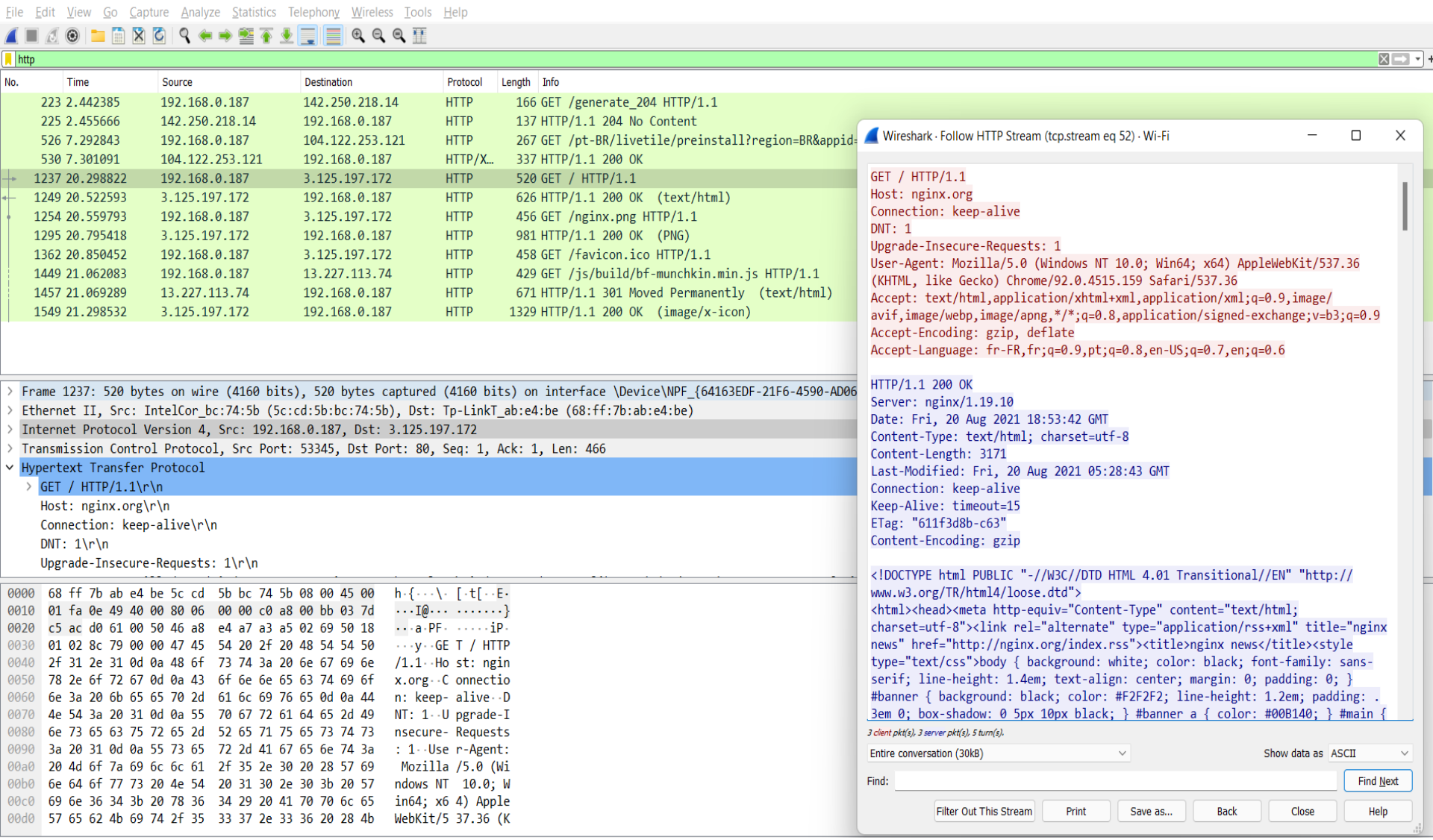
1. Abra o navegador

2. Abra o Wireshark e inicialize a varredura de pacotes no enlace usado para acesso à rede (e.g., WiFi)

3. Acesse pelo navegador o endereço http://nginx.org/

4. Siga (‘*follow*’ no Wireshark) a troca HTTP para este pedido

Experimentando o HTTP 1.1 (lado cliente)



Observações - HTTPS

- O HTTPS (*Hyper Text Transfer Protocol Secure* - protocolo de transferência de hipertexto seguro) é uma implementação do protocolo HTTP que tem se tornado o padrão na *web*.
- Possui uma camada adicional de segurança que utiliza o protocolo SSL/TLS.
- Essa camada adicional permite que os dados sejam transmitidos por meio de uma conexão criptografada e que se verifique a autenticidade do servidor e do cliente por meio de certificados digitais. Passou a ser o padrão a partir do HTTP/2.
- A porta usada para o protocolo HTTPS é a 443.

Observações - HTTP/3 e QUIC

- Diferentemente das versões anteriores, o HTTP/3 não é baseado no TCP mas sim no QUIC (RFC 9000), desenvolvido inicialmente pelo Google.
- Em 2025, cerca de 33% do tráfego HTTP já é realizado sobre o QUIC. A principal vantagem é a diminuição significativa do tempo de resposta.
- Nesta disciplina, tomamos por base a versão 1.1 do HTTP por questões didáticas. Dessa forma, fica mais fácil se concentrar nos princípios da camada de aplicação, deixando o problema da comunicação confiável para a camada de transporte que será vista a partir da próxima aula.
- Um vídeo inicial sobre as diferenças do HTTP/3 em relação às versões anteriores pode ser visto aqui.
- Mais detalhes sobre o HTTP/3 e o QUIC são deixados para cursos mais avançados.

(Kurose, p. 125) Considere o seguinte *string* de caracteres ASCII que foram capturados pelo *Wireshark* quando o navegador enviou uma mensagem HTTP GET. Os caracteres <*cr*><*lf*> são caracteres *carriage return* e *line-feed* . Responda as seguintes questões, indicando onde na mensagem HTTP GET abaixo você encontra a sua resposta.

```
GET /cs453/index.html HTTP/1.1<cr><lf>Host: gai
a.cs.umass.edu<cr><lf>User-Agent: Mozilla/5.0 (
Windows;U; Windows NT 5.1; en-US; rv:1.7.2) Gec
ko/20040804 Netscape/7.2 (ax) <cr><lf>Accept:ex
t/xml, application/xml, application/xhtml+xml, text
/html;q=0.9, text/plain;q=0.8,image/png,*/*;q=0.5
<cr><lf>Accept-Language: en-us,en;q=0.5<cr><lf>Accept-
Encoding: zip,deflate<cr><lf>Accept-Charset: ISO
-8859-1,utf-8;q=0.7,*;q=0.7<cr><lf>Keep-Alive: 300<cr>
<lf>Connection:keep-alive<cr><lf><cr><lf>
```

(a) Qual o URL do documento requisitado pelo navegador?

(b) Qual a versão de HTTP o navegador está rodando?

(c) O navegador requisitou uma conexão persistente ou não persistente?

(d) Qual é o endereço IP do *host* no qual o navegador está rodando?

(e) Que tipo de navegador iniciou a mensagem? Por que é necessário o tipo de navegador numa mensagem de pedido HTTP?