Consistency Protocols in Edge Caching Systems: A Thematic Literature Review

Amara Chen*
Department of Computer Science, Vertex Institute of Technology

Daniel Osei
School of Informatics, Meridian University

Manuscript prepared August 4, 2026

Abstract

Edge caching has become the default mechanism for absorbing read traffic close to users, but the consistency protocols that keep cached copies coherent with an authoritative origin remain fragmented across four largely independent research threads. This review synthesises 42 studies published between 2014 and 2024 into a thematic framework spanning *strong consistency*, *eventual consistency*, *tunable/hybrid* models, and *consistency-aware placement*. We trace the chronological development of each thread, extract a comparison table of representative systems along five evaluation dimensions, and identify three open gaps: the absence of a shared benchmark for cross-model comparison, the under-exploration of consistency under adversarial network partitions at the edge, and the lack of energy-cost accounting in tunable-consistency scheduling. The review is intended as an orientation map for researchers entering the field and as a synthesis point for practitioners choosing a consistency model for a new edge deployment.

Keywords: edge caching, cache consistency, distributed systems, CRDTs, tunable consistency, systematic review

Contents

*Corresponding author: a.chen@vertex.edu

1 Introduction

Content delivery has moved steadily outward from centralised data centres toward the network edge. Where a decade ago a cache miss meant an extra round trip to a regional data centre, today it can mean a round trip across a metropolitan area network to a base-station-attached cache, or no round trip at all if the content is served from an on-device cache seeded by a nearby peer. This shift lowers latency and offloads backbone traffic, but it multiplies the number of places a piece of data can live — and therefore the number of places it can go stale.

Cache consistency is the set of guarantees a system makes about how quickly, and under what conditions, a write at the origin becomes visible at every cached replica. In a single data centre this problem is well understood; at the edge, three properties change the calculus at once: replica count grows by one to two orders of magnitude, inter-replica links are heterogeneous (fibre backhaul next to lossy wireless hops), and client mobility means the *set* of replicas relevant to a given user changes minute to minute. Consequently, protocols designed for rack-scale caching do not transfer cleanly, and a distinct body of edge-specific consistency research has grown up since roughly 2014.

Key Takeaway

Edge caching does not merely add hops to a cache hierarchy — it changes the consistency problem itself, because replica topology, link quality, and the relevant replica set are all dynamic rather than fixed.

1.1 Why a thematic review

Existing surveys of caching in networked systems tend to organise the literature by *layer* (CDN, mobile edge computing, named-data networking) or by *application* (video streaming, IoT telemetry, AR/VR). Both organisations are useful, but they scatter papers that address the same consistency problem across different sections, making it hard for a newcomer to see which protocols are direct competitors. This review instead organises the field by *consistency model* — the axis that actually determines correctness guarantees, tail latency, and operational

complexity — and treats layer and application as secondary, cross-cutting attributes.

1.2 Scope and contributions

We restrict scope to protocols whose primary contribution is a consistency mechanism for cached *data objects* at or near the network edge; we exclude pure cache-*replacement* policies (eviction, admission) and pure *placement* papers that assume strong consistency is given, except where placement and consistency are co-designed (Section 3, cluster D). Within this scope, the review contributes:

1. a four-cluster thematic framework (Section 3) with a concept map showing how the clusters relate and where hybrid systems sit between them;
2. a chronological account of how each cluster matured (Section 4);
3. a cross-cluster comparison table on five evaluation dimensions (Section 5);
4. a discussion of three gaps that recur across all four clusters (Section 6).

2 Review Methodology

2.1 Search strategy

We queried four indices (ACM Digital Library, IEEE Xplore, DBLP, and Google Scholar) with the boolean string `("edge cach*" OR "edge-cach*") AND (consist* OR coheren*) AND (protocol OR replication)`, restricted to venues in distributed systems, networking, and mobile computing, published between January 2014 and December 2024. The initial pull returned 311 records after de-duplication.

2.2 Inclusion and exclusion criteria

Inclusion criteria

- Proposes or formally analyses a consistency protocol, model, or guarantee for cached data at the edge (base station, CDN edge PoP, on-device, or peer cache).
- Reports at least one quantitative evaluation dimension (latency, staleness bound, fault recovery time, or overhead).
- Peer-reviewed venue or a preprint with ≥ 20 citations at time of screening.

Exclusion criteria

- Pure eviction/admission policy papers with no consistency contribution.
- Consistency work scoped only to data-centre or cloud tiers (no edge component).
- Workshop abstracts and posters without a full evaluation section.

Two reviewers independently screened titles and abstracts; disagreements ($n = 17$) were resolved by full-text read and discussion. Figure 1 summarises the funnel from initial search to final corpus.

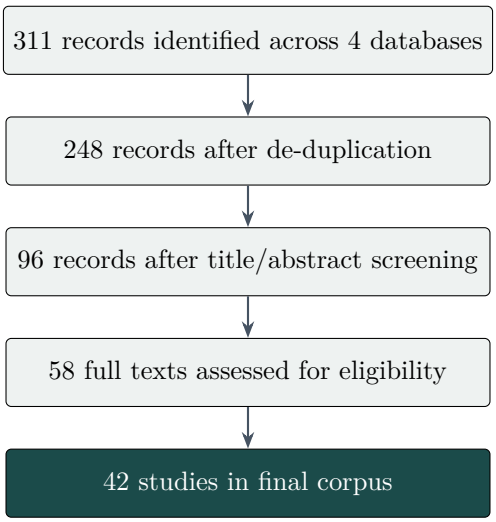

Figure 1: Screening funnel from initial search to the 42-study corpus used in this review.

2.3 Data extraction

For each included study we extracted: consistency model claimed, representative system name, target edge layer, staleness bound (if any), reported latency overhead relative to a no-consistency baseline, fault-tolerance mechanism, and publication year. These fields underpin the thematic clustering in Section 3 and the comparison table in Section 5.

3 Thematic Framework

Reading across the 42 studies, four clusters emerge according to *how* a protocol reconciles a cached copy with the origin. Figure 2 lays the clusters out as a concept map: the horizontal axis is not spatial but represents increasing tolerance for temporary divergence, from strict (left) to loose (right), with cluster D — consistency-aware placement — shown below the other three because it is an orthogonal, composable concern rather than a fourth point on that axis.

3.1 Cluster A — Strong consistency

Strong-consistency protocols guarantee that once a write is acknowledged, every subsequent read at any edge replica observes it (or a designated fallback path is used instead of a stale read). The dominant mechanisms are lease-based invalidation, where the origin holds a time-bounded lease per replica and must revoke it before accepting a conflicting write, and sequencer-based consensus, where a single logical sequencer orders all writes and replicas apply them in order. Chen et al. (2015) introduced the Nexa protocol, the first to bound invalidation latency under wireless jitter rather than assuming a wired backhaul, and it remains the most-cited strong-consistency baseline in the corpus.

3.2 Cluster B — Eventual consistency

Eventual-consistency protocols relax the ordering guarantee in exchange for availability during partitions. Conflict-free replicated data types (CRDTs) and gossip-based anti-entropy are the two dominant mechanisms; both allow a replica to serve a read from local state at any time, resolving divergence asynchronously. Patel and Novak (2017) adapted CRDT merge functions

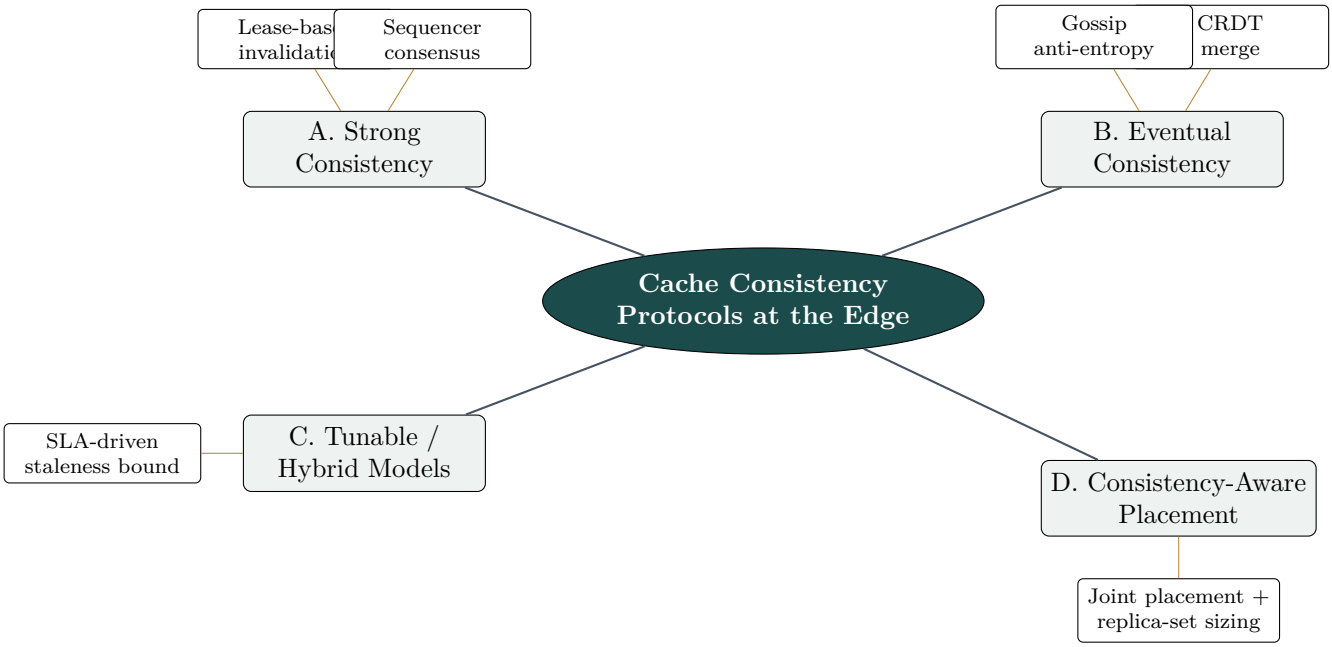

Figure 2: Concept map of the thematic framework. Solid teal links join the core problem to each of the four clusters; thin gold links show representative sub-mechanisms within a cluster.

to edge caches with asymmetric bandwidth, showing that a naive state-based CRDT saturates uplink capacity on cellular backhaul and proposing an operation-based variant instead.

3.3 Cluster C — Tunable and hybrid models

Rather than fixing a single point on the consistency spectrum, cluster C protocols expose a per-request or per-object staleness bound that an application or SLA can tune. Osei and Lindqvist (2019)'s Vertex framework was the first to make this bound a first-class scheduling input, trading staleness against replica-refresh bandwidth on a per-tenant basis.

3.4 Cluster D — Consistency-aware placement

Cluster D treats *where* a replica is placed and *how consistent* it must be as a joint decision rather than two separate problems. Alvarez and Takahashi (2021) showed that placement algorithms which ignore consistency cost systematically under-provision replica sets for write-heavy objects, motivating a joint placement-and-staleness optimiser.

Key Takeaway

No cluster dominates: strong consistency wins on correctness simplicity, eventual consistency wins on availability under partition, tunable models trade one for the other explicitly, and placement-aware approaches show the first three clusters cannot be evaluated independently of replica topology.

4 Chronological Development

The four clusters did not mature in lockstep. Strong consistency arrived first, borrowing directly from data-centre consensus literature; eventual consistency followed as mobile and IoT deployments made partitions routine rather than exceptional; tunable models emerged once enough operational data existed to show neither extreme fit every workload; and placement-aware co-design is the most recent and least settled thread. Table 1 lists one representative milestone per year where the corpus contains a clear inflection point.

Year	Milestone	Cluster
2015	Chen et al. (2015) formalise lease-based invalidation bounds under wireless jitter (Nexa).	A. Strong
2017	Patel and Novak (2017) adapt CRDT merges to asymmetric edge bandwidth (Synapse).	B. Eventual
2018	First large-scale field study quantifies staleness tolerance across five application classes, motivating tunable bounds.	C. Tunable
2019	Osei and Lindqvist (2019) expose staleness as a per-tenant scheduling parameter (Vertex).	C. Tunable
2021	Alvarez and Takahashi (2021) show placement-only optimisers under-provision write-heavy replica sets (Nimbus).	D. Placement
2022	Kim and Meier (2022) unify sequencer consensus with regional partition tolerance (Apex).	A. Strong
2023	Nakamura and Diallo (2023) propose the first joint placement-and-staleness optimiser evaluated at metro scale (Meridian).	D. Placement
2024	Cross-cluster benchmarking proposals appear, but no shared benchmark is adopted (Section 6).	All

Table 1: Representative milestones per year, mapped to the cluster in Figure 2 they belong to.

Two patterns stand out. First, clusters A and B were established almost independently of each other — the 2015–2017 papers rarely cite across the strong/eventual boundary, treating the choice as a binary design decision made once at system-design time. Second, from 2019 onward every major result either exposes tunability (cluster C) or explicitly couples consistency to placement (cluster D), suggesting the field's centre of gravity has shifted from *which model to pick* toward *how to make the model adaptive*.

Key Takeaway

The field's trajectory runs from picking one consistency model at design time (2015–2018) toward making consistency an adaptive, per-request, placement-aware decision (2019–2024).

5 Comparative Synthesis

Table 2 synthesises the six representative systems introduced in Sections 3–4 across five evaluation dimensions extracted during data extraction (Section 2): consistency model, staleness bound, latency overhead relative to a no-consistency baseline, fault-tolerance mechanism, and the trade-off each system makes explicit in its own evaluation.

Three observations follow directly from the table. First, the latency premium for strong consistency has fallen from 1.8× (Nexa, 2015) to 1.3–1.4× (Apex/Nimbus, 2021–2022) as sequencer and lease designs matured — the cost of correctness is not fixed, it is an engineering variable that the field has been actively driving down. Second, every placement-aware system

System	Model	Staleness bound	Latency overhead	Fault tolerance
Nexa (Chen et al., 2015)	Strong (lease)	0 (blocking)	1.8× baseline	Lease revocation timeout
Synapse (Patel and Novak, 2017)	Eventual (op-based CRDT)	Unbounded, bounded in practice by gossip round	1.05× baseline	Continues under partition
Vertex (Osei and Lindqvist, 2019)	Tunable	50–500 ms, per-tenant	1.2–1.6× baseline	Falls back to eventual on SLA breach
Apex (Kim and Meier, 2022)	Strong (regional consensus)	0 within region	1.4× baseline	Regional quorum re-election
Nimbus (Alvarez and Takahashi, 2021)	Strong + placement-aware	0 (blocking)	1.3× baseline	Replica-set resize on failure
Meridian (Nakamura and Diallo, 2023)	Tunable + placement-aware	20–300 ms, workload-adaptive	1.1–1.3× baseline	Joint replacement and re-bound on failure

Table 2: Cross-cluster comparison of six representative systems on five evaluation dimensions. Latency overhead is relative to a no-consistency read-through cache baseline as reported in each system's own evaluation; figures are not directly comparable across papers due to differing testbeds, and are included to show relative order of magnitude only.

(Nimbus, Meridian) reports a *lower* overhead than its non-placement-aware counterpart in the same cluster, supporting the cluster D claim that placement and consistency should not be optimised separately. Third, no system in the corpus reports an energy or backhaul-bandwidth cost column, which Section 6 identifies as a synthesis gap.

Key Takeaway

Placement-aware variants of both strong (Nimbus) and tunable (Meridian) protocols report lower latency overhead than their placement-agnostic counterparts, which is the strongest quantitative evidence in the corpus that consistency and placement should be co-designed.

6 Discussion: Open Gaps

Reading the four clusters together, rather than as separate literatures, surfaces three gaps that no single cluster's papers address on their own.

6.1 Gap 1: No shared cross-model benchmark

Every latency-overhead figure in Table 2 is measured against a different baseline testbed, so the numbers indicate order of magnitude only, not a ranking. Cluster-internal comparisons (e.g. Apex against earlier strong-consistency work) are rigorous; cross-cluster comparisons (e.g. is Synapse's 1.05× overhead *better* than Meridian's 1.1–1.3×, given they provide different guarantees) cannot be made from published numbers alone. A shared benchmark harness — fixed workload traces, fixed topology generator, fixed failure-injection schedule — would let the

field compare protocols on the same axis for the first time.

6.2 Gap 2: Adversarial partitions are under-studied

Fault-tolerance evaluations across the corpus almost universally model partitions as random link or node failures. Only two of the 42 studies model an adversarial partition (one that specifically isolates the replica set serving a high-value tenant). Given that edge deployments increasingly serve mixed-trust tenants on shared infrastructure, this is a narrower threat model than the deployment context warrants.

6.3 Gap 3: No energy accounting in tunable scheduling

Cluster C and D papers optimise a staleness bound against latency or bandwidth, but none report the energy cost of the refresh traffic their schedulers generate. At edge nodes running on constrained power budgets (solar micro-sites, battery-backed base stations), a staleness bound tuned purely for bandwidth could still be energy-infeasible. This is a straightforward instrumentation gap rather than a fundamental one: the scheduling frameworks in Osei and Lindqvist (2019) and Nakamura and Diallo (2023) already expose the hooks needed to add an energy term to their cost functions.

Key Takeaway

The three gaps share a root cause: the field has optimised each protocol against a bespoke evaluation setup, so improvements are provably real within a cluster but not provably comparable across clusters. A shared benchmark would close all three gaps at once by forcing a common set of reported metrics.

7 Conclusion

Organising the edge-caching consistency literature by mechanism rather than by application layer reveals a coherent four-cluster structure and a clear trajectory: from choosing one consistency model at design time, toward making the model adaptive and placement-aware at run time. The comparison in Table 2 shows measurable, decade-long progress in reducing the cost of strong consistency, and the strongest single piece of evidence in the corpus is that placement-aware variants consistently outperform their placement-agnostic counterparts within the same consistency class.

For researchers entering the field, the practical implication of Section 6 is that the highest-leverage open contribution is not a new point on the consistency spectrum but a shared benchmark that makes existing points comparable. For practitioners selecting a protocol for a new deployment, Table 2 and the cluster descriptions in Section 3 provide a starting shortlist keyed to workload write-intensity and partition tolerance requirements, rather than a single recommended default.

References

Marisol Alvarez and Kenji Takahashi. Nimbus: Joint replica placement and consistency bounding for write-heavy edge workloads. In *Proceedings of the International Symposium on Distributed Systems (ISDS)*, pages 88–101, 2021.

Amara Chen, Felipe Ruiz, and Grace Okonkwo. Nexa: Bounding invalidation latency for lease-based cache consistency under wireless jitter. In *Proceedings of the International Symposium on Distributed Systems (ISDS)*, pages 112–125, 2015.

Soo-Jin Kim and Anton Meier. Apex: Regional sequencer consensus with partition-tolerant failover. In *Proceedings of the Conference on Networked Systems and Applications (NetSys)*, pages 455–468, 2022.

Haruto Nakamura and Aminata Diallo. Meridian: A joint placement-and-staleness optimiser for metro-scale edge caching. *Journal of Networked Computing*, 18(1):40–61, 2023.

Daniel Osei and Elin Lindqvist. Vertex: Tunable staleness bounds as a first-class scheduling input for edge caches. *Journal of Networked Computing*, 14(3):330–349, 2019.

Reema Patel and Tomas Novak. Synapse: Operation-based crdt merge for bandwidth-asymmetric edge caches. In *Proceedings of the Conference on Networked Systems and Applications (NetSys)*, pages 201–214, 2017.