Nick Clark · Published July 2, 2026 · PDF

How to Keep Identifiers Stable Across a Network Split

If you build distributed systems, you have hit the case where a network partition splits your cluster and identifiers drift, collide, or dangle when the halves reconnect. This guide teaches an architectural approach that keeps a resource's identifier stable across rename, split, merge, relocate, and partition, by separating the name a client uses from the underlying identifier it resolves to. The approach is disclosed in United States Patent Application 19/326,036, and it is an architecture you implement, not a shipping library. It rests on the Adaptive Indexing inventive step.

What You Are Building

You are building a naming and resolution layer in which the identifier a resource carries survives events that normally break references: a rename, a structural reorganization of your namespace, a physical relocation of the data, and, hardest of all, a network split in which two halves of your system keep operating without seeing each other and then have to reconcile when the link returns.

The search-intent problem is specific. During a partition, both sides stay live. Both sides may accept writes, register new resources, and reorganize their local view of the namespace. When the partition heals, you need the identifiers issued on each side to still mean the same thing they meant before, to not have silently collided, and to

reconcile without a global rebind that forces every client to update its references. This is the problem faced by anyone building federated social infrastructure, edge and mesh deployments, offline-first mobile sync, DAOs, or any decentralized index where global coordination is either undesirable or impossible.

The architecture described here comes from a filed patent disclosure, United States Patent Application 19/326,036. It is a design you build against your own substrate, not a package you install.

Why the Obvious Approaches Fall Short

The common approaches each break in a predictable place under partition.

Central registry (DNS-style delegation). A hierarchical authority like DNS gives you stable, human-readable names by delegating zones to authoritative servers. It is accurate and battle-tested for its purpose. Its structural assumption, though, is that the authority for a name is reachable to resolve or change it. Under a partition that isolates the authority, the isolated side cannot make authoritative changes to that zone; it can only serve cached records until they expire. Renames and reorganizations are a coordination event against the authority, not a local operation.

Content-addressed identifiers (hash-of-content). Systems that name a resource by the hash of its bytes get partition-tolerant, collision-resistant identifiers for free, which is why content addressing is a good fit for immutable data. The tradeoff is deliberate: the identifier is the content, so any mutation produces a new identifier. That is exactly what you do not want when the whole point is a stable identifier that outlives edits, renames, and relocations.

Global consensus (single replicated log). Putting every structural change through one globally ordered log gives you a clean, collision-free namespace. But global consensus needs a quorum of the whole system. A partition that denies quorum to a

side stalls structural changes on that side entirely, trading availability for consistency at precisely the moment you wanted the isolated side to keep working.

The structural gap is shared: each approach couples the identifier's stability to the reachability of a global or authoritative coordinator. Remove reachability, and you lose either the ability to change the namespace or the ability to keep the identifier fixed.

The Architecture

The disclosed approach closes that gap with a few reinforcing ideas. Every mechanism below traces to the filed specification.

Separate the name from the identifier. The spec's central move is that each alias resolves to a unique identifier (UID) that remains stable even as the alias is renamed, delegated, or restructured. Renaming `user@elizabeth` to `user@liz` does not break links, because applications, permissions, and data bindings are keyed to the UID, not the alias. An asset's underlying identifier is described as globally unique, anchor-verifiable, and resistant to alias churn: the asset may move between index paths, change alias bindings, or migrate across physical infrastructure while the underlying identifier stays fixed. This is the property you are after, stated directly.

Resolve stepwise through scope-owning anchors. Names are structured hierarchically, for example `article@org.wikipedia/articles/article123`. Resolution is performed stepwise, one segment at a time, using anchor-local logic at each level. Each alias segment is interpreted relative to its parent scope. An anchor is the governance unit for its entry: it resolves aliases in its scope, validates mutation proposals, and participates in restructuring, and it does so only within its jurisdictional boundary. There is no requirement for a global resolver.

Make consensus local to a scope. Structural changes to a scope are decided by that scope's anchor group, operating as a quorum under a policy attached to the anchor (the spec gives examples like a 3-of-4 quorum). The spec is explicit that only anchors governing the affected index scope participate; no coordination is required from unrelated anchor groups and no global finality condition is invoked. This is the property that lets a partitioned side keep working: authority over a scope travels with the anchors that hold it, not with the whole network.

Preserve lineage through structural mutation. When a container is split, merged, or relocated, its aliases are automatically remapped to the resulting container or successor using anchor-stored lineage metadata. Each approved mutation appends a lineage record (previous anchor map, justification, quorum configuration at ratification), cryptographically committed alongside the container's metadata. Because structural mutations preserve lineage metadata and anchor mappings, alias resolution stays continuous after segmentation, merging, or relocation, with no global rebind: each alias trace recursively maps through preserved anchor-scoped identifiers by reconstructing container ancestry. This is what makes split and merge non-breaking rather than reference-destroying events.

Accept votes asynchronously and reconcile on reconnection. This is the partition story. Anchors may accept mutation proposals asynchronously; proposals are cached and validated later when quorum is available, giving eventual consistency and delayed reconciliation without interrupting the resolution path. Anchors may operate under temporary partition and complete mutation votes offline; on reconnection, their signed vote records are reconciled against the canonical ledger for that scope. Anchor groups can form isolated quorums during disconnection, validate mutations, and stay responsive, and upon reconnection mutation lineage is reconciled using policy-defined arbitration. The spec frames this as suitable for fragmented or high-latency environments, and even names self-stabilization after prolonged network partitions as an intended behavior.

Put together: a client holds a stable UID; it reaches that UID by resolving an alias stepwise through scope-owning anchors; each scope governs its own structure locally so a partition does not freeze it; and the lineage records let each side's structural changes reconcile against the scope's canonical ledger when the link returns, without rebinding references held by clients.

How to Approach the Build

You implement this yourself over your substrate. A workable order:

1. Mint stable UIDs and key everything to them. Assign every resource a persistent identifier at creation and store permissions, versions, and bindings against that UID rather than against any name. The spec keeps anchor-held metadata (permissions, version pointers) keyed to this identifier so relocation does not disrupt visibility or governance. Design UID minting to be safe under partition: neither side can consult a global counter, so use a scheme (for example, an anchor-scoped prefix plus a locally unique suffix) that cannot collide across scopes without inter-anchor coordination.

2. Model the namespace as scoped, nested containers. Each entry corresponds to a unique semantic scope identified by a structured alias and may contain nested subentries. Define your alias grammar (the spec uses `[tld]@[domain].[subdomain]/[subindices]/[asset]`) and resolve longest-match first, one segment per scope.

3. Give each scope an anchor group and a policy. An illustrative, spec-faithful sketch of what an anchor holds:

```
# ILLUSTRATIVE ONLY — not a shipping API; faithful to the disclosure
Anchor {
  scope:        "wiki"            # the segment this group governs
  members:      [a4, a7]          # current anchor map
  policy_ref:   "//wiki"          # quorum thresholds, admission, roles
  lineage_log:  [ ... ]           # append-only, signed mutation records
}
```

The policy defines quorum thresholds (the spec varies these by operation sensitivity, e.g. 2-of-3 for a content directory change, up to 100% for a policy rekey), admission and retirement rules, and which anchors are eligible to vote.

4. Make structural mutations lineage-preserving. Implement split, merge, and relocate so that each records a lineage entry (mutation type, quorum composition, previous container state, and a delta linking to the prior alias chain) before it takes effect. Never garbage-collect these records; resolution of an older alias reconstructs ancestry through them.

5. Build asynchronous, partition-aware voting. Let an anchor accept a signed mutation proposal even without live quorum, cache it, and ratify when quorum is reachable. Under partition, allow an isolated anchor group to form a local quorum and keep serving. Represent each vote as a signed record so it can be replayed later.

6. Reconcile on reconnection with policy-defined arbitration. When the link returns, replay each side's signed vote records and lineage entries against the scope's canonical ledger and apply the policy's arbitration rules to converge. Because clients hold UIDs and aliases resolve through preserved lineage, reconciliation converges the structure without asking clients to rebind.

7. Keep legacy fallback. The spec supports falling back to legacy DNS when an alias fails to resolve in-network, and describes retrofitting the anchor-and-alias layer over existing systems without altering their core protocols. Treat this as an overlay you add,

not a rewrite.

What This Does Not Give You

Be clear-eyed about scope. This is an architecture disclosed in a patent filing, not a drop-in library, an SDK, or a benchmarked product. There is no package to install and nothing here "just works" out of the box; you design the UID scheme, the anchor group membership protocol, the quorum and policy engine, the lineage log format, and the reconciliation arbitration yourself, and you are responsible for their correctness.

The disclosure describes mechanisms and their intended behavior; it does not supply performance numbers, and this guide invents none. Partition tolerance here is a design property of local consensus plus lineage-based reconciliation, not a proof: your reconciliation arbitration rules determine what happens when the two sides made genuinely conflicting structural decisions, and getting that policy right is the hard part of the build. The approach also assumes your problem actually wants a mutable, human-meaningful name over a stable identifier. If your data is immutable and you are happy naming it by its content hash, plain content addressing is simpler and you do not need this. If you require a single global total order over every structural change, that is the consistency-over-availability tradeoff this architecture deliberately declines.

Disclosure Scope

The architecture described in this guide is disclosed in United States Patent Application 19/326,036. The mechanisms attributed to the disclosure here (alias-to-UID decoupling, stepwise anchor-scoped resolution, anchor-local quorum governance, lineage-preserving structural mutation, and asynchronous consensus with reconciliation on reconnection) trace to that filing. This guide is educational: it teaches an approach a developer can implement independently. It is not a warranty, a specification of any product, or an offer of software, and nothing in it should be read as a claim that a benchmarked or production implementation is being distributed.

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)