Nick Clark · Published April 25, 2026 · PDF

How to Model Contagion in a Financial or Transportation Network

If you need to predict how a failure at one node spreads to the rest of a financial or transportation network, and to actually contain it rather than just chart it, the hard part is not the graph math. It is governing how state and authority travel hop to hop so that predictions and mitigations stay trustworthy across owners. This guide describes an architecture disclosed in U.S. Provisional Application No. 64/049,409, not a shipping library. The approach is the Cascade Propagation inventive step, which treats cascade modeling as a governed primitive rather than a central simulation.

What You Are Building

You are building a system that answers three questions when a disruption hits one point of a network: where will it spread, how hard and how soon will each downstream point be affected, and what coordinated action should each affected party take. The network might be a payment or interbank exposure graph where a single counterparty default drains liquidity from its neighbors, or a transportation graph where a closed bridge, a grounded hub, or a stalled rail junction reroutes and overloads adjacent links.

The people who need this are risk teams at clearinghouses and banks, operators of ports, rail, road, and air networks, and anyone responsible for systemic risk across infrastructure they do not fully own. The recurring pain is not drawing the graph. It is

that the graph spans multiple owners with different authority, the disruption data arrives with uneven trust, and a prediction that cannot be acted on across those boundaries is just a dashboard. The Cascade Propagation inventive step, disclosed in U.S. Provisional Application No. 64/049,409, addresses exactly that boundary problem: it makes both the model and the response carry their governance chain hop to hop.

Why the Obvious Approaches Fall Short

The standard tools are real and useful, and this guide does not dismiss them. Power-grid analysis built on SCADA cascade tooling, agent-based traffic simulation, epidemic compartment models, and supply-chain disruption models all propagate an effect across a graph and produce a forecast. If you only need a forecast inside one organization, they may be enough.

Where they leave a structural gap is at ownership and action boundaries. The filed disclosure characterizes prior cascade-modeling architectures as operating on centrally maintained models with ad hoc trust assumptions, producing unstructured alerts or a central dashboard, and being narrowly scoped to a single cascade domain. Three consequences follow for real financial and transportation networks. First, a single central model has to be trusted by everyone, but no bank fully trusts another bank's exposure numbers and no port fully trusts a neighbor's berth status; there is no built-in notion of who is authorized to assert a given edge or observation. Second, the output is an alert, not a directive tied to an accountable authority, so when the prediction reaches a downstream operator there is no structured way for that operator to accept, adapt, or decline the recommended mitigation. Third, contagion in the real world jumps domains: a financial shock closes a facility, which disrupts transportation, which feeds back into supply and price. Single-domain simulators do not compose across those jumps.

The architecture below is aimed squarely at those three gaps.

The Architecture

The disclosure frames cascade propagation as a first-class architectural primitive whose job is the governance-preserving projection of an observed disruption at one region of a topology to other regions, producing governed coordination directives downstream that preempt, mitigate, halt, or otherwise coordinate the response. Concretely, the primitive is composed of the following elements, each drawn directly from the filed specification.

A governance-credentialed topology graph. Nodes represent points of a physical-world domain and edges represent propagation channels between them. Critically, the graph is maintained by one or more governance authorities with domain responsibility, not by a single central owner. In a financial network a node is a counterparty or a market and an edge is an exposure channel; in a transportation network a node is a hub or junction and an edge is a route or link. The point is that each edge and node carries the credential of the authority responsible for it.

A per-edge propagation function. This defines how a disruption at a source node projects to a connected node, with governance-policy-defined transit, attenuation, transformation, or amplification characteristics. This is where domain physics or economics lives: an edge may delay the effect (transit), dampen it (attenuation), convert it into a different quantity (transformation, for example a liquidity shock becoming a delivery failure), or make it worse (amplification, for example a fire sale).

A per-node aggregation function. This defines how multiple incoming contributions combine at a receiving node. A junction hit from two directions, or a bank losing several counterparties at once, is not the sum of independent shocks, and the aggregation function is where you encode that combination rule per governance policy.

A cascade-trigger ingest interface. This consumes governed disruption observations and maps them to originating cascade nodes. In the disclosure the disruption itself is detected, classified, and attributed by a separate primitive; the

cascade engine ingests those governed observations rather than raw sensor noise, so every trigger arrives with its authority credential and lineage already attached.

A cascade-computation engine. This executes the propagation function across the topology and produces, per node, predicted affected regions, magnitudes, and arrival times. That is the forecast, but produced as governed output rather than a screen.

A cross-domain cascade composition mechanism. This combines propagation across two or more topology domains to produce composite, cascade-of-cascade determinations. This is the element that lets a financial cascade drive a transportation cascade and back, through a shared architectural mechanism rather than by bolting two simulators together.

A cascade-authority resolution mechanism. When a topology spans multiple governance authorities, this resolves who is responsible. This is the direct answer to the multi-owner trust gap: the model does not require every owner to accept one central custodian, and it has an explicit rule for whose assertion governs where authorities overlap.

A preemptive-mitigation directive generator. This produces governed coordination directives routed to downstream receiving agents, so the output of the model is an actionable, credentialed instruction, not just an alert.

A cascade-halting and containment mechanism. This specifies governance-policy-defined stop-conditions under which propagation is actively interrupted. This is the containment lever: the conditions under which you cut an edge, isolate a node, or freeze a channel to stop the spread.

A refusal and upstream-coordination mechanism. This handles the case where a downstream agent cannot or should not apply a proposed mitigation. The refusal is itself a first-class governed observation, which lets upstream coordinators seek an alternative, solicit corroborating observations, or escalate to a higher authority. The

disclosure enumerates refusal reasons including evidential insufficiency, capability exceedance, cost-threshold, priority conflict, authority insufficiency, dispositional, safety-boundary, and composite. This is what turns a rejected instruction into governed coordination instead of silent failure.

A topology-learning and adaptive-refinement mechanism. This updates the topology graph, propagation functions, and aggregation functions from observed outcomes, including refusal outcomes, so the model improves as real cascades and real responses accumulate.

A cascade-lineage recording mechanism. This records every topology reference, propagation computation, directive emission, mitigation, halting event, refusal, and topology update in a governance-chain lineage field, giving you an auditable account of why each prediction and each action happened.

The connective tissue under all of this is a governed observation format. The disclosure specifies that observations carry, at minimum, an authority credential, a dynamic device hash, spatial and temporal references, a time-to-live, a payload, and lineage fields, and that admissibility is evaluated against a governance-configurable authority taxonomy with supersession semantics, so that a higher-authority assertion can supersede a conflicting lower-authority one. That is how state and authority travel together hop to hop instead of being stripped off at each boundary.

How to Approach the Build

You are implementing this yourself. The steps below are an ordered way to approach it, faithful to the disclosed primitive.

1. **Model your topology as a credentialed graph.** Enumerate nodes and edges for your domain, and for each, record the governance authority responsible. For an interbank graph that is the institution or regulator asserting the exposure; for a

transportation graph it is the operator or authority owning the link. Do not start from a single central model.

2. **Define the observation envelope.** Decide the concrete fields for a governed observation in your system. The disclosed minimum is an authority credential, a device or source hash, spatial and temporal references, a time-to-live, a payload, and lineage. An illustrative interface, faithful to the spec and not a shipping API, looks like this:

```
Observation {
  authority_credential   // who asserts this, per the authority taxonomy
  source_hash            // identity attestation of the emitter
  spatial_ref            // which node/region
  temporal_ref           // when
  ttl                    // staleness bound
  payload                // the disruption or state
  lineage[]              // provenance chain, appended hop to hop
}
```

3. **Encode per-edge propagation and per-node aggregation as policy.** Write the transit, attenuation, transformation, and amplification behavior for each edge class, and the combination rule for each node class, as governance-policy-defined functions you can change per deployment rather than hardcoded constants.
4. **Wire the trigger ingest to governed disruption input.** Have the cascade engine consume already-attributed disruption observations and map them to origin nodes. Keep detection and attribution upstream of the cascade engine so triggers arrive credentialed.
5. **Run the computation engine to produce per-node forecasts.** Emit affected region, magnitude, and arrival time per node as governed observations, not as a dashboard row, so downstream consumers can admit them under their own policy.

6. **Generate mitigation directives and define stop-conditions.** For high-magnitude predicted nodes, generate credentialed coordination directives, and define the governance-policy stop-conditions under which the halting mechanism actively interrupts propagation.
7. **Implement refusal as a first-class path.** Give every downstream agent the ability to decline a directive with a classified reason and emit that refusal as a governed observation upstream, then have upstream coordinators respond by proposing alternatives, soliciting corroboration, or escalating. Do not let a rejected mitigation fail silently.
8. **Close the loop with lineage and learning.** Record every computation, directive, mitigation, halt, and refusal in lineage, and feed observed outcomes back into topology and function refinement.
9. **Compose domains only after each stands alone.** Once your financial and transportation topologies each run, connect them through the cross-domain composition mechanism so a shock in one becomes a credentialed trigger in the other.

What This Does Not Give You

This is an architecture, not a drop-in library. There is no package to install and nothing here "just works." You implement every element yourself: the topology store, the propagation and aggregation functions for your domain, the credential and admissibility machinery, the directive routing, and the refusal handling. The pseudocode above is illustrative only.

The disclosure does not provide, and this guide does not claim, benchmarked accuracy, latency numbers, or a validated economic or physical model of any specific network. The per-edge propagation functions are yours to calibrate and validate; a governance framework that faithfully carries authority hop to hop does not by itself make your contagion physics correct. The approach also presumes you can actually establish

governance authorities and credentials across the parties in your topology; where a single owner controls the whole network and trusts its own central model, most of the value here, which is the multi-authority and refusal machinery, does not apply and a conventional simulator may serve you better. Nothing here is a shipping product, a benchmark result, or a production guarantee.

Disclosure Scope

The Cascade Propagation approach described here is disclosed in U.S. Provisional Application No. 64/049,409. This guide is educational: it explains the disclosed architecture so a skilled developer can understand and build an implementation. It is not a warranty, a product offering, or an offer of software, and it does not grant any license. Every statement about how the approach works traces to that filing; where the filing is silent, this guide does not add mechanisms, parameters, or performance claims.

Cascade Propagation (/cascade-propagation)

All 40 steps → (/inventive-steps)

Refusal as first-class observation. Topology that learns from every event.

Provisional application

PRIMARY TECHNICAL DISCLOSURE

- Cascade Propagation: Refusal as First-Class Observation (/articles/cascade-propagation-refusal-as-first-class-observation)

SECONDARY TECHNICAL

- Credentialed Topology Graph (/articles/cascade-propagation/credentialed-topology-graph)
- Refusal as a First-Class Governed Observation (/articles/cascade-propagation/refusal-as-observation)
- Upstream Cascade Coordination (/articles/cascade-propagation/upstream-coordination)

- Cross-Domain Cascade Composition (/articles/cascade-propagation/cross-domain-cascade-composition)
- Preemptive Cascade Mitigation (/articles/cascade-propagation/preemptive-mitigation)
- Cascade Halting Mechanisms (/articles/cascade-propagation/cascade-halting)
- Multi-Authority Cascade Resolution (/articles/cascade-propagation/multi-authority-resolution)
- Topology Learning From Operations (/articles/cascade-propagation/topology-learning)

APPLICATIONS · GENERAL

- Preventing Cross-Operator Cascade Failures in Communication Networks (/articles/cascade-propagation/communication-network-cascade)
- Preventing Power Grid Cascade Failures: Credentialed Topology and Cross-Utility Resilience (/articles/cascade-propagation/power-grid-cascade-resilience)
- Supply Chain Cascade Management (/articles/cascade-propagation/supply-chain-cascade-management)
- Coordinating IT-to-OT Cyber-Physical Cascades Across Critical-Infrastructure Sectors (/articles/cascade-propagation/cyber-physical-cascade)
- Financial System Cascade Risk Management (/articles/cascade-propagation/financial-system-cascade)
- Cross-Modal Transportation Cascade Coordination Across Aviation, Rail, Motor-Carrier, and Maritime Networks (/articles/cascade-propagation/transportation-network-cascade)
- NERC CIP Cross-Utility Cascade Containment: A Grid Cybersecurity Architecture (/articles/cascade-propagation/nerc-cip-grid)

APPLICATIONS · SPECIFIC

- GE Vernova Grid Software vs Governed Cross-Utility Cascade Propagation (/articles/cascade-propagation/ge-grid-cascade)
- Hitachi Energy Grid vs Governed Cross-Utility Cascade Propagation (/articles/cascade-propagation/hitachi-energy-grid)
- Governed Cross-Utility Cascade Handling Beyond Siemens Grid Software (/articles/cascade-propagation/siemens-grid-software)
- ABB Grid Software Alternative: Credentialed Cross-Domain Cascade Propagation (/articles/cascade-propagation/abb-grid-software)
- Emerson Ovation vs Governed Cross-System Cascade Propagation (/articles/cascade-propagation/emerson-ovation)
- Schneider Electric EcoStruxure Grid vs Governed Cross-Domain Cascade (/articles/cascade-propagation/schneider-electric-grid)

- Form Energy Alternative: Governed Cascade Propagation for Long-Duration Storage (/articles/cascade-propagation/form-energy-storage)
- Honeywell Experion vs Governed Cross-System Cascade Propagation (/articles/cascade-propagation/honeywell-experion)
- Rockwell Automation FactoryTalk vs Governed Cross-Plant Cascade (/articles/cascade-propagation/rockwell-automation)
- Yokogawa CENTUM vs Governed Cascade Propagation (/articles/cascade-propagation/yokogawa-centum)

Cascade Propagation overview → (/cascade-propagation)