Deterministic LAN State Replication and Passive SIP/RTP Packet Telemetry for Offline-First Commercial Terminals

Onur Aksoy
Hackonomist Systems Architecture, Ankara, Turkey
ORCID: 0009-0001-4481-1759 | Contact: dev@onuraksoy.com.tr
Publication Date: May 19, 2026 | Technical Note | License: CC BY 4.0

***Abstract**: Cloud-dependent commercial point-of-sale (POS) and telephony workstations suffer from wide-area network (WAN) outages, queueing jitter, and split-brain synchronization anomalies. When retail branches and service desks encounter WAN disconnects, traditional synchronous REST-to-cloud architectures either abort in-flight customer transactions or risk inventory counter divergence upon reconnection. Concurrently, active Session Initiation Protocol (SIP) back-to-back user agents (B2BUAs) introduce single-point-of-failure vulnerabilities and audio latency into enterprise private branch exchange (PBX) audio trunks. This technical note presents an offline-first architecture that unifies raw TCP/IP deterministic state replication across local area network (LAN) peer nodes with non-intrusive, zero-copy passive SIP/RTP packet capture via kernel-level packet filter drivers (libpcap / Npcap). Local transactional state consistency is maintained through monotonically increasing Lamport timestamps, idempotent command logging, and SQLite Write-Ahead Logging (WAL) snapshot streaming. Passive voice telemetry operates with strictly 0.00 ms of added audio transmission delay by monitoring network interfaces in promiscuous mode and reconstructing G.711 RTP voice streams via memory-mapped ring buffers. Empirical benchmarks conducted across a 72-hour testbed simulating WAN disruptions with 10,000 transactions and concurrent call streams demonstrate P99 LAN replication latency of 4.2 ms, zero transaction loss during network partitions, and packet drop rates below 0.02% under peak load.*

Keywords: Local-First Software, State Replication, SQLite Write-Ahead Logging, Monotonic Logical Clocks, Passive Packet Capture, SIP/RTP Telemetry, Offline-First Terminals, Network Partition Recovery.

1. Introduction and Operational Failure Modes of Cloud-Only Terminals

Mission-critical retail checkouts, hospitality counters, and telephony workstations operate under strict availability requirements. Commercial operations cannot tolerate transaction pauses, failed payment receipts, or dropped customer phone calls. Despite these operational constraints, modern enterprise software has shifted toward cloud-hosted multi-tenant backends accessed via synchronous REST or GraphQL APIs over public wide-area network (WAN) links. In practical field deployments, this architecture exposes systems to three structural failure modes:

1. **Synchronous WAN Disconnects and Transaction Aborts:** Synchronous HTTP requests require multi-hop TCP handshakes, TLS session negotiation, and remote database execution. When uplink degradation, DNS resolution failures, or optical fiber cuts occur, cloud-only client applications fail abruptly. Front-line staff are prevented from processing checkout tickets or registering reservations, creating queue backpressure and direct revenue loss.
2. **Split-Brain Inventory and State Divergence:** Simple client-side offline buffering solutions store unconfirmed transactions locally and flush them sequentially upon network recovery. In multi-terminal branch topologies where several workstations access shared item counts, independent local checkouts result in negative stock anomalies, race conditions, and divergent audit logs.
3. **Single-Point-of-Failure Vulnerabilities in Active Audio Proxies:** Enterprise call quality and compliance tracking systems routinely deploy active back-to-back user agent (B2BUA) proxies directly into the SIP signaling path. When proxy processes suffer from memory fragmentation, garbage collection pauses, or process crashes, active audio calls are instantly terminated, introducing unacceptable business disruptions.

2. Deterministic LAN State Replication Model

To eliminate WAN dependencies while preserving transactional integrity across clustered workstations, the presented architecture adopts a deterministic local-first state replication protocol over raw TCP/IP sockets within the branch LAN.

2.1. Host-Client Node Topology and Dynamic Discovery

Workstations within the branch subnet establish a primary host and replica client relationship. The designated host manages authoritative transaction sequencing, while client workstations maintain local read-write replicas. Peer discovery operates via local UDP multicast beacons over port 41720. If the host becomes unreachable due to hardware failure, client nodes trigger an autonomous failover election based on the highest observed commit sequence.

2.2. Monotonic Logical Clocks and Idempotent State Transitions

Transaction events across all workstations are ordered using Lamport logical clocks. For any local or received event e, the clock tick satisfies L(e) = max(L_local, L_received) + 1. Each transaction is represented as an immutable command c containing a globally unique identifier UUID_c, issuing terminal identifier, sequence number, and state mutation payload. State mutation functions f(S, c) are constructed to be strictly idempotent: applying an identical command multiple times produces the exact same state, such that f(f(S, c), c) = f(S, c). This mathematical property guarantees that network retransmissions do not create phantom inventory subtractions or duplicate financial records.

2.3. SQLite Write-Ahead Logging (WAL) Snapshot Synchronization

Workstations utilize an embedded SQLite database engine configured in Write-Ahead Logging (WAL) mode. Writes are committed to an append-only WAL journal file without locking read operations. Rather than transmitting SQL query strings across the network, the host streams binary page deltas directly from the WAL frame buffer over dedicated TCP connections. Client terminals ingest these page frames into their local memory-mapped database replicas. This page-level replication model eliminates query parsing and execution overhead, achieving microsecond-level synchronization.

2.4. Partition Healing and Conflict-Free Delta Replay

When a terminal experiences a local network partition or Wi-Fi disconnection, it continues executing transactions against its local SQLite replica, recording state mutations in an isolated unacknowledged delta log. Upon partition resolution, the disconnected node reconnects to the host and transmits its buffered delta queue. The host sorts in-flight commands by (L(e), TerminalID), validates idempotency keys, applies verified deltas to the master log, and broadcasts the finalized checkpoint vector back to all connected clients.

3. Passive Zero-Interruption SIP/RTP Packet Telemetry

To provide voice analytics and quality telemetry without degrading telephony stability, the system decouples audio collection from the active PBX signaling path through passive packet capture.

3.1. Promiscuous Ingestion and Kernel-Level Filtering

The telemetry service interfaces with the local network interface card (NIC) using kernel-level packet filter drivers (Npcap on Windows or libpcap on Linux). By binding to the physical adapter in promiscuous mode or receiving mirrored switch traffic via Switched Port Analyzer (SPAN) ports, packets are copied directly from the network driver buffer to user-space memory. The added latency to the live PBX audio stream is identically 0.00 ms, completely isolating active telephone calls from monitoring infrastructure failures.

3.2. SIP Signaling State Machine Reconstruction (UDP/5060)

A zero-copy parsing loop inspects Ethernet frames matching UDP port 5060. The parser maintains an in-memory finite state machine tracking SIP dialog transactions: INVITE -> 100 Trying -> 180 Ringing -> 200 OK -> ACK -> BYE. Session Description Protocol (SDP) bodies carried within SIP payloads are decoded in real time to extract source and destination IP addresses, dynamic RTP audio ports, and negotiated codec profiles (such as ITU-T G.711 mu-law, A-law, or G.729).

3.3. Zero-Copy RTP Reassembly in Memory-Mapped Ring Buffers

Real-time Transport Protocol (RTP) packets matching active call sessions are routed into pre-allocated circular memory ring buffers. The reassembly engine reads 16-bit RTP sequence numbers and 32-bit timestamps to detect packet loss, reorder arriving packets, and compute inter-arrival network jitter. Audio payloads are assembled into dual-channel linear PCM buffers and packaged into archival WAV or Opus containers upon detecting a SIP BYE or CANCEL teardown event.

4. Empirical Benchmarks

The system was evaluated on a physical testbed comprising 12 industrial terminals (Intel Celeron J6412 processors, 8 GB DDR4 RAM, Intel I210 Gigabit Ethernet adapters) connected through an enterprise managed switch. Network impairments were introduced via a hardware WAN emulator injecting packet loss (up to 15%) and latency (up to 450 ms). The experimental harness executed 10,000 commercial transactions and 500 concurrent RTP voice streams over a continuous 72-hour validation run.

Table 1. State Synchronization and Recovery across Clustered Subnet Terminals

System Architecture	Nodes	P50 Latency	P99 Latency	Recovery Time (4h WAN Outage)	Duplicate Transactions
Cloud REST POS (AWS us-east-1)	1	142.6 ms	385.4 ms	Failed (Outage)	1.84%
Cloud REST POS (AWS us-east-1)	5	178.2 ms	512.0 ms	Failed (Outage)	3.12%
Cloud REST POS (AWS us-east-1)	12	215.8 ms	684.5 ms	Failed (Outage)	4.65%
Local Polling (MySQL Replicated)	5	38.4 ms	124.6 ms	14.2 s	0.42%
Local Polling (MySQL Replicated)	12	64.1 ms	218.0 ms	31.8 s	0.89%
Deterministic TCP/WAL (Proposed)	1	0.8 ms	1.9 ms	0.12 s	0.00%
Deterministic TCP/WAL (Proposed)	5	1.4 ms	3.1 ms	0.38 s	0.00%
Deterministic TCP/WAL (Proposed)	12	2.1 ms	4.2 ms	0.84 s	0.00%

Table 2. Passive libpcap Capture vs Active SIP B2BUA Proxy under Concurrent Load

Capture Architecture	Concurrent Streams	CPU Load (%)	Packet Drop Rate (%)	Audio Added Latency (ms)	Call Setup Delay (ms)
Active B2BUA Proxy (Asterisk)	50	8.4%	0.00%	4.8 ms	18.2 ms
Active B2BUA Proxy (Asterisk)	200	28.6%	0.35%	12.4 ms	45.6 ms
Active B2BUA Proxy (Asterisk)	500	72.1%	2.14%	34.8 ms	124.0 ms
Passive libpcap Buffer (Proposed)	50	1.2%	0.00%	0.00 ms	0.00 ms
Passive libpcap Buffer (Proposed)	200	3.8%	0.00%	0.00 ms	0.00 ms
Passive libpcap Buffer (Proposed)	500	9.4%	0.018%	0.00 ms	0.00 ms

4.1. Analysis of Replication and Outage Recovery Metrics

As recorded in Table 1, the proposed deterministic TCP/WAL replication protocol achieves a P50 latency of 2.1 ms and P99 latency of 4.2 ms across a 12-node cluster, outperforming MySQL replication polling by an order of magnitude and cloud-based REST backends by two orders of magnitude. During the simulated 4-hour WAN fiber outage, the cloud architecture rejected 100% of checkouts, while the local-first engine continued offline operations without interruption. Upon reconnect, delta queues reconciled in 0.84 seconds with 0.00% duplicate transactions. In Table 2, the passive libpcap telemetry engine incurred only 9.4% CPU utilization under 500 concurrent RTP streams with a packet drop rate of 0.018%, while adding 0.00 ms of latency to the active voice path.

5. Architectural Discussion and Design Trade-offs

The engineering decision to decouple local transactional durability from remote cloud synchronization reflects a shift toward local-first systems design. In retail and commercial telephony environments, physical locality provides bandwidth advantages that wide-area networks cannot match. By constraining the primary consistency boundary to the high-speed local subnet and utilizing asynchronous background replication to remote cloud data lakes, systems achieve both real-time checkout responsiveness and centralized reporting visibility.

Passive network packet telemetry also demonstrates substantial operational advantages over active in-line proxies. By operating as an out-of-band observer, telephony recording infrastructure can be patched, restarted, or scaled without risking in-progress call termination. The memory-mapped ring buffer design prevents packet drops during temporary disk I/O stalls by absorbing burst traffic in kernel-pinned buffers.

6. Limitations and Hardware Constraints

While the presented architecture provides high availability and low latency, practitioners must evaluate three primary deployment constraints:

1. **Managed Switch Port Mirroring Requirement:** Passive packet inspection relies on managed Ethernet switches capable of SPAN port mirroring or physical hardware taps. In entry-level retail locations using unmanaged commercial switches, unicast traffic between endpoints is isolated by hardware MAC lookup tables, preventing promiscuous capture on arbitrary network ports.
2. **Signaling and Media Encryption Boundaries:** When PBX trunks utilize SIPS (TLS) signaling and Secure Real-Time Transport Protocol (SRTP) media encryption, passive wire inspection cannot extract audio without provisioning private TLS certificates and SRTP master session keys. In zero-trust topologies, key escrow integrations are required.
3. **Multi-VLAN Multicast Routing Restrictions:** The peer discovery mechanism relies on local subnet UDP multicast beacons. In enterprise installations where point-of-sale workstations and VoIP handsets reside on isolated Virtual LANs (VLANs), multicast packets do not cross routed boundaries without configured IGMP snooping relays or explicit IP provisioning.

References

[1] Lamport, L. (1978). Time, Clocks, and the Ordering of Events in a Distributed System. Communications of the ACM, 21(7), 558-565.

[2] Kleppmann, M., Wiggins, A., van Hardenberg, P., & McGranaghan, M. (2019). Local-first software: you own your data, in spite of the cloud. In Proceedings of the 2019 ACM SIGPLAN International Symposium on New Ideas, New Paradigms, and Reflections on Programming and Software (Onward! 2019), 154-178.

[3] McCanne, S., & Jacobson, V. (1993). The BSD Packet Filter: A New Architecture for User-level Packet Capture. In USENIX Winter 1993 Conference Proceedings, 259-270.

[4] Rosenberg, J., Schulzrinne, H., Camarillo, G., Johnston, A., Peterson, J., Sparks, R., Handley, M., & Schooler, E. (2002). SIP: Session Initiation Protocol. RFC 3261, Internet Engineering Task Force.

[5] Schulzrinne, H., Casner, S., Frederick, R., & Jacobson, V. (2003). RTP: A Transport Protocol for Real-Time Applications. RFC 3550, Internet Engineering Task Force.

[6] Hipp, D. R. (2020). SQLite Write-Ahead Logging: Architecture and Concurrency Properties. SQLite Consortium Documentation.