Nick Clark · Published July 2, 2026 · PDF

# How to Build a Decentralized Alternative to DNS

If you are building infrastructure that needs to turn human-readable names into machine locations without a delegated root and a chain of registrars, you are looking for a DNS alternative. This guide describes an architectural approach where names are semantic aliases resolved stepwise by the local groups that own each scope, mapped to stable underlying identifiers rather than to fixed addresses. It is an architecture disclosed in United States Patent Application 19/326,036, not a shipping library, and it centers on the Adaptive Indexing inventive step.

---

## What You Are Building

You want a naming system that does what DNS does, turn a name a human can read into something a machine can reach, but without a single delegated root, without a hierarchy of registrars who own the branches, and without a name being permanently welded to one location. You want names that keep resolving when a service moves, when a network partitions, or when the group governing a branch changes membership, and you want the authority to hand out names to live with the people running each part of the system, not with a global operator everyone must trust.

This guide teaches an architecture for that resolution layer. The core idea is to treat a name not as a fixed pointer to an address but as a semantic alias resolved one segment at a time by the group that owns that segment's scope, and to bind every alias to a stable underlying identifier so renaming, moving, or restructuring never breaks a reference. This approach is disclosed in United States Patent Application 19/326,036 as part of the Adaptive Indexing inventive step. You implement it yourself; there is no package to install.

## Why the Obvious Approaches Fall Short

Three approaches are commonly reached for, and each has a real, structural gap. None is a straw man; they all work, within limits.

**DNS itself, or a DNS-shaped clone.** The Domain Name System resolves names through a hierarchy of delegated zones: a root delegates to top-level domains, which delegate to registrars and zone operators, and a resolver walks that chain. It is effective and battle-tested. The structural property to notice is that authority flows down from a delegated root and mappings are essentially static bindings changed through zone updates, so a name's continuity depends on that chain staying intact and cooperative.

**Blockchain naming.** Put the name-to-value mapping in a globally replicated ledger so no single registrar controls it. This genuinely removes the central operator and gives strong auditability. The cost is that every mutation is a global coordination event against replicated state, exactly the rigidity, globally replicated state and monolithic consensus, that makes structural evolution a network-wide problem rather than a local one.

**Content-addressed identifiers.** Name things by the hash of their content or by a public key, as distributed file and identity systems do. This eliminates naming coordination because the identifier is derived, not assigned. But the identifier is not

human-readable and is tied to a specific content state, so you bolt a human-readable, mutable name on top, which puts you back at the original problem.

The common thread: each binds the name directly to a delegation chain, to global replicated state, or to a fixed content hash, and inherits that binding's rigidity. The architecture below decouples the name from all three.

## The Architecture

The disclosed approach organizes names in an adaptive index and resolves every alias stepwise through locally governed groups, binding each alias to a stable underlying identifier. Several mechanisms do the work, each tracing to the filed specification.

**1. Names are structured aliases in a parent-child index.** The index is a plurality of entries organized in a parent-child hierarchy, where each entry corresponds to a unique semantic scope identified by a structured alias. The spec's alias form is `[top-level domain]@[domain].[subdomain]/[subindices]/[asset]`, for example `article@org.wikipedia/articles/article123`. This looks DNS-like on purpose, but the hierarchy is a nesting of semantic scopes, not a chain of delegation from a root you must trust.

**2. Resolution is stepwise and scope-local.** Alias resolution runs stepwise using anchor-local logic at each level; each segment is interpreted relative to its parent scope. The spec's worked example: resolving `org@wikipedia` first matches the `org` top-level entry, then `w`, then proceeds through `wiki` to reach `wikipedia`, each anchor group resolving its own segment and delegating downward. Queries resolve by best-match, the longest-matching entry within a namespace. No global root need be reachable; resolution is a walk through independently governed scopes.

**3. Anchors govern each scope, and they are not the data hosts.** Each entry is governed by one or more anchors, units that perform two roles: caching and scoped voting. Anchors maintain index metadata, permissions, and lineage references, but they are not data hosts; actual storage and delivery is performed by participating nodes. This mirrors the useful part of DNS, separating the name lookup from the thing the name points at, but here the anchor for a scope is a local governance group, not a delegated zone server.

**4. Aliases resolve to stable underlying identifiers.** Each alias resolves to a unique identifier (UID) that remains stable even as the alias is renamed, delegated, or restructured. The spec's concrete case: renaming `user@elizabeth` to `user@liz` does not break links or references, because applications, permissions, and data bindings persist through the UID. Assets carry a persistent, anchor-verifiable identifier that stays fixed even as they move between index paths, change alias bindings, or migrate across infrastructure. This is the decisive break from a name being a fixed pointer: the name is a mutable label over a durable identifier.

**5. Structural change is governed by local quorum, not global consensus.** When a resolution or mutation request targets a scope, only that scope's active anchors form a quorum to validate it, under a pre-registered policy defining the quorum threshold (the spec's worked examples include 3-of-4 and 4-of-6). No unrelated anchor group participates and no network-wide finality is invoked. Scopes can split when overloaded (the spec's example: `wikipedia` splitting into `wikipedia/a–m` and `wikipedia/n–z`, each governed by new anchor sets) or merge when dormant, through scoped voting rather than a global registry update.

**6. Lineage keeps names resolving across every structural change.** Because structural mutations preserve lineage metadata and anchor mappings, alias resolution stays continuous even after segmentation, merging, or relocation, with no global rebind. Each container records its structural lineage as an immutable traversal path, and when a container is split, merged, or relocated, its aliases are automatically

remapped to the successor using anchor-stored lineage metadata, executed at resolution time. This is what a delegation chain does not give you: a name survives its scope being reorganized underneath it.

**7. It reconciles asynchronously across partitions.** Anchors may accept proposals asynchronously and can operate under temporary partition, forming isolated quorums and completing votes offline. On reconnection, signed vote records are reconciled against the canonical ledger for that scope using policy-defined arbitration. The spec calls out fragmented and high-latency environments explicitly. A partitioned segment keeps resolving names locally and reconciles on rejoin rather than going dark.

Two further properties matter for a DNS replacement. The spec describes a hybrid, backward-compatible model: an alias that fails to resolve within the network may fall back to a corresponding legacy `.org`, `.com`, or other domain, with bidirectional DNS bridging for alias continuity across traditional and anchor-scoped domains. And the design is explicitly retrofittable, meant to overlay existing decentralized systems by introducing anchors and aliases without altering their core protocols.

## How to Approach the Build

You are implementing several cooperating pieces. Order matters, because early decisions constrain later ones.

**Step 1: Define the identifier split first.** Decide your UID scheme before anything else. Every named thing, service, endpoint, asset, gets a persistent identifier that is never shown to humans, never reused, and never referenced directly by your application layer. The alias is a separate, mutable record mapping to that UID. This is the single most important decision; if consumers of resolution bind to names instead of UIDs, you lose the continuity property that motivates the design.

**Step 2: Model the namespace as nested scopes, not a flat zone file.** Represent names as a parent-child hierarchy of scopes, each with a structured alias, resolved segment by segment. An illustrative entry, faithful to the spec's model and labeled as a sketch, not shipping code:

```
// Illustrative only, not a library
Entry {
  alias:    "wikipedia"        // segment, resolved relative to parent
  scope:    "wiki"             // parent scope
  uid:      "uid:9f3a...",     // stable, never shown, never reused
  anchors:  [anchorId, ...],   // governing group for THIS scope
  lineage:  [ /* signed mutation records */ ]
}
```

Resolution walks the hierarchy by longest-match within each namespace, each segment resolved by the anchors of its parent scope. Do not build a global lookup table; build a stepwise resolver.

**Step 3: Separate the index from the hosts.** Anchors resolve names, permissions, and lineage; nodes store and serve the content. Return from resolution a UID plus, if requested, a set of candidate host nodes, rather than a single fixed address. This is what lets a name point at a moving, replicated target instead of one server.

**Step 4: Implement anchor groups and scoped quorum.** For each scope, define a governing anchor group and a policy object stating the quorum threshold and who may vote. Registering, re-binding, or retiring an alias is a mutation proposal evaluated only by that scope's anchors and committed when the policy's quorum approves. Keep the threshold a policy parameter, not a constant, the spec notes different operation sensitivities warrant different quorums (a routine directory update low, a rekey up to full participation). Never let a proposal in one scope require votes from another.

**Step 5: Make every mutation append lineage.** On each approved change, append a signed record capturing the prior state, the approving quorum composition, and the justification. Do not overwrite or garbage-collect these. Your split, merge, and relocate operations then remap aliases to their successor by walking this lineage, so existing references keep resolving with no rebind.

**Step 6: Implement asynchronous reconciliation.** Allow a scope's anchors to accept and validate proposals while partitioned, holding signed votes locally, then reconcile against the scope's canonical ledger under a defined arbitration policy on reconnection. Write this arbitration policy deliberately and test it against a real partition; it is the place a naive implementation silently corrupts state.

**Step 7: Design the legacy bridge from the start.** If you need to coexist with DNS during adoption, build in the fallback the spec describes, an alias that misses inside the network falls back to a legacy domain lookup, with bidirectional bridging so references cross cleanly. Retrofitting this later is harder than designing it in.

## What This Does Not Give You

Be clear-eyed about the boundaries.

This is an architecture, not a drop-in library. There is no SDK to install and nothing "just works" out of the box. You implement every piece yourself, including the genuinely hard parts, the reconciliation arbitration and quorum policy design above all.

It is disclosed in a patent filing. It is not presented here as a benchmarked or production-proven system, and this guide states no performance numbers, because the specification states none for you to rely on. Treat latency, throughput, and scaling as things you must measure in your own build.

It does not remove governance; it relocates it. Someone still decides who owns a name within a scope, now a local anchor group under a policy you design. Weak policies or a scope stacked with colluding anchors produce bad outcomes. The architecture makes authority local and auditable; it does not make it disappear, nor does it defend against a scope's own governance group misbehaving beyond what your policy encodes.

It does not fit every problem. If you want one flat global namespace with a single authoritative root, DNS already does that well. This approach earns its complexity specifically when you need decentralized control, location-independent names, structural reorganization without breaking references, and partition healing together.

## Disclosure Scope

The architecture described in this guide, semantic hierarchical aliases resolved stepwise by scope-owning anchors to stable underlying identifiers, anchor-scoped local consensus, asynchronous reconciliation on reconnection, alias stability across rename, split, merge, and relocation, and optional legacy DNS bridging, is disclosed in United States Patent Application 19/326,036. This guide is educational: it explains an approach a developer can study and implement themselves. It is not a warranty, a specification of a shipping product, or an offer of software, and it does not grant any license under that application. Every mechanism described above is grounded in the filed disclosure; where the specification is silent, this guide makes no claim.

---


**Adaptive Indexing (/adaptive-indexing)** All 40 steps → (/inventive-steps)

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

U.S. 19/326,036 (/patents/19-326036)

## PRIMARY TECHNICAL DISCLOSURE

- The Adaptive Index: A Scalable Foundation for Decentralized Systems (/articles/the-adaptive-index-a-scalable-foundation-for-decentralized-systems)

## SECONDARY TECHNICAL

- Anchor-Governed Hierarchical Nesting: Recursive Semantic Containers at Unlimited Depth (/articles/adaptive-indexing/anchor-nesting)
- Entropy-Triggered Index Splitting: Deterministic Partitioning Under Mutation Load (/articles/adaptive-indexing/entropy-splitting)
- Dormant Index Merging: Recursive Consolidation of Low-Entropy Subindices (/articles/adaptive-indexing/dormant-merging)
- Elastic Anchor Group Management: Governance That Scales With Criticality (/articles/adaptive-indexing/elastic-anchors)
- Trust-Weighted Quorum Voting: Consensus Where Weight Reflects Earned Trust (/articles/adaptive-indexing/trust-weighted-voting)
- Asynchronous Consensus Coordination: Offline Vote Completion With Reconciliation (/articles/adaptive-indexing/async-consensus)
- Best-Match Alias Querying: Longest-Match Resolution With Stepwise Delegation (/articles/adaptive-indexing/best-match-aliases)
- Action-Typed Aliases: Behavioral Intent Embedded in the Namespace (/articles/adaptive-indexing/action-typed-aliases)
- UID Persistence Through Alias Mutation: Stable Identity Across Structural Change (/articles/adaptive-indexing/uid-persistence)
- Lineage-Preserving Structural Mutation: Cryptographic History Through Every Change (/articles/adaptive-indexing/lineage-preserving-mutation)
- Proximity-Based Routing With Trust Scoring: Dynamic Path Selection in Decentralized Networks (/articles/adaptive-indexing/proximity-routing)
- Dynamic Device Hash for Pseudonymous Authentication: Volatile Identity Without Stored Credentials (/articles/adaptive-indexing/device-hash-auth)
- On-Demand Adaptive Caching: Cache Instances That Follow Usage, Not Configuration (/articles/adaptive-indexing/adaptive-caching)
- Predictive Cache Prefetching: Forecasting Models That Proactively Instantiate Caches (/articles/adaptive-indexing/predictive-prefetching)
- Contextual Access Enforcement: Policy Graphs Evaluated With Real-Time Telemetry (/articles/adaptive-indexing/contextual-access)
- Mutation Router With Contextual Signals: Policy-Aware Propagation Path Selection (/articles/adaptive-indexing/mutation-routing)

- Impact Simulation During Mutation Staging: Pre-Execution Analysis of Proposed Changes (/articles/adaptive-indexing/impact-simulation)
- DNS Bidirectional Fallback: Hybrid Resolution With Legacy DNS Compatibility (/articles/adaptive-indexing/dns-fallback)
- Asset Versioning as First-Class Metadata: Version Entries Under UIDs With Lineage Tracking (/articles/adaptive-indexing/asset-versioning)
- Telemetry-Driven Topology Mutation: Autonomous Network Reconfiguration From Operational Data (/articles/adaptive-indexing/telemetry-topology)
- The Index Is the Territory: The Navigable Substrate Beneath Both Axes (/articles/adaptive-indexing/the-index-is-the-territory)

## APPLICATIONS · GENERAL

- Decentralized AI Agent and Model Federation Without a Central Registry: Adaptive Indexing for Cross-Organization Discovery and Addressing (/articles/adaptive-indexing/decentralized-ai-federation)
- Payload-Aware Edge Caching and Live Retransmission: Replacing Address-Based CDN Heuristics With Adaptive Indexing (/articles/adaptive-indexing/cdn-and-live-media)
- How to Retrofit Adaptive Indexing onto Legacy Decentralized Systems (Web3, Fediverse, DAOs) (/articles/adaptive-indexing/applying-to-legacy-systems)
- Why Edge Platforms Still Depend on a Central Authority (/articles/adaptive-indexing/why-edge-platforms-depend-on-central-authority)
- Supply Chain Tracking Through Governed Namespace Resolution (/articles/adaptive-indexing/supply-chain-provenance)
- Social Media Platforms Without Central Namespace Authority (/articles/adaptive-indexing/decentralized-social)
- Healthcare Data Federation Through Scoped Governance (/articles/adaptive-indexing/healthcare-data-federation)
- Sovereign Government Digital Identity Without a Central Registry (/articles/adaptive-indexing/government-identity-infrastructure)
- Governed Securities Identifier Resolution for Financial Market Data (/articles/adaptive-indexing/financial-market-data)
- Cross-Platform Gaming and Metaverse Namespace Governance for Portable Player Identity and Assets (/articles/adaptive-indexing/gaming-metaverse-namespace)
- IoT Device-Fleet Identity and Telemetry Without a Central Registry: Adaptive Indexing for Pseudonymous, Revocable Device Naming (/articles/adaptive-indexing/iot-device-fleet-identity)
- Coordinating Autonomous Vehicles at the Edge Without a Central Server: Adaptive Indexing for V2V and V2I (/articles/adaptive-indexing/autonomous-vehicle-edge-coordination)

- Coordinating Smart Grids and Islanding Microgrids Without a Central Controller Using Adaptive Indexing (/articles/adaptive-indexing/smart-grid-microgrid-coordination)
- Delay-Tolerant and Interplanetary Networking: Resolving Names and Governing State Across Variable-Latency, Intermittently-Connected Links (/articles/adaptive-indexing/delay-tolerant-interplanetary-networking)

## APPLICATIONS · SPECIFIC

- Cloudflare Workers Alternative: Governed Namespace Beyond the Central Control Plane (/articles/adaptive-indexing/cloudflare)
- DNS vs. Adaptive Indexing: which holds namespace authority locally? (/articles/adaptive-indexing/dns)
- ENS vs. anchor-governed adaptive indexing: who governs namespace mutation? (/articles/adaptive-indexing/ens)
- Handshake vs Governed Namespace: Who Governs Below the Root? (/articles/adaptive-indexing/handshake)
- IPFS vs Adaptive Indexing: Content Addressing Without Governed, Mutable Naming (/articles/adaptive-indexing/ipfs)
- Fastly Alternative for Governed Edge Caching: Distributed Purge Speed vs Distributed Cache Authority (/articles/adaptive-indexing/fastly)
- Akamai Property Manager vs Anchor-Governed Edge Namespaces: Where Should Configuration Authority Live? (/articles/adaptive-indexing/akamai)
- Bluesky PLC directory vs. adaptive indexing: how do you decentralize did:plc resolution? (/articles/adaptive-indexing/bluesky)
- HashiCorp Consul vs. Adaptive Indexing: Does a Raft-Backed Service Catalog Govern Namespace Structure? (/articles/adaptive-indexing/consul)
- Istio Solved Programmable Traffic Policy. The Namespace That Routes Traffic Is Still Central. (/articles/adaptive-indexing/istio)
- Unstoppable Domains Alternative for Governed Namespace Mutation: Adaptive Indexing (/articles/adaptive-indexing/unstoppable-domains)
- The Graph vs Governed Indexing: Who Holds Authority Over the Index Structure Itself (/articles/adaptive-indexing/the-graph)
- Filecoin Proved Verifiable Storage. Discovery and Namespace Governance Are Still Unsolved. (/articles/adaptive-indexing/filecoin)
- Arweave Made Data Permanent. It Has No Governance Model for How the Namespace of Permanent Data Evolves. (/articles/adaptive-indexing/arweave)
- Ceramic vs Adaptive Indexing: Mutable Data Streams Without Governed Namespace Authority (/articles/adaptive-indexing/ceramic)

- Does Kubernetes Govern Cross-Cluster Namespaces Without a Central Control Plane? (/articles/adaptive-indexing/kubernetes)
- Amazon Route 53 vs. Anchor-Governed Namespace Authority: Reliability or Governance? (/articles/adaptive-indexing/amazon-route53)
- HashiCorp Nomad Alternative for Governed Namespaces: Distributed Scheduling, Central Namespace (/articles/adaptive-indexing/hashicorp-nomad)
- ZooKeeper Coordinates Distributed Systems. The Coordinator Is a Single Point of Authority. (/articles/adaptive-indexing/zookeeper)
- etcd Stores the State of Kubernetes. The State Store Has No Scoped Governance. (/articles/adaptive-indexing/etcd)
- Consul KV Distributes Configuration. The Distribution Authority Is Still Central. (/articles/adaptive-indexing/consul-kv)
- Raft vs Scope-Governed Consensus: A Governed Alternative to Single-Log Replication (/articles/adaptive-indexing/raft-protocol)
- Paxos vs Scope-Governed Adaptive Indexing: Consensus Without Namespace Governance (/articles/adaptive-indexing/paxos)
- Cosmos and Tendermint Alternative for Cross-Chain Namespace: Governed Adaptive Indexing (/articles/adaptive-indexing/cosmos-tendermint)
- AWS Cloud Map vs. Adaptive Indexing: Who Governs the Namespace? (/articles/adaptive-indexing/aws-service-discovery)
- Azure Traffic Manager Routes Globally. The Routing Authority Is Centrally Defined. (/articles/adaptive-indexing/azure-traffic-manager)
- GCP Service Directory Centralizes Service Registration. Registration Is Not Governance. (/articles/adaptive-indexing/gcp-service-directory)
- Netlify DNS Simplifies Deployment Routing. The Namespace Authority Is Still Netlify's. (/articles/adaptive-indexing/netlify-dns)
- Vercel Edge Alternative: Distributed Execution vs Deployer-Governed Routing Authority (/articles/adaptive-indexing/vercel-edge)
- Bunny CDN Alternative: Adaptive Indexing and Governed Edge Cache Resolution (/articles/adaptive-indexing/bunny-cdn)
- KeyCDN Optimized Content Delivery. The Delivery Namespace Is Centrally Controlled. (/articles/adaptive-indexing/keycdn)
- Limelight Networks Built Private Infrastructure for Delivery. The Namespace Governance Is Still Central. (/articles/adaptive-indexing/limelight)
- StackPath Alternative for Governed Edge: Unified Edge Services vs Distributed Namespace Authority (/articles/adaptive-indexing/stackpath)

- Envoy Proxy Made Service Mesh Data Planes Programmable. The Control Plane Still Governs. (/articles/adaptive-indexing/envoy-proxy)
- NGINX Powers the Web's Reverse Proxy Layer. Its Configuration Is Statically Defined. (/articles/adaptive-indexing/nginx)
- Traefik Alternative for Governed Routing: Beyond Provider-Derived Service Discovery (/articles/adaptive-indexing/traefik)
- Linkerd Alternative for Governed Namespaces: Service Mesh Beyond the Kubernetes Registry (/articles/adaptive-indexing/linkerd)
- Namecheap Made Domain Registration Accessible. Domain Governance Remains the Registrar Model. (/articles/adaptive-indexing/namecheap)
- GoDaddy Registered More Domains Than Anyone. The Namespace Model Has Not Changed. (/articles/adaptive-indexing/godaddy)
- DNSimple Made DNS Management Developer-Friendly. The Governance Model Is Still DNS. (/articles/adaptive-indexing/dnsimple)
- Datadog Alternative for Governed Namespaces: Observability vs Adaptive Indexing (/articles/adaptive-indexing/datadog)
- Grafana Alternative for Governed Observability: The Data Namespace It Queries Has No Governed Structure (/articles/adaptive-indexing/grafana)
- Prometheus vs Governed Namespace Indexing: The Metric Namespace Has No Adjudication Layer (/articles/adaptive-indexing/prometheus)
- New Relic Alternative: Governed Telemetry Namespace Beyond Centralized Indexing (/articles/adaptive-indexing/new-relic)
- Splunk Alternative for Governed Namespaces: Machine-Data Indexing vs Adaptive Indexing (/articles/adaptive-indexing/splunk)
- GitHub Copilot Workspace vs Governed Cross-Repository Resolution (/articles/adaptive-indexing/github-copilot-workspace)
- Tableau Pulse alternative for cross-authority analytics: governed adaptive indexing (/articles/adaptive-indexing/tableau-pulse)
- Notion AI vs Federated Anchor-Governed Retrieval (/articles/adaptive-indexing/notion-ai)
- Matrix (matrix.org / Element) alternative: adaptive semantic naming for federated identity and resolution (/articles/adaptive-indexing/matrix-protocol)
- BitTorrent Mainline DHT (Kademlia) vs adaptive indexing: semantic aliases and scoped governance over a content-hash lookup (/articles/adaptive-indexing/bittorrent-dht)
- Tailscale alternative: naming and resolution when the coordination plane is offline (/articles/adaptive-indexing/tailscale)

Adaptive Indexing overview → (/adaptive-indexing)