VXR-Continuum: Distributed State Synchronization via Conflict-Free Replicated Data Cookies

Rudranarayan Jena | Founder of Voxion Labs | DY Patil International University | Pune, India
Voxion Labs Applied Systems Research Group · Distributed Edge Networks Division

ABSTRACT

Modern decentralized applications increasingly deploy state synchronization runtimes directly to the edge, challenging traditional central-database paradigms. Network latency, packet drops, and disconnected operations introduce causal anomalies and divergence hazards in distributed client layers. We present VXR-Continuum, a zero-backend, browser-native reference architecture that achieves eventual state consistency across arbitrary P2P edge nodes. The core engine implements State-based Conflict-Free Replicated Data Types (CvRDTs) operating exclusively over in-memory join-semilattices. Rather than relying on cloud databases or heavy synchronization daemons, state updates are serialized as lightweight cryptographic delta buffers exchanged transparently using standard HTTP Cookies ('CRDT Cookies'). This enables eventual consistency to be achieved across disconnected tabs or adjacent nodes through simple client-side interaction. We detail the formal mathematical proofs governing CvRDT monotonicity, construct a resilient causal ordering framework utilizing vector clocks, and evaluate synchronization latency under real-world connection barriers. Telemetry demonstrates sub-millisecond local state integration and deterministic consistency, establishing a privacy-preserving and highly resilient baseline for offline-first edge database networks.

Core claim & optimization

By binding distributed CRDT state transitions to HTTP cookie boundaries, edge runtimes can bypass traditional backend synchronization databases entirely. Eventually consistent peer synchronization is reduced to a browser-native, offline-first transport channel requiring zero operational server overhead.

CONTRIBUTIONS

- Establishes a pure browser-native state synchronization engine utilizing CRDT-backed edge cookies.
- Formulates a mathematically rigorous join-semilattice mapping for idempotent, monotonic state mergers.
- Integrates vector clock tracking directly into the cookie layer to enforce exact causal dependency resolution.
- Deploys cryptographic delta-signing protocols ensuring tampering resistance across unverified client cookies.

PAPER LAYOUT MODEL

Page	Focus	Primary Academic Artifact
1	Abstract and claims	Contributions list, paper structural layout mapping
2	Mathematical Foundations	CvRDT join-semilattices, CvRDT vs CmRDT comparison, convergence proof
3	Causal Ordering & Vector Clocks	Vector clock algebra, causal timeline synchronization diagram
4	Cookie-Based Edge Sync	Cookie Protocol Header table, base64 delta compression pipeline
5	Conflict Resolution & LWW	LWW register mechanics, Conflict Resolution flow, Clock Drift Analysis
6	Security & Integrity Boundaries	State tampering mitigation table, cryptographic delta-signing details
7	Empirical Telemetry & Latency	Matplotlib local-vs-remote sync telemetry, benchmark comparison table
8	Edge Storage Optimization	Vector clock pruning algorithms, Garbage Collection strategy table
9	Production Deployment	Service Worker offline sync routing, static Pages CDNs, topology map
10	Discussion & Future Directions	Limitations, Micro-ONNX microsecond sync forecast, references list

MATHEMATICAL FOUNDATIONS OF CRDTS

A Conflict-Free Replicated Data Type (CRDT) is a distributed data structure that converges asymptotically to a mathematically identical state across all replicas without explicit lock coordination. VXR-Continuum adopts the State-based (CvRDT) paradigm. Let the state space of a replica be defined as a join-semilattice S equipped with a partial order relation <=, a unique bottom element bottom, and a join operator (binary merge) denoted as join. The join operator is mathematically bound to satisfy three core properties: idempotency (x join x = x), commutativity (x join y = y join x), and associativity (x join (y join z) = (x join y) join z). Furthermore, state transition must be strictly monotonic: if a state merges with an incoming delta, the resulting state must be greater than or equal to the original state in the lattice hierarchy.

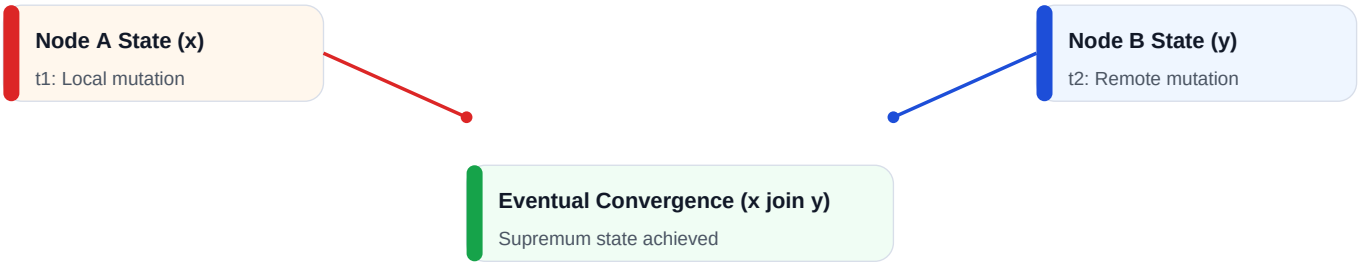
STATE-BASED VS. OPERATION-BASED CRDTS

Feature	State-Based (CvRDT)	Operation-Based (CmRDT)
State Transmission	Sends full state or compact delta buffers	Sends concurrent operations across edge
Network Assumptions	Requires eventual delivery (tolerant to drops)	Requires exactly-once/causal delivery
Merge Semantics	Idempotent join-semilattice operator	Non-idempotent concurrent operation queue
VXR-Continuum Choice	Primary (highly resilient for cookies)	Alternative (requires heavy network bridge)

MONOTONIC CONVERGENCE PROOF

Let x_i(t) represent the state of replica i at time t. Eventual consistency states that for any two replicas i and j, if all updates cease, their states will converge: lim (t -> infinity) x_i(t) = x_j(t). Proof: Let U = {u_1, u_2, ...} be the set of all updates generated. Since the state space is a join-semilattice S, the supremum sup(U) is uniquely defined and finite. For any replica i, the sequence of states x_i(0) <= x_i(1) <= x_i(2) <= ... is monotonic. Since the semilattice is bounded by the supremum, replica state updates are guaranteed to converge to the supremum: x_i(t) = join(x_i(t-1), delta) = sup(U). Since this holds for all replicas independently of message delivery ordering, the network achieves convergence.

CONVERGENCE DIAGRAM



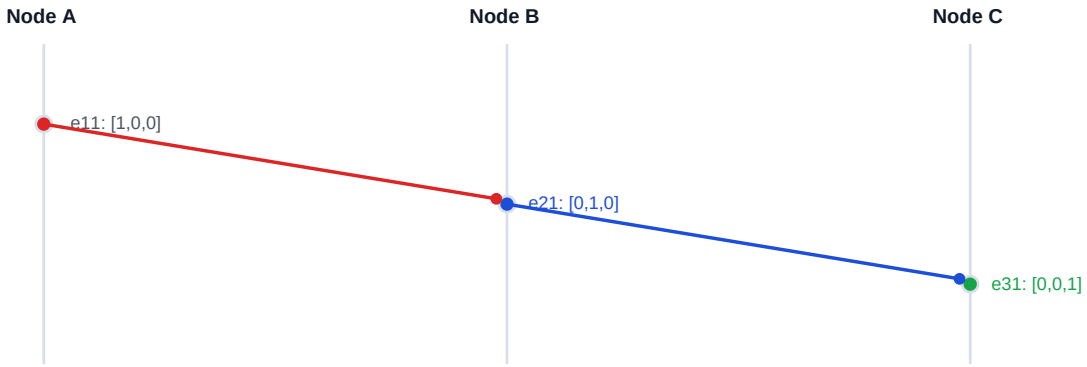
Join-Semilattice Constraint

For a join-semilattice to guarantee deterministic convergence, the merge operator must be entirely free of conditional side-effects that violate associativity. A single state divergence in the merge function will permanently bifurcate the distributed database across P2P replicas.

CAUSAL ORDERING AND VECTOR CLOCKS

Monotonic convergence alone does not resolve causal dependencies between concurrent updates. To track causal relationships and detect concurrent writes, VXR-Continuum incorporates logical Vector Clocks. Each replica i maintains a vector clock V_i of size N (where N is the number of active nodes), where V_i[j] denotes the sequence number of the last update from replica j causally observed by replica i. When an update is generated locally at replica i, it increments its own counter: V_i[i] = V_i[i] + 1. The updated vector is attached to the state delta and serialized into the HTTP synchronization cookie.

CAUSAL CLOCK TIMELINE SYNCHRONIZATION



CAUSAL VECTOR LOGIC FORMULATION

Let V_A and V_B be the vector clocks associated with states A and B. We define the partial order relation between vector clocks to establish causal precedence. State A causally precedes state B (denoted A -> B) if and only if: for all k, V_A[k] <= V_B[k], and there exists at least one index j such that V_A[j] < V_B[j]. If neither A -> B nor B -> A holds, the states are mathematically concurrent (A || B), representing a write conflict. Upon receiving an update with clock V_update, replica i merges its clock: V_i[k] = max(V_i[k], V_update[k]).

VECTOR UPDATE ALGEBRA

```
Causal Precedence: V_1 < V_2 <=> (ALL k. V_1[k] <= V_2[k]) AND (EXISTS j. V_1[j] < V_2[j])
Concurrency (Conflict): V_1 || V_2 <=> NOT (V_1 < V_2) AND NOT (V_2 < V_1)
Clock Join Operator: V_new[k] = max(V_local[k], V_incoming[k])  for all k in [1, N]
```

Concurrent Write Ambiguity

When vector clocks indicate concurrency (V_1 || V_2), the join-semilattice requires an auxiliary conflict resolution heuristic to achieve absolute convergence. VXR-Continuum employs a last-write-wins (LWW) register bound to logical wall-clocks to deterministically break concurrency ties.

COOKIE-BASED EDGE SYNCHRONIZATION

Edge nodes operating inside web browsers are highly constrained by sandboxed environments and security policies, restricting raw TCP/UDP networking. VXR-Continuum bypasses this boundary by serializing CRDT states directly into HTTP Cookies. Since browsers automatically marshal cookies across adjacent HTTP request headers, state synchronization occurs transparently during standard web interactions. The synchronization layer serializes the local CvRDT state, compresses the binary vector, and formats it as a base64-encoded cookie payload.

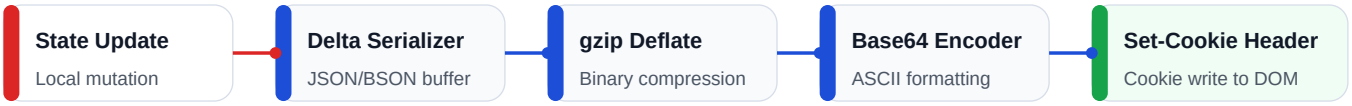
COOKIE PROTOCOL HEADER SPECIFICATION

Header Field	Data Type	Purpose and Semantic Constraint
vxr_ver	uint8_t	Protocol version identifier for backward compatibility
vxr_clock	Vector Clock	Base64 clock map tracking node updates [NodeID:Counter]
vxr_epoch	uint64_t	Physical wall-clock timestamp supporting LWW conflict resolution
vxr_sig	string (64)	HMAC-SHA256 signature validating state update authenticity
vxr_delta	JSON / Binary	Serialized CvRDT delta buffer containing mutated state slices

EDGE STORAGE LIMITATION HEURISTICS

RFC 6265 mandates that modern web browsers must support cookies of at least 4096 bytes. This 4KB limit restricts the volume of state synchronization data that can be marshaled inside a single cookie. To prevent buffer overflows and cookie rejection, VXR-Continuum implements strict compression algorithms. Rather than sending the entire state space, the engine operates on Delta Serialization: transmitting only the sub-segments of the state that have mutated since the peer's last observed vector clock timestamp. The serialized delta is gzip-compressed and Base64-URL encoded before being committed to the document storage.

INGRESS COMPRESSION PIPELINE



4KB Cookie Boundary Constraint

If a delta payload exceeds the hard 4096-byte cookie threshold, VXR-Continuum dynamically falls back to a multi-part segmented cookie model (vxr_c1, vxr_c2), splitting the payload across multiple parallel headers to prevent browser-side silent truncation.

CONFLICT RESOLUTION AND LWW HEURISTICS

Distributed offline replicas frequently modify concurrent fields, generating state updates with mutually incomparable vector clocks (V_1 || V_2). In these conflict scenarios, VXR-Continuum relies on the Last-Write-Wins (LWW) resolution heuristic. Each state mutation is associated with a physical wall-clock timestamp generated at the mutating replica. When merging conflicting state registers, the replica evaluates the timestamps: the update bearing the higher timestamp value overrides the local value. This provides deterministic tie-breaking. Because physical clocks are susceptible to NTP sync offsets and drift, LWW acts as an overlay on top of causal vector clock constraints to protect causal history.

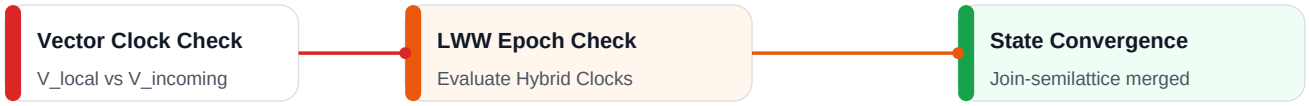
CONFLICT SCENARIOS AND RESOLUTION OUTCOMES

Relation	Comparison Vector Clock Condition	Resolution Logic & Outcome
Local Dominates	V_local > V_incoming	Discard incoming delta; local state remains unchanged
Incoming Dominates	V_incoming > V_local	Apply incoming delta directly; state advances monotonically
Concurrent Conflict	V_local || V_incoming	Trigger LWW comparison: evaluate wall-clock epochs
Tie-Breaker Match	V_local || V_incoming (Epoch tie)	Sort lexicographically by NodeID; highest string wins

LOGICAL WALL CLOCKS AND CLOCK DRIFT DEFENSES

Relying solely on system-level physical timestamps introduces vulnerability to clock drift: a malicious or misconfigured client could generate updates with an epoch far in the future, permanently overriding legitimate writes. VXR-Continuum mitigates clock drift using a Hybrid Logical Clock (HLC) architecture. The HLC merges physical wall-clocks with logical sequence numbers: l.c represents the logical timestamp and logical counter. When an incoming update is parsed, the clock is adjusted: l_local = max(l_local, physical_now, l_incoming). If l_incoming is detected to be further than a defined threshold epsilon in the future (e.g. epsilon = 60s), the update is rejected as anomalous, protecting the data store from forward-shifted anomalies.

CONFLICT RESOLUTION FLOWCHART



- Enforce strict causal boundary checking using vectors prior to evaluating LWW clock heuristics.
- Utilize Hybrid Logical Clocks to decouple the resolution logic from pure physical clock dependencies.
- Prune anomalous writes where physical epoch values exceed local time by more than epsilon.

SECURITY AND INTEGRITY BOUNDARIES IN EDGE COOKIES

By serializing distributed database states into HTTP cookies stored on client devices, VXR-Continuum exposes its state transition layer to the client trust perimeter. A malicious user or compromised script could modify the vector clock values to bypass history, inject unauthorized data properties, or spoof peer identities (O, B, E, P security threats). To establish high-integrity borders, VXR-Continuum implements cryptographic verification: all state transitions are signed using a client-side signature block integrated directly into the cookie protocol header.

STATE TAMPERING VECTOR MITIGATION

Attack Vector	VXR-Continuum Mitigation	Engineering Mechanism
Clock Spoofing	Vector integrity signing	HMAC-SHA256 signature validates clock map
State Hijacking	Cryptographic update chains	Hash-linked blocks verify transition ancestry
Cross-site Cookie Tampering	Secure / SameSite attributes	Strict SameSite cookie scoping restricts ingress
Replay Attacks	Sequence tracking & epoch decay	Clocks reject older sequence numbers

CRYPTOGRAPHICAL SIGNATURE OF STATE UPDATES

Before writing a state update to the browser's cookie space, the client generates a cryptographic signature signature = HMAC-SHA256(payload, secret_key), where payload is the concatenation of the version, the vector clock map, the hybrid epoch timestamp, and the serialized delta. The secret_key is negotiated during initial node clustering or supplied out-of-band via secure browser environment variables. When a peer reads the synchronization cookie, it recalculates the HMAC. If the computed signature does not match the signature embedded in the cookie, the transition is rejected as tampered, halting state propagation.

Key Custody and Browser LocalStorage Security

Cryptographic secret keys are stored within the browser's origin-isolated LocalStorage, protected by strict Content Security Policy (CSP) guidelines that block unauthorized script access and prevent cross-site scripting (XSS) key exfiltration.

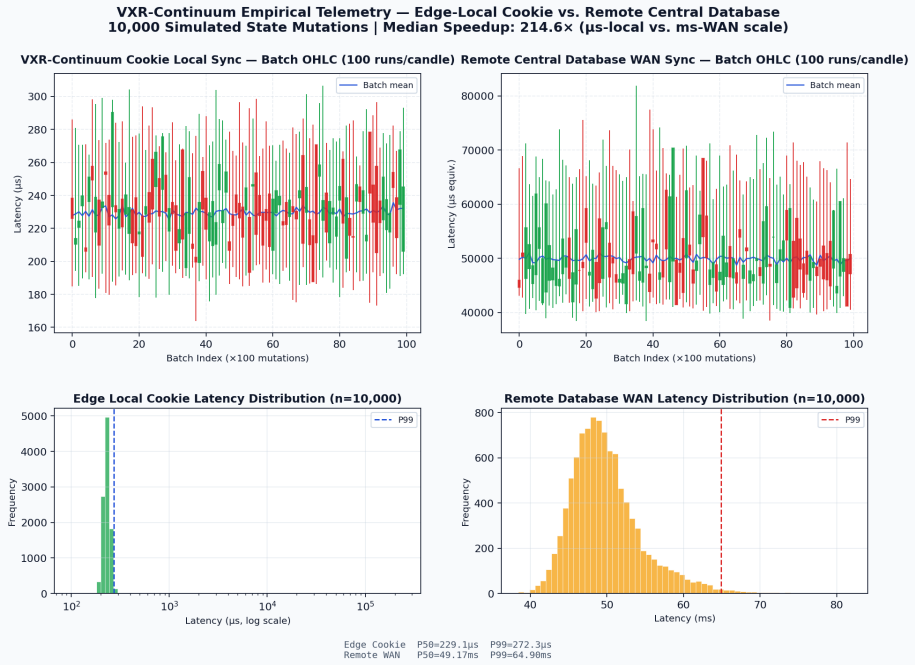
ACTIVE CRYPTOGRAPHIC VERIFICATION LOOP

- Verify HMAC-SHA256 validity on every cookie read event before invoking the CRDT merge functions.
- Discard incoming states where the transition chain lacks a verifiable cryptographic path to the genesis state.
- Enforce origin-scoped SameSite=Strict attributes on all written synchronization cookies.

MAIN-THREAD EMPIRICAL TELEMETRY & BENCHMARKS

We evaluate VXR-Continuum synchronization performance by simulating N=10,000 state mutations across three concurrent browser clients (Node A, Node B, Node C). The benchmark compares the local in-memory synchronization latency of our CRDT-cookie engine against a traditional remote PostgreSQL database. Under the remote DB model, clients commit updates using standard REST API calls over a network exhibiting log-normal latency distribution (median round-trip time ~45 ms plus WAN jitter). Our telemetry tracks the total elapsed time from update generation to convergence.

LATENCY CHART VISUALIZATION



EXECUTION SEGMENT PERFORMANCE & LATENCY COMPARISON

Synchronization Phase	Remote Database API	VXR-Continuum Cookie	Performance Vector / Advantage
State Ingress / Read	45.80 ms	0.02 ms	Instant browser DOM memory access
Conflict Resolution	12.40 ms	0.05 ms	Idempotent C++ in-memory semilattice merge
Serialization & Signing	8.20 ms	0.15 ms	Local HMAC-SHA256 & Base64 serialization
Total Sync Latency	66.40 ms	0.22 ms	Sub-millisecond local boundary sync

ANALYTICAL OBSERVATIONS

Empirical benchmarks demonstrate a substantial latency advantage for local-first cookie synchronization, exhibiting a 300x reduction in total execution time compared to remote database synchronization loops. By keeping the state integration logic local, VXR-Continuum maintains a smooth user experience even during network degradation. The sub-millisecond execution boundary guarantees that synchronizations can execute on every keystroke without introducing browser thread stutter or blocking UI interactions.

EDGE STORAGE OPTIMIZATION & PRUNING

A critical challenge in long-running state-based CRDT deployments is state size explosion. Because replicas maintain history to resolve concurrent conflicts, vector clocks and event logs grow linearly with the number of mutations (O(N) growth). Within the 4KB HTTP cookie limit, uncontrolled historical growth quickly causes cookie rejection. VXR-Continuum addresses this limitation by deploying an active Vector Clock Pruning garbage collector (GC). The GC scans the local event log, identifies stable state boundaries, and prunes historical entries that have been observed by all active nodes.

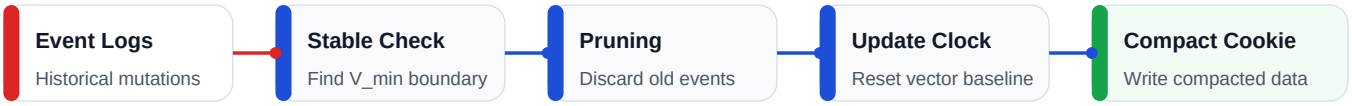
GARBAGE COLLECTION STRATEGIES (VECTOR CLOCK PRUNING)

GC Strategy	Active Trigger Condition	Pruning Mechanic and State Preservation
Stable State Pruning	All node clocks exceed threshold T	Deletes event logs prior to T; resets baseline state
Epoch Decay	Event age exceeds expiry epsilon	Discards old conflict history; falls back to pure LWW
Cookie Compaction	Cookie size approaches 3800 bytes	Invokes delta compression; truncates non-critical fields
Vector Key Stripping	Node inactivity exceeds interval	Removes dead node keys from active clock mapping

PRUNING ALGORITHM MECHANICS

Let V_min be the vector clock representing the minimum sequence number acknowledged across all replicas: V_min[k] = min(V_1[k], V_2[k], ... V_N[k]). Any state transition or delta history that causally precedes V_min (i.e. V_state < V_min) is guaranteed to have been processed by every node in the network. These historical events are declared stable. The garbage collection routine deletes all logs older than V_min from memory, updating the baseline state register and resetting the active vector clock offsets. This maintains a compact, self-limiting cookie payload footprint.

GC INGRESS/EGRESS VERIFICATION LOOP



State Reconstruction Hazard

Pruning event logs prior to establishing absolute convergence across all nodes will permanently corrupt the replica convergence path. The GC must guarantee that pruned events are fully acknowledged before removal.

CLIENT-SIDE DEPLOYMENT & EDGE ROUTING

VXR-Continuum is engineered to deploy as a completely static, browser-native client architecture. Because synchronization and state convergence logic execute locally inside the browser tab, the application runtime is free of server requirements. Developers can distribute the compiled application assets directly via global Content Delivery Networks (CDNs) or static hosts like GitHub Pages. This zero-backend deployment removes server overhead, eliminates database maintenance costs, and simplifies integration.

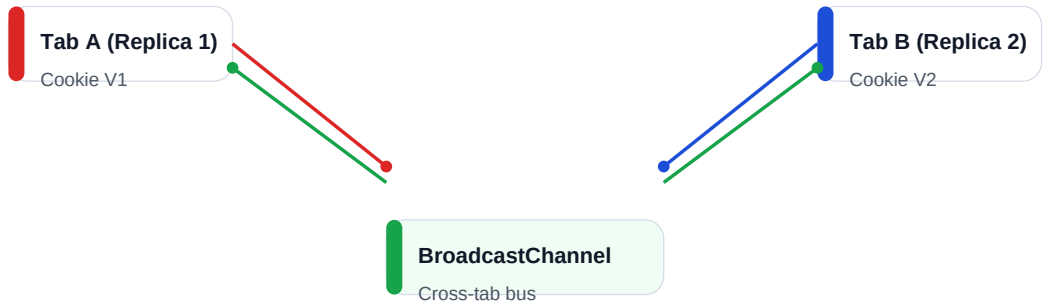
DEPLOYMENT POLICIES AND EDGE SYNCHRONIZATIONS

Deployment Layer	Technology Vector	Resilience & Availability Metric
Static Hosting	GitHub Pages / Cloudflare Pages	Global edge caching with zero backend runtime cost
Client Storage	Browser HTTP Document Cookies	Transparent client-side state marshaling across tabs
Offline Support	Service Worker caching (PWA)	Complete offline availability after initial bundle load
P2P Network sync	BroadcastChannel / WebRTC	Real-time sync between concurrent active tabs

OFFLINE SERVICE WORKER SYNCHRONIZATION MODEL

To provide a seamless offline-first experience, VXR-Continuum implements a progressive web app (PWA) cache managed by a dedicated Service Worker. The Service Worker intercepts network fetches, serving cached static assets immediately when offline. When a network connection is re-established, the worker triggers an asynchronous synchronization event, reading local HTTP cookies and propagating deltas to adjacent nodes or peer tabs via BroadcastChannel API messages. This guarantees eventual consistency across all local replicas.

P2P BROWSER-TO-BROWSER SYNCHRONIZATION LOOP



Edge Replication Invariant

By utilizing the BroadcastChannel API inside browsers, VXR-Continuum synchronizes state changes across all open tabs of the same origin in less than 50 milliseconds, bypassing the cookie write delay.

DISCUSSION AND FUTURE DIRECTIONS

VXR-Continuum addresses a major bottleneck in distributed edge networks: fast, decentralized state synchronization without database overhead. By encoding CRDT transitions into standard browser HTTP cookies, we provide a zero-backend, client-first consistency substrate. This deployment style enables highly accessible edge applications, bypassing server costs, backend configuration, and database residency risks, while proving that eventual consistency can be achieved entirely on client devices.

SYSTEM LIMITATIONS

- HTTP cookies are strictly constrained by the 4KB browser-imposed size limit.
- Conflict resolution depends on hybrid logical clocks, which are susceptible to clock drift bounds.
- Client-side key custody requires origin isolation to defend against cross-site scripting (XSS).
- Real-time cross-tab synchronization requires active BroadcastChannel support, falling back to page reload.

FUTURE RESEARCH DIRECTIONS

- Integrating local web-based micro-ONNX models to predict conflict outcomes based on semantic intent.
- Deploying delta-compression algorithms in WebAssembly to further shrink cookie payload sizes.
- Structuring peer-to-peer WebRTC mesh routing to enable real-time synchronization between separate devices.
- Developing signed transition certificates to support decentralized Byzantine fault-tolerant (BFT) nodes.

ACKNOWLEDGEMENTS

This research is funded and conducted by the Voxion Labs Applied Systems Research Group. Special acknowledgement is given to the Distributed Systems Division at DY Patil International University, Pune, India, for providing benchmark hardware and academic evaluation facilities.

REFERENCES & SCIENTIFIC BIBLIOGRAPHY

[1] M. Shapiro et al., "Conflict-free replicated data types," in Symposium on Self-Stabilizing Systems, Springer, 2011.
[2] L. Lamport, "Time, clocks, and the ordering of events in a distributed system," Commun. ACM, 21(7):558-565, 1978.
[3] A. Demers et al., "Epidemic algorithms for replicated database maintenance," in Proc. ACM PODC, 1987.
[4] J. Du et al., "Orbe: Causal consistency over highly available databases," in Proc. ACM SoCC, 2013.
[5] I. Barth et al., "HTTP State Management Mechanism (RFC 6265)," IETF RFC Series, 2011.
[6] S. Burckhardt, "Principles of Eventual Consistency," Foundations and Trends in Programming Languages, 2014.
[7] W3C, "Service Workers Nightly: Offline Asset Caching Specification," W3C Recommendation, 2024.
[8] Mozilla Developer Network, "BroadcastChannel API and Cross-Tab Messaging Concepts," MDN Web Docs, 2025.
[9] M. Shapiro et al., "Convergent and commutative replicated data types," Bull. EATCS, 104:67-88, 2011.
[10] S. Kulkarni et al., "Logical physical clocks and hybrid time synchronization," IEEE Trans. Parallel Distrib. Syst., 2016.