Concurrence Collision

Why independence is a function of timing, and why the fix is not another layer

THOMAS ROCHA III

APR 21, 2026

Two earlier pieces set up a question neither of them answers.

The **Coordination Limit** described the physics. Coordination cost scales as a product, not a sum. Participants, modalities, features, authority domains, transport boundaries. The system becomes coordination-bound before it becomes compute-bound.

The **Compliance Boundary** described the consequence. The burden of proof has moved from configuration to behavior. Fragmented

architectures reconstruct. They do not prove. Procurement has begun excluding what cannot be proven.

Both pieces describe what is happening. Neither explains why it is happening **now**.

Identity, policy, transport, modality, authority. These dimensions have existed for decades. Sessions existed in 1998. Policy enforcement existed in 2005. Agents existed before this cycle. Why did an architecture that worked for twenty years stop working in the last eighteen months?

The answer is not complexity. It is **timing**.

The Claim

Two constraints are independent only when the interval between their state changes exceeds the time required to reconcile them.

That sentence is the entire piece.

If identity changes once per session and reconciliation across subsystems takes three seconds, identity and policy are independent. You can enforce them in separate systems. The reconciliation fits inside the interval.

If identity changes every hundred milliseconds because an agent is switching contexts, and reconciliation still takes three seconds, identity and policy are no longer independent. They are coupled, whether you designed them that way or not. The architecture still treats them as independent. The system does not.

Independence is not a design property. It is a **timing property**. When the interval collapses, so does the independence.

SOSUS, Inverted

I worked with **SOSUS** in the Navy as an Ocean Systems Technician Analyst. The system tracked submarines across ocean basins for months. A contact detected off Iceland could be the same contact tracked off the Azores three weeks later. Arrays rotated. Cables failed. Sensors dropped out. The contact kept its designation.

The reason it worked is the reason people miss when they look at modern systems. SOSUS was not continuity of transport. Sensors failed constantly. **Authority continuity survived transport discontinuity** because something above the sensors held the contact.

What we have now is the inverse condition. Transport continuity is excellent. TCP connections stay up. Retries work. APIs return. The packets arrive. **Authority fragments anyway.**

No system holds the interaction. Identity is refreshed in one subsystem. Policy is evaluated in another. AI context is rebuilt at every step. The transport is alive. The interaction is not. SOSUS worked because authority was the primitive and transport was assumed to fail. Modern systems fail because transport is the primitive and authority is assumed to be reconstructable.

That assumption held while reconstruction was fast enough. It is no longer fast enough.

What Fragmentation Assumed

Fragmentation was never neutral. It made a bet: The interval between state changes would stay larger than the interval required to reconcile across subsystems.

That bet held for two decades. In the **IoE paper** I published on December 31, 2025, I described the architectural condition. Fragmentation succeeded because operational benefits outweighed coordination costs. Modularity simplified debugging. Independent scaling allowed targeted allocation.

What the last eighteen months have made visible is why those coordination costs stayed bounded for so long. Time was on the architecture's side. State changes were sparse. Reconciliation windows were wide. Eventual consistency was not a compromise; it was a correct description of how the domains actually interacted. They barely touched.

That is no longer the regime.

What Compressed the Timing

Three things collapsed the interval:

- **Agents:** An agent does not wait. One agent produces more state changes per second than a room of humans produces per hour.
- **Regulatory attachment to runtime:** Consent, residency, and accessibility must be enforced at the moment of interaction, not verified at storage. None of these can wait for nightly reconciliation.
- **Feature convergence:** Transcription, translation, summarization, and fraud detection are now the primary flow. Each produces state the others must respect instantly.

At some point, the ratio inverts. State changes faster than reconciliation. Independence ends.

What This Frame Predicts

If the problem were complexity, throwing compute at it would help. It does not. Hyperscalers have thrown compute at coordination failures for three years; the failures scaled with the compute.

The variable was not the scale. The variable was the **interval**. A system that handled a workload before agentic access cannot handle the same workload once agents are introduced. The compute is identical. The topology is identical. The event rate is not.

Why Additive Fixes Cannot Touch This

Every patch the industry has deployed adds a subsystem. Idempotency layers. Durable workflow engines. Policy evaluation points. Each one is a new domain that must be reconciled.

Adding a subsystem lengthens reconciliation. It does not compress the event rate. The ratio moves in the wrong direction. The harder the industry tries to fix this with more tooling, the faster it crosses the threshold. Every added layer is another slot the state must pass through before the system can claim to know itself.

The Condition

Concurrence is not concurrency.

Concurrency assumes a place where conflicts are resolved (locks, transactions, consensus protocols). **Concurrence** describes the condition where no such place exists. Multiple domains, designed to be independent, become coupled by timing. No arbitration authority exists because none was built. Each domain still believes it is operating in isolation.

Concurrence is the condition that forces the **Coordination Limit**. It is the condition that makes the **Compliance Boundary** unsatisfiable.

Fragmentation worked while time was on its side. Agents, runtime regulation, and feature convergence compressed the interval between state changes below the interval required to reconcile them. At that point, independence became a property the architecture could no longer provide.

It lost the ability to reconcile what it was doing while it was doing it. That is not a performance problem. That is the regime.

Independence is a function of timing. Time ran out.