

What the Internet Actually Is

Reading time: ~10 min · Prereqs: Lesson 01

Why this matters to you

Every IT job, every modern product, every "the WiFi is broken" conversation touches the internet at some point. The thing is — almost nobody outside IT can describe what actually happens when you type a URL and press Enter. Five different machines you've never heard of are involved before the page paints. Once you know the basic flow, a huge slice of everyday tech problems — "this site is slow", "this site is down", "my browser shows a padlock warning", "VPN broke everything" — goes from "magic gone wrong" to "I can guess which layer is at fault". That diagnostic instinct is the entire payoff of this lesson.

1. Names vs numbers: DNS

Computers on the internet talk in **IP addresses** — numeric tags like `208.80.154.224` (IPv4) or `2620:0:861:ed1a::1` (IPv6). People, sensibly, prefer `wikipedia.org`. **DNS** — the Domain Name System — is the global phone book that translates names into numbers.

When you press Enter, your browser asks itself "have I looked up this name recently?" If yes (a cached answer), it skips ahead. If no, it asks the operating system, which asks a *recursive resolver* — usually your ISP, your router, or a public resolver like Cloudflare's `1.1.1.1` or Google's `8.8.8.8`. The resolver does the legwork on your behalf: it asks a **root server** "who knows about `.org`?", then asks that **top-level-domain server** "who knows about `wikipedia.org`?", then asks that **authoritative server** "what's the IP for `wikipedia.org`?" The answer comes back the way it came, and every step along the way caches it for a while — that's the **TTL** (time-to-live) on each record.

The recursive-resolver step is where a lot of real-world plumbing lives. Corporate filtering, Pi-hole ad-blocking on your home network, parental controls, and country-

level censorship typically intercept DNS here. If someone tells you "switch your DNS to 1.1.1.1", they're changing which resolver does the looking-up on your behalf.

2. The phone call: TCP + TLS

Now your browser has an IP. Next it needs to open a conversation. **TCP** — Transmission Control Protocol — is the reliable byte-stream protocol of the internet. Think of it as the dial tone: it doesn't care what you say, only that everything you send arrives in order, with nothing lost. Your browser opens a TCP connection to that IP on **port 443**, the universal port for HTTPS traffic.

Then **TLS** layers on top — that's the "S" in HTTPS, Transport Layer Security. The browser asks the server for its *certificate*, validates that it was signed by a **Certificate Authority** (CA) the browser trusts — brands like Let's Encrypt, DigiCert, Sectigo, GlobalSign — and then the two sides do a brief cryptographic handshake to derive a shared symmetric key. From that moment on, everything in the conversation is encrypted. The little padlock icon in the URL bar means two things at once: TLS succeeded, *and* the certificate's name matches the hostname you typed.

3. The conversation: HTTP

Inside the encrypted tunnel, the actual conversation is **HTTP** — Hypertext Transfer Protocol. It's a simple request/response pattern. Your browser sends something like:

```
GET / HTTP/2
Host: wikipedia.org
Accept: text/html
User-Agent: Mozilla/5.0 ...
```

The server replies with a status code, some headers, and the body:

```
HTTP/2 200 OK
Content-Type: text/html
Cache-Control: max-age=3600

<html>...
```

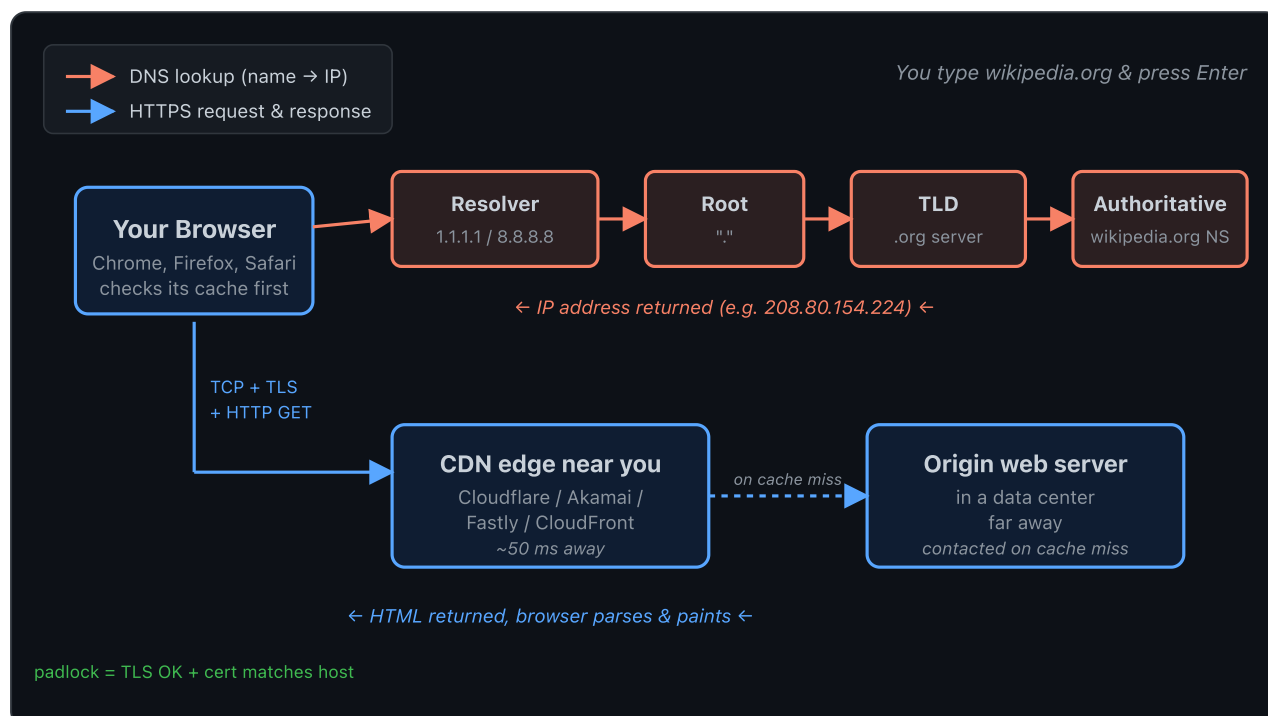
Methods you'll hear named: **GET** (read), **POST** (create / submit), **PUT** (replace), **DELETE** (remove). Status codes: **200** OK, **301** permanent redirect, **404** not found, **500** generic server error, **504** upstream timeout. HTTP/2 and HTTP/3 are newer wire-format upgrades (smaller, faster, multiplexed) but the request/response model is unchanged from the original 1990s design.

4. The "near you" trick: CDNs

A real website is hosted in maybe one or two data centers. Its readers, meanwhile, are spread across every continent. If every request for Wikipedia had to fly from Madrid to San Francisco and back, the web would feel like 1998. The fix is the **CDN** — Content Delivery Network. A CDN keeps cached copies of a site at hundreds of *edge locations* close to users. When you request Wikipedia from Madrid, you talk to a CDN node sitting in Madrid — about 50 ms away — instead of the origin server 9000 km west. The big names here are **Cloudflare**, **Akamai**, **Fastly**, **AWS CloudFront**, **Google Cloud CDN**, and the smaller-but-cheap **Bunny**.

5. The whole flow, drawn

Here's everything in one picture. Orange arrows = DNS lookup ("what's the IP?"). Blue arrows = the actual encrypted content fetch.



6. What happens when things break

Once you have the picture, common symptoms map almost mechanically to layers of the flow:

- **"One site won't load, others work."** Likely the DNS record for that site, or that site's CDN edge in your region. Not your internet.
- **"All sites are slow."** Local network or ISP. Your DNS lookups are slow, your TCP connections are slow, everything downstream is slow.
- **"Padlock warning."** The TLS certificate is expired, self-signed, or doesn't match the hostname. Could also be a real man-in-the-middle — rare but the warning is exactly the same.
- **"Page spins, then 504."** The server you're talking to is up — it answered with 504 — but *its* upstream dependency timed out. Common in micro-service architectures.

What people argue about

Three current debates inside the networking world:

- **HTTP/3 over QUIC vs HTTP/2 over TCP.** QUIC (the protocol under HTTP/3) is faster on lossy mobile networks and eliminates head-of-line blocking. It's also slower to roll out, harder for corporate middleboxes to inspect, and adoption is still patchy — many CDNs and clients support it, many enterprise proxies don't.
- **Centralisation of DNS resolvers.** Cloudflare's [1.1.1.1](#) and Google's [8.8.8.8](#) now answer a huge share of the world's DNS queries. Critics argue this is a privacy concentration risk and a single point of pressure for censorship orders. Supporters argue these resolvers are faster, more private (DNS-over-HTTPS), and more reliable than most ISP resolvers.
- **CDN consolidation.** When Cloudflare or AWS CloudFront has a bad day, a startling slice of the public internet has a bad day with them. The 2024 and 2025 outages re-opened the question of whether this dependence is fixable, or just the price of modern web performance.

Reading on

- **High Performance Browser Networking** — Ilya Grigorik. Free online at hpbn.co. The canonical readable deep-dive on everything in this lesson, plus the parts we skipped.
- **Cloudflare Learning Center** — learning.cloudflare.com. Short, well-illustrated explainers for every term in this lesson (DNS, TLS, CDN, BGP, DDoS, and more). Great when you want a second angle.
- **Julia Evans' *dns-zines*** — short, friendly, accurate comic-style explainers of how DNS actually works. Excellent companion to section 1 of this lesson.

LAST-REVIEWED: 2026-05-22