Persistent Global Liquidity State

Distributed Custody, Shared Settlement Environments, and the Transition from Event-Based FX to State-Based Resolution

Richard Lane & Vladan Jovanovic
Co-founders, derien

Paris, France | May 2026

This paper is the second in a series. The first paper, Machine-Native Payment Economies and the Structural Limits of Pair-Based FX Resolution (Lane, 2026), established that sequential pair-based FX architectures degrade structurally under machine-native transaction density. This paper examines the coordination architecture that follows.

Abstract

Traditional foreign exchange systems operate as event-driven architectures. Currency conversion is initiated transaction by transaction, resolved pair by pair, and reconciled sequentially through arbitrage and settlement processes external to the pricing environment itself.

The emergence of machine-native payment systems alters these assumptions fundamentally. Autonomous systems transact continuously rather than intermittently, optimize across global state spaces rather than bilateral relationships, and depend upon deterministic settlement consistency in order to maintain coherent operational behavior.

The consequence is not merely increased demand for existing FX infrastructure, but a transition from event-based liquidity coordination toward persistent global liquidity state management.

This paper examines the architectural implications of that transition. It argues that machine-native settlement environments make possible a new class of liquidity coordination system in which custody remains distributed, settlement remains heterogeneous, but liquidity state becomes globally observable and continuously coordinated. The paper further argues that simultaneous multi-currency resolution does not require unified custody, sovereign harmonization, or singular balance-sheet ownership. It requires only that independently controlled custody domains exist as observable nodes within a shared programmable settlement environment. Under these conditions, liquidity coordination evolves from sequential pair matching toward continuous feasible closure discovery across persistent global state.

---

1. Introduction

Existing FX infrastructure evolved within a world characterised by intermittent human transaction flow, fragmented institutional balance sheets, and operational settlement latency. Under these conditions, pair-based markets represented an effective abstraction for coordinating global liquidity.

The emergence of programmable settlement environments alters those assumptions. Payment blockchains, tokenised deposits, stablecoin systems, and persistent shared ledger architectures introduce an environment in which balances are continuously observable, settlement state is persistently available, and machine-originated transaction flow increasingly dominates operational activity. This changes the nature of the liquidity coordination problem itself.

The previous paper in this series argued that sequential pair-based FX architectures degrade structurally under machine-native transaction density. This paper examines the next implication: what form of liquidity coordination architecture emerges once settlement environments become globally observable programmable state systems.

The central claim advanced is that machine-native liquidity coordination does not require unified custody infrastructure. It requires only:

- distributed custody nodes,
- shared settlement observability,
- and continuously coordinated global liquidity state.

This distinction is foundational. Section 2 examines the structural properties of event-based financial systems and their limitations under machine-native conditions. Section 3 introduces the concept of persistent global liquidity state. Section 4 addresses the relationship between distributed custody and coordinated state nodes. Section 5 examines the role of resolution primitives within shared settlement environments. Section 6 considers the structural implications of this transition. Section 7 concludes.

2. Event-Based Financial Systems

Traditional FX systems are fundamentally event-driven architectures. Transactions initiate resolution. Liquidity is discovered reactively. Settlement occurs downstream. The system therefore behaves as a sequence of discrete coordination events: order entry, routing, execution, settlement, and reconciliation.

Under human-originated transaction flow this structure remains manageable because transaction frequency is low relative to settlement cycles, liquidity conditions evolve slowly, and humans tolerate operational inconsistency between execution and settlement.

Machine-native systems alter all three assumptions simultaneously. Autonomous systems transact continuously, optimise globally, and maintain persistent operational dependencies on settlement state itself.

Under these conditions, event-driven liquidity coordination becomes increasingly unstable because each transaction modifies the state upon which subsequent transactions depend. The system no longer behaves as a collection of independent transactions. It behaves as a continuously evolving liquidity graph.

3. Persistent Global Liquidity State

Machine-native settlement environments make possible a different architectural model. Rather than discovering liquidity reactively transaction by transaction, the system maintains a persistently observable global liquidity state: balances exist continuously within shared settlement environments, obligations evolve continuously, and feasible resolution paths emerge dynamically from aggregate system state.

Traditional FX systems are event-centric. Persistent liquidity coordination systems are state-centric. The implications are profound.

3.1 Continuous Feasible Closure

In pair-based systems, liquidity resolution begins only after a transaction request appears. In persistent state systems, liquidity relationships already exist implicitly within the continuously evolving graph structure itself.

Resolution therefore becomes the discovery of feasible closure states inside an already observable global network. The system does not wait to construct bilateral relationships, discover routing chains, or reconcile fragmented execution afterward. Instead, feasible multi-currency closure emerges continuously from aggregate state conditions.

This is not equivalent to periodic netting or synchronised batch settlement. The system does not require global synchronisation, discrete resolution windows, or universal transaction batching. It requires only persistent shared state visibility sufficient to maintain continuously converging liquidity coordination.

3.2 State Convergence Under Asynchrony

Global commerce is inherently asynchronous. Transactions originate continuously across jurisdictions, settlement environments, custodians, and time zones. Machine-native liquidity coordination therefore does not eliminate asynchrony. It manages consistency under asynchrony.

This distinction parallels distributed systems architecture more closely than traditional financial market structure. Distributed databases do not require globally synchronised writes or singular

physical infrastructure ownership. They maintain coherent state consistency across distributed nodes under asynchronous conditions (Lamport, 1978; Lynch, 1996).

Similarly, persistent liquidity coordination systems maintain continuously converging feasible closure across distributed custody and settlement environments. The relevant constraint is not synchronisation but consistency: the system must guarantee that coordinated closure states remain feasible across all participating nodes, even as individual nodes update asynchronously (Brewer, 2000).

The architecture therefore evolves:

- from transaction sequencing,
- toward persistent state convergence.

4. Distributed Custody as Coordinated State Nodes

One of the most common misconceptions surrounding simultaneous multi-currency resolution is the assumption that it requires unified custody infrastructure. It does not. The architecture does not require a single custodian, a single issuer, a unified sovereign reserve pool, or singular balance-sheet ownership. It requires only that independently controlled custody domains exist as observable nodes inside a shared programmable settlement environment.

4.1 Separation of Coordination and Custody

Traditional financial systems frequently collapse custody, settlement, issuance, and liquidity coordination into tightly coupled institutional structures. Machine-native settlement environments allow these functions to separate cleanly.

Under such systems, custody remains distributed, issuance remains heterogeneous, and settlement infrastructure remains independently controlled. The coordination layer operates above these systems as a neutral liquidity state management architecture.

This is structurally analogous to packet routing within communication networks. The internet did not eliminate independent networks, sovereign jurisdictions, or heterogeneous infrastructure ownership. It coordinated them. Similarly, persistent liquidity coordination architectures do not eliminate financial fragmentation. They coordinate fragmentation coherently.

4.2 Shared Settlement Visibility

The critical requirement is therefore not unified custody. It is shared settlement visibility. Once balances exist inside a common programmable settlement environment, custody nodes become observable, settlement states become deterministic, and liquidity relationships become continuously computable.

Under these conditions, multi-currency coordination can emerge above heterogeneous custody systems without requiring sovereign or institutional consolidation. The settlement layer provides observable state, deterministic transitions, and programmable finality. The coordination layer manages feasible closure, liquidity optimisation, and state convergence. These are distinct architectural functions.

This architecture has a precise parallel in distributed consensus systems. Nakamoto (2008) demonstrated that deterministic shared state could be maintained across fully distributed, untrusted nodes without requiring a central coordinating authority. The implication for financial infrastructure is that settlement finality and custody observability are separable from ownership and control. Coordination above heterogeneous custody systems becomes structurally feasible once shared state visibility is established at the settlement layer.

5. Resolution Primitives and Shared Settlement Environments

Payment blockchains are frequently described as settlement infrastructure, stablecoin rails, or programmable payments systems. Increasingly, they may be better understood as persistent shared financial state environments. This distinction matters because persistent global liquidity coordination becomes possible only once balances remain continuously observable, state transitions remain deterministic, and settlement consistency exists natively inside the environment itself.

Under these conditions, a resolution primitive occupies a structurally distinct role. It:

- does not intermediate credit,
- does not custody assets,
- does not manage issuance,
- and does not require ownership of settlement infrastructure.

Its function is singular: to maintain continuously converging global liquidity coordination across observable settlement nodes. This positioning is analogous to routing layers within communication systems, consistency coordination layers within distributed databases, or synchronisation architectures within distributed computation environments (Tanenbaum and van Steen, 2007).

The coordination layer does not replace the infrastructure beneath it. It organises it. The resolution primitive therefore sits above the settlement environment in the same way that a routing protocol sits above physical network infrastructure: indispensable to the function of the system without owning any component of it.

The mathematical foundation for this class of coordination problem is well established. Ford and Fulkerson (1956) demonstrated that feasible flow states could be identified within network

graphs subject to capacity constraints. Machine-native settlement environments do not introduce a new mathematical framework. They introduce operating conditions — persistent shared state, deterministic transitions, continuous autonomous flow — under which such frameworks become structurally necessary at global scale.

6. Structural Implications

The transition from event-based FX toward persistent global liquidity state coordination produces several structural consequences.

6.1 Liquidity Becomes Endogenous to State

In pair-based systems, liquidity is discovered transaction by transaction. In persistent state systems, liquidity emerges from aggregate network conditions continuously. A conversion path that appears illiquid bilaterally may resolve efficiently once evaluated across globally observable state conditions.

Liquidity ceases to behave as fragmented bilateral inventory. It becomes a property of continuously coordinated network state. This represents a qualitative change in how liquidity is understood, managed, and optimised within financial infrastructure.

6.2 Settlement and Resolution Converge

Traditional systems separate execution, settlement, and reconciliation. Persistent state architectures increasingly collapse these distinctions. Once feasible closure emerges directly from globally observable settlement state, execution and settlement become increasingly inseparable. The system state after resolution already represents the coordinated settlement state itself.

This convergence has direct implications for latency, counterparty risk, and capital efficiency. Settlement that is a consequence of resolution rather than a downstream process eliminates the interval during which obligations exist but have not yet been discharged — the primary source of settlement risk in traditional FX infrastructure.

6.3 Financial Topology Changes

The most important implication is architectural. Traditional FX systems coordinate transactions. Persistent liquidity coordination systems coordinate state. This represents a deeper transition than faster payments, lower latency, or improved routing efficiency. It represents a change in the topology of liquidity coordination itself.

Once the coordination layer operates on persistent global state rather than discrete transaction events, the system acquires properties that event-driven architectures cannot replicate incrementally: continuous closure discovery, endogenous liquidity emergence, and settlement consistency that exists at execution rather than being restored afterward.

7. Conclusion

Machine-native payment systems alter the operating assumptions upon which traditional FX architecture evolved. Autonomous systems transact continuously, optimise globally, and depend upon deterministic settlement consistency. Under these conditions, event-driven pair-based liquidity coordination becomes increasingly unstable.

The resulting transition is not merely from slower systems to faster systems, or from manual systems to automated systems. It is a transition from event-based financial coordination toward persistent global liquidity state management.

This transition does not require:

- unified custody,
- sovereign harmonisation,
- or singular infrastructure ownership.

It requires only:

- distributed custody nodes,
- shared programmable settlement environments,
- and continuously observable state.

Under such conditions, liquidity coordination evolves from sequential pair matching toward continuous feasible closure discovery across persistent global financial state.

The ultimate consequence is not simply improved FX infrastructure. It is the emergence of a new class of financial coordination architecture optimised for machine-native economic systems. The coordination layer does not own the infrastructure. It makes the infrastructure coherent.

References

Brewer, Eric (2000). Towards Robust Distributed Systems. *Proceedings of the ACM Symposium on Principles of Distributed Computing (PODC).*

Ford, L.R.; Fulkerson, D.R. (1956). Maximal Flow Through a Network. *Canadian Journal of Mathematics*, 8, 399–404.

Lamport, Leslie (1978). Time, Clocks, and the Ordering of Events in a Distributed System. *Communications of the ACM*, 21(7), 558–565.

Lane, Richard (2026). Machine-Native Payment Economies and the Structural Limits of Pair-Based FX Resolution. *Working Paper, derien.*

Lynch, Nancy (1996). *Distributed Algorithms.* Morgan Kaufmann.

Lyons, Richard (2001). *The Microstructure Approach to Exchange Rates.* MIT Press.

Nakamoto, Satoshi (2008). Bitcoin: A Peer-to-Peer Electronic Cash System. *bitcoin.org.*

O\'Hara, Maureen (1995). *Market Microstructure Theory.* Blackwell.

Tanenbaum, Andrew; van Steen, Maarten (2007). *Distributed Systems: Principles and Paradigms.* Prentice Hall.