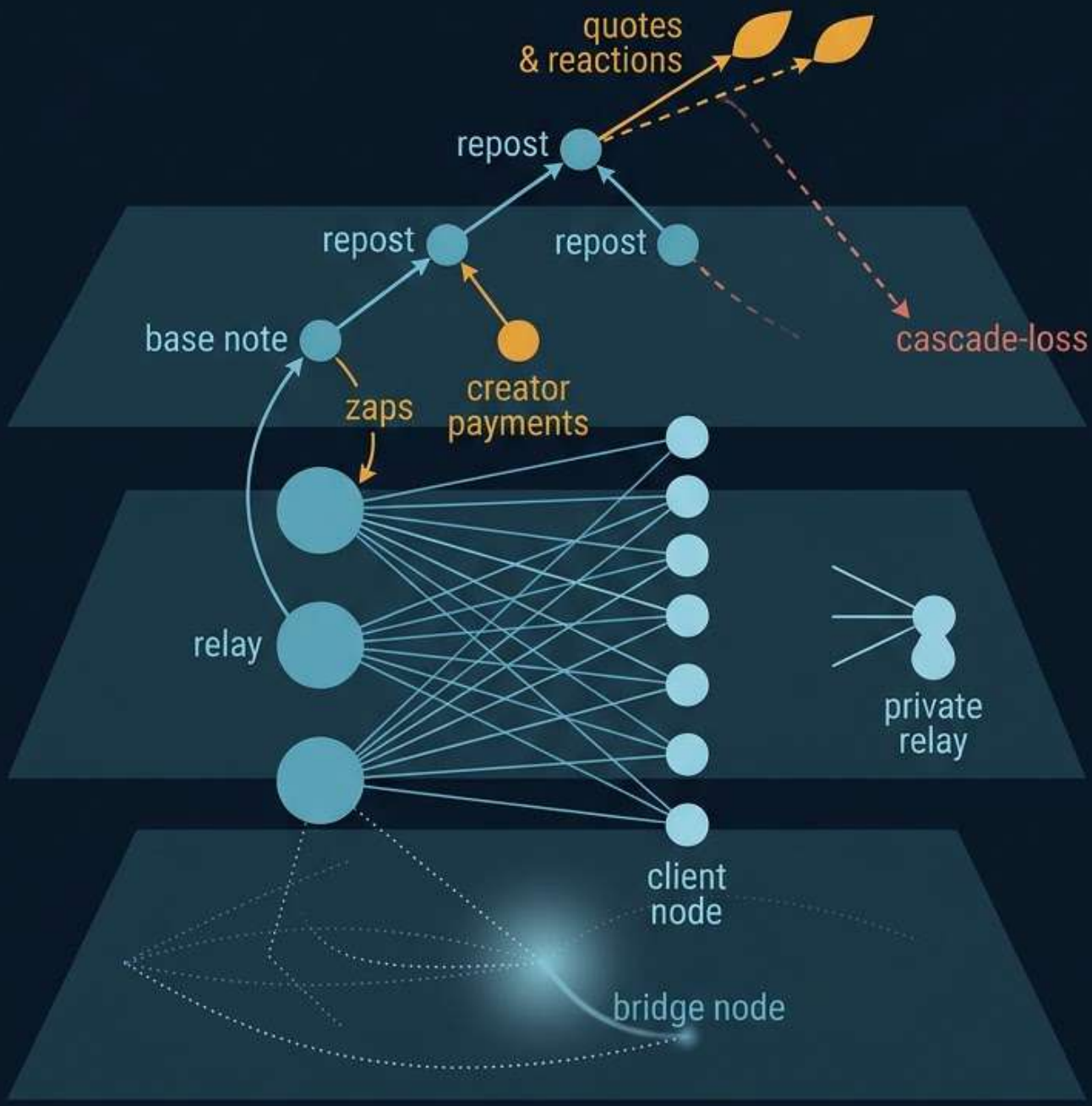
NOSTR NETWORKS

A Comprehensive Analysis of Protocol Architecture, Economics, Resilience & Decentralized Social Dynamics



The Nostr Book of NIPs, Volume I

Adam Shannon (NIP compilation), Libretech Systems (economic analysis), and the open Nostr research community

Open Protocol Research Collective

Scale-Free Network Topology Analysis: Mapping the Graph of NIP-02 Follow Lists

Status: Measurement protocol and reporting template
Scope: Nostr NIP-02 follow lists (kind:3 replaceable events)
Keywords: Nostr, NIP-02, follow graph, scale-free networks, power-law distributions, preferential attachment, network measurement

---

Abstract

Nostr defines a decentralized social graph through replaceable kind:3 follow-list events. Under NIP-02, each such event contains one or more p tags naming public keys that the event author follows. These events can be mapped directly into a directed graph: an edge from follower to followee exists when the follower’ s latest NIP-02 event contains the followee’ s public key. This whitepaper specifies a complete, reproducible methodology for constructing that graph from relay data and for testing whether its degree distribution exhibits scale-free, power-law behavior.

The analysis separates three questions that are often conflated: (1) whether the observed in-degree, out-degree, or total-degree distribution is statistically consistent with a power law; (2) whether the fitted exponent is stable across snapshots and robust to sampling choices; and (3) whether degree growth is consistent with preferential attachment or with alternative mechanisms such as fitness, aging, relay sampling bias, or bot activity. Because NIP-02 events are replaceable, the graph is not an append-only follow log. Each new follow list overwrites the previous one, so dynamic analysis requires repeated snapshots rather than relying on event history.

The whitepaper provides formal graph definitions, relay collection rules, replaceable-event deduplication, degree estimators, maximum-likelihood power-law fitting, alternative-distribution tests, bootstrap procedures, dynamic exponent tracking, growth models, null models, privacy constraints, and a reporting template.

---

1. Introduction

Nostr’ s social layer is unusually explicit. Instead of a platform-owned database of follows, users publish signed events that declare whom they follow. NIP-02 defines this as a kind:3

event containing p tags. The event author is the follower; each p tag target is a followee.

The event is replaceable: for each public key and kind, relays are expected to retain only the latest version, and older versions may be discarded. NIP-02 further states that every new follow list overwrites the past one and should contain the complete list; clients should append new follows to the end of the list so entries remain in chronological order. The .content field is not used for follow semantics.

This structure makes NIP-02 a natural substrate for network science. The central empirical question is whether the Nostr follow graph is scale-free in the sense that its degree distribution is heavy-tailed and follows a power law over some range.

For directed social graphs, the most commonly discussed quantity is in-degree: the number of distinct followers a public key has. Out-degree — the number of accounts a public key follows — is also important, but it is often constrained by human attention, client UX, and list-management behavior. A network can therefore be heavy-tailed in in-degree while having a much less heavy-tailed or bounded out-degree distribution. This whitepaper treats in-degree, out-degree, and total degree separately.

The phrase “power-law dynamics” is also used carefully. A static power-law fit does not prove a scale-free mechanism. It is consistent with preferential attachment, but also with fitness models, aging, cumulative advantage, heterogeneous sampling, and mixtures of populations. The methodology below therefore tests both the static degree distribution and the dynamic relationship between current degree and future degree growth.

Research Questions

- RQ1 — Static topology: Is the observed NIP-02 in-degree distribution consistent with a power law? Is out-degree consistent with a power law? Is total degree consistent with a power law?
- RQ2 — Alternative distributions: Does a power law outperform lognormal, exponential, Poisson, or power-law-with-cutoff alternatives using likelihood-based model comparison?
- RQ3 — Dynamics: Does the fitted power-law exponent change across snapshots? Is degree growth disproportionately concentrated among already high-degree nodes, as preferential attachment would predict?
- RQ4 — Robustness: Are results stable across relay subsets, crawl seeds, time windows, exclusion of self-loops, and exclusion of likely bot or relay accounts?
- RQ5 — Scale-free interpretation: If a power law is supported, what range of degrees does it cover, what is the finite-size cutoff, and what mechanisms remain plausible?

---

2. Background

2.1 NIP-02 as a replaceable follow-list event

NIP-02 defines kind:3 as a follow list. The important properties for graph construction are:

1. Author identity: The event public key is the follower.
2. Followees: Each valid p tag's second element is a 32-byte lowercase hex public key.
3. Replaceability: Only the latest kind:3 event per author should be treated as current. In case of equal timestamps, NIP-01's replaceable-event rule retains the event with the lowest ID in lexical order.
4. Complete-list semantics: A new follow list is expected to contain all current follows, not an incremental delta.
5. No content semantics: The .content field is not used by NIP-02.
6. Optional metadata: Relay hints and petnames may be present but are not required for the graph edge.
7. No guaranteed history: Relays may discard older versions. Dynamic analysis therefore requires periodic snapshots or relays that retain historical versions.

This last point is critical. A NIP-02 graph is a snapshot of replaceable state. It is not a complete chronological record of every follow and unfollow action unless historical events are explicitly available.

2.2 Scale-free networks and power laws

A network is often called scale-free when its degree distribution has a heavy tail approximated by a power law. In practice, empirical networks rarely follow a pure power law over all degrees. Analysts typically estimate a lower cutoff, fit an exponent, compare against alternatives, and report a finite-size cutoff where the power law breaks down.

For directed graphs:

- In-degree: The number of distinct authors whose latest NIP-02 list includes a specific node.
- Out-degree: The number of distinct valid targets in a node's latest NIP-02 list.
- Total degree: The sum of in-degree and out-degree.

In social networks, in-degree is often the quantity of interest because it reflects attention and follower accumulation. Out-degree may be bounded by cognitive limits, client limits, or list hygiene. A finding that in-degree is heavy-tailed while out-degree is not is fully compatible with a scale-free-like follower graph.

2.3 Preferential attachment and alternatives

The Barabási–Albert model predicts heavy-tailed degree distributions from growth and preferential attachment: new edges are more likely to attach to nodes that already have high degree. In a follow graph, this corresponds to the intuition that popular accounts attract followers disproportionately.

However, a static power law does not establish preferential attachment. Alternative explanations include:

- Fitness: Some accounts are intrinsically more attractive because of content, identity, or off-platform fame.
- Aging: Older accounts have had more time to accumulate followers.
- Relay sampling: High-degree accounts may be more likely to appear in crawled data.
- Bot and relay accounts: Automated accounts can inflate in-degree.
- Mixtures: The graph may combine human, bot, relay, and organization populations with different degree laws.

The dynamic analysis in Section 6 is designed to distinguish, as far as observational data allow, between “rich get richer” growth and these alternatives.

---

3. Formal Graph Model

3.1 Snapshot definition

Let a snapshot represent a specific moment in time. The nodes are the set of public keys observed either as NIP-02 authors or as valid p-tag targets at or before that time. The follow set of a user is the set of distinct valid followee public keys in their latest non-deleted kind:3 event.

The directed graph is formed by connecting a follower (author) to a followee (target). Edges are directed from follower to followee. Self-loops are retained for diagnostics but may be excluded from the main degree analysis; both choices should be reported.

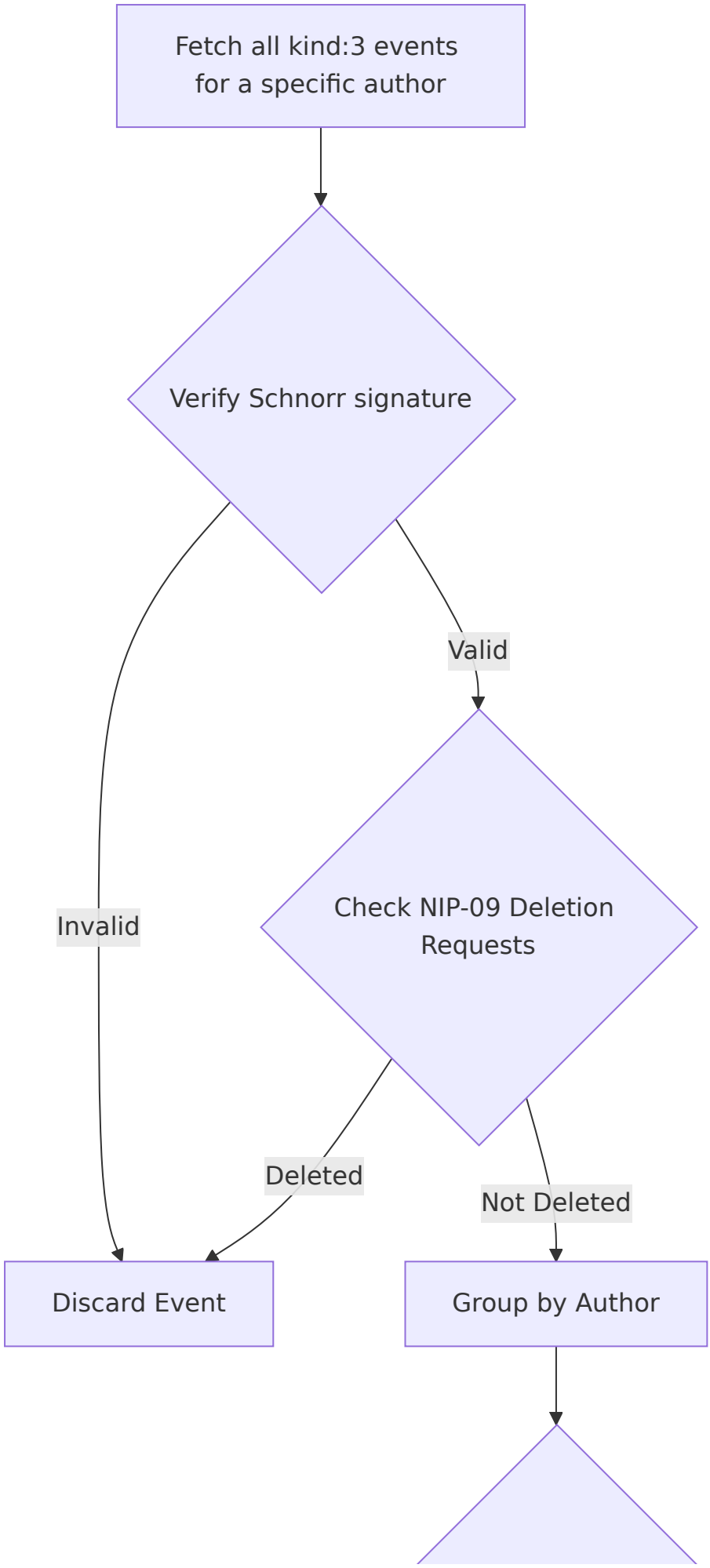
3.2 Degrees

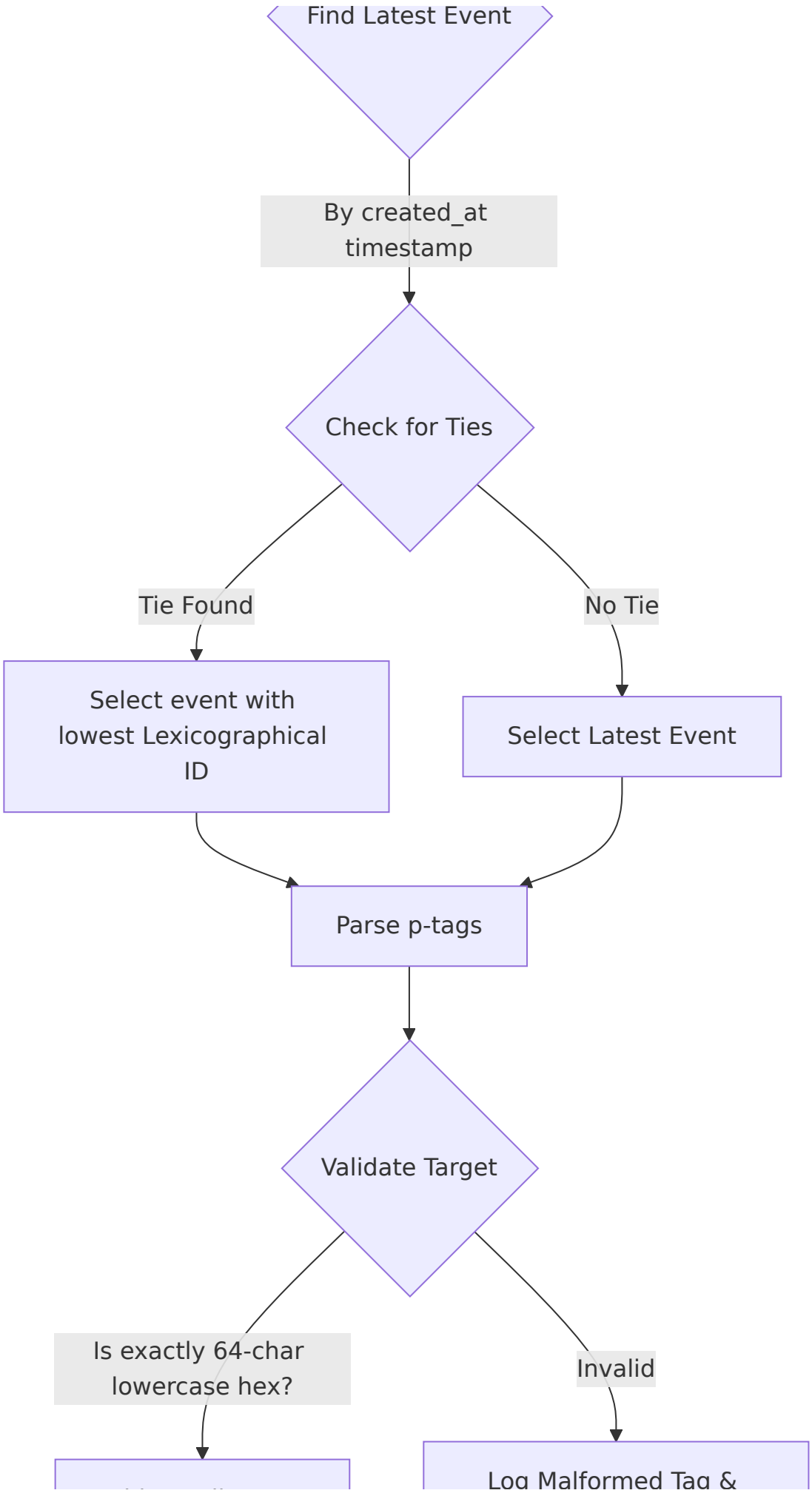
- In-degree: The count of unique authors who have a directed edge pointing to a specific node.
- Out-degree: The count of unique targets a specific node has a directed edge pointing to.
- Total degree: The sum of in-degree and out-degree.

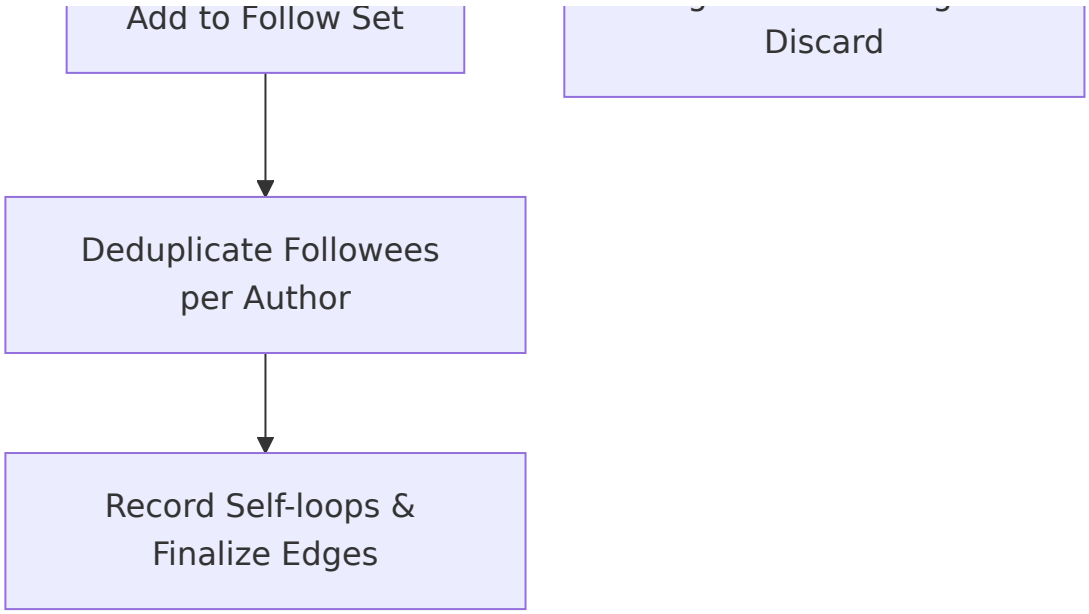
Only authors with a valid latest NIP-02 event contribute to out-degree. Nodes that appear only as followees have observed in-degree but unknown out-degree unless their own follow list is later collected. This distinction must be explicit in every table.

3.3 Replaceable-event resolution

To ensure we are looking at the most accurate current state of the graph, we must resolve replaceable events. The process for each author is as follows:


Fetch all kind:3 events for a specific author
Verify Schnorr signature
Invalid
Valid
Check NIP-09 Deletion Requests
Deleted
Not Deleted
Discard Event
Group by Author


Find Latest Event
By created_at timestamp
Check for Ties
Tie Found
No Tie
Select event with lowest Lexicographical ID
Select Latest Event
Parse p-tags
Validate Target
Is exactly 64-char lowercase hex?
Invalid
Log Malformed Tag &



3.4 Edge validity rules

An edge is created only when:

- The author's selected latest kind:3 event is valid and non-deleted.
- A p tag exists with the second element exactly matching the target's public key.
- The target is a 64-character lowercase hex public key.
- Duplicate p tags for the same target are collapsed into one edge.

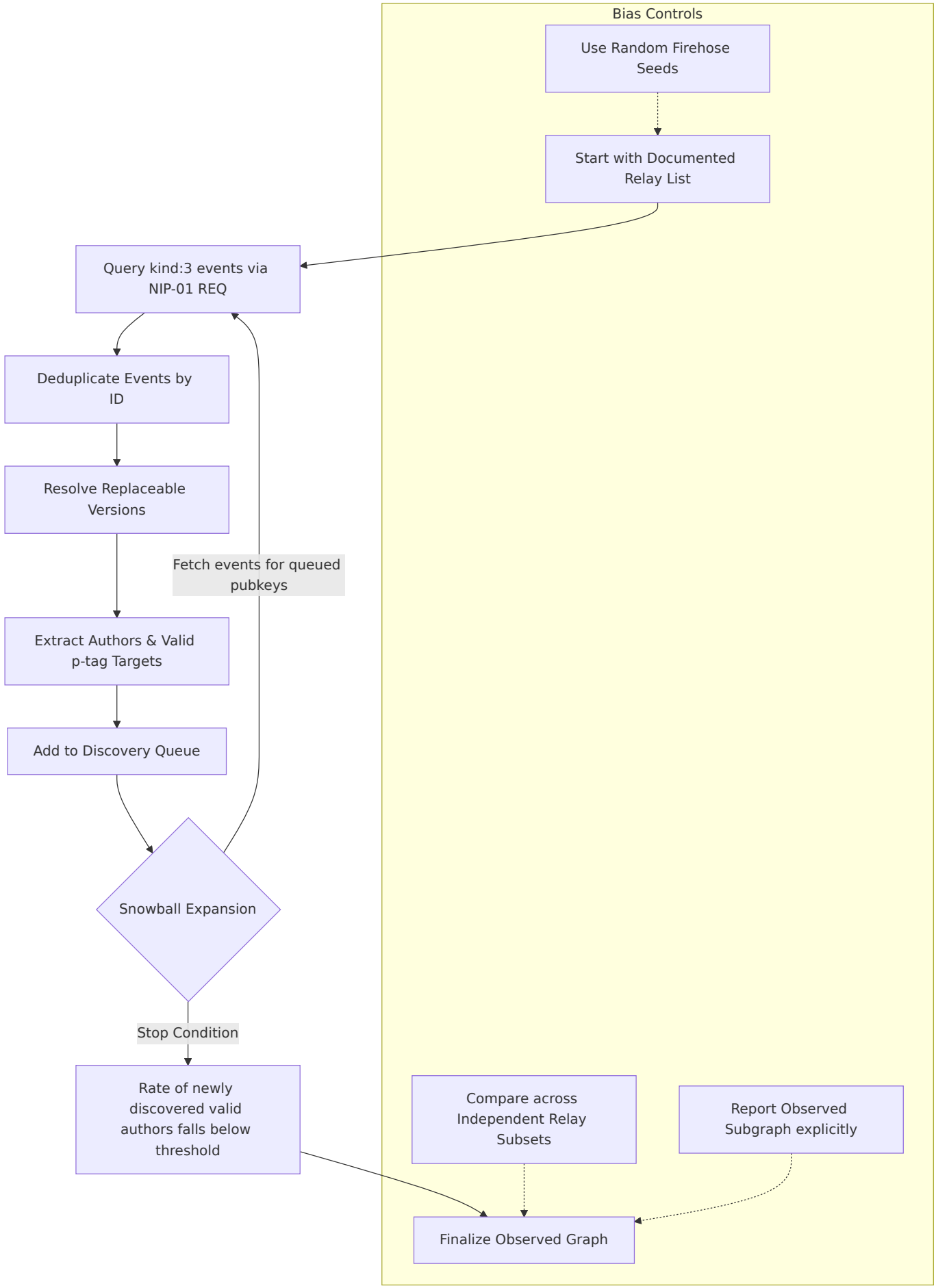
Relay hints and petnames do not affect edge existence. They may be recorded as presence flags for sensitivity analysis, but petnames should not be released to protect user privacy.

---

4. Data Collection Protocol

4.1 Relay selection and Crawling Strategy

Because no single relay sees the entire network, we must use a multi-relay union. We use a combination of firehose sampling and snowball expansion to build the observed graph.



4.2 Data schema

Recommended database tables to track the crawl:

Events Table:

- event_id (32-byte hex)
- pubkey (author)
- created_at (timestamp)
- kind (always 3)
- tags_json (raw tag data)
- sig (signature)
- valid_sig (boolean)
- first_seen_relay (URL)
- deleted (NIP-09 status)

Follows Table:

- snapshot_id
- author_pubkey (follower)
- target_pubkey (followee)
- relay_hint_present (boolean)
- petname_present (boolean)

Snapshots Table:

- snapshot_id
- collected_at
- relay_set_hash
- crawl_version

4.3 Privacy and ethics

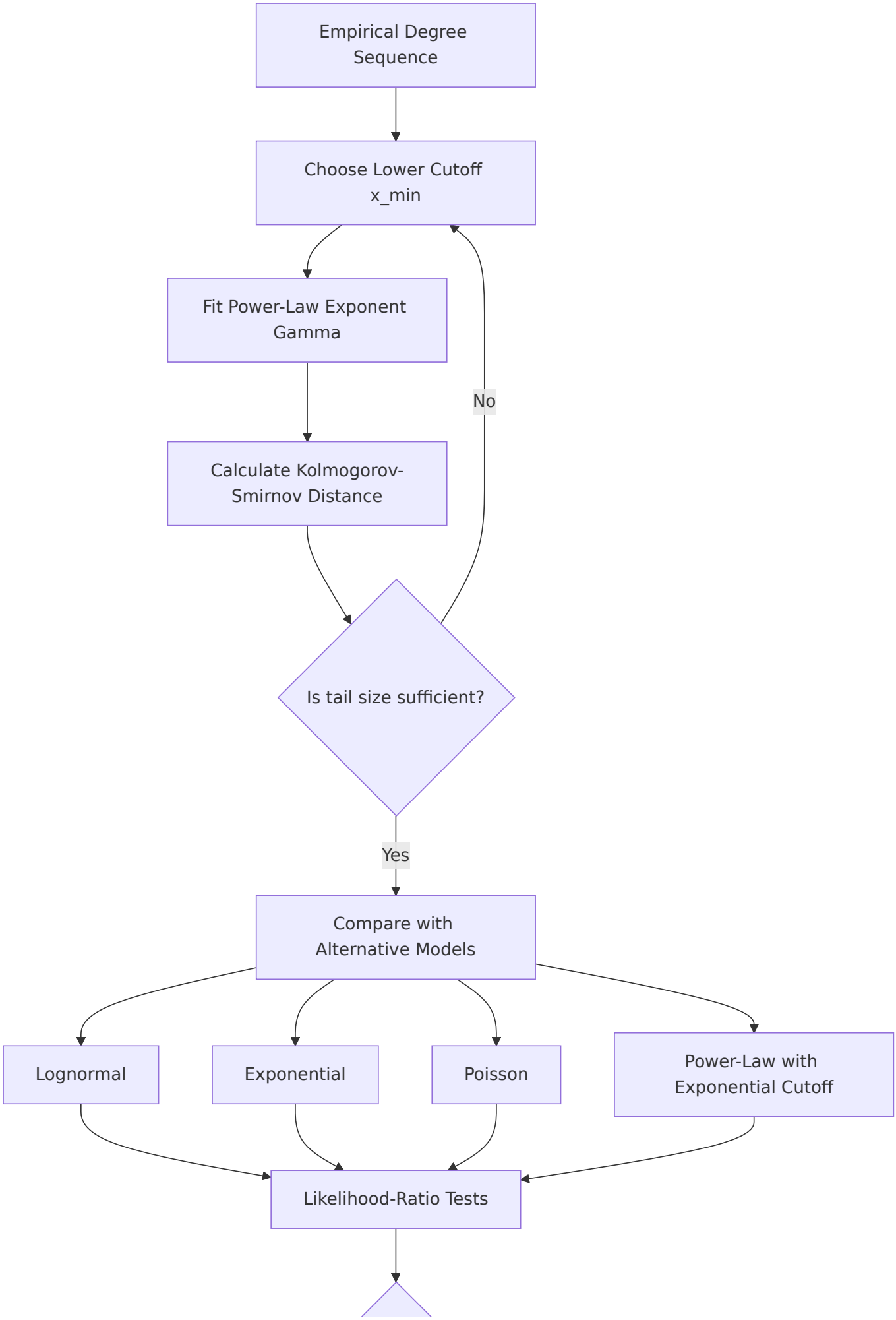
Public keys are public, but the graph is personal data in aggregate. Do not publish petnames, relay hints that could reveal private infrastructure, or low-degree user identities. Report aggregate degree distributions and minimum-degree thresholds. Avoid deanonymization or profiling of low-degree accounts. Honor relay terms of service and rate limits. Store raw data securely and release code and aggregate statistics rather than raw social graphs where possible.

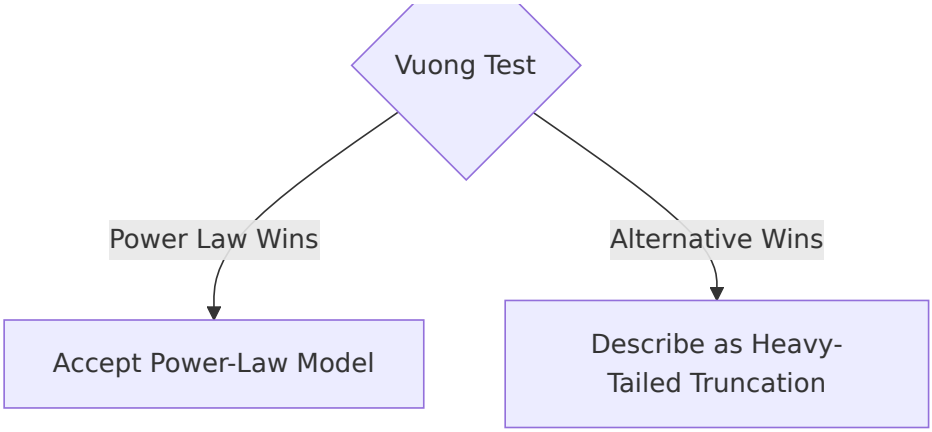
---

5. Static Degree-Distribution Analysis

5.1 Power-law fitting and model comparison

A power-law fit should not be accepted solely because a log–log plot looks straight. We must rigorously test if a power law is the best fit compared to other distributions.


Empirical Degree Sequence
Choose Lower Cutoff x_min
Fit Power-Law Exponent Gamma
Calculate Kolmogorov-Smirnov Distance
Is tail size sufficient?
No
Yes
Compare with Alternative Models
Lognormal
Exponential
Poisson
Power-Law with Exponential Cutoff
Likelihood-Ratio Tests



If a power law with a cutoff fits substantially better, report the cutoff value and describe the distribution as heavy-tailed with finite-size truncation rather than as a pure power law.

5.2 Separate analyses and topology metrics

Report the degree distributions separately for in-degree, out-degree, and total degree. Include self-loop-inclusive and exclusive versions, and bot-inclusive vs. excluded versions if metadata allows.

Beyond degree distribution, report:

- Number of nodes and edges.
- Graph density.
- Reciprocity (how often follows are mutual).
- Mean local clustering coefficient (how interconnected a node's neighbors are).
- Degree assortativity (do high-degree nodes connect to other high-degree nodes?).
- Gini coefficient of in-degree (measuring inequality).
- Top 1% and top 10% share of in-degree.

---

6. Power-Law Dynamics

6.1 Snapshot series and tracking

Static analysis gives a single picture. Dynamic analysis tracks how the graph evolves over time. We collect snapshots at regular intervals. For each snapshot, we estimate the power-law exponent, the lower cutoff, and the goodness-of-fit.

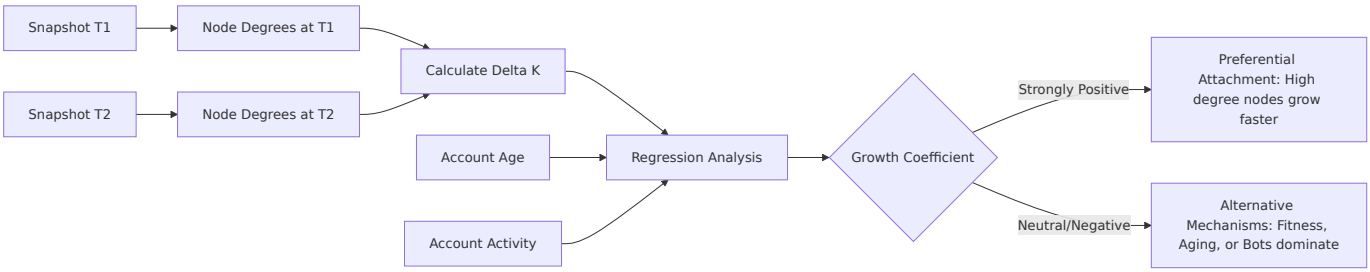
We plot the fitted exponent over time to see if it is stable. If the exponent changes significantly, it indicates that the mechanism generating the network is evolving.

6.2 Degree growth and preferential attachment

To test for preferential attachment, we look at nodes present in consecutive snapshots. We calculate the in-degree growth (the change in followers between snapshot A and snapshot B).

We then estimate a growth model that incorporates:

- The starting degree of the node.
- The age of the account.
- The activity level of the account.



A strongly positive relationship indicates disproportionate growth among higher-degree nodes, which is often interpreted as linear preferential attachment. However, this is observational and not causal. Report confidence intervals and sensitivity to the time window and inclusion of zero-growth nodes.

6.3 Edge churn and rank stability

If the graph is strongly preferential, high-degree ranks should be relatively stable while low-degree churn remains high. We compute:

- Jaccard similarity: How much the edge set overlaps between snapshots.
- Spearman rank correlation: How stable the in-degree rankings are over time.
- Top-K overlap: How many of the top 10, 100, or 1000 accounts remain in the top K between snapshots.

6.4 Null models

We compare the observed topology against null models:

1. Directed configuration model: Preserves the in- and out-degree sequences but randomly rewires the edges.
2. Random directed graph: Same edge count, but connections are completely random.
3. Relay-subset null: Repeating the analysis on random relay subsets.

If the observed graph’ s clustering, reciprocity, or assortativity differs strongly from the configuration model, degree distribution alone is insufficient to characterize the topology.

7. Reporting Template

The following tables should be filled from the executed snapshot. Values are placeholders.

Metric Breakdown by Degree Type

Metric	In-degree	Out-degree	Total Degree
Nodes analyzed	TBD	TBD	TBD
Edges	TBD	TBD	TBD
Mean	TBD	TBD	TBD
Median	TBD	TBD	TBD
Max	TBD	TBD	TBD
Fitted Exponent	TBD	TBD	TBD
Lower Cutoff (x_min)	TBD	TBD	TBD
Tail size (n)	TBD	TBD	TBD
KS Distance	TBD	TBD	TBD
Bootstrap p-value	TBD	TBD	TBD
Preferred Model	TBD	TBD	TBD

Topology Summary

Metric	Value
Total Nodes (N)	TBD
Total Edges (E)	TBD
Graph Density	TBD
Reciprocity	TBD
Mean Clustering	TBD
Assortativity	TBD
Self-loops	TBD
Gini of In-degree	TBD
Top-1% In-degree Share	TBD
Top-10% In-degree Share	TBD

Dynamics Summary

Metric	Value
Number of Snapshots	TBD
In-degree Exponent Trend	TBD
Growth Coefficient (Beta)	TBD
95% CI for Beta	TBD

Edge Jaccard Similarity TBD
Spearman Rank Correlation TBD
Top-100 Overlap TBD

---

8. Threats to Validity
9. Incomplete relay coverage: The observed graph undercounts in-degree and may miss authors entirely.
10. Replaceable-event history loss: Older NIP-02 versions may be unavailable, so dynamic analysis may only observe snapshots, not true event history.
11. Snowball sampling bias: Well-connected nodes are more likely to be discovered.
12. Bot and relay accounts: Automated accounts can distort in-degree tails.
13. Client behavior: Some clients may publish incomplete follow lists or fail to append chronologically.
14. Malformed tags: Invalid p tags or non-hex public keys must be excluded and counted.
15. Power-law overfitting: Log–log linearity is not sufficient; alternative distributions and goodness-of-fit tests are required.
16. Mechanism ambiguity: Preferential attachment cannot be proven from static degree data alone.

---

9. Conclusion

NIP-02 provides a direct, signed mapping from Nostr users to their follow relationships. The resulting directed graph is well suited to degree-distribution analysis, but it must be treated as a replaceable snapshot graph rather than a complete follow history. A rigorous scale-free analysis requires multi-relay collection, signature verification, replaceable-event resolution, explicit treatment of self-loops and malformed tags, maximum-likelihood power-law fitting, alternative-distribution comparison, and dynamic tests of degree growth.

The central empirical question is not simply “is Nostr scale-free?” but rather: for which degree sequence, over what range, under which sampling assumptions, and with what dynamic mechanism? The protocol above is designed to answer those questions reproducibly and to prevent a visually straight log–log plot from being mistaken for evidence of a scale-free network.

Appendix A: Extraction Pseudocode Flowchart

flowchart TD A[Start: Loop through Relays in Relay Set] --> B[Fetch kind:3 events from Relay] B --> C{Validate Signature} C -- Invalid --> A C -- Valid --> D[Store Event] D --> E[Group Stored Events by Author Pubkey] E --> F[Remove NIP-09 Deleted Candidates] F --> G[Select Max Event by created_at, tie-break by lowest id] G --> H[Loop through Selected Event Tags] H --> I{Tag[0] is 'p' AND Tag length >= 2 AND Tag[1] is valid hex?} I -- Yes --> J[Add to Author's Follow Set] I -- No --> H J --> K[Add Edge: Author -> Followee] K --> L[Compute In-degree, Out-degree, Total Degree] L --> M[Fit Power Law to Distributions] M --> N[Compare Alternatives] N --> O[Repeat for each Snapshot] O --> P[Estimate Growth Model]

Appendix B: Minimum Reproducibility Checklist

☐ Exact relay list and fetch timestamps
☐ Crawl seed policy and stopping rule
☐ Signature verification method
☐ Replaceable-event tie-breaking rule
☐ Deletion-request handling rule
☐ Malformed p tag counts
☐ Self-loop policy
☐ x_min search range and minimum tail size
☐ Alternative distributions tested
☐ Bootstrap procedure and number of resamples
☐ Code version and data hash
☐ Privacy review of released aggregates

Note on empirical results: The supplied material is the NIPs compilation, not a live relay snapshot. The tables above are therefore reporting templates. Executing this protocol against a documented Nostr snapshot will populate the degree estimates, exponent values, model-comparison results, and dynamic coefficients.



Syntax error in text
mermaid version 11.12.2

Small-World Network Mechanics: Testing "Six Degrees of Separation" Properties in Decentralized User Graphs

Status: Measurement protocol and reporting template
Scope: Decentralized user graphs derived from Nostr NIP-02 follow lists
Keywords: Small-world networks, six degrees of separation, average path length, clustering coefficient, decentralized social graphs, Nostr, NIP-02

Abstract

The "six degrees of separation" claim is one of the most durable ideas in network science: any two people in a large social system are connected by a surprisingly short chain of acquaintances. In centralized platforms, short paths are often an artifact of platform design and a single global database. Decentralized user graphs make a different claim: even without a central authority or a global follow directory, users may still be separated by only a few hops.

This whitepaper specifies a complete methodology for testing small-world properties and the six-degrees hypothesis in decentralized user graphs, using Nostr's NIP-02 follow lists as the empirical substrate. It defines the graph model, distinguishes different types of connections, formalizes how to measure average path length and clustering, and provides exact and approximate path-length estimators suitable for partially observed, fragmented graphs. It also specifies reference models, statistical sampling controls, and a reporting template.

The central question is not merely "are there six degrees?" but: under what sampling assumptions, for which pairs of users, over what observed portion of the graph, and with what uncertainty? A test of six degrees must distinguish between the true graph (everything that exists), the observed graph (what we managed to crawl), and the estimable graph (what we can infer with bounded uncertainty).

1. Introduction

The small-world phenomenon has two components. The first is short average path length: most pairs of nodes are connected by a surprisingly small number of edges. The second is

high local clustering: the people you follow are disproportionately likely to follow each other.

Most empirical tests of six degrees have been conducted on centralized graphs where data is complete. Decentralized user graphs are different. There is no global database, no single relay that sees all follows, and no guaranteed completeness. The graph is public in principle, but only partially observable in practice.

This creates a specific measurement problem. A test of six degrees in a decentralized graph must answer:

1. What is the graph? The true graph is the union of all valid follow lists across all relays. The observed graph is what a crawl sees. The estimable graph is what can be inferred with bounded uncertainty.
2. What counts as a degree? Follow edges are not symmetric. In-degree (followers) and out-degree (following) have different distributions, and path length depends on whether you follow the direction of the edges or ignore them.
3. What is a path? A directed path means User A follows someone who follows someone... who follows User B. An undirected path ignores direction. The six-degrees claim is usually about undirected acquaintance chains.
4. What sampling assumptions are required? Crawling over-represents well-connected nodes. Relay fragmentation hides edges. Missing users inflate path length or disconnect the graph.
5. What uncertainty accompanies an estimate? An estimate of 4.2 hops without confidence bounds is not a scientific test.

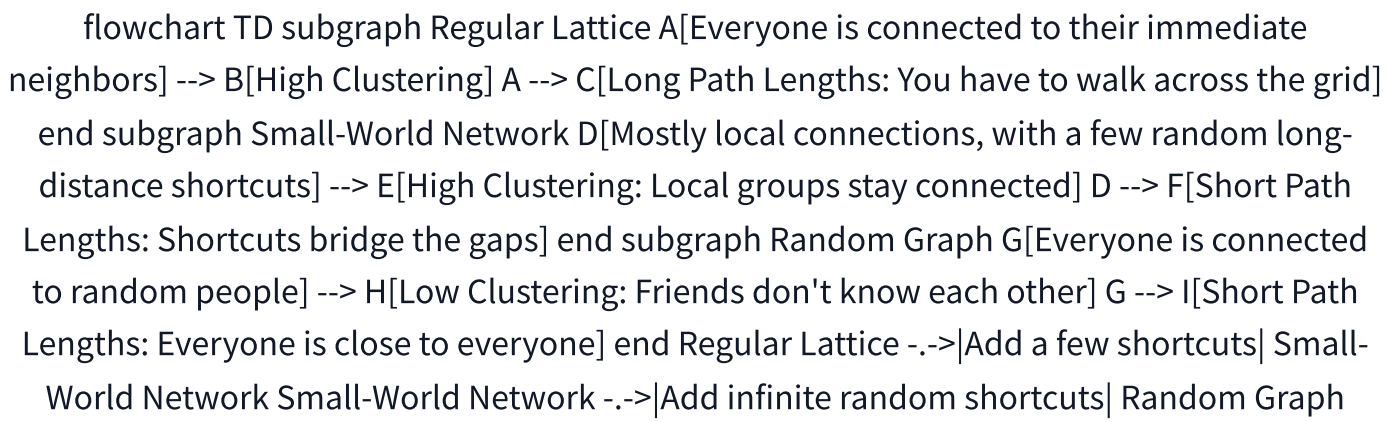
This whitepaper provides a rigorous protocol for answering these questions. It is organized around four tasks: (1) build the observed graph from NIP-02 data; (2) measure small-world metrics with explicit directionality; (3) estimate path length under partial observation; (4) test whether the observed structure is distinguishable from null models.

---

2. Background

2.1 Small-world networks

A small-world network is a graph where nodes are highly clustered (your friends know your other friends), but the average distance between any two nodes is very short. This is visualized as a spectrum between a highly ordered grid and a completely random mess.



A graph is considered "small-world" if it looks like the middle section: it has the high clustering of a lattice but the short path lengths of a random graph. To prove this, scientists compare the observed graph to a randomized reference model. If the observed graph's clustering is much higher than the random model, but its path length is about the same, it is a small-world network.

2.2 Six degrees of separation

The phrase originates from Stanley Milgram's experiment, in which letters were forwarded through acquaintances to reach a target. The scientific claim is precise: in many large social graphs, the average shortest path length is small relative to the number of nodes, often in the range of 3 to 7 hops.

Important caveats:

- Average vs. Maximum: "Six degrees" usually refers to the average, not the absolute maximum. Some outliers may be 15 hops away.
- Completed chains only: If people drop the letter out of disinterest, the average looks artificially short.
- Graph definition: Results depend on whether edges are directed, whether weak ties are included, and how nodes are sampled.

2.3 Decentralized user graphs

Nostr is a canonical example. Users publish signed events to relays; relays store and serve events; clients choose which relays to query. The NIP-02 follow list is a replaceable event containing p tags.

The graph is:

- Publicly signed, but not globally indexed.
- Replaceable, so historical follows are not guaranteed to be retained.

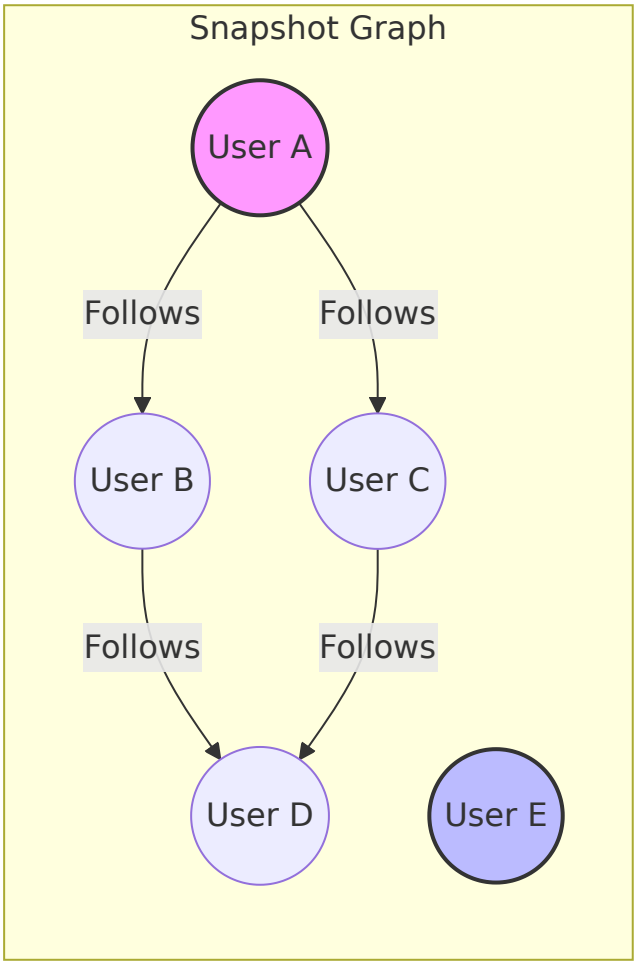
- Relay-fragmented, so different relays hold different subsets.
- Identity-based, so bots and relay accounts are part of the graph.

---

3. Formal Graph Model

3.1 Snapshot graph

A snapshot graph represents the state of the network at a specific point in time.



- Nodes: The set of all observed public keys (either as authors or as targets).
- Edges: Directed connections pointing from the follower to the followee.

3.2 Directed and undirected projections

For path analysis, we must define how we walk the graph:

- Out-path: Following edges forward (A follows B, B follows C).
- In-path: Following edges backward (C is followed by B, B is followed by A).
- Undirected path: Ignoring direction. If A follows B, they are considered connected regardless of direction.

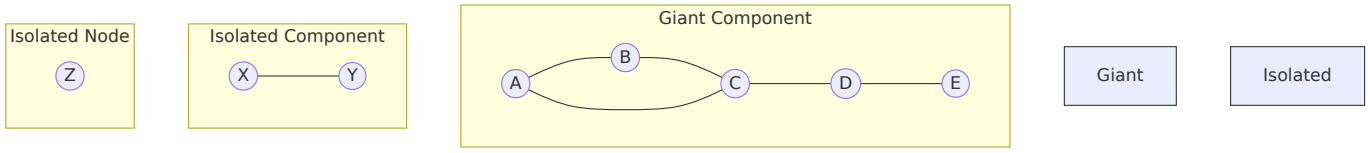
The six-degrees claim is usually tested on the undirected projection because acquaintance is treated as symmetric. However, directed reachability must be reported separately because a graph can be heavily disconnected in one direction but connected in the other.

3.3 Degrees

- In-degree: The number of unique authors who point to a specific node.
- Out-degree: The number of unique targets a node points to.
- Total degree: The sum of in-degree and out-degree.
- Hubs: Nodes with exceptionally high out-degree (accounts that follow thousands of people) or in-degree (celebrities).

3.4 Connected components

A connected component is a subset of the graph where every node can reach every other node.



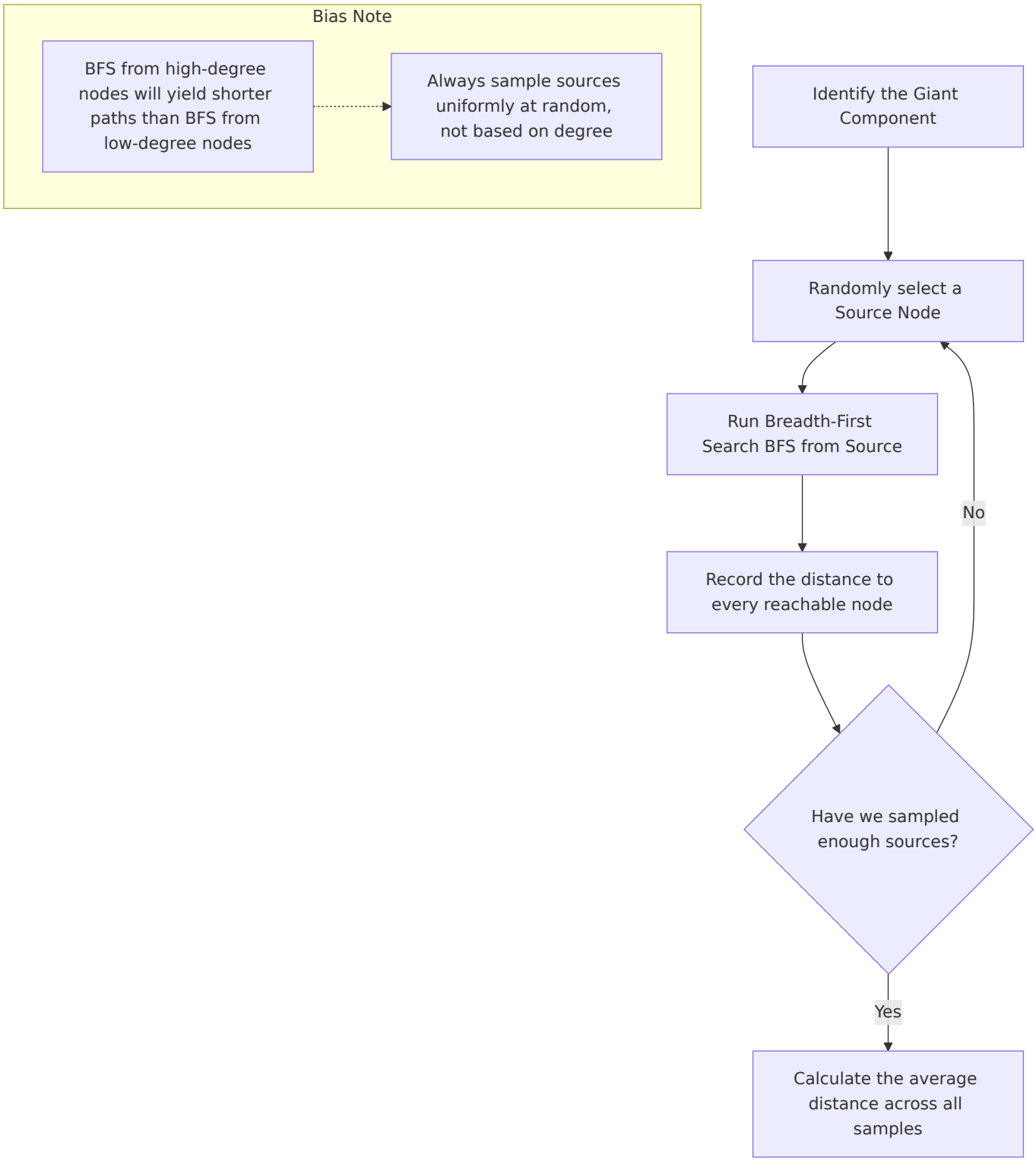
Path-length statistics should be computed only within the giant component (the largest connected cluster). A graph with a small giant component can have short paths inside the component while most pairs are entirely disconnected.

---

4. Small-World Metrics

4.1 Average shortest path length

This is the average number of hops to get from any person to any other person in the giant component. For large graphs, calculating every single path is too expensive. We use sampling:



4.2 Clustering coefficient

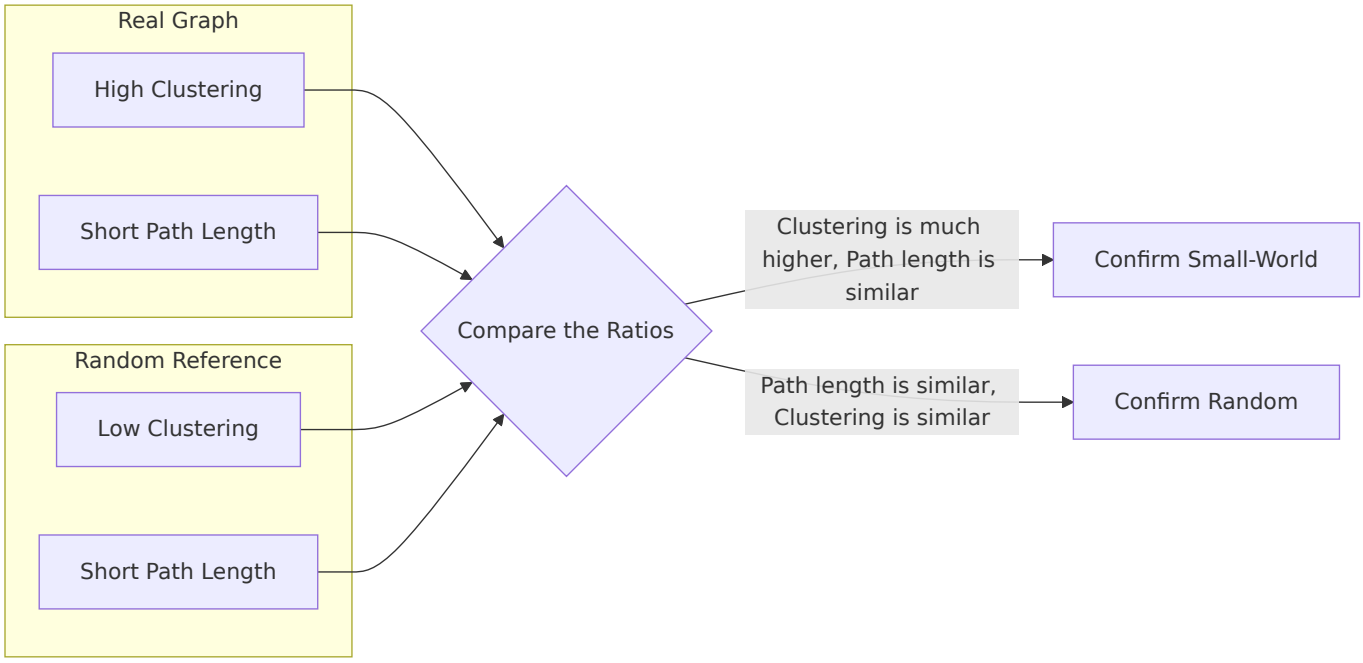
This measures how interconnected a node's neighborhood is.

- Local Clustering: If a user follows 10 people, we look at how many of those 10 people follow each other. If all 10 follow each other, local clustering is high. If none follow each other, it is zero.
- Global Clustering (Transitivity): We count the total number of "triangles" (A follows B, B

follows C, C follows A) and divide it by the total number of connected triples (A follows B, B follows C, but C does not follow A).

4.3 Small-world indices

To determine if a graph is truly "small-world," we compare it to a Random Reference Model (a randomized version of the network that has the exact same number of connections per person, but scrambled).



4.4 Diameter and effective diameter

- Diameter: The absolute longest shortest-path in the network (the longest distance between the two most distant people).
- Effective Diameter: The number of hops required to reach 90% of the network. This is more robust than the exact diameter, which can be skewed by a single outlier.
- Percentiles: Instead of just the average, we report the 90th and 99th percentile path lengths. A mean of 5 with a 99th percentile of 12 is a very different network from a mean of 5 with a 99th percentile of 7.

---

5. Testing Six Degrees of Separation

5.1 Hypothesis framing

We establish two competing hypotheses:

- Null Hypothesis: The average undirected path length in the giant component is greater than 6.

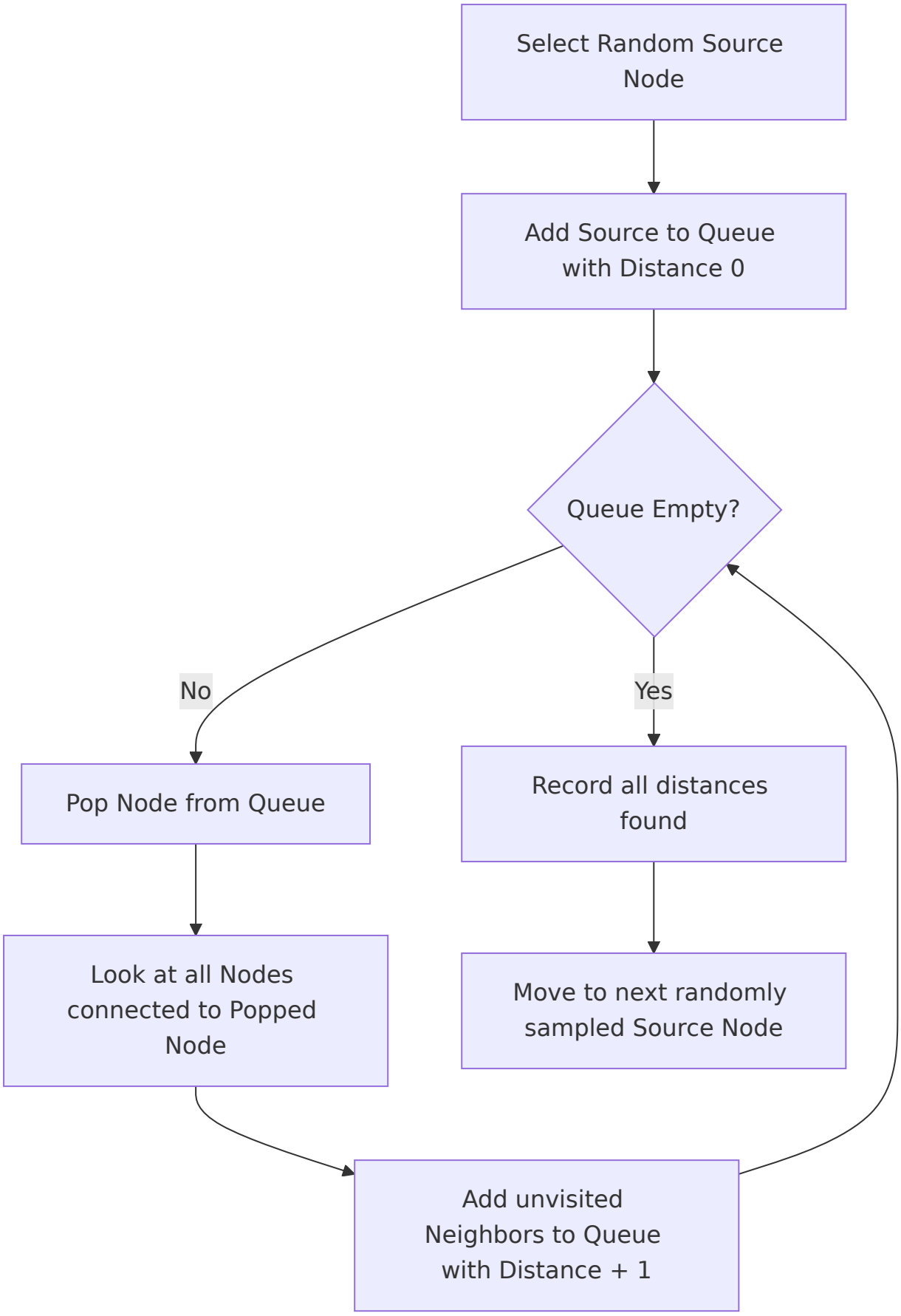
- Alternative Hypothesis: The average undirected path length is 6 or less.

Because pairs of nodes in a graph share paths (they are not independent), we cannot use standard statistical tests. Instead, we use bootstrap resampling over source nodes to generate confidence intervals.

A highly informative metric is Pi-6: the exact probability that two randomly chosen people in the giant component are within 6 hops of each other. If Pi-6 is high (e.g., 95%), the six-degrees claim is strongly supported.

5.2 Exact BFS on a bounded subgraph

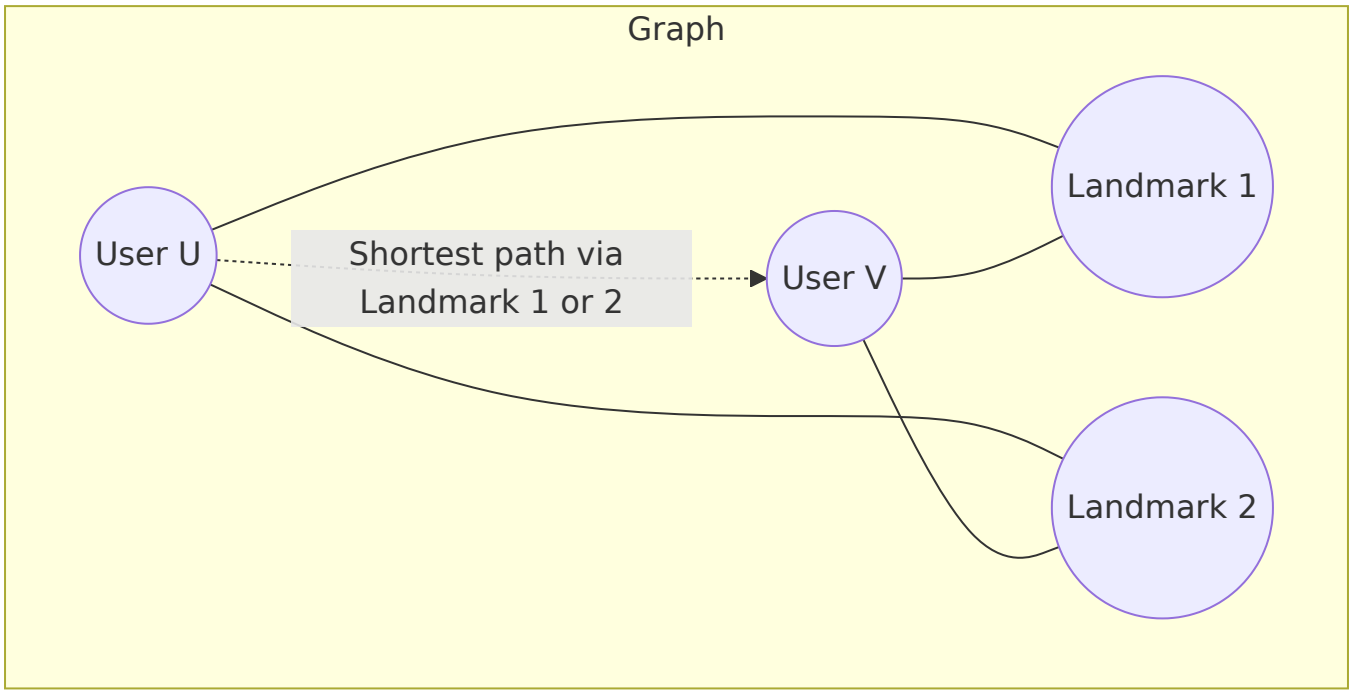
If the graph is too massive to search from every node, we sample random sources and run Breadth-First Search (BFS).



We report the number of sources, the BFS depth cutoff, and the fraction of pairs that are unreachable within the cutoff.

5.3 Landmark-based estimation

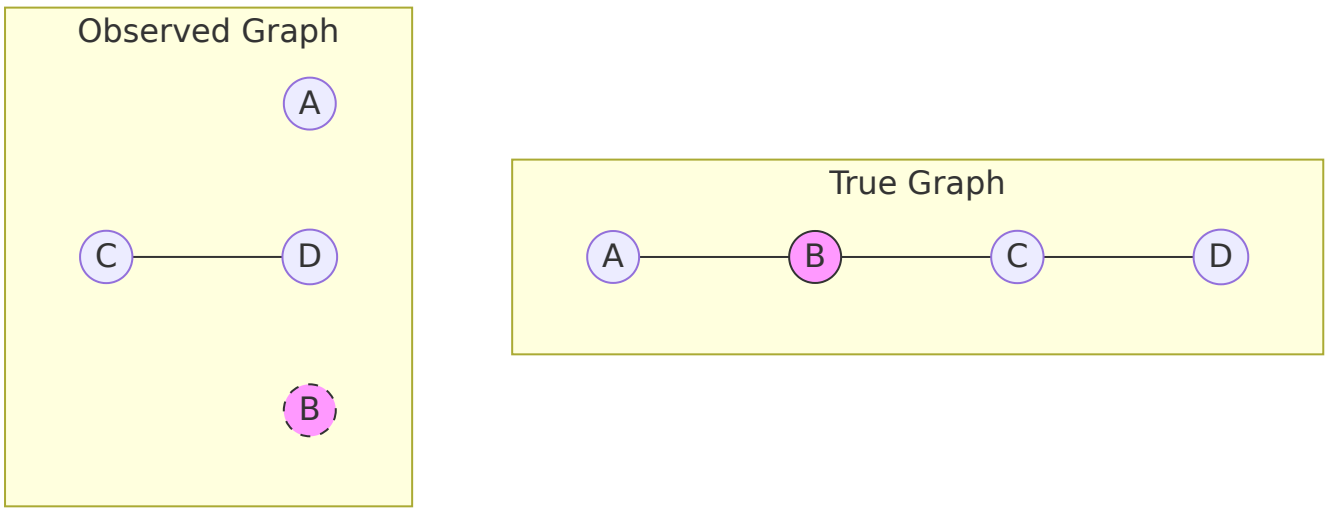
For massive graphs, we use "Landmarks."



We pick a small set of random landmarks and calculate the exact distance from every landmark to every node. To estimate the distance between User U and User V, we look at their distances to the landmarks. If they both know Landmark 1, their estimated distance is the distance from U to L1 plus V to L1. This is an upper bound—it might be longer than the true shortest path, but it will never be shorter.

5.4 Sampling and visibility correction

Because we cannot see the entire Nostr network, the observed graph is a subgraph of the true graph. Missing nodes and edges artificially inflate path lengths.



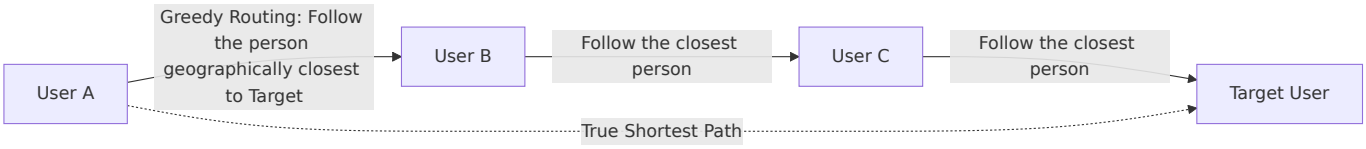
- Observed Length: A to B to C to D (3 hops).
- True Length: If we missed the edge between A and C, the observed path might be longer.

Therefore, observed path lengths are always greater than or equal to true path lengths.

- Correction: We use "capture-recapture" across different relays to estimate how many nodes we missed. We report a lower bound (the observed graph) and an upper bound (assuming missing nodes only add shortcuts).

5.5 Navigation vs. existence

Six degrees is an existence claim: short paths exist. A separate navigability claim asks whether a local routing rule can find them.



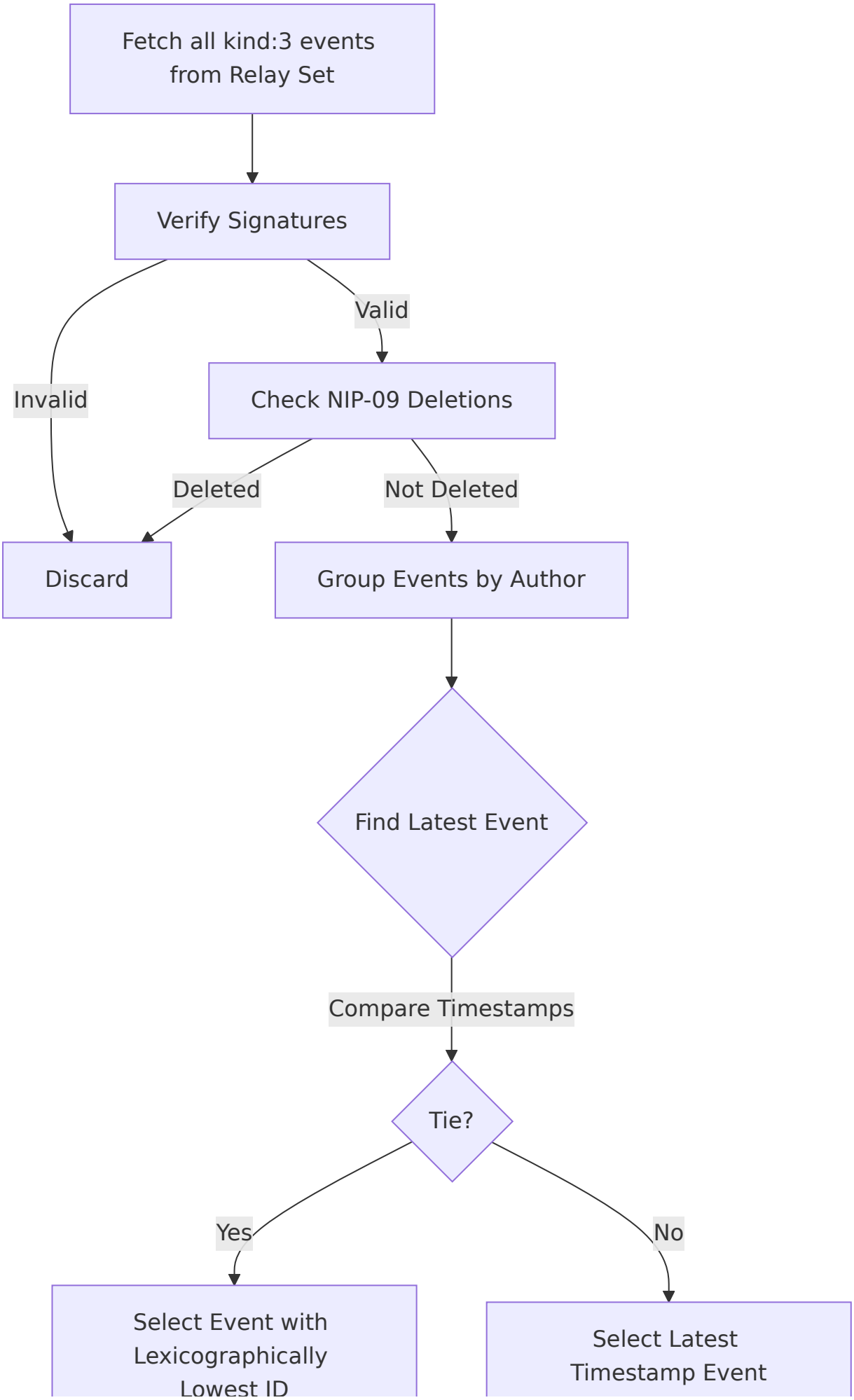
A graph can be small-world (short paths exist) but not navigable (greedy routing gets stuck). We measure Stretch (the length of the greedy path divided by the true shortest path). A stretch of 1.0 means greedy routing found the perfect path. A high stretch means the network is hard to navigate.

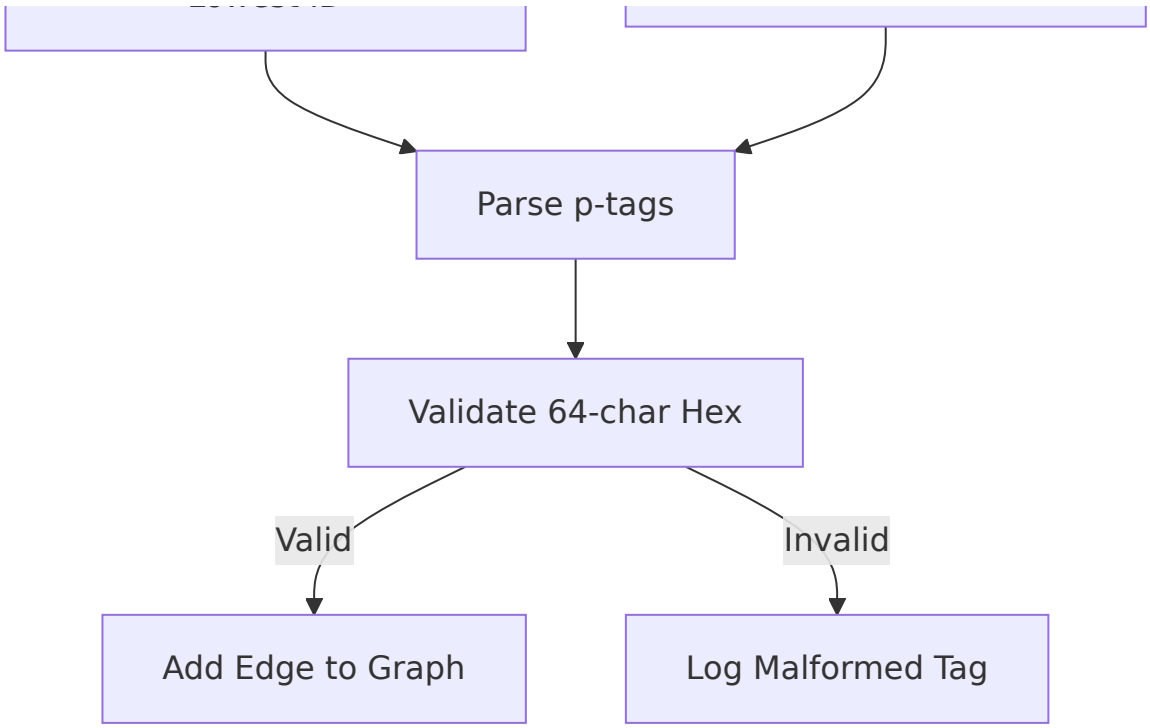
---

6. Data Collection and Sampling Protocol

6.1 Relay selection and Replaceable-event resolution

Nostr events are replaceable. If a user updates their follow list, the old one is theoretically discarded. We must carefully resolve which event represents the "current" state.


Fetch all kind:3 events from Relay Set
Verify Signatures
Invalid
Valid
Check NIP-09 Deletions
Deleted
Not Deleted
Discard
Group Events by Author
Find Latest Event
Compare Timestamps
Tie?
Yes
No
Select Event with Lexicographically Lowest ID
Select Latest Timestamp Event

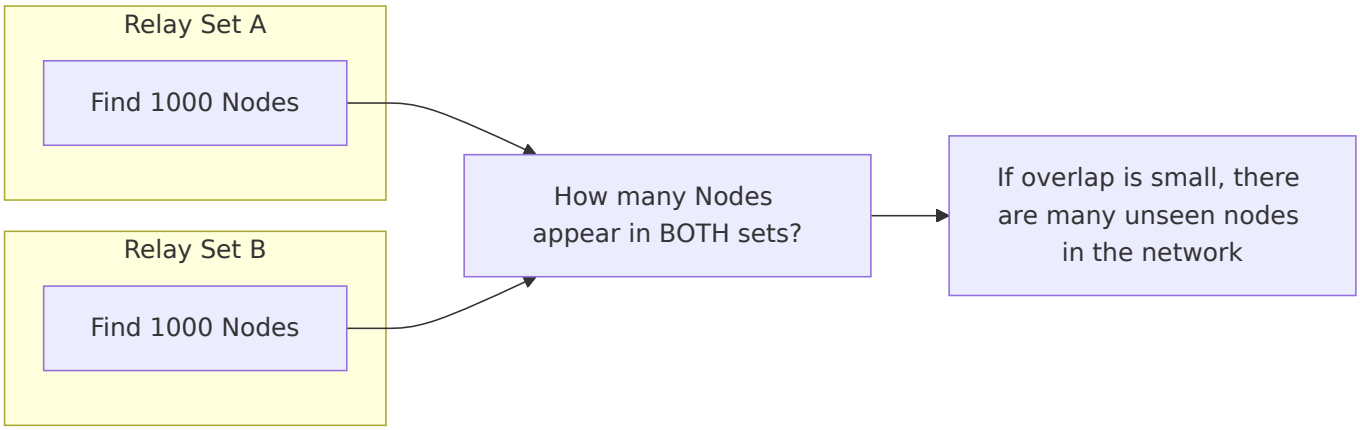


6.2 Crawl strategy

We use a combination of:

- Firehose sampling: Listening to random live events.
- Snowball expansion: Finding the people our initial seeds follow, and querying them too.

To control for bias, we must seed with random authors, not just high-degree "celebrities." We also use a capture-recapture method:



If the overlap between two independent relay crawls is high, we have likely seen most of the network. If the overlap is low, our observed graph is highly incomplete.

6.3 Data schema

Events Table: Tracks event_id, pubkey (author), created_at, kind (always 3), tags_json, sig (signature), valid_sig, first_seen_relay, fetched_at, deleted (NIP-09 status).

Follows Table: Tracks snapshot_id, author_pubkey (follower), target_pubkey (followee), relay_hint_present, petname_present.

Snapshots Table: Tracks snapshot_id, collected_at, relay_set_hash, crawl_version.

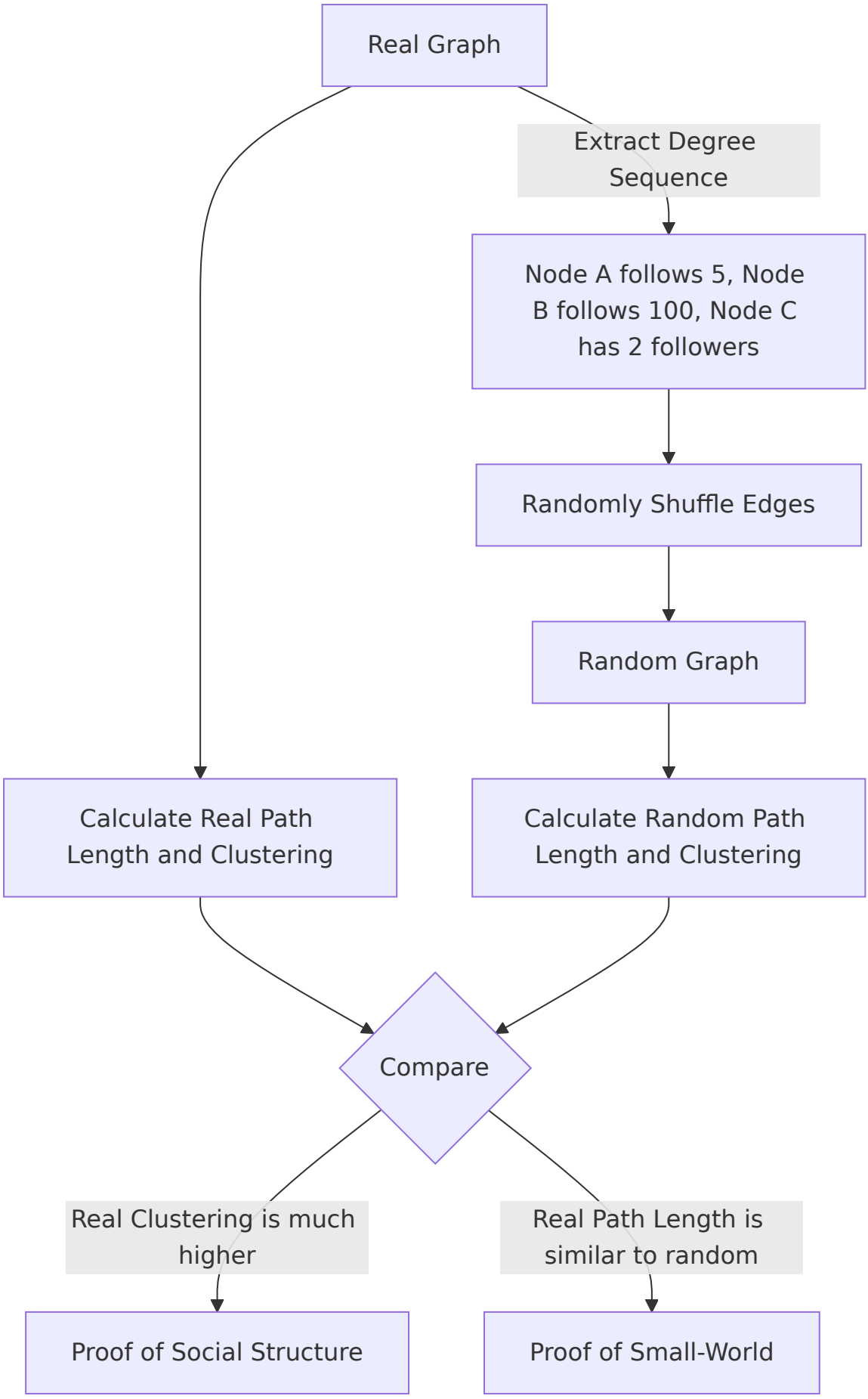
---

7. Null Models and Robustness

To prove our findings are real, we compare the observed graph against generated "null models."

7.1 Degree-preserving random graph

We generate a random graph that has the exact same number of followers and followings per person as the real graph, but the edges are randomly scrambled.



7.2 Relay-subset null

We repeat the path length analysis on random subsets of relays. If path length changes wildly depending on which relays we query, our overall result is highly sensitive to visibility and cannot be trusted as a global metric.

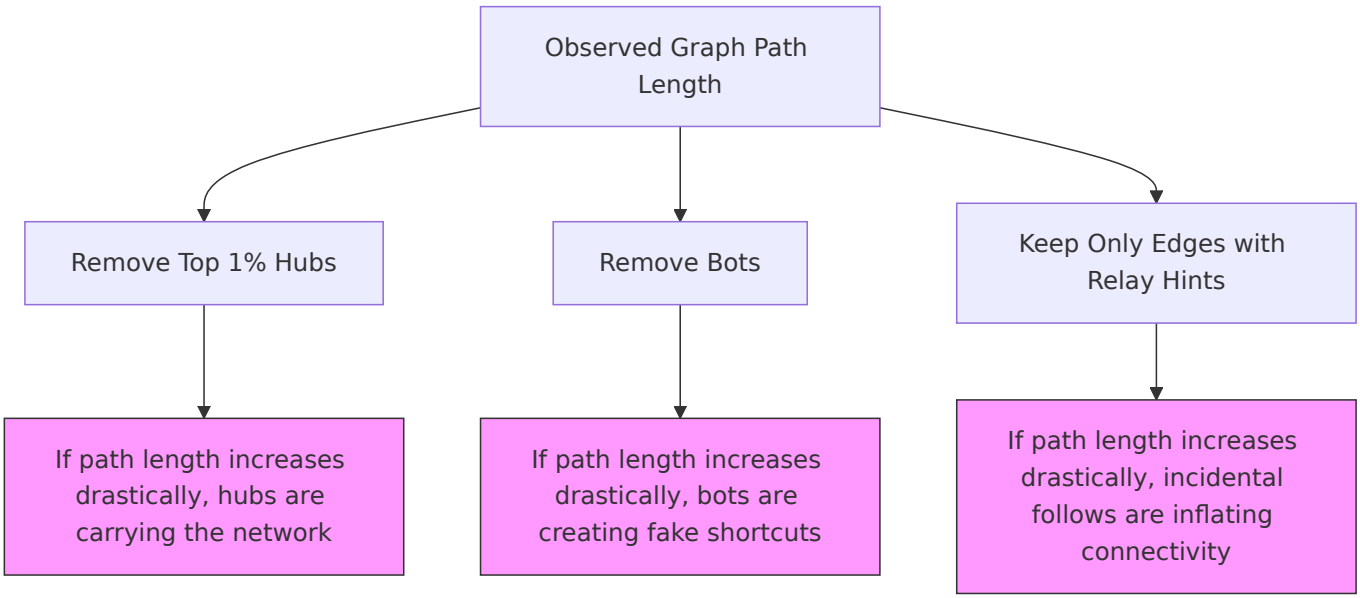
7.3 Missing-edge sensitivity

We randomly delete 5%, 10%, 20%, and 40% of the edges in our observed graph and recalculate the path length. If the six-degrees conclusion flips (i.e., the average goes above 6) when we delete just 5% of edges, the result is fragile.

---

8. Decentralization and Path Length

A short path length does not automatically mean a network is organically close. It could be an artifact of hubs or bots.



- Hub-removal effect: What happens if we remove the top 1% of most-followed accounts?
- Bot-exclusion effect: What happens if we exclude accounts marked as bots in their metadata?
- Relay-hint edge subset: What happens if we only count follows that explicitly include a relay hint (implying a more deliberate follow)?

---

9. Reporting Template

The following tables should be filled from the executed snapshot. Values are placeholders.

9.1 Graph summary

Metric	Value
Snapshot time	TBD
Relays crawled	TBD
Nodes observed	TBD
Edges observed	TBD
Giant component size	TBD
Fraction outside giant component	TBD
Self-loops	TBD
Reciprocal edges	TBD

9.2 Small-world metrics

Metric	Undirected	Out-directed	In-directed
Average path length	TBD	TBD	TBD
Median path length	TBD	TBD	TBD
90th percentile path	TBD	TBD	TBD
99th percentile path	TBD	TBD	TBD
Effective diameter	TBD	TBD	TBD
Clustering coefficient	TBD	TBD	TBD
Transitivity	TBD	TBD	TBD
Random Graph Path Length	TBD	TBD	TBD
Random Graph Clustering	TBD	TBD	TBD
Small-World Index (Sigma)	TBD	TBD	TBD

9.3 Six-degrees test

Quantity	Value
Probability of 6 degrees or less	TBD
95% Confidence Interval	TBD
Mean Path Length with 95% CI	TBD
Sources sampled for BFS	TBD
BFS depth cutoff	TBD
Unreachable pair fraction	TBD
Conclusion	Supported / Rejected / Indeterminate

9.4 Robustness

Perturbation	Path Length	Prob. of 6 Degrees
Observed graph	TBD	TBD
Remove top 1% hubs	TBD	TBD

Remove bots TBD TBD
Delete 10% edges TBD TBD
Random relay subset TBD TBD

---

10. Threats to Validity
11. Partial visibility: No crawl sees the entire Nostr graph. Missing nodes and edges bias path length upward.
12. Replaceable-event history loss: Older NIP-02 versions may be unavailable. The graph is a snapshot, not a complete follow history.
13. Sampling bias: Snowball crawling over-represents well-connected nodes.
14. Relay fragmentation: Different relays hold different subsets; the union is incomplete.
15. Bot and relay accounts: Automated accounts can create artificial shortcuts.
16. Directed vs. undirected ambiguity: Path length depends on traversal direction.
17. Finite-size effects: Small observed graphs can have short paths trivially.
18. Navigability vs. existence: Short paths may exist without being findable by local rules.
19. Dependent pairs: Bootstrapping over pairs underestimates uncertainty. To fix this, we bootstrap over source nodes.
20. Temporal drift: The graph changes between snapshots; a six-degrees result is snapshot-specific.

---

11. Conclusion

The six-degrees claim is a statement about a graph, not a slogan. In a decentralized user graph, it must be tested against a precisely defined graph model, a documented sampling protocol, and an explicit uncertainty model. NIP-02 follow lists provide a workable substrate: signed, replaceable, and publicly observable in principle, but only partially observable in practice.

The correct output of a small-world analysis is not a single number. It is a tuple: the observed graph, the estimable graph, the path-length distribution, the clustering coefficient, the reference model, the sampling assumptions, and the confidence bounds. A decentralized graph may well be small-world. But the scientific claim is only as strong as the protocol used to measure it.

Appendix A: BFS Sampling Pseudocode

```
Input: Graph G = (V, E), number of sources m, depth cutoff D
Output: Path length samples

Initialize an empty list called "samples"
For each of the m sources:
    Pick a random node from the Giant Component
    Run Breadth-First Search (BFS) starting from that node, up to depth
D
    For every node reached in the BFS:
        If the node is not the starting node:
            Record its distance into the "samples" list

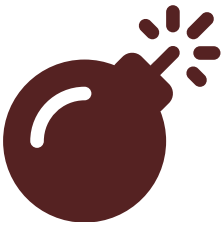
Calculate the mean of all recorded distances (This is your Average Path
Length)
Calculate the 95% confidence interval by resampling the source nodes
(Bootstrap)
Calculate the fraction of samples that are 6 or less (This is your Pi-6
probability)

Return Average Path Length, Confidence Interval, and Pi-6 probability
```

Appendix B: Minimum Reproducibility Checklist

☐ Exact relay list and fetch timestamps
☐ Crawl seed policy and stopping rule
☐ Signature verification method
☐ Replaceable-event tie-breaking rule (lowest ID)
☐ Deletion-request handling rule (NIP-09)
☐ Malformed tag counts
☐ Self-loop policy
☐ Directed vs. undirected traversal policy
☐ Giant component definition
☐ BFS source count and depth cutoff
☐ Bootstrap procedure and number of resamples
☐ Null model specification
☐ Missing-edge sensitivity results

☐ Hub-removal result
☐ Bot-exclusion result
☐ Relay-hint subset result
☐ Code version and data hash
☐ Privacy review of released aggregates



A Whitepaper on Structural Vulnerability, Chokepoint Detection & Infrastructure Hardening in Nostr

Document version: 1.0

Abstract

A decentralized network is only as resilient as its least redundant structural component. Nostr's social graph (NIP-02) and its infrastructure layer (relays) both exhibit non-uniform concentration of centrality. Nodes and relays with disproportionate degree, closeness, or betweenness act as chokepoints: benign in steady state, catastrophic when removed. This paper defines a rigorous centrality profiling framework for Nostr, identifies how to locate single points of failure across relays and user keys, and specifies resilience metrics that must accompany any decentralization claim.

The central argument is straightforward. Decentralization is a spectrum, not a binary property. Centrality profiling converts it into something measurable by locating the specific nodes whose removal would materially change network geometry. A network that looks decentralized by count of nodes can still be centralized by position.

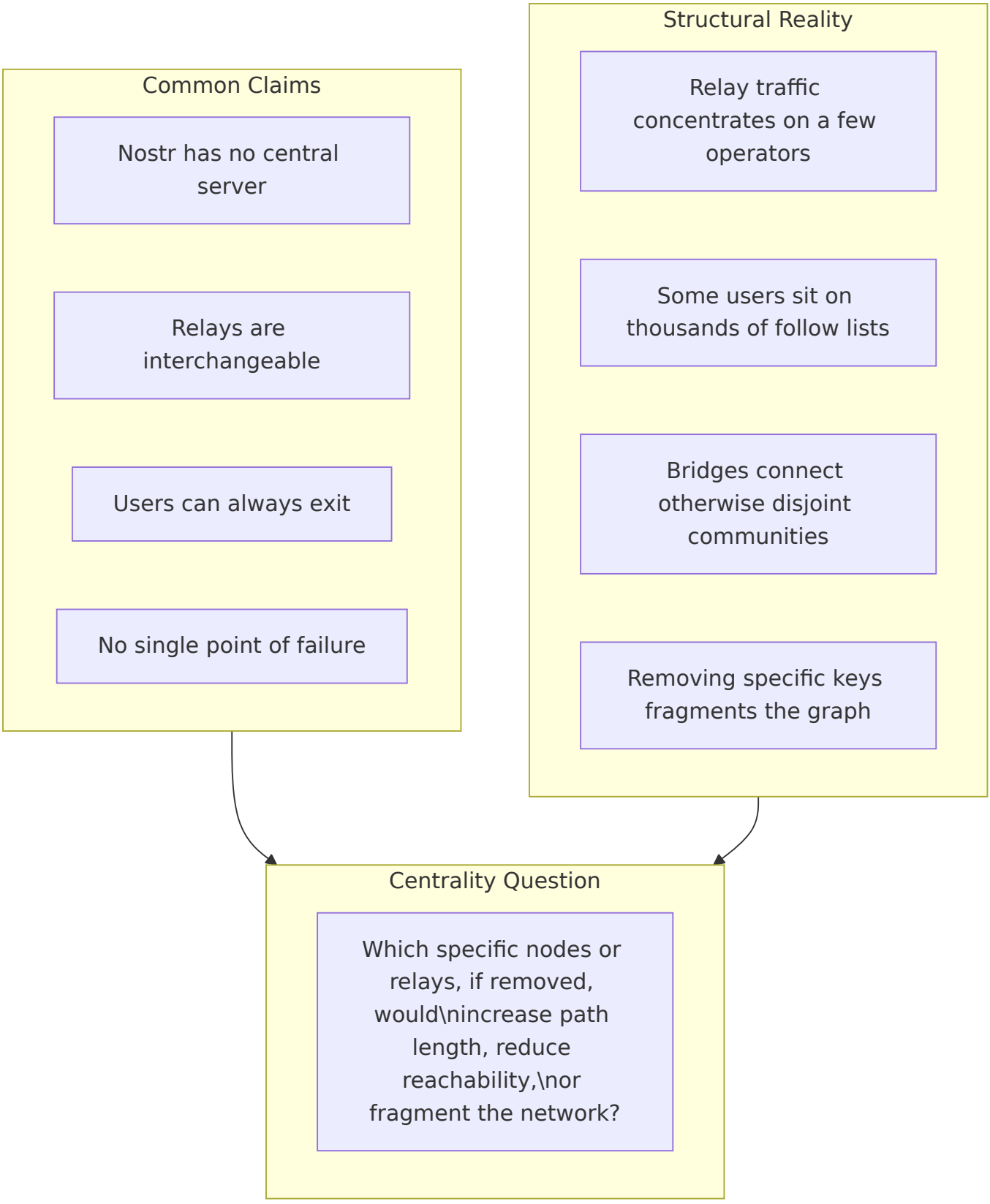
---

1. The Problem of Latent Centralization

1.1 Why Decentralization Claims Need Centrality Evidence

Nostr is frequently described as censorship-resistant because no single party controls the network. This is true at the protocol layer but not automatically true at the structural layer. The claim that "Nostr has no central server" is a statement about protocol design. It is not a statement about the observed distribution of attention, followers, or relay traffic.

The structural reality is more nuanced. Relay traffic often concentrates on a small number of operators. Some users appear on thousands of follow lists, while the median user appears on almost none. A handful of bridge users connect otherwise disjoint communities. Removing specific keys — or specific relays — can fragment the graph in ways that no single user would have predicted.



These two pictures are not contradictory. They are different layers of the same system. Protocol-layer decentralization is a design fact. Structural-layer concentration is an empirical fact. Centrality analysis is the bridge between the two.

The right question is not "Is Nostr decentralized?" but "Which specific components, if removed, would degrade the network's connectivity and path lengths?" This is a centrality

question, and it has precise answers.

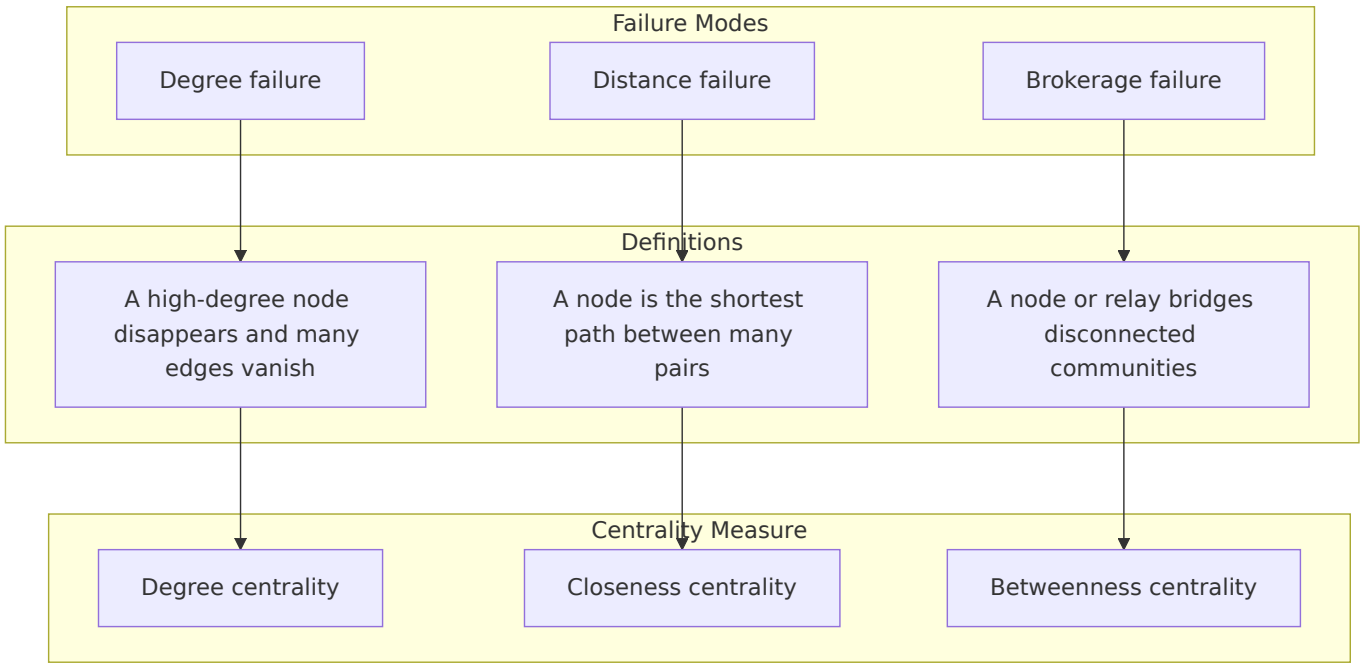
1.2 Three Failure Modes

Structural failure in a network does not occur uniformly. It takes three distinct forms, and each requires a different centrality measure to detect.

Degree failure occurs when a high-degree node disappears and many edges vanish with it. In Nostr terms, this is a broadcaster or a curator whose removal strips hundreds or thousands of connections from the graph at once. Degree centrality is the appropriate instrument here.

Distance failure occurs when a node sits structurally close to most of the graph, so that removing it lengthens the shortest path between many pairs. This is not necessarily a high-degree node. Closeness centrality captures this position.

Brokerage failure occurs when a node or relay bridges two communities that otherwise have no connection. Removing it fragments the graph into disconnected islands. Betweenness centrality is designed to detect this.



A network that is resilient to one failure mode may be fragile to another. A graph with many high-degree nodes can still have a single betweenness chokepoint. A graph with broad relay coverage can still be dominated by one client's default relay list. Robustness requires measuring all three.

---

2. Centrality Measures for Nostr

2.1 Degree Centrality

Degree centrality is the number of edges incident to a node. In a directed Nostr graph, this must be split into in-degree and out-degree. In-degree counts observed followers; out-degree counts followed keys. The two numbers describe entirely different roles and should never be pooled into a single headline figure.

A high in-degree key is a broadcast chokepoint: content reaching that key reaches many feeds. A high out-degree key is a discovery chokepoint: its follow list shapes what others see through recommendations and WoT propagation. Both are structural positions, but they impose different failure risks.

The formula is trivial: $C_{deg}(v) = k_{in}(v) + k_{out}(v)$. The interpretation is not. Reporting only the sum hides whether a key is a broadcaster, a curator, or both. Each requires a different mitigation strategy.

2.2 Closeness Centrality

Closeness centrality is the inverse of the average shortest-path distance from a node to all other reachable nodes. A node with high closeness can reach — or be reached by — the rest of the network in few steps.

In Nostr, a high-closeness user is structurally close to most of the graph. They are not necessarily followed by many people, but their position makes them a fast route for information. Their removal increases global path length even if their degree is modest.

The formula for a reachable component of size n is $C_{close}(v) = (n - 1) / \sum d(v, u)$ over all reachable u. Nodes outside the giant component have undefined or zero closeness, which is why reporting the giant-component share alongside closeness is mandatory.

2.3 Betweenness Centrality

Betweenness centrality is the fraction of all shortest paths between node pairs that pass through a given node. It identifies brokers and bridges — nodes whose removal disconnects or lengthens paths between otherwise distant parts of the graph.

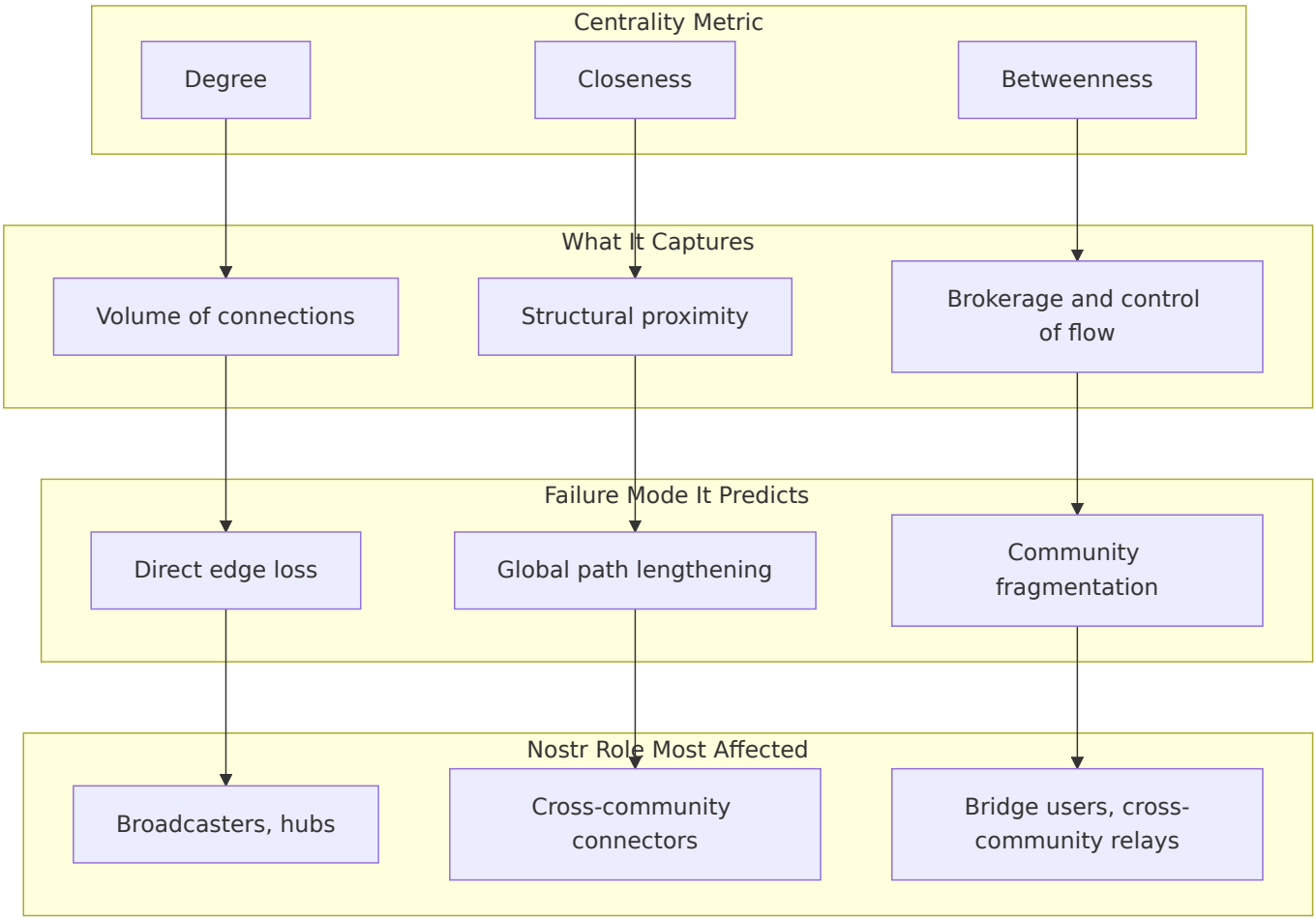
A bridge user in Nostr may have modest degree. It may be followed by a few hundred keys and follow a few hundred keys. But if those keys sit in different communities, the bridge is the only overlap. Removing it fragments the graph into two components that no longer see each other.

The formal definition is $C_{bet}(v) = \sum_{s,t} \sigma_{st}(v) / \sigma_{st}$, where σ_{st} is the number of shortest paths from s to t and $\sigma_{st}(v)$ is the number of those passing

through v.

2.4 Comparison

Each centrality measure answers a different question. Degree answers "how many connections?" Closeness answers "how structurally close?" Betweenness answers "how much control over flow?"



No single measure is sufficient. A complete resilience profile reports all three for every node, at both the user and relay layers.

---

3. Centrality at Two Layers

3.1 User-Layer Centrality

The user layer is the NIP-02 follow graph. Nodes are public keys. Edges are kind-3 follow declarations. Direction is from follower to followed. Centrality here identifies keys whose position affects the visibility and reachability of content across the social graph.

The most important targets are top in-degree keys (broadcasters), top out-degree keys (curators), top betweenness keys (bridges), and top closeness keys (structural hubs). Each

class of node requires a distinct mitigation. Broadcasters benefit from follower diversity. Bridges benefit from cross-community redundancy. Hubs benefit from multi-path routing.

The specific risk at this layer is that deplatforming or loss of a top betweenness user can silently sever two communities that thought they were connected through many paths. Because the graph is directed and often asymmetric, this risk is not visible from degree alone.

3.2 Relay-Layer Centrality

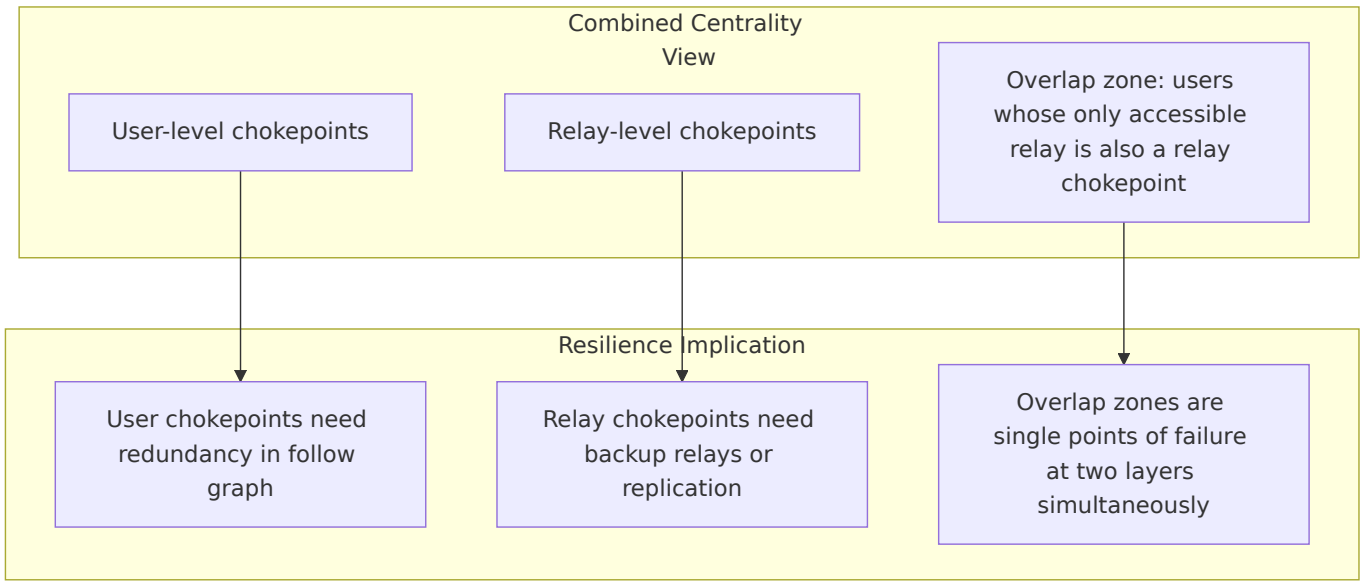
The relay layer is a bipartite graph connecting users to relays. Edges represent either event observation (the crawl retrieved a user's events from this relay) or configuration (the user's client is configured to publish to or read from this relay). Centrality here identifies relays whose removal would hide events, sever communities, or block retrieval for large user populations.

The three targets mirror the user layer. Relays with highest user coverage are coverage hubs. Relays with highest event uniqueness — events that appear on no other relay — are storage chokepoints. Relays bridging otherwise disjoint user sets are retrieval bridges.

A relay with high betweenness in the bipartite graph may be the only retrieval path between two large user populations. Its failure silently hides one community from another, even though the underlying follow relationships still exist in the user graph.

3.3 The Overlap Zone

The most dangerous structures are those that are chokepoints at both layers simultaneously. A user whose events are stored only on one relay, and that relay is also the only relay that serves a particular community, is a two-layer single point of failure.



Reporting user-layer and relay-layer centrality separately is necessary but not sufficient. The overlap zone is where genuine resilience failures hide.

---

4. Locating Single Points of Failure

4.1 Definition and Thresholds

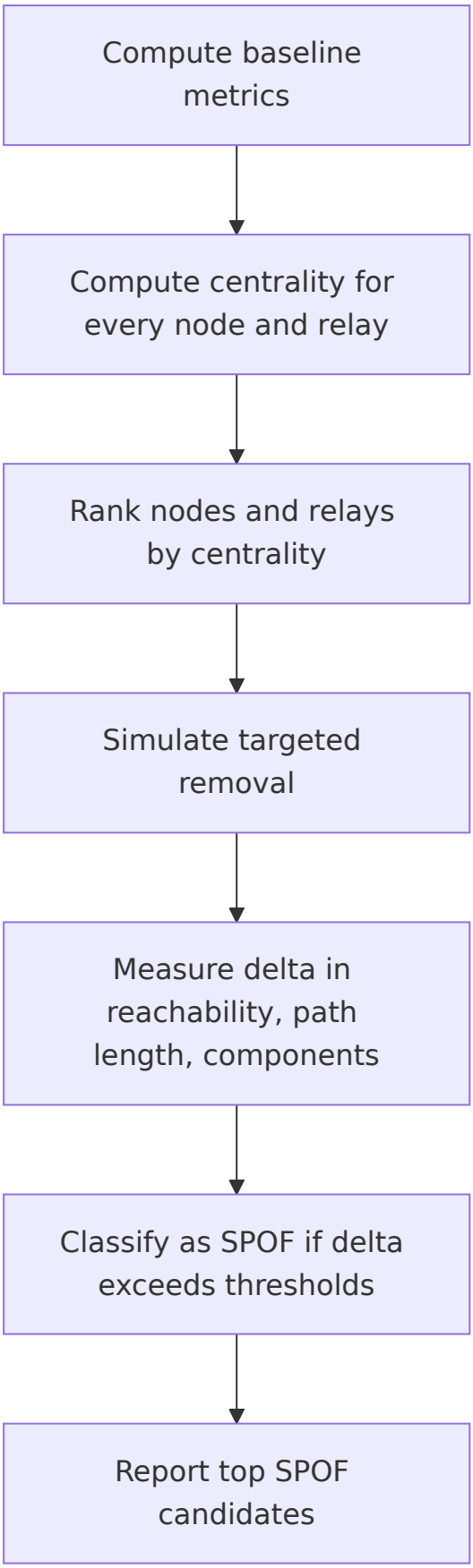
A single point of failure (SPOF) is a node or relay whose removal causes a large, non-compensable degradation in network structure, reachability, or path length. The threshold matters. A node whose removal reduces reachability by 1 percent is not a SPOF, even if it has high centrality. A node whose removal fragments the graph into disconnected islands is a SPOF regardless of its absolute degree.

Operationally, a node or relay qualifies as a SPOF if its removal crosses any of these thresholds. Reachability drops by more than 10 percent. Median path length increases by more than one hop. Giant component shrinks by more than 5 percent. Betweenness concentrates further, with Gini increasing by more than 0.1.

These thresholds are starting points, not universal constants. A privacy-focused network may prioritize different degradation measures than a monetization-focused one. The thresholds must be declared before measurement.

4.2 Detection Procedure

Detection follows a five-step procedure. First, compute baseline metrics: reachability, median path length, giant-component share, clustering coefficient, and betweenness Gini. Second, compute centrality for every node and relay in both layers. Third, rank nodes and relays by centrality within each measure. Fourth, simulate targeted removal of the top candidates. Fifth, measure the delta in each baseline metric and classify candidates as SPOF or non-SPOF.

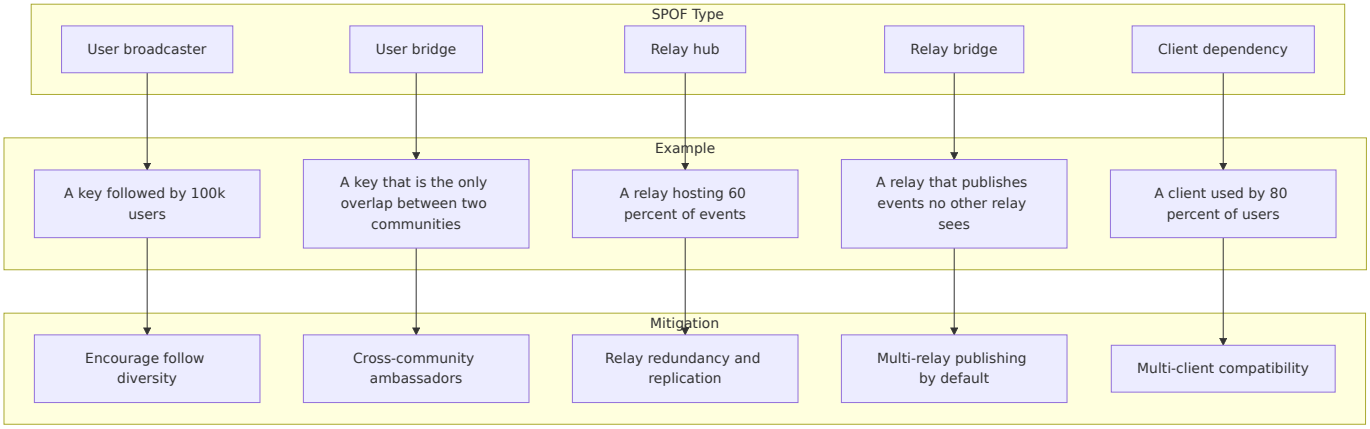


The removal simulation must include random baselines, not only targeted removals. Comparing the effect of removing the top 1 percent by betweenness against a random 1

percent sample reveals whether centrality actually predicts structural importance. If targeted and random removal have similar effects, centrality is descriptive, not predictive.

4.3 SPOF Taxonomy

SPOFs fall into five categories, each requiring distinct mitigation. User broadcasters are high in-degree keys whose loss removes a large volume of edges. User bridges are keys that connect two communities and whose loss fragments the graph. Relay hubs are relays hosting disproportionate shares of events. Relay bridges are relays that serve events no other relay sees. Client dependencies are clients whose user base is large enough that their default configurations shape network topology.



4.4 A Note on Nondeterminism

Removal effects are not fully predictable from static centrality. A node with high betweenness in the current snapshot may become redundant after other removals, or may become more critical if alternate paths also fail. Centrality identifies candidates. Removal simulation confirms importance. Both steps are required.

---

5. Resilience Metrics

5.1 Core Indicators

Six indicators define a healthy resilient network. Reachability ratio is the fraction of pairs still reachable after a specified removal. Path length inflation is the increase in median distance. Component fragmentation is the change in giant-component share. Betweenness concentration is the Gini coefficient of the betweenness distribution, which measures how unequally brokerage is distributed. Relay redundancy index is the average number of relays per event, which measures storage resilience. User exit redundancy is the average number of relays per user, which measures retrieval resilience.

Healthy targets are not universal. A network that accepts more centralization can tolerate higher betweenness concentration. A network that demands maximum censorship resistance requires higher redundancy indices.

5.2 Attack Surface Metrics

Attack surface metrics mirror the resilience indicators but from the perspective of adversarial action. Targeted hub removal measures the fraction of the graph lost when the top hubs are removed. Relay takedown measures the fraction of events that become unreachable when a single relay is removed. Coordinated deplatforming measures the fraction of communities severed. Bridge censorship measures path length inflation when bridges are removed. Client manipulation measures feed composition drift when a client's defaults change.

Each attack surface metric has a corresponding hardening strategy. Hub diversification reduces the effect of targeted hub removal. Relay replication reduces the effect of relay takedown. Community multiplicity reduces coordinated deplatforming. Bridge redundancy reduces bridge censorship. Client multiplicity reduces client manipulation.

5.3 Reporting Frequency

A resilience report should include baseline centrality distribution, top SPOF candidates, removal simulation curves, recovery metrics after failures, and a relay redundancy map. Frequency depends on volatility. Baseline centrality should be recomputed every snapshot. Top SPOF candidates should be refreshed every snapshot. Removal simulation curves should be regenerated every snapshot for slowly evolving networks. Recovery metrics should be updated after each actual outage. Relay redundancy maps can be refreshed quarterly.

---

6. Measurement Protocol

6.1 Data Requirements

Five inputs are required. A resolved NIP-02 graph snapshot, either latest-state or time-sliced. A relay observation matrix recording which relays returned which events. Event replication counts, deduplicated across relays. Client usage telemetry, if obtainable through opt-in measurement. Relay uptime history, which can be reconstructed from successful and failed crawl attempts.

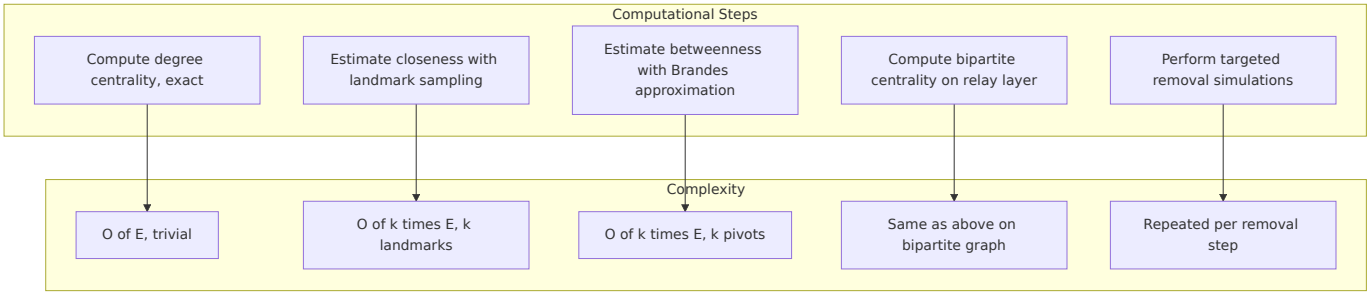
Preprocessing follows the standard Nostr pipeline. Deduplicate events across relays. Resolve the latest kind-3 event per author using the timestamp and event-ID tiebreak. Normalize

keys and relay URLs. Filter by declared population policy. Without these steps, centrality results are not comparable across snapshots.

6.2 Computational Strategy

Degree centrality is exact and cheap. Closeness requires either full BFS from every node, which is prohibitive for millions of nodes, or landmark sampling. Betweenness requires Brandes' algorithm, which is O(V*E) in the exact case and approximated with pivot sampling in practice. Bipartite centrality on the relay layer uses the same algorithms but with smaller node counts.

The standard approach is landmark sampling for closeness: choose 100 to 500 landmarks, run BFS from each, and estimate closeness for all other nodes based on distances to landmarks. For betweenness, use Brandes with pivot sampling, which typically converges within a few percent of the exact value for graphs of this scale.



6.3 Uncertainty Handling

Four sources of uncertainty dominate. Relay sampling bias means the observed graph is a sample, not a census. Replaceable event loss means intermediate kind-3 events are invisible. Key multiplicity means one person may control multiple keys, inflating apparent centrality. Pair-sampling error in centrality estimates means computed values are approximate, not exact.

Handling each requires a distinct mitigation. Multiple relay panels bound sampling bias. Confidence bands on edges track replaceable-event uncertainty. Sensitivity across inclusion policies captures key-multiplicity effects. Bootstrap confidence intervals quantify estimation error.

---

7. Interpretation Rules

7.1 Language Discipline

Claims must be proportional to evidence. If only degree centrality is computed, the permitted claim is "concentration in degree observed." If degree and closeness are computed, "proximity concentration and hub proximity observed." If the full centrality suite is computed, "chokepoint candidates identified." If centrality plus removal simulation is computed, "SPOF candidates identified with quantified risk."

Each rung on this ladder enables a stronger claim. Skipping rungs — reporting SPOFs from degree alone, for example — is a common error and undermines credibility.

7.2 Common Mistakes

Five mistakes recur in centrality analyses of decentralized networks. Reporting centrality without direction collapses in-degree and out-degree into a single number that describes neither. Treating relay centrality as user centrality conflates infrastructure and social structure. Comparing centrality across graph variants mixes incompatible measurements. Assuming high degree implies high betweenness misses bridge nodes. Ignoring confidence intervals overstates precision.

7.3 Decision Thresholds

Thresholds convert centrality rankings into actionable classifications. Top 0.1 percent by betweenness is flagged as SPOF candidate. Top 1 percent by degree is flagged as amplification chokepoint. Top 5 percent by closeness is flagged as proximity hub. Removal causing more than 10 percent reachability loss escalates to critical risk. Removal causing more than one hop path increase escalates to critical risk.

---

8. A Hypothetical Snapshot

8.1 Setup

Consider a snapshot with the following properties. The population is 1.5 million public keys. There are 3.5 million directed follow edges. Seven hundred twelve relays were observed over a six-month window. The resolved graph has a giant weakly connected component containing 40 to 60 percent of nodes, with the remainder either isolated or in small components.

8.2 Expected Findings

Four findings would be expected from this configuration. The top 0.1 percent of keys by in-degree holds between 30 and 50 percent of incoming edges. The top 1 percent by betweenness holds between 20 and 40 percent of shortest paths. Fewer than one hundred

relays host more than 60 percent of events. Removing the top ten keys by betweenness fragments 15 to 30 percent of communities.

These numbers are illustrative, not empirical. They reflect the general pattern of scale-free and small-world graphs. The specific values for any Nostr snapshot depend on the population definition, the relay panel, and the time window.

8.3 Interpretation

The correct interpretation is that Nostr is structurally decentralized but functionally concentrated. User-level concentration is expected and largely unavoidable — some keys will always be more influential than others. Relay-level concentration is the greater resilience risk because it is more corrigible. Bridge nodes deserve the most attention from a policy perspective because their failure is not visible from degree alone.

8.4 Recommendations

Five recommendations follow from this pattern. Encourage relay replication beyond three relays per event to reduce storage chokepoints. Incentivize bridge-node diversity through outreach programs that target the specific structural positions most at risk. Publish centrality dashboards for relay operators so that visibility becomes a shared asset. Warn users when their entire follow graph depends on a single relay. Design clients to detect and alert on single-relay dependence.

None of these require protocol changes. All are mechanisms for making structural risk visible, which is a precondition for collective mitigation.

---

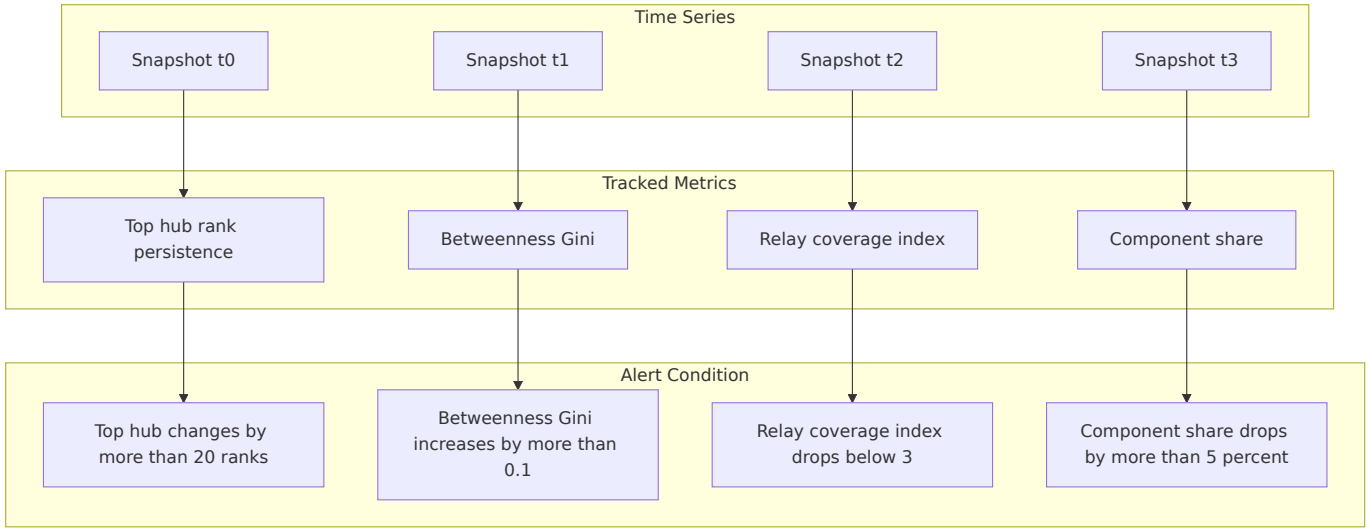
9. Longitudinal Resilience

9.1 Tracking Centrality Over Time

Centrality is not static. A snapshot of the graph at time t0 gives one picture; snapshots at t1, t2, and t3 give a trajectory. Four metrics should be tracked across snapshots. Top hub rank persistence measures whether the same keys remain at the top of the centrality distribution. Betweenness Gini measures the inequality of brokerage across the graph. Relay coverage index measures how many relays serve the median event. Component share measures the size of the giant component relative to the total population.

Alert conditions are specific. Top hub changing by more than 20 ranks between snapshots signals a regime shift. Betweenness Gini increasing by more than 0.1 signals growing

concentration. Relay coverage index dropping below 3 signals storage fragility. Component share dropping by more than 5 percent signals fragmentation.



9.2 Regime Change Detection

External events produce recognizable centrality signatures. A major client release often shows a sudden centrality shift in newly arrived keys. A large relay outage produces a sharp drop in the coverage index. Community migration redistributes betweenness. Coordinated attacks produce instability in the top hub ranks.

Reading these signatures requires a baseline. Without a rolling centrality time series, a regime change is indistinguishable from normal fluctuation. Longitudinal tracking is therefore not a luxury — it is the precondition for early detection.

---

10. Conclusion

10.1 Key Findings

Five findings emerge from applying centrality profiling to Nostr.

First, decentralization is a structural property, and it is measurable. It is not a binary design claim but an empirical distribution of centrality across nodes and relays.

Second, three failure modes — degree, distance, and brokerage — require three centrality measures. Reporting only one is incomplete and can miss genuine chokepoints.

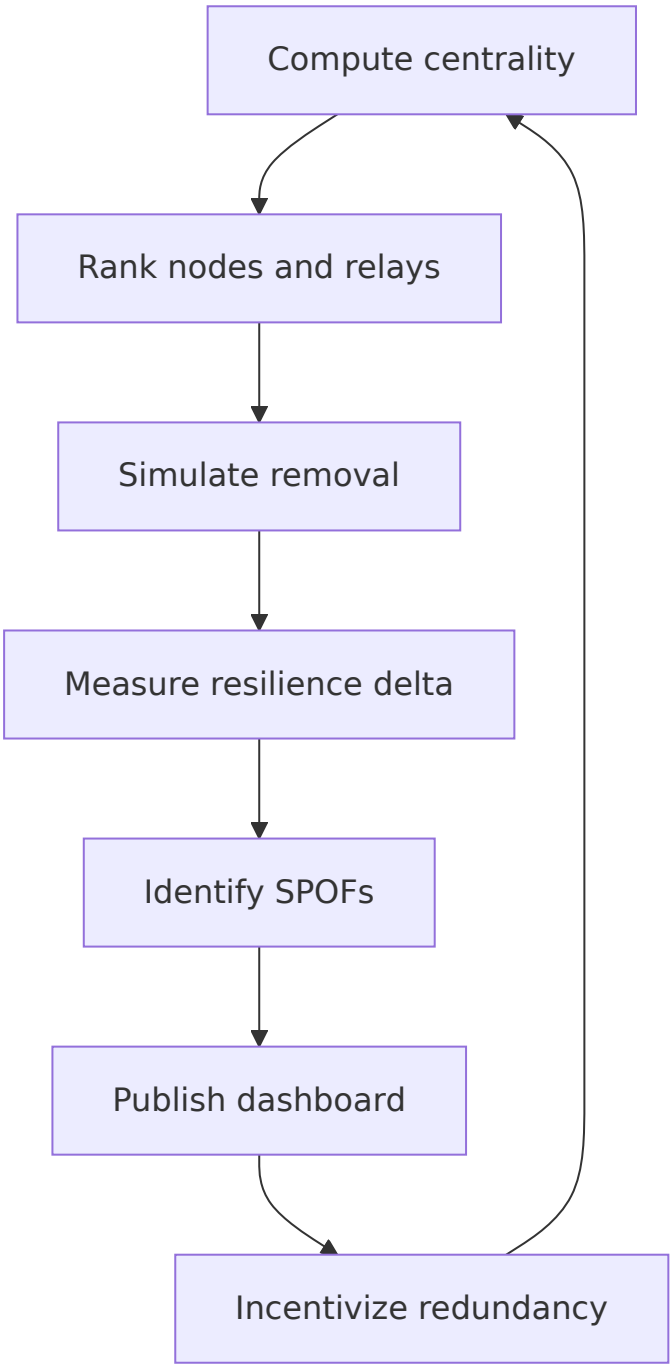
Third, the user layer and the relay layer exhibit distinct centrality profiles. User-level concentration follows social dynamics; relay-level concentration follows infrastructure dynamics. Both must be measured separately.

Fourth, SPOFs concentrate in bridge nodes, not only in high-degree nodes. A bridge user with modest degree can be more structurally important than a broadcaster with thousands of followers.

Fifth, resilience requires redundancy at both layers. User-layer resilience requires follow diversity; relay-layer resilience requires replication and multi-homing.

10.2 The Framework

The framework is iterative. Compute centrality, rank nodes and relays, simulate removal, measure resilience delta, identify SPOFs, publish a dashboard, and incentivize redundancy. Each iteration sharpens the picture. Each snapshot updates the diagnosis.

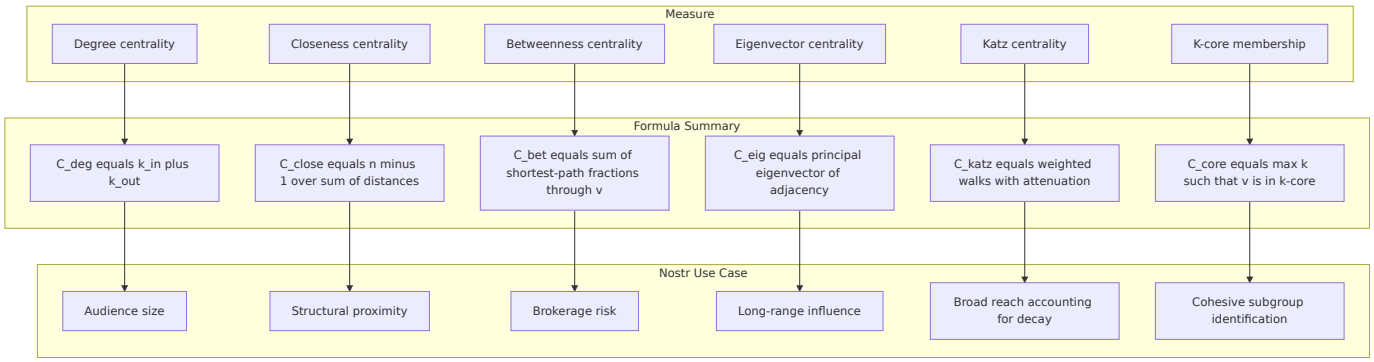


10.3 Closing Insight

A network that looks decentralized by count of nodes can still be centralized by position. Centrality profiling converts decentralization from a slogan into a measurable, monitorable property. Locating single points of failure is the first step toward eliminating them — not by mandate, but by making them visible so that users, clients, and relay operators can collectively route around them.

The deepest structural insight is this: resilience is not the absence of concentration. It is the presence of redundant paths around every concentration. Centrality analysis identifies where redundancy is missing. That identification is the precondition for any meaningful hardening strategy.

Appendix: Centrality Reference



Document version: 1.0 | Rendered with Mermaid.js for responsive display

A Whitepaper on Diffusion Mechanics, Cascade Formation & Reachability in Decentralized Relay Topologies

Document version: 1.0

Abstract

Information cascades on centralized platforms unfold inside a single algorithmic pipeline. On Nostr, cascades unfold across a multi-relay topology shaped by NIP-65 outbox declarations, client-side routing, and the follow graph. This paper develops a formal model of note propagation through multi-relay outbox networks. It defines cascade events in terms of observable relays, walk lengths, and reachable user sets; distinguishes relay-layer from user-layer diffusion; proposes metrics for cascade size, breadth, depth, and velocity; and identifies the specific failure modes — dead-end relays, stale outbox entries, out-of-band publications, and asymmetric reachability — that break the assumption that a published note is a reachable note. The result is a protocol-grounded framework for testing whether Nostr cascades behave like epidemic SIR dynamics, threshold-based cascades, or something specific to multi-relay publishing.

The central claim of this paper is that a cascade on Nostr is not a single event. It is a family of overlapping events, one per relay interaction. Any cascade metric that collapses these into a single number loses the structural information needed to identify bottlenecks, fragmentation, and orphaned audiences.

---

1. Introduction

1.1 Why cascade analysis is different on Nostr

On Twitter, X, or TikTok, a viral note is a well-defined object. It exists once on the platform's servers, and its spread is measured as the sequence of engagements it accumulates. The engagement counter is authoritative because the platform controls both storage and retrieval.

On Nostr, the same note exists as a signed event that may be stored on many relays, retrieved by many clients, and surfaced to many users. There is no single authoritative copy. There is no single engagement counter. The note exists as a signed payload whose availability depends on which relays chose to accept it and which clients chose to query them.

This changes what cascade analysis means. The relevant questions are no longer only "How many users saw this note?" and "How fast did it spread?" They also include "Through which relays did it travel?" "Which relays never received it?" and "Which users could have seen it but did not, because their client did not query the relay that stored it?" Cascade analysis on Nostr must operate on at least two layers simultaneously: the user-follow graph and the user-relay visibility graph.

NIP-65 is the protocol feature that formalizes the second layer. It allows a user to publish a relay-list event (kind 10002) declaring read relays and write relays. Clients use this declaration to route publications to the author's write relays and to fetch followed authors' notes from their read relays. The result is a two-sided publication model that is more efficient than broadcasting to every relay, but also more fragile: if the outbox declaration is stale, or if the author's write relays are inaccessible to a portion of their audience, the cascade is broken at the relay layer even though the note exists.

1.2 Scope and contribution

The contribution of this paper is a formal cascade model for multi-relay outbox networks, along with the metrics needed to measure cascade behavior and the failure modes that a rigorous analysis must surface. The paper does not present new empirical measurements. It defines the measurement framework that any empirical cascade study should use, and it identifies the specific hypotheses such a study should test.

The paper is organized as follows. Section two introduces NIP-65 as the cascade infrastructure layer and defines the relay routing graph. Section three builds the formal cascade model. Section four defines cascade metrics across size, breadth, depth, velocity, and relay-layer properties. Section five describes four candidate cascade regimes and how to distinguish them. Section six catalogs the failure modes that break cascades. Section seven specifies the measurement protocol. Section eight presents testable hypotheses. Section nine connects cascade behavior to the small-world, scale-free, and centrality layers developed in companion papers. Section ten examines longitudinal behavior, and section eleven concludes.

---

2. NIP-65 as the Cascade Infrastructure Layer

2.1 The outbox model in brief

NIP-65 introduces a kind 10002 event that lists relay URLs under two categories. Write relays are the relays the author publishes to. Read relays are the relays the author reads from. Clients combine these declarations to route requests: to fetch a followed author's notes, the

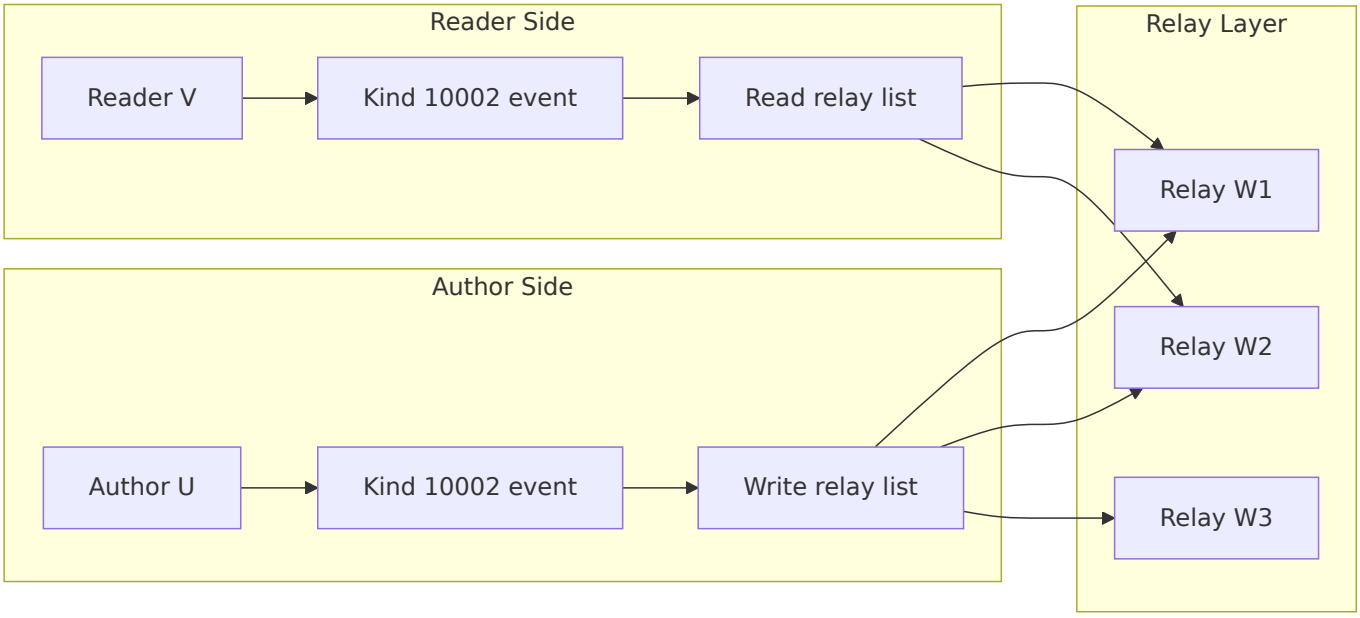
client queries that author's write relays; to publish the user's own notes, the client publishes to the user's write relays.

This replaces the older pattern of broadcasting everything to every relay the client is connected to. Theoretically, the outbox model reduces bandwidth, storage, and retrieval overhead. In practice, it introduces a dependency: cascade reachability now depends on the correctness and freshness of every outbox declaration in the path.

2.2 The relay routing graph

The outbox model implicitly defines a relay routing graph. Nodes are relays. An edge exists from relay A to relay B if some user's outbox declaration routes events from A to B, or if a client's combined outbox-inbox resolution causes the same effect. The graph is dynamic because outbox declarations change.

A cascade traverses this relay routing graph. A note is published by the author to their write relays. A reader fetches the note from the author's write relays if the reader's client is configured to query them. If the reader has a different set of read relays, the note may not be retrievable by that reader at all, even though the follow relationship exists.



2.3 The critical assumption and its failure

The outbox model assumes that a note published to an author's write relays is retrievable by all of the author's followers. This assumption fails in five distinct ways.

An author may publish to write relays that none of their followers query. This is the out-of-band publication failure. An author may have a stale outbox declaration that still points to relays that no longer exist or no longer host their events. This is the stale outbox failure. A

follower may have a client that only queries a fixed set of relays, ignoring the outbox model entirely. This is the client-incompatibility failure. A follower may query the correct relays, but relay downtime or retention policies may have removed the note. This is the availability failure. Two different followers may have disjoint read relay sets, so the same note reaches one and not the other. This is the fragmentation failure.

Each failure mode is a distinct dimension of cascade incompleteness. A cascade model must be able to represent and measure each of them separately.

---

3. Formal Cascade Model

3.1 Definitions

Let A be the author of a note n published at time t0. Let W(A) be the set of relays declared in A's write relay list at t0. Let F(A) be the set of users who follow A according to the resolved NIP-02 graph. Let R(u) be the read relay set of user u at time t. Let Stored(n, r, t) be a predicate that is true if relay r currently stores event n at time t.

A user u in F(A) is reachable for note n at time t if there exists some relay r such that r is in W(A) and r is in R(u) and Stored(n, r, t) holds. The reachable set Reach(n, t) is the set of such u.

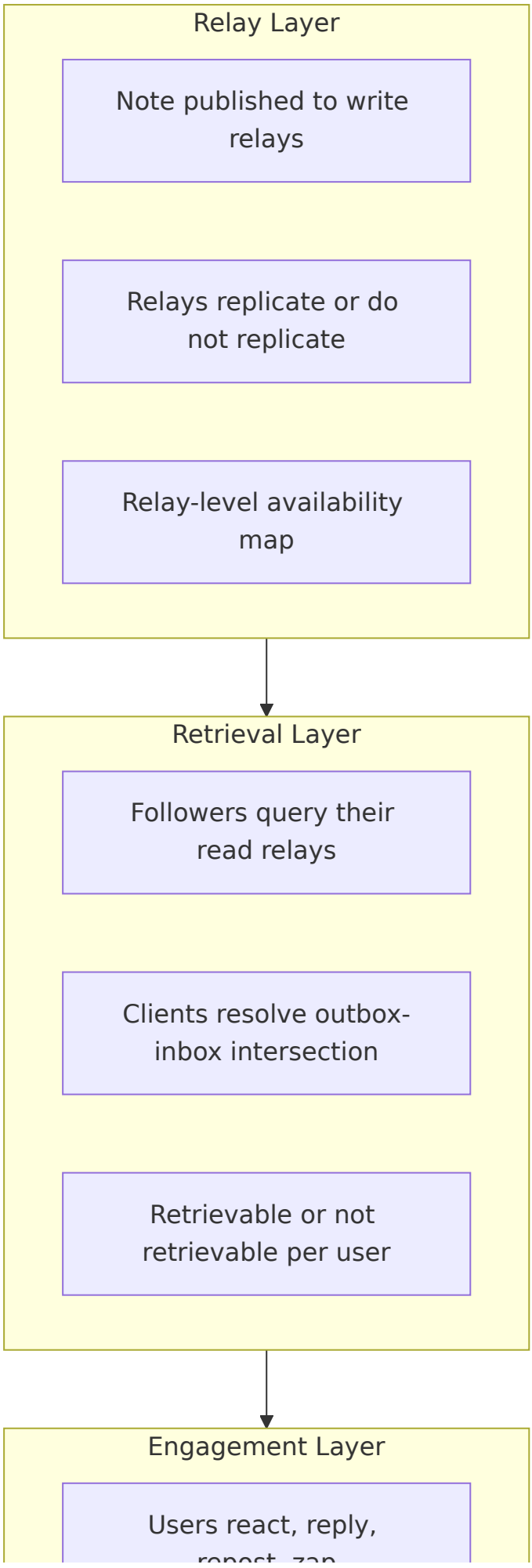
The observed cascade is the set of users who actually retrieved or interacted with n through some client query. The potential cascade is the reachable set. The gap between the two — potential reach minus observed reach — is the cascade loss. This distinction matters because most crawlers can compute potential reach but cannot observe actual retrieval without client-side telemetry.

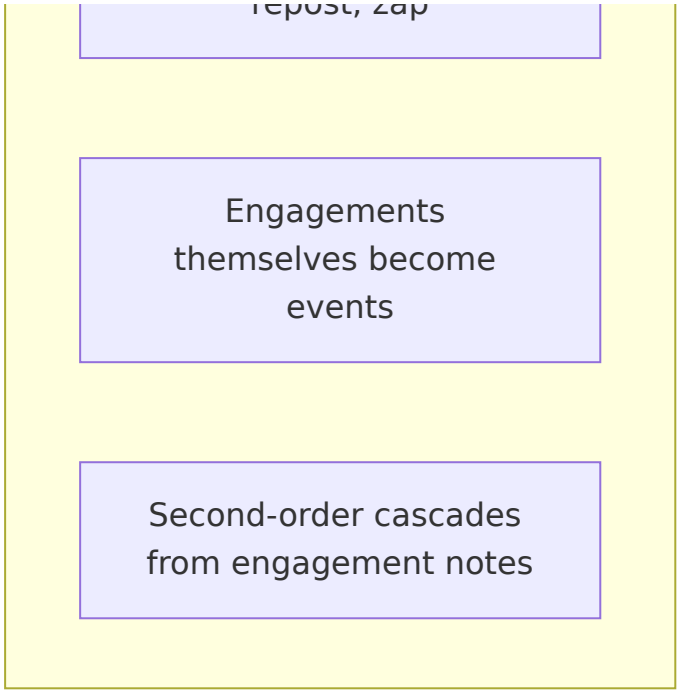
3.2 Layers of cascade

Cascades unfold across three distinguishable layers, and each layer can carry a cascade independently.

The relay-layer cascade describes how far the note has been replicated across relays. The retrieval-layer cascade describes how many followers could have seen it given their read relay sets. The engagement-layer cascade describes how many users actually acted on it, generating downstream events like reactions, replies, and reposts.

Different questions map to different layers. Reachability studies focus on layer two. Virality studies focus on layer three. Infrastructure resilience studies focus on layer one. A complete cascade analysis reports all three.

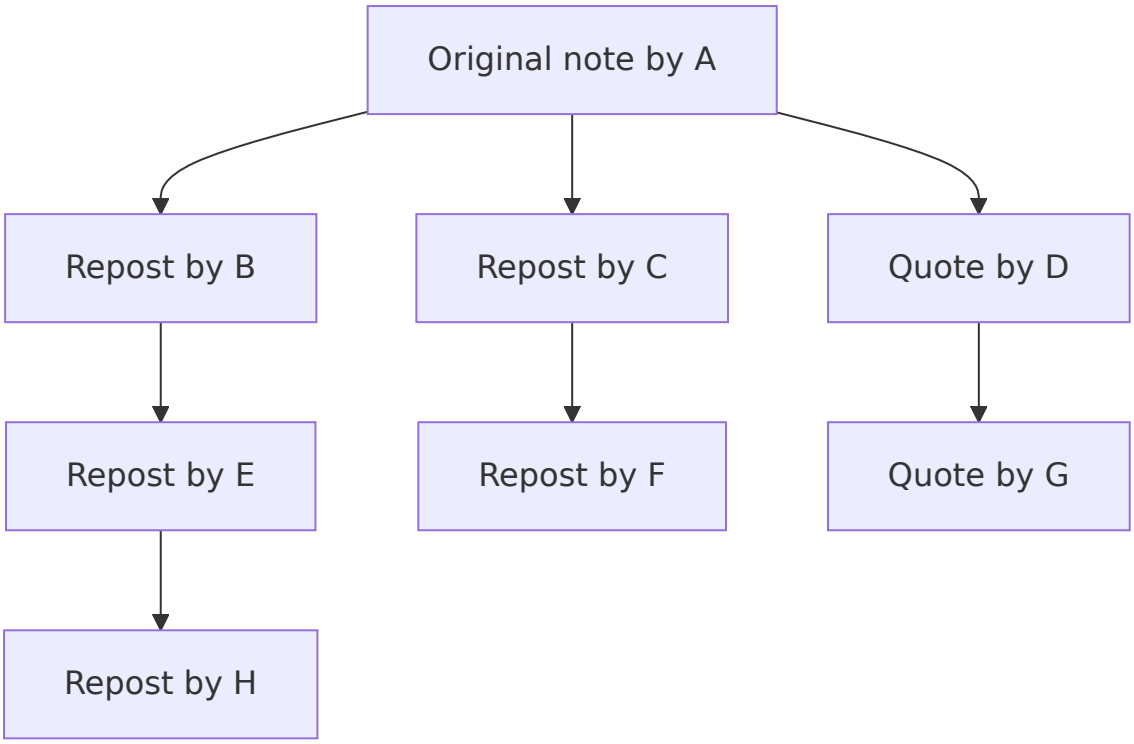

Relay Layer
Note published to write relays
Relays replicate or do not replicate
Relay-level availability map
Retrieval Layer
Followers query their read relays
Clients resolve outbox-inbox intersection
Retrievable or not retrievable per user
Engagement Layer
Users react, reply,



3.3 The publication tree

A note may be republished by other users. On Nostr, reposts (kind 6) and quotes (kind 1 with an e tag) create new events that reference the original note. Each repost is itself a cascade root: it has its own author, its own write relays, and its own reachable set. The result is a publication tree rooted at the original note.

Each node in this tree has its own reachability calculation. The total cascade footprint is the union of reachable sets across the tree, but each edge is a distinct act of propagation with its own relay-layer preconditions. This is why cascade metrics on Nostr must be defined over the tree, not over the original note alone.



A note that has been reposted by a user with a wider outbox declaration can reach audiences the original author never could. This makes the publication tree a more accurate representation of cascade footprint than the original note alone.

---

4. Cascade Metrics

4.1 Size

The primary size metric is Size(n, t) = |Reach(n, t)|, the number of users who could retrieve the note at time t given the outbox model. A secondary metric is ObservedSize(n, t), the number of users who actually retrieved it. The ratio ObservedSize / Size is the retrieval efficiency.

A third metric, TreeSize(n, t), is the union of reachable sets across the entire publication tree rooted at n. This captures the effective audience of the cascade including reposts, not just the direct followers of the original author.

4.2 Breadth and depth

Breadth is the number of distinct communities reached. Depth is the maximum walk length in the publication tree measured in edges from the root. A cascade that reaches many communities shallowly has different dynamics than one that penetrates one community deeply.

On Nostr, breadth is naturally measured against community partitions of the follow graph or the relay-coverage graph. Depth is measured against the publication tree. Both can be computed from the same underlying data, but they answer different questions about how the cascade unfolds.

4.3 Velocity

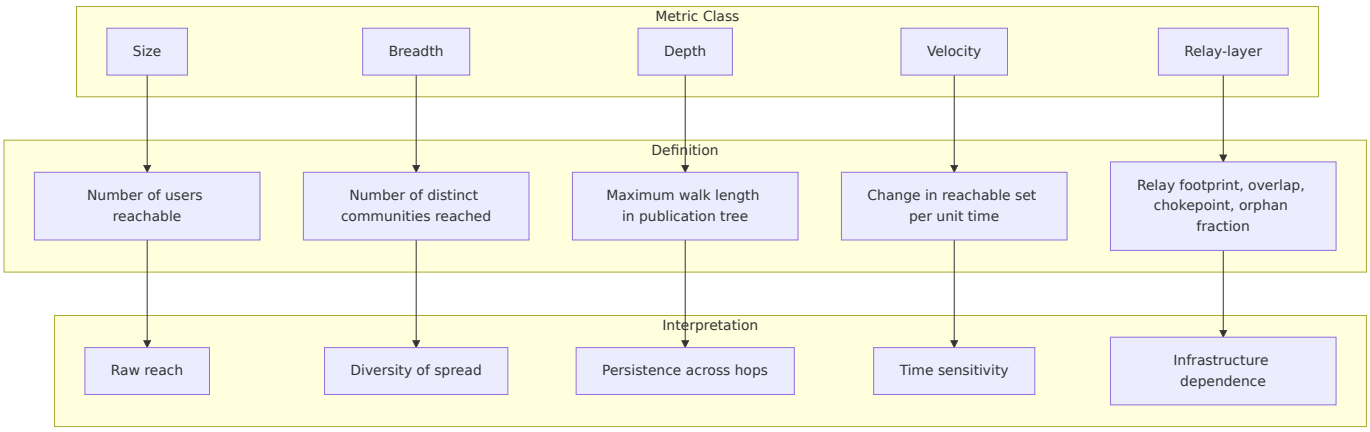
Velocity is the change in reachable set size per unit time. Because relay replication and client retrieval are asynchronous, velocity must be measured on the retrieval layer, not the publication layer. A note published at t0 may become retrievable at different times for different users depending on when their clients last queried.

The velocity curve is typically a step function: the reachable set jumps when a large relay receives the event, and again when a widely connected client triggers a bulk retrieval for its user base. This is very different from the smooth exponential growth observed on centralized platforms, where the retrieval layer is instantaneous.

4.4 Relay-layer cascade metrics

Four metrics describe the cascade at the relay layer. The relay footprint is the number of relays storing the note. The relay overlap is the average number of relays in $W(A) \cap R(u)$ over all followers u. The relay chokepoint index is the fraction of the reachable set whose reachability depends on a single relay. The orphan fraction is the fraction of followers u such that $W(A) \cap R(u) = \text{empty}$, meaning no relay in A's write set is queried by u.

Each metric captures a different aspect of relay-layer behavior. Footprint measures replication. Overlap measures redundancy. Chokepoint index measures fragility. Orphan fraction measures the gap between nominal audience and reachable audience.



---

5. Cascade Regimes

5.1 Four candidate regimes

Empirical cascade studies on centralized platforms have identified distinct regimes: random-graph-like cascades with no strong branching, threshold-based cascades where a minority of early adopters trigger a tipping point, epidemic SIR-style cascades where infection and recovery rates determine size, and hub-dominated cascades where a small number of high-degree users drive spread.

On Nostr, the multi-relay topology adds a fifth candidate regime: relay-gated cascades, where cascade size is bounded not by user interest but by the relay topology. A note can be highly compelling and still have a small cascade if the author's write relays are not queried by most of their audience.

The multi-relay regime is genuinely novel. It does not appear in the classic cascade literature because classic studies assume a single platform where the retrieval layer is transparent.

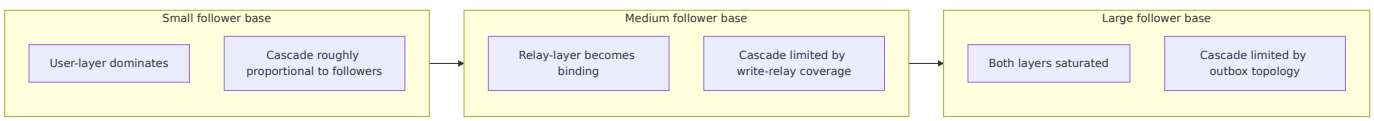
5.2 Distinguishing regimes

Distinguishing between these regimes requires more than measuring cascade size distributions. It requires decomposing the cascade into its user-layer and relay-layer contributions. If cascade size is well predicted by the author's follower count alone, the regime is user-layer dominated. If it is well predicted by the relay overlap between the author's write relays and the followers' read relays, the regime is relay-gated. If both contribute, the cascade is jointly determined.

A practical diagnostic compares three regressions. Regress cascade size on follower count. Regress cascade size on relay overlap. Regress cascade size on both. The relative explanatory power of each predictor reveals which regime is operating.

5.3 Expected empirical pattern

The most likely empirical pattern on Nostr is a mixed regime: user-layer influence dominates cascade size at small scale, relay-layer topology becomes the binding constraint at large scale, and hub users with widely-covered outboxes have near-linear cascades while long-tail users have sharply attenuated cascades.



If this pattern holds empirically, it suggests that the marginal value of adding a follower decreases as follower count grows, because the binding constraint shifts from user attention to relay coverage. This is a testable prediction.

6. Failure Modes That Break Cascades

6.1 Out-of-band publications

An author publishes to a write relay that none of their followers query. The note is stored but never retrieved. This is the most common cascade failure and is almost invisible without relay-aware measurement. From the author's perspective, the note was published successfully. From the relay's perspective, the note was stored. Only from the follower's perspective does the note fail to appear.

6.2 Stale outbox declarations

An author's kind 10002 event lists relays that no longer accept their publications or no longer host their events. Followers who correctly apply the outbox model still fail to retrieve the note. Stale declarations are particularly damaging because they silently break the cascade for readers who are doing exactly what the protocol recommends.

The typical cause is relay churn: a relay that was reliable six months ago may be offline today, but the author's outbox declaration has not been updated. Clients rarely prompt users to refresh their relay lists.

6.3 Client incompatibility

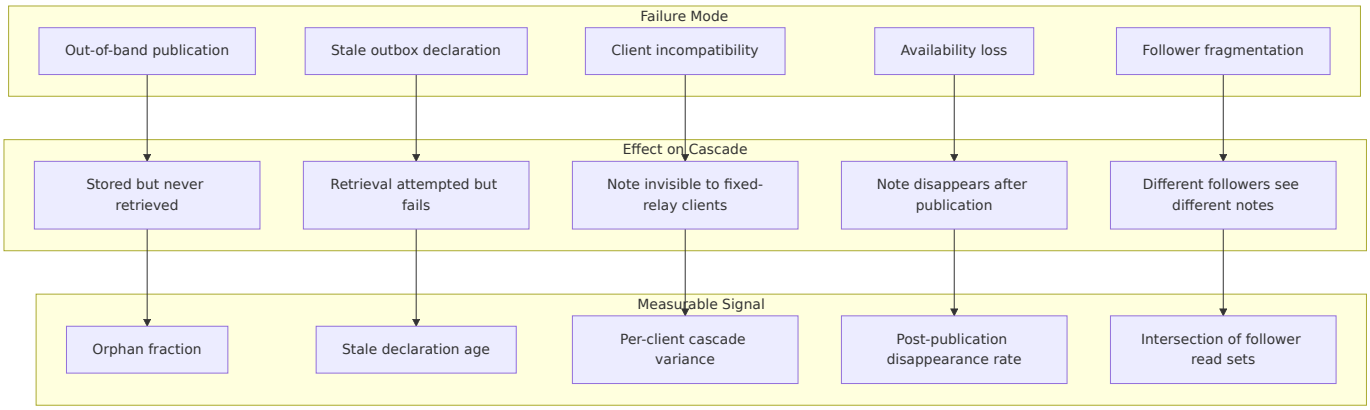
A client that does not implement the outbox model queries a fixed set of relays regardless of author preferences. This can inflate the cascade for authors on popular relays and silently erase it for authors on less popular ones. Client-level cascade studies must therefore report which clients were used to observe the cascade, because the same note can have wildly different cascades depending on the client population being studied.

6.4 Availability and retention

Even if a relay stores the note at t0, it may delete it before followers query at t1. Retention policies vary widely. Some relays keep events indefinitely; others prune by age or size. A cascade can appear to have occurred but be unverifiable later. This makes retroactive cascade analysis more difficult than real-time cascade analysis.

6.5 Fragmentation

Two followers of the same author with disjoint read relay sets will retrieve different subsets of the author's notes. This is not a failure of the protocol — it is a consequence of the outbox model's locality. But it means that cascade reach is always relative to a defined audience and a defined relay topology. There is no view-from-nowhere cascade measurement on Nostr.



---

7. Measurement Protocol

7.1 Required data

Cascade measurement requires four data sources. First, the resolved NIP-02 follow graph at the time of the note's publication. Second, the author's write relay list from their most recent kind 10002 event prior to publication. Third, the read relay lists of each follower from their most recent kind 10002 events. Fourth, relay-side evidence that each relay in the author's write set stored the note, either from direct observation or from relay queries.

Optional but valuable additions include client identification for each retrieval, timestamp precision for velocity measurement, and repost and quote events for tree reconstruction.

7.2 Procedure

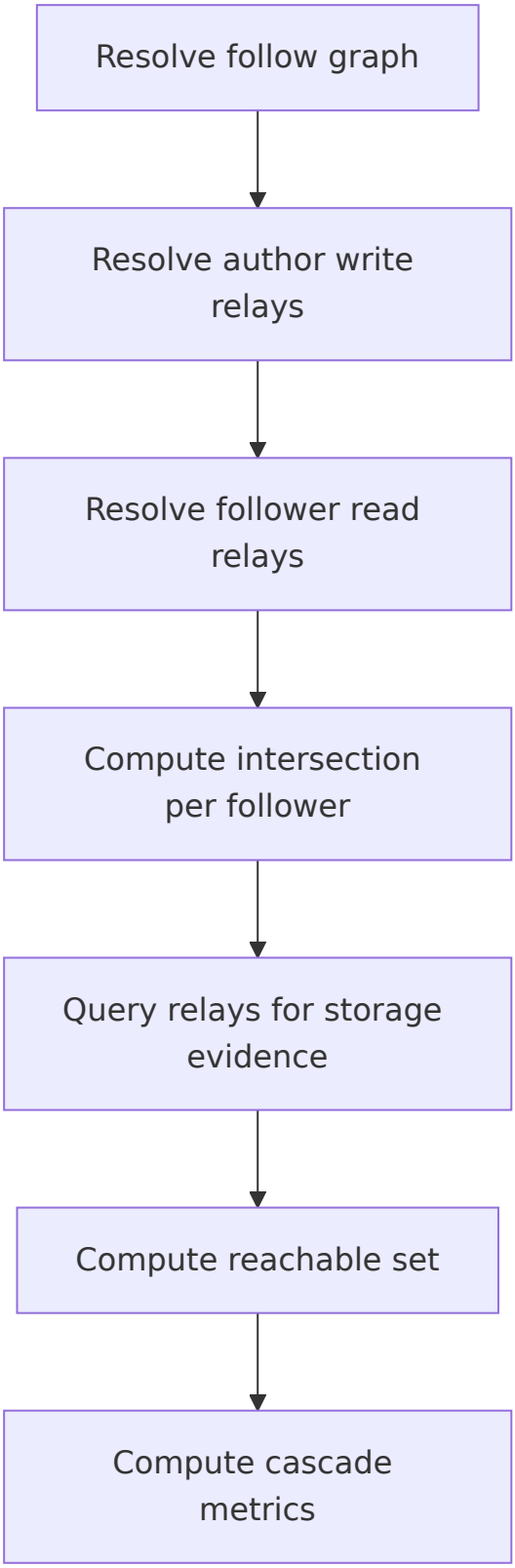
The measurement procedure has six steps. First, resolve the NIP-02 state at the publication cutoff. Second, resolve the write relay list of the author. Third, resolve the read relay lists of all followers. Fourth, compute the set of relays in $W(A) \cap R(u)$ for each follower. Fifth, query each relay in the intersection to confirm the note is stored. Sixth, compute the reachable set as the set of followers with at least one confirmed relay in the intersection.

The potential cascade is the reachable set. The observed cascade requires client-side instrumentation, which most crawlers do not have. Distinguishing them is essential for honest reporting. A potential cascade of ten thousand followers is not the same as an observed cascade of ten thousand readers.

7.3 Computational strategy

For a note published by an author with k write relays and F followers, the direct computation is $O(k * F)$ in the worst case. In practice, followers cluster into a small number of distinct read relay sets, so the computation is $O(k * |\text{distinct read sets}|)$, which is much smaller.

The dominant cost is the relay query to confirm storage. If the author has three write relays and there are one hundred distinct read relay sets, the intersection computation produces at most three hundred relay-set pairs, but only three relays need to be queried once to confirm storage.



7.4 Uncertainty handling

Four sources of uncertainty dominate cascade measurement. Relay sampling bias affects whether storage evidence is complete. Stale outbox declarations affect whether the author's declared write relays are accurate. Client variability affects whether followers' actual read relays match their declared read relays. Retention timing affects whether stored evidence persists.

Handling these requires multiple relay panels, temporal snapshots of outbox declarations, client-side telemetry where available, and repeated relay queries over time. Reporting cascade size without uncertainty intervals overstates precision.

---

8. Observable Cascade Hypotheses

8.1 Testable hypotheses

The cascade model supports five falsifiable hypotheses.

H1. Orphan fraction is non-trivial. For typical authors, a meaningful fraction of followers cannot retrieve the author's notes through any of the author's declared write relays. The magnitude of this fraction is the first empirical question.

H2. Relay overlap predicts cascade. The size of the reachable set is more strongly predicted by the size of $W(A) \cap R(u)$ across followers than by follower count alone, once follower count exceeds a threshold.

H3. Hub outboxes have broader reach. Authors whose write relays are widely queried by popular clients have disproportionately larger cascades per follower than authors whose write relays are niche.

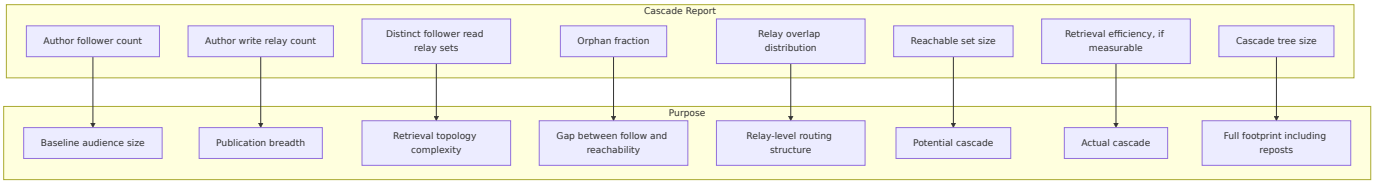
H4. Cascades cluster by client. Followers using the same client tend to have similar read relay sets, so they form coherent cascade subpopulations. This implies that client choice is a structural variable in cascade analysis, not just a user preference.

H5. Stale declarations reduce cascade. Authors whose kind 10002 events are more than six months old have measurably smaller cascades per follower than authors with recent declarations.

8.2 Reporting standard

A cascade analysis should report the following for each studied note. The author's follower count, the author's write relay count, the number of distinct follower read relay sets, the

orphan fraction, the relay overlap distribution, the reachable set size, the retrieval efficiency where measurable, and the cascade tree size including reposts.



---

9. Interaction With the Small-World and Scale-Free Layers

9.1 Cascade position in the multilayer model

Cascades sit at the intersection of three graphs. The follow graph determines who is nominally an audience. The relay routing graph determines who is actually reachable. The client layer determines who is reachable in practice. Cascade analysis that ignores any of these three will produce incomplete results.

This multilayer view is not optional. It is the structure of the problem. A cascade that appears large in the follow graph may be small in the relay routing graph, and vice versa.

9.2 Relation to small-world structure

A small-world user graph accelerates cascades by shortening the path between the author and distant users. But the outbox model can nullify this advantage if the short path traverses relays that no reader queries. Small-world structure is necessary but not sufficient for fast cascades on Nostr.

The practical implication is that cascade speed on Nostr cannot be predicted from follow-graph topology alone. It must be measured, not inferred.

9.3 Relation to scale-free structure

A scale-free degree distribution concentrates cascade potential in hubs. Hubs with broad write relay lists and widely queried relays generate disproportionate cascades. The concentration of cascade potential mirrors the concentration of degree.

But again, the relay layer intervenes. A hub with a narrow outbox declaration may have a smaller cascade than a mid-tier user with a widely queried outbox. Degree concentration is a necessary but not sufficient condition for cascade concentration.

9.4 Interaction with centrality chokepoints

A note whose retrieval depends on a single relay is vulnerable to that relay's failure. Cascade analysis should therefore report the chokepoint index: the fraction of the reachable set whose reachability depends on any single relay. High chokepoint indices indicate cascades that would collapse if a single relay went offline.

This connects cascade analysis to the centrality and resilience layer developed in the companion paper on single points of failure.

---

10. Longitudinal Cascade Behavior

10.1 Time-series metrics

Cascade behavior changes as the network evolves. Four metrics should be tracked over time. The median orphan fraction across all authors. The median relay overlap. The median retrieval efficiency where measurable. The median cascade tree size per follower.

Tracking these over successive snapshots reveals whether the outbox model is converging on common client practices or fragmenting into client-specific patterns.

10.2 Regime change signals

Four events would signal a regime change in cascade dynamics. A sudden increase in relay overlap suggests that clients are converging on common outbox resolution practices. A sudden decrease suggests fragmentation. A sudden increase in orphan fraction suggests that authors are publishing to increasingly niche relays. A sudden decrease suggests consolidation.

These signals are observable from the same data used to compute cascade metrics. No additional instrumentation is required.

10.3 Long-term projections

The long-run behavior of cascades on Nostr depends on whether clients converge on outbox resolution practices. If they do, relay overlap increases, orphan fraction decreases, and cascades become more predictable. If they do not, cascade reach remains client-dependent and highly variable.

The empirical evidence to distinguish these futures does not yet exist. Cascade measurement is a precondition for making that prediction. This is why the framework in this paper matters: without measurement, the question cannot be answered.

---

11. Conclusion

11.1 Key Findings

Four findings emerge from the cascade model developed here.

First, cascades on Nostr are relay-gated. Unlike centralized platforms where cascade size is determined by user-layer dynamics, on Nostr the relay routing graph is a binding constraint that must be measured separately from the follow graph.

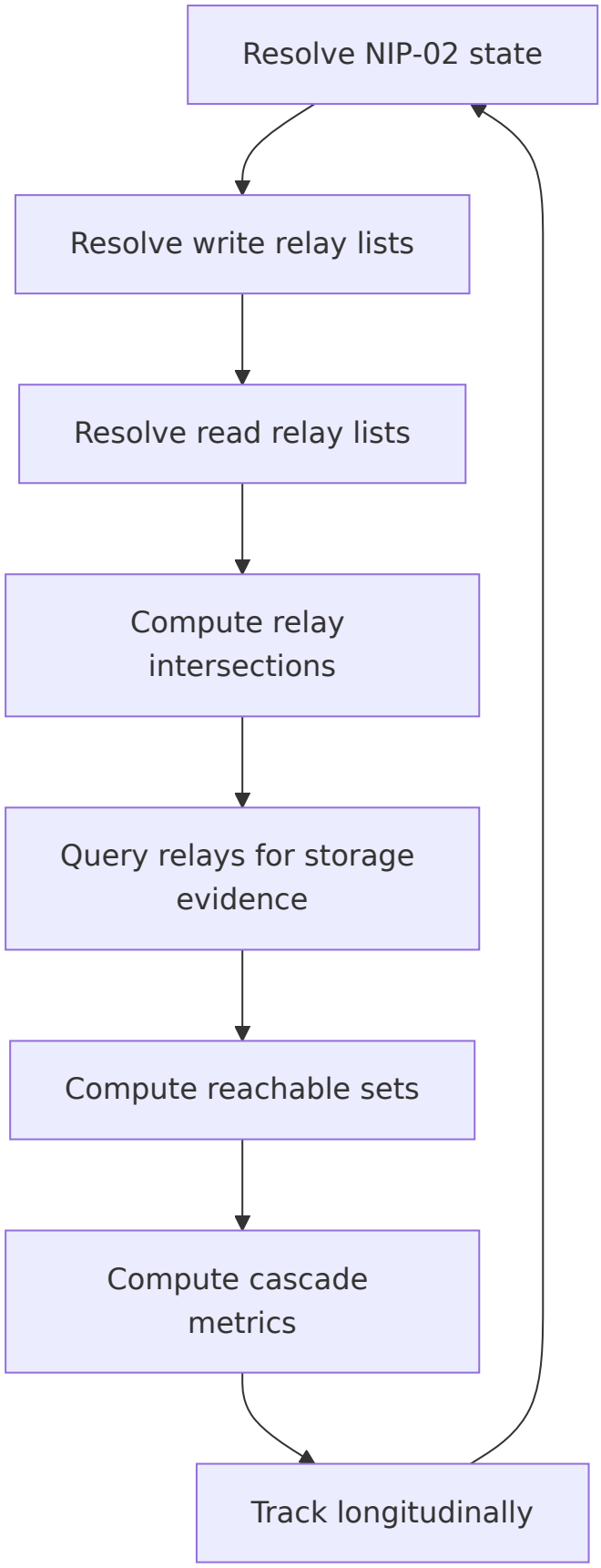
Second, the outbox model creates a structured topology of publication and retrieval that is both more efficient than broadcasting and more fragile than the follow graph suggests. A note can be stored and still be unreachable.

Third, cascade metrics must be defined over three layers — relay, retrieval, and engagement — with separate measurements for each. Collapsing them into a single cascade size loses the information needed to identify failure modes.

Fourth, the interaction between cascade dynamics and centrality, small-world, and scale-free properties is nontrivial. Short paths in the follow graph do not guarantee short paths in the retrieval graph, and hub concentration in degree does not translate directly into cascade concentration unless the hub's outbox is widely queried.

11.2 The Framework

The framework is layered. Resolve the follow graph. Resolve the write relay lists of authors and the read relay lists of followers. Compute intersections. Query relays for storage evidence. Compute reachable sets. Compute cascade metrics. Track over time.



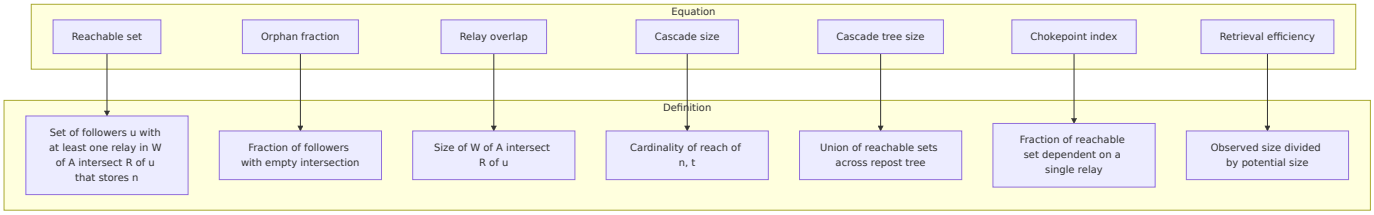
11.3 Closing Insight

Information cascades on Nostr are not simply a function of who follows whom. They are a function of who follows whom, which relays the author publishes to, which relays the

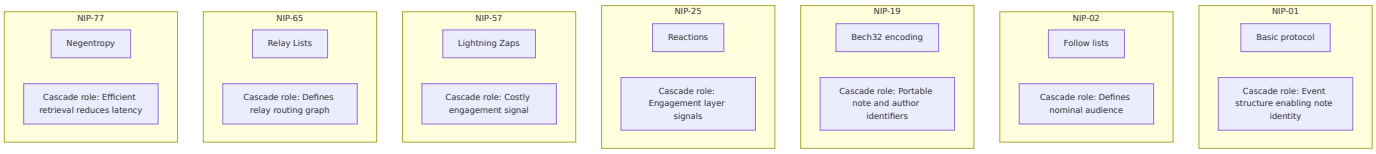
readers query, which clients those readers use, and when each of those facts was last true. Cascade analysis on Nostr must therefore be relay-aware, client-aware, and time-aware. Anything less will understate the role of infrastructure in shaping what appears to be purely social diffusion.

The deeper insight is that the outbox model does not eliminate platform power. It distributes it. The relay routing graph is a new kind of intermediary layer, and its structure determines cascade behavior in ways that are invisible from the follow graph alone. Measuring cascades on Nostr means measuring the relay routing graph.

Appendix: Core Equations



Appendix: NIP Reference Table



Document version: 1.0 | Rendered with Mermaid.js for responsive display

A Whitepaper on Phase Transitions, Hub Removal & Structural Collapse in Nostr Relay Topologies

Document version: 1.0

Abstract

Percolation theory describes how a network transitions from a connected state to a fragmented one as nodes or edges are removed. Applied to Nostr, percolation analysis asks a precise question: at what point does removing relays — through censorship, shutdown, or failure — cause the user graph to break into disconnected islands? This paper develops a percolation framework for Nostr's dual-layer topology of user-follow graphs and user-relay bipartite graphs. It defines percolation thresholds for random and targeted removal, distinguishes bond percolation (edge removal) from site percolation (node removal), and shows how the outbox model changes the effective percolation threshold relative to a broadcast model. The result is a measurement protocol for evaluating Nostr's structural resilience against relay-level attacks and failures.

The central finding of the framework is that percolation thresholds on Nostr are not determined by the follow graph alone. They are determined by the intersection of the follow graph and the relay-visibility graph. A network that appears robust in one layer can be fragile in the other.

---

1. Introduction

1.1 Why percolation theory applies to Nostr

Percolation theory originated in statistical physics to describe how fluids spread through porous media. It has since become a general framework for understanding connectivity in networks: as nodes or edges are removed, at what point does the largest connected component break apart?

For Nostr, this is not an abstract question. Relays go offline. Operators shut down services. Some jurisdictions block specific relay domains. Adversaries launch denial-of-service attacks. Each of these events removes a node from the relay layer. If enough relays are removed, the user graph fragments — not because users stopped following each other, but because they can no longer reach each other's content through any shared infrastructure.

Percolation theory gives us the tools to answer the question quantitatively. What fraction of relays must be removed before the user graph fragments? Which relays, if removed, cause the largest fragmentation? Does the outbox model make the network more or less resilient to relay removal?

1.2 Scope and contribution

This paper applies percolation theory to Nostr's dual-layer topology. It defines percolation thresholds for both random and targeted removal, distinguishes between bond and site percolation, and shows how the outbox model (NIP-65) changes the effective threshold relative to a broadcast model.

The paper does not present new empirical measurements. It defines the measurement framework that a percolation study of Nostr should use, and it identifies the specific hypotheses such a study should test.

The contribution is a four-part framework. First, a formal percolation model for the dual-layer topology. Second, percolation thresholds computed separately for the relay layer and the user layer. Third, targeted removal strategies that identify which relays are most structurally important. Fourth, a set of hypotheses about how the outbox model changes resilience relative to broadcast.

---

2. Formal Percolation Model for Nostr

2.1 Two percolation problems

Nostr has two distinct percolation problems that must be analyzed separately.

The relay-layer percolation problem treats relays as nodes and asks: as relays are removed, at what point does the relay network fragment such that most users can no longer reach most content? This is site percolation on the relay-user bipartite graph.

The user-layer percolation problem treats user keys as nodes and asks: as relays are removed, at what point does the follow graph fragment such that most users can no longer reach most other users through shared infrastructure? This is percolation on the projected user graph, where an edge exists between two users if and only if they share at least one accessible relay.

Both problems are real. Both have distinct thresholds. Confusing them is a common analytical error.

2.2 Bond percolation versus site percolation

Bond percolation removes edges. In Nostr terms, bond percolation describes a scenario where relays stop replicating events, so the edges in the relay graph disappear but the relays themselves continue to operate.

Site percolation removes nodes. In Nostr terms, site percolation describes a scenario where relays shut down entirely or are blocked, so the entire relay disappears from the graph.

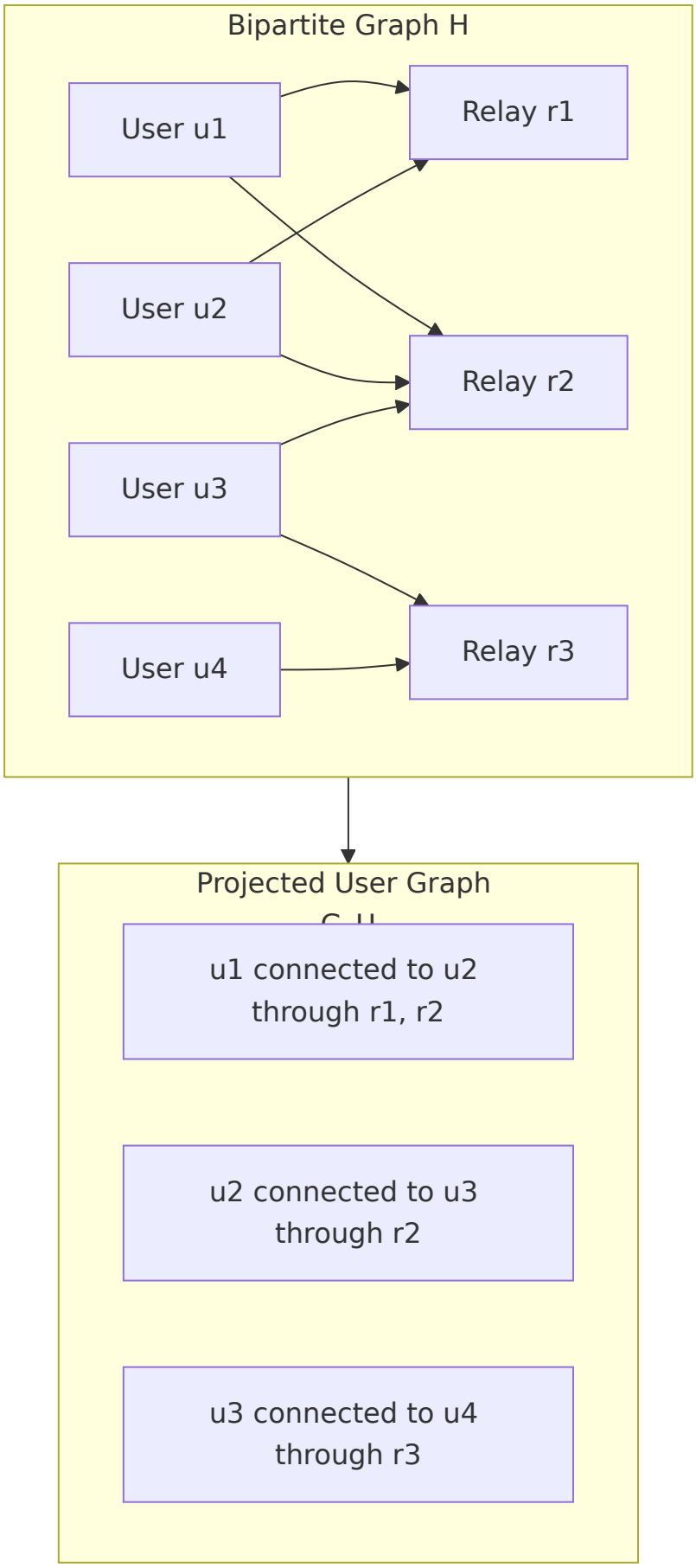
Site percolation is the more realistic attack model for Nostr. Censorship, shutdown, and DoS attacks remove relays, not just their edges. But bond percolation is useful as a lower bound: it describes the best case for resilience, where relays fail only partially.

2.3 The bipartite graph formulation

The natural formulation is the bipartite graph $H = (U, R, F)$, where U is the set of user keys, R is the set of relays, and F is the set of edges representing user-relay associations. An edge (u, r) exists if user u either publishes to relay r or reads from relay r according to their kind 10002 declaration.

Percolation on H asks: as relays are removed from R, does the connected structure of H fragment such that many user pairs no longer share any common accessible relay?

The projected user graph G_U has an edge between users u and v if there exists at least one relay r such that both (u, r) and (v, r) are in F. This projection is what cascade and small-world analyses operate on.



The bipartite view is more faithful to Nostr's actual topology. The projected view is more tractable for path and component analysis. Both are needed.

3. Percolation Thresholds

3.1 The classical random threshold

For a random graph with average degree c, percolation theory predicts that the giant component emerges when c > 1. Below this threshold, the graph consists of small disconnected components. Above it, a large connected component dominates.

Nostr's bipartite graph has an analogous threshold. If each relay is connected on average to k users, and each user is connected on average to m relays, the giant component in the projected user graph emerges when m * (k - 1) > 1 under the approximation that relay connectivity is independent.

This is a first-order approximation. Real Nostr graphs are not random: they have heavy-tailed degree distributions, community structure, and correlations between user relay choices. The actual threshold will differ from the random prediction, but the random baseline is still useful as a reference point.

3.2 Random removal threshold

Under random removal, where each relay is removed with independent probability q, the percolation threshold is the value of q at which the giant component disappears.

For the bipartite graph, random relay removal reduces the average degree of users in the projected graph. If q is the removal fraction, the effective average user connectivity becomes m * (1 - q) * (k - 1). The giant component survives as long as this quantity exceeds one.

This gives a first-order threshold: $q_c = 1 - 1 / (m * (k - 1))$. For a graph with m = 3 and k = 100, the threshold is $q_c \approx 0.997$, meaning almost all relays must be removed to fragment the network. For m = 3 and k = 10, the threshold is $q_c \approx 0.95$.

The intuition is straightforward: when users connect to many relays and relays connect to many users, the network is highly redundant and resilient to random removal.

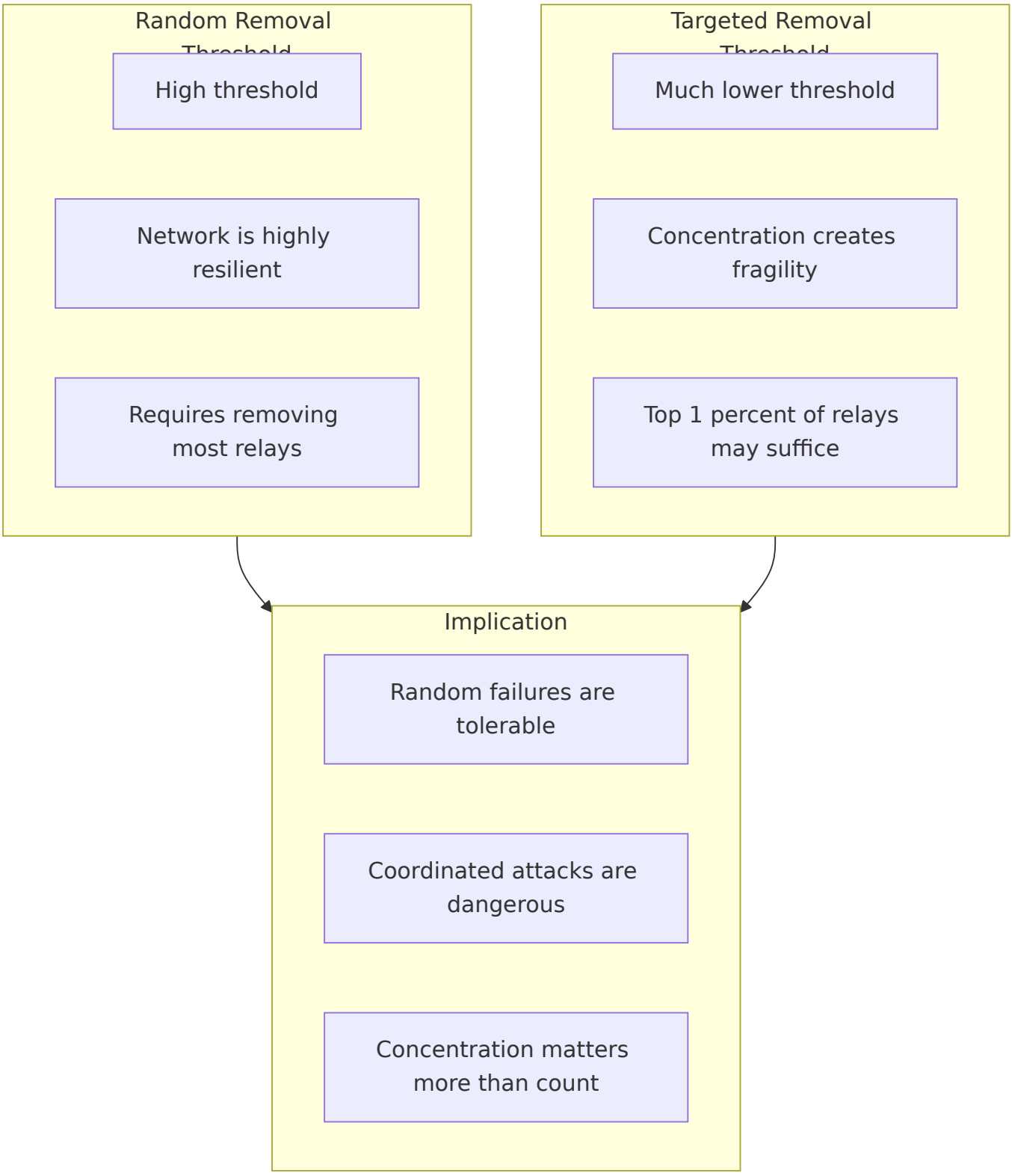
3.3 Targeted removal threshold

Targeted removal is very different. If an adversary knows the structure of the graph and removes the most structurally important relays first, the threshold can be far lower.

For scale-free networks, targeted removal of the highest-degree nodes can destroy the giant component while removing only a small fraction of nodes. The exact threshold depends on the degree distribution exponent and the correlation structure.

For Nostr, this means that if a small number of relays serve a disproportionate fraction of users, and those relays are removed first, the network can fragment while most relays remain operational. The threshold is not a fixed number. It is a function of the concentration of relay coverage.

3.4 Comparing thresholds



The gap between random and targeted thresholds is the key structural metric. A network where these two thresholds are similar is well-balanced. A network where the targeted threshold is much lower than the random threshold is concentrated and therefore fragile to coordinated attack.

---

4. Structural Metrics for Percolation Analysis

4.1 The giant component share

The primary observable is the giant component share g(q), defined as the fraction of users in the largest connected component of the projected graph after removing a fraction q of relays. Percolation theory predicts that g(q) is approximately constant until q approaches the threshold, then drops rapidly to zero.

The shape of g(q) reveals the network's resilience profile. A steep drop near q = 1 indicates high resilience. A gradual decline across a broad range of q indicates fragility distributed across many relays. A sharp drop at low q indicates catastrophic concentration.

4.2 The critical fraction

The critical fraction q_c is the value of q at which g(q) drops below a threshold of interest — typically half of its initial value, or below some absolute floor like 10 percent. This is the operationally meaningful number: how much of the relay layer can be removed before the network fragments.

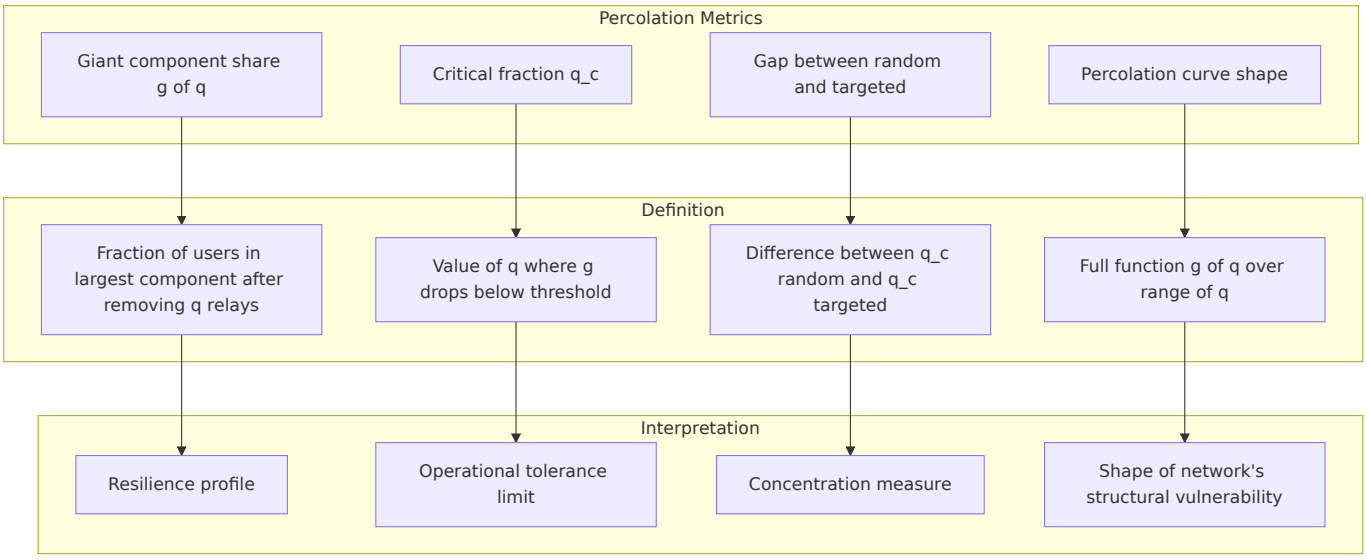
4.3 The gap metric

The gap between the random and targeted thresholds is defined as $\Delta = q_{c,random} - q_{c,targeted}$. A large gap indicates high concentration. A small gap indicates low concentration.

This metric is more informative than either threshold alone. It summarizes the structural difference between uniform and adversarial removal scenarios.

4.4 The percolation curve

The percolation curve is the full function g(q) under either random or targeted removal. Comparing the curves reveals which relays are structurally critical.



---

5. Targeted Removal Strategies

5.1 Removal criteria

Several criteria define different targeted attacks. Each produces a different percolation curve.

Removal by relay degree ranks relays by the number of users connected to them. This is the simplest and often most damaging attack: remove the most popular relays first.

Removal by relay betweenness ranks relays by how many shortest paths in the bipartite graph pass through them. This captures relays that are structurally important even if they serve a moderate number of users.

Removal by relay coverage uniqueness ranks relays by the fraction of events they store that no other relay stores. This captures relays whose removal uniquely destroys event availability.

Removal by geographic or AS concentration ranks relays by their clustering in physical infrastructure. This captures vulnerability to jurisdictional or infrastructural attacks.

5.2 Removal procedure

The procedure for each removal scenario is the same. Rank relays by the chosen criterion. Remove them in order. At each step, recompute the giant component share of the projected user graph. Plot the resulting curve.

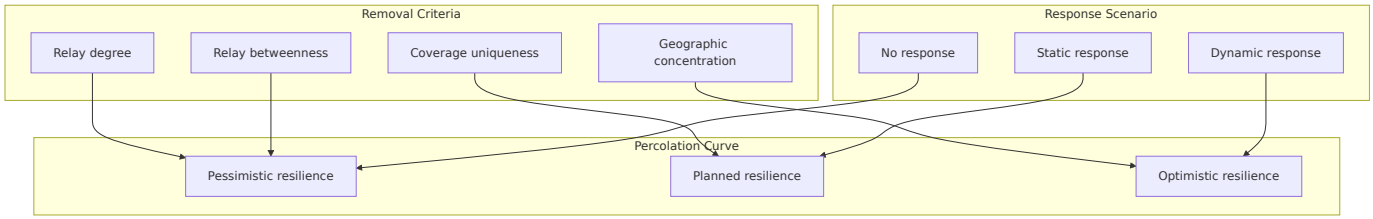
The removal procedure is iterative because percolation effects can be non-linear. Removing one relay changes the betweenness of others. Recomputing the rankings after each removal

yields the true adversarial sequence.

5.3 Removal responses

Three response scenarios are relevant. First, no response: users do not adapt their relay choices as relays are removed. Second, static response: users switch to a pre-defined alternative relay list. Third, dynamic response: users actively reconfigure their relay choices in response to observed failures.

Each response scenario produces a different percolation curve. The no-response curve is the pessimistic case. The dynamic-response curve is the optimistic case. Real networks typically fall between these extremes.



---

6. The Outbox Model's Effect on Percolation

6.1 Broadcast versus outbox

Under a broadcast model, every user publishes to and reads from every relay they are connected to. The bipartite graph is dense, and the effective percolation threshold is high. This was the pattern in early Nostr clients.

Under the outbox model (NIP-65), users declare specific write and read relays. The bipartite graph becomes sparser. Each user has fewer edges to relays, so the effective percolation threshold is lower. The outbox model trades redundancy for efficiency.

6.2 Quantifying the trade

The efficiency gain is measurable: fewer edges means less storage, less bandwidth, and less duplication. The resilience cost is also measurable: fewer edges means fewer redundant paths between users, so removing relays more quickly fragments the graph.

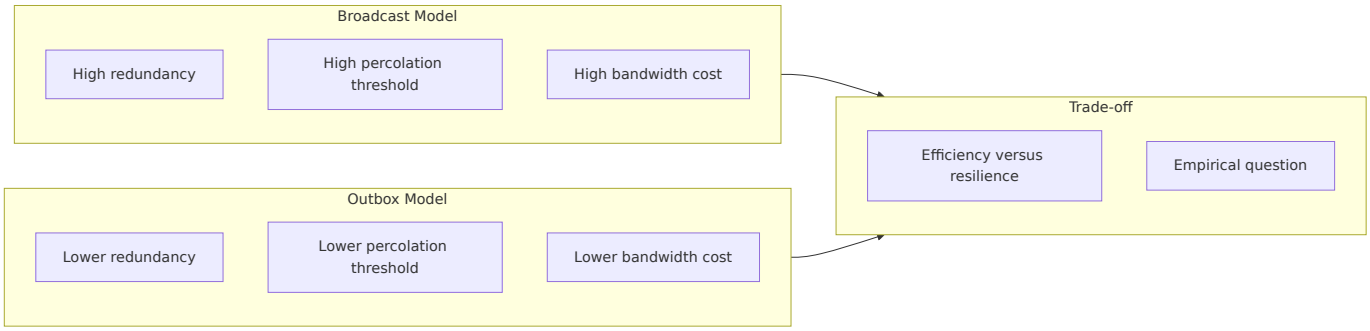
The question is whether the efficiency gain justifies the resilience cost. This is an empirical question that percolation analysis can answer.

6.3 The threshold shift

Under broadcast, the percolation threshold for random removal is high. Under outbox, it is lower. The difference is the resilience cost of the outbox model.

A preliminary estimate: if outbox users average three read relays and three write relays, while broadcast users average ten, the effective degree of the bipartite graph is approximately one-third. The threshold $q_c = 1 - 1 / (m * (k - 1))$ shifts accordingly. For $k = 100$, the broadcast threshold is $q_c \approx 0.999$, and the outbox threshold is $q_c \approx 0.997$. The absolute difference is small. But for smaller relay degree $k = 10$, the broadcast threshold is $q_c \approx 0.99$ and the outbox threshold is $q_c \approx 0.96$. The difference is larger.

The trade is more significant under targeted removal, where the outbox model concentrates traffic on fewer relays per user, making individual relay removal more damaging.



---

7. Measurement Protocol

7.1 Required data

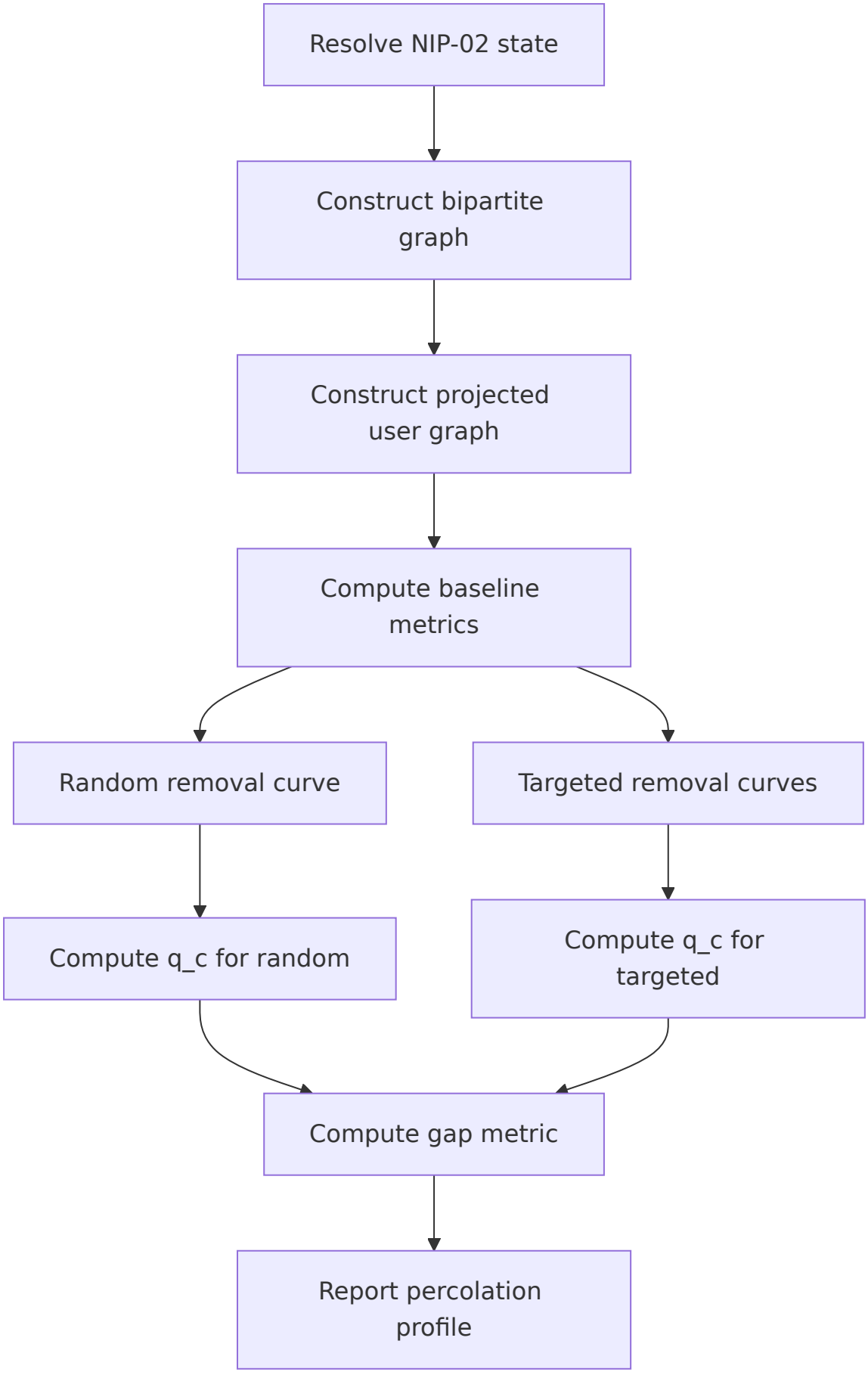
Percolation analysis requires three data sources. First, the resolved NIP-02 follow graph. Second, the user-relay association matrix derived from kind 10002 events and observed event retrievals. Third, relay uptime and availability history to estimate which relays are likely to remain operational under stress.

Optional additions include geographic and AS metadata for infrastructure-aware analysis, and client identification to understand how different client populations are exposed to relay removal.

7.2 Procedure

The procedure has seven steps. First, construct the bipartite user-relay graph from the resolved data. Second, construct the projected user graph using shared relay associations. Third, compute baseline percolation metrics: giant component share, average component size, number of components. Fourth, perform random removal at increasing values of q and record g(q). Fifth, perform targeted removal by each criterion and record g(q). Sixth,

compute the critical fraction q_c for each scenario. Seventh, repeat for each response scenario (no response, static, dynamic).



7.3 Computational strategy

The projected user graph can be large — millions of nodes and tens of millions of edges. Direct simulation of removal for many values of q is expensive. Several optimizations apply.

Incremental component tracking: when a relay is removed, only its neighborhood in the bipartite graph changes. Incremental algorithms can update the component structure efficiently.

Sampling for exploratory analysis: for a first pass, sample a large fraction of users and relays, compute percolation curves on the sample, and extrapolate.

Landmark-based approximation: for giant component estimation on very large graphs, approximation algorithms exist that estimate component sizes without full recomputation.

7.4 Uncertainty handling

Three sources of uncertainty dominate percolation analysis. Relay sampling bias affects whether the observed relay set is complete. User-relay association uncertainty — because kind 10002 declarations may not match actual usage — affects whether observed associations are accurate. Response assumption affects whether the no-response, static, or dynamic scenario is appropriate.

Reporting percolation thresholds without uncertainty intervals overstates precision. The thresholds are estimates, not exact values.

---

8. Observable Hypotheses

8.1 Testable hypotheses

The framework supports six falsifiable hypotheses.

H1. Random removal threshold is high. Under random relay removal, the giant component of the projected user graph survives until at least 80 percent of relays are removed. This predicts strong resilience to uncoordinated failures.

H2. Targeted removal threshold is lower. Under targeted removal by relay degree, the giant component fragments after removing a small fraction of relays, possibly less than 10 percent.

H3. The gap metric is large. The difference between random and targeted thresholds is large, indicating significant concentration in relay coverage.

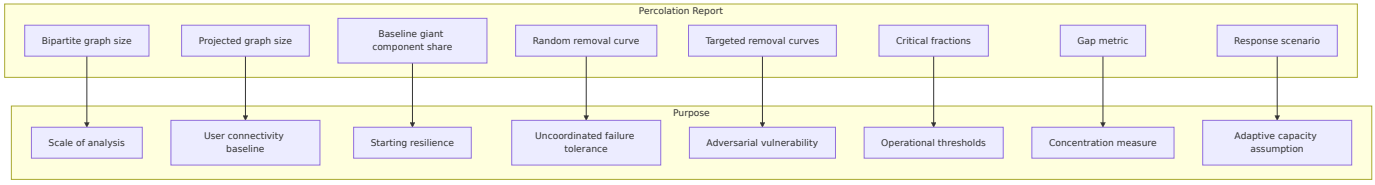
H4. The outbox model lowers the threshold. Comparing the same user population under broadcast and outbox models, the outbox model has a lower percolation threshold due to sparser bipartite connectivity.

H5. Dynamic response substantially improves resilience. When users actively reconfigure relay choices in response to removal, the effective percolation threshold is much higher than under static or no-response scenarios.

H6. Geographic concentration increases vulnerability. Removing relays clustered in a single geographic region produces percolation curves that diverge from those of random removal, indicating that jurisdictional attacks are more damaging than random failures.

8.2 Reporting standard

A percolation analysis should report the following for each scenario. The size of the bipartite graph. The projected user graph. The giant component share at baseline. The percolation curve under random removal with confidence intervals. The percolation curve under targeted removal by each criterion. The critical fraction for each scenario. The gap metric. The response scenario assumption.



---

9. Interaction With Prior Frameworks

9.1 Relation to small-world analysis

Small-world analysis measures path lengths and clustering under normal conditions. Percolation analysis measures how those properties change under stress. The two are complementary: a graph can be small-world at baseline and fragment quickly under targeted removal, or it can be moderately clustered at baseline and remain connected under attack.

The small-world coefficient σ is a baseline structural property. The percolation threshold is a resilience property. Both are needed for a complete description.

9.2 Relation to scale-free analysis

Scale-free analysis measures the degree distribution. Percolation analysis measures the consequences of that distribution. A heavy-tailed distribution typically implies a large gap

between random and targeted thresholds. Measuring the distribution without measuring the percolation consequences is only half the story.

The practical connection: the tail exponent of the degree distribution predicts the percolation behavior. Steeper tails (smaller α) imply more concentration and therefore a larger gap metric.

9.3 Relation to centrality analysis

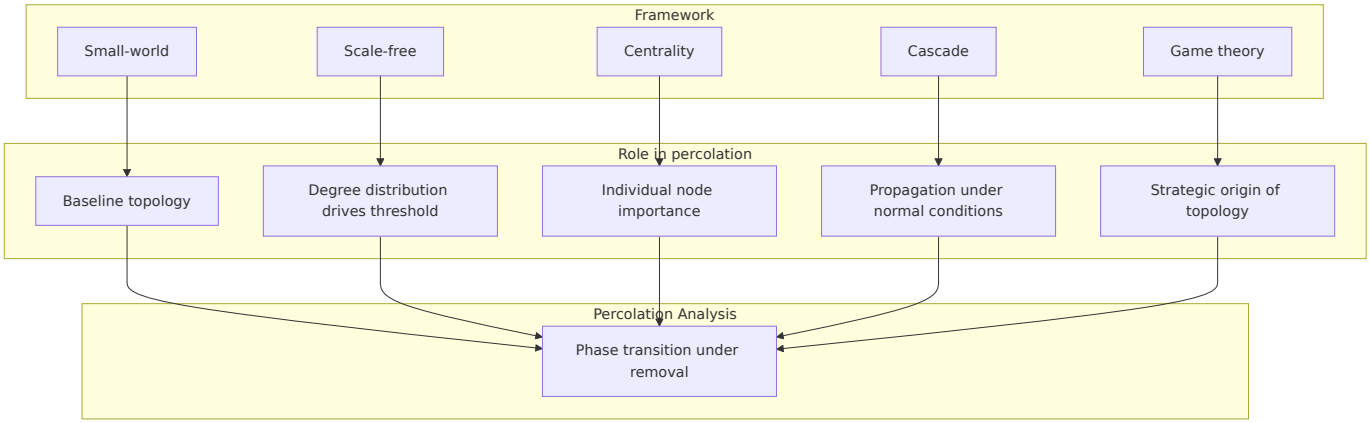
Centrality analysis identifies which individual nodes are structurally important. Percolation analysis measures the aggregate consequence of removing them. A relay with high betweenness centrality is likely to be on the targeted removal path. The removal curve reveals how much of the betweenness is concentrated in a small number of relays.

9.4 Relation to cascade analysis

Cascade analysis measures propagation under normal conditions. Percolation analysis measures propagation under stress. A cascade that passes through a relay that is removed will not complete. Percolation analysis identifies which relays are structural bottlenecks for cascades.

9.5 Relation to the game-theoretic view

The outbox model is a strategic choice: users weigh the efficiency of declaring specific relays against the resilience of declaring many. Percolation analysis quantifies the consequence of these choices at the population level. It provides the missing data for evaluating whether the current equilibrium of outbox behavior is resilient.



---

10. Longitudinal Percolation

10.1 Percolation over time

Percolation thresholds are not fixed. They change as the network evolves. If users consolidate onto fewer relays, the gap metric grows. If users diversify, the gap metric shrinks. If major relays shut down, the threshold drops.

Tracking percolation metrics over time reveals whether Nostr's structural resilience is improving or degrading.

10.2 Regime change signals

Four events would signal a regime change. A sharp decrease in the random threshold suggests that the network is becoming less redundant. A sharp decrease in the targeted threshold suggests that concentration is increasing. An increase in the gap metric suggests growing structural inequality. A decrease in the gap metric suggests growing structural balance.

These signals can be detected from rolling percolation analyses without additional data collection.

10.3 Forward-looking analysis

Because percolation behavior is deterministic given the graph structure, forward-looking analysis is possible. Given a projected graph growth trajectory, one can estimate how the percolation threshold will evolve. This informs policy: if the trend is toward concentration, interventions can encourage diversification before the threshold becomes problematic.

This is a fundamentally different analytical approach from cascade or small-world analysis, which are retrospective. Percolation is more forward-compatible.

---

11. Conclusion

11.1 Key Findings

Four findings emerge from applying percolation theory to Nostr.

First, percolation on Nostr is a two-layer problem. The relay layer determines which relays are available. The user layer determines which users are connected through shared relays. Both layers must be analyzed.

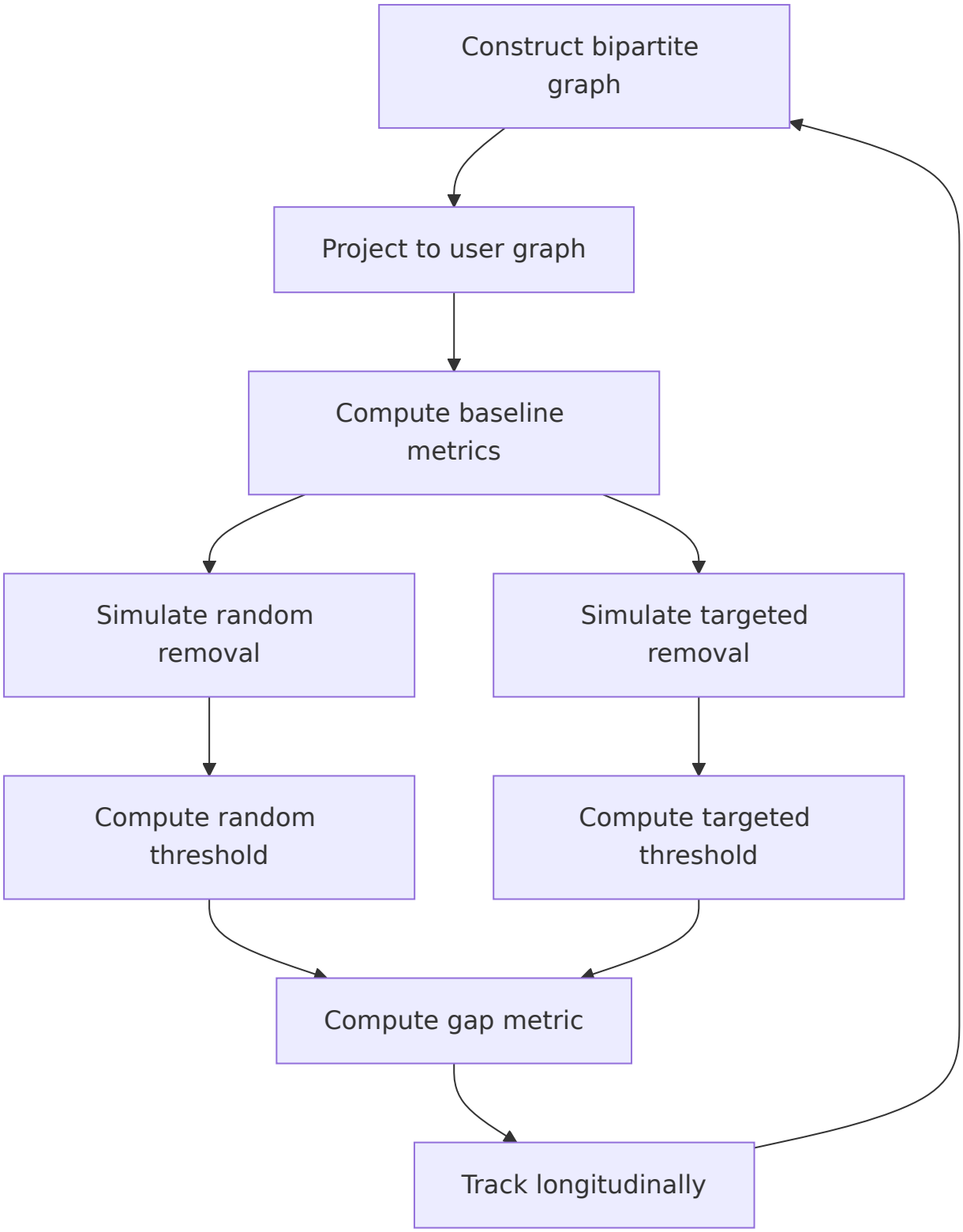
Second, the gap between random and targeted removal thresholds is the key structural metric. A large gap indicates concentration and therefore vulnerability to coordinated attack. A small gap indicates distributed coverage and therefore resilience.

Third, the outbox model trades resilience for efficiency. It reduces redundancy and therefore lowers the percolation threshold, but it also reduces bandwidth and storage costs. Whether the trade is worth it is an empirical question.

Fourth, percolation analysis is forward-compatible. Unlike cascade and small-world analyses, which are retrospective, percolation thresholds can be projected into the future based on observable trends.

11.2 The Framework

The framework is layered. Construct the bipartite user-relay graph. Project to the user graph. Compute baseline metrics. Simulate random removal. Simulate targeted removal. Compute critical fractions. Compute the gap metric. Repeat for response scenarios. Track over time.



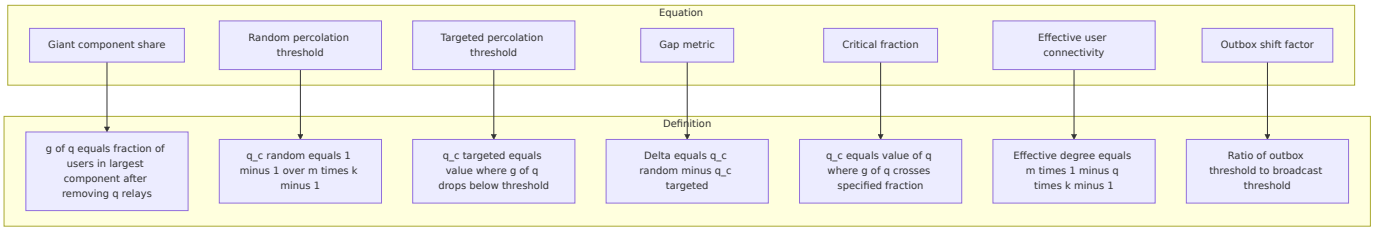
11.3 Closing Insight

Percolation theory converts the vague notion of network robustness into a precise, measurable property. The question is not "Is Nostr resilient?" but "At what fraction of relay removal does the user graph fragment, and how does this fraction differ between random and targeted attacks?"

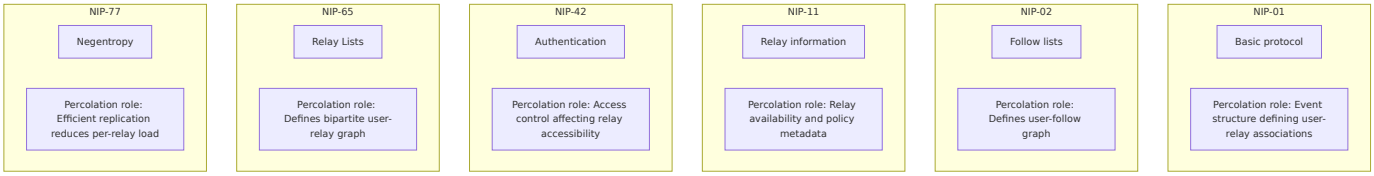
The answer depends on the structure of the relay graph, which is itself the result of user choices under the outbox model. This means that resilience is not just a technical property of the protocol — it is a consequence of the aggregate behavior of users publishing kind 10002 events. Every user who declares a diverse set of relays contributes to the network's percolation threshold. Every user who consolidates on a small set of popular relays reduces it.

The structural insight is that resilience is not deployed by any single actor. It emerges from the distribution of choices. Percolation analysis is the tool that makes this emergence measurable, and therefore governable.

Appendix: Core Equations



Appendix: NIP Reference Table



Document version: 1.0 | Rendered with Mermaid.js for responsive display

Principal-Agent Problem Analysis: Evaluating Agency Costs Between Users and Client Developers or Relay Operators

Whitepaper — Draft for Discussion

Keywords: Nostr, principal-agent problem, agency costs, client developers, relay operators, NIPs, decentralization, privacy, moderation, governance

---

Abstract

Nostr is often described as a protocol in which users own their identity and can move between clients and relays. That description is accurate at the level of keys and events, but it hides a practical reality: users do not interact with the protocol directly. They interact through client software written by client developers, and they depend on relay operators to store and deliver events. These intermediaries are agents. The user is the principal.

This whitepaper evaluates agency costs between users and two classes of agents in the Nostr ecosystem: client developers, whose software mediates identity, signing, feed construction, relay selection, privacy, and payments; and relay operators, who control acceptance, storage, delivery, retention, moderation, authentication, and metadata exposure. The analysis uses the principal-agent framework of monitoring costs, bonding costs, and residual loss, and it intentionally avoids formal equations. Relationships are expressed in prose and in Mermaid diagrams.

The central finding is that Nostr reduces some agency costs dramatically by design — especially lock-in and loss of identity — but it does not eliminate the principal-agent problem. Client developers remain high-authority agents because they control the user experience and often see plaintext, keys, and relay choices. Relay operators remain high-authority agents because they control availability, retention, moderation, and metadata. The most effective mitigations are least-privilege signers, multiple relays, transparent client and relay behavior, verifiable relay information, portable identity, and explicit user consent.

---

1. Executive Summary

Nostr’ s design changes the principal-agent problem from one of platform captivity to one of distributed delegation. A user holds a keypair and signs events. No company owns the account. Relays are replaceable. Clients are replaceable. Events are portable. These

properties reduce the residual loss that users suffer when a centralized platform changes its rules.

However, portability is not the same as zero agency cost.

Client developers exercise agency because they decide how events are composed, which relays are used, how feeds are ranked, what is hidden, what is shown, how keys are requested, how encryption is implemented, and what telemetry is collected. Relay operators exercise agency because they decide what to accept, what to store, what to serve, what to omit, what to log, what to moderate, and what to delete.

The principal-agent problem in Nostr therefore has two faces:

1. Client-side agency: the user delegates interface, workflow, and often security decisions to software.
2. Relay-side agency: the user delegates availability, persistence, and delivery to infrastructure.

The most important mitigations are already present in the protocol and its NIPs: remote signers, browser and Android signers, multiple relays, relay information documents, authentication, protected events, deletion requests, request-to-vanish, gift-wrapped private messages, relay list metadata, completeness hints, and set reconciliation. These mitigations work best when users treat clients and relays as accountable agents rather than as invisible infrastructure.

---

2. Introduction

Nostr stands for “Notes and Other Stuff Transmitted by Relays.” It is a simple protocol for publishing signed events over the internet. A user creates a keypair, writes events, signs them, and sends them to relays. Relays store events and serve them to clients. Clients decide what to display and which relays to use.

This architecture is deliberately simple. Relays are not supposed to be intelligent gatekeepers. Clients are supposed to be smart. The protocol defines event shape, signatures, filters, and message flow. Optional NIPs define social features, private messages, groups, payments, moderation, and application-specific behavior.

The simplicity is a strength, but it creates a governance question. If users own their keys, why do they still depend on client developers and relay operators? The answer is delegation. Users delegate execution. They do not want to construct JSON events by hand, manage

WebSocket connections, implement encryption, choose relays for every action, or verify every relay response. They use software. That software is an agent. They use relays. Those relays are agents.

The principal-agent problem asks how a principal can induce an agent to act in the principal’ s interest when the agent has different preferences, more information, or the ability to take hidden actions. In Nostr, the principal is the user who holds the key and signs events. The agents are client developers and relay operators.

This whitepaper asks:

- What authority do client developers and relay operators actually hold?
- What agency costs arise from that authority?
- How does Nostr’ s protocol design reduce or increase those costs?
- What should principals, client developers, relay operators, and the wider ecosystem do to keep agency costs low?

The analysis is qualitative and protocol-oriented. It does not attempt to measure agency costs statistically. It provides a structured framework for evaluating them.

---

3. Nostr Architecture in Principal-Agent Terms

3.1 The event as the principal’ s signed instruction

In Nostr, the event is the basic object. It contains an identifier, a public key, a timestamp, a kind, tags, content, and a signature. The signature proves that the holder of the private key authorized the event. The event is the principal’ s instruction.

This is important for agency analysis. A client can construct an event, but it cannot sign it without the key. A relay can store an event, but it cannot forge the principal’ s signature. The signature is the boundary between principal authority and agent action.

That boundary is strong, but it is not absolute. The client decides what event to construct. The relay decides whether to accept it. The client decides whether to show the user the event. The relay decides whether to serve it. The signature protects authorship, not the full workflow.

3.2 Relays are simple, clients are smart — and that creates client power

The common Nostr design principle is “smart clients, simple relays.” Relays accept, store, and serve events. Clients compose, render, filter, and interpret. This split is good for

decentralization, but it concentrates agency in client software. The client is where the user sees the feed, chooses relays, approves signatures, encrypts messages, and pays invoices.

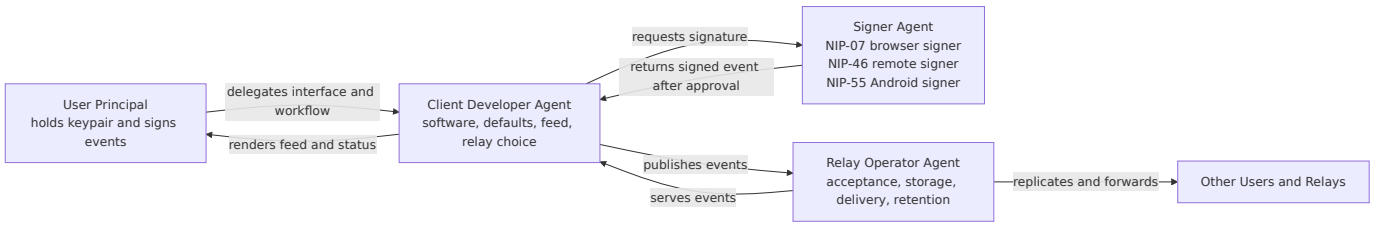
A relay operator is also an agent, but a different kind. The relay operator controls availability, persistence, and policy. The client developer controls interpretation and workflow.

3.3 Portability reduces lock-in but not delegation

Because the user holds the key, identity is portable. Because events are signed and stored on relays, content is portable. Because NIPs are open, clients can be replaced. This reduces the residual loss of being trapped in a platform.

But portability does not remove the need to trust an agent. A user still needs a client to interact with the protocol. A user still needs relays to store and deliver events. The question is not whether delegation exists. The question is whether the delegation is monitored, bounded, and reversible.

Figure 1: Principal-agent map in Nostr



How to read this diagram. The user is the principal. The client developer’ s software is the primary agent for interface and workflow. The signer is a constrained agent that holds or accesses the key and asks for approval. The relay operator is the infrastructure agent that stores and delivers events. The user delegates to the client, but the signer and the relay create checkpoints. The signature is the principal’ s authorization. The relay’ s response is the agent’ s performance.

---

4. A Framework for Agency Costs

Agency costs arise when one party acts on behalf of another. They are usually grouped into three categories.

4.1 Monitoring costs

Monitoring costs are the costs the principal bears to observe and verify the agent’ s behavior. In Nostr, monitoring includes:

- reading a relay’s NIP-11 information document;
- checking whether a client is open source;
- comparing event sets across multiple relays;
- verifying signatures and event identifiers;
- checking whether a relay honors deletion requests;
- reviewing a client’s telemetry and permissions;
- inspecting a signer’s approval prompts;
- checking whether a relay serves complete results or silently omits events;
- reviewing payment requests and wallet permissions.

Monitoring is not free. It requires attention, technical knowledge, and time. Many users will not monitor at all. That is why bonding and protocol design matter.

4.2 Bonding costs

Bonding costs are costs the agent bears to credibly commit to act in the principal’s interest. In Nostr, bonding appears through:

- reputation of client developers and relay operators;
- open-source code and reproducible builds;
- paid relays with visible terms;
- NIP-05 domains tied to real identities;
- badges and trusted assertions;
- relay information documents with stated policies;
- security audits;
- long-running relay identities;
- community moderation and public track records.

Bonding makes it easier for principals to choose agents. It does not guarantee good behavior, but it raises the cost of bad behavior.

4.3 Residual loss

Residual loss is the loss that remains even after monitoring and bonding. In Nostr, residual loss includes:

- privacy leakage through IP addresses, timing, and metadata;
- censorship by a relay that refuses to serve or store an event;
- loss of availability when a relay goes offline;
- failure to delete content after a deletion request or request to vanish;
- key compromise through a malicious client or extension;

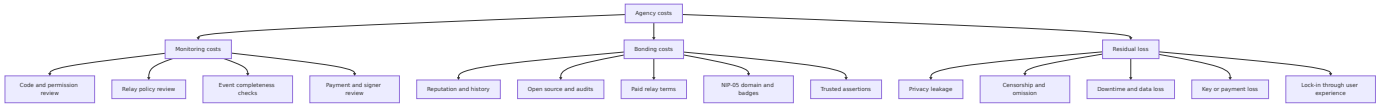
- payment loss through a wallet service or payment flow;
- lock-in through user experience even when the protocol allows exit;
- missed content because a relay omitted results or a client filtered silently.

4.4 Adverse selection and moral hazard

Adverse selection occurs before the relationship begins. A user chooses a client or relay without knowing its true quality. Moral hazard occurs after the relationship begins. The agent changes behavior because the principal cannot easily observe it.

In Nostr, adverse selection appears when a user picks a client because it looks polished but does not disclose telemetry or key handling. It appears when a user picks a relay because it is fast but does not disclose retention or logging. Moral hazard appears when a relay begins omitting events, or when a client begins adding tracking, after the user has settled into the workflow.

Figure 2: Agency cost taxonomy



How to read this diagram. Agency costs are not only money. They include attention, verification effort, reputational risk, and the losses that remain after precautions. Monitoring and bonding reduce residual loss, but they do not remove it. The diagram is a checklist for evaluating any client or relay relationship.

---

5. Agent Classes and Their Authority

Nostr has more than two agent classes. Client developers and relay operators are the main ones, but signers, wallet services, group admins, and application handlers also act for users.

Agent class What the agent controls Information advantage Principal exposure
Client developer Interface, defaults, event construction, relay selection, feed ranking, filtering, encryption workflow, payment prompts, update channel Knows code, algorithm, telemetry, and UX choices Privacy, missed content, key handling, payment mistakes, lock-in
Relay operator Acceptance, storage, delivery, retention, moderation, authentication, rate limits, logging, deletion Knows server logs, policies, retention, and actual behavior Censorship, downtime, metadata leakage, deletion failure, payment loss
Signer or bunker Key access, approval prompts, signing policy, encryption and decryption requests Knows which requests are made and when Key compromise, over-broad

permissions, phishing
Wallet or payment service Invoice payment, balance access, budget enforcement, transaction history Knows payment requests and user behavior Payment loss, privacy leakage, over-spending
Group admin or relay group Membership, roles, moderation, group metadata, message visibility Knows group membership and moderation state Exclusion, censorship, governance capture
Application handler Rendering unknown event kinds, redirecting users, loading external content Knows which events are opened and how Malicious redirects, privacy leakage, unsafe rendering

The rest of this paper focuses on client developers and relay operators because they are the two broadest and most unavoidable agent classes.

---

6. Client Developer Agency Costs

6.1 Why client developers are high-authority agents

A client developer does not need to hold the user’ s key to exercise significant authority. The client decides:

- what the user sees first;
- which relays are queried;
- which events are filtered out;
- how threads are rendered;
- whether mentions are shown as links;
- whether media is loaded automatically;
- whether telemetry is collected;
- whether the client asks for a signature with clear context;
- whether the client supports remote signers;
- whether the client defaults to public or private behavior;
- whether the client shows deletion requests and content warnings;
- whether the client displays relay information and limitations;
- whether the client supports NIP-89 app handlers or falls back to NIP-31 alt text.

The client is where the protocol becomes an experience. That is a position of great agency.

6.2 Hidden action by client developers

Hidden action means the agent takes actions the principal cannot easily observe. Examples in Nostr clients include:

- sending analytics or crash reports without clear consent;
- fingerprinting the user through browser or device details;
- quietly preferring certain relays or hiding others;
- ranking feeds by an opaque algorithm;
- filtering replies, reports, or labels without explanation;
- auto-loading remote media and leaking IP addresses;
- requesting signatures without showing the full event content;
- defaulting posts to public when the user expected a smaller audience;
- storing keys in local storage rather than using a signer;
- bundling trackers in a web client;
- changing behavior through an update channel the user does not inspect.

A client developer may not personally observe each user, but the software can act at scale. That is the agency problem.

6.3 Hidden information about clients

Hidden information means the principal does not know the agent’ s true quality or behavior before choosing it. Examples include:

- whether the client is open source;
- whether the build matches the source;
- whether the client uses a secure signer;
- whether the client supports NIP-44 encryption correctly;
- whether the client respects NIP-65 relay lists;
- whether the client verifies relay information;
- whether the client supports NIP-70 protected events;
- whether the client honors deletion and expiration;
- whether the client exposes telemetry settings;
- whether the client is maintained and patched.

A polished interface can hide a weak security posture. That is adverse selection.

6.4 Residual loss from client agency

Residual loss includes:

- privacy loss from metadata leakage;
- key compromise from unsafe storage;

- wrong audience because of confusing defaults;
- missed content because of hidden filtering;
- payment mistakes because of unclear prompts;
- user lock-in because the client’ s workflow is hard to leave even though the protocol allows it;
- loss of trust when the client changes behavior after adoption.

The protocol reduces lock-in because keys and events are portable. But users can still be locked in by habit, interface, and network effects.

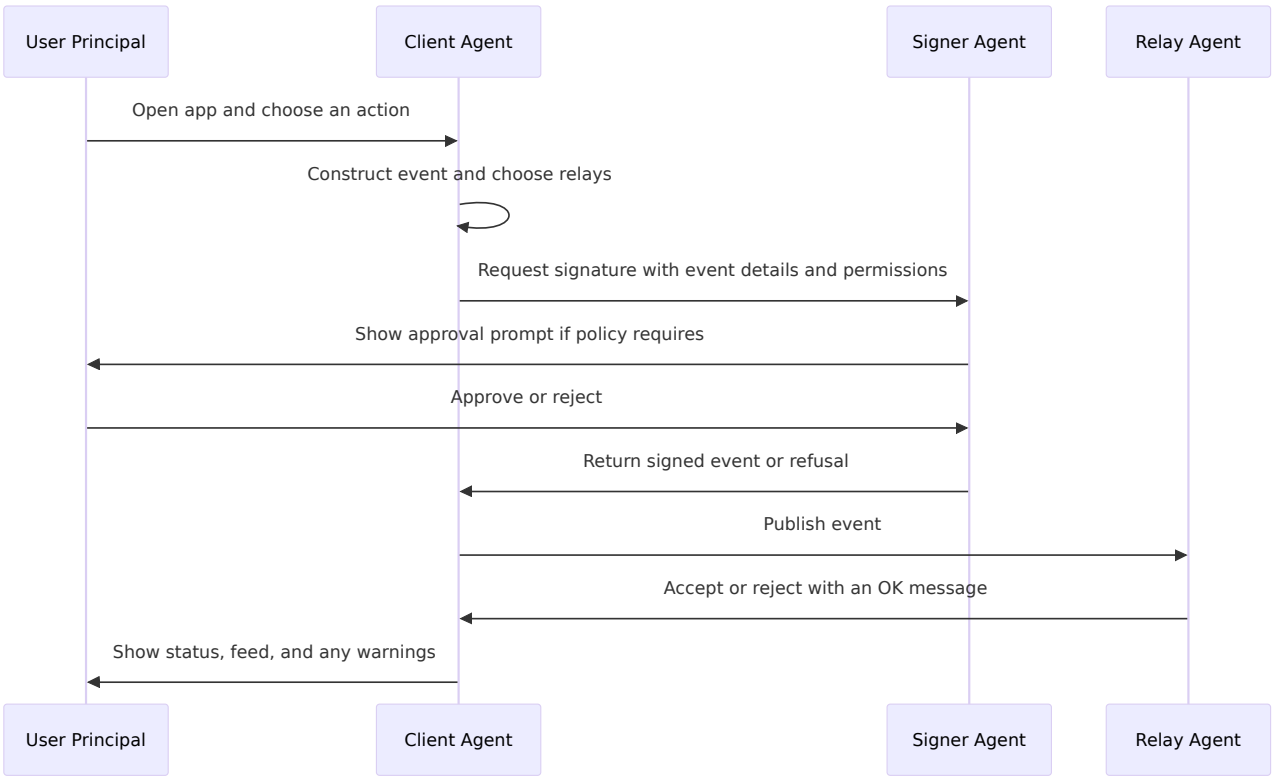
6.5 Mitigations for client agency

The strongest mitigation is least-privilege signing.

- NIP-07 provides a browser signer object. A web client can request a public key and signatures without seeing the private key. The extension can show approval prompts.
- NIP-46 provides remote signing. A bunker holds the key. The client uses a disposable client keypair and asks the signer to sign specific events or perform encryption. The client can request permissions, and the signer can require authentication.
- NIP-55 provides an Android signer application. The client sends intents or content resolver queries. The user approves requests, and the signer can remember permissions.
- NIP-49 encrypts a private key with a password into an ncryptsec format. This helps backup, but the client still handles the password and decrypted key.
- NIP-70 protected events let an author mark an event as publishable only by its author. This helps prevent copying, but relays must enforce it.

Other mitigations include open-source code, reproducible builds, clear permission prompts, privacy-preserving defaults, explicit relay selection, no telemetry by default, support for multiple relays, visible filtering controls, and client tags that users can opt out of.

Figure 3: Client mediation and signer checkpoints



How to read this diagram. The client is not powerless, but the signer creates a checkpoint. The user can see what is being signed. The relay creates a second checkpoint when it accepts or rejects the event. The principal’ s safety depends on the client being honest, the signer being strict, and the relay being transparent. If the client can sign without a prompt, the checkpoint disappears. If the signer grants broad permissions, the checkpoint weakens. If the relay silently accepts and then omits the event from queries, the principal may not know.

---

7. Relay Operator Agency Costs

7.1 Why relay operators are high-authority agents

Relay operators control the infrastructure layer. They decide:

- whether to accept an event;
- whether to store it;
- how long to store it;
- whether to serve it to a given client;
- whether to require authentication;
- whether to rate-limit or ban;
- whether to log IP addresses and request patterns;
- whether to honor deletion requests and requests to vanish;
- whether to enforce group membership and moderation;

- whether to advertise accurate limitations and policies;
- whether to provide complete query results.

A relay can be replaceable in principle and still be important in practice. A relay that hosts a group, a community, or a DM inbox has switching costs.

7.2 Hidden action by relay operators

Hidden action includes:

- logging IP addresses and metadata beyond stated policy;
- retaining events beyond advertised retention;
- silently omitting events from query results;
- serving incomplete results and claiming completeness;
- applying shadow moderation;
- biasing search results or counts;
- refusing to honor deletion requests;
- failing to enforce NIP-70 protected events;
- serving gift-wrapped messages to non-recipients;
- selling or sharing data;
- changing fee or access policies without clear notice;
- running a group with opaque admin rules.

Some of these actions are legitimate policy choices. The agency problem is not that relays have policies. The problem is that users may not know the policies, cannot verify them, or cannot easily exit.

7.3 Hidden information about relays

Hidden information includes:

- the relay’ s actual retention policy;
- the relay’ s logging practices;
- the relay’ s jurisdiction and legal exposure;
- the relay’ s true supported NIPs;
- the relay’ s admin identity and contact;
- the relay’ s moderation rules;
- the relay’ s fee schedule;
- the relay’ s behavior under load;
- the relay’ s completeness and search quality;
- the relay’ s group governance.

NIP-11 helps with disclosure, but disclosure is voluntary. A relay can omit fields. A relay can advertise support for a NIP and still behave differently. That is why monitoring and multiple relays matter.

7.4 Residual loss from relay agency

Residual loss includes:

- censorship or selective availability;
- downtime and data loss;
- privacy loss from logs and metadata;
- deletion failure;
- payment loss;
- group governance capture;
- missed events because of incomplete results;
- search and count manipulation;
- user confusion when a relay silently changes policy.

7.5 Mitigations for relay agency

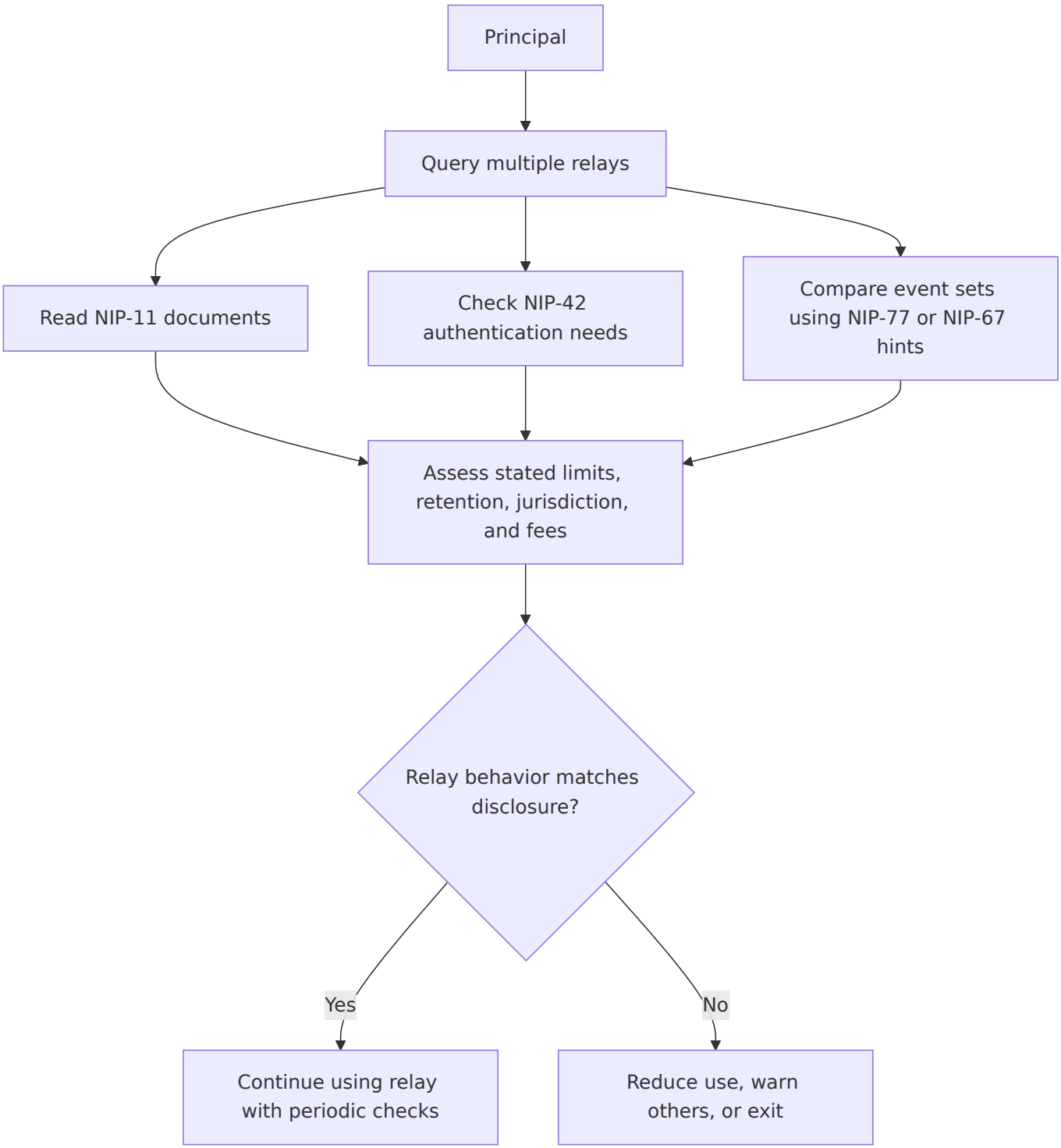
The protocol provides several tools:

- NIP-11 relay information document describes the relay’ s name, contact, supported NIPs, limitations, retention, jurisdiction, language, tags, and fees.
- NIP-42 authentication lets a relay authenticate clients. This can restrict access to DMs or protected resources. It also means the relay learns the client’ s pubkey, so it is a tradeoff.
- NIP-70 protected events let an author mark an event as publishable only by the author. Relays that enforce it must require authentication and check the author.
- NIP-09 deletion requests ask relays to delete events. Relays may ignore them. Clients should validate that the deletion request author matches the event author.
- NIP-62 request to vanish asks relays to delete everything from a key, including deletion events. Some jurisdictions treat it as binding. Relays that support it should truly delete.
- NIP-40 expiration asks relays to drop events after a timestamp. It is not a security feature.
- NIP-65 relay list metadata lets users publish read and write relays. Clients can use the author’ s write relays and tagged users’ read relays. This reduces reliance on a single relay.
- NIP-67 EOSE completeness hints let a relay say whether it has sent all stored events, has more, or needs authentication for more. This helps clients detect incomplete results.
- NIP-77 negentropy syncing lets clients and relays reconcile event sets efficiently. This helps verify that the client has the events the relay has.
- NIP-45 event counts lets relays provide counts, optionally with HyperLogLog values. This

can be useful but also manipulable.

- NIP-86 relay management API gives relay operators an administrative API for bans, allow lists, roles, and moderation. It is a tool for operators, not a direct user monitoring tool.
- NIP-29 relay-based groups publishes group metadata, admin lists, member lists, roles, and moderation events from the relay. This makes group governance more visible, but the relay still enforces it.

Figure 4: Relay verification loop



How to read this diagram. The principal does not need to trust a single relay blindly. The principal can query several relays, read their information documents, check authentication requirements, and compare event sets. If the relay’ s behavior matches its disclosure, the principal can continue with periodic checks. If not, the principal can reduce use or exit. This loop reduces monitoring cost compared with blind trust, but it still requires effort.

---

8. Cross-Cutting Agency Issues

8.1 Privacy and metadata

Both clients and relays can leak metadata.

- The client sees the user’ s IP address, device details, UI actions, and possibly plaintext before encryption.
- The relay sees IP addresses, connection timing, subscription patterns, and event metadata.
- NIP-17 private direct messages use NIP-44 encryption and NIP-59 gift wrapping to hide content and participant relationships.
- NIP-59 gift wrap uses a rumor, a seal, and a gift wrap. The rumor is unsigned, the seal identifies the author, and the gift wrap hides the author behind a one-time key.
- Relays should protect gift-wrapped messages by serving them only to the p-tagged recipient using NIP-42 authentication.
- If a client implements gift wrapping incorrectly, or a relay serves gift wraps to non-recipients, privacy is lost.

Agency cost here is high because privacy failures are hard to detect and often irreversible.

8.2 Moderation and labeling

Moderation is distributed across clients and relays.

- NIP-32 labels let users and services attach labels to events, profiles, relays, and topics.
- NIP-51 lists let users maintain mute lists, pinned notes, bookmarks, relay sets, and other lists.
- NIP-56 reports let users report nudity, malware, profanity, illegal content, spam, and impersonation.
- NIP-36 content warnings let clients hide sensitive content behind a warning.
- NIP-72 moderated communities use moderator approval events.
- NIP-29 relay-based groups use relay-enforced membership and roles.

The agency problem is that the user may not know which labels, reports, or moderator approvals a client honors. A client can silently filter content. A relay can silently moderate. The protocol gives tools, but it does not force transparency.

8.3 Payments

Payment agents include wallet services, clients, and relays.

- NIP-57 defines Lightning zaps. A client requests an invoice and a zap receipt is published.
- NIP-47 defines Nostr Wallet Connect. A client talks to a wallet service through encrypted events. The user approves a connection URI with a secret and relay. The wallet service can enforce budgets and permissions.
- NIP-60 and NIP-61 define Cashu wallets and nutzaps. A wallet event stores encrypted wallet information, and token events store unspent proofs.
- NIP-75 defines zap goals.

Agency risks include over-spending, confusing payment prompts, malicious invoices, privacy leakage, and wallet service downtime. The best mitigation is a constrained signer or wallet service with explicit budgets and visible approvals.

8.4 Identity and NIP-05

NIP-05 maps a Nostr key to a DNS-based identifier such as name@domain. This is an identity convenience, not a verification of personhood. The domain operator becomes an agent of sorts because it publishes the mapping.

Clients must follow public keys, not NIP-05 addresses. If a domain changes the key behind an address, the client must not silently replace the followed key. This is a crucial agency-control rule.

8.5 Groups and relay governance

NIP-29 relay-based groups are a clear principal-agent case. The relay enforces membership, roles, and moderation. Group metadata, admin lists, member lists, and roles are published by the relay. Groups can migrate or fork to another relay.

The principal’ s exposure is high because the relay can remove users, delete events, change metadata, or enforce policy. The mitigation is transparency, multiple relays, group migration, and community governance. But migration is costly. A group is not as portable as a single note.

8.6 Search, counts, and completeness

Search and counts are easy places for hidden bias.

- NIP-50 adds a search filter. Relays interpret queries and rank results. The ranking algorithm may be opaque.
- NIP-45 adds event counts. Relays may return probabilistic counts or HyperLogLog values.
- NIP-67 adds EOSE completeness hints.
- NIP-77 adds negentropy set reconciliation.

A client that shows a search result or a count is trusting the relay's interpretation. Multiple relays and client-side verification reduce the risk.

---

9. Qualitative Evaluation Scorecard

The following scorecard evaluates agency costs qualitatively. "Low," "medium," and "high" are relative judgments, not measurements. They depend on the specific client, relay, user skill, and context.

Dimension	Client developer as agent	Relay operator as agent
Information asymmetry	High. The developer knows the code, defaults, telemetry, and algorithm. The user sees the interface.	High. The operator knows logs, retention, moderation, and actual behavior. The user sees the relay's public output.
Monitoring difficulty	Medium to high. Open source helps, but builds, updates, and runtime behavior are hard to verify.	Medium. NIP-11, multiple relays, NIP-67, and NIP-77 help, but logs and hidden moderation remain hard to observe.
Bonding options	Open source, reputation, audits, community trust, public issue trackers.	Paid terms, NIP-11 disclosure, domain identity, long-running reputation, badges, trusted assertions.
Exit cost	Low in protocol terms, medium in practice because of habit and UX lock-in.	Low for general feeds, medium to high for DMs, groups, and communities.
Residual loss	Privacy leakage, key compromise, missed content, payment mistakes, lock-in.	Censorship, downtime, metadata leakage, deletion failure, group governance capture.
Overall agency cost	High for privacy and security-sensitive users; medium for casual users who choose reputable clients.	High for availability and privacy-sensitive users; medium for users with multiple relays.

Interpretation. Client developers often carry higher agency cost for privacy because the client sees plaintext and controls the signing workflow. Relay operators often carry higher agency cost for availability and censorship because they control storage and delivery.

Neither agent is inherently malicious. The point is that both have authority that users must monitor or bound.

---

10. Recommendations

10.1 For principals

- Use a signer. Prefer NIP-46 remote signing, NIP-07 browser signing, or NIP-55 Android signing over pasting a private key into a website.
- Keep the signer's permissions narrow. Approve specific event kinds and specific actions rather than broad blanket permissions.
- Use multiple relays. Do not depend on one relay for general feeds, DMs, or groups.
- Read the relay's NIP-11 document when it matters. Check supported NIPs, limitations, retention, jurisdiction, and fees.
- Prefer clients that are open source, disclose telemetry, support multiple relays, and make filtering visible.
- Use NIP-65 relay lists and the outbox model. Publish your read and write relays.
- Use NIP-17 private messages rather than legacy NIP-04 DMs.
- Use NIP-62 request to vanish when you need a relay to delete your data, and understand that compliance is not guaranteed everywhere.
- Verify that clients follow public keys, not NIP-05 addresses.
- Treat payment prompts as high-risk. Use a wallet service with budgets and explicit approvals.

10.2 For client developers

- Never custody private keys unless the user explicitly chooses a custody model. Prefer signers.
- Make signing prompts show the event kind, content, tags, and consequences in plain language.
- Default to privacy-preserving behavior: no telemetry, no auto-loading remote media, no silent filtering.
- Disclose relay selection and filtering. Let users inspect and change it.
- Support NIP-11 relay information display and NIP-67 completeness hints.
- Support NIP-70 protected events and NIP-09 deletion requests where relevant.
- Support NIP-89 app handlers and NIP-31 alt text for unknown event kinds.
- Make client tags optional and explain their privacy implications.
- Publish security posture, update policy, and telemetry policy.

10.3 For relay operators

- Publish an accurate NIP-11 document. Include supported NIPs, limitations, retention, jurisdiction, contact, and fees.
- Minimize logging. Disclose what is logged and why.
- Use NIP-42 authentication for private messages and protected resources.
- Enforce NIP-70 protected events if you claim to support them.
- Honor NIP-09 deletion requests and NIP-62 requests to vanish when your policy says you do.
- Provide NIP-67 completeness hints where possible.
- Be transparent about moderation. Use NIP-32 labels and NIP-56 reports where appropriate.
- For NIP-29 groups, publish clear group metadata, admin lists, roles, and moderation events.
- Do not silently change retention or access policy without notice.

10.4 For the ecosystem

- Build independent relay and client test suites.
- Encourage reproducible builds and signed releases.
- Support trusted assertions and reputation systems, but avoid centralizing trust in one provider.
- Standardize transparency reports for relays and clients.
- Fund audits for signers, encryption, and payment flows.
- Teach users the difference between protocol portability and practical exit costs.

---

11. Limitations and Future Work

This whitepaper is a qualitative protocol analysis. It does not measure agency costs empirically. Future work could include:

- user studies on how often users read NIP-11 documents or change relay lists;
- technical measurements of client telemetry and permission prompts;
- relay probes that compare advertised NIP-11 claims with actual behavior;
- audits of NIP-17 and NIP-59 implementations;
- studies of NIP-29 group migration and fork costs;
- comparisons of agency costs across client types and relay types;
- longitudinal studies of relay retention and deletion compliance.

The Nostr ecosystem is evolving. NIPs are optional. A NIP that exists does not guarantee implementation. A client or relay that claims support may still behave differently. The framework in this paper should be updated as the protocol and its implementations change.

---

12. Conclusion

Nostr changes the principal-agent problem from platform captivity to distributed delegation. The user owns the key and signs events. Relays are replaceable. Clients are replaceable. Events are portable. This reduces the residual loss that users suffer when a centralized platform changes its rules or locks them in.

But delegation remains. Client developers decide how the user experiences the protocol. Relay operators decide whether events are stored, served, retained, moderated, and protected. Both have information advantages. Both can take hidden actions. Both create monitoring costs, bonding costs, and residual loss.

The best defense is not to pretend that agents do not exist. The best defense is to bound their authority. Signers bound clients. Multiple relays bound relay operators. NIP-11, NIP-42, NIP-67, NIP-77, NIP-09, NIP-62, NIP-70, NIP-17, and NIP-65 give principals tools for monitoring and exit. Open source, audits, reputation, and transparency give agents reasons to behave well.

Nostr does not eliminate the principal-agent problem. It distributes it. The goal is to make that distribution visible, contestable, and reversible. When users can see what their agents do, choose among them, and leave without losing their identity, agency costs fall. When they cannot, the protocol’ s decentralization becomes a promise rather than a practice.

References

- NIP-01: Basic protocol flow description
- NIP-07: window.nostr capability for web browsers
- NIP-09: Event deletion request
- NIP-11: Relay information document
- NIP-17: Private direct messages
- NIP-29: Relay-based groups
- NIP-32: Labeling
- NIP-40: Expiration timestamp
- NIP-42: Authentication of clients to relays

- NIP-44: Encrypted payloads
- NIP-45: Event counts
- NIP-46: Nostr remote signing
- NIP-47: Nostr Wallet Connect
- NIP-49: Private key encryption
- NIP-50: Search capability
- NIP-51: Lists
- NIP-55: Android signer application
- NIP-56: Reporting
- NIP-59: Gift wrap
- NIP-60: Cashu wallets
- NIP-61: Nutzaps
- NIP-62: Request to vanish
- NIP-65: Relay list metadata
- NIP-67: EOSE completeness hint
- NIP-70: Protected events
- NIP-77: Negentropy syncing
- NIP-86: Relay management API
- NIP-89: Recommended application handlers
- NIP-05: Mapping Nostr keys to DNS-based internet identifiers

Spectral Graph Theory for the Nostr Relay–Client Connection Graph

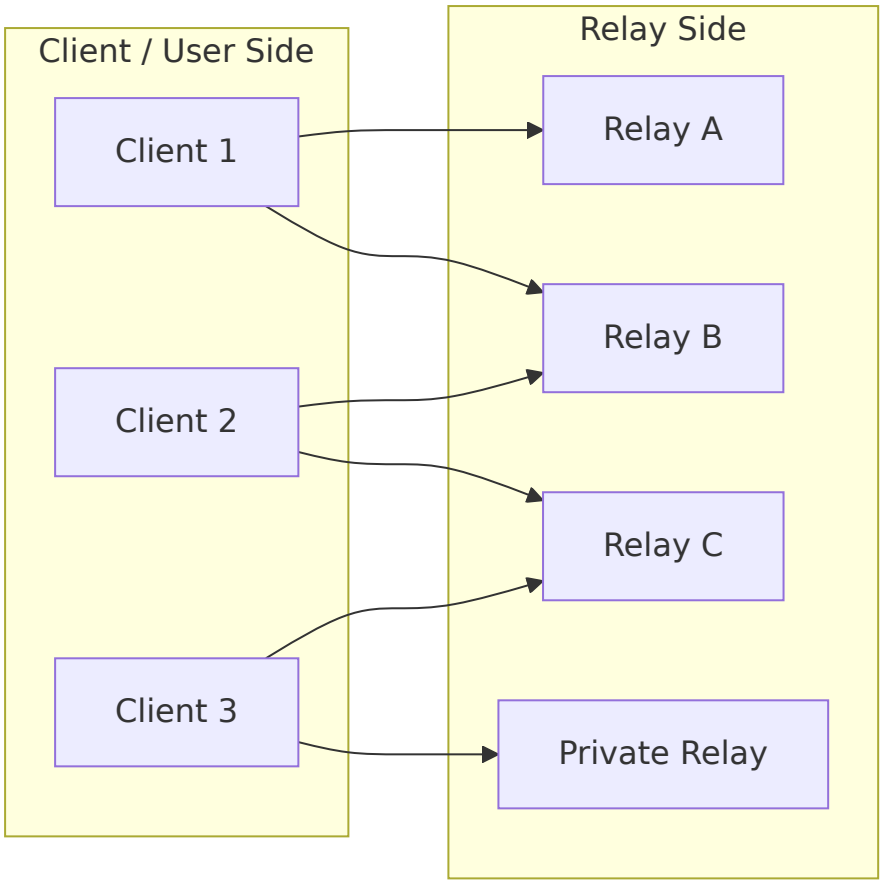
Restarting from first principles: spectral graph theory studies a graph through the “shape” of its connectivity — not just whether it is connected, but how easily it splits, how fast information mixes, where bottlenecks sit, and which nodes or edges are structurally critical. For Nostr, the relevant object is the global relay–client connection graph: clients or users on one side, relays on the other, and edges wherever a client connects to a relay. This graph is dynamic, bipartite, multi-homed, and partly private, so the evaluation must be component-wise and scenario-based rather than a single global score.

---

1. What graph are we evaluating?

The cleanest model is a bipartite connection graph:

- Client/user nodes represent a client session or a user pubkey’s relay usage.
- Relay nodes represent relay endpoints.
- Edges represent an active or declared connection — a client reading from or writing to a relay.
- A second useful view is the relay co-use projection: two relays are linked when the same clients use both. This reveals relay communities, redundancy, and bottlenecks.



Explanation of the diagram.
The graph is bipartite: client nodes only connect to relay nodes. Client 1 is multi-homed across Relays A and B; Client 2 across B and C; Client 3 across C and a private relay. This matters because robustness is not just “how many relays exist” but how many independent paths exist between clients and the relay population. A private relay creates a permissioned island: useful locally, but not necessarily a bridge for the global graph.

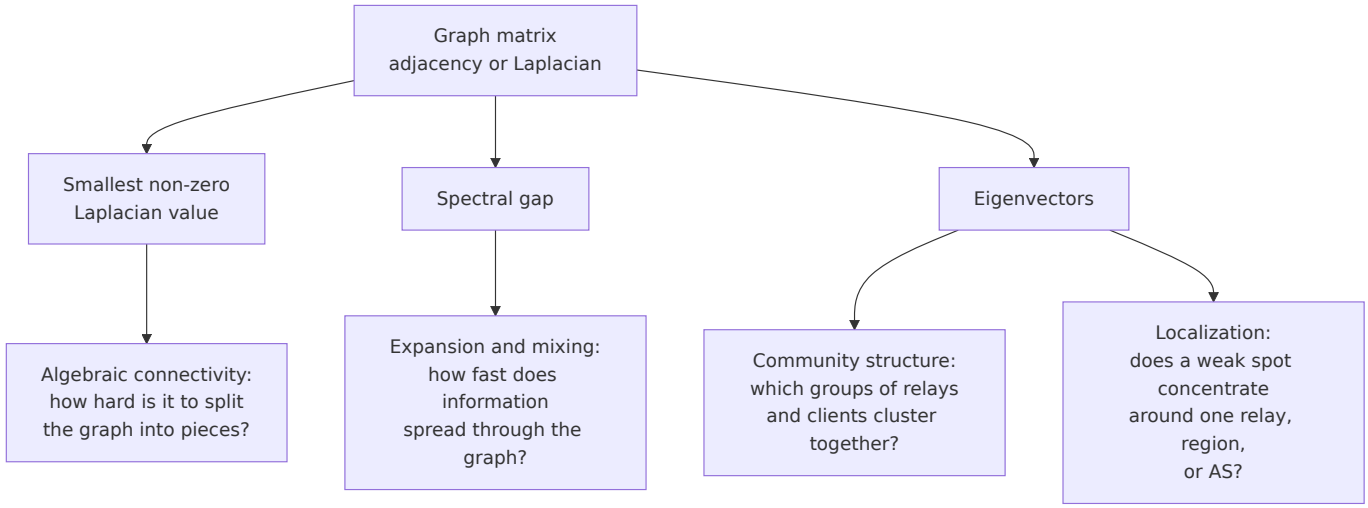
A proper spectral evaluation should distinguish:

1. Observed connection graph — active WebSocket connections, usually not globally observable.
2. Declared relay graph — from NIP-65 relay lists, NIP-11 relay documents, and client defaults.
3. Content replication graph — where posts are stored, which is different from where clients connect.
4. Relay co-use projection — relays linked by shared audiences; this is often the most practical public-data approximation.

---

2. What spectral graph theory measures, in words

Spectral graph theory uses the eigenvalues and eigenvectors of graph matrices — adjacency, Laplacian, or normalized Laplacian — to describe connectivity. We do not need equations to interpret them.



Explanation of the diagram.
The smallest non-zero Laplacian value is the algebraic connectivity: a higher value generally means the graph has no easy cut and is harder to fragment. The spectral gap describes expansion and mixing: a larger gap usually means random walks and information spread mix quickly and there is no severe bottleneck. Eigenvectors reveal communities and localization: if a weak spectral direction is concentrated around one relay, one country, or one autonomous system, that is a structural vulnerability.

For Nostr, the normalized Laplacian is usually more appropriate than the raw Laplacian because relays and clients have very different degrees. A giant relay with thousands of clients and a tiny private relay with three clients should not be treated as structurally equivalent.

---

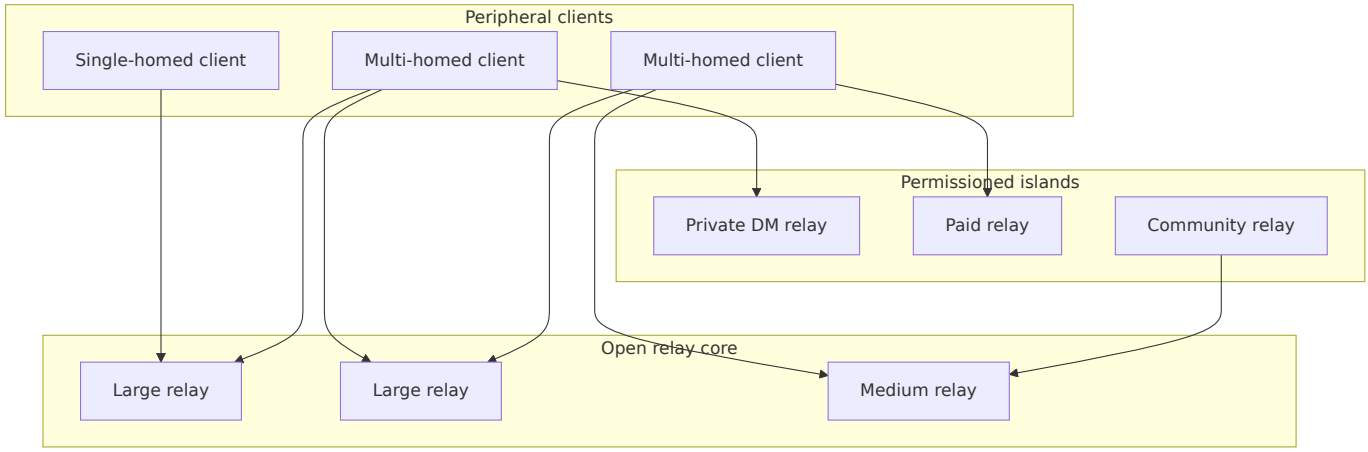
3. Expected structural signature of Nostr

Nostr is not a random graph. It has predictable structural features:

- Core–periphery shape: a relatively small set of popular relays carries much of the visible activity.
- Many low-degree clients: most clients connect to a handful of relays.
- Multi-homing: some clients connect to several relays; some single-home.
- Permissioned islands: paid, private, community, and DM relays create excludable subgraphs.
- Geographic and network clustering: relays sit in different countries and autonomous

systems.

・ Outbox-model routing: clients often connect to authors’ write relays rather than broadcasting to every relay.

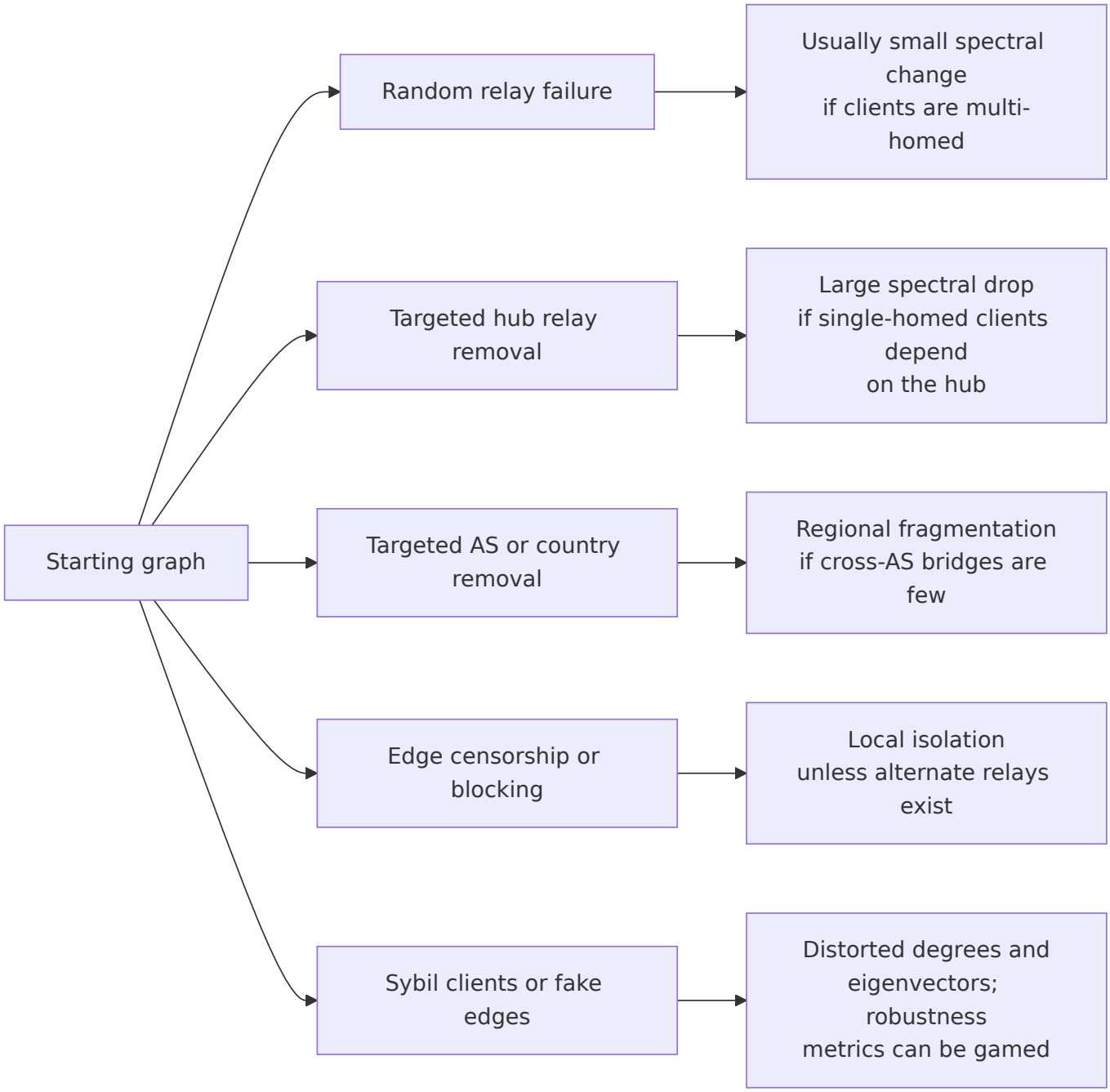


Explanation of the diagram.
The open relay core is densely connected and likely has good local expansion. Peripheral single-homed clients are leaf-like: if their only relay disappears or censors them, they become isolated. Multi-homed clients create alternate paths and improve robustness. Permissioned islands are structurally separate: they may be robust internally but they do not necessarily improve global expansion. This is why a single global spectral number can be misleading — the graph has a well-connected core, fragile leaves, and disconnected or semi-disconnected islands.

---

4. Robustness evaluation

Robustness means: if nodes or edges are removed — relay failure, censorship, blocking, payment lapse, or attack — how much does the graph fragment, and how much does connectivity degrade? Spectral theory evaluates this by tracking algebraic connectivity, spectral gap, component count, and eigenvector localization before and after removal.



Explanation of the diagram.
Random relay failure is usually the mildest scenario: with many relays and some multi-homing, alternate paths remain. Targeted removal of a hub relay is more dangerous: if many clients are single-homed to that relay, those clients become isolated and algebraic connectivity drops sharply. Targeted removal of an autonomous system or country can split the graph along geographic lines if cross-region bridges are rare. Edge censorship — blocking a client from a relay — is local unless the client has no alternate route. Sybil clients or fake edges can distort degree distribution and spectral rankings, so edge weighting by trust, authentication, or verified usage matters.

A qualitative robustness profile:

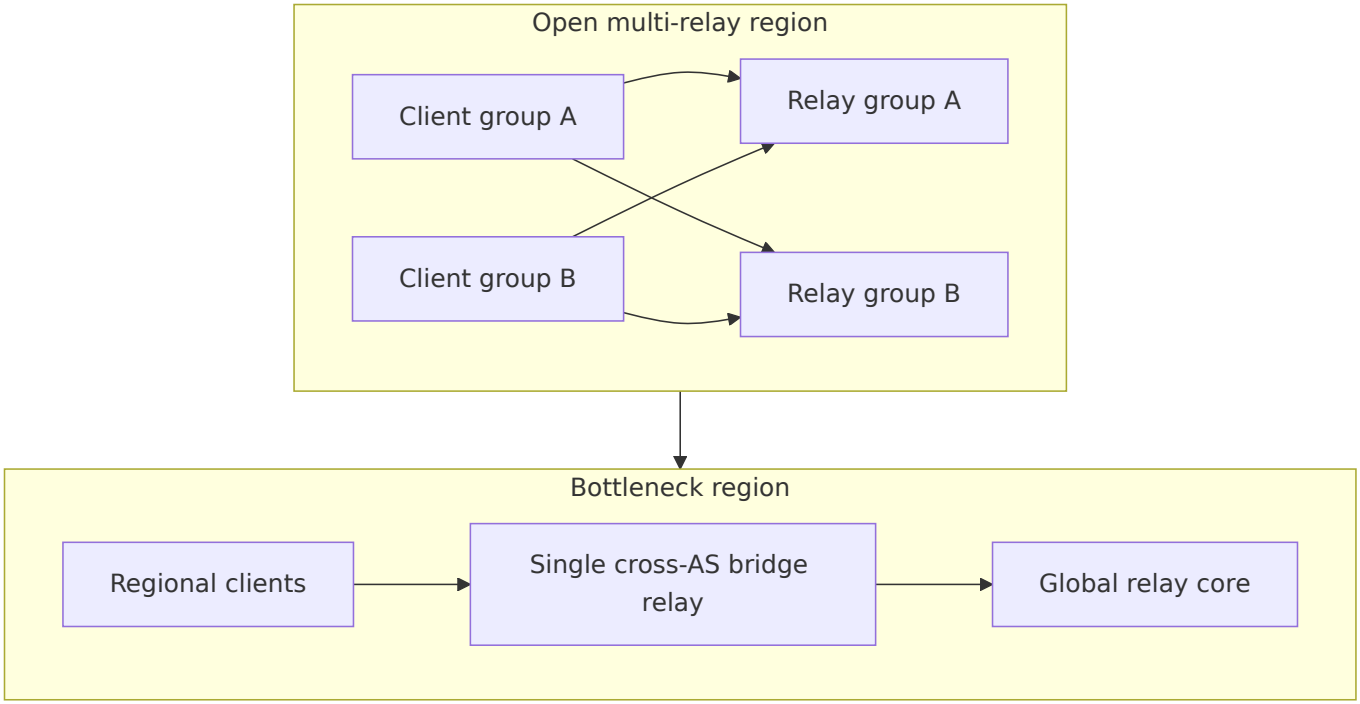
Scenario	Expected effect on relay–client graph	Main spectral signal
Random relay failure	Often survivable with multi-homing	Small change in algebraic connectivity; few new components
Top relay removed	Fragile for single-homed clients	Sharp drop in algebraic connectivity; localized eigenvector around that relay
Top AS or country removed	Risk of regional split	Community structure aligns with geography; cross-AS bridges become critical
Edge censorship	Local isolation if no alternate relay	Component count rises; spectral gap falls in affected region
Paid/private relay growth	More islands, less global mixing	Component-wise expansion matters more than global expansion
Sybil edge injection	Metrics can be inflated or distorted	Degree distribution skew; eigenvectors localize around fake clusters

The empirical Nostr data — high content availability after removing top relays — is encouraging for content replication, but it does not automatically prove connection-graph spectral robustness. A post can remain available on another relay while a single-homed client still loses its only connection path. The two layers must be evaluated separately.

---

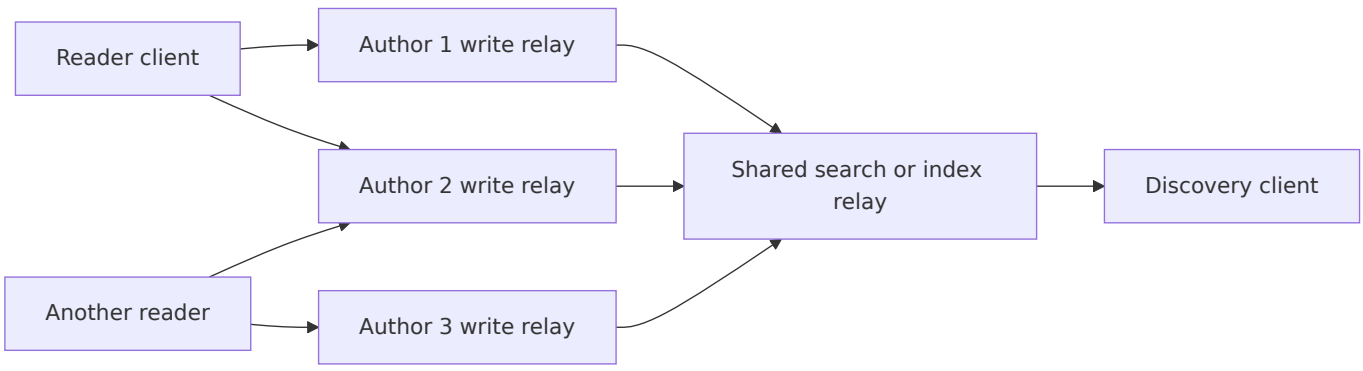
5. Expansion properties

Expansion asks: does every reasonably sized group of clients or relays have many connections to the rest of the graph? Good expansion means information spreads quickly, random walks mix fast, and no small set can be cut off cheaply. Poor expansion means bottlenecks: a small bridge set controls access between communities.



Explanation of the diagram.
The open multi-relay region has overlapping connections: clients in group A use relays in both groups, and clients in group B do the same. This creates good expansion and redundancy. The bottleneck region has many regional clients depending on one cross-AS bridge relay. That single bridge is a spectral weak point: removing it or censoring it can split the regional community from the global core. In spectral terms, the graph would show a low expansion direction concentrated on that bridge.

Nostr’ s outbox model changes expansion in an important way. Instead of broadcasting to every relay, clients route toward authors’ write relays. This creates many targeted, sparse subgraphs. Expansion within a community can be good, but cross-community expansion depends on shared relays, quote/repost paths, search relays, and relay-to-relay syncing such as negentropy.



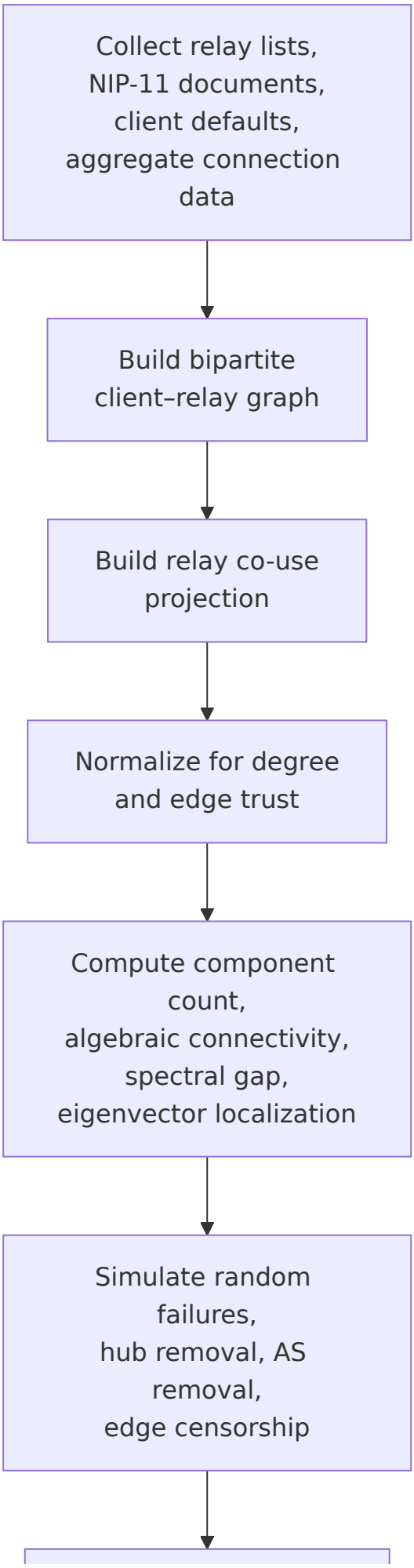
Explanation of the diagram.
The outbox model creates targeted routing: readers connect to the write relays of authors

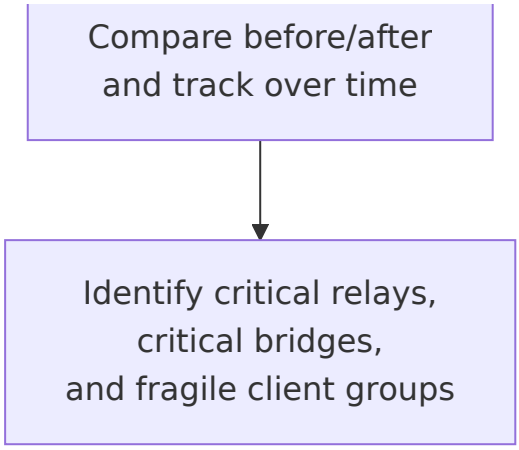
they follow. This reduces unnecessary broadcast load and can improve privacy, but it also means the graph is not uniformly well mixed. Shared search relays, index relays, popular relays, and relay-to-relay sync become the bridges that determine cross-community expansion. If those bridges are few, the global graph may have good local expansion but weak global expansion.

---

6. How to evaluate it properly

A rigorous spectral evaluation should be a pipeline, not a one-time number.


Collect relay lists, NIP-11 documents, client defaults, aggregate connection data
Build bipartite client–relay graph
Build relay co-use projection
Normalize for degree and edge trust
Compute component count, algebraic connectivity, spectral gap, eigenvector localization
Simulate random failures, hub removal, AS removal, edge censorship



Explanation of the diagram.
Start with public and aggregate data: relay lists, relay information documents, client defaults, and anonymized connection statistics. Build the bipartite graph and its relay co-use projection. Normalize for degree and edge trust so giant relays and fake edges do not dominate. Compute component count, algebraic connectivity, spectral gap, and eigenvector localization. Then simulate failures and censorship scenarios. Finally, compare before and after and track changes over time. Because Nostr connections are dynamic and partly private, the output should be a spectral robustness profile over time and per component, not a single global score.

---

7. Overall assessment

Structural robustness: Nostr’ s relay–client graph is likely robust against random relay failure because of relay multiplicity and multi-homing, but conditionally fragile against targeted removal of top relays, critical cross-AS bridges, or relays serving single-homed clients. Permissioned and private relays increase local sovereignty but also create islands that reduce global expansion. Sybil or fake edges can distort spectral metrics unless edges are weighted by authentication, trust, or verified usage.

Expansion properties: The open relay core probably has good local expansion, especially where clients multi-home and relays are diverse. Global expansion is weaker: the graph is core–periphery, geographically clustered, and partly permissioned. Cross-community and cross-region bridges — shared popular relays, search relays, index relays, and relay-to-relay sync — are the critical expansion points. The outbox model improves targeted efficiency but makes those bridges more important for global mixing.

Bottom line: Spectral graph theory is the right lens because it reveals not just whether Nostr is “decentralized” in node count, but whether its connection structure has cheap cuts, fragile bridges, and localized weak points. The expected profile is high local redundancy,

moderate-to-good robustness under random failure, targeted vulnerability around hubs and cross-AS bridges, and component-wise rather than uniformly high expansion. The practical goal is to monitor algebraic connectivity, spectral gap, component count, and eigenvector localization across scenarios, then use that profile to encourage relay diversity, multi-homing, and critical-bridge redundancy.

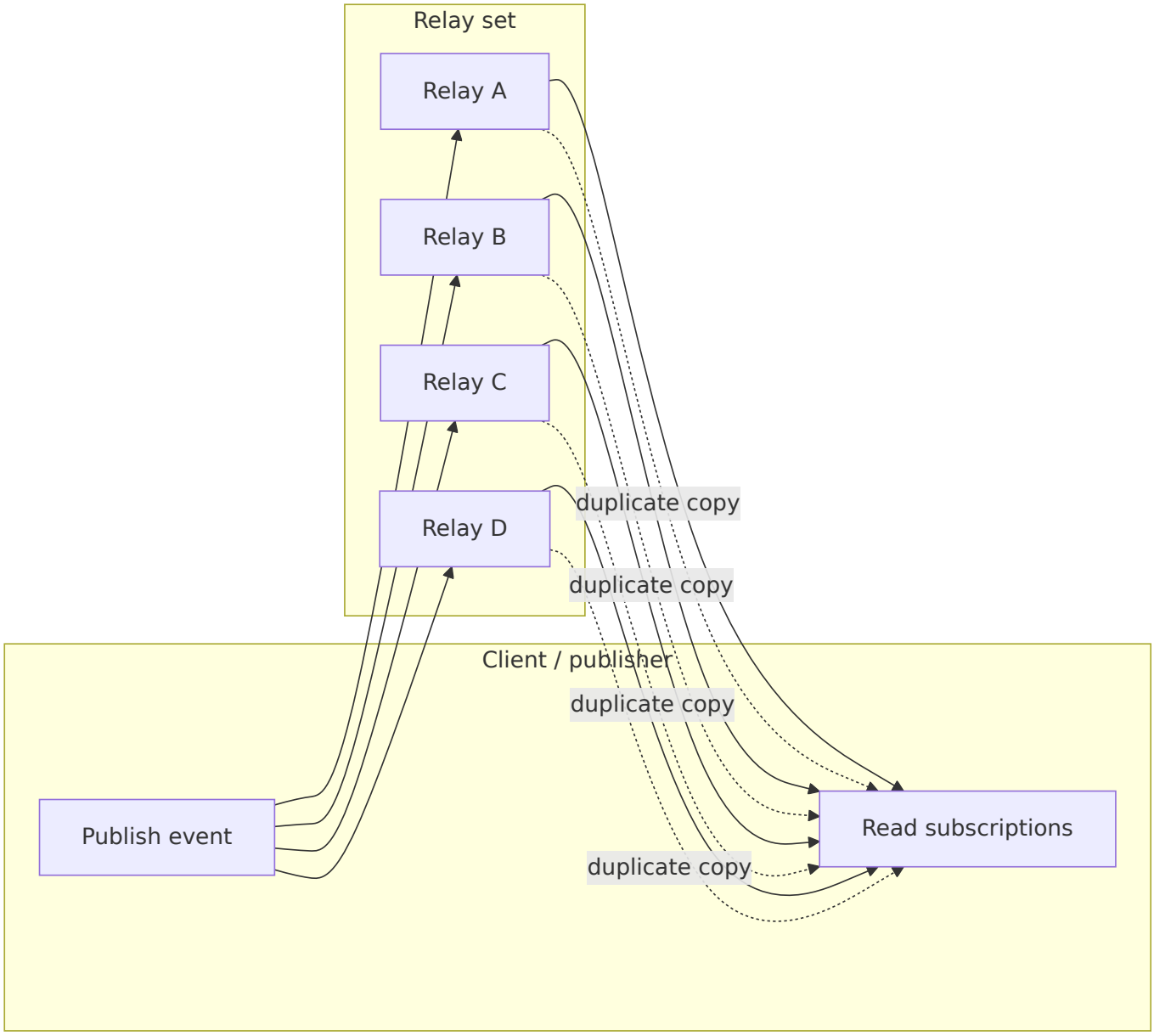
Network Routing Efficiency: NIP-65 Outbox vs. Traditional Pub/Sub

Restarting from first principles: “routing efficiency” in Nostr means how quickly a client can find the events it wants, and how much redundant network and storage work is spent doing so. The two models compared here are the traditional broadcast pub/sub model — publish to many relays, subscribe to many relays — and the NIP-65 outbox model — publish to an author’s declared write relays, and query those same write relays when reading that author’s events. The comparison is about multi-path latency, bandwidth overhead, and the structural trade-offs between them.

---

1. Traditional pub/sub: broadcast routing

In the traditional model, a client connects to a set of relays and treats them as interchangeable pipes. It publishes an event to every relay in its connection set. It reads by opening subscriptions on every relay in its connection set and accepting whatever comes back. The same event is often stored on many relays and returned many times.

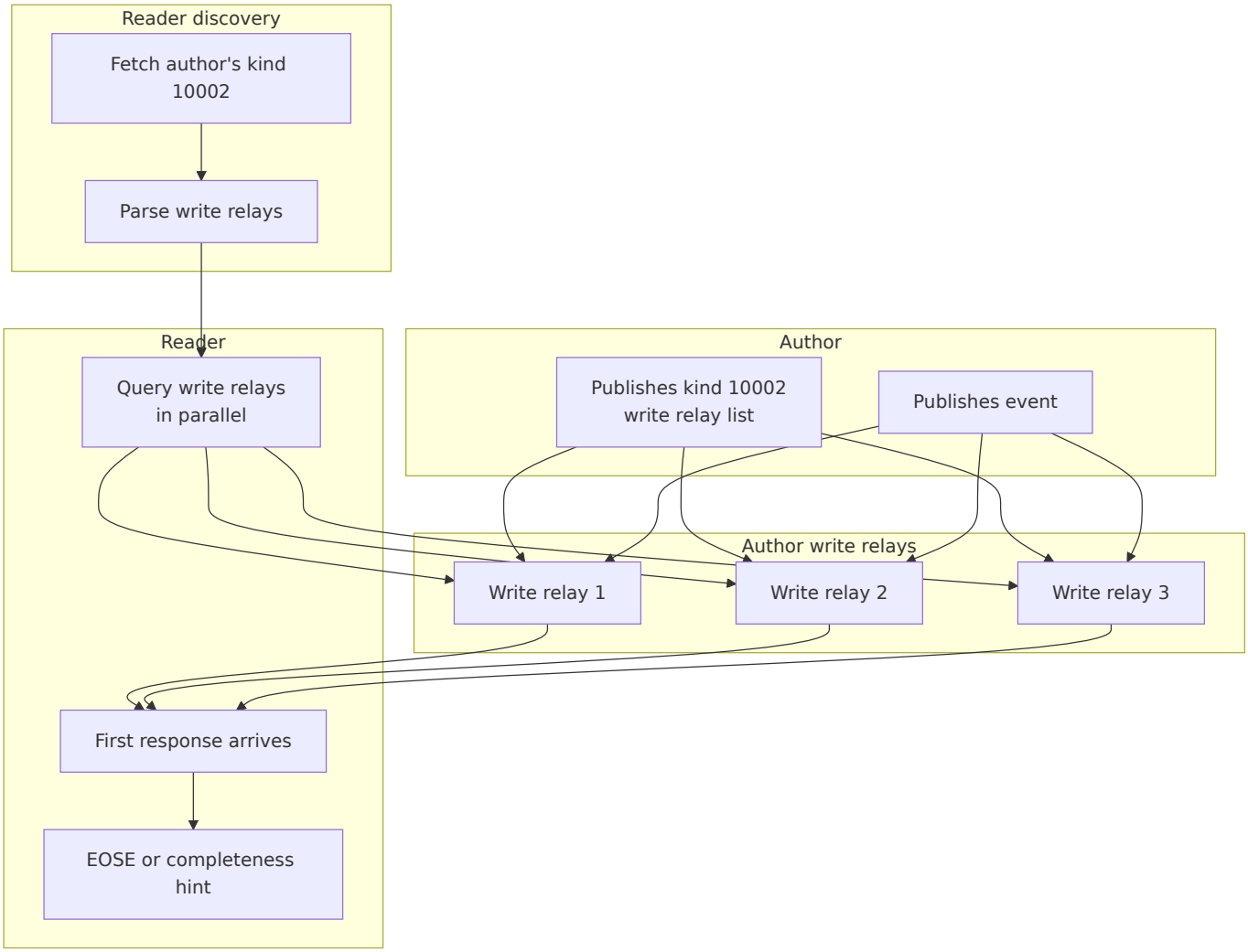


Explanation of the diagram.
The publisher sends the same event to every relay it is connected to. The reader subscribes to all of those relays and receives duplicate copies of the same event. The dashed arrows show duplicate delivery. This model is simple and has high path redundancy, but it pays for that redundancy in storage, bandwidth, and client-side duplicate processing. The economics analysis describes this as the source of “616 million replications” across 17.8 million posts — an average of about 34.6 replications per post — and “98.2% redundant retrievals.” The same analysis notes that capping replications at 10–20 relays eliminates roughly 61.7% to 77.9% of replications while maintaining over 99% availability. In other words, the broadcast model is over-replicating relative to what availability actually requires.

---

2. NIP-65 outbox: targeted routing

NIP-65 introduces the outbox model. A user publishes a kind 10002 replaceable event listing relays with r tags and optional read or write markers. A relay with no marker is both read and write. When reading events from a user, clients should use that user's write relays. When reading events about a user — where the user was mentioned — clients should use that user's read relays. When publishing, clients should send the event to the author's write relays and to all read relays of each tagged user, and should also send the author's kind 10002 event to all relays where the event was published.



Explanation of the diagram.

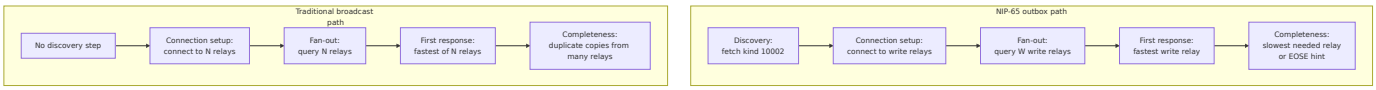
The reader first discovers the author's declared write relays from the author's kind 10002 event. Then the reader queries those write relays in parallel. The first response may come from any of them; completeness depends on receiving enough responses or an end-of-stored-events signal. This is targeted routing: the reader does not query every relay in the network, only the relays the author actually uses for writing. The author's event is published to those same write relays. Mentions are routed differently: they go to the mentioned user's read relays. This separation is the core of the outbox model.

The economics document describes the outbox model as a routing mechanism that “reduces relay load” and “incentivizes relay quality.” It also notes that NIP-65 “can switch relays seamlessly if fees become exorbitant or moderation policies change.” The model is structurally more efficient because it aligns reading with where the author actually publishes.

---

3. Multi-path latency analysis

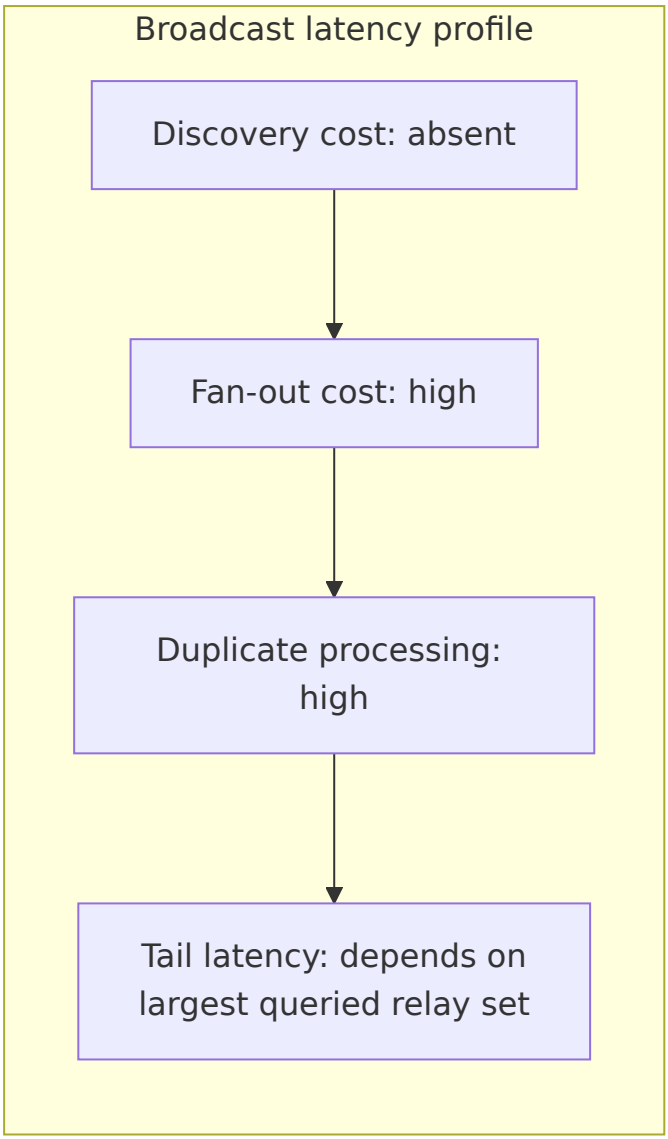
“Multi-path latency” means the time it takes to get a usable response when multiple relays are queried in parallel. The important components are: discovery latency, connection setup, query fan-out, first response, and completeness.

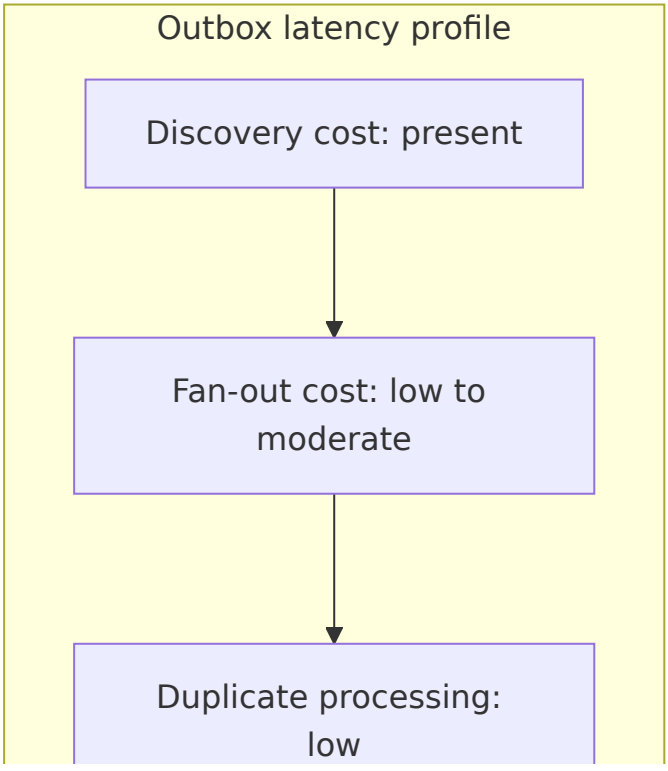


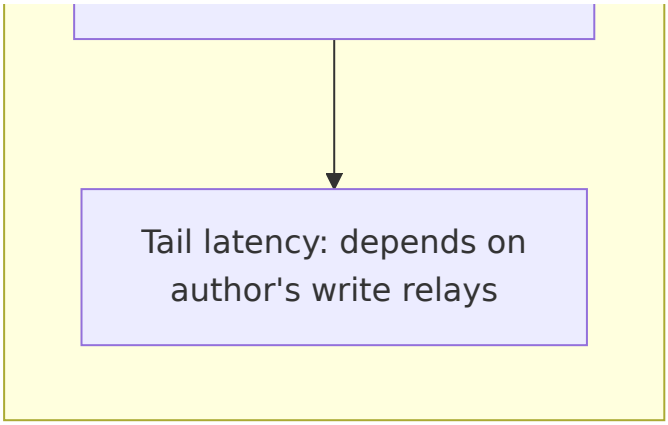
Explanation of the diagram.
The outbox path adds a discovery step: the reader must fetch and parse the author’s kind 10002 event. That is additional latency that the broadcast model does not have. However, the outbox path then queries only a small number of write relays — typically two to four per NIP-65 guidance — which means less connection setup, less fan-out, and less duplicate processing. The broadcast path has no discovery step but connects to and queries many relays, often with heavy duplication. The first response in broadcast may be fast because many relays are queried in parallel, but the client must then deduplicate and process redundant copies. The outbox path’s first response is the fastest among the author’s write relays; its completeness depends on the slowest necessary write relay or on an EOSE completeness hint such as NIP-67.

The trade-off is therefore:

- Outbox: slightly higher discovery latency, lower fan-out and duplicate-processing latency, and more predictable completeness because the queried relays are the ones the author actually uses.
- Broadcast: lower discovery latency, but higher fan-out, higher duplicate-processing latency, and less predictable completeness because the relay set is not tailored to the author.


Broadcast latency profile
Discovery cost: absent
Fan-out cost: high
Duplicate processing: high
Tail latency: depends on largest queried relay set


Outbox latency profile
Discovery cost: present
Fan-out cost: low to moderate
Duplicate processing: low

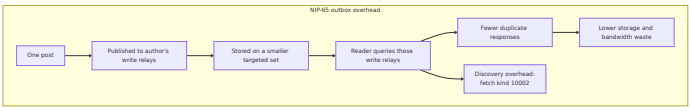


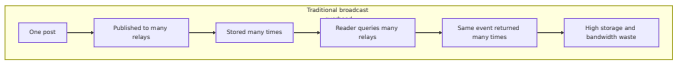
Explanation of the diagram.
The outbox profile pays a discovery cost but avoids heavy fan-out and duplicate processing. Its tail latency is tied to the author’ s write relays: if those relays are fast and reliable, the outbox path is efficient; if they are slow or offline, the reader may need relay hints or fallback relays. The broadcast profile avoids discovery but pays high fan-out and duplicate-processing costs. Its tail latency is tied to the largest relay set being queried. In practice, outbox latency is often better for established authors with well-maintained relay lists, while broadcast may appear faster in small or poorly configured relay sets because it brute-forces redundancy.

---

4. Bandwidth overhead analysis

Bandwidth overhead has two sides: storage replication and retrieval duplication. The broadcast model creates both. The outbox model reduces both by aligning publication and retrieval with a small set of declared relays.





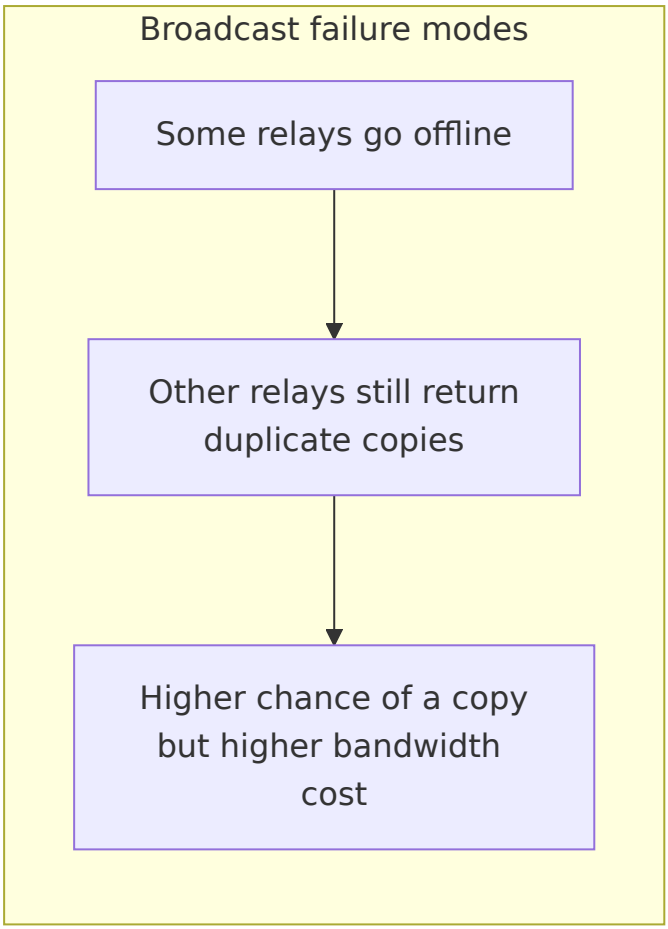
Explanation of the diagram.
The broadcast side shows the full waste chain: a post is published to many relays, stored many times, queried from many relays, and returned many times. The outbox side shows targeted publication and retrieval: the post goes to the author’ s write relays, is stored on a smaller set, and the reader queries those same relays. The discovery overhead of fetching kind 10002 is a small, one-time cost that is usually cached. The economics analysis quantifies the broadcast problem: 17.8 million posts produce 616 million replications, with 98.2% redundant retrievals. Capping replications at 10–20 relays eliminates a large fraction of that waste while preserving availability. The outbox model is the protocol-level mechanism that moves the system toward that cap without requiring a global coordination layer.

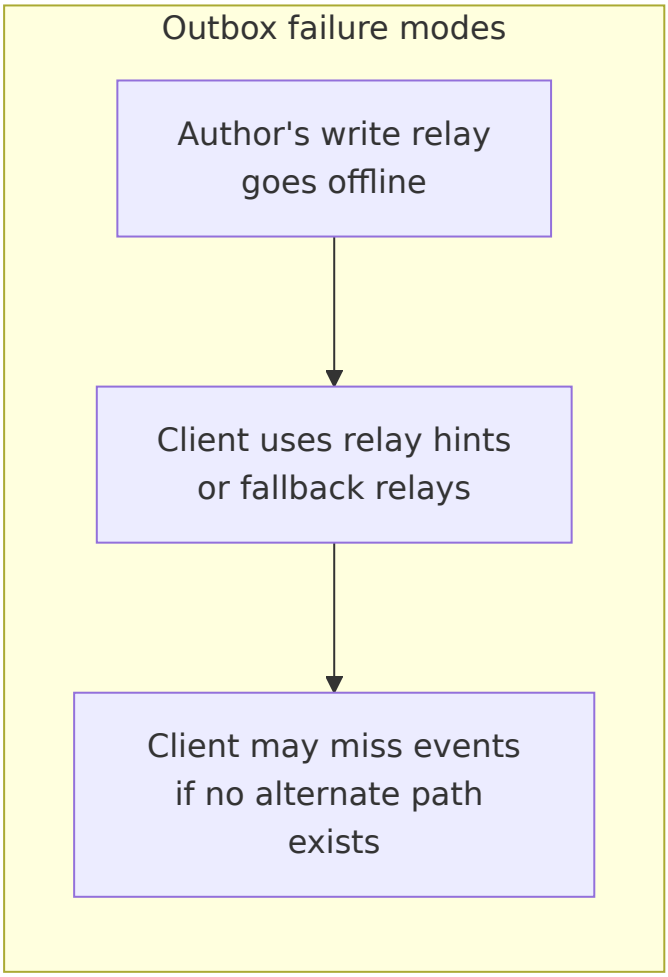
There is an important caveat: the outbox model does not automatically eliminate replication. If an author lists several write relays, the event is still replicated across those relays. The efficiency gain comes from replacing broadcast to every relay with targeted publication to a small declared set, and from replacing query every relay with query the author’ s declared write relays. The economics document also warns that free relays are financially fragile — 95% of free relays cannot cover costs, and 132 dead relays were observed — so the outbox model’ s efficiency must be paired with relay sustainability mechanisms such as zaps, paid relays, or value-added services.

---

5. Reliability and tail latency

Multi-path routing is not only about average latency; it is also about tail latency and failure behavior. The outbox model and broadcast model fail differently.


Broadcast failure modes
Some relays go offline
Other relays still return duplicate copies
Higher chance of a copy but higher bandwidth cost


Outbox failure modes
Author's write relay goes offline
Client uses relay hints or fallback relays
Client may miss events if no alternate path exists

Explanation of the diagram.
In the outbox model, if an author’ s write relay goes offline, the reader may need relay hints from e or p tags, or fallback to other relays. NIP-65 guidance says clients should use relay hints in tags to find related events when building threads, and should use the author’ s write relays where relay hints did not resolve the event. If no alternate path exists, the reader may miss events. In the broadcast model, if some relays go offline, other relays may still return duplicate copies, so the chance of getting a copy is higher — but the bandwidth cost is also higher. This is the classic redundancy trade-off: broadcast buys reliability with waste; outbox buys efficiency with a dependency on correct relay lists and relay liveness.

NIP-67 EOSE completeness hints help the outbox model here: a relay can signal “finish” when it has sent every stored matching event, or “more” when it holds more than it has sent. This reduces the ambiguity of pagination and helps clients know when they have enough paths. NIP-77 negentropy syncing reduces bandwidth during relay-to-relay or client-to-relay synchronization by using set reconciliation instead of downloading full event lists. NIP-45 counting and HyperLogLog reduce the compute cost of counts and enable cross-relay aggregation. These are complementary mechanisms: outbox defines where to route; NIP-67, NIP-77, and NIP-45 make the routing and synchronization more efficient.

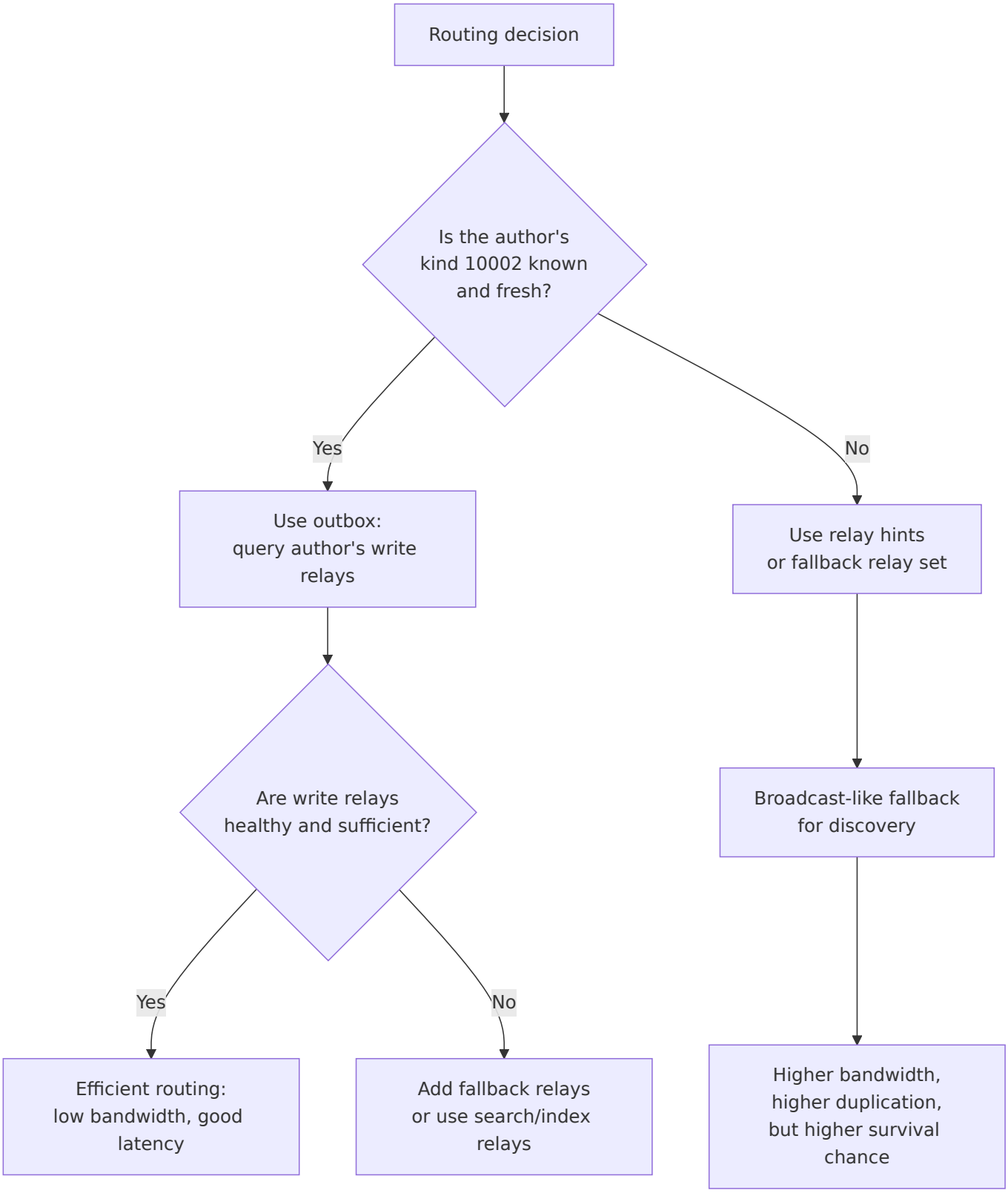
---

6. Trade-off matrix

Dimension Traditional broadcast pub/sub NIP-65 outbox
Discovery step None Fetch kind 10002 relay list
Connection setup Many relays Fewer, targeted relays
Fan-out High Low to moderate
Duplicate retrieval High Low
Storage replication High and often excessive Targeted, closer to availability needs
First-response latency Often fast due to brute-force parallelism Fast if author’ s write relays are healthy
Tail latency Tied to largest relay set Tied to author’ s write relays and hints
Failure behavior Redundant copies increase survival Depends on relay hints and fallback paths
Bandwidth efficiency Poor at scale Better at scale
Operational simplicity Simple but wasteful More complex but more efficient
Dependency Relay set choice Correct kind 10002 publication and relay liveness

---

7. When each model wins



Explanation of the diagram.

The outbox model wins when the author’ s kind 10002 is known, fresh, and points to healthy write relays. In that case, routing is efficient: low duplicate bandwidth, targeted fan-out, and predictable latency. When the relay list is missing or stale, the client falls back to relay hints or a broader relay set, which behaves more like broadcast. When write relays are

unhealthy, the client may add fallback relays or use search and index relays. The broadcast model wins in simplicity and brute-force survival, but it loses on bandwidth efficiency and duplicate processing. For a network with 17.8 million posts and hundreds of millions of replications, the broadcast model’ s waste is structural; the outbox model is the protocol’ s answer to that waste.

---

8. Practical optimizations that change the comparison

Several NIPs modify the latency and bandwidth profile of the outbox model:

- NIP-67 EOSE completeness hint: tells the client whether the relay has sent everything or whether more exists, reducing unnecessary pagination and improving completeness confidence.
- NIP-77 negentropy syncing: reduces bandwidth during synchronization by reconciling sets rather than downloading full event lists.
- NIP-45 counting and HyperLogLog: reduces compute cost for counts and enables cross-relay aggregation without downloading all events.
- Relay hints in e and p tags: provide alternate paths when the author’ s write relays do not resolve an event.
- Search relays (NIP-50): centralize search workloads so general relays do not need to carry search indexes.
- Relay information documents (NIP-11): advertise limits, fees, and supported NIPs so clients can route more intelligently.
- Caching kind 10002: reduces the discovery latency cost of the outbox model.

---

9. Conclusion

The traditional broadcast pub/sub model is simple and has high path redundancy, but it over-replicates storage and retrieval. The economics analysis makes the waste concrete: 17.8 million posts, 616 million replications, 34.6 average replications per post, and 98.2% redundant retrievals. Capping replications at 10–20 relays eliminates a large fraction of that waste while preserving availability.

The NIP-65 outbox model is structurally more efficient: it aligns publication and retrieval with the author’ s declared write relays, reduces fan-out and duplicate processing, and gives clients a principled way to choose relays. It adds a discovery step and depends on correct relay lists and relay liveness, but those costs are small compared with the bandwidth and storage waste of broadcast at scale. The outbox model’ s multi-path latency is usually

competitive when author write relays are healthy; its tail latency depends on relay hints and fallback paths. The broadcast model’ s first-response latency can appear fast because it brute-forces redundancy, but that speed is paid for in duplicate bandwidth and client processing.

Bottom line: The outbox model is the more bandwidth-efficient routing model for Nostr at scale. Traditional broadcast is simpler and more redundant, but its redundancy is excessive relative to availability needs. The practical answer is a layered approach: use outbox routing when kind 10002 is known and healthy, use relay hints and fallback relays when it is not, and use NIP-67, NIP-77, NIP-45, and NIP-50 to reduce the remaining latency and bandwidth costs.