Nick Clark · Published April 25, 2026 · PDF

# How to Retrofit Decentralized Naming onto an Existing System

You have a working resolution system (DNS, an on-chain registry, a federated identity handle scheme) and you want portable, semantic, decentralized names on top of it without rewriting the protocol underneath. This guide describes an architectural approach for doing exactly that: an anchor-and-alias overlay that resolves names stepwise through scope-owning governance units and falls back to your existing resolver. The approach is disclosed in United States Patent Application 19/326,036 and its home inventive step is the Adaptive Indexing inventive step. This is an architecture you build, not a library you install.

---

## What You Are Building

You already run a naming or resolution system. It might be DNS, an on-chain contract registry, an IPFS content-hash scheme, or a federated handle system like the fediverse. It works, but it is rigid: names are bound tightly to locations or hosts, renaming breaks references, discoverability is poor, and governance is either fully centralized or requires network-wide coordination to change anything.

What you want is a naming layer that is decentralized, human-readable, semantically structured, and portable, so a name survives when the thing it points to moves, gets renamed, or migrates between hosts. And you want it without ripping out the resolver

you already depend on.

This guide describes an architecture for adding that layer as a structural overlay. The core idea, drawn from the filed disclosure, is that you introduce two new primitives on top of your existing system: anchors (scope-owning governance units) and aliases (structured, human-readable names). You do not replace your consensus layer, your contracts, or your DNS zone files. You augment them. The disclosure is explicit that anchors and aliases are meant to be introduced "without altering their core protocols or consensus layers," operating as a "structural augmentation rather than a disruptive replacement."

## Why the Obvious Approaches Fall Short

The usual ways to make names more portable each solve part of the problem and leave a structural gap.

A global registry (a single smart contract, a central directory, a canonical index) gives you one place to look up any name, but every structural change becomes a global coordination problem. The disclosure frames existing indexing mechanisms such as DNS, IPFS, and contract-based registries as relying on "static alias mappings, centralized delegation hierarchies, or cryptographic immutability," where mutation control is bolted on through "external wrappers, permissioned interfaces, or off-chain logic."

A pure content-hash scheme (as in IPFS or BitTorrent) gives you immutability and integrity, but the identifier changes when the content changes, so a hash is not a stable, human-meaningful name for an evolving thing. You end up maintaining a separate mutable pointer layer anyway.

A federated handle system distributes hosting, but identity fragments across servers and discoverability suffers: the same person is a different handle on every instance, and moving hosts breaks references.

The common gap is that names are governed either globally (one authority, one coordination bottleneck) or per-host (fragmented, non-portable), with nothing in between. What is missing is scoped local authority: the ability for an independently governed subtree of the namespace to evolve, split, delegate, or rekey on its own, while names still resolve continuously and reference integrity holds. That middle ground is what the disclosed architecture targets.

## The Architecture

The overlay has two primitives and one resolution rule. Everything below traces to the filed disclosure.

Anchors are the governance units. An anchor governs one entry in an index, and each entry corresponds to a "unique semantic scope." An anchor is not a data host. Per the disclosure, anchors "maintain index metadata, permissions, and lineage references," while actual storage and delivery stay with your existing nodes. Each anchor encodes three things: a mutation policy, an alias mapping, and access-control metadata. Anchors perform two roles within their scope: resolving aliases locally, and voting (in quorum with the other anchors governing the same scope) on structural changes.

Aliases are the names. An alias is structured and human-readable, taking a hierarchical form the disclosure gives as `[top-level domain]@[domain].[subdomain]/[subindices]/[asset]`, for example `article@org.wikipedia/articles/article123`. Crucially, each alias resolves to a stable unique identifier (UID) that "remains stable even as the alias itself is renamed, delegated, or restructured." Renaming `user@elizabeth` to `user@liz` does not break links, because the bindings hang off the UID, not the display name.

The index is a parent-child hierarchy of these scopes. In the disclosure's worked example, `w` contains `wiki` contains `wikipedia`, each governed by its own independent anchor group. Arrows between entries represent "scope-resolved delegation, not physical routing."

Resolution is stepwise and local. This is the central mechanism. Resolution is "performed stepwise, using anchor-local logic at each level of the hierarchy," where "each alias segment is interpreted relative to its parent scope." Resolving `org@wikipedia` matches `org`, then `w`, then delegates down through `wiki` to `wikipedia`, with each anchor group resolving only its own segment and handing off downward. Matching is longest-prefix ("best-match querying"): resolution starts by "identifying the longest-matching entry within a given namespace." There is no global consensus and no central authority in the resolution path.

Continuity across structural change is what makes it an overlay you can trust. When a scope is split, merged, or relocated, its aliases are "automatically remapped to the resulting container or its successor, using anchor-stored lineage metadata." Each container records "its structural lineage as a cryptographically immutable traversal path," so a name still resolves after mutation without any "global rebind." Anchors perform the redirection at resolution time.

The retrofit hook: DNS and legacy fallback. This is the specific mechanism that lets the overlay coexist with what you already run. The disclosure states that "legacy DNS lookups are still supported: if an alias fails to resolve within the network, it may fall back to a corresponding .org, .com, or other legacy domain." It further describes "bidirectional DNS bridging to support alias continuity across traditional and anchor-scoped domains," and aliases "compatible with legacy namespace hierarchies including Ethereum smart contracts, IPFS content hashes, and ActivityPub URIs." So an unresolved alias does not fail; it degrades to your existing resolver.

## How to Approach the Build

The following is an ordered path a developer would follow to implement this overlay themselves. The interface sketches are illustrative and faithful to the disclosure; they are not a package you can install.

1. Map your existing namespace to scopes. Decide what a "scope" is in your system. The disclosure's retrofit examples are concrete: in a DeFi protocol, "each contract namespace may become a parent node" with a path like `defi > uniswap > v3 > pools > eth-usdc`; in a DAO, "each governance domain" anchors separately (`dao > optimism > grants > round5 > proposal42`); on a federated social platform, a handle resolves to `users@bluesky > e > elizabeth`. Start by drawing your parent-child tree.
2. Define the alias grammar and the UID binding. Adopt the `tld@domain.subdomain/subindices/asset` shape or a variant that fits your domain. The non-negotiable design rule is that every alias binds to a stable UID, and all references, permissions, and data bindings attach to the UID rather than the display name. An illustrative anchor entry:

```
// illustrative, not a shipping API
AnchorEntry {
  scope:        "wiki"           // the segment this anchor owns
  uid:          "uid:9f3a...",   // stable identity, survives renames
  alias_map:    { "wikipedia": <child-scope-ref> },
  policy_ref:   "//wiki",        // mutation policy + quorum thresholds
  access_meta:  { ... },
  lineage:      [ <prior-state-hash>, ... ]
}
```

3. Implement stepwise resolution with best-match traversal. Write a resolver that walks the alias segment by segment. At each level it asks the anchor that owns the parent scope to resolve the next segment and delegate downward. Match longest-prefix first. Keep each step's logic anchor-local; do not introduce a global lookup.

```
// illustrative resolution loop
resolve(alias):
  scope = root
  for segment in split(alias):
    anchor = scope.anchor_group
    next   = anchor.resolve_local(segment)   // longest-match, delegates d
    if next is None:
      return legacy_fallback(alias)           // e.g. DNS .org/.com bridge
    scope = next
  return scope.uid
```

4. Wire the legacy fallback path first. Before you have any anchors at all, the fallback is what keeps the system usable during adoption. Any alias that does not resolve in the overlay should route to your existing resolver (DNS, the on-chain registry, the ActivityPub URI). Per the disclosure this "hybrid model enables smooth adoption while advancing toward a fully decentralized naming foundation." Building fallback first means you can deploy the overlay incrementally, scope by scope.
5. Attach mutation policy and quorum to each anchor group. A scope's structural changes (splitting an overloaded entry, merging a dormant one, relocating) are decided by quorum among the anchors governing that scope, under a pre-registered policy object. The disclosure's example policy `//wiki` "defines quorum thresholds, governance logic, and anchor admission criteria," with concrete thresholds like 3-of-4 or 4-of-6. Sensitive operations can demand higher quorums; the disclosure notes rekey operations may require 100 percent participation. Only anchors in the affected scope participate, so no unrelated part of the network is involved.

6. Record lineage on every mutation. Each approved structural change must append a lineage entry to the container's metadata capturing the mutation type, the quorum composition, and the previous state, cryptographically committed. This is what lets a renamed or relocated name still resolve later: the resolver reconstructs container ancestry through these records. Do not garbage-collect them; the disclosure describes them as append-only and continuously validated.
7. Add the adaptive behaviors only once the core resolves correctly. Splitting and merging based on load ("entropy heuristics derived from mutation telemetry, access frequency"), trust-weighted voting, proximity-aware node selection, and predictive caching are all disclosed, but they are refinements on top of a correct stepwise resolver plus lineage. Get resolution and continuity right first.

## What This Does Not Give You

This is an architecture, not a drop-in library. There is no package to `npm install` and no SDK being shipped here. You implement the anchors, the alias grammar, the resolver, the quorum logic, and the lineage store yourself, against your own substrate. The disclosure is deliberately substrate-agnostic (it can run on "containerized microservices, edge devices, embedded processors, or resource-constrained mesh nodes"), which means it prescribes structure, not code.

It is not benchmarked or productized. The disclosure describes mechanisms and design intent. It does not state performance numbers, and nothing here should be read as a throughput, latency, or availability guarantee. Any such figures would have to come from your own implementation and measurement.

It does not remove the need for your existing system. The overlay is explicitly a fallback-preserving augmentation. If your names are already fully portable and your governance model already supports scoped local authority, the overlay may add complexity without a matching payoff. It is aimed at systems where names are location-bound, governance is a coordination bottleneck, or identity fragments across hosts.

It does not by itself decide your policies. Quorum thresholds, trust-weighting functions, admission criteria, split and merge heuristics, and access rules are all things you must define per scope. The architecture gives you where those decisions live (in anchor policy objects), not what they should be for your domain.

## Disclosure Scope

The anchor-and-alias overlay architecture described in this guide, including stepwise anchor-local resolution, UID-stable aliases, lineage-preserving structural mutation, quorum-scoped governance, and legacy DNS or protocol fallback, is disclosed in United States Patent Application 19/326,036. This guide is educational: it explains an architectural approach a developer can build themselves and is faithful to that filing. It is not a warranty, a benchmark, a claim of a shipping product, or an offer of software, and it does not grant any license. Third-party technologies named for context (DNS, IPFS, Ethereum, ActivityPub, and federated platforms) are referenced only to situate the approach and are described as they generally work.

---


## 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)