Why your website should be under 14kB in size

Having a smaller website makes it load faster — that's not surprising.

What is surprising is that **a `14kB` page can load much faster than a `15kB` page** — maybe `612ms` faster — while the difference between a `15kB` and a `16kB` page is trivial.

This is because of the **TCP slow start** algorithm. **This article will cover what that is, how it works, and why you should care.** But first we'll quickly go over some of the basics.

What is TCP?

Transmission Control Protocol (TCP) is a way of using the **Internet Protocol (IP)** to send packets of data reliably — sometimes this is referred to as **TCP/IP**.

When a browser requests your website (or an image or a stylesheet) it makes that request using **HTTP**.

HTTP is built on top of TCP and a single HTTP request is usually made up of many TCP *packets*.

On its own IP is just a system for sending packets of data from one location on the internet to another. IP doesn't have a way of checking if a packet has successfully arrived at its destination.

When it comes to websites, knowing that all the data has arrived is important — otherwise we could end up with missing chunks of web page. There are other uses of the web where this doesn't matter so much — like streaming live video.

TCP is an extension of IP that allows a browser and your website's server to tell each other which packets have successfully arrived.

The server sends some packets, then waits for a response from the browser saying it has received the packets (this is called an acknowledgement or ACK), then it sends some more — or if it hasn't received an ACK it can send the packets again.

What is TCP slow start?

TCP slow start is an algorithm used by servers to find out how many packets it can send at a time.

When a browser first makes a connection to your server — the server has no way of knowing the amount of **bandwidth** between them.

Bandwidth is how much data can be transmitted over a network per unit of time. Usually it's measured in bits per second (`b/s`). Plumbing is a common analogy — think of bandwidth as how much water can come out of a pipe per second.

Your server doesn't know how much data the connection can handle — so it starts by sending you a small and safe amount of data — usually 10 TCP packets.

If those packets successfully reach your site's visitor, their computer sends back an acknowledgement (ACK) saying the packets have been received.

Your server then sends more data back, but this time it doubles the amount of packets.

This process is repeated until packets are lost and your server doesn't receive an ACK. (At which point the server continues to send packets but at a slower rate).

That's the gist of TCP slow start — in real life the algorithm varies, but that's essentially how it works.

So where does 14kB come from?

Most web servers TCP slow start algorithm starts by sending 10 TCP packets.

The maximum size of a TCP packet is `1500 bytes`.

This maximum is not set by the TCP specification, it comes from the ethernet standard

Each TCP packet uses **40 bytes** in its header — 16 bytes for IP and an additional 24 bytes for TCP

That leaves **1460 bytes** per TCP packet. `10 x 1460 = 14600 bytes` or roughly 14kB!

So if you can fit your website — or the critical parts of it — into 14kB, you can save visitors a lot of time — the time it takes for one round trip between them and your website's server.

How bad can one round trip be?

People are very impatient — and one round trip can be surprisingly long. How long depends on latency…

Latency is the time it takes a packet of data to travel from its source to it's destination. If bandwidth is how much water can go through a pipe per second — latency is the time it takes a droplet of water to enter the pipe and then exit the other end.

Here's a fun example of how bad latency can be:

Satellite internet

Satellite internet is provided by a satellite in orbit around the earth. It's used by people in very unpopulated areas, on oil rigs, cruise ships, and for inflight WiFi on airlines.

To illustrate this example of bad latency, let's imagine a bunch of oil rig bros have forgotten their dice at home and need to use the excellent (under 14kB) missingdice.com to play Dungeons & Dragons.

First one of them uses their phone to make a request for the web page…

The **phone** sends that request to the rig's **WiFi router** — which sends that data to the on-rig **satellite dish** — lets be nice and say that takes `1ms`.

The satellite dish then has to send that data to the **satellite** in orbit above the earth.

Typically, this is achieved using a satellite in geostationary orbit at `35786km` above the earths surface. The speed of light travels at `299792458 m/s`, so a message sent from earth to the satellite takes `120ms`. The satellite then sends the message back to a **ground station**, which takes another `120ms`.

Then the ground station has to send the request to wherever the **server** is on earth (light slows down to `200000000 m/s` when it's in a fibre optic cable). If the distance between the ground station and the server is the same as the distance between New York and London it'll take about `28ms` — but if it's more like the distance between New York and Sydney it'll take `80ms` — so we'll call it `60ms` (a convenient number for our math)

Then the request needs to be processed by the server which could take maybe `10ms` then the server sends it back again.

Back to the ground station, up into space, back down to the satellite dish, then to the wifi router, and back again to our oilers phone.

```
phone -> WiFi router -> satellite dish -> satellite -> ground sta
```

If we do the math that's `10 + ( 1 + 120 + 120 + 60 ) x 2 = 612ms`.

That's an extra `612ms` every round trip — perhaps that doesn't seem like a long time to wait but your website could easily have many round trips just to fetch it's first resource.

Also HTTPS requires two additional round trips before it can do the first one — which gets us up to `1836ms`!

What about latency for people who live on dry land?

Satellite internet might seem like a deliberately bad example — I chose it because it illustrates the point and is weird — but for landlubbers latency can get worse than that for lots of reasons:

- 2g mobile typically has latency between `300ms` and `1000ms`
- 3g networks can have anywhere between `100ms` and `500ms` latency

- Noisy mobile networks — say in an unusually crowded place like a music festival.
- Servers dealing with high amounts of traffic
- Bad stuff

Spotty connections can also cause packets to be lost — resulting in another round trip to get the lost packets.

Now that you know about the 14kB rule, what can you do?

Of course, you should make your website as small as possible — **you love your visitors and you want them to be happy**. Aiming for each page to fit in under 14kB is good target.

That 14kB includes compression — so it could actually be more like ~50kB of uncompressed data — which is generous. *Consider that the Apollo 11 guidance computers only had 72kB of memory.*

Once you lose the autoplaying videos, the popups, the cookies, the cookie consent banners, the social network buttons, the tracking scripts, javascript and css frameworks, and all the other junk nobody likes — you're probably there.

But, assuming you've tried your very best to fit everything into 14kB, and can't — the 14kB rule is still useful.

If you **make sure the first 14kB of data you send to your visitors can be used to render something useful** — for instance some critical CSS, JS and the first few paragraphs of text explaining how to use your app.

Note — The 14kB rule includes ***HTTP headers*** *— which are uncompressed (even with HTTP/2 on the first response) — it also includes* ***Images****, so load only what's above the fold and keep them very small, or use placeholders so your visitors know they're waiting for something good.*

Some caveats to the rule

The 14kB rule is more like a rule of thumb, than a fundamental law of computing:

- Some servers have increased the TCP slow start initial window to 30 packets instead of 10
- Sometimes the server knows it can start with a larger number of packets because it's used the TLS handshake to establish a larger window can be used.
- Servers can cache how many packets a route can manage, and send more next time it connects.
- There's other caveats too — here's a more in depth article about why the 14kB rule isn't always the case

HTTP/2 and the 14kB rule

There is an idea that the 14kB rule no longer holds true when using HTTP/2. I've read all I can about this without boring myself to death — but I haven't seen any evidence that servers using HTTP/2 have stopped using TCP slow start beginning with 10 packets.

HTTP/3 and QUIC

Similarly to HTTP/2, there is a notion that HTTP/3 and QUIC will do away with the 14kB rule — this is not true. QUIC recommends the same 14kB rule.

Further reading

Much of the content of this post comes from the following resources:

- High performance browser networking *by Ilya Grigorik*
- Increase HTTP Performance by Fitting In the Initial TCP Slow Start Window *by Simon Hørup Eskildsen*
- Critical Resources and the First 14 KB - A Review
- HTTP3 performance improvements

published	25 Aug 2022
modified	26 Aug 2022
author	Nathaniel
tags	posts post web performance HTTP TCP

Newsletter

Sign up to occasionally receive more of the same stuff that's on this website.

Email

Sign up