# HTTP

5 de janeiro de 2024 21:18

## HTTP

*Como funciona?*

**Utiliza o TCP:**

- O cliente inicia uma conexão TCP (cria um *socket*) com um servidor HTTP (porta 80).
- O servidor TCP aceita o pedido de conexão do cliente
- São trocadas mensagens HTTP (mensagens de protocolo de nível de aplicação) entre o browser (cliente HTTP) e o servidor Web (servidor HTTP)
- A ligação TCP é terminada

**O HTTP não tem estado**

- O servidor não mantém estado acerca dos pedidos anteriores dos clientes

**Os protocolos orientados ao estado são mais complexos!**

- O passado tem que ser armazenado
- Se o servidor/cliente falham a sua visão do estado pode ficar inconsistente e terá que ser sincronizada

- Lista de operações sobre um **RECURSO** (ex: livros) é definida aproveitando a semântica dos métodos do protocolo HTTP:

| Recurso | POST (Create) | GET (Read) | PUT (Update) | DELETE (Delete) |
|---|---|---|---|---|
| /livros | Cria um novo livro; Pedido: objeto "livro" no corpo do HTTP Request! | Lista todos os livros; Pedido: vazio; Resposta: listagem de livros; | Atualiza um conjunto de livros passados no corpo do pedido HTTP | Apaga todos os livros; Pedido: vazio; Resposta: sucesso ou insucesso; |
| /livros/01 | Normalmente não é usado! Erro! | Devolve o objeto que representa o livro com id 01 | Se existe livro 01 então atualiza-o; Senão dá erro! | Se existe livro 01 apaga-o; |

**CRUD (Create / Read / Update / Delete)**

HTTP não persistente
↳ 2 RTTs por objeto
⇒ reservar recursos para cada ligação TCP
⇒ ligações paralelas

HTTP persistente
→ dentro ligação aberta
- s/ Pipelining
  → envia novo pedido depois de receber anterior
- c/ Pipelining

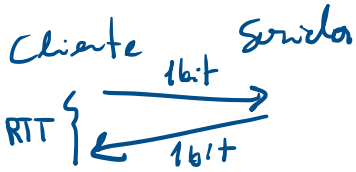
depois de receber anterior

- c/ Pipelining

→ mínimo 1 RTT por todos os objetos

Definição de RTT

tempo que 1 bit demora




2x tempo propagação
+ Nx tempo espera nos Queues
+ Nx tempo processamento

1 RTT → iniciar conexão
1 RTT → request message
? → transmissão do ficheiro

Exercício

Documento web
↳ 6 objetos
↳ Base HTML
→ 5 imagens

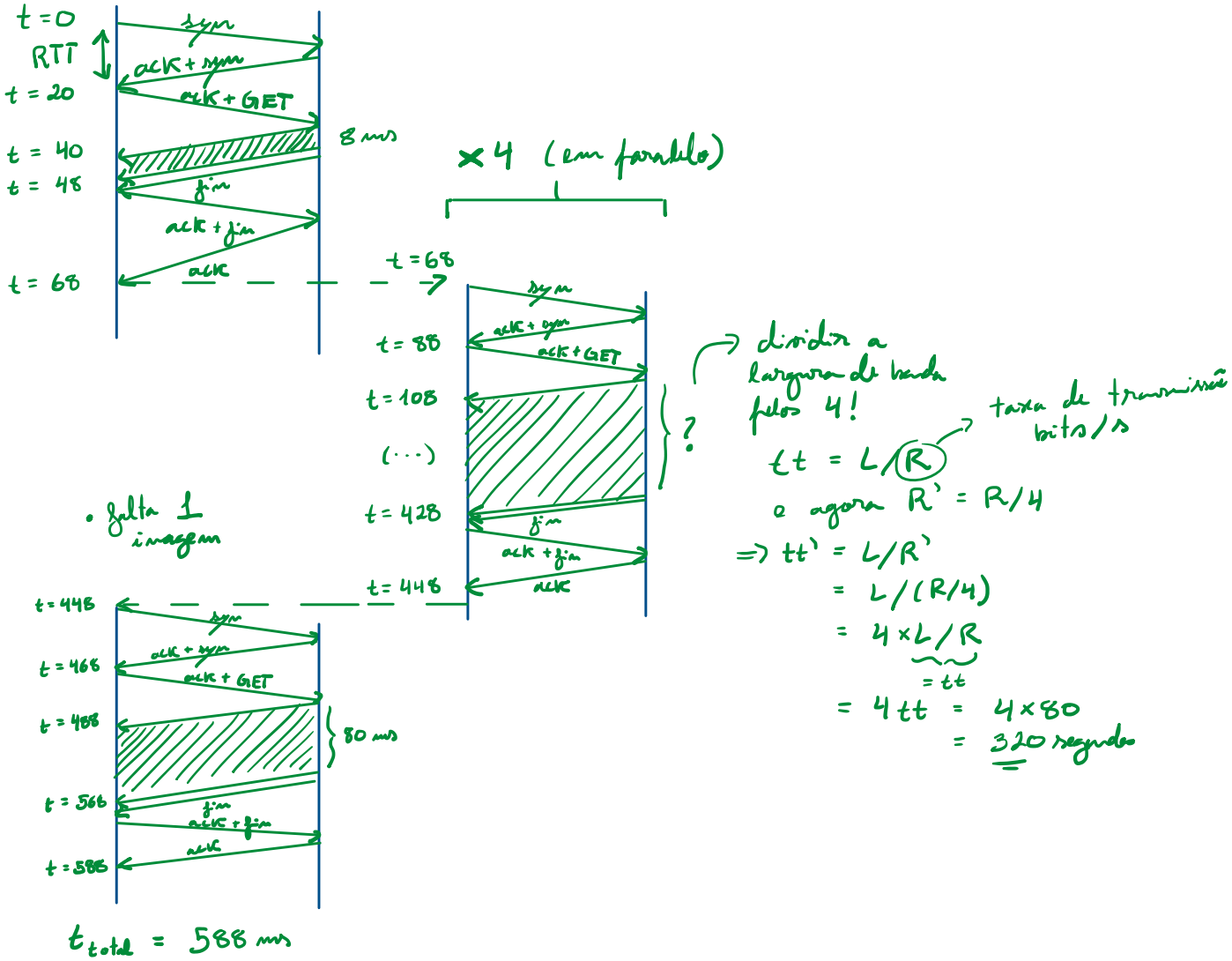
RTT = 20 ms
tempo transmissão = 8 ms
= 80 ms

- Admita que o browser só pode pedir as imagens quando receber completamente o objeto base (html)
- Admita que o utilizador sabe o endereço IP do servidor, indicando-o no browser
- Admita que a dimensão dos pacotes de estabelecimento de ligação, de confirmação, e de envio dos pedidos HTTP é desprezável
- Admita que os tempos de processamento dos pacotes também são desprezáveis

a) HTTP não persistente

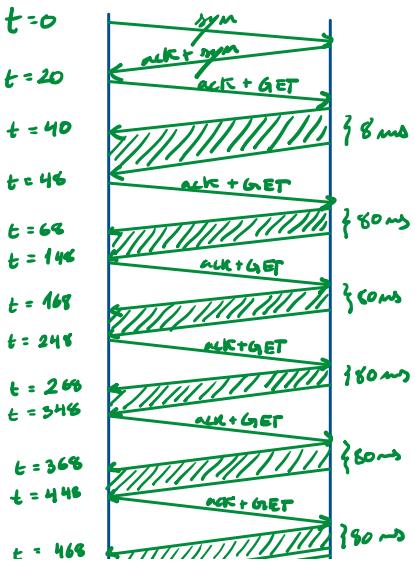
a) HTTP não persistente

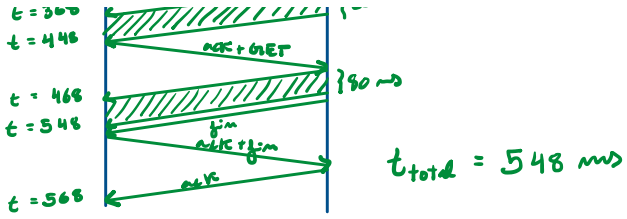
↳ 4 conexões em paralelo



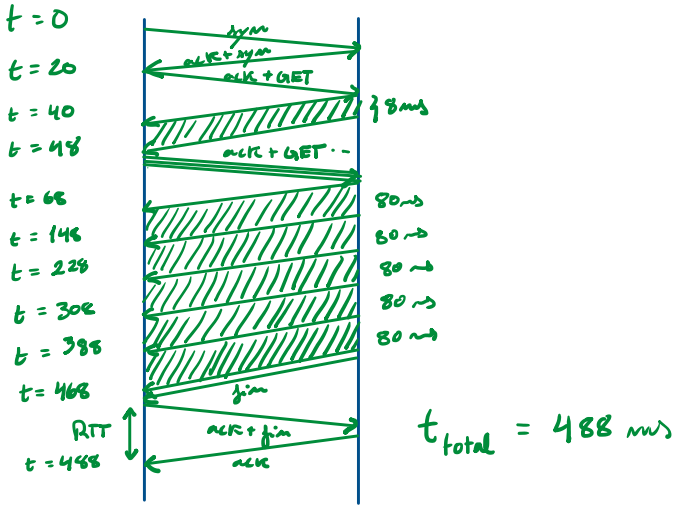

$t_{total} = 588$ ms

b) HTTP persistente, sem fifeline, sem conexões em paralelo.



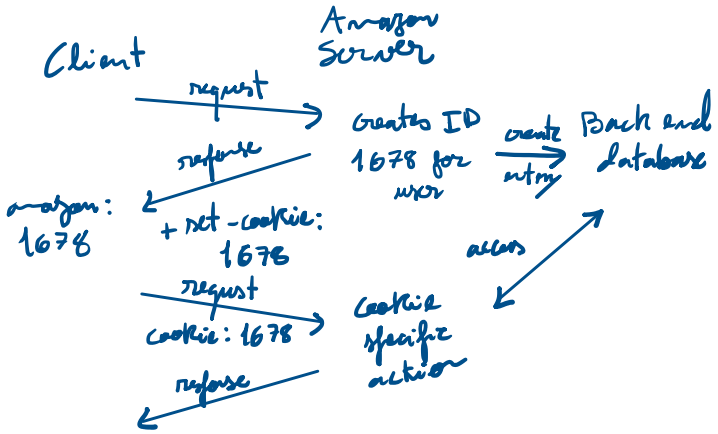



c) HTTP persistente, com pipeline, sem conexões em paralelo




COOKIES: informação de estado

1) no cabeçalho de HTTP Response
2) no cabeçalho de HTTP request
3) ficheiro com cookies na máquina do cliente
4) uma base de dados de suporte do lado do servidor Web




Cookies permitem:

- Autorização (cartões)
- Cabaz de compras
- Sugestões ao utilizador
- Informação de sessão

- Informação de sessão

→ Forma como as mensagens HTTP transportam a informação de estado

Web caches (servidor proxy)

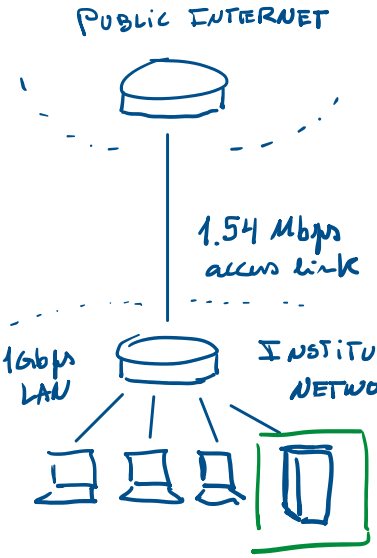
↳ O objeto da request pode estar na cache do servidor proxy:

↳ tem de atuar simultaneamente como cliente e como servidor

↳ instalados pelos ISPs ou instituições (universidades, empresas, etc.)

→ reduz o tempo de resposta

→ reduz o tráfego




Tamanho médio dos objetos : 100 000 bits

Taxa média pedidos : 15/sec

Tempo médio de atraso : 2sec

⇒ LAN : 15%

access link : 99%

$T_{delay}$ : Internet delay + access delay + LAN delay

2 sec + minutes + milliseconds

INSTALAR WEB PROXY

→ taxa de acerto : 0.4

⇒ 40% dos pedidos satisfeitos imediatamente

⇒ 60% redirecionados

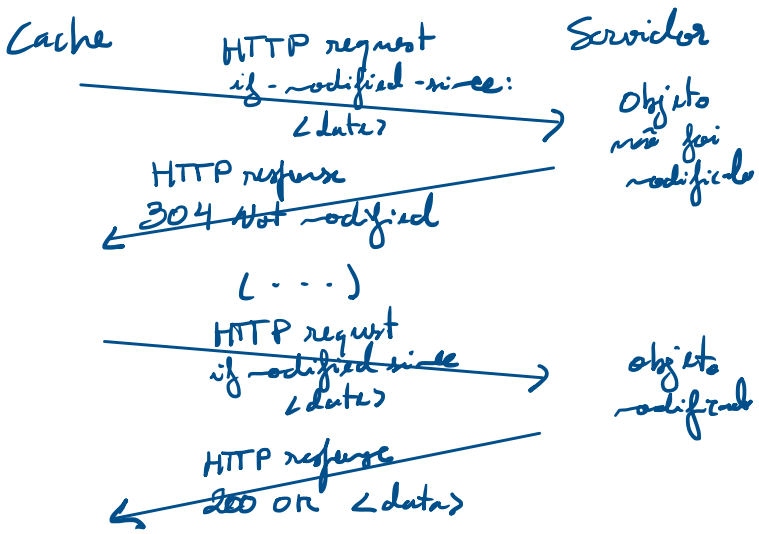
⇒ $T_{delay}$ = 0.6 × 2 sec + 0.4 × 10 msec < 1.4 sec (aceitável)

## GET Condicional

↳

- Não enviar objeto se a cópia mantida em cache está atualizada
- *cache*: inclui no cabeçalho do pedido HTTP a data da cópia guardada

na cache "if-modified-since: <date>"




## Exercício

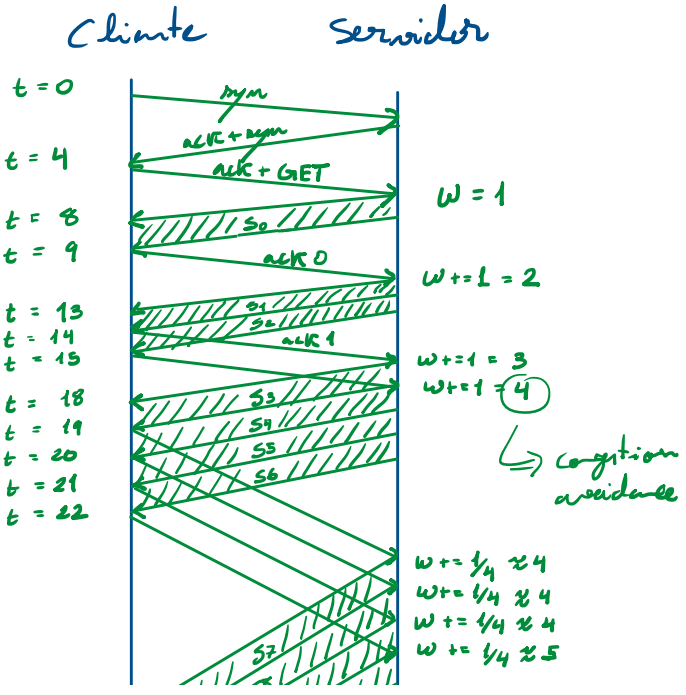
RTT = 4 ms
R = $1024 \times 10^3$ bits/s
Segmento : 128 bytes
= 1024 bits

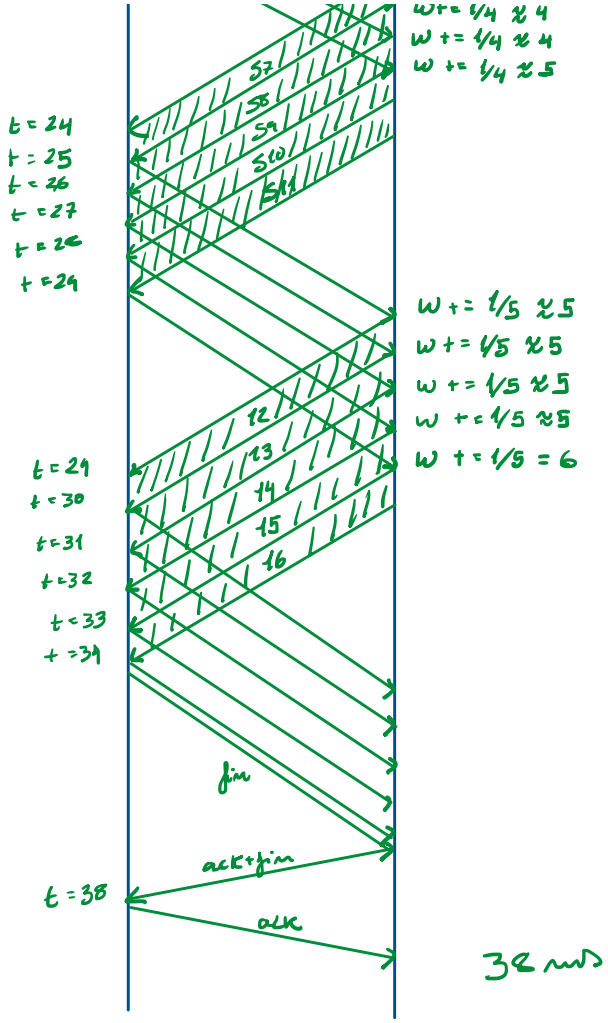
a) objeto base: 2048 bytes = 16384 bits
TCP "slow start"
window = 4 ⇒ "congestion avoidance"

2048/128 = 16 segmentos

1 seg ——— $1024 \times 10^3$ bits
t ——— 1024 bits

$\Rightarrow t = \frac{1 \times 1024}{1024 \times 10^3} = 10^{-3} = 1$ ms






b) Objeto base contém 4 imagens

1 imagem = 1024 bytes

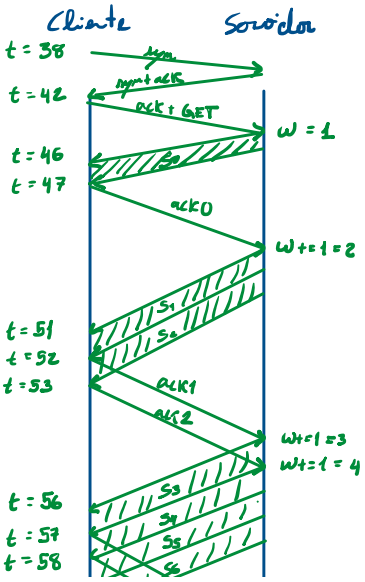
1024 / 128 = 8 segmentos

$tt = L/R$ ; $R' = R/2$

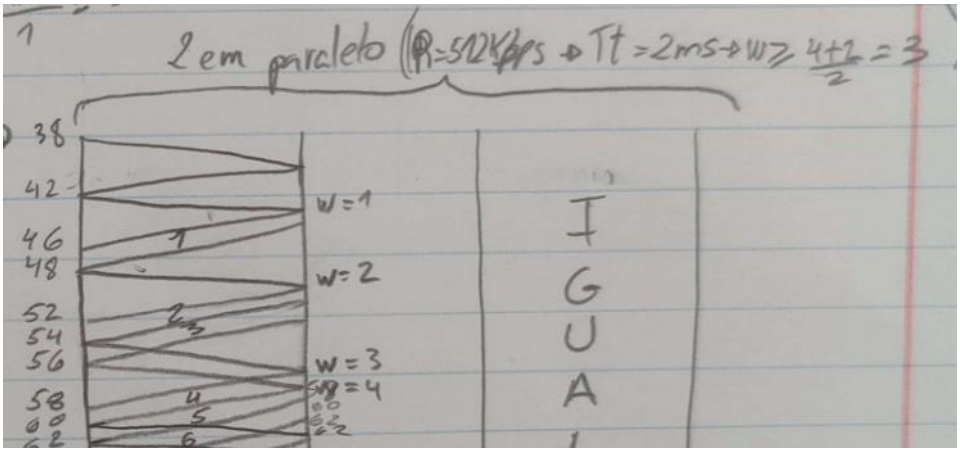
$tt' = L/(R/2)$

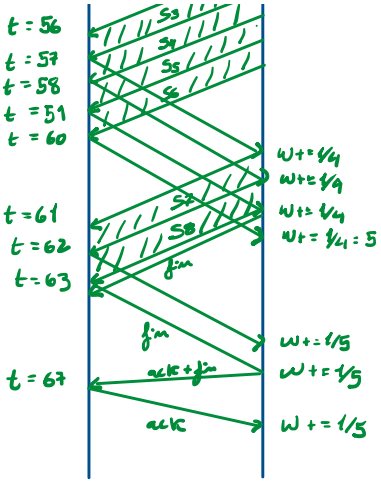
$= 2 \times (L/R) = 2 \times (tt)$

$= 2 \times 1 = 2$ ms



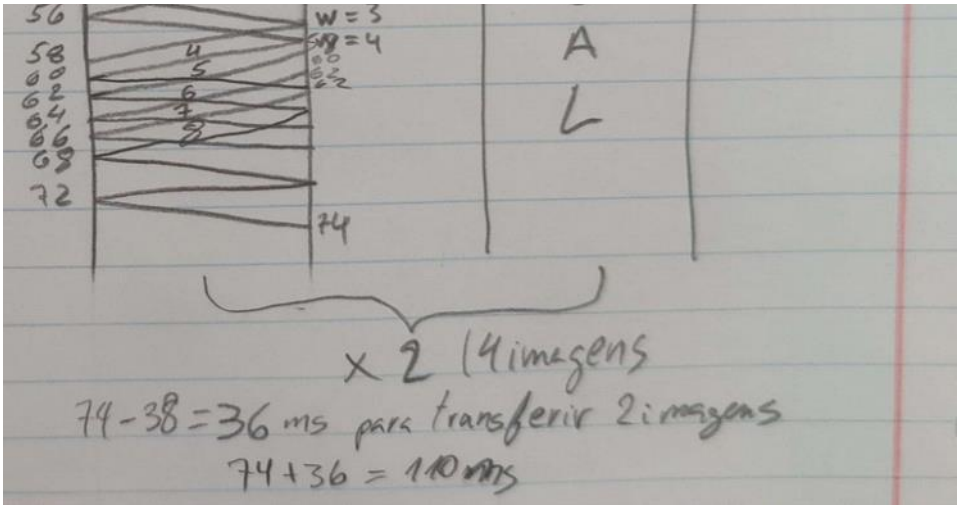

devia ter usado 2ms






67 - 38 = 29 ms

Total = 38 ms + 29 ms × 2
= 96 ms




**HTTP não persistente**

- Só pode ser enviado no máximo um objeto Web por cada conexão estabelecida
- O HTTP/1.0 utiliza HTTP não persistente

**HTTP persistente**

- Podem ser enviados múltiplos objetos Web por cada ligação estabelecida entre o cliente e o servidor.
- O HTTP/1.1 usa por defeito conexões persistentes