

How the Internet Works

Volume 3: Naming, Discovery, and Data Transmission

Generated: September 11, 2025

A Complete Technical Guide

From Physical Infrastructure to Future Technologies

Table of Contents

1. Naming and Discovery Systems
2. DNS Resolution Process
3. DNS Security
4. Service Discovery Protocols
5. Data Transmission Mechanics

Naming and Discovery Systems

The internet would be nearly unusable without naming systems. Imagine having to remember 142.250.217.78 instead of google.com, or needing to manually configure the IP address of every device on your network. Naming and discovery systems provide the abstraction layer that makes the internet accessible to humans while enabling automatic configuration.

The Domain Name System (DNS) Architecture

DNS is one of the internet's most critical services, translating billions of domain names to IP addresses every second. It's a globally distributed, hierarchical database with no single point of failure. The system is designed to be fast (responses in milliseconds), scalable (handling the entire internet), and resilient (continuing to function despite failures).

The genius of DNS lies in its hierarchical delegation model. Rather than one massive database, authority is delegated down a tree structure. The root servers know who's responsible for .com, the .com servers know who's responsible for google.com, and Google's servers know the IP address for www.google.com. This delegation distributes both the administrative burden and the technical load.

The DNS Hierarchy:

The DNS namespace is organized like an inverted tree with the root at the top. Each node in the tree represents a domain, and the path from a node to the root represents the full domain name. This structure allows for distributed administration - each organization manages its own portion of the namespace.

The Root Zone is the top of the DNS hierarchy, containing information about all TLDs. There are 13 root server addresses (A through M root), but actually approximately 1,500 servers via anycast. They handle 1+ trillion queries per year with a root zone file of only about 2MB that lists all TLDs. Managed by IANA and operated by 12 organizations, changes require DNSSEC signing ceremonies.

Top-Level Domains (TLDs) are the first level below root:

Generic TLDs (gTLDs):

- Original: .com, .org, .net, .edu, .gov, .mil
- Sponsored: .aero, .museum, .coop
- New gTLDs: 1,200+ added since 2013 (.app, .dev, .xyz)
- Restricted: .bank, .pharmacy (strict requirements)

Country-Code TLDs (ccTLDs):

- Two-letter ISO codes: .uk, .de, .jp
- Some repurposed: .tv (Tuvalu), .io (British Indian Ocean)
- National sovereignty over policies

Infrastructure TLD: .arpa for technical infrastructure

Second-Level Domains are what organizations register (google.com, wikipedia.org, gov.uk). The registry maintains authoritative records, registrars sell domains to the public, and WHOIS database contains ownership info.

Subdomains are subdivisions of domains (www.google.com, mail.google.com). They can have multiple levels (server1.datacenter.east.company.com) with a maximum of 127 levels and 253 characters total.

DNS Servers and Their Roles:

Different types of DNS servers work together to resolve queries. Each type has specific responsibilities and typically stores different portions of DNS data. Understanding these roles helps diagnose DNS issues and optimize performance.

Recursive Resolvers do the heavy lifting. Usually operated by ISPs or public services (8.8.8.8, 1.1.1.1), they query multiple servers to find answers, cache responses for performance, handle DNSSEC validation, maintain typical caches of millions of entries, and achieve cache hit rates of 70-90%.

Authoritative Servers are the source of truth. They host actual DNS records for domains, don't perform recursion, and include:

- Primary (master): Holds original zone file
- Secondary (slave): Copies from primary
- Hidden primary: Not listed in NS records

Forwarding Servers pass queries along. Corporate DNS servers often forward to ISPs, can apply filtering/security policies, and reduce external DNS traffic.

DNS Resolution Process

DNS resolution is the process of translating a domain name into an IP address. While conceptually simple, the actual process involves multiple servers, caching layers, and fallback mechanisms. A typical resolution takes 10-100 milliseconds, though cached responses return instantly.

The Resolution Journey

When you type a URL into your browser, a complex chain of events occurs. The browser first checks its own cache, then the operating system cache, then queries a recursive resolver. If the resolver doesn't have the answer cached, it begins a recursive query process, working its way down from the root servers.

Local Caching Layers: Multiple caches to check

- Browser cache: Per-browser, shortest TTL respect
- OS resolver cache: System-wide, honors TTL
- Local DNS server: Router or local resolver
- ISP recursive resolver: Largest cache

Recursive Resolution: When not cached

1. Query root server: "Who handles .com?"
2. Response: "Ask a.gtld-servers.net"
3. Query .com server: "Who handles google.com?"
4. Response: "Ask ns1.google.com"
5. Query Google's NS: "What's the IP for www.google.com?"
6. Response: "142.250.217.78"

Caching and TTL: Reducing future queries

- Each record has TTL (Time To Live)
- Common TTLs: 5 minutes to 24 hours
- Shorter TTL: More flexibility, more queries
- Longer TTL: Better performance, harder to change
- Negative caching: Remember non-existent domains

DNS Query Types and Flags

DNS queries contain various flags and options that control how resolution occurs. Understanding these helps troubleshoot DNS issues and optimize configurations.

Query flags:

- RD (Recursion Desired): Client wants recursive query
- RA (Recursion Available): Server offers recursion
- AA (Authoritative Answer): Response from authoritative server
- TC (Truncated): Response too large for UDP
- AD (Authenticated Data): DNSSEC validated
- CD (Checking Disabled): Don't validate DNSSEC

Query types:

- Standard query: Normal resolution
- Inverse query: IP to name (deprecated)
- Status: Server status request
- Notify: Primary tells secondaries of changes
- Update: Dynamic DNS updates
- AXFR/IXFR: Zone transfers

DNS Performance Optimizations

Modern DNS employs numerous optimizations to improve performance and reduce load. These range from simple caching to sophisticated geographic routing systems.

Anycast: Same IP in multiple locations

- Root servers use extensively
- Queries route to nearest instance
- Automatic failover if location fails
- Reduces latency and improves resilience

GeoDNS: Location-aware responses

- Different IPs for different regions
- CDNs return nearest edge server
- Based on client IP or EDNS Client Subnet

DNS Prefetching: Browsers resolve ahead of time

- Resolve links on current page
- Predictive resolution based on typing
- Can save 100-300ms on navigation

DNS Pipelining: Multiple queries on one connection

- TCP connection reuse
- Reduces connection overhead
- Important for DNS over TLS/HTTPS

DNS Security

DNS was designed in a more trusting era and originally had no security mechanisms. This made it vulnerable to various attacks. Modern DNS security additions like DNSSEC and encrypted transport protect against many threats, though adoption remains incomplete.

DNS Vulnerabilities and Attacks

Understanding DNS vulnerabilities helps appreciate the importance of DNS security measures. Many major internet outages and security breaches have involved DNS manipulation.

Cache poisoning: Inserting false records

- Attacker races legitimate response

- Spoofs source IP and guesses query ID
- Redirects traffic to attacker's servers
- Kaminsky attack: Poisoning entire domains

DNS hijacking: Changing DNS settings

- Malware changes local DNS settings
- Router compromise affects entire network
- Registrar compromise changes authoritative servers

DDoS attacks: Overwhelming DNS servers

- Direct attacks on DNS infrastructure
- Amplification: Small query triggers large response
- Reflection: Response sent to victim
- NXDomain attacks: Queries for non-existent domains

DNS tunneling: Using DNS for data exfiltration

- Encodes data in DNS queries/responses
- Bypasses firewalls that allow DNS
- Used by malware and attackers

DNSSEC - DNS Security Extensions

DNSSEC adds cryptographic signatures to DNS records, allowing resolvers to verify that responses haven't been tampered with. It creates a chain of trust from the root zone down to individual records.

How DNSSEC works:

- Each zone signs its records with private key
- Public key published in DNS (DNSKEY record)
- Parent zone signs hash of child's key (DS record)
- Creates chain of trust to root

DNSSEC records:

- RRSIG: Signature for record set
- DNSKEY: Public key for zone

- DS: Delegation signer (hash of child's key)
- NSEC/NSEC3: Authenticated denial of existence

Challenges:

- Larger responses may require TCP
- Complex key management
- Clock synchronization required
- Only ~30% of domains signed
- Breaks some middleboxes and filters

Encrypted DNS

Traditional DNS queries are sent in plaintext, allowing eavesdropping and manipulation. Encrypted DNS protocols protect privacy and integrity of DNS queries.

DNS over TLS (DoT):

- DNS queries over TLS connection
- Port 853 (dedicated port)
- Clear protocol, identifiable as DNS
- Supported by Android, systemd-resolved

DNS over HTTPS (DoH):

- DNS queries inside HTTPS
- Port 443 (same as web traffic)
- Harder to identify and block
- Supported by browsers, controversial with ISPs

Privacy implications:

- Hides queries from local network
- Shifts trust to DoH/DoT provider
- Bypasses local filtering/monitoring
- Debate over centralization

Service Discovery Protocols

While DNS handles internet-scale naming, local networks need additional protocols for devices to discover each other and available services. These protocols enable zero-configuration networking where devices "just work" when connected.

DHCP - Dynamic Host Configuration Protocol

DHCP automatically configures network settings for devices, eliminating manual configuration. It's why you can connect to a network and immediately start browsing without entering IP addresses, subnet masks, or DNS servers.

DHCP process (DORA):

1. Discover: Client broadcasts request for DHCP server
2. Offer: Server offers IP address and settings
3. Request: Client requests offered configuration
4. Acknowledge: Server confirms assignment

Information provided:

- IP address and subnet mask
- Default gateway (router)
- DNS servers
- Domain name
- NTP servers
- Custom options (VOIP servers, etc.)

Lease management:

- Addresses leased, not permanently assigned
- Typical lease: 1-24 hours
- Client renews at 50% of lease time
- Rebind at 87.5% if renewal fails

- Prevents address exhaustion

DHCP relay: For multiple subnets

- Relay agent forwards between subnets
- Allows central DHCP server
- Adds giaddr (gateway IP address) field

mDNS - Multicast DNS

mDNS allows devices to resolve names on local networks without a DNS server. It's how your computer finds your printer by name or how you connect to computer.local addresses.

How it works:

- Uses .local domain
- Multicast address: 224.0.0.251 (IPv4), ff02::fb (IPv6)
- Port 5353
- Each device responds for its own name
- No central server required

Conflict resolution:

- Device probes before claiming name
- If conflict, adds number (computer-2.local)
- Continuous conflict detection

Used by:

- Apple Bonjour
- Avahi (Linux)
- Windows 10/11
- IoT devices

Service Discovery

Beyond finding devices, networks need ways to discover what services are available. Modern protocols allow automatic discovery of printers, file shares, media servers, and other services.

DNS-SD (DNS Service Discovery):

- Uses DNS records to advertise services
- Works with mDNS or regular DNS
- SRV records specify service locations
- TXT records provide service metadata

SSDP (Simple Service Discovery Protocol):

- Part of UPnP
- HTTP-like protocol over multicast
- Used by media devices, routers

WS-Discovery: Web services discovery

- SOAP-based protocol
- Used by Windows for network discovery
- Printers, file shares

Data Transmission Mechanics

Data transmission is the core function of the internet - moving bits from one computer to another across a complex, unreliable network. This involves breaking data into packets, finding paths through the network, detecting and correcting errors, and managing congestion. The mechanics of how this works are surprisingly complex yet elegantly designed.

Packet Switching Fundamentals

The internet uses packet switching rather than circuit switching (like traditional phone networks) for fundamental reasons of efficiency and resilience. In circuit switching, a dedicated path is established for the entire communication session. In packet switching, data is broken into packets that find their own paths through the network. This means multiple communications can share the same infrastructure, and the network can route around failures.

Packet switching was a radical idea when proposed in the 1960s. The conventional wisdom was that reliable communication required dedicated circuits. The insight that you could build reliable communication over unreliable networks by breaking data into packets and letting them find their own way was revolutionary. This approach makes the internet possible - efficient, scalable, and resilient.

Anatomy of a Packet:

Every packet is like a self-contained envelope with addressing information and data. As packets traverse the network, different layers add and remove their headers, like nested envelopes. Understanding packet structure is essential for network troubleshooting and security analysis.

Ethernet Frame (Layer 2): The local network envelope

- Preamble: 7 bytes of alternating 1s and 0s for synchronization
- Start Frame Delimiter: 1 byte marking frame start
- Destination MAC: 6 bytes hardware address
- Source MAC: 6 bytes hardware address
- EtherType/Length: 2 bytes indicating payload type
- Payload: 46-1500 bytes (the IP packet)
- Frame Check Sequence: 4 bytes CRC for error detection
- Total: 64-1518 bytes typically
- Jumbo frames: Up to 9000 bytes in data centers

IP Packet (Layer 3): The internet envelope

- IPv4 header: 20-60 bytes with fixed fields (Version, length, TTL, addresses) and options (Source routing, timestamps - rarely used)
- IPv6 header: Fixed 40 bytes, simplified compared to IPv4, with extension headers for options
- Payload: Transport layer segment

TCP Segment (Layer 4): The reliability envelope

- Header: 20-60 bytes
- Sequence/acknowledgment numbers
- Window size for flow control
- Flags: SYN, ACK, FIN, RST, PSH, URG
- Options: Timestamps, window scaling, SACK

- Data: Application payload

Fragmentation and Path MTU:

Networks have maximum packet sizes (MTU - Maximum Transmission Unit), typically 1500 bytes for Ethernet. When packets need to traverse networks with smaller MTUs, they must be fragmented. This adds overhead and can cause problems, so modern networks try to avoid fragmentation.

IPv4 fragmentation: Routers can fragment

- Fragments identified by same ID field
- Fragment offset indicates position
- More Fragments flag indicates continuation
- Destination reassembles
- Problems: Some fragments lost = entire packet lost

IPv6 approach: Only source fragments

- Routers send "Packet Too Big" messages
- Source adjusts packet size
- More efficient, less prone to problems

Path MTU Discovery: Finding optimal packet size

- Send packets with Don't Fragment flag
- Receive ICMP error if too large
- Binary search to find maximum
- Cached per destination
- Black hole detection when ICMP blocked