

Largura de banda e latência, problemas da conexão mobile, TCP

Largura de banda e latência (RTT)

Existem alguns conceitos sobre redes que afetam diretamente a performance da página. Vamos fazer uma pausa para entendermos um pouco sobre tais conceitos.

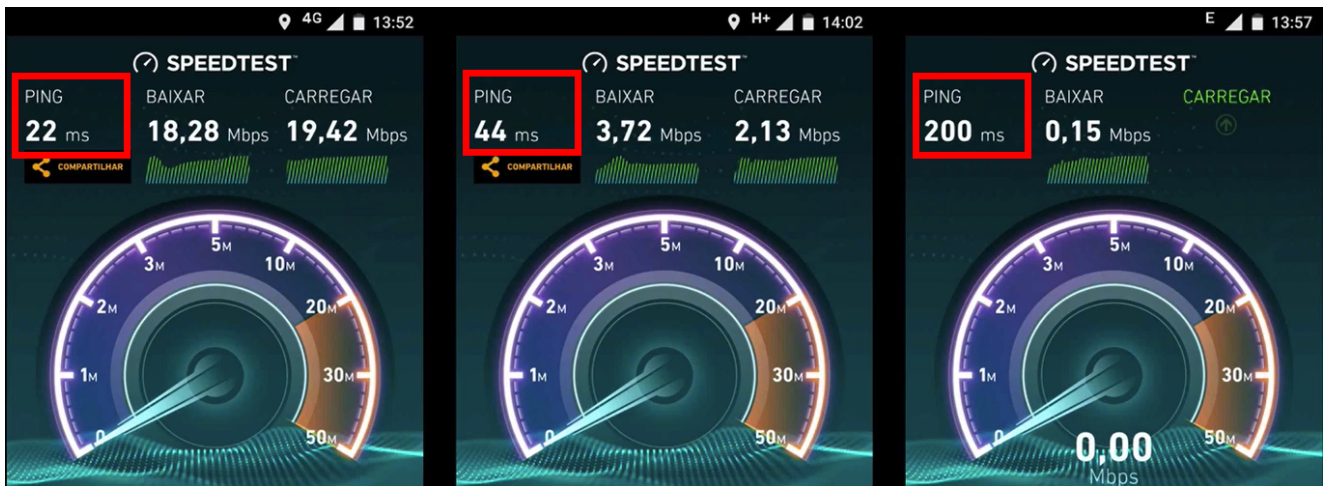
1) Largura de banda e latência (RTT):

Largura de banda é a quantidade de dados que passam simultaneamente. A latência está mais relacionada à distância que estamos do servidor final, o tempo que um *byte* leva para sair de nossa casa, chegar ao servidor final e voltar, é o que chamamos de *RTT - Round Trip Time*.

A latência afeta muito a performance, a partir do momento que um usuário abre um site e dá um "Enter", desse instante até visualizar a página ser renderizada acontece muita coisa! Para ver o primeiro *byte* precisamos que pelo menos 4 RTTs tenham acontecido. Observe essas etapas:

DNS lookup	TCP handshake	TLS handshake	HTTP request
1 RTT	1 RTT	1-2 RTT	> 1 RTT

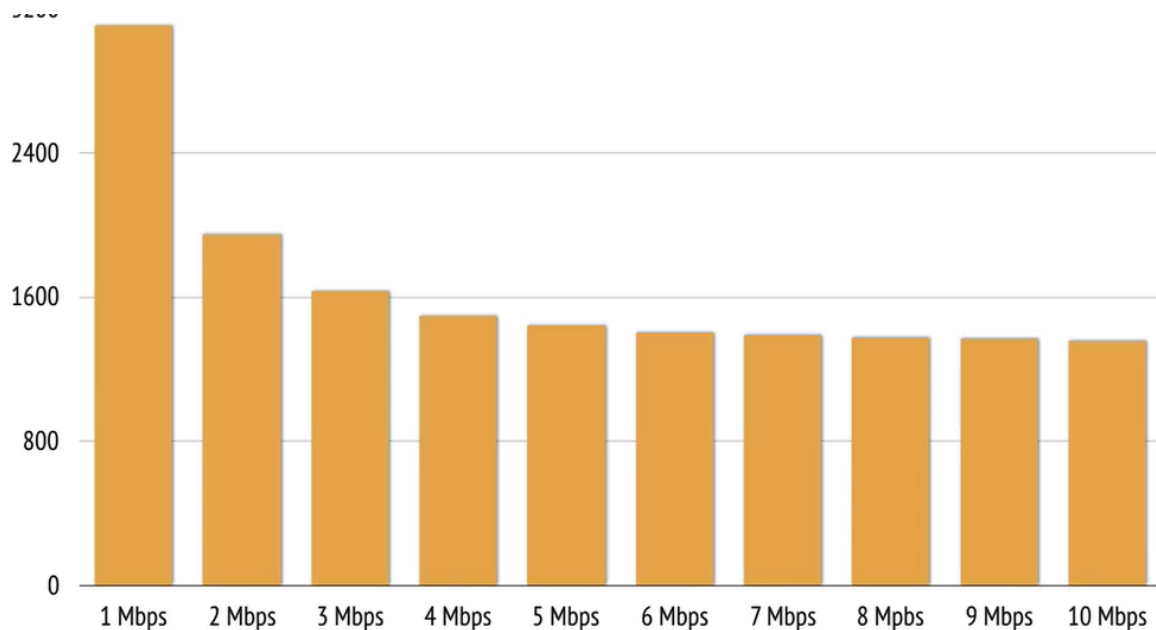
Observe que estamos pagando a latência pelo menos 4 vezes. Se usamos um servidor cuja latência é ruim isso tem efeitos no tempo em que o usuário fica sem visualizar nada. Vejamos uma comparação de como funciona o "RTT" em outros cenários, principalmente com *Mobile*. Faremos uma simulação de um acesso em 4G, H+ (que é uma espécie de 3,5G) e E (que é o 2G).



Usando o 2G que corresponde a imagem mais a direita, temos um *ping* de 200 ms e isso deve-se ao fato de estarmos em São Paulo acessando um servidor de nossa própria cidade. Usando um 3G temos um ping de 44 ms e, por fim, no 4G temos os 22 ms. Isso varia muito de servidor para servidor, momento, celular, localização geográfica e etc... Observe a diferença das medições em diferentes locais e períodos:

	WI-FI Mac	WI-FI celular	4G	3.5G	3G	3G frio	2G	2G frio
SP	12ms	20ms	22ms	44ms			200ms	
SP 2013		38ms			88ms	212ms	472ms	622ms
NY	139ms				300ms			

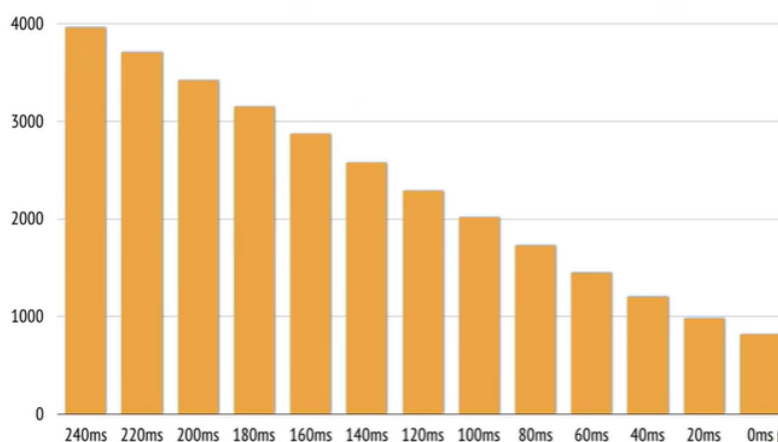
Falamos tudo isso para deixar claro que toda vez que conversamos com o servidor estamos pagando uma latência. Toda vez que mandamos algo e esperamos uma resposta, isso é RTT, é latência. Justamente por isso, conversamos sobre a possibilidade de reduzir o número de *requests*, para melhorar a performance. Ademais, temos que pensar nas diversas redes que existem, nos celulares que vão sofrer interferências e que nem todos os usuários possuem o melhor e mais rápido plano de internet. Se você acredita que seu foco é a rede de casa é importante se atentar pois o acesso via *Mobile* tem crescido cada vez mais e inclusive ultrapassando o *Desktop*. Para finalizarmos essa parte vamos observar um estudo que o *Google* realizou a despeito do "Tempo de carregamento da página vs Largura da Banda":



TEMPO DE CARREGAMENTO DA PÁGINA vs LARGURA DE BANDA

O gráfico mostra que o tempo de carregamento da página melhora quando aumentamos a largura de banda no começo do gráfico, mas depois de 5 Mbps para 10 Mbps não verificamos tanta diferença. Uma internet de 100 mega serve para fazer um download mais depressa, assistir um filme no **Netflix** sem problema, mas para abrir uma página isso não tem tanta diferença. Aumentar a banda não significa que acessaremos ou navegaremos na Internet mais depressa.

O que melhora a sensação de performance na hora do carregamento da página é a latência. Observe:



TEMPO DE CARREGAMENTO DA PÁGINA vs LATÊNCIA (RTT)



Através desse gráfico podemos perceber que latências menores diminuem o tempo de

carregamento. Como programadores a ideia é que busquemos maneiras de diminuir a latência para solucionarmos os gargalos.

Conexões: da teoria à prática

Quando falamos de acesso em dispositivos móveis temos um ponto importante que é a bateria e é importante considerarmos o **Mobile**, pois tem crescido o acesso da Internet através dele. Quando falamos de bateria o grande ponto é que a conexão, 3G, 4G é o componente que mais gasta bateria depois da tela. Observe os gastos em termos de bateria no uso de uma rede "Wi-fi" e uma "3G":



WI-FI - 30 ~ 200 mW

3G - 1000 ~ 3500 mW

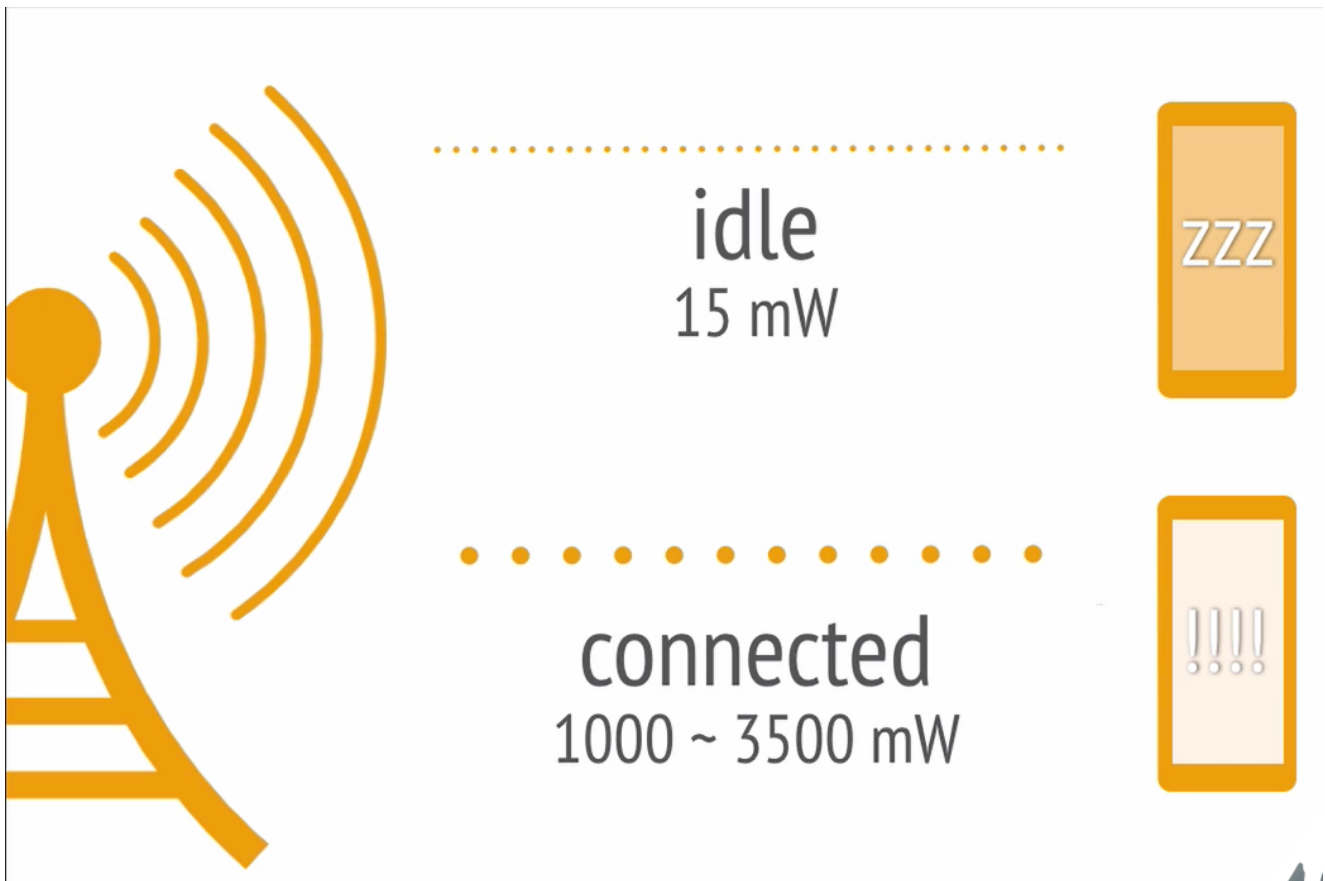
Entra em o "RRC" que é o "Radio Resource Controller" a ideia é que o celular não fique conectado 100% do tempo com a Internet e economize sua bateria. Quando for necessária se reconectar o celular reativa a conexão através do "RCC".

RRC Radio Resource Controller



Quando tiramos o celular do bolso e buscamos algo ocorre que o celular conversa com o RCC para pedir permissão para reabrir conexão com a internet. Dependendo do tipo de rede isso é feito de uma maneira distinta, nas redes 3G e 2G, em geral, a comunicação é feita com permissão da central da operadora e isso acrescenta uma latência. Nas redes 3,5 e 4G essa comunicação é feita diretamente com a antena mais próxima o que diminui a latência.

O estágio em que o celular fica dormente é conhecido como estágio *Idle* e nesse momento ele possui um gasto de bateria bem menor. Quando vamos conectar o celular é como se ele fosse acordado e nesse momento ele tem um gasto de bateria que varia de 1000 a 3500mW.



A ideia é que sempre que o celular faz essa passagem entre estados se faz uma transação

"RRC" e isso tardar alguns segundos de latência. Em redes mais antigas a latência é maior, em versões mais novas essa latência é menor.

Não vamos apenas em uma única direção, isto é, do *Idle* para o *connected*. Temos o contrário também, voltar do *connected* para o *idle*. Diante disso podemos pensar em um ponto a mais em nosso gráfico de "RTT":

RRC negociação	DNS lookup	TCP handshake	TLS handshake	HTTP request
200 ~ 2500ms	1 RTT	1 RTT	1-2 RTT	> 1 RTT

Temos que fazer uma negociação com o controlador da rede, que pode ser de poucos segundos até vários segundos, dependendo da rede. Temos sempre que pensar em um caso médio do usuário brasileiro e isso não é considerar que se pode partir do princípio da melhor rede. Temos as seguintes redes possíveis:

4G - LTE

3.5G - HSPA+

3G - UMTS, HSPA

2G - EDGE, GPRS

Na prática, mesmo tendo um celular que permite acesso 4G, nós não ficamos conectados nessa rede o tempo todo, ele faz *fallback* conforme a necessidade e a rede se modifica. Por exemplo, é bastante comum vermos o "+H" se transformando em "E". Na prática, cada uma dessas redes possui uma velocidade máxima:

4G - 300 Mbit/s

3.5G - 168 Mbit/s

3G - 14 Mbit/s

2G - 384 Kbit/s

Esse é o limite teórico, na prática ele é muito menor:

4G - 5 Mbps

3G - 1 Mbps

2G - 120 Kbps

Quando passamos de 5 Mbps não interessa mais tanto a velocidade, a não ser que você deseje assistir um **Netflix** no celular. A questão que mais importa desse valor em diante é a latência. Podemos visualizar na imagem abaixo as latências teóricas:

4G - <100ms

3.5G - <100ms

3G - 100 ~ 500 ms

2G - 300 ~ 1000 ms

Mesmo que metade do tempo do carregamento de nosso site seja culpa da operadora e das

empresas que oferecem esses serviços em geral, o usuário não sabe disso, quando ele acessa pelo celular seu site, para ele a culpa é do site. A ideia é que precisaremos levar em conta nas otimizações o tempo que perdemos devido ao limite da tecnologia.

Diminuir número de requests

Depois que fizemos essa discussão conceitual você deve estar se perguntando o que acontece na prática. Temos discutido diversas práticas e uma dela foi:

Diminuir o número de requests

E foi exatamente por isso que tocamos nesse ponto, cada *request* paga a latência de ir e voltar, ou seja, seu *request* e *response* e essa latência vai se acumulando e somando-se e no fim, temos um tempo de espera desnecessário.

Para diminuir o número de *requests* vimos diversas opções, concatenações, *sprits*, *inline* e etc. A ideia é que diminuir o número de *request* é essencial e para isso, podem ser utilizadas essas diversas práticas.

Temos que pensar também que a latência está diretamente relacionada a distância geográfica entre nós e nosso servidor. Para buscar diminuir essa latência a ideia é diminuir a distância. Quanto mais próxima é essa distância menor é o custo, em termos de latência:













Se possível é recomendável utilizar um servidor mais próximo geograficamente, pois na prática é melhor para o usuário final e também para a conversão. Isso, também, se você tiver uma audiência com localização geográfica definida.

Uma ideia para sites globais é trabalhar com "CDNs - content delivery network" e espalhar servidores pelo mundo e seu usuário pode baixar daquele que está mais próximo dele geograficamente. Essa ideia não é para todos, mas é interessante para quem tem um alcance maior. A ideia base do CDN é diminuir a latência.

Uma outra questão, é não gastar *requests* desnecessários, por exemplo, no caso dos *redirects*, observe o exemplo do site da *American Airlines* que possuía diversos *redirects*:

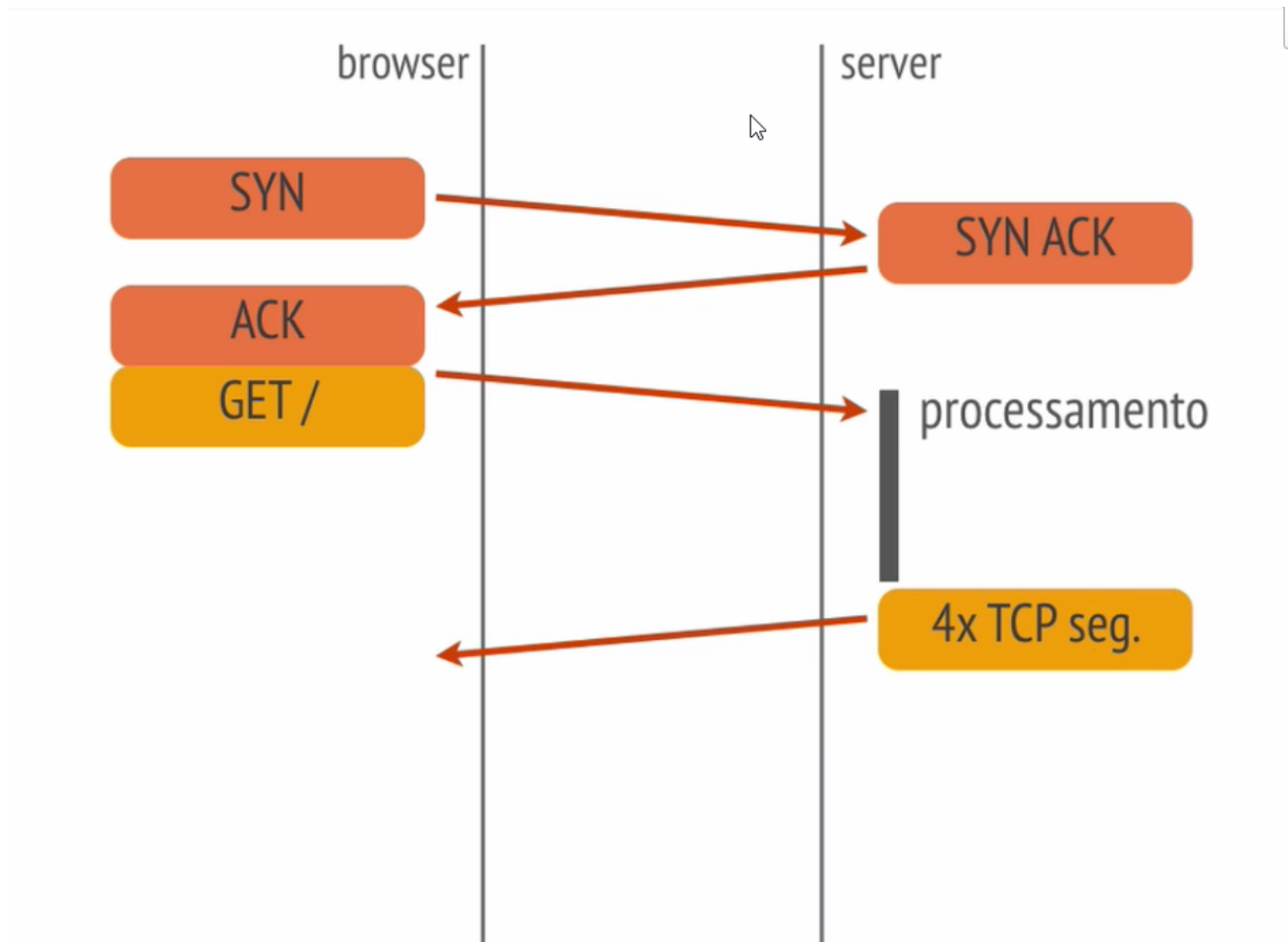
SEM REDIRECTS

Name Path	Method	Status Text	Type	Initiator	Size Content
 aa.com	GET	301 Moved Perm	text/html	Other	360 B 0 B
 www.aa.com	GET	301 Moved Perm	text/html	http://aa.com/ Redirect	527 B 0 B
 homePage.do www.aa.com	GET	301 Moved Perm	text/html	http://www.aa.com/ Redirect	23.6 KB 0 B
 mobile.aa.com	GET	302 Found	text/html	http://www.aa.co... Redirect	287 B 0 B
 www.aa.com /mt	GET	302 Found	text/html	http://mobile.aa.c... Redirect	523 B 0 B
 homePage.do?locale=en_US&pref=t... /mt/www.aa.com	GET	200 OK	text/html	http://mobile.aa.c... Redirect	0 B 59.0 KB
 loading.gif /mt/images	GET	200 OK	image/gif	homePage.do:3 Parser	0 B 1.4 KB
 selection.png	GET	200 OK	image/webp	homePage.do:3 Parser	0 B 1.4 KB

Essa escada de latência que os *redirects* produzem é extremamente desnecessária.

TCP

Vamos comentar sobre o protocolo TCP e mais algumas dicas para melhorar o seu projeto. O protocolo TCP funciona da seguinte maneira:



O *browser* manda um pacote SYNC para o servidor que manda um pacote de volta dizendo que recebeu o SYN ACK. Nesse momento o *browser* manda um pacote ACK e logo de cara começa o *request* do "http". Depois disso o GET é enviado para o servidor que recebe o *request*, processa e manda a resposta para o *browser*, nesse estágio da conversa ele é capaz de enviar 4 TCPs de uma única vez, isso é aproximadamente 5,8 KB. O cliente manda uma requisição GET e começa a mandar a resposta com esses quatro segmentos onde só cabem os 5,8 KB. Se tivermos uma página maior do que o segmento não será possível mandar ela. Por fim, quando o browser recebe esses 4 TCPs, o servidor enviará o dobro e assim por diante até a rede aguentar.

Esse é um cenário bastante simples e que não contém diversas outras considerações.

O começo é bastante demorado, até que a conexão realmente iniciar. O que podemos fazer para tentar resolver esse problema é diminuir o impacto do começo. Toda vez que vai e volta a seta é um RTT, então, até termos a resposta temos diversos RTTs. Queremos economizar nisso, a maioria dos servidores modernos trabalha com mecanismo que chamamos de *keep alive*, quando o servidor estabelece uma conexão com o navegador essa conexão será mantida, reaproveitada.

O próximo passo que podemos pensar é na etapa de processamento que trava o cliente. Algo que podemos fazer quando estivermos mexendo no *back end* é já ir mandando coisas para o cliente o mais rápido possível, principalmente se isso couber na primeira janela TCP. Isso chama-se *flush*.

Outro aspecto que queremos comentar é em relação a janela inicial do "TCP" que tem basicamente 4 segmentos e no qual temos que encaixar o máximo possível ou, se possível, o "html inteiro em 5,8 KB. Existe uma mobilização virtual para que a janela seja aumentada. Se reivindica que ocorra um aumento de pelo menos 10 segmentos. Existe um processo de aumento gradual, de *low start*, mas a primeira janela poderia ser maior. Se você possui um servidor moderno e pessoas que acompanham esse debate é bem provável que você já possua um servidor que contem 10 segmentos.

Tudo o que trabalhamos afetam sua performance.