Emergent Automation

On the Latent Interrupt Architecture of Bitcoin's UTXO Model

Feardorcha Ó hUanacháin
freddie@yenpoint.jp

9 March 2026

Abstract

We present a mechanism in which autonomous cascade execution emerges from Bitcoin's existing UTXO mechanics without modification to the base consensus rules. The mechanism is built on BOLT [1], the Bitcoin Original Layer-1 Token Protocol, which unlocks a native execution layer within Bitcoin's existing script engine, without a secondary ledger or externally imposed virtual machine. A BOLT token settle transaction (`settleTx`) presented to an EventListener UTXO gives miners a direct, irrevocable incentive — fees locked in the UTXO graph itself — to construct a trigger transaction at block assembly time, propagating a cascade of state changes through the mempool. Execution uniqueness is guaranteed by double-spend prevention. Cascade depth is unbounded under Teranode's SPV architecture. Overlay nodes that watch the mempool may act before block assembly, providing a pre-confirmation speed tier, but the mechanism is complete without them. We provide evidence that no existing smart contract platform achieves this combination of properties, and analyse the structural reasons why.

1 Introduction

Every major blockchain platform has approached the problem of on-chain automation the same way: define an execution environment, charge a fee for computation, and trust the validator set to run the code when instructed. Conventional chains treat the ledger as a host for programs; however, these programs remain inert until externally invoked. Nobody asks whether the chain itself might already contain the properties required for automation without running any programs at all.

Brian Armstrong, CEO of Coinbase, recently observed: "Very soon there are going to be more AI agents than humans making transactions. They can't open a bank account, but they can own a crypto wallet. Think about it" [2]. The observation is correct. The implication points toward specific architectural requirements that AI agent economies impose: machine-speed settlement, fees committed in advance with no runtime uncertainty, execution that does not depend on a live process to initiate it, and throughput that scales with the number of agents rather than capping it.

Bitcoin's UTXO model has been consistently underestimated because its simplicity was mistaken for limitation. UTXO scripts are not Turing complete. They have no loops, no persistent state, no inter-contract calls. These observations are true. They are also beside the point, because the requirements for automation — execution uniqueness, incentivised response, causal verifiability, and depth without artificial limits — do not depend on Turing completeness. They depend on economic structure and graph topology.

The foundation of this mechanism is BOLT: the Bitcoin Original Layer-1 Token Protocol. BOLT defines tokens as UTXO-native constructs whose validity is enforced entirely by Bitcoin's existing script engine and miner validation. There is no secondary ledger, no side chain, no virtual machine. A BOLT token is a Bitcoin UTXO with a locking script that encodes token state; spending it under the correct conditions is the token operation. Miners validate BOLT token transactions the same way they validate every other transaction — by script execution. BOLT requires no modifications to the base protocol and no cooperation from any party other than the miner who includes the transaction in a block.

This paper presents a mechanism that extends upon BOLT tokens with three properties Bitcoin has always possessed:

1. UTXO spending conditions can encode full on-chain validation of a BOLT token settle transaction, enabling deterministic trigger logic enforced entirely at the script level without any

off-chain verification step.

2. A spent UTXO cannot be spent again — execution uniqueness by arithmetic, enforced by every node on the network.

3. Fees embedded in a UTXO graph compel miners to construct trigger transactions at block assembly time — incentivised execution with no keeper network, no external watcher, and no additional infrastructure.

The combination produces what we call *emergent automation*: cascade behaviour that arises from the native mechanics of BOLT settle transaction graphs operating under Nakamoto consensus. We present the mechanism architecture in full, analyse the economic model that sustains it, and provide a comparative analysis demonstrating the structural reasons why no existing alternative satisfies all four properties simultaneously.

2 Background

2.1 The Automation Problem

An automated ledger action executes as a deterministic consequence of a prior event without requiring a human-operated system to submit the executing transaction. The goal is not merely scheduled tasks but automation that is cryptographically guaranteed, auditable on-ledger, and persistent without application infrastructure. No existing platform provides this in full; each relies on at least one exogenous component.

2.2 Interrupt-Driven Programming as the Correct Frame

The correct analogy is *interrupt-driven programming*: a processor remains in a low-power state until an interrupt signal triggers a specific handler, immediately returning to its baseline state upon completion. Cost is proportional to events, not elapsed time. Every other blockchain automation model is polling-based — a keeper bot or canister heartbeat that wakes on a clock, checks state, and acts if a condition is met. The cost is continuous; execution is not causally linked to the trigger on the ledger. In this mechanism, the EventListener UTXO is the interrupt handler: a BOLT token issuer identity registered at deployment, dormant at zero cost, firing exactly once when a valid `settleTx` satisfying its validation conditions is presented as an unlock argument. The response — the `triggerTx` constructed by the miner — is recorded in the same block as the triggering settle transaction.

3 Related Work

3.1 Bitcoin Script Automation Prior Art

Early attempts to achieve scheduled execution on Bitcoin used `OP_CHECKLOCKTIMEVERIFY` (CLTV) and `OP_CHECKSEQUENCEVERIFY` (CSV) timelocks to enforce temporal spending conditions. These primitives provide passive temporal constraints on UTXO spending but do not create event-driven execution: a timelock-encumbered UTXO can only be spent after a point in time, not *as a result of* a specific on-chain event. The Bitcoin covenant proposals — most notably BIP-119 (`OP_CHECKTEMPLATEVERIFY`) — extend script introspection to constrain the structure of the spending transaction, enabling rudimentary output routing. Neither timelocks nor covenant proposals provide the full EventListener pattern: on-chain parsing and validation of an arbitrary incoming transaction, combined with miner-incentivised response.

3.2 Token Protocols on UTXO Chains

Prior UTXO-native token protocols, including Omni Layer and the RGB protocol, define token validity through overlays or client-side validation rather than through Bitcoin's native script engine. Token validity in these systems is enforced off-chain by the recipient and is invisible to miners; there is no miner-enforced token validity guarantee. BOLT differs from these approaches in that token validity is enforced at the script level, making miners the validation layer rather than an external client. This distinction is the prerequisite for the EventListener pattern: a locking script can only parse and validate an incoming token transaction if the token's structure is deterministic and miner-enforceable.

3.3 Smart Contract Automation Platforms

The smart contract automation problem has been recognised since at least 2015 with the Ethereum Alarm Clock project [3]. The subsequent development of keeper networks — Gelato Network (2019), Keep3r Network (2020), and Chainlink Automation (2021) — represents the industry's canonical response: exogenous bots that poll chain state and submit execution transactions when conditions are met [4]. These systems correctly identify the problem

but solve it by introducing infrastructure dependencies that the EventListener architecture eliminates. The Internet Computer's canister timer system represents the most architecturally serious prior attempt to provide on-chain scheduling without external infrastructure [5]; its limitations are analysed in Section 6.

4 Protocol Architecture

Throughout this section it is important to distinguish two conceptual layers. **BOLT** (Bitcoin Original Layer-1 Token Protocol) is the token protocol: it defines how tokens are represented and transferred as native Bitcoin UTXOs. The **EventListener** and **EventTrigger** contracts form a separate automation layer that is *built on top of* BOLT — they parse and validate a BOLT protocol transfer `settleTx` to trigger downstream execution, but they are not part of BOLT itself.

Table 1 summarises the role and resting state of each component in the Settle-Trigger loop.

Table 1: Settle-Trigger Loop: Component State Transition

Component	Responsibility	Status at Rest
EventListener	Validates the incoming BOLT protocol transfer `settleTx`	Dormant UTXO (Zero Cost)
EventTrigger	Enforces co-spending and output routing	Created by BOLT protocol transfer `settleTx`
TriggerTx	Executes the state change and collects fees	Finalised On-Chain

4.1 The EventListener UTXO

An EventListener UTXO is a Bitcoin output whose locking script encodes two deployment-time arguments: `issuerPubKey` (the 33-byte compressed public key of the token issuer) and an optional `fundingOutpoint` (a 36-byte outpoint identifying a funding UTXO that must be co-spent in the trigger transaction, or `OP_0`). The output carries embedded satoshis above the dust threshold that are irrecoverable until a valid trigger transaction claims them; they are the execution fee committed at deployment.

To spend an EventListener UTXO — that is, to fire the trigger — the spender supplies a raw BOLT *settle transaction* (`settleTx`) as an unlock argument alongside the SigHash preimage (`ctx`). The locking script then fully parses and validates that `settleTx` on-chain, performing the following checks.

Token input validation. The script parses all 42 fields of the token input's scriptSig. Fields 1–31 carry ancestor data and may be `OP_0` in the no-bolt (first-transfer- settle) case. Fields 32–36 are always present and size-validated: `fundOutpoint` (36 bytes), `changeOutput` (variable), `beneficiaryPubKeyHash` (20 bytes), `sig` (variable DER), and `pubKey` (33 bytes, retained for issuer verification). The token's locking script suffix bytes are `SHA256`-hashed and compared against the known BOLT template suffix hash, confirming the token was locked under an authentic BOLT script.

Two-input vs. three-input cases. A no-bolt settle transaction has `vinCount = 2`: the token input and a P2PKH funding input. Because no ancestor BOLT proof input is present, the script enforces `issuerPubKey == pubKey` from field 36, confirming the issuer signed the genesis settlement. A with-bolt settle transaction has `vinCount = 3`: the token input, a p2pb proof input carrying the `b017` marker, and the funding input. In this case the issuer equality check is skipped; the ancestor proof chain provides ownership proof.

Token and change output validation. The settle transaction's first output (vout0) is validated against the BOLT template suffix hash, confirming a valid BOLT token continues to exist. The second output (vout1) must be locked to an EventTrigger UTXO with `outpointTrigger` set to this EventListener's own outpoint, binding the downstream contract back to this specific deployment.

Co-spend enforcement. The script computes:

$$\texttt{settle\_change\_outpoint} = \texttt{hash256(settleTx)} \,\|\, \texttt{01000000}$$

then assembles serialised prevouts:

$$P = \texttt{EL\_outpoint} \,\|\, \texttt{settle\_change\_outpoint} \; [\,\|\, \texttt{fundingOutpoint}\,]$$

and verifies $\texttt{hash256}(P) = \texttt{hashPrevouts}$ from the SigHash preimage. This ensures the EventListener

UTXO and the EventTrigger UTXO created by the settle transaction are spent together in the same `triggerTx`; neither can be spent independently.

Between deployment and trigger the EventListener UTXO consumes no resources. No process monitors it. No cycle balance depletes. It persists on the UTXO set, dormant, until a valid `settleTx` satisfying all conditions above is presented.

4.2 The EventTrigger Contract

The EventTrigger UTXO holds the settle change output created by the BOLT settle transaction. Its locking script bakes in one deployment-time argument: `outpointTrigger` (36 bytes) — the outpoint of the EventListener UTXO this EventTrigger is paired with.

To spend an EventTrigger UTXO the spender supplies the SigHash preimage (`ctx`) plus two optional unlock arguments: `changeOutput` (the full serialised bytes of the `triggerTx` output to verify, or `OP_0`) and `fundingOutpoint` (a 36-byte outpoint or `OP_0`).

Co-spend validation. The locking script constructs:

$$P = \texttt{outpointTrigger} \,\|\, \texttt{ctx.outpoint} \,[\,\|\, \texttt{fundingOutpoint}\,]$$

where `ctx.outpoint` is the EventTrigger's own outpoint in the spending transaction (i.e., the settle change outpoint). It verifies:

$$\texttt{hash256}(P) = \texttt{hashPrevouts}$$

This mirrors the EventListener co-spend check: the EventListener verifies the EL outpoint and settle change outpoint appear together in `hashPrevouts`; the EventTrigger verifies its own outpoint and the EL outpoint appear together. Both checks must pass simultaneously, so the `triggerTx` must spend both UTXOs as inputs — and no other arrangement satisfies both scripts.

Optional output verification. If `changeOutput` is non-zero, the script verifies `hash256(changeOutput) = hashOutputs`, constraining the `triggerTx`'s output set to exactly the specified bytes. This enables deterministic downstream cascade routing without a more complex locking script.

The EventTrigger carries the satoshis from the settle change output. Together with the embedded fee in the EventListener, these form the composite economic incentive for a miner or overlay node to construct the `triggerTx`.

It is worth noting that the combination of BOLT's native Layer-1 execution and Teranode's unbounded parallelised scaling renders Layer-2 constructs such as the Lightning Network structurally obsolete for this class of application. Lightning and analogous off-chain channel networks exist solely to work around base-layer throughput constraints; they do so by deferring settlement, weakening finality guarantees, and introducing counterparty risk during the off-chain period. Where the base layer can process arbitrary transaction volumes with immediate finality and miner-enforced security, there is no architectural justification for accepting those trade-offs. Every hop that Lightning routes off-chain to avoid congestion, this protocol settles on-chain with full Nakamoto security at negligible cost.

4.3 TriggerTx Construction

When a qualifying BOLT token arrives at an EventListener UTXO, the primary executor is the miner. During block assembly, a Teranode miner reads the `EventTrigger`, constructs the `triggerTx` it specifies, and includes it in the block alongside the triggering token transaction. The fee embedded in the EventListener UTXO is the miner's incentive. No overlay infrastructure exists; the protocol is complete.

Overlay nodes — specialised operators watching the mempool for trigger conditions — provide a welcome additional execution tier. An overlay node that spots a qualifying token transaction can construct and broadcast a valid `triggerTx` before block assembly, capturing the embedded fee first. If it does, the miner includes the already-broadcast `triggerTx` rather than constructing one. All subsequent overlay attempts referencing the same EventListener UTXO input are rejected as double spends. Overlay nodes therefore compete on speed, driving pre-block execution latency toward the physical minimum — but their participation is an enhancement, not a requirement.

4.4 Cascade Propagation Under Teranode SPV

Each hop in a cascade carries a Merkle proof of input validity. Teranode nodes verify the proof without traversing the ancestor chain, eliminating the 25-ancestor mempool limit. Cascade depth becomes a fee and block-space question, not a protocol limit. All hops propagate in the mempool simultaneously; the entire cascade settles in a single block.

5 Execution Uniqueness

A trigger must fire exactly once. Every automation platform must solve this problem. The solutions vary in their complexity, their trust assumptions, and their failure modes.

This protocol solves it with arithmetic. A UTXO can be spent exactly once. The first `triggerTx` referencing an EventListener UTXO as input is accepted by the mempool. All subsequent attempts referencing the same input are rejected as double spends by every honest node on the network. No coordination layer is required. No registry tracks which node executed which trigger. No queue enforces ordering.

The trust assumption is identical to the trust assumption for any Bitcoin transaction: that honest nodes constitute a majority of hash power. This is the assumption the entire industry already accepts for the security of every satoshi on the network. Execution uniqueness in this protocol is secured by the same guarantee that secures Bitcoin itself — not by a supplementary mechanism added for this purpose.

Put simply: execution uniqueness is the difference between an automation system and a slot machine. Without it, you cannot know whether your contract ran once, never, or three times.

6 Comparative Analysis

We evaluate each major platform against the four properties defined in Section 1. A platform satisfies a property only if it does so without introducing an exogenous dependency. Where a platform fails a property, we identify the specific architectural decision responsible and explain why it cannot be resolved through incremental improvement.

6.1 Ethereum and EVM-Compatible Chains

No Ethereum contract executes without an external transaction initiating it. The contract is a program waiting to be called. Ethereum's response is the keeper network — Chainlink Automation, Gelato, Keep3r — external bots that watch the chain and submit execution transactions [4, 3]. Each reintroduces the exogenous dependency smart contracts were supposed to eliminate. As Gelato's own documentation states, smart contracts are "functionally inactive" without an external trigger: "if there is no external push, they are functionally inactive. In order to execute the logic of these programs and change the state, an external party is first required to send it a transaction" [3]. Fees are unknown at deployment, uniqueness is enforced by an off-chain registry rather than cryptography, and the causal chain between trigger and response exists in keeper logs rather than on the ledger. EVM-compatible Layer 2s inherit all of these limitations and add sequencer censorship risk.

6.2 Solana

Solana's `onAccountChange` and `onLogs` subscriptions are client-side WebSocket connections running on developer infrastructure. Logs are output, not input: nothing on the chain reacts natively to another transaction's log.

Solana's sub-second block time is frequently cited as making this gap irrelevant. It does not. Even granting a 400ms block time, an event-driven response on Solana requires: (1) an off-chain watcher receives the WebSocket notification; (2) the watcher constructs a response transaction; (3) the watcher submits it to the network; (4) the network confirms it in the next block. Steps 1–3 alone introduce network and processing latency measured in hundreds of milliseconds to seconds. The confirmation in step 4 adds another block period. Furthermore, independent analysis shows that roughly 70–80% of all transactions recorded on Solana are automatic validator votes rather than user-generated transactions, meaning a substantial fraction of Solana's advertised TPS figure represents consensus overhead rather than application throughput [6]. The total round-trip is a function of infrastructure latency and block cadence, not block time alone — and it requires a live watcher process at every step. Remove the watcher and nothing happens, regardless of how fast the chain runs.

Speed describes how quickly the chain processes transactions it has already received. Reactivity describes whether the chain responds autonomously to state changes. Solana optimises the former. This protocol addresses the latter. They are not the same property.

6.3 Internet Computer (ICP)

ICP is the most architecturally serious competitor. Canisters can schedule tasks and operate without off-chain infrastructure. The autonomy mechanism is, however, timer-based: ICP supports two scheduling primitives, timers (single-expiration or periodic calls with a specified minimum interval) and heartbeats (legacy periodic invocations with intervals close to the 1-second blockchain finalization rate) [5]. A canister wakes on a clock interval, checks state, and acts. This is polling, not event-driven execution. The canister is triggered by a clock, not by the state change

itself.

Additional structural problems compound this limitation. The ICP documentation explicitly states that during canister upgrades, all timers are deactivated and the timer list is cleared [5]: it is the canister developer's responsibility to serialise and reactivate timers across upgrades. Silent failure on cycle depletion, non-atomic cross-subnet settlement, and the absence of a competitive execution market further distinguish ICP's model from the one presented here.

6.4 Other Platforms

Tezos, Cosmos, Aptos, and Sui all require an external transaction to invoke any contract. Cardano has the UTXO model but not the trigger architecture: Plutus scripts require off-chain construction and submission. Near and Avalanche subnets use time-based schedulers with the same fundamental limitation as ICP timers.

6.5 Consolidated Property Matrix

Table 2 summarises twelve properties most relevant to automated cascade execution across the platforms examined.

† **BOLT** (Bitcoin Original Layer-1 Token Protocol) is the token protocol; it defines how tokens are represented and transferred as native Bitcoin UTXOs. **EventListener** and **EventTrigger** are the automation-layer contracts that extend BOLT settle transactions into an interrupt-driven execution mechanism. The two are distinct: BOLT can be used without the automation layer, and the "Bitcoin BOLT + EA" column reflects the full Emergent Automation stack.

7 Economic Model

7.1 Fee Commitment at Deployment

The most significant economic difference between this protocol and all alternatives is the moment at which execution cost is committed. In gas-based systems, the cost of execution is unknown at the time the automation is defined and is paid at the time it runs. This creates two failure modes: insufficient gas causing execution failure, and gas price spikes making execution prohibitively expensive at critical moments.

In this protocol, the execution fee is embedded in the UTXO at deployment. It is locked, irrevocably, until a valid trigger transaction claims it. The developer knows the exact execution cost at deployment time. There is no runtime cost uncertainty. There is no possibility of execution failure due to insufficient fee balance.

7.2 Miner Execution and the Overlay Enhancement

The protocol's liveness guarantee rests entirely on miners. A fee committed in a BOLT EventListener UTXO at deployment will be claimed by the miner who assembles the block that includes its trigger. That incentive cannot be switched off, censored by an external party, or exhausted over time. It persists on the UTXO set until the trigger fires.

Overlay nodes add a pre-block execution layer. Because the embedded fee is claimable by whoever constructs a valid `triggerTx` first, overlay operators are economically motivated to act before block assembly. The double-spend rule ensures only one can succeed. This creates a natural speed market without any protocol-level coordination mechanism. Miners who also run overlay infrastructure capture fees at both layers, creating an organic incentive for vertical integration.

Four fee layers operate: a *protocol fee* embedded at deployment; an *overlay fee* won by the first valid pre-block broadcast; a *miner inclusion fee* at block assembly (the unconditional base); and a *cascade hop fee* per downstream UTXO. When no overlay node acts, the miner claims both the overlay and inclusion fees combined.

7.3 Cascade Depth Pricing

A cascade of depth N involves N EventListener UTXOs each carrying embedded fees. Total execution cost scales linearly with cascade complexity:

$$\text{Total fee} = \sum_{i=0}^{N} \left(f_{\text{overlay},i} + f_{\text{miner},i} + f_{\text{protocol},i} \right)$$

This is not metering — it is a natural reflection of work performed. No gas limit estimation, no stack depth calculation, no artificial ceiling on expressiveness. Complexity costs proportionally because it requires proportional work.

Critically, while complexity scales linearly, the *certainty* of execution remains absolute. Because every fee is committed into the UTXO graph at deployment, there is no runtime market to clear and no price to discover at execution time. In gas-based systems a mid-cascade price spike can render subsequent hops economically inviable, leaving the automation stranded in a partially executed state. Here the fee

Table 2: Property comparison across platforms. Green = satisfied natively; Red = not satisfied / exogenous dependency; Orange = partial.

Property	ETH	SOL	ICP	L2	Bitcoin BOLT + EA†
On-chain listener	No	No	Part.	No	Yes
State-triggered	No	No	No	No	Yes
External server required	Yes	Yes	No	Yes	No
Execution uniqueness	Reg.	Bot	Queue	Reg.	Double-spend
Ancestor/depth limit	Yes	Yes	No	Yes	No (SPV)
Settlement = trigger	No	No	No	No	Yes
Causal chain on-ledger	Part.	No	No	Part.	Yes
Fee committed at deploy	No	No	No	No	Yes
Survives server down	No	No	Yes	No	Yes
Liveness guarantee	SLA	Bot	Cycles	SLA	UTXO persists
Atomic multi-party settle	No	No	No	No	Yes
Protocol upgrade safe	No	No	No	No	Yes

for every hop is already on-chain, irrecoverable until the trigger fires; no market condition can revoke the miner's incentive after deployment.

An additional optional `fundingOutpoint` can be co-spent in the `triggerTx` to inject supplemental satoshis at execution time, should the embedded fee prove insufficient to attract timely inclusion under exceptional fee-market conditions. This provides a deadlock-prevention escape hatch without weakening the core deployment-time commitment guarantee: the funding input is enforced by the locking script only when its outpoint is non-null, so the normal (no supplemental funding) path remains unchanged.

8 Teranode and Unbounded Cascade Depth

Prior UTXO-based implementations imposed a 25-ancestor mempool limit, forcing a block confirmation every 25 hops — roughly 40 minutes for a depth-100 cascade. Teranode eliminates this through SPV proof attestation. Each transaction carries a Merkle proof of input validity; nodes verify the proof without traversing the ancestor chain. The limit existed because validation required traversal; SPV proofs remove that requirement and therefore the limit. A depth-100 cascade settles in one block. Teranode's Merkle subtree propagation further allows prevalidation during the block assembly window, making single-block confirmation of arbitrarily deep cascades the expected case.

In terms of base throughput, Teranode in partnership with Aerospike has demonstrated sustained performance of over one million global BSV blockchain transactions per second — equivalent to 100 billion transactions per day — in live distributed trials [7, 8]. As of this writing, a two-node Teranode test at one billion transactions per second is actively under way, with reported infrastructure costs of $629,469.95 USD per 24-hour test period [9]. The architecture scales horizontally with commodity hardware; the practical ceiling is set by the global data centre industry, not by any protocol constraint.

The one remaining external dependency at global scale is ISP implementation of IPv6. True peer-to-peer transaction propagation — between any two nodes or devices on the network without an intermediary — requires globally routable IP addresses. The IPv4 address space was exhausted in February 2011 [10]; networking over IPv4 now universally depends on Network Address Translation (NAT), which modifies packet headers in transit and prevents direct end-to-end peer-to-peer communication. At billion-TPS scale, routing transactions through NAT introduces latency and creates propagation bottlenecks that are absent from the BSV protocol itself. IPv6, with its 2^{128}-address space, restores the original end-to-end Internet model: every device holds a globally routable address, peers communicate directly, and no translation layer interposes between sender and recipient. The BSV Blockchain is designed to integrate natively with IPv6 as the transport layer for peer-to-peer transaction routing, and Teranode's architecture explicitly anticipates this integration [11]. The protocol constraint on scale is therefore zero; the remaining

constraint is the pace at which ISPs worldwide complete their IPv6 rollout.

9 Application Domains

Four application classes illustrate the protocol's range.

NFT access token lifecycles. A single token presentation triggers cascades that activate credentials, decrement inventory, route royalties, and schedule post-event token transformation, all atomically from one gate event.

Multi-party royalty distribution. A single payment cascades splits to all rights holders simultaneously. Rights graph changes propagate to all future payment events automatically; audit trails are the transaction graph.

Parametric financial instruments. Any instrument whose payout condition is a measurable event — weather data, flight delays, commodity thresholds — is encoded as an EventListener UTXO watching for a signed oracle transaction. Trigger-to-settlement latency is seconds; no claims process or adjuster is required.

Autonomous treasury cascades. Revenue flows execute tax allocations, vendor payments, reserve contributions, and profit distributions without human intervention. Every allocation is traceable to its initiating revenue event; compliance is architectural.

10 Comparative Advantages and Structural Properties

The question of which infrastructure can support global-scale automation is not a theoretical one. It is answered by the throughput figures that exist on the ledger and in active test today, and by the structural property analysis presented in Section 6.

This architecture satisfies all four properties required for machine-scale economic automation simultaneously. To be precise: (1) base-layer throughput sufficient for machine-scale economic activity has been demonstrated in live distributed testing [7, 8, 9]; (2) automation triggers are enforced by the same miner validation that secures every satoshi; (3) fees are committed at deployment with no runtime uncertainty; and (4) execution uniqueness is guaranteed by double-spend prevention rather than a registry, queue, or coordination layer.

Each of these properties is a structural consequence of the underlying architecture, not a configurable feature. Platforms that satisfy fewer than all four are not slower or more expensive versions of the same thing — they are architecturally distinct systems. Platforms built on account models cannot provide UTXO-native execution uniqueness without adding a coordination layer. Platforms without miner-enforced token validity cannot provide on-chain EventListener logic. Platforms with gas-based fee models cannot provide deployment-time fee commitment. These are not performance gaps that can be closed through optimisation; they are design incompatibilities that would require replacing foundational architectural decisions to resolve.

The cascade cannot be recreated on another chain by adding a plugin. It requires unbounded base-layer throughput, UTXO-native token validity enforced by miners, and irreversible double-spend prevention — three properties that co-exist only in this stack.

11 Limitations and Open Questions

We identify the following limitations and areas for further work.

Fee-bounded cascade depth. Cascade depth is unbounded in protocol but fee-bounded in practice. A cascade of depth N requires N embedded EventListener fees paid at deployment. At current BSV fee levels, this is economically tractable for deep cascades, but the practical depth limit under real fee market conditions warrants empirical study.

Overlay node market equilibrium. The two-tier fee market described in Section 6 is characterised informally. A formal game-theoretic treatment of the overlay node competition — in particular, its equilibrium properties under varying miner and overlay fee configurations — is left for future work.

Teranode deployment dependency. The unbounded cascade depth claim is contingent on Teranode adoption by the mining network. Under the legacy node architecture with a 25-ancestor mempool limit, deep cascades require inter-block coordination. The protocol is complete under either architecture; the depth guarantee is specific to Teranode.

ISP IPv6 adoption. The one constraint on full-scale peer-to-peer transaction propagation that lies entirely outside the protocol is ISP implementation of IPv6. Under IPv4 with NAT, direct end-to-end

routing between arbitrary peers is not possible without traversal mechanisms that add latency. This is not a BSV or Teranode limitation — the protocol is designed for IPv6 native operation — but it is the single remaining external dependency on the path to fully unconstrained global-scale deployment. As IPv6 adoption among ISPs continues, this constraint diminishes autonomously [10, 12].

Formal security analysis. The execution uniqueness argument is presented informally, relying on the well-established double-spend prevention properties of Nakamoto consensus. A formal security reduction to the underlying consensus security assumption is a natural extension of this work.

12 Conclusion

We have shown that interrupt-driven cascade automation is not a feature that must be built into a blockchain — it is a property that emerges from the correct composition of Bitcoin's existing primitives. BOLT provides miner-validated, UTXO-native tokens with no secondary ledger or virtual machine. The UTXO spend model provides execution uniqueness. Embedded fees give miners a direct, persistent incentive to execute trigger transactions at block assembly time. Teranode's SPV architecture removes depth limits. The causal record is the transaction graph.

The system is operationally complete with miners alone. Overlay nodes that watch the mempool and race to broadcast trigger transactions before block assembly are a beneficial addition — they drive execution latency toward the pre-block minimum and create a healthy speed market — but the protocol does not depend on them. No overlay network need exist for every cascade to execute correctly and with finality.

Every alternative approach requires at least one of the following: a trusted off-chain watcher, a funded keeper registry, a developer-maintained execution balance, a time-polling mechanism that approximates but does not achieve state-triggered reactivity, or a sequencer that can be censored. This protocol requires none of them.

The properties that make this protocol distinctive were not designed into Bitcoin for this purpose. They were always there. BOLT composes them.

The cascade cannot die. It can only wait.

A Glossary

BOLT (Bitcoin Original Layer-1 Token Protocol). A token protocol in which tokens are defined as UTXO-native constructs whose validity is enforced entirely by Bitcoin's existing script engine. No secondary ledger, side chain, or virtual machine is required. BOLT token transactions are validated by miners identically to all other Bitcoin transactions.

EventListener UTXO. A Bitcoin output whose locking script takes two deployment-time arguments (`issuerPubKey`, optional `fundingOutpoint`) and fully validates a raw BOLT settle transaction on-chain when spent. Validates all 42 token input fields including the locking script suffix hash, enforces issuer identity in the no-bolt (2-input) case, checks that a valid BOLT token output continues to exist, verifies the change output is locked to a correctly configured EventTrigger UTXO, and enforces co-spend of both UTXOs via a `hashPrevouts` equality check. Carries embedded satoshis as the irrecoverable execution fee.

EventTrigger UTXO. A Bitcoin output holding the settle change created by a BOLT settle transaction. Its locking script bakes in the paired EventListener's outpoint (`outpointTrigger`) and verifies `hash256(outpointTrigger‖ctx.outpoint)` = `hashPrevouts`, enforcing that both UTXOs are spent together in the same `triggerTx`. Optionally verifies the `triggerTx` output set via `hash256(changeOutput)` = `hashOutputs`.

triggerTx. The transaction constructed by a miner or overlay node in response to a trigger event. Spends the EventListener UTXO, creating downstream cascade UTXOs and fee-capturing outputs.

Overlay node. Specialised infrastructure that watches the mempool for trigger conditions and broadcasts `triggerTx` transactions before block assembly. Overlay nodes provide a pre-block speed tier and compete on latency, but are not required for protocol correctness or liveness.

Cascade. A directed acyclic graph of EventListener UTXOs and `triggerTx` transactions in which spending one UTXO produces inputs or tokens that trigger one or more downstream UTXOs.

SPV proof. A Merkle proof attesting to transaction input validity without requiring ancestor chain traversal. Enables unbounded cascade depth under Teranode architecture.

Emergent automation. Autonomous cascade behaviour arising from the native composition of UTXO mechanics, fee incentives, and double-spend enforcement — not from virtual machine execution, but mimics such.

Two-tier market. The competitive structure in which both overlay nodes and miners are independently incentivised to construct trigger transactions, providing speed competition and unconditional liveness.

Nakamoto consensus. The proof-of-work consensus mechanism described in the original Bitcoin whitepaper, in which the chain with the greatest accumulated work is accepted as valid by all honest nodes. Provides the double-spend prevention guarantee on which execution uniqueness in this protocol rests.

Simplified Payment Verification (SPV). The verification method described in Section 8 of the Bitcoin whitepaper, in which a transaction's inclusion in a block is confirmed via a Merkle path proof rather than full node validation. Teranode extends SPV principles to mempool propagation, enabling unbounded-depth cascade validation without ancestor chain traversal.

IPv6. The sixth version of the Internet Protocol, featuring a 2^{128}-address space that assigns a globally unique, routable address to every device. IPv6 restores the original end-to-end peer-to-peer model of the Internet that was broken by the exhaustion of IPv4 addresses in 2011 and the subsequent universal deployment of Network Address Translation (NAT). Full ISP implementation of IPv6 is the single remaining external dependency for unconstrained peer-to-peer transaction routing at global scale on the BSV network.

References

[1] S. Nakamoto and S.-C. Nakamoto. BOLT: A Bitcoin Transaction Latching Mechanism Token Protocol. https://www.researchgate.net/publication/388000992_BOLT_A_Bitcoin_Transaction_Latching_Mechanism_Token_Protocol, 2025.

[2] B. Armstrong. Tweet, March 2026. https://x.com/brian_armstrong/status/2031021867973194172.

[3] D. Liebowitz. The DeFi Future is Automated. *The Defiant*, July 2021. https://thedefiant.io/the-defi-future-is-automated.

[4] Chainlink Labs. Chainlink Automation Documentation. https://docs.chain.link/chainlink-automation, 2024.

[5] DFINITY Foundation. Periodic Tasks and Timers — Internet Computer Developer Documentation. https://internetcomputer.org/docs/current/developer-docs/smart-contracts/advanced-features/periodic-tasks/, 2024.

[6] RockawayX. A Technical Assessment of Solana Protocol Opportunities and Current Progress. https://www.rockawayx.com/insights/a-technical-assessment-of-solana-protocol-opportunities-and-current-progress, 2024.

[7] Aerospike, Inc. BSV Association and Aerospike Achieve 100 Billion BSV Blockchain Transactions Per Day on Teranode. Press release, 30 July 2024. https://www.prnewswire.com/news-releases/bsv-association-and-aerospike-achieve-100-billion-bsv-blockchain-transactions-per-day-on-teranode-html.

[8] BSV Association. Teranode: Powered by Aerospike — Case Study. https://bsvassociation.org/publication/case-study/teranode-powered-by-aerospike/, 2025.

[9] S. Tominaga (aka Dr. Craig Wright). Tweet, 9 March 2026. https://x.com/CsTominaga/status/2030917121367232938. Archived: https://archive.is/gRlF5.

[10] BSV Blockchain Association. IPv6 Unlocks the True Power of the BSV Blockchain. Interview with Prof. Latif Ladid, Chair, IPv6 Forum. https://bsvblockchain.org/news/ipv6-unlocks-the-true-power-of-the-bsv-blockchain/, 2023.

[11] BSV Association. BSV Begins Technical Testing of Teranode: A Watershed Blockchain Upgrade. Press release, 22 February 2024. https://www.prnewswire.com/news-releases/bsv-begins-technical-testing-of-teranode-a-watershed-html.

[12] BSV Blockchain Association. P2P Blockchain: Blockchain Scaling with the IPv6-Based New Internet. https://bsvblockchain.org/p2p-blockchain-blockchain-scaling-with-the-ipv6-based 2024.