International Journal of Computer Information Systems and Industrial Management Applications
ISSN 2150-7988 Volume 18(23s) 2026 pp. 1625-1636
https://doi.org/10.70917/ijcisim-2026-5717
Article

Fault-Tolerant Architecture for Cross-Border Payment Systems Under Regulatory and Latency Constraints

Anath Bandhu Chatterjee

Independent Researcher, India. Email:- anath04jgec@gmail.com

Abstract: Cross-border remittance systems operate across trust and legal boundaries in which no single control plane can commit an entire transaction atomically. One step, the disbursement to a receiver, is irreversible and takes place at a counterparty that cannot participate in a commit protocol. This paper contributes an architecture in which the payment intent is the durable append-only primitive, commit occurs before the irreversible act, and every recovery mechanism follows from that ordering. The paper's protocol contribution is the minting rule: a new external reference for a retry may be created only from a predecessor proven to be in a terminal-and-not-settled state; otherwise the retry must reuse the previous reference so that the partner's own deduplication converts at-least-once transport into at-most-once effect. Compliance is expressed as a state-transition precondition rather than as a middleware layer, so that no path to disbursement bypasses screening. Evidence for the architecture's guarantees is categorical, not statistical: three counts must be zero (duplicate settlements, releases without screening, payments without an intent record). The paper deliberately refuses production telemetry as the primary evidence base because rates cannot demonstrate absolutes. Correctness is established across three testing layers: exhaustive model checking of the specification, deterministic simulation of the implementation, and adversarial fault injection against per-capability-class partner simulators that include a partner that lies about settlement state. The paper argues that "at-most-once effect with eventual determination" is the strongest achievable guarantee at this boundary and should be adopted as the honest replacement for exactly-once framing that classical distributed-transaction literature carries into a setting where its assumptions do not hold

Keywords: Cross-border payments, fault tolerance, idempotency, at-most-once effect, saga, deterministic simulation testing, sanctions screening, DORA, distributed systems, formal verification

1. INTRODUCTION

A cross-border remittance is not a single database transaction. It is a long process that moves across areas of trust and law in which participants share no protocol for finalising data at the same time. As a sender starts a transfer, the system decomposes the action into steps: funding from the sender, setting and locking of an exchange rate, screening for risk and sanctions, reservation of funds from a local prefunded account, submission of a payout request to a disbursement partner, and finally reconciliation against the partner's settlement file. By design no central controller locks a record while it is at a payout partner or a local clearing system. Because of this, partial failure is not a rare error class. It is a regular part of daily operations. The architecture's responsibility is to bound the ambiguity that those failures create.

In the consumer-facing model, a structural risk exists because the two parts of a transfer have different speeds and different rules for reversal. Funding is slow and reversible: a bank debit can return as unauthorised days later, and a card payment can be reversed after weeks. Disbursement is fast and irreversible: on an instant-payment rail the settlement is final in seconds, and on a cash-collection route the money is gone when the receiver presents identification to the agent. The operator therefore extends unsecured credit for every transfer. If funding fails after disbursement is complete, the operator absorbs the full loss. As the operator shortens promised delivery times to

compete in the market, the exposure window widens. On that account the goal of fast delivery and the goal of low financial loss are in direct conflict.

The hardest failures are not clear rejections. It is inexpensive to handle a case in which a partner returns an explicit rejection code. It is expensive when the outcome is not clear. This happens when the system submits a payout request and the connection stops before an answer arrives. In that state the system does not know whether the receiver received the money. To retry is to risk a double payment. To stop is to leave a sender debited without their funds. Idempotency keys help only when a partner honours the key through the whole process, and across a diverse partner set many do not. Because status-code semantics and reversal capabilities differ per partner, a single recovery policy is not available. On cash-collection routes the ambiguity can persist for hours or days, and a simple timeout cannot resolve it.

For the reasons above, the goal of the architecture is not to prevent all failures. The goal is that every intermediate step is durable, traceable to a source, and correctable within a bounded time, that no correction path bypasses a legal rule, and that the irreversible act is never triggered by a status the system cannot later prove was reached correctly.

This paper contributes:

1. An architecture in which the payment intent is the durable, append-only primitive of record, and all recovery follows from that primitive (§III).

2. A commit-first ordering at the disbursement boundary, defended as the only survivable ordering when the counterparty cannot participate in commit (§IV).

3. The minting rule, the paper's protocol contribution: a new external reference is minted only from a predecessor proven terminal-and-not-settled; otherwise a retry must reuse the previous reference, converting at-least-once transport into at-most-once effect without requiring counterparty participation (§IV).

4. Compliance expressed as a state-transition precondition, not a service call: screening decisions have validity ranges, are stored by reference on the intent, and re-evaluate on list updates. No disbursement path bypasses the control (§V).

5. A two-budget latency model, 3 s synchronous and 30 s asynchronous, with durability placed on the irreversible leg and 27 % reserve on the synchronous path for recovery (§VI).

6. A three-layer testing methodology, model checking of the specification plus deterministic simulation of the implementation plus adversarial partner simulators including a partner that lies about settlement state, offered as categorical evidence in place of production telemetry (§VII).

7. Reporting discipline that separates submission-path availability from disbursement-path availability, splits success into four terminal categories, and states three counts that must be zero rather than low (§VIII).

The paper is not an empirical study. Section X states this explicitly and defends the position.

2. Related Work

Sagas and long-lived transactions. Garcia-Molina and Salem [1] introduced Sagas as sequences of local transactions paired with compensating transactions that logically reverse partial executions. The pattern remains the standard reference for long-running distributed transactions [3]. This paper's architecture uses a Saga-shaped decomposition but treats compensation as a recovery claim rather than a logical inverse. A reversal in cross-border payments is a new payment with its own exchange rate, cost, screening obligation, and chance of failing. The Saga literature has not typically distinguished between compensations that can be executed without counterparty participation and compensations that themselves depend on the counterparty. This paper's framing draws that distinction.

Two-phase commit and its absence. The classical two-phase commit protocol [2] requires participants to hold a prepared state on the coordinator's behalf. Disbursement partners in cross-border payments do not run 2PC and will not hold prepared state. On cash-collection routes a human hands over currency. There is no coordinator role a partner will accept. On that account the disbursement boundary is one at which 2PC-based reasoning does not apply and a different discipline is required.

Exactly-once semantics. Modern messaging systems have converged on producer idempotency + transactional writes as the pattern for exactly-once semantics within a single control plane [4, 5]. The pattern relies on all participants sharing a coordinator. Across control planes, and specifically across the operator-partner boundary studied here, that assumption does not hold. This paper argues that "at-most-once effect with eventual determination" is the strongest available guarantee across the disbursement boundary, and that framing this outcome as an honest guarantee rather than a weakened one is part of the architectural argument.

Deterministic simulation testing. FoundationDB [6] and TigerBeetle [7] have demonstrated that deterministic simulation testing produces categorical evidence for distributed-system correctness at a level that ordinary integration testing cannot reach. The pattern controls all sources of non-determinism (clock, network, disk, random-number generator) and replays years of simulated operational time in minutes. Neither system's methodology has been widely adopted in the cross-border payments literature. §VII adopts and extends the pattern with adversarial partner simulators that model per-capability-class behaviour, including partner behaviours that violate declared contracts.

Formal methods for distributed protocols. TLA+ [15] and the AWS experience report [16] establish that model checking of distributed protocol specifications is tractable at industrial scale and produces bugs that behaviour testing cannot. This paper's §VII uses model checking as the first evidence layer for the state machine, the lease-and-fencing discipline, and the minting rule.

Consensus primitives. The intent log commits through a linearizable compare-and-swap primitive traced to Paxos [12] and Raft [13]. Neither innovation is claimed here; the paper uses established consensus as the substrate on which the payment-intent primitive rests.

Cross-border payments infrastructure. SWIFT's ISO 20022 programme and the Bank for International Settlements CPMI harmonisation report [11] provide the message-layer standardisation context. This paper does not extend ISO 20022. It positions ISO 20022 as the message-layer standard the state machine composes with rather than replaces.

Regulatory frameworks. The EU Digital Operational Resilience Act (DORA, Regulation 2022/2554) [8], US SEC Regulation SCI [9], and FINRA Rule 4370 [10] establish the operational-resilience and business-continuity expectations that constrain the design. §V engages these frameworks directly as design inputs, not as compliance overlays.

3. Payment Intent as the Durable Primitive

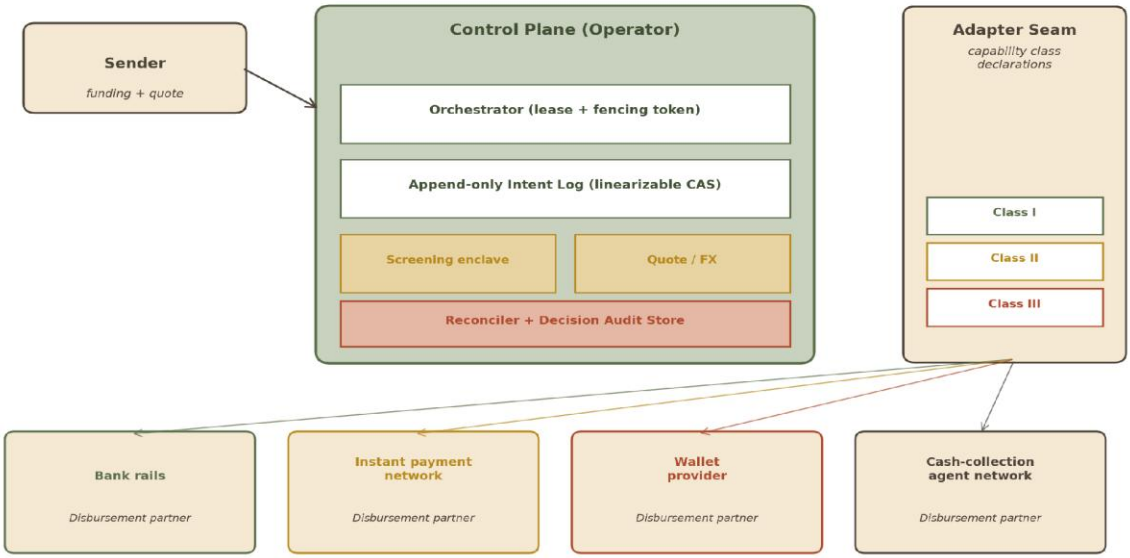
The system treats a remittance as a sequence of events in which exactly one part is permanent and cannot be reversed. Every other mechanism is organised to protect that permanent part. The primary rule is that no irreversible disbursement may begin unless a durable and unique record of the right to pay exists first. The system must be able to demonstrate that it held this right.

The payout intent is the artefact of record. It is written to an append-only log in the same local process as the outbound message but before the system calls any external service. Each intent contains the sender's unique identifier, the locked currency terms, the screening-decision reference, the selected partner, and a status that only advances. State transitions are additions to the log. No component may move an intent to an earlier status. Because the intent is the entity that survives failure, a stopped process or a regional outage does not cause loss of data.

Figure 1 shows the platform architecture. The operator's control plane hosts the orchestrator, the append-only intent log committed through linearizable compare-and-swap, the residency-pinned screening enclaves, and the

reconciler with its decision-audit store. The adapter seam sits between the control plane and the partner ecosystem. Each adapter carries a capability-class declaration that names what its partner guarantees (idempotency-key honouring, read-after-write consistency on partner references, terminal-state declaration semantics). The declaration governs which recovery policy the platform selects for the partner.

Figure 1. Cross-border payment platform architecture: control plane, adapter seam, and disbursement-partner capability classes



Three architectural properties follow from the intent's status as the durable primitive.

Recovery is a read of the log, not a reconstruction of state. Any surviving orchestrator instance can resume an intent by reading the log. No process holds authoritative state in memory. The recovery-time bound is the lease TTL plus the acquisition time, not a human intervention window.

Ownership is enforced by fencing tokens on the intent, not by locks in code. Only one worker may execute an intent at a time. Concurrent workers are refused at the boundary through a stale-token check rather than detected later through conflict.

Audit is a projection of the log, not a separate concern. Every state transition carries a reason code and a policy-version reference. The decision-audit store is a materialisation of the log for compliance reviewers; it is not a parallel record whose consistency must be maintained.

4. Commit-First Ordering and the Minting Rule

4.A The commit-first ordering

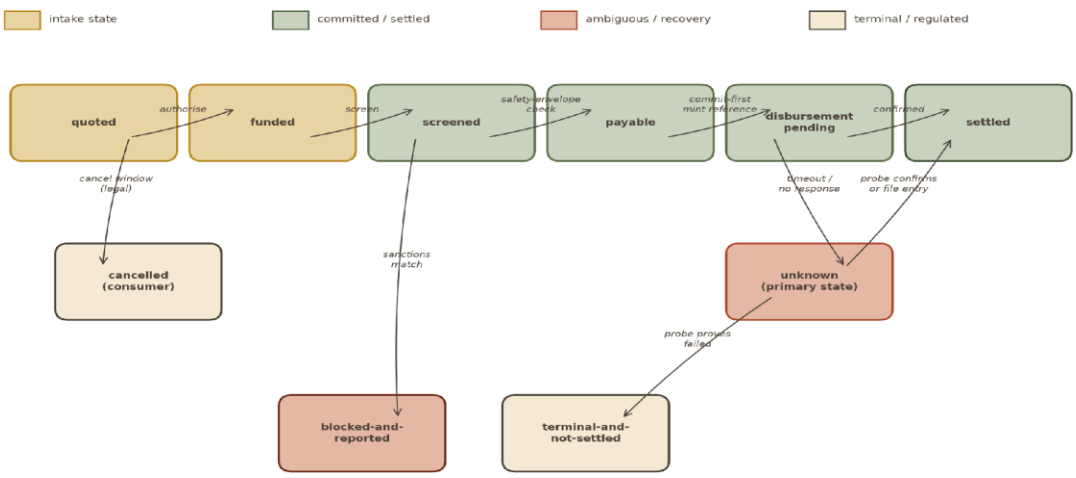
The system must perform two actions that cannot be a single atomic step. It must record durably that it holds the right to pay, and it must cause the payment to happen at an external party. The external party cannot participate in a commit protocol. Between these two actions the process may crash.

Only one ordering is survivable. To act first and record afterward is catastrophic: if the process crashes in the interval, money is settled and no record of the entitlement exists. Nothing in the system knows the payment happened, so nothing looks for it. To record first and act afterward is merely difficult: if the process crashes, a durable intent exists and the disbursement effect is unknown, and the system can search for it.

The design accepts the record-first ordering and works to shrink the ambiguous interval as much as possible. Two commits and a strict durability discipline before the adapter runs bound the interval to milliseconds. The interval cannot be closed. At any moment a set of intents exists in which the system has recorded a durable intent to pay and cannot yet determine whether the payment occurred. At scale that set is never empty. The architecture is built around this fact.

The design became tractable when the goal shifted from preventing the ambiguous interval to a goal that is achievable: every intent in that interval must be decidable. Nothing irreversible may be attempted again until the system determines the state of the intent. In this system unknown is a primary permanent state, not an error. The only legal exits from unknown are positive evidence: a partner terminal statement, an entry (or absence) in a settlement file, or an operator decision. Figure 2 shows the state machine, with unknown drawn as a first-class state alongside the settled, terminal-and-not-settled, blocked-and-reported, and cancelled terminal states.

Figure 2. Payment-intent state machine: monotonic transitions with 'unknown' as a primary permanent state

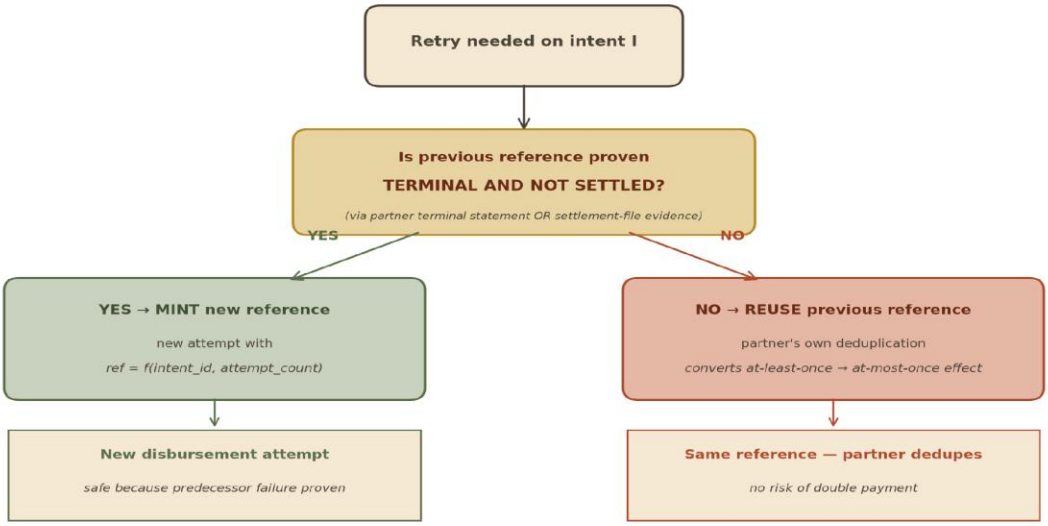


4.B The minting rule

The core protocol contribution follows from the observation that at the disbursement boundary the partner performs the actual deduplication. The sender only requests the property. Every disbursement attempt is submitted with a deterministic external reference. To create a reference the system uses the intent identity and the attempt count.

As a hard rule: workers may create a new external reference only if they have proof that the previous reference is in a terminal-and-not-settled state. If that proof is absent, a retry must use the previous reference so that the partner deduplicates the request. Figure 3 renders this decision.

Figure 3. The minting rule: when a retry may create a new external reference and when it must reuse the previous one



This rule converts at-least-once transport into at-most-once effect without requiring the counterparty to participate in a commit protocol. The guarantee is not exactly-once. It is the strongest guarantee available at this boundary.

Many implementations fail this rule because creating a new reference during a retry is common practice in general distributed-systems engineering. That practice is safe for idempotent HTTP handlers. It is a duplicate-payment generator at the disbursement boundary. The rule surfaces exactly the moment at which general distributed-systems intuition and payments-domain reality diverge.

4.C The lying oracle

The rule holds only if the evidence used to resolve unknown is reliable. The automated evidence source is not. A query-by-reference is not a linearizable read of the settlement state the partner maintains internally. It is a read of the partner's reporting interface, which is a separate system with its own replication lag, batch boundaries, and rules for when a payout becomes visible. When a probe returns not found, the result can mean the transaction is unsettled or that settlement occurred thirty seconds ago and has not yet propagated to the queryable view.

For this reason the terminal-and-not-settled requirement is strict. A negative probe result is not accepted as evidence unless the partner guarantees read-after-write consistency on its own reference namespace. Where that property is missing, the intent stays in unknown, and determination waits for the settlement file. The file is the partner's accounting record rather than a view of it. The absence of a record after a cycle closes is true evidence. Determination latency is therefore reported per capability class: with the weakest class the bound is the reconciliation-cycle interval, not the online operation budget.

4.D Reference namespace ordering

The rule requires a total order over every attempt on an intent, expressed through the external reference the partner observes. In production, partners often truncate references to fit legacy fields, change case, remove characters, or scope uniqueness to a single settlement window. When they do, the reference can be reused after a cycle boundary. If two attempts appear as one reference to the partner, or one attempt appears as two references, the ordering argument fails at exactly the point that matters. The system therefore treats reference construction as a negotiated property per partner, validated during integration and written into the capability-class declaration. If a partner's namespace cannot hold a globally unique reference, that partner is placed in the weakest recovery class regardless of API-documentation

claims. This is a difficult part of the design because it enforces a protocol on a partner who did not agree to it and finds violations through observation rather than through contract.

5. Compliance as State-Transition Precondition

Compliance is not a layer. Enforcement occurs at multiple locations across a single transfer and applies to different items at different times. The source jurisdiction imposes licensing, disclosure, and consumer-protection obligations. The destination jurisdiction imposes payout, identity-verification, and reporting obligations. The settlement currency and any intermediary attach further obligations. Every jurisdiction that processes identity data imposes data-protection obligations. These groups do not form a hierarchy and are sometimes in direct conflict.

Because no single point of the transfer can evaluate all obligations at once, compliance must be part of the same state machine that provides fault tolerance. The payment intent is both the recovery artefact and the compliance artefact. Design choices that appear to be for distributed systems are also compliance mechanisms, and their dual purpose is what makes the architecture defensible.

Screening as a transition precondition. The screening decision is attached by reference to the intent with a validity range: the list version evaluated, the policy version applied, and a temporal limit. The presence of a valid decision is what protects the transition from screened to payable. Because the control is a state-transition precondition, no path to disbursement bypasses it. This is not a "helpful side effect" of the recovery discipline; it is the actual compliance requirement, and recovery uses the same mechanism.

Re-evaluation on list update. When a sanctions list updates, an event triggers re-evaluation of intents currently in progress rather than only new intents. A delayed state is an active state that re-enters the control plane, not a queue of dormant items. The system records the policy version on the intent so that a later inquiry can judge the transfer against the rules in force at the time of the transfer.

Terminal states the payments world requires. Two states in Figure 2 exist because compliance defined them rather than because payment operations required them. Blocked-and-reported is the state a sanctioned intent enters. Regulation does not permit return of funds to the sender in this case; the operator must hold the funds and file the required report. The state must be terminal, not-settled, and not-refundable. Cancelled-consumer is the state that consumer-protection cancellation windows require. Product speed frequently outpaces the legal cancellation window; in such corridors the operator loses the ability to reverse and accepts the cost of that choice, recorded on the intent rather than left implicit.

Data residency as topology. Identity data does not replicate to all regions. Screening occurs where the data legally resides. The global intent record carries a screening-decision reference and the material transaction fields but not personal data. This structure is what allows the operator to fail over across regions without breaching data-localisation. It is also the reason for a floor on synchronous latency: the residency hop is unavoidable, and no caching removes it.

6. Latency Budget and Durability Placement

Durability is a precondition for irreversible action, not a requirement for every step. The system commits durably at two moments and no more, and the two commits carry different costs.

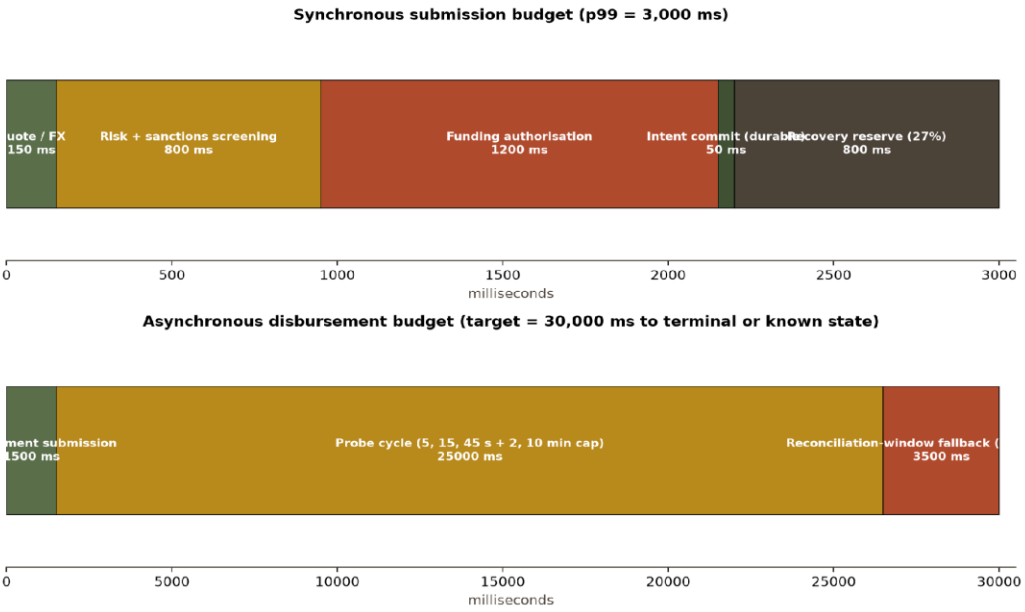
First commit — intent creation. The system persists the intent on the fast path, before any external call, into a local consensus group co-located with the outbound message. Because the record is small and confined to a single partition, the commit completes in 50 ms at p99. A local commit is sufficient because no irreversible action follows this step. If a regional failure loses the record, the sender simply resubmits.

Second commit — payable transition. After screening and funding return and immediately before the adapter runs, the system marks the intent payable. This commit requires cross-region acknowledgement because if the record

is lost after the payment is sent, the ledger is empty and the money is gone. The 30 s asynchronous budget on the disbursement leg accommodates this cost. On the 3 s synchronous path, this commit does not run.

Figure 4 shows both budgets. The synchronous budget of 3,000 ms is decomposed into quote and rate-lock (150 ms), risk scoring and sanctions screening (800 ms), funding authorisation (1,200 ms), and durable intent commit (50 ms), leaving 800 ms (27 %) as recovery reserve. The asynchronous budget targets 30,000 ms to a terminal or known state, sized around the bounded probe schedule (5, 15, 45 s and 2, 10 min, capped at five attempts within 30 min) and a reconciliation-window fallback for the weakest partner capability class.

Figure 4. Two-budget latency model: synchronous submission and asynchronous disbursement



Sequenced, not parallelised. Screening and funding are sequenced, not parallel. Parallelising would save 800 ms but would create a scenario in which the system takes funds and then screening blocks the transfer, at which point returning funds may be legally impossible. The design accepts the latency cost for the correctness guarantee.

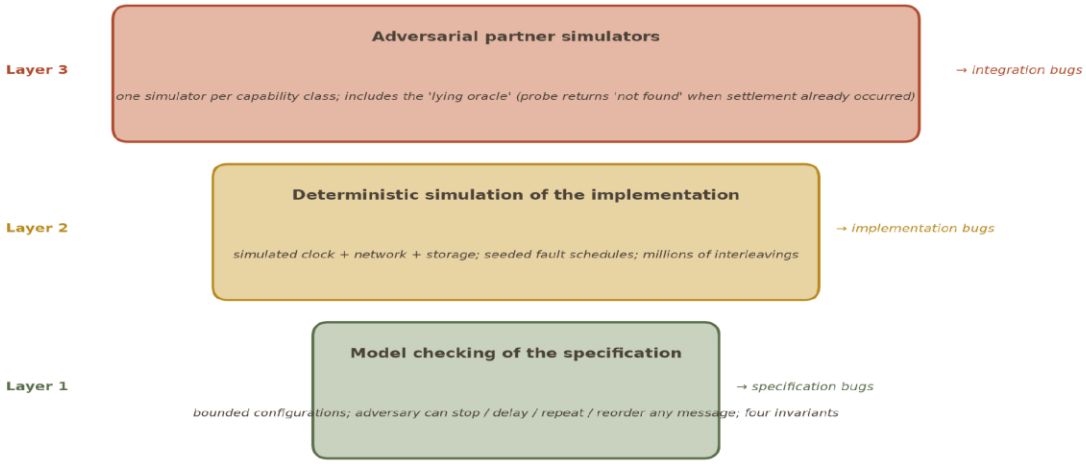
Group commit and deadline discipline. Durability writes are batched across intents that arrive at the same time, spreading the fsync cost. Each request carries a deadline that allows early abort so that late computation does not consume resources for a result that has already expired. Hedging is used on read-only or idempotent legs (quote in particular) and forbidden on the disbursement leg, where a hedge is a duplicate.

Optimisations explicitly refused. The system refuses four optimisations that could hit the latency budget but would break the guarantee: (i) acknowledging a submission from a buffer that is only durable later; (ii) relaxing the intent-log CAS to eventual consistency; (iii) holding lease state only in memory; (iv) treating a timeout as a failure. Each is documented in §IX as an explicit non-goal.

7. Testing Without Production

Production telemetry cannot demonstrate the properties the paper claims. A platform that has not duplicated over one million transfers has said nothing about the sequences of faults that did not occur. Categorical guarantees require categorical evidence. Figure 6 shows the three-layer testing methodology the paper adopts in place of production data.

Figure 6. Three-layer testing methodology: replacing production telemetry with categorical evidence



Layer 1 — model checking of the specification. The intent-state machine, the lease and fencing discipline, and the minting rule are specified in TLA+ [15, 16] and checked across bounded configurations (two controllers, two regions, one partner of each capability class). An adversary may stop, delay, repeat, or reorder any message and may crash any component at any time. Four invariants are checked: no irreversible action occurs twice; no funds are released without a valid screening decision on any path; the intent state monotonically advances; and every intent reaches a terminal state. The most valuable output is a counterexample: running the same specification without the minting rule produces a specific interleaving in which a duplicate settlement occurs, which stands as proof that the rule is necessary rather than one of several viable approaches.

Layer 2 — deterministic simulation of the implementation. The orchestrator runs in a single process with a simulated clock, a simulated network, and simulated storage, following the FoundationDB [6] and TigerBeetle [7] pattern. A single seeded generator controls every source of non-determinism (thread scheduling, timer firing, message ordering, random values, partner latency). A run is a pure function of the seed. This gives three properties that staging environments do not: simulated time collapses schedules that ordinarily take 30 min (probe cycle) or hours (reconciliation) into milliseconds; a rare failure caught by a seed becomes a permanent regression fixture; and millions of interleavings are explored automatically instead of hand-scripted. Because runs are inexpensive, the search runs continuously as a background process rather than as a release-gate step.

Layer 3 — adversarial partner simulators. One simulator exists per capability class (Figure 5). Class I partners honour idempotency keys and provide read-after-write consistency on their reference namespace. Class II partners honour keys but do not provide read-after-write consistency, so their reporting interface can lie about settlement state during replication lag. Class III partners are reconcilable only through the settlement file. The Class II simulator implements the lying oracle: it returns not found for a reference that has already settled but has not yet propagated to the reporting view. If the platform naively treats this as evidence of terminal-not-settled and mints a new reference, the simulator confirms the duplicate settlement. This is the failure mode the minting rule prevents, and the adversarial simulator is what proves the prevention works.

Figure 5. Disbursement-partner capability classes and the adversarial simulator built for each

Capability	Class I	Class II	Class III
Idempotency-key honouring	**Yes**	**Yes**	Best-effort
Read-after-write consistency on partner references	**Yes**	No	No
Query-by-reference support	**Yes**	**Yes**	No
Terminal-state declaration	**Real-time**	Delayed	File-only
Reference-namespace stability	**Stable**	Windowed	Reused across cycles
Determination path in this paper	Probe cycle	Probe cycle + fallback	Reconciliation file
Simulator role	**Baseline correctness**	Probe-lag simulation	Adversarial 'lying oracle'

Together the three layers replace production telemetry as the correctness argument for the architecture's categorical properties.

8. Reporting Discipline

The paper argues that dominant industry metrics for payment platforms are misleading. Three reporting practices follow from the architecture.

Availability is split by path. Submission-path availability is higher because failures in dependencies no longer cause the system to refuse a send. A rate-source failure produces a secondary-source quote or a cached fallback with a wider spread. A screening degradation places the intent in review rather than failing the send. Disbursement-path availability is lower by design: a screening-dependency outage produces a corridor-level halt, and the platform reports the halt separately so that operators see the inherent limit of an SLA-strict system.

Success is split into four terminal categories. Delivered within disclosure, delivered after deferral, determined-and-not-delivered with funds returned, and undetermined at report time. The rate of delivered within disclosure alone falls in this architecture compared to systems that assert outcomes they cannot prove, because the platform will not claim delivery it cannot demonstrate. The rate of undetermined at report time rises because the platform holds those cases rather than resolving them incorrectly. Both moves are honesty gains rather than performance losses.

Three counts must be zero. Duplicate settlements caused by retries or recovery; releases to disbursement that lacked a valid screening decision; and payments that exist without a record of the original right to pay. These counts are not rates. A design that avoids duplicates across a million transfers has not proven it can handle the specific sequences of faults that produce duplicates. The counts are verified through Layer 1 model checking over bounded configurations and Layer 3 fault injection against each capability class, not through operational volume. If the platform reported a rate of duplicate settlements, it would admit that duplicates are possible.

Two costs of this reporting discipline are stated openly. The 27 % reserved latency budget is not consumed under normal operation, so the p99 of the slowest one percent of tasks is higher than in a system that uses all its capacity. Determination latency for the weakest capability class is bounded by the partner's reconciliation cadence and is not improvable by internal engineering.

9. LIMITATIONS AND NON-GOALS

• No production telemetry (Q10 of the intake, as an active design principle). Rates are potentially deceptive when outcomes include assertions rather than proofs. The paper's evidence is categorical.

• No claim of exactly-once. The claim is at-most-once effect with eventual determination.

• Fault-catalogue completeness. Categorical counts are verified through Layer 3 injection against a defined catalogue of partner behaviours. A partner behaviour outside the catalogue is not covered.

• Simulator idealisation. Partner-latency distributions in the simulator are stylised; they do not capture every tail behaviour a real production feed produces.

• Regulatory scope. The paper cites DORA, SEC Reg SCI, and FINRA Rule 4370 at framework level. It does not offer a jurisdiction-specific compliance recipe.

• Optimisations explicitly refused (§VI) are documented so that a reader who wishes to relax them can see what would break.

10. CONCLUSION

The paper contributes an architecture for cross-border payments in which the payment intent is the durable append-only primitive, commit precedes irreversible action, and every recovery mechanism follows from that ordering. The minting rule is the paper's protocol contribution and converts at-least-once transport into at-most-once effect without counterparty participation. Compliance is a state-transition precondition rather than a middleware layer, so no disbursement path bypasses screening, and two terminal states (blocked-and-reported, cancelled-consumer) exist because regulation requires them rather than because payment operations request them. Latency and durability are traded through a two-budget model in which durability sits on the irreversible leg only. Evidence for the architecture is categorical: three testing layers replace production telemetry, and three counts must be zero rather than low. The paper argues that "at-most-once effect with eventual determination" is the strongest achievable guarantee at the disbursement boundary and should be adopted as the honest replacement for exactly-once framing that classical distributed-transaction literature carries into a setting where its assumptions do not hold. Reporting discipline is a first-class contribution: availability is split by path, success is split into four terminal categories, and the three zero-counts are stated as absolute claims rather than as targets.

Data Availability

Model-checking specifications, deterministic-simulation harnesses, and adversarial-partner simulators referenced in §VII are described at the level of detail required for independent implementation. No production telemetry is used, and none is required by the paper's evidence stance.

Ethics

No human-subjects data. No personal or transactional data of any real sender or receiver appears in this work.

Competing Interests

The author declares no competing interests.

REFERENCES

1. H. Garcia-Molina, K. Salem, "Sagas," in Proc. 1987 ACM SIGMOD, pp. 249-259, 1987. DOI: 10.1145/38713.38742.
2. J. N. Gray, "Notes on Data Base Operating Systems," in Operating Systems: An Advanced Course, LNCS 60, Springer, 1978.
3. R. Guerraoui, L. Rodrigues, Introduction to Reliable and Secure Distributed Programming, 2nd ed., Springer, 2011.
4. Apache Kafka contributors, "Exactly-Once Semantics in Kafka: Producer Idempotency and Transactions," KIP-98 and subsequent releases, 2017 onward.
5. N. Narkhede, G. Shapira, T. Palino, Kafka: The Definitive Guide, 2nd ed., O'Reilly, 2021.
6. Apple Inc. / FoundationDB team, "The FoundationDB Simulator," FoundationDB technical documentation, 2015 onward.
7. J. D. Greef, "TigerBeetle: Financial-Grade Distributed Systems via Deterministic Simulation," TigerBeetle documentation, 2020 onward.
8. European Parliament and Council, Regulation (EU) 2022/2554 on digital operational resilience for the financial sector (DORA), OJ L 333, 27 Dec 2022, pp. 1-79.

9. U.S. Securities and Exchange Commission, "Regulation Systems Compliance and Integrity (Reg SCI)," 17 CFR § 242.1000-1007, 2014, as amended.
10. FINRA, "Rule 4370: Business Continuity Plans and Emergency Contact Information," FINRA Manual, 2004, as amended.
11. Bank for International Settlements CPMI, "Harmonisation of ISO 20022 for enhanced cross-border payments," CPMI Report No. 211, 2023.
12. L. Lamport, "The Part-Time Parliament (Paxos)," ACM TOCS, vol. 16, no. 2, pp. 133-169, 1998. DOI: 10.1145/279227.279229.
13. D. Ongaro, J. Ousterhout, "In Search of an Understandable Consensus Algorithm (Raft)," USENIX ATC 2014, pp. 305-319.
14. R. Rodrigues, P. Druschel, "Peer-to-peer systems," Communications of the ACM, vol. 53, no. 10, pp. 72-82, 2010. doi:10.1145/1831407.1831427.
15. L. Lamport, Specifying Systems: The TLA+ Language and Tools for Hardware and Software Engineers, Addison-Wesley, 2002.
16. C. Newcombe et al., "How Amazon Web Services Uses Formal Methods," Communications of the ACM, vol. 58, no. 4, pp. 66-73, 2015. doi:10.1145/2699417.