

THE IN A HURRY SERIES

The Internet *in a Hurry*

What actually happens when you click

Max Brocklesby

BOOKSINAHURRY.COM

Contents

The Whole Thing in One Page 4

Why You Should Care 5

The Core Ideas 7

How It Actually Works 21

What People Get Wrong 31

Use It 36

Terms 41

Go Deeper 44

Notes and Sources 45

Bibliography 54

The Whole Thing in One Page

The internet is usually drawn as a cloud, which is a useful confession. A cloud means something too complicated to draw and too distant to inspect. You select a link, a page appears, and the interval between the two is treated as electronic weather.

What follows is a chain of translations and limited promises. The browser reads a URL and separates its scheme, host, path and other instructions. Before it sends anything, it checks what already exists: the current document, a service worker, stored responses, security policy, remembered name records and connections that may still be open. A fast click often owes more to work done earlier than to speed after the finger moved.

If the request needs the network, the host name must lead to an address. The Domain Name System may answer from one of several caches or consult a distributed hierarchy of delegated authority. An address is still not a route. Networks announce which blocks of addresses they can reach, routers install forwarding state, and each router makes a local next-hop choice. A packet carries a destination. It does not carry an itinerary.

The first responding machine may not belong to the organisation named in the address bar. It may be a nearby content-delivery node, a reverse proxy, a load balancer or a cache. Before protected content moves, the endpoints may establish TCP and TLS, or use QUIC, which integrates TLS and supports several streams over UDP. IP itself attempts delivery. Ordering, loss recovery, flow control and congestion control are stronger services built at the endpoints.

HTTP then expresses the application request: give me this resource, with these conditions and credentials. A cache may answer. An edge service may forward the request. An origin may run code, check a session, query storage and call other services before constructing a response. The reply crosses radios, cables, switches and routers as packets. Loss is ordinary enough that transports are designed to detect and repair it.

The page has still not travelled as one page-shaped object. The browser receives HTML, builds a document tree, discovers style sheets, scripts, fonts, images and data, then sends more requests. It calculates layout, paints pixels and combines layers while scripts can alter the document and fetch again. One visible action can create dozens or hundreds of exchanges among organisations the address bar never names.

No one sequence covers every click. A cache can remove network work; an existing connection can remove setup; a browser may prefer IPv6 and HTTP/3 while another falls back; an enterprise proxy or mobile core may add machinery. The right model is conditional, but its distinctions hold.

This works because unlike networks can join through common protocols without surrendering their internal design. That choice produced extraordinary adaptability. It also produced centres at particular layers: recursive resolvers, transit networks, cable landings, exchanges, certificate authorities, cloud regions, identity systems and content networks. Decentralised does not mean evenly distributed.

The internet is therefore neither a cloud nor one pipe. It is prepared state and narrow agreements composing across machines and institutions that do not share one owner. The machinery disappears when each part keeps its promise. Understanding begins by asking which promise was required, which had already been prepared, and which failed.

That is the book.

Why You Should Care

At 22:30 on 29 October 1969, Charles Kline, a student programmer at UCLA, tried to send the word `login` to a computer at the Stanford Research Institute. The remote system failed after the first two letters. The first recorded host-to-host message across the new ARPANET was therefore `lo`, followed by a crash. About an hour later the full command worked.

That is a better origin story than a polished demonstration. The internet was

never made from perfect links. It was made by deciding where failure should be expected, who should notice it and how communication could continue when parts of the system did not behave. The modern network is faster, larger and less visible, but the same questions sit inside every page load.

You should care because the answers govern ordinary frustrations that otherwise look identical. A page can fail because your device cannot reach the local router, a resolver cannot find the name, a route has vanished or a connection is being blocked. It can also fail because the certificate is wrong, an edge cache is stale, the application has failed, or the browser has received data it cannot run. "The internet is down" collapses several systems into one sentence and therefore gives you almost no clue what to do next.

The model changes how you read speed. A broadband advert sells bandwidth, the capacity to move data per second. A click often feels slow because of latency, the delay before useful work begins, multiplied across name lookups, handshakes, redirects, server decisions and dependent requests. A wide motorway does not make a distant city nearby. Fibre can carry a large flow and still be bound by geography, queuing and the number of conversations required before the first useful pixel appears.

Then there is trust. The familiar browser padlock became a widely misunderstood security signal: protection of a connection looked like approval of a site. HTTPS can protect data against interception and alteration on the path and can authenticate the server according to rules about certificates and names. It does not certify that the seller is honest, the advice is sound, the download is harmless or the organisation deserves your money. The network can deliver a lie with excellent cryptography.

The physical world matters too. The internet seems weightless because its cables are buried, its exchanges occupy anonymous buildings and its data centres are designed to be noticed only when they fail. Yet routes follow commercial agreements, borders, cable systems, power supplies and rights of way. A service described as global may depend on a few facilities, providers or control systems. No global operator directs the entire internet,

but that does not make every participant equally powerful or every dependency replaceable.

Finally, this is one of the clearest cases in which standards shape daily life. A browser, home router, access provider, resolver, transit network, content-delivery company and origin service can be built by different organisations in different countries. They cooperate because public specifications define what messages mean and because operators choose to implement them. The agreement is partial, revised and frequently untidy. It is still enough for an unfamiliar application to communicate across independently operated networks without asking one central owner for permission.

Once you can follow the click, the cloud loses its mystery. You begin to see prior state, separable layers, local choices and precise claims. Failures become easier to diagnose. Security promises become narrower and more useful. Speed becomes an explicit budget rather than a mood. The internet becomes more impressive when it stops looking magical, because what replaces the magic is a working settlement among machines and institutions that were never placed under one command.

The Core Ideas

A Network of Networks

A packet can leave your laptop by radio and cross an ocean in glass. The networks carrying it need not use the same equipment, belong to the same company or agree about much beyond the handover. This is an internetwork: separate networks made to communicate. Your home Wi-Fi, a mobile operator and a data-centre network can retain their own designs. They need to agree on what crosses the boundary.

That distinction came from a practical problem. Early computer networks were built for different environments. A wired research network, a packet-radio network and a satellite network did not fail in the same ways or offer the same service. Forcing each one to adopt a common internal design

would have made expansion depend on redesigning every participant. The open-architecture answer was thinner. Let each network stand on its own. Join them with gateways, later called routers. Give packets a common addressing and forwarding format. Put stronger guarantees where the communicating endpoints can see the whole exchange.

Packets are the decisive object. A continuous message is divided into manageable units, each carrying control information and a portion of the data. Links can then be shared statistically. A quiet user occupies little capacity; a busy one sends more packets when capacity is available. If a route or link fails, later packets can be forwarded differently once routing changes. The network does not have to reserve one end-to-end circuit before communication begins.

At each physical hop, the IP packet is wrapped inside the local link's own frame. An Ethernet switch, a Wi-Fi access point and a mobile radio system solve different local delivery problems. The wrapper can change at the next router while the IP destination remains. This is how one packet crosses unlike media without carrying one universal link format.

This does not mean every packet wanders independently through random routers. Forwarding tables tend to produce stable paths for a time, and networks use traffic engineering to keep flows manageable. The point is that the architecture does not require a dedicated physical circuit or one central controller to establish the entire journey. Each router examines a destination and chooses a next hop according to its local table. The global path emerges from a sequence of local decisions.

IP, the Internet Protocol, is intentionally modest. It identifies source and destination addresses and defines how packets are carried across interconnected networks. Delivery is best effort. Packets can be lost, duplicated, delayed or received out of order. This sounds defective only if you assume every layer should solve every problem. A thin common layer is easier to implement across many kinds of network. Applications that require reliable, ordered delivery can obtain it from a transport such as TCP or QUIC. Applications with different needs can make a different bargain.

The layering is not a perfect stack of sealed boxes. Real equipment combines functions, providers inspect traffic, and design choices leak across boundaries. Yet the discipline remains powerful. A Wi-Fi link can change without rewriting HTTP. A browser can request a page without knowing which optical system carried the packets. A new transport can run over IP while routers continue forwarding the same kind of destination. Independent parts can evolve because their contracts are narrower than their implementations.

This is the first translation after the click. The human action becomes application data, then transport units, then IP packets, then link-specific frames or radio transmissions. At the far end the wrapping is removed in reverse. The apparent unity of the internet is therefore produced at the edges. Inside, it is a collection of networks passing labelled pieces to neighbours.

The Click Starts With What Already Exists

The clean classroom story starts at zero. You click, DNS runs, a connection opens, a request leaves and a response returns. A cold start can look like that. An ordinary click can begin in the middle of work already done.

The browser has state before the action. It knows the current page and how a relative link should be completed. It may hold a fresh HTTP response, a redirect, a security rule or data managed by a service worker. It may have cookies or other credentials that turn an anonymous request into one associated with an account. It may already have parsed code that intercepts the fetch and decides to answer locally, consult storage or use the network. The click is interpreted inside an existing application, not delivered to an empty machine.

Names also arrive with memory. A browser, operating system or recursive resolver may retain a DNS answer until its permitted lifetime ends. The resolver may remember the delegation even when it lacks the final address. A lookup described as a walk from the root is therefore a recovery path for missing knowledge, not a ritual performed for every page.

The transport can be warm too. A connection to the required origin may still be open and able to carry another request. A browser may possess session-resumption information that reduces later cryptographic setup. QUIC connection identifiers can allow some connections to survive a change in network path after validation. These mechanisms have conditions and security limits, but the common purpose is clear: preserve useful agreement rather than renegotiate it without cause.

The network itself depends on older decisions. Your device obtained addresses, a default route and resolver information when it joined the local network. Routers forward packets using tables installed before the packet arrived. BGP and other routing protocols belong to the control plane: they exchange information and help decide what forwarding state should exist. The arriving packet belongs to the data plane: it is matched against that prepared state and sent towards a next hop. No packet pauses at each router while the internet debates a route from scratch.

The far side is equally stateful. A content network may already hold a reusable copy near the user. A load balancer knows which service instances appear healthy. The application may recognise a session, find an object in memory or reuse a database connection. What looks like one fresh response can be the last act of a long-running system maintaining copies, routes, keys, processes and records.

State buys speed and creates disagreement. A cache can be fresh under its rules while the origin has changed. One resolver may have an older answer than another. Some routers may have accepted a new route while others are still converging. A connection may exist but no longer be usable. The system does not share one perfectly synchronised memory; it carries many bounded memories with different owners and expiry rules.

That is why repeating a click can change the result without the user changing the request. The first attempt may populate a cache, open a connection or reveal a failed route. The second inherits a different world. What happens after a click depends partly on what happened before it.

Names, Addresses, Routes and Services Are Different

You select `https://example.com/article`. The browser cannot send a packet to the word `example.com`, because routers do not forward prose. It needs to move through four different kinds of reference, each answering a different question.

The URL describes how software should interpret the resource reference. Its scheme is `https`; its host is `example.com`; its path is `/article`. The host name is a stable public handle, not a declaration of where one machine stands. The Domain Name System supplies records under delegated authority. A recursive resolver can answer from cache or ask through the hierarchy until it reaches an authoritative source for the relevant zone.

Calling DNS a phone book hides its design. A phone book is one list. DNS divides authority down a tree and permits answers to be copied into caches. The root does not store every site's address. It points towards top-level-domain authority, which points further down. The familiar thirteen root servers are thirteen named server identities, A through M, served by many machines in many locations. Anycast routing lets the same service address be announced from several places, so one name can describe a distributed service.

The DNS answer commonly supplies one or more IP addresses. An address is a label for packet delivery, not a set of turn-by-turn directions, proof of physical location or complete description of the service. Several names can share one address. One name can lead to several addresses. A content network can answer the same name differently according to resolver location, network conditions or service policy. Indirection lets infrastructure move while the public name remains.

The address still needs reachability. Routers work with prefixes, blocks of addresses sharing leading bits. Within one organisation, routing protocols help build paths across its own network. Between organisations, BGP exchanges reachability and path information among autonomous systems.

Operators then apply policy shaped by customer, peer and provider relationships, capacity, resilience and engineering choices. BGP does not promise the geographically shortest path or the smallest delay.

The same distributed trust creates risk. A mistaken or misapplied announcement can divert traffic, overload an unsuitable path or send packets into a black hole. The Resource Public Key Infrastructure lets operators check whether an autonomous system is authorised to originate a prefix, but origin validation does not prove that every intermediate path or policy is correct. It strengthens one claim inside the routing system rather than certifying the route as a whole.

Routing and forwarding are related but separate. The control plane constructs and updates the tables. The data plane performs the rapid local act: match the destination to the most specific applicable prefix, select a next hop and send. A packet from London towards Singapore does not carry a route chosen by one global planner. It crosses administrative boundaries because each network has installed enough compatible state to pass it onwards.

Finally, reaching an address does not identify one server. The address may terminate at an edge node, reverse proxy or load balancer that selects among many service instances. The visible service may then call databases, storage systems, identity providers and private APIs. “The server” is often a role played by several machines, while one machine can play roles for many names.

These distinctions explain both flexibility and failure. DNS can fail while the address and service remain healthy. A route can fail while the name remains correct. A service can fail after packets reach the right edge. Moving a name, changing an address and relocating data are different operations. The click works when human-readable identity, packet addressing, network reachability and application service line up for long enough to complete the exchange.

Reliability Is an Endpoint Bargain

IP will try to deliver a packet. It will not promise that the packet arrives, arrives once, arrives in order or arrives before it is useful. The communicating endpoints decide what stronger service they need.

TCP, the Transmission Control Protocol, provides the familiar reliable byte stream. The endpoints establish shared connection state. Bytes are numbered. The receiver acknowledges progress. The sender keeps unacknowledged data long enough to retransmit when loss is inferred. Data that arrives out of order can be buffered until the missing portion turns up. To the application, the result looks like one ordered stream even though the network underneath carried separate packets and may have dropped some.

TCP's stream abstraction also removes message boundaries. If an application sends two pieces, the receiver may read them together or in smaller parts; the application protocol must define its own framing. Reliability delivers ordered bytes, not an understanding of where one request ends.

Two controls solve different problems and are often confused. Flow control stops a fast sender overwhelming the receiving endpoint's available buffer. Congestion control tries to stop senders overwhelming the network. Common congestion-control algorithms start cautiously, learn from acknowledgements and loss or delay, and adjust how much data is in flight. There is no central traffic officer allocating every flow. Endpoints infer conditions from feedback and compete under algorithms intended to keep shared links usable.

Reliability has a price. Waiting for missing data is sensible when downloading a document: a missing sentence is not an acceptable speed improvement. In a live voice call, an audio packet that arrives late may be worthless. UDP, the User Datagram Protocol, offers a thinner datagram service with ports and a checksum but no built-in ordering, retransmission or connection state. Applications can tolerate loss, add their own recovery or build a richer transport above it. The absence of TCP's guarantees is not

carelessness when delay matters more than completeness.

QUIC shows how the layers can be rearranged without abandoning IP. It carries encrypted QUIC packets inside UDP datagrams, integrates the TLS handshake, supports reliable delivery and congestion control, and offers multiple ordered streams within one connection. Loss delays the streams whose data was in the missing packet, but unrelated streams need not wait for those missing bytes before making progress. Connection identifiers can also allow a client connection to survive some changes of network path, such as moving between access networks, subject to validation and implementation.

HTTP/3 maps web requests and responses onto QUIC. HTTP/2 commonly runs over a TLS-protected TCP connection and can multiplex several streams, but all its bytes still sit inside TCP's one ordered stream. A lost packet can therefore delay later TCP bytes even when they belong to another HTTP stream. QUIC moves the relevant ordering into its separate streams. This does not abolish congestion, distance, packet loss or shared implementation limits. It changes where ordered waiting is required.

Existing connections can be reused, and previous sessions can reduce later setup work. A click does not always pay for a fresh transport and cryptographic handshake. That is why two apparently identical page loads can have different timings, and why a browser's network panel separates connection setup from request and response time.

The central bargain is easy to miss because success hides the repair. You see a complete image, not the missing data that had to be sent again. The internet's reliability is therefore not evidence of a perfect network. It is evidence that the endpoints expect imperfection and keep enough state to recover from it.

Trust Is Added to the Path

Routing can carry packets to an address without proving that the endpoint deserves your secrets. DNS can provide an answer without telling you that the resulting service is honest. The early internet connected communities with more shared context than the public network has now. Confidentiality and authentication had to be added.

For an HTTPS connection, TLS, Transport Layer Security, establishes a protected channel between client and server. In its current 1.3 specification, the handshake negotiates cryptographic parameters, authenticates the server, and derives shared traffic keys. The resulting protection provides confidentiality and integrity: an observer on the path should not be able to read the application data, and an attacker should not be able to alter it without detection.

For a new certificate-authenticated connection, server authentication depends on names and delegated trust. The browser asks whether the certificate presented by the server is valid for the host it intended to reach, whether its signatures and dates are acceptable, and whether the chain leads to a certificate authority the browser or operating system trusts. The server must also prove possession of the corresponding private key; copying a public certificate is not enough. The authority does not need to know the moral character of the site. Under common domain-validation procedures, the relevant claim is much narrower: the certificate was issued under rules intended to establish control of the named domain or another permitted validation route.

This is why the padlock cannot mean safe. A fraudulent shop can control its own domain and obtain a valid certificate. A phishing site can protect the connection to the phishing site. TLS can deliver confidentiality, integrity and endpoint authentication while the application above it lies, manipulates or serves malicious code. Security claims improve when they are kept at the layer that can support them.

Encryption also has visible edges. The access network can still observe that

your device is communicating, how much data moves and when. IP headers must remain usable for forwarding. Depending on the protocol and deployment, some names or connection information may be exposed, protected or inferred. Encrypted DNS can hide queries from the local network while moving trust towards the chosen resolver. A virtual private network can hide destination details from the local access provider while giving the VPN operator a privileged view of the tunnel. These protections reduce exposure, but they also change whom you must trust.

TLS itself does not choose the route and does not stop every denial of service. It protects data between authenticated endpoints under its threat model. Compromised endpoints can read their own plaintext. A browser extension, stolen session token or vulnerable application can defeat the user's objective without breaking the cipher. Cryptography narrows the set of places where trust must be placed; it does not remove trust from the system.

QUIC integrates TLS rather than placing a separate TLS record stream above TCP. Much of QUIC's transport control information is protected as well as application data. This makes interference and passive inspection harder, which can improve security and make network management less comfortable for intermediaries accustomed to seeing transport details. Protocol design is also a negotiation over who is allowed to observe and modify what.

The cryptographic exchange makes a claim that no reassuring icon can enlarge. It is a machine-checkable claim with exact boundaries: the browser has established keys with an endpoint authenticated for the requested name under its trust rules, and subsequent data is protected on the channel. Everything beyond that sentence belongs to another form of evidence.

A Page Is Assembled, Not Delivered Whole

A headline appears, then a photograph fills the space beneath it. The page already looks like one object, although the browser is still building it. On the network it is a set of requests, responses, instructions and assets arriving from several places at different times.

HTTP supplies the application semantics. A client sends a method, target and header fields, sometimes with a body. A server returns a status, header fields and perhaps a representation. GET asks for a resource. A redirect tells the client to try another location. Cache controls describe when a stored response may be reused or must be checked. Cookies and other credentials can associate the request with prior state. None of this says how the bytes cross the network; HTTP can use different transports while preserving its request-and-response meaning.

The first responding system may be an edge server in a content-delivery network. A shared cache stores eligible responses so later equivalent requests can be answered with less delay and less upstream traffic. If a suitable response is fresh, the origin may not be contacted. If it is stale, the cache may validate it and learn that its copy is still current, or fetch a replacement. Personalised and authenticated material requires stricter rules because serving one user's response to another would be an impressive optimisation of the wrong thing.

A cache miss can travel through reverse proxies and load balancers towards an origin service. The origin may map the URL path to a file, run application code, check permissions, query databases, call other services and construct a response. “The server” is therefore often a role rather than one computer. One visible request can trigger many private requests inside a data centre, each with its own latency and failure modes.

When HTML arrives, the browser decodes bytes into characters, tokenises the markup and constructs a document tree. The HTML may refer to CSS, JavaScript, images, fonts, video and data. Those references produce further fetches, perhaps to other host names with separate DNS, connection and

trust work. CSS contributes style information; the browser combines it with document structure to calculate layout. It paints visual results and composites layers for display. JavaScript can alter the document, react to input and request more data long after the first view appears.

A modern browser also enforces local security boundaries. For ordinary web URLs, an origin is the combination of scheme, host and port, not the company name or the physical server. The same-origin policy and related controls limit when code from one origin can read data from another. The packets may arrive successfully and the browser may still refuse access, because network delivery and application permission answer different questions.

The sequence is partly parallel and partly constrained. A browser can discover and fetch several resources while parsing continues, reuse open connections and prioritise work likely to affect the display. Yet some scripts can block parsing, styles can delay rendering, fonts can change text layout and images can shift content if dimensions were not reserved. The network may have delivered the main HTML quickly while the visible page remains blank or unstable because the browser is waiting on dependencies or running code.

This explains why a page can seem loaded before it is finished. The first text, the main visual content and the point at which input responds are separate events. It also explains why a fast origin can coexist with a slow experience. Third-party scripts, distant APIs, oversized images and chains of redirects spend the latency budget elsewhere.

The click is therefore not a command to fetch a picture of a page. It starts a controlled construction project. The network delivers materials and instructions. The browser is the final machine in the production line, and what you see is its interpretation of a changing set of resources.

Decentralisation Creates New Centres

The internet has no global operations room. No switch can turn off every network, no company owns every cable, and no authority approves every new application. This decentralisation is real. So are the centres that grew inside it.

Start with physics. Radio links connect devices locally, but long-distance capacity is carried largely by fibre. Light moves through glass, equipment converts and amplifies signals, and cables follow routes that can be dug, leased, landed, repaired and powered. Networks meet in facilities where operators exchange traffic. Data centres place computation and storage near power, fibre and customers. The cloud is this equipment presented through remote services and software control.

Then economics rearranges the physical map. Carrying every request to one distant origin wastes time and capacity, so content networks copy eligible material to many edge locations. Large services build private backbones, connect directly to access providers and place caches inside or near other networks. Cloud platforms let organisations rent global infrastructure, which can lower the cost of deploying a service while concentrating many services on shared providers, regions and control systems.

The logical systems have centres too. Registries and resolvers occupy positions of leverage in DNS; transit networks can sit between many customers and the wider internet. Browsers rely on selected certificate authorities. One shared software component can spread a defect across independently operated systems. Separate ownership does not guarantee separate failure.

This is not a contradiction in the architecture. Open interfaces make substitution possible in principle, but scale rewards organisations that can place infrastructure widely, absorb attacks, negotiate interconnection and operate complex systems continuously. A protocol can be decentralised while the market using it becomes concentrated. A service can have

redundant servers while depending on one identity provider. A company can use several availability zones that share a regional failure or control plane.

Governance follows the same pattern. Technical standards are developed in bodies such as the Internet Engineering Task Force through published documents, working groups and implementation experience. Names and number resources are coordinated through institutions including IANA and regional registries. Governments regulate networks and services within jurisdictions. Private operators set terms, routing policies, platform rules and trust stores. There is no single governor, but there are many decisions with governing force.

Facebook's outage on 4 October 2021 made that distinction visible. According to its engineering account, a maintenance command disconnected its backbone. Its DNS systems then withdrew the routes advertising their availability because they could no longer reach the data centres. The name servers were still running, but other networks could not reach them. Cloudflare independently observed the route withdrawals and failed DNS lookups. Internal recovery tools also depended on the systems that had failed. This was not the whole internet going down. It was a large service discovering how many apparently separate parts shared one dependency.

Now the first idea returns. A thin common network layer let unlike systems join without surrendering their internal design. That openness allowed the internet to expand far beyond its founders' expected machines and applications. It also allowed successful intermediaries to become the places where naming, delivery, trust, computation and defence are purchased at scale. The internet did not acquire one centre. It acquired many centres at different layers.

That is the mature shape of the click. It passes through a decentralised system whose power is unevenly distributed. Understanding both halves prevents two opposite mistakes: imagining one omnipotent network owner, and assuming that the absence of one owner means the absence of chokepoints.

How It Actually Works

The click

Suppose you are reading a page in a browser and select a link to `https://example.com/article`. The pointer has moved a few millimetres; the browser now has to decide whether anything needs to leave the device. This is a hypothetical page at a domain reserved for examples, not a report about the service running there.

The click first becomes an event inside the browser. The browser identifies the element, resolves any relative address against the current page, and parses the resulting URL. `https` selects the scheme and its security expectations. `example.com` is the host. `/article` is the path supplied to the web service. A query could carry additional parameters. A fragment after `#` would usually identify a location within the returned document and would not be sent as part of the HTTP request.

Before reaching for the network, the browser checks what it already has. A service worker may intercept the fetch and answer from application-managed storage. The browser's HTTP cache may contain a reusable response. An open connection to the same origin may still be available. Security policy can upgrade, block or redirect the attempt. The first technical question is therefore whether a network request is needed at all. The clean textbook sequence of DNS, connection and request is a list of possible work, not a toll paid on every click.

Suppose this request needs the network. The browser's networking code uses operating-system facilities for sockets, interfaces and routes, and begins turning a name into reachable packets.

Out of the room

The device needs a local route before it can reach the public internet. It has usually acquired network configuration when it joined Wi-Fi, connected by Ethernet or attached to a mobile network. That configuration includes one or more IP addresses, a default route and the address of a DNS resolver, though privacy software or the browser may choose another resolver through an encrypted protocol.

On a home IPv4 network, the device often holds a private address that is not routed across the public internet. The home router translates many internal connections onto one or more public addresses, recording enough state to direct replies back to the right device. This network address translation helped stretch the limited IPv4 address space, but it also made the path stateful. Incoming traffic cannot be treated as though every private device were directly reachable.

IPv6 uses much larger addresses and reduces the pressure to share public addresses, though firewalls and policy still control reachability. Dual-stack devices may receive both IPv4 and IPv6 answers and try the usable option without making the user choose. The page is indifferent to the notation. The network path is not.

To send to the local gateway, the device also needs the link-layer destination used on that immediate network. It can discover which local interface or neighbour should receive the packet, wrap the IP packet in a Wi-Fi or Ethernet frame, and transmit it. Wi-Fi gets the packet across the room to an access point. It is no more the whole internet than a front door is the road system. From the router, the packet enters the access provider over fibre, copper, cable, fixed wireless or a mobile radio and backhaul system.

Each link has its own framing, error detection and capacity. The IP packet can be carried across several link technologies during one journey. Routers remove the old link wrapping and create new wrapping for the next link while preserving the end-to-end packet, apart from permitted header changes and

translation.

Finding the name

The browser needs an address for example .com. It asks through the resolver facilities available to it, but several caches may answer before any public hierarchy is contacted. The browser may remember a previous result. The operating system may remember it. The recursive resolver may have answered the same question for another user moments earlier.

Assume the resolver has no usable address record but knows enough to avoid starting from the root. It may already have cached the servers responsible for .com. It asks one of them which name servers are authoritative for example .com, then asks an authoritative server for the requested address record. Separate record types supply IPv4 and IPv6 addresses; the browser may request both. Answers can offer several addresses, and resolution may follow aliases. Each reusable answer arrives with a time limit set by DNS policy.

The resolver is called recursive because it accepts responsibility for obtaining a final answer on the client's behalf. The authoritative server has a different job: it publishes data for a delegated zone. Confusing the two makes failures hard to read. A public recursive resolver can be working while one domain's authoritative service is broken. An authoritative service can be healthy while a local resolver returns an old cached answer or cannot be reached.

The query itself may use ordinary DNS over UDP or TCP, or encrypted DNS over HTTPS or TLS. Encryption changes who can observe and alter the lookup. It does not change the logical task of finding authoritative data and applying cache rules.

The answer may point towards an address announced from many locations. Content networks and DNS providers often combine naming decisions with anycast routing, geography and load information. The “address of the website” can therefore be an invitation to reach one suitable edge, not a coordinate for the organisation's one server.

Finding a route

The device sends packets for the chosen address to its default gateway because the destination is outside the local prefix. The home or office router sends them towards the access provider. Inside that provider, routers use an interior routing system to move traffic towards an exit. At the boundary, BGP-learned information helps decide which neighbouring network offers the preferred route towards the destination prefix.

No packet asks BGP for directions as it passes. BGP and other routing processes belong to the control plane: they have already helped create the installed state. Forwarding belongs to the data plane: match the packet's destination against known prefixes, choose the most specific applicable route, select a next hop and send. The next router repeats the act under its own table.

The outward path may cross a direct private interconnection, a shared internet exchange or one or more transit providers. A large content network may have placed servers within the access provider itself, in which case the public path is short. Another service may be reached across an ocean. Physical distance adds delay that more bandwidth cannot erase, and commercial relationships can make the network path longer than a line on a map.

Routers also have finite queues. When packets arrive faster than an outgoing link can send them, they wait or are dropped. A short queue absorbs a burst. A bloated queue adds delay. A full queue discards packets and leaves the endpoints to recover. Congestion is therefore visible as timing and loss, even though no router sends the browser a plain-English note saying the link is busy.

The path can change between measurements or during a long-lived connection. Routing reconverges after failures and policy changes. Load balancers and anycast can direct separate connections to different sites. Yet it is misleading to picture every packet taking an independent scenic route. Providers often try to keep a flow on a stable path because reordering

creates work and unstable paths are difficult to operate.

Opening the conversation

Suppose the browser has learnt that the edge offers HTTP/3 and chooses to try it. It sends QUIC packets inside UDP datagrams to the server's address and port. The port is a number that helps the receiving machine direct traffic to the appropriate application service. The initial exchange negotiates QUIC and cryptographic parameters, validates enough of the path to limit abuse, and performs the TLS handshake integrated into QUIC. A returning client may use remembered state to reduce setup, but early data carries replay limits and is not a free shortcut for every request.

In a fresh certificate-authenticated handshake, the server presents its certificate information. The browser checks the requested host name, validity conditions, signatures and a chain to a trusted authority under its policy. A resumed connection can instead use secrets established earlier. The endpoints derive traffic keys. Once protection is established, application data and much of QUIC's own control information are encrypted and authenticated.

If HTTP/3 is unavailable or blocked, the browser may use HTTP/2 or HTTP/1.1 over TCP protected by TLS for an HTTPS URL. TCP establishes its own connection state, TLS then establishes the secure channel, and HTTP runs above it. Protocol negotiation lets client and server agree on an application version they both support.

Round trips matter here. Every exchange that requires one side to wait for the other's reply spends at least the current path delay. Modern protocols reduce setup work, combine negotiations and reuse connections because computation close to the user cannot cancel the time required for a signal to travel. A connection that already exists can be more valuable than a marginally faster processor at the far end.

Once data flows, acknowledgements tell the sender what has arrived. Loss detection prompts recovery of missing data. Flow-control limits protect the receiver. Congestion control adjusts the amount in flight according to

feedback. The result is a conversation continuously estimating how hard it can push a path it does not own.

Asking for the resource

The browser sends an HTTP request for `/article`, identifying the authority and attaching header fields that describe acceptable formats, cache conditions, language preferences, credentials and other context as applicable. It does not send every fact the browser knows, and browser privacy controls can restrict what cross-site requests may carry. Even so, the request can reveal more than the path alone: cookies, referrer information and client hints can link the action to a session or device context.

An edge server receives and interprets the request. If its cache contains a response that matches the cache key and remains fresh under the response's instructions, it can reply without contacting the origin. That may make a distant organisation feel local. The user is communicating with a nearby copy authorised by the service's delivery design.

If the stored response needs checking, the edge can send a conditional request using a validator such as an entity tag. The origin can reply that the representation has not changed, avoiding transfer of the full body, or send a new version. If the material is private or personalised, the cache rules may prevent shared reuse. Caching is governed by request method, status, headers, freshness and validation, not by a general sense that a page looks static.

A miss is forwarded. The edge may terminate the public connection and open or reuse another protected connection towards the origin. It can add information about the original request under controlled rules, apply security filters, compress responses and balance traffic among back-end systems. To the browser it is the server. To the application it is an intermediary.

Behind the address

The origin side can be less like a machine and more like a small bureaucracy. A load balancer chooses an available service instance. Application code identifies the route, checks authentication and decides what data is required. A cache inside the service may answer. Otherwise the application queries a database, calls an object store, contacts a payment or identity service, or sends requests to other internal systems.

Well-designed internal calls have timeouts because waiting forever is not a recovery strategy. Systems retry some failures, but careless retries can multiply load precisely when a dependency is struggling. A request that changes state, such as placing an order, must be designed so that uncertainty about a lost reply does not create duplicate action. Distributed computing turns “did it work?” into a technical question when the request may have succeeded and only the application reply was lost.

The service constructs an HTTP response with a status, metadata and a body. It may choose compressed content based on the request, attach cache instructions, set a cookie, and include security policies for the browser. The edge can store an eligible copy, then send the response over the existing public connection.

The body is divided among transport frames or segments and then IP packets. Those packets can share links with video calls, backups and thousands of other page loads. They may arrive in a different grouping from the application's writes. The transport reassembles ordered stream data and exposes it to HTTP. Protocol boundaries are logical promises, not matching parcel sizes all the way down.

Building what you see

The first response may be an HTML document, written in Hypertext Markup Language. The browser can begin processing it before every byte has arrived. It decodes the byte stream, tokenises the markup and builds a document tree. As it discovers references, it schedules more fetches for style sheets, scripts, images, fonts and other resources. Some may be found in local cache. Some use the same connection. Others require new DNS answers and connections to different hosts.

CSS, Cascading Style Sheets, supplies rules that can be matched against the document. The browser combines structure and style to calculate the size and position of boxes. It paints text, backgrounds, borders and images, then composites layers for the display. Changes do not always require repeating every stage, but a script that alters structure or dimensions can trigger new layout and paint work.

JavaScript complicates the sequence because it can run while the page is being built, modify the document, attach event handlers and fetch data. A news page may send the initial shell quickly, then request articles, recommendations, adverts and analytics from separate services. A single click can therefore open several administrative and trust relationships that the address bar does not list.

Browsers enforce controls around these relationships. The same-origin model restricts how documents from one origin can read resources from another. Cross-origin resource sharing lets servers grant selected access. Content security policy can constrain which sources a page may use. These controls sit above transport encryption. A perfectly encrypted cross-site request can still be forbidden by the browser's application security model.

The page becomes visible in stages. Text may appear before a web font. An image may fill later. A script may keep the main thread busy after the network is quiet. The response can be complete while interaction remains sluggish. "Load time" is therefore not one event, and network speed is one contributor among server work, dependency order and browser

computation.

How the design arrived

Packet switching did not emerge from one inventor or one country. Work at MIT, RAND and Britain's National Physical Laboratory developed in parallel during the 1960s. ARPANET provided a working research network, and its first recorded host-to-host transmission attempt in October 1969 sent 10 before the receiving system failed. By the end of that year four hosts were connected.

The deeper change came when researchers stopped treating one network as the whole environment. Robert Kahn and Vint Cerf developed an architecture for communication across unlike networks. Early TCP combined forwarding and transport functions; the design was later divided so IP handled addressing and packet forwarding, TCP handled reliable streams, and UDP gave applications a thinner transport. ARPANET's planned switch from NCP to TCP/IP on 1 January 1983 became a practical declaration that independent networks could share one internetworking system.

Scale then broke the host list. Paul Mockapetris designed DNS as a distributed hierarchical replacement for a centrally maintained table of names and addresses. Routing became hierarchical as one uniform system could no longer carry the growing network. Commercial providers, research networks and exchanges changed the internet from a funded research environment into shared infrastructure operated through many institutions and contracts.

The World Wide Web arrived later. At CERN in 1989, Tim Berners-Lee proposed joining hypertext documents across the existing internet. He developed the first server, browser and core web conventions around URLs, HTTP and HTML. The Web spread because it was an application built on the general network, not a replacement for it. Email, file transfer, voice, games and later applications continued to use the same underlying internet in other ways.

Each later layer answered a limit exposed by success. Caches and content networks reduced distance and repeated work. TLS made public communication safer against hostile paths. HTTP/2 multiplexed web exchanges; QUIC and HTTP/3 moved more transport control into encrypted software at the endpoints. IPv6 expanded addressing. None replaced the whole system. The internet grew by changing pieces while preserving the interfaces that other pieces depended on.

How we know

The internet leaves unusually strong but incomplete evidence. Its public standards define packet formats, state machines and protocol obligations in enough detail for independent implementations to interoperate. Source code, packet captures and browser developer tools show what one device sent and received. DNS queries, routing collectors, active probes and traceroute-like measurements reveal parts of naming and forwarding behaviour. Historical RFCs, project records, the UCLA IMP log, CERN archives and participant accounts document design choices and chronology.

None supplies a complete outside trace of every click. Encryption hides application content from observers without endpoint access. Providers keep much topology, traffic engineering and data-centre operation private. Routes and cache contents change, and measurements from one location may not generalise to another. Traceroute probes can follow different paths through load-balancing routers, producing an apparent sequence of links that no single flow uses.

The worked journey in this book is therefore a standards-constrained representative case. Each mechanism is real; the exact sequence is conditional. The honest answer to what happens when you click begins with a map and ends with a measurement.

What People Get Wrong

"The internet is the Web"

The Web is the part you visit through URLs, browsers and HTTP. The internet is the underlying system that joins networks and carries IP packets. Email, voice calls, multiplayer games, software updates, remote login and many machine-to-machine services can use the internet without being web pages.

The confusion became natural because the Web turned a technical network into something ordinary people could browse, link and publish. For many users, the browser was their first visible doorway and www became shorthand for being online. Later, mobile apps hid web technologies and internet connections behind icons, preserving the same confusion in a cleaner interface.

The correction matters because control and failure sit at different layers. A web service can be broken while email works. A government can filter particular applications without cutting every IP path. A browser rule can withhold a response even though the network delivered it. Treating the Web as the internet makes one successful application look like the whole machine that supports it.

The distinction also separates ownership. Internet protocols can support applications nobody anticipated, while a web platform can govern its users without controlling IP beneath it. Calling every online restriction an internet rule gives an application operator more apparent authority than it has.

"Wi-Fi is the internet"

Wi-Fi is a way for devices to communicate over a local radio network. The access point then needs an upstream connection if those devices are to reach wider networks. You can have strong Wi-Fi and no internet service, or a healthy internet connection behind an access point that your device cannot reach.

The mistake survives because the same symbol reports local radio attachment and is commonly used as a proxy for connectivity. Yet the bars mainly describe the link to the access point, not the access provider, DNS resolver, distant route or server. A crowded radio channel can also perform badly despite a high signal reading.

This distinction is useful during failure. If several wired and wireless devices cannot reach anything beyond the router, the upstream service deserves attention. If one device has trouble while others work, the local device or link is a better suspect. "Restart the Wi-Fi" sometimes works because home equipment combines access point, router, firewall, address translation and resolver forwarding in one box. The label hides the bundle.

Mobile data makes the contrast obvious: a phone can lose Wi-Fi, attach to a cellular network and keep the same internet conversation alive if the transport and application support the change. The access technology changed; the wider service did not.

"Every click starts from scratch"

A second visit can be faster even though the broadband has not improved. The browser may reuse a response without leaving the device, or send another request over a connection it already has. Farther away, a resolver may remember the address and an edge cache may hold the content. These are different shortcuts: one avoids a transfer, another avoids setup, another avoids a distant origin. A timing comparison that ignores them can mistake preparation for a faster network.

The mistake persists because reused work is invisible. A request that never leaves the device looks like an unusually fast trip. A DNS answer from cache looks like no lookup. A reused connection looks like no handshake. Success compresses preparation into the instant after the click.

The correction matters because cold and warm requests are different experiments. Clearing browser data, changing networks or waiting for cached state to expire can expose work that a repeat visit avoided. The first attempt can also alter the second by opening a connection, populating a

cache or discovering that a remembered connection has failed.

Existing state can also hurt. A stale response, expired session, obsolete DNS answer or half-broken connection can preserve the wrong past.

Reloading sometimes helps because it changes which state is reused, not because the internet has been commanded to try harder.

"Packets take the fastest route"

Routers do not conduct a global race among every possible path for each packet. BGP distributes reachability and path attributes among autonomous systems. Operators apply local preferences shaped by customer, peer and provider relationships, policy, resilience and capacity. Inside a network, other routing systems make further choices.

The resulting path follows installed fibre, cable landings, rights of way and places where networks have agreed to meet. A nearby service can be topologically distant. Traffic can leave a country or region and return because that route is available or preferred. The return path need not mirror the outward one.

A route that costs less or fits a commercial relationship can be selected over one with lower delay. The phrase “best path” therefore means best under the selecting system's rules, not fastest for the user. Speed can matter, but no universal objective function governs every network.

This explains why a speed test to one server does not measure every destination and why changing provider, resolver or service edge can change performance. It also limits what one trace proves. Some routers do not answer diagnostic probes, replies can follow another route and a content network may send another user to another edge. One measurement describes one path, from one vantage point, at one time.

"HTTPS means the site is safe"

HTTPS means HTTP is being carried through a protected channel, normally with the server authenticated for the requested name under the browser's certificate rules. It is designed to resist eavesdropping, tampering and message forgery on the path. That is a major protection.

It is not a character reference. A criminal can register a domain, obtain a valid certificate and run an encrypted fraud. A legitimate site can be compromised above the transport layer. Harmful content can cross an authentic, confidential connection. The browser cannot infer truth from possession of cryptographic keys.

The old padlock icon encouraged the wider reading because a simple visual signal had to compress certificate status and connection protection. The folk meaning remains. Read the claim precisely: the channel to the named endpoint has passed specified checks. Then evaluate the endpoint, transaction and content with other evidence. Cryptography can tell you whom the channel reached under its naming rules. It cannot tell you whether to believe them.

The name deserves scrutiny too. A certificate for a lookalike domain can be valid for that lookalike. The browser has authenticated the spelling it was given, not the brand you meant. A security signal cannot repair an imprecise human destination.

"The cloud has no physical location"

Cloud computing turns machines, storage and networks into rented services controlled through software. The abstraction is valuable because a customer can request capacity without choosing a rack or replacing a failed disk. The equipment still occupies buildings, consumes power, rejects heat and connects through fibre.

"Stored in the cloud" may mean copies across several facilities, fragments under an erasure code, or one regional service with backups elsewhere. It may also mean dependence on a provider's identity system, management

plane and private backbone. Physical redundancy and administrative independence are different things.

The correction matters when latency, law and failure matter. A region can be distant. Two services can share one facility or supplier. Data placement can affect legal obligations. A power event, software deployment or control-plane fault can disable resources that looked separate in a dashboard. The cloud removes the customer's need to handle much of the hardware. It does not remove hardware from the world.

Availability zones, regions and backup copies are useful labels only when their failure boundaries are understood. Two resources with different labels may still share identity, billing, deployment or network control. Recovery requires a tested path to usable data, not merely another copy in an inaccessible system.

"The internet has no centre"

There is no single operational centre, and the system was designed to connect networks that retain their own administration. That fact is often stretched into the claim that the internet has no centres at all.

In practice, different layers have different concentrations. Root and top-level DNS functions coordinate naming. Large recursive resolvers answer for many users. Transit networks, internet exchanges and cable systems shape reachability. Certificate authorities sit inside browser trust. Cloud and content-delivery companies host or carry many services. A widely used software component can become a common failure point without owning a metre of fibre.

The mistake arises from treating decentralisation as a physical property rather than an arrangement of authority and dependency. A system can lack one sovereign while containing many powerful intermediaries. It can route around one damaged link while remaining exposed to one shared control system. The useful question is not whether a centre exists. It is which function is concentrated, what alternatives are independent, and how difficult switching would be under pressure.

Calling the system decentralised should begin the enquiry, not settle it. The relevant unit is the dependency: a service can have many servers and still have only one way to manage them.

Use It

A blank page tells you almost nothing about where the failure happened. The useful move is to turn that one symptom into a smaller question. These six lenses follow from the machinery behind the click.

Locate the failing layer

Suppose a site works on your phone's mobile connection but not on your laptop's Wi-Fi. That is a clue, not a verdict on the router: you changed both the device and the network. Trying the phone on the same Wi-Fi narrows the comparison. Then separate local attachment, name resolution, transport connection, certificate checks and the application's response. Each test should answer a smaller question than whether the internet works.

Higher layers depend on lower ones while producing similar symptoms. A DNS failure and a dead server can both yield a blank page. A browser extension can break one site while every route remains healthy. Typing a server's IP address into a browser is not a clean substitute for testing its name: HTTPS identity checks and shared hosting can require the original host name.

This is better than random restarting. A restart clears state across several layers, so success teaches little. Diagnosis preserves the distinction and makes the next failure cheaper.

Spend the latency budget

Count the waits that have to happen one after another. Suppose a page requires five sequential round trips, each taking 40 milliseconds. Those waits alone cost 200 milliseconds, before allowing for server work or data transfer. Ten parallel requests do not necessarily take ten times as long. The slowest dependency chain matters more than the length of the request list.

Bandwidth matters when enough data must move. Latency dominates many small sequential exchanges. That is why a large download can run quickly while a complex site feels hesitant, and why placing content closer helps even when the user's advertised line rate is unchanged.

Ask which work is sequential and which can overlap. Reusing a connection, caching a response, removing a redirect or avoiding one blocking script can save an entire wait. Optimisation becomes clearer when the unit is not “faster internet” but one round trip, one queue, one dependency or one transfer.

Separate name, address and service

A domain name, IP address, server and organisation are not interchangeable. One name can map to several addresses. Several names can share an address. One address can lead through a load balancer to many machines. One visible service can call dozens of private services.

Carry this distinction into outages and ownership claims. Moving a domain does not necessarily move the content. Changing DNS can redirect users while old answers remain cached. Blocking an address can affect unrelated names. Locating an IP address does not identify every place where data is processed or stored.

The same lens improves architecture. Stable names let services move. Short-lived addresses let infrastructure change. Indirection creates flexibility, but every indirection adds configuration, cache state and another place where reality can disagree with the intended map.

Read the security claim precisely

Translate each security signal into the property it supports. TLS can authenticate an endpoint under a naming and certificate system, protect confidentiality on the channel and detect alteration. It does not validate claims made by the application. Routing-origin validation can test an advertised origin against authorised prefix data; on its own, it does not prove the whole route legitimate. A VPN protects a tunnel and changes which intermediary can observe traffic; it does not make the endpoint safe.

Then identify the unprotected edges. What can the endpoint read? Which metadata remains visible? Who controls the resolver, browser trust store, identity account and software update path? What happens after decryption?

Precise reading sounds less reassuring than a padlock. It is more useful because it tells you which threat has been reduced and which evidence you still need.

Expect repair, duplication and stale state

Distributed systems are built around partial knowledge. A sender may know that it transmitted data without knowing whether the receiver acted. A cache may hold a valid older response while the origin has changed. A resolver may reuse an answer until its TTL expires. A route may be withdrawing while some networks still prefer it.

This means retries need design. Repeating a read is usually harmless. Repeating a payment or account change can be costly unless the operation carries an identifier that lets the server recognise a duplicate. Timeouts need room for ordinary delay, but cannot mean waiting forever. Caches need explicit freshness and validation rules.

When an online action hangs, do not assume nothing happened. The request may have succeeded and the reply may have been lost. The interface should make status inspectable rather than turning uncertainty into another blind click.

Follow shared dependencies

Redundancy is a claim about independence, not a count of logos. Two links can share a duct. Two services can run in one cloud region. Several domains can use one authoritative DNS provider. Different applications can depend on one identity platform, certificate authority, software library or content network.

Map dependencies by function: naming, addressing, routing, power, physical path, computation, storage, authentication and control. Then ask which failures cross the boundaries you assumed were separate. The answer often sits outside the visible service.

This lens also clarifies power. A company may not own the internet yet can hold leverage at one layer because many others chose it as an intermediary. Concentration is neither proved nor disproved by the architecture's decentralised design. It has to be measured function by function.

The limits

A layered model can become too tidy. Real networks use middleboxes, tunnels, private backbones, carrier-grade address translation, enterprise inspection, satellite links, mobile cores and proprietary delivery systems. Browsers race connection options, cache aggressively and change implementation. Providers combine layers in one product and optimise around standards in ways the standards do not prescribe.

The model also explains movement better than meaning. It cannot tell you whether a platform's ranking is fair, whether data collection is legitimate, whether access is affordable or whether a government should regulate an intermediary. Those questions need economics, law, politics and ethics as well as packet mechanics.

Finally, knowing the layers does not grant a view from nowhere. Your measurement begins at one device, under one resolver and route, at one moment. The path seen from elsewhere may differ. Use the model to ask

cleaner questions, then keep the answer tied to the vantage point that produced it.

The one thing to keep

The page on your screen is the last act, not the whole event.

Somewhere before it appeared, separate machines had to agree about different things. Where a name led. Where the next packet could go. Whether the endpoint could prove its identity. Which copy was usable. None of those answers, alone, would put a sentence on your screen. Together they let a browser assemble something you could read, including material that might have arrived before you asked for it.

This is a more demanding achievement than sending one object down one pipe. The participants do not need a common owner or a complete picture of the work. They need compatible boundaries, enough memory to continue the exchange and ways to notice when a promise has failed. A lost packet need not spoil the picture. A changed server need not change the name. A new kind of application need not ask every router to learn what it means.

The limits belong beside the achievement. Reliable delivery does not make a claim true. A valid certificate does not make a shop honest. A rack of backup servers does not help when the only system that can reach them is down. These are not reasons to distrust everything. They are reasons to ask for the right evidence from the right part of the system.

That changes the next ordinary click. When it is slow, you can distinguish moving too much data from waiting too many times. When it fails, you can separate the name from the route and the route from the service. When it succeeds, you can recognise how much preparation and repair the smooth surface has concealed.

The cloud was a convenient way to stop drawing. It need not be where understanding stops. Behind the page are people maintaining equipment, organisations making agreements and machines doing limited jobs well enough to cooperate. None of them has to understand why you wanted to

read the article. Their achievement is that you can.

Terms

A glossary of the words hidden inside one ordinary click, and the distinctions that keep them useful.

Internet. The global internetwork formed when independently operated networks exchange IP packets under shared protocols. It is infrastructure for many applications, of which the World Wide Web is one.

World Wide Web. A linked information system built on the internet around URLs, HTTP, HTML and browsers. Web pages use internet transport, but email, games and voice need not use the Web.

Protocol. An agreed set of message formats, meanings and behaviours. Protocols let independently built systems interoperate without sharing one internal design or one manufacturer.

Layer. A useful division of responsibility in which one protocol offers a service to another. Layers hide implementation detail, though real systems often combine functions or expose cross-layer effects.

Packet. A bounded unit of data and control information carried through a packet-switched network. Large application messages are divided, forwarded and reassembled rather than sent as one continuous object.

Link. A one-hop communication path between neighbouring network interfaces, such as an Ethernet segment, Wi-Fi connection, fibre circuit or mobile-radio link. IP crosses many links during one end-to-end journey.

Bandwidth. The rate at which a link or path can carry data, commonly measured in bits per second. High bandwidth moves large transfers faster but cannot remove propagation delay.

Latency. Delay between cause and response. It includes signal travel, queuing, processing and protocol waits. Interactive performance often depends on several sequential latencies rather than peak bandwidth.

IP address. A numerical label used as a source or destination in Internet Protocol packets. It supports forwarding but does not by itself identify a person, organisation, device type or physical location.

IPv4 and IPv6. The two deployed Internet Protocol versions. IPv4 uses 32-bit addresses, whose scarcity encouraged address translation. IPv6 uses 128-bit addresses and a revised packet format. Both depend on routing and higher-layer transports.

Router. A device or software system that forwards packets between networks. It uses routing information to choose a next hop, normally without retaining application-level understanding of the conversation.

Prefix. A block of IP addresses sharing leading bits, written with a length such as /24 or /48. Routing aggregates destinations by prefixes rather than listing every endpoint separately.

Autonomous system. A network or collection of networks presenting a coherent external routing policy, identified by an autonomous system number. Providers, universities and large content networks can operate one or more.

BGP. Border Gateway Protocol, the system through which autonomous systems exchange reachability and path information. Its selections reflect local policy and relationships, not a universal search for lowest latency.

Control plane. The processes that create and update network state, including routing information. The data plane uses that prepared state to forward packets. BGP does not run for each packet.

Peering. An arrangement in which networks exchange traffic directly, commonly for their respective customers. Terms, locations and capacity vary, and direct exchange can reduce reliance on paid transit.

Transit. A service in which one network pays another to carry traffic towards the wider internet. Transit providers advertise routes and provide reachability beyond a direct pair of networks.

Internet exchange point. A facility and shared switching environment where many networks can interconnect. An exchange lowers the physical burden of arranging numerous separate connections but does not dictate every peering agreement.

URL. Uniform Resource Locator, a structured identifier containing a scheme and other components such as host, port, path, query and fragment. It tells software how to identify and process a resource reference.

DNS. Domain Name System, the distributed hierarchical database used to publish names and related records. It commonly maps host names towards IP addresses through delegation, authority and caching.

Recursive resolver. A DNS service that accepts a client's query and obtains or retrieves the final answer, following referrals and using caches. The resolver's location and policy can affect results and privacy.

Authoritative server. A DNS server publishing data for a delegated zone. It supplies the zone's records; it does not perform the same user-facing search role as a recursive resolver.

Cache. Stored data reused to avoid repeated work. DNS and HTTP caches reduce latency and upstream load, but their rules must decide freshness, validation, variation and when reuse is unsafe.

TTL. Time to live. In DNS, it sets how long a cached record is treated as fresh. Short and long values trade faster change against query load and resilience.

TCP. Transmission Control Protocol, which presents applications with a reliable ordered byte stream. It uses sequencing, acknowledgements, retransmission, flow control and congestion control above IP's best-effort delivery.

UDP. User Datagram Protocol, a thin transport that adds ports and a checksum without built-in connection state, ordering or retransmission. Applications can value timeliness or build other guarantees above it.

QUIC. A secure transport carried in UDP datagrams. It integrates TLS, reliable delivery, congestion control, multiple streams and support for some path changes, and is the transport used by HTTP/3.

TLS. Transport Layer Security, the protocol used to authenticate endpoints under application rules and protect channel confidentiality and integrity. It secures communication, not the honesty of the content.

HTTP. Hypertext Transfer Protocol, the request-and-response application protocol of the Web. Its methods, status codes, headers and representations can run over different transport arrangements.

Content-delivery network. A distributed system that serves content from edge locations nearer users or network interconnections. It can reduce distance and origin load while becoming an important shared dependency.

Go Deeper

Four routes into the subject, chosen for different questions.

The physical journey. Andrew Blum, *Tubes: A Journey to the Center of the Internet* (Ecco, 2012). Blum visits cable landings, internet exchanges and data centres to recover the buildings and people hidden by the cloud icon. It is the most inviting next step after this book because it begins with places rather than packet formats. Read it for physical intuition and reporting, while remembering that it predates widespread HTTP/3, current cloud concentration and much of today's encrypted transport. Pair its site visits with current protocol documents so the visible buildings and the invisible rules stay connected.

The working system. James F. Kurose and Keith W. Ross, *Computer Networking: A Top-Down Approach*, 9th edition (Pearson, 2025). This is the mechanism book: applications first, then transport, routing, links, wireless and security, supported by packet-capture exercises. It is a university textbook and therefore much longer and more formal than an hour's read, but its sequence is unusually humane. Use it when you want to calculate, inspect and implement rather than retain a general model. Its packet-level

examples are especially useful with a capture tool open, because fields that look ceremonial on paper become evidence of a live exchange.

The institutional history. Janet Abbate, *Inventing the Internet* (MIT Press, 1999). Abbate explains why the internet's architecture cannot be separated from research funding, military aims, academic culture, standards communities and users who turned general infrastructure towards applications its sponsors had not predicted. It is strongest on the contest between alternative designs and on decentralisation as an organisational achievement. Its narrative ends before the modern platform, cloud and mobile era, which makes it history rather than a guide to current deployment. That stopping point is useful: it prevents later commercial success from making one architectural path look inevitable.

The primary Web account. Tim Berners-Lee with Mark Fischetti, *Weaving the Web* (HarperSanFrancisco, 1999). The inventor of the World Wide Web describes the path from his CERN proposal to URLs, HTTP, HTML and the effort to keep the Web open. Read it to hear the design intentions from inside the work, and to keep the Web distinct from the older internet beneath it. It is personal evidence and advocacy, not a neutral or current technical survey, and that is the reason to read it beside Abbate rather than instead of her. Its strongest contribution is the design problem: how to make linking universal without placing every document under one publisher or database.

Notes and Sources

The book follows a representative web click rather than claiming one universal trace. Browser caches, service workers, existing connections, address-family selection, encrypted DNS, content networks, private backbones, enterprise controls and provider policy can remove, reorder or add steps. The click begins against prior state: DNS caches, forwarding tables, connection state, edge copies and application records may have been prepared before the action. The standards below establish protocol roles and permitted behaviour. The worked sequence states its assumptions where deployment varies.

The Whole Thing in One Page and Why You Should Care

The first ARPANET message. UCLA's reproduction of the IMP log records the attempt at 22:30 on 29 October 1969 to send login from UCLA to SRI, the successful transmission of l and o, the crash and the completed login about an hour later. The Internet Society's participant history supplies the wider ARPANET and internetworking chronology. The opening uses the failure as a documented event, not as evidence that ARPANET alone was already the modern internet.

Latency and bandwidth. The distinction is used in its standard engineering sense. Bandwidth describes carrying rate; latency includes propagation, transmission, processing, queuing and protocol waits. The motorway comparison is explanatory, not a claim that every delay is geographical. Kurose and Ross provide the main synthesis, while the transport and HTTP standards below establish why sequential exchanges and connection setup matter.

Standards and independent operation. The claim that different organisations can interoperate through public specifications is supported by RFC 2026 on the Internet Standards Process, the protocol RFCs listed below and the existence of independent browser and network implementations. This does not imply that every specification is implemented identically or that standards bodies operate deployed networks.

The Core Ideas

Open architecture and packets. Vinton Cerf and Robert Kahn's 1974 paper proposed intercommunication among packet networks without requiring them to share one internal design. David Clark's 1988 account explains the design goals and trade-offs of the DARPA internet protocols after deployment experience. RFC 1958 records later architectural principles, including the enduring aim of connecting heterogeneous networks. Janet Abbate supplies the institutional history and prevents a clean technical genealogy from displacing funding, organisations, rival systems and users.

Existing state and the control plane. RFC 1034 and RFC 1035 establish DNS caching; RFC 9111 establishes HTTP freshness, reuse and validation; the W3C Service Workers specification defines fetch interception and application-managed responses; RFC 9000 and RFC 9846 define reusable transport and cryptographic state under bounded conditions. RFC 4271 defines the routing information exchange that helps create forwarding state before packets arrive. These sources support the claim that a click often reuses prepared state. They do not imply that every browser, network or service retains the same state or observes the same expiry rules.

IP and link boundaries. IPv4 is specified in RFC 791 and IPv6 in RFC 8200. Their packet formats and forwarding role support the distinction between the end-to-end IP destination and the local link frame used for the next hop. IPv6 defines 128-bit addresses; IPv4 uses 32 bits. The manuscript treats IP as best effort and assigns reliability to higher layers where required. It does not imply that routers never modify permitted fields or that tunnels, translation and middleboxes preserve a visibly identical packet throughout.

DNS. RFC 1034 and RFC 1035 define the distributed hierarchical name system, delegation, recursive and iterative resolution, authoritative data and caching. The DNS was designed so names need not contain addresses or routes and so the database could be maintained in distributed form with local caching. TTL is a freshness lifetime, not a countdown carried by a

travelling update. RFC 8767 permits retained stale data to be served in exceptional circumstances when authoritative data cannot be refreshed. RFC 3596 defines the AAAA records used for IPv6 addresses.

Root service and anycast. IANA lists thirteen named root-server identities, A through M, and directs readers to the operators' maps of the many deployed instances. RFC 4786 explains anycast operation, in which the same service address can be originated from multiple locations and routing selects a reachable instance. The text therefore rejects the familiar but false image of thirteen individual machines.

Addresses, prefixes and BGP. RFC 4271 defines BGP as an inter-autonomous-system routing protocol that exchanges reachability, autonomous-system path information and prefixes, and permits policy decisions and route aggregation. The description of customer, peer and transit preferences is a common operational pattern rather than a universal ranking. Contracts and policies vary by network, geography and time.

Route leaks and routing security. RFC 7908 defines and classifies BGP route leaks. RFC 6480 describes the Resource Public Key Infrastructure, and RFC 6811 specifies prefix-origin validation. Origin validation can test whether the originating autonomous system is authorised for a prefix. It does not prove that every intermediate path or policy choice is legitimate, which is why the manuscript states the limitation directly.

TCP. RFC 9293 is the current base TCP specification. It supports the reliable ordered byte-stream account, including sequence numbers, acknowledgements, retransmission, flow control and the absence of application message boundaries. RFC 5681 supplies the standard TCP congestion-control framework. Particular algorithms and implementations differ, so the prose explains the feedback bargain without claiming one universal window trajectory.

UDP, QUIC and HTTP/3. RFC 768 defines UDP's thin datagram service. RFC 9000 defines QUIC as a UDP-based, flow-controlled, multiplexed and secure transport with connection migration support. RFC 9001 specifies

QUIC's use of TLS, and RFC 9002 specifies loss detection and congestion control. RFC 9114 maps HTTP semantics over QUIC and supports the account of per-stream flow control and the avoidance of TCP-level head-of-line blocking between HTTP streams. QUIC does not remove congestion, packet loss or all waiting. RFC 9000, section 13.3, distinguishes resending lost information from retransmitting whole packets. RFC 9114, section 3.1, describes endpoint discovery, including prior alternative-service advertisements.

TLS and the limits of the padlock. The current TLS 1.3 specification used here is RFC 9846, published in July 2026. It supersedes the earlier TLS 1.3 specification while retaining TLS version 1.3 and describes authentication, confidentiality, integrity, the handshake and record protection. RFC 5280 supplies the certificate and path-validation framework, while RFC 9525 covers service identity in TLS applications. The CA/Browser Forum Baseline Requirements, version 2.2.9 dated 6 August 2026, establish current issuance and domain or IP validation rules for publicly trusted TLS server certificates. Together these sources support a narrow claim about the protected channel and authenticated service identity under application rules. They do not warrant claims about an operator's honesty, the truth of content or the safety of downloaded code.

Metadata and shifted trust. TLS does not hide all traffic characteristics or the IP information needed for delivery. Encrypted DNS and virtual private networks can reduce what one observer sees while increasing the information available to another service. These are changes in exposure and trusted intermediaries, not a claim that privacy gains must be zero-sum. RFC 8484 specifies DNS over HTTPS and its privacy considerations; RFC 7858 specifies DNS over TLS.

HTTP requests, responses and caches. RFC 9110 defines HTTP semantics, including methods, status codes, fields, representations and intermediaries. RFC 9111 defines private and shared caches, freshness, validation and reuse. RFC 9113 and RFC 9114 supply the HTTP/2 and HTTP/3 transport mappings. The edge, proxy, origin and cache sequence is

representative. A particular service may combine or omit those roles.

Browser construction. The WHATWG URL, Fetch and HTML living standards establish URL parsing, origins, fetching, cross-origin rules, tokenisation and tree construction. Mozilla's "Populating the page: how browsers work" explains the rendering pipeline, including processing before the full document arrives. Painting and compositing details vary by engine and platform. The text therefore explains a common processing pipeline without presenting one browser's implementation as the standard itself.

Same-origin controls. The same-origin model, Fetch's cross-origin rules and related browser policies explain why successful packet delivery does not guarantee that page code may read a response. The W3C Content Security Policy specification defines the separate application control. Transport protection and browser permission are kept distinct throughout.

Content delivery and centres. HTTP caching, DNS delegation, anycast and interconnection explain the technical means by which services can be distributed. Blum supplies reported physical detail about exchanges, data centres and cable landings. Abbate and Clark supply the architectural and institutional background. The book makes no quantitative claim about one provider's market share or one universal traffic concentration. Facebook's 5 October 2021 engineering account explains the previous day's backbone failure and resulting loss of DNS reachability; Cloudflare's contemporaneous measurements corroborate route withdrawals and lookup failures, not the private initiating command. The event illustrates shared dependence, not its frequency.

Governance. RFC publication does not amount to command over deployment. IETF standards are developed through an open process, IANA coordinates unique names and numbers, operators choose routing and interconnection policy, browsers maintain trust stores, and governments regulate within jurisdictions. The claim is plural governance, not the absence of power or the existence of one world regulator.

How It Actually Works

The example domain. IANA reserves `example.com`, `example.net` and `example.org` for documentation. The worked click uses `example.com` so no factual behaviour is attributed to a commercial service. The path `/article` and every application action behind it are explicitly illustrative.

Local configuration and address translation. RFC 1918 defines private IPv4 address blocks. RFC 3022 describes traditional network address translation. The home-network account is common but not universal: some users receive public addresses, some providers use carrier-grade translation, some networks are IPv6-only with transition mechanisms, and enterprise networks can add proxies or inspection.

DNS path. The recursive sequence is deliberately conditional. A browser, operating system or resolver cache can answer; the resolver may already know the relevant delegation; aliases can add queries; and resolver policy can vary the answer. The text does not imply that every click contacts a root or top-level server.

Routing path. Forwarding is distinguished from route calculation. BGP helps networks exchange reachability before a packet arrives; forwarding then applies installed state. Longest-prefix matching, aggregation, asymmetric return paths, finite queues and reconvergence are described at the level needed to follow a click. Private traffic engineering, multiprotocol label switching and tunnels are acknowledged but left outside the one-hour route.

Connection setup. The HTTP/3 case is presented with a fallback to HTTP/2 or HTTP/1.1 over TCP and TLS. Session resumption, connection reuse and early data can reduce setup, but their availability and security conditions differ. No fixed number of round trips is claimed for every browser, server and network state.

Caching and the origin. RFC 9111 supports freshness and validation. The back-end sequence is an illustrative architecture made explicit as such. It shows why one public request can produce private service calls, retries and

timeout questions without alleging that every site uses a load balancer, database, object store or third-party identity system.

Distributed uncertainty. The warning about a lost reply after a successful state change is a general distributed-systems property. It motivates idempotency and inspectable status without turning this book into a full treatment of distributed databases, consensus or transaction design.

Web history. CERN's Web history and timeline support Tim Berners-Lee's March 1989 proposal and the development of the first server, browser and site by the end of 1990. Berners-Lee's participant account supplies design intention but is treated as advocacy and memory, not as the only historical authority. Abbate supplies the broader network history.

How we know. Protocol standards define intended and permitted behaviour, while packet captures and endpoint tools reveal one local exchange. Routing collectors, DNS measurements and active probes reveal partial external state. Encryption, private provider systems, changing routes and vantage-point dependence prevent a complete universal trace. The Paris Traceroute researchers document how load balancing can produce false links and missing paths in conventional traceroute results. The final sentence therefore separates a standards-constrained map from a measurement of one live click.

What People Get Wrong

The internet and the Web. The distinction follows from chronology and architecture: IP internetworking predates the Web, and HTTP is one application protocol above internet transport. The correction does not deny that web technologies also operate inside many mobile applications.

Wi-Fi. IEEE 802.11 is a local-link family, while internet reachability requires further routing and service. The book avoids treating a Wi-Fi signal indicator as a measurement of resolver health, upstream reachability or distant server performance.

Existing state and fastest routes. DNS, HTTP and service-worker caches,

reusable connections and installed forwarding tables support the correction that every click does not start from scratch. RFC 4271 supports policy-based interdomain route selection, not a universal latency optimiser. Traceroute-like tools expose replies from some hops under particular forwarding and filtering conditions; they are useful measurements rather than permanent maps. Geographic location, network topology and administrative path are kept separate.

HTTPS. RFC 9846 and RFC 9525 support the narrow transport and identity claim. Chromium's 2023 explanation of its lock-icon redesign documents confusion between connection protection and website trustworthiness. Domain-valid certificates can protect connections to fraudulent or lookalike sites because certificate validation is not a moral or commercial investigation.

Cloud and centres. The physical correction draws on Blum and on the operational requirements implicit in routing, interconnection, data-centre and caching systems. The concentration correction is structural rather than quantitative. It asks which function is concentrated and whether alternatives are independent; it does not claim that the internet has one hidden centre.

Use It and Terms

The diagnostic, latency, naming, security, stale-state and dependency lenses are deductions from the protocol model rather than a method attributed to one author. Each is bounded by the limitations stated in the section. The five round trips at 40 milliseconds are a hypothetical arithmetic illustration, not a performance measurement. The device/network comparison is hypothetical too. The glossary follows current conventional meanings in the cited standards. Terms with broad industry use, including cloud and content-delivery network, are defined only to the depth needed for the click.

Go Deeper

Publisher and author records were checked for the four recommended works. Pearson dates publication of Kurose and Ross's ninth edition to 20 June 2025; its copyright date is 2026. Blum is the accessible physical journey; Kurose and Ross provide the current technical textbook and authors' exercises; Abbate provides the major institutional history; Berners-Lee and Fischetti provide a primary participant account of the Web's design. Their dates and viewpoints are made visible so an older work is not mistaken for a current protocol manual.

Bibliography

Primary standards, papers and technical records

Abley, Joe, and Kurt Lindqvist. "Operation of Anycast Services." RFC 4786. Internet Engineering Task Force, December 2006.

<https://www.rfc-editor.org/rfc/rfc4786>

Allman, M., V. Paxson and E. Blanton. "TCP Congestion Control." RFC 5681. Internet Engineering Task Force, September 2009.

<https://www.rfc-editor.org/rfc/rfc5681>

Berners-Lee, Tim. "Information Management: A Proposal." CERN, March 1989. <https://www.w3.org/History/1989/proposal.html>

Bishop, Mike, ed. "HTTP/3." RFC 9114. Internet Engineering Task Force, June 2022. <https://www.rfc-editor.org/rfc/rfc9114>

Bradner, Scott. "The Internet Standards Process, Revision 3." RFC 2026. Internet Engineering Task Force, October 1996.

<https://www.rfc-editor.org/rfc/rfc2026>

CA/Browser Forum. Baseline Requirements for the Issuance and Management of Publicly-Trusted TLS Server Certificates. Version 2.2.9, 6 August 2026. Accessed 5 September 2026.

<https://cabforum.org/working-groups/server/baseline-requirements/requirements/>

Carpenter, Brian, ed. “Architectural Principles of the Internet.” RFC 1958. Internet Architecture Board, June 1996.

<https://www.rfc-editor.org/rfc/rfc1958>

Cerf, Vinton G., and Robert E. Kahn. “A Protocol for Packet Network Intercommunication.” IEEE Transactions on Communications 22, no. 5 (May 1974): 637-648. <https://doi.org/10.1109/TCOM.1974.1092259>

Clark, David D. “The Design Philosophy of the DARPA Internet Protocols.” ACM SIGCOMM Computer Communication Review 18, no. 4 (August 1988): 106-114. <https://doi.org/10.1145/52324.52336>

Cooper, D., S. Santesson, S. Farrell, S. Boeyen, R. Housley and W. Polk. “Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile.” RFC 5280. Internet Engineering Task Force, May 2008. <https://www.rfc-editor.org/rfc/rfc5280>

Deering, Stephen, and Robert Hinden. “Internet Protocol, Version 6 (IPv6) Specification.” RFC 8200. Internet Engineering Task Force, July 2017. <https://www.rfc-editor.org/rfc/rfc8200>

Eddy, Wesley, ed. “Transmission Control Protocol (TCP).” RFC 9293. Internet Engineering Task Force, August 2022. <https://www.rfc-editor.org/rfc/rfc9293>

Fielding, Roy, Mark Nottingham and Julian Reschke, eds. “HTTP Caching.” RFC 9111. Internet Engineering Task Force, June 2022. <https://www.rfc-editor.org/rfc/rfc9111>

Fielding, Roy, Mark Nottingham and Julian Reschke, eds. “HTTP Semantics.” RFC 9110. Internet Engineering Task Force, June 2022. <https://www.rfc-editor.org/rfc/rfc9110>

Hoffman, Paul, and Patrick McManus. “DNS Queries over HTTPS (DoH).” RFC 8484. Internet Engineering Task Force, October 2018. <https://www.rfc-editor.org/rfc/rfc8484>

Hu, Zi, Liang Zhu, John Heidemann, Allison Mankin, Duane Wessels and

Paul Hoffman. “Specification for DNS over Transport Layer Security (TLS).” RFC 7858. Internet Engineering Task Force, May 2016.

<https://www.rfc-editor.org/rfc/rfc7858>

IANA. “IANA-managed Reserved Domains.” Accessed 5 September 2026.

<https://www.iana.org/help/example-domains>

IANA. “Root Servers.” Accessed 5 September 2026.

<https://www.iana.org/domains/root/servers>

Iyengar, Jana, and Ian Swett, eds. “QUIC Loss Detection and Congestion Control.” RFC 9002. Internet Engineering Task Force, May 2021.

<https://www.rfc-editor.org/rfc/rfc9002>

Iyengar, Jana, and Martin Thomson, eds. “QUIC: A UDP-Based Multiplexed and Secure Transport.” RFC 9000. Internet Engineering Task Force, May 2021. <https://www.rfc-editor.org/rfc/rfc9000>

Lawrence, David C., Warren Kumari and Puneet Sood. “Serving Stale Data to Improve DNS Resiliency.” RFC 8767. Internet Engineering Task Force, March 2020. <https://www.rfc-editor.org/rfc/rfc8767>

Lepinski, Matt, and Stephen Kent. “An Infrastructure to Support Secure Internet Routing.” RFC 6480. Internet Engineering Task Force, February 2012. <https://www.rfc-editor.org/rfc/rfc6480>

Mockapetris, Paul. “Domain Names: Concepts and Facilities.” RFC 1034. Internet Engineering Task Force, November 1987.

<https://www.rfc-editor.org/rfc/rfc1034>

Mockapetris, Paul. “Domain Names: Implementation and Specification.” RFC 1035. Internet Engineering Task Force, November 1987.

<https://www.rfc-editor.org/rfc/rfc1035>

Mohapatra, Pradosh, John Scudder, David Ward, Randy Bush and Rob Austein. “BGP Prefix Origin Validation.” RFC 6811. Internet Engineering Task Force, January 2013. <https://www.rfc-editor.org/rfc/rfc6811>

Mozilla. “Critical rendering path.” MDN Web Docs. Accessed 5 September

2026.

https://developer.mozilla.org/en-US/docs/Web/Performance/Guides/Critical_rendering_p

Mozilla. “Populating the page: how browsers work.” MDN Web Docs. Accessed 5 September 2026.

https://developer.mozilla.org/en-US/docs/Web/Performance/Guides/How_browsers_wor

Paris Traceroute project. “About.” Research explanation of traceroute deficiencies under load balancing. Accessed 5 September 2026.

<https://paris-traceroute.net/about/>

Postel, Jon. “Internet Protocol.” RFC 791. Defense Advanced Research Projects Agency, September 1981. <https://www.rfc-editor.org/rfc/rfc791>

Postel, Jon. “User Datagram Protocol.” RFC 768. Information Sciences Institute, August 1980. <https://www.rfc-editor.org/rfc/rfc768>

Rekhter, Y., B. Moskowitz, D. Karrenberg, G. J. de Groot and E. Lear. “Address Allocation for Private Internets.” RFC 1918. Internet Engineering Task Force, February 1996. <https://www.rfc-editor.org/rfc/rfc1918>

Rekhter, Yakov, Tony Li and Susan Hares, eds. “A Border Gateway Protocol 4 (BGP-4).” RFC 4271. Internet Engineering Task Force, January 2006. <https://www.rfc-editor.org/rfc/rfc4271>

Rescorla, Eric. “The Transport Layer Security (TLS) Protocol Version 1.3.” RFC 9846. Internet Engineering Task Force, July 2026. <https://www.rfc-editor.org/rfc/rfc9846>

Saint-Andre, Peter, and Rich Salz. “Service Identity in TLS.” RFC 9525. Internet Engineering Task Force, November 2023. <https://www.rfc-editor.org/rfc/rfc9525>

Sriram, K., D. Montgomery, D. McPherson, E. Osterweil and B. Dickson. “Problem Definition and Classification of BGP Route Leaks.” RFC 7908. Internet Engineering Task Force, June 2016. <https://www.rfc-editor.org/rfc/rfc7908>

Srisuresh, P., and K. Egevang. “Traditional IP Network Address Translator

(Traditional NAT).” RFC 3022. Internet Engineering Task Force, January 2001. <https://www.rfc-editor.org/rfc/rfc3022>

Thomson, Martin, and Cory Benfield, eds. “HTTP/2.” RFC 9113. Internet Engineering Task Force, June 2022. <https://www.rfc-editor.org/rfc/rfc9113>

Thomson, Martin, and Sean Turner, eds. “Using TLS to Secure QUIC.” RFC 9001. Internet Engineering Task Force, May 2021. <https://www.rfc-editor.org/rfc/rfc9001>

Thomson, Susan, Christian Huitema, Vladimir Ksinant and Mohsen Souissi. “DNS Extensions to Support IP Version 6.” RFC 3596. Internet Engineering Task Force, October 2003. <https://www.rfc-editor.org/rfc/rfc3596>

WHATWG. “Fetch Standard.” Living Standard. Accessed 5 September 2026. <https://fetch.spec.whatwg.org/>

WHATWG. “HTML Standard.” Living Standard. Accessed 5 September 2026. <https://html.spec.whatwg.org/>

WHATWG. “URL Standard.” Living Standard. Accessed 5 September 2026. <https://url.spec.whatwg.org/>

World Wide Web Consortium. “Content Security Policy Level 3.” W3C Specification. Accessed 5 September 2026. <https://www.w3.org/TR/CSP3/>

World Wide Web Consortium. “Service Workers.” W3C Specification. Accessed 5 September 2026. <https://www.w3.org/TR/service-workers/>

Historical and institutional sources

CERN. “The Birth of the Web.” CERN historical timeline and archive. Accessed 5 September 2026. <https://timeline.web.cern.ch/timeline-header/90>

Chromium Blog. “An Update on the Lock Icon.” 2 May 2023. <https://blog.chromium.org/2023/05/an-update-on-lock-icon.html>

Cloudflare. “Understanding how Facebook disappeared from the Internet.” 4 October 2021. <https://blog.cloudflare.com/october-2021-facebook-outage/>

Engineering at Meta. “More details about the October 4 outage.” 5 October 2021.

<https://engineering.fb.com/2021/10/05/networking-traffic/outage-details/>

Internet Society. “A Brief History of the Internet.” Participant history by Barry M. Leiner and colleagues. Accessed 5 September 2026.

<https://www.internetsociety.org/internet/history-internet/brief-history-internet/>

Kleinrock, Leonard. “The Day the Infant Internet Uttered its First Words.” UCLA IMP log reproduction. Accessed 5 September 2026.

https://www.lk.cs.ucla.edu/internet_first_words.html

World Wide Web Consortium. “Tim Berners-Lee: Weaving the Web.” Publication and author notes. Accessed 5 September 2026.

<https://www.w3.org/People/Berners-Lee/Weaving/>

Modern works

Abbate, Janet. *Inventing the Internet*. MIT Press, 1999.

Berners-Lee, Tim, with Mark Fischetti. *Weaving the Web: The Original Design and Ultimate Destiny of the World Wide Web by Its Inventor*. HarperSanFrancisco, 1999.

Blum, Andrew. *Tubes: A Journey to the Center of the Internet*. Ecco, 2012.

Kurose, James F., and Keith W. Ross. *Computer Networking: A Top-Down Approach*. 9th ed. Pearson, 2025. Copyright 2026.