# DNS Is 40 Years Old and Still Running the Internet. Something Has to Change.

by Nick Clark | Published March 27, 2026 | PDF

DNS was designed for a static, hierarchically administered internet and has governed name resolution for over forty years without structural change. Its assumptions about global consistency, centralized root authority, and propagation-based updates cannot support edge-native, decentralized, and AI-driven systems. Replacing DNS requires a fundamentally different resolution architecture built on local authority, governed mutation, and structural adaptation.

---

DNS was specified in 1983. The internet it was designed for had a few hundred hosts. The directory fit in a text file called HOSTS.TXT, maintained by a single organization and distributed manually.

That file became a protocol. The protocol became infrastructure. The infrastructure became, effectively, permanent.

Today DNS handles hundreds of billions of queries per day across every device, application, and service connected to the internet. It is the reason you can type a name and reach a machine. It is also a 40-year-old system designed for a world where one organization could reasonably know the name of every computer on the network.

That world ended around 1985.

## What DNS actually does, and doesn't do

DNS resolves names to addresses. You ask for a name; it returns a number. That's the job.

The architecture that performs this job is hierarchical in shape but centralized in authority. At the top is a root zone, maintained by ICANN and served by 13 root server clusters. Below that are the TLD registries: Verisign for .com, various national registries for country codes. Below that are the authoritative nameservers for each domain, usually operated by whoever registered it or by a DNS hosting provider.

Resolution traverses this hierarchy top-down. A resolver asks the root where to find .com, asks the .com registry where to find example.com, asks example.com's nameserver for the actual record. The answer is cached for a period defined by the TTL, then discarded and refetched.

This works. It has worked, at scale, for four decades. The engineering is genuinely impressive.

It also has structural properties that were acceptable in 1983 and are now load-bearing constraints.

**Names are static.** A DNS record is a mapping: name points to address. The record can change, but change requires propagating a new record through the cache hierarchy, waiting for TTLs to expire, hoping resolvers respect them. The namespace does not evolve; it is updated.

**Authority is fixed and hierarchical.** The authority for a name is determined by who controls the zone containing it. You cannot hold authority for a name without controlling the zone. Zones are defined by the hierarchy. The hierarchy is defined by the root. There is no path to local authority that doesn't trace back to ICANN.

**The system does not govern itself.** DNS has no native mechanism for a segment of the namespace to define its own mutation policy, manage its own cache coordination, or make structural decisions about its own scope. Those decisions happen outside the protocol, through registrars and zone administrators, and propagate downward.

## Why this matters now

For most of the web, DNS's limitations are invisible. You register a domain, point it at a server, it works. The TTL delay when you change records is annoying but manageable.

The constraints become structural in a few specific contexts.

**Edge and distributed systems.** A CDN or edge platform distributes execution globally but still resolves names through DNS. Naming authority remains central even when compute is not. The edge node closest to your user knows where to serve the request from, but it doesn't govern what the request's name means or when that meaning can change. That authority still lives upstream, in a zone controlled by a registrar or a platform's control plane.

**Jurisdictional fragmentation.** DNS has no native concept of a name that means different things in different jurisdictions, or a namespace segment that can implement region-specific policy without forking the zone. Jurisdictional variations are handled through application logic layered on top, not through the resolution architecture itself.

**Dynamic and ephemeral resources.** Microservices, serverless functions, AI agents, IoT devices: these are resources that are created, mutated, and retired at a rate that DNS's TTL-based consistency model was not designed for. Workarounds exist: low TTLs, service meshes, sidecar proxies. They all involve working around DNS rather than through it.

**Identity and persistence.** DNS names are owned, not identity-bearing. The name example.com has no structural relationship to what it names, no continuity through mutation, no provenance. If the registrar is seized, the domain transferred, or the TTL expired, the name means something different or nothing at all.

## What replacing it actually requires

DNS has survived this long partly because it is genuinely good at its original job and partly because the cost of replacing foundational internet infrastructure is enormous. Every device, every application, every library has DNS resolution baked in. HTTPS depends on it. Email depends on it. The web depends on it.

Replacement doesn't mean ripping it out. It means building a resolution architecture with different structural properties that can coexist with DNS and eventually subsume the cases where DNS's constraints are the binding constraint.

The structural properties that matter:

**Local authority.** A segment of the namespace should be governable by the nodes responsible for that segment, under locally held policy. The authority for a name should be traceable to the scope that contains it, not to a hierarchy whose root is a single organization.

**Governed mutation.** Changes to a name's meaning, the structure of a namespace, or the validity of a cached mapping should be subject to a validation process defined by the governing scope. Not a TTL countdown and a zone file update. Mutation should be proposable, evaluable, and approvable through a defined process that produces an auditable lineage.

**Structural adaptation.** A segment of the namespace that grows beyond its governance capacity should be able to split into child scopes. A segment that becomes inactive should be able to merge. These structural decisions should be made by the nodes governing the segment, not by a zone administrator working outside the resolution protocol.

**Alias continuity.** A name should remain resolvable through structural changes. If the scope containing a name splits, the name should trace through the new structure. If a resource is relocated, the alias should redirect through the new scope. Continuity through mutation is a structural property, not a post-hoc redirect.

These four properties describe a resolution architecture that does not currently exist as deployed infrastructure. DNS does not have them. ENS and other blockchain-based naming systems are experiments with some of them, but introduce global ledger constraints that reproduce a different form of centralization. Existing service mesh and discovery solutions are scoped to private infrastructure and don't address global resolution.

## The indexing layer

The structural approach that satisfies these properties is an adaptive, anchor-governed index: a hierarchical namespace where each segment is governed by the nodes responsible for it, mutations are validated through scoped consensus, resolution traverses the hierarchy stepwise through local anchor groups, and structural changes preserve lineage continuity.

This is not a routing protocol. It is an indexing and resolution layer that sits above transport, making names meaningful, governable, and persistent across the full lifecycle of the resources they identify.

The reason DNS has lasted 40 years is that it solved a genuinely hard coordination problem at a scale nobody had reached before. The reason something has to change is that the coordination problem has changed. The internet no longer has a few hundred hosts with stable addresses. It has billions of resources: ephemeral, distributed, jurisdictionally complex, AI-mediated. Their naming requirements are not what DNS was built for.

The replacement will not announce itself. It will appear as an indexing layer that handles the cases DNS cannot, coexist quietly with the cases DNS still handles fine, and expand from there.

Adaptive Indexing All 21 steps →

Resolution without global consensus. Anchor-governed self-organization.

Patent
US 19/326,036 · published
Primary Technical Disclosure
○ The Adaptive Index: A Scalable Foundation for Decentralized Systems
Secondary Technical
○ Anchor-Governed Hierarchical Nesting: Recursive Semantic Containers at Unlimited Depth○ Entropy-Triggered Index Splitting: Deterministic Partitioning Under Mutation Load○ Dormant Index Merging: Recursive Consolidation of Low-Entropy Subindices○ Elastic Anchor Group Management: Governance That Scales With Criticality○ Trust-Weighted Quorum Voting: Consensus Where Weight Reflects Earned Trust○ Asynchronous Consensus Coordination: Offline Vote Completion With Reconciliation○ Best-Match Alias Querying: Longest-Match Resolution With Stepwise Delegation○ Action-Typed Aliases: Behavioral Intent Embedded in the Namespace○ UID Persistence Through Alias Mutation: Stable Identity Across Structural Change○ Lineage-Preserving Structural Mutation: Cryptographic History Through Every Change○ Proximity-Based Routing With Trust Scoring: Dynamic Path Selection in Decentralized Networks○ Dynamic Device Hash for Pseudonymous Authentication: Volatile Identity Without Stored Credentials○ On-Demand Adaptive Caching: Cache Instances That Follow Usage, Not Configuration○ Predictive Cache Prefetching: Forecasting Models That Proactively Instantiate Caches○ Contextual Access Enforcement: Policy Graphs Evaluated With Real-Time Telemetry○ Mutation Router With Contextual Signals: Policy-Aware Propagation Path Selection○ Impact Simulation During Mutation Staging: Pre-Execution Analysis of Proposed Changes○ DNS Bidirectional Fallback: Hybrid Resolution With Legacy DNS Compatibility○ Asset Versioning as First-Class Metadata: Version Entries Under UIDs With Lineage Tracking○ Telemetry-Driven Topology Mutation: Autonomous Network Reconfiguration From Operational Data
Applications (General)
○ Applying Adaptive Indexes to Legacy Decentralized Systems○ Why Edge Platforms Still Depend on a Central Authority○ Supply Chain Tracking Through Governed Namespace Resolution○ Social Media Platforms Without Central Namespace Authority○ Healthcare Data Federation Through Scoped Governance○ Government Identity Infrastructure at Scale○ Financial Market Data With Governed Resolution○ Gaming and Metaverse Namespace Governance
Applications (Specific)
○ Cloudflare's Edge Has a Namespace Problem● DNS Is 40 Years Old and Still Running the Internet○ ENS Solved the Wrong Half of the Naming Problem○ Handshake Decentralized the Root. Everything Below It Is Still Ungoverned.○ IPFS Solved Content Addressing. It Didn't Solve Naming, Persistence, or Governance.○ Fastly Built the Fastest Cache Invalidation in the Industry. The Authority to Invalidate Still Lives in One Place.○ Akamai Built the Internet's Delivery Infrastructure. It Was Designed for a World That Needed Central Control.○ Bluesky Identified the Right Problem. The Architecture That Solves It Is the Adaptive Index.○ Consul's Service Catalog Is Brilliant Infrastructure. It Is Still a Central Registry.○ Istio Solved Programmable Traffic Policy. The Namespace That Routes Traffic Is Still Central.○ Unstoppable Domains Proved NFT Ownership Works. The Namespace Governance Model Is Still Unresolved.○ The Graph Built the Index Layer for Web3. The Index Itself Still Has a Governance Problem.○ Filecoin Proved Verifiable Storage. Discovery and Namespace Governance Are Still Unsolved.○ Arweave Made Data Permanent. It Has No Governance Model for What Permanent Data Means Over Time.○ Ceramic Built Mutable Data Streams for Web3. The Governance of Those Streams Is Still Not Local.○ Kubernetes Service Discovery Resolves Within Clusters. Cross-Cluster Namespace Is Central.○ Amazon Route 53 Is the Most Reliable DNS on Earth. It Is Still DNS Architecture.○ HashiCorp Nomad Distributes Scheduling. The Namespace That Organizes It Is Still Central.○ ZooKeeper Coordinates Distributed Systems. The Coordinator Is a Single Point of Authority.○ etcd Stores the State of Kubernetes. The State Store Has No Scoped Governance.○ Consul KV Distributes Configuration. The Distribution Authority Is Still Central.○ Raft Made Consensus Understandable. It Did Not Make Consensus Scope-Aware.○ Paxos Proved Consensus Is Possible. It Did Not Address Namespace Governance.○ Cosmos Tendermint Enabled Sovereign Blockchains. The Namespace Between Them Is Ungoverned.○ AWS Cloud Map Discovers Services. The Discovery Authority Lives in One Region's Control Plane.○ Azure Traffic Manager Routes Globally. The Routing Authority Is Centrally Defined.○ GCP Service Directory Centralizes Service Registration. Registration Is Not Governance.○ Netlify DNS Simplifies Deployment Routing. The Namespace Authority Is Still Netlify's.○ Vercel's Edge Network Executes at the Boundary. Routing Authority Does Not.○ Bunny CDN Delivers Content Globally. Cache Governance Is Still Central.○ KeyCDN Optimized Content Delivery. The Delivery Namespace Is Centrally Controlled.○ Limelight Networks Built Private Infrastructure for Delivery. The Namespace Governance Is Still Central.○ StackPath Combined CDN With Edge Computing. Namespace Authority Remained Central.○ Envoy Proxy Made Service Mesh Data Planes Programmable. The Control Plane Still Governs.○ NGINX Powers the Web's Reverse Proxy Layer. Its Configuration Is Statically Defined.○ Traefik Discovers Services Automatically. The Discovery Namespace Is Still External.○ Linkerd Simplified the Service Mesh. The Namespace It Meshes Is Still Kubernetes.○ Namecheap Made Domain Registration Accessible. Domain Governance Remains the Registrar Model.○

GoDaddy Registered More Domains Than Anyone. The Namespace Model Has Not Changed.○ DNSimple Made DNS Management Developer-Friendly. The Governance Model Is Still DNS.○ Datadog Observes Everything. The Namespace It Observes Has No Governed Structure.○ Grafana Unified Observability Visualization. The Data Namespace It Queries Has No Governed Structure.○ Prometheus Defined Cloud-Native Monitoring. Its Metric Namespace Has No Governance Layer.○ New Relic Pioneered APM. The Telemetry Namespace It Built Is Centrally Indexed.○ Splunk Indexes Machine Data at Scale. The Index Namespace Is Centrally Administered.
Adaptive Indexing overview →
AQ
deterministic
autonomy

Legal

Last updated: 2026-03-03

- 
- nick@qu3ry.net
- 72 28 14 36 01

Invented by Nick Clark | Founding Investors: Devin Wilkie