ISSN: 2096-3246
Volume 58, Issue 04, 2026

CASCADING FAILURES IN MULTI-AGENT AI SYSTEMS: A CAUSAL ATTRIBUTION AND REAL-TIME INTERVENTION FRAMEWORK

Krishna Sai Sevilimedu Veeravalli
Independent Researcher, USA

Abstract
Production deployments of multi-agent AI systems have exposed a failure class that modular design alone cannot prevent: cascading errors, where a degraded output from one agent propagates through downstream components and amplifies into system-wide instability. Existing orchestration frameworks resolve execution ordering reliably but offer no mechanism to govern the reliability of intermediate outputs, leaving a structural gap in failure governance that grows more consequential as pipeline complexity scales. This paper introduces the Causal Intervention Control Plane (CICP), a conceptual framework addressing cascading failures through three coordinated capabilities: a graph-based model of failure propagation, a causal attribution mechanism that assigns quantified Causal Responsibility Scores to agents, and a set of cost-aware intervention primitives that contain and correct failures without full pipeline disruption. An Intervention Cost Model and Failure Severity Score govern when interventions are applied, ensuring corrective actions remain within the latency budgets of real-time operating environments. The CICP establishes a theoretical foundation for control-oriented multi-agent AI architectures — systems that govern their own reliability rather than deferring failure management to external monitoring.
Keywords: Multi-agent AI systems; cascading failures; causal attribution; fault tolerance; real-time intervention; failure propagation graph; orchestration frameworks.

1. Introduction
Enterprise AI systems built on multi-agent pipelines now coordinate specialized components across tasks of considerable complexity: contract analysis, autonomous code review, real-time customer support, and orchestrated data transformation at scale. Frameworks such as LangChain, CrewAI, and LangGraph have lowered the barrier to constructing such pipelines considerably, enabling practitioners to compose modular agents without deep platform engineering expertise [7]. What these frameworks do not provide — by design, not by oversight — is any mechanism for reasoning about whether the outputs passing between agents are reliable.
The practical consequence becomes visible at scale. When one agent in a pipeline produces output that deviates from expected quality, subsequent agents receive that deviation as input and build upon it. A document summarizer that omits a critical clause feeds a contract analysis agent working from an incomplete representation; that agent's output then shapes a recommendation engine operating on faulty premises. None of these agents fail in isolation — each executes correctly given its inputs. The failure is distributed, emergent, and causally attributable to the original deviation, yet it manifests far downstream, presenting diagnostically as a general output quality problem rather than a localized agent error. Cemri et al. [7] analyzed more than 1,600 annotated execution traces across seven frameworks, documenting failure rates between 41% and 86.7%, with dominant failure mechanisms tied to inter-agent misalignment and specification failures rather than individual model capability limits.

Existing recovery approaches are blunt. Pipeline restarts discard valid partial results accumulated before the failure point. Full re-execution imposes latency costs incompatible with real-time service constraints. Manual trace inspection does not scale. The attribution problem itself is severe: state-of-the-art language models achieve below 10% accuracy when asked to identify the agent responsible for a failure within a multi-agent execution trace [11], underscoring how poorly the problem is currently understood even by the most capable diagnostic tools available.

Three compounding challenges define the design space for a principled solution. The causal structure of multi-agent failures is non-trivial to recover: a node exhibiting low-quality output may be a root cause or a downstream victim of an earlier deviation. Recovery mechanisms must be targeted — applying full-pipeline restarts to single-agent transient errors is wasteful and counter-productive.

Real-time constraints impose a hard bound on intervention latency; any correction mechanism adding more than a few hundred milliseconds of overhead per event is impractical in production conversational and decision-support contexts.

The Causal Intervention Control Plane (CICP) proposed in this paper addresses all three challenges. It models agent interaction as a directed graph where confidence scores decay or amplify through dependency edges, providing continuous failure-state visibility. It assigns Causal Responsibility Scores based on graph position, output deviation, and downstream influence, disambiguating root causes from secondary effects. Five intervention primitives of varying cost and scope, governed by a cost-aware decision policy, enable proportionate responses to failures of differing severity. The remainder of the paper presents the theoretical formulation of each component and their integration into a real-time control loop.

2. BACKGROUND AND RELATED WORK

Three bodies of prior work converge on the problem this framework addresses. Their convergence also reveals the structural gap the CICP fills.

2.1 Why Orchestration Frameworks Are Insufficient

The dominant multi-agent orchestration frameworks share a design premise: agent execution is the primary concern, and output quality is the agent's own responsibility. LangGraph adds explicit state management and conditional branching to LangChain's linear composition model. CrewAI introduces role-based agent specialization and task delegation. None of these extensions adds a layer for monitoring inter-agent output reliability or for intervening when reliability degrades. Cemri et al. [7] captured this gap empirically: their taxonomy of 14 failure modes, developed from rigorous analysis of 150 execution traces with inter-annotator agreement of $\kappa = 0.88$, shows that specification and inter-agent misalignment failures account for the majority of production breakdowns — neither class is addressed by execution-centric framework design.

Structural topology matters independently of model capability. Huang et al. [8] demonstrate that hierarchical multi-agent structures exhibit a performance drop of only 5.5% under faulty-agent conditions, compared to 10.5% for sequential structures and 23.7% for flat peer-to-peer configurations. Critically, an inspection mechanism that reviews inter-agent messages before propagation recovered up to 96.4% of errors caused by faulty agents — confirming that intermediate output review is a highly effective mitigation and directly motivating the CICP's FPG-based propagation model and intervention design.

2.2 Fault Tolerance Patterns in Distributed Systems

Decades of fault-tolerance engineering in distributed systems have produced patterns that, while not directly portable to AI pipelines, offer structural analogies the CICP adapts. The Erlang/OTP supervision tree [1] separates failure detection from execution: supervisors apply configurable restart strategies (one-for-one, rest-for-one, one-for-all) based on process dependency structure — the decoupling the CICP brings to multi-agent systems. Nygard's circuit breaker and bulkhead formalization [2] provides the failure containment model, preventing failures from crossing component boundaries through isolation and traffic control. Mohammad's systematic review [6] of 26 studies across the microservice resilience literature identifies circuit breaking, adaptive backpressure, and compensation-based recovery as the three most consistently effective resilience tactics — findings that inform the intervention primitive design in Section 4.3. The critical adaptation challenge is that these patterns assume stateless, deterministic service interactions, whereas agent outputs are probabilistic and semantically variable, requiring the confidence-weighted propagation model in the FPG rather than binary failure detection.

2.3 Causal Attribution in AI and Distributed Pipelines

Graph-based causal methods for distributed system diagnosis have matured significantly. Wang et al. [3] build incremental causal graphs from live telemetry for online root cause analysis, achieving scalable propagation of conditional failure risk as the graph evolves in real time. Wang et al. [4] apply knowledge graph-enhanced GCN methods to microservice root cause localization, reaching 93.5% top-3 accuracy at second-level response times — demonstrating that graph-structural attribution at operational speed is achievable. In the AI agent domain specifically, Zhang et al. [5] transform

ISSN: 2096-3246
Volume 58, Issue 04, 2026

execution logs into hierarchical causal graphs and use counterfactual backtracking to separate root causes from propagated symptoms. Bonagiri et al. [13] compute Causal Responsibility Scores at the step level through counterfactual intervention — an approach extended in the CICP to graph-level multi-agent attribution. Wang [12] reports that graph-based causal tracing achieves sub-second root cause localization in deployed multi-agent workflows, confirming that runtime attribution at production latency budgets is feasible.

Table 1: Comparison of Existing Frameworks Against CICP Capabilities

Capability	LangChain	CrewAI	LangGraph	AutoGen	CICP (Proposed)
Execution orchestration	Yes	Yes	Yes	Yes	Yes
Failure propagation modeling	No	No	Partial	No	Yes (FPG)
Agent-level causal attribution	No	No	No	No	Yes (CRS)
Targeted intervention primitives	No	No	No	No	Yes (5 primitives)
Cost-aware intervention policy	No	No	No	No	Yes (ICM + FSS)
Real-time control loop	No	No	No	No	Yes

3. PROBLEM FORMALIZATION

Representing a multi-agent system as a directed graph G = (V, E) makes the propagation structure explicit. Each node v ∈ V corresponds to an agent; each directed edge (u, v) ∈ E represents a dependency where agent v processes output from agent u.

The baseline confidence score c(v) ∈ [0, 1] captures the estimated reliability of an agent's local output. The dependency weight $w_{=u,v(}$ ∈ [0, 1] is defined as the semantic sensitivity factor. A high $w_{=u,v(}$ indicates that node v is highly dependent on the syntactic and semantic correctness of node u, whereas a low $w_{=u,v(}$ implies node v extracts only loose contextual signals from node u.

The propagated confidence for any node v is formally bounded by the minimum incoming confidence, scaled by the sensitivity weight:

$$\mathbf{c_{prop}(v) = min_{=}u \in pred(v)_{(} [\, w_{=}u,v_{(} \cdot c(u) \,]}$$

This formulation links directly to the Laplacian spectral properties of the dependency graph described by Liu et al. [9]. By treating $w_{=}u,v_{(}$ as a transmission coefficient, cascading failure risk can be simulated prospectively. Even when empirical derivation of $w_{=}u,v_{(}$ is challenging in live systems, assigning these weights theoretically allows the CICP to map the predicted boundary of a cascade before the next agent sequence executes — enabling proactive rather than purely reactive intervention.

Three conditions jointly define a cascading failure event. First, at least one agent v* exhibits confidence below a detection threshold θ^d, indicating anomalous output quality. Second, the confidence drop has affected at least one other agent that is two or more hops (transitively) from the originating faulty agent. Third, overall system performance, captured by the aggregate confidence across all agents, has fallen below a system-level threshold θ_s^{ys}. The Failure Propagation Graph (FPG) is the subgraph comprising all nodes that have exceeded threshold θ^d, together with their connecting edges. The FPG remains small while failures are localized but expands as the failure propagates, directly highlighting the causal relationships the attribution mechanism exploits.

ISSN: 2096-3246
Volume 58, Issue 04, 2026

Liu et al. [9] provide a formal risk quantification framework showing that cascading failure risk in multi-agent networks scales explicitly with the Laplacian spectral properties of the dependency graph and communication delay, confirming that topology-aware representations are essential for principled failure analysis. Zhang et al. [10] demonstrate that failure attribution based on information dependency graph structure rather than temporal execution order substantially improves attribution accuracy — a finding that validates the graph-structural basis of the FPG formulation.

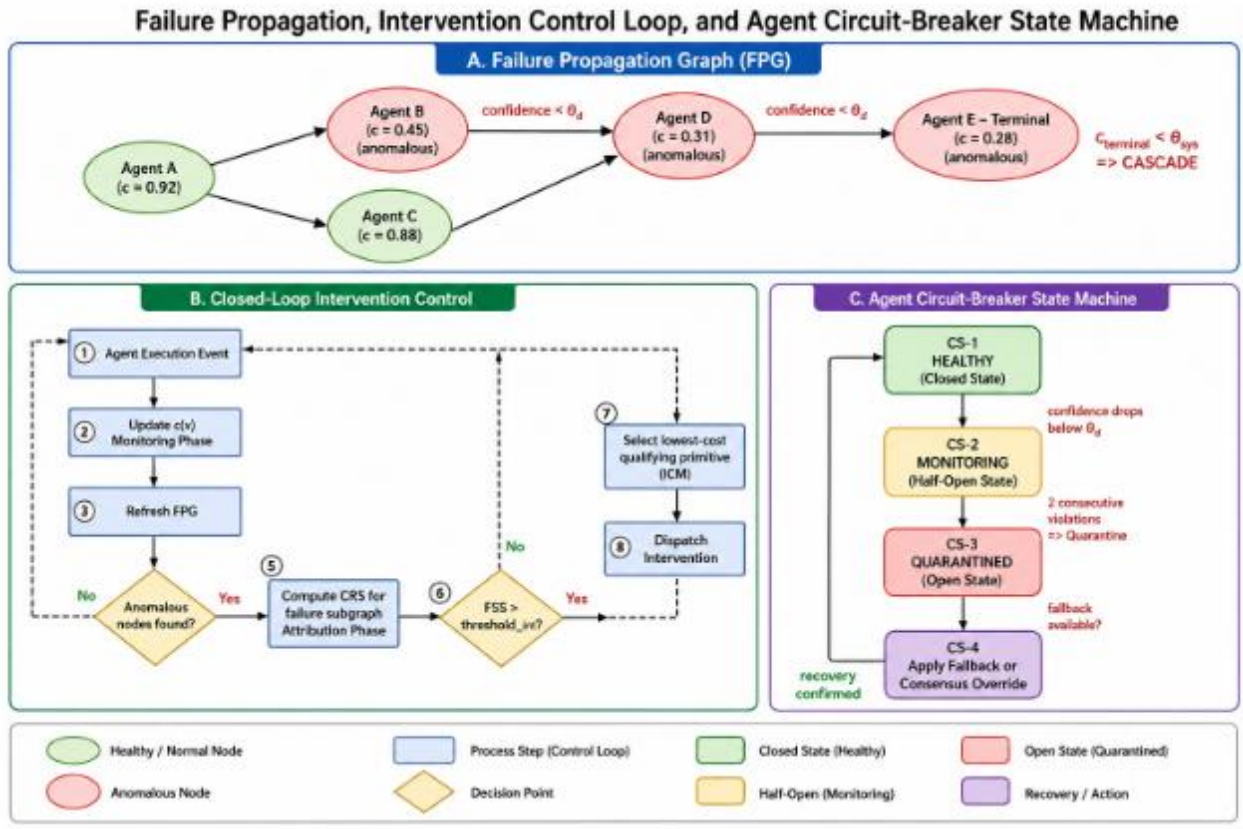

Figure 1: Failure Propagation Graph — five-node directed agent pipeline with per-node confidence scores c(v), edge dependency weights w(u,v), highlighted failure subgraph (nodes below θ^d), and cascade propagation paths.

4. THE CAUSAL INTERVENTION CONTROL PLANE (CICP)

The CICP operates as a side-channel control layer. It observes agent execution events, maintains the FPG, performs attribution, and dispatches interventions through a defined interface — without modifying agent logic or execution sequencing. Five components compose the framework.

4.1 Failure Propagation Graph (FPG)

On each agent execution event, the FPG is updated: the confidence score c(v) for the executing node is recomputed, propagated confidence values for all downstream dependents are updated, and the failure subgraph is re-derived. The resulting structure provides a continuously accurate representation of where reliability is degrading and how degradation is spreading through the pipeline.

Two topological regimes produce qualitatively different propagation behavior. Di Gioia [14] establishes that tree-structured execution graphs exhibit exponential cascade growth per hop, while dense cyclic structures exhibit self-limiting behavior within approximately one hop; cascade-aware routing informed by this geometric distinction recovers up to 87.2% of overall system performance — a gain of approximately 36.8 percentage points over topology-agnostic approaches. The FPG model is topology-sensitive by construction, enabling different intervention strategies for tree-like versus cyclic pipeline segments. Luo et al. [15] demonstrate that Ollivier-Ricci curvature of the agent communication graph encodes early-warning signals of cascade risk before full failure development, suggesting that geometric graph properties can augment the FPG's anomaly detection pass with proactive rather than purely reactive triggering.

4.2 Causal Responsibility Score (CRS)

Attribution in a cascading failure scenario requires distinguishing agents that originated degradation from those that merely propagated it. The CRS addresses this through three weighted factors: graph position (topological proximity to terminal failure nodes, computed as the inverse of path length from v to the set of terminal degraded nodes), output deviation magnitude (absolute difference between the current confidence score and the agent's historical baseline), and downstream influence (cumulative confidence degradation attributable to this agent across all reachable dependents in the FPG).

$$\mathbf{CRS(v) = \alpha \cdot position(v) + \beta \cdot deviation(v) + \gamma \cdot influence(v), \quad where\ \alpha + \beta + \gamma = 1}$$

The agent with the highest CRS is identified as the primary causal agent. A node with moderate deviation but high downstream reach may outscore a node with severe local deviation and no downstream impact — preventing the common diagnostic error of attributing failures to the most visibly degraded node rather than the most causally influential one. Current state-of-the-art LLM-based attribution tools achieve below 10% accuracy on multi-agent failure attribution benchmarks [11]; the CRS formulation provides a structured alternative grounded in graph topology rather than language model judgment.

4.3 Intervention Primitives

Selective Re-execution retries the causal agent with its original inputs, optionally with a modified prompt that increases output specificity. Applied when CRS is high, deviation appears transient, and the agent's historical success rate supports retry. Computational cost: medium (one additional agent execution cycle).

Propagation Blocking prevents transmission of output from the anomalous node to downstream dependents before that output is consumed. Applied when the anomaly cannot be immediately corrected or when downstream agents have not yet ingested the faulty output. Computational cost: low (graph state update only).

Agent Quarantine removes the causal agent from the active pipeline and replaces it with a null or stub output. Applied when the agent maintains low confidence across two or more consecutive runs, indicating a persistent rather than transient failure. Computational cost: low.

Fallback Substitution replaces a quarantined agent with a lower-capability, higher-reliability alternative when the agent's functionality is essential to pipeline completion and an alternative exists. Computational cost: low to medium (overhead of the fallback invocation).

Consensus Override executes the critical step in parallel using multiple independent agents and aggregates results via consensus. Applied when no reliable fallback exists and the step is non-optional. Computational cost: high (N-1 additional execution cycles plus aggregation).

4.3.1 State Management During Interventions

A critical divergence between classic microservice fault tolerance and multi-agent AI systems lies in state management. While microservices are commonly designed to be stateless and deterministic, AI agents in conversational or reasoning pipelines maintain probabilistic memory arrays — historical message lists, scratchpads, or retrieval buffers — that accumulate context across execution steps.

To maintain the integrity of the intervention primitives, the CICP must orchestrate Contextual State Rollbacks. When Selective Re-execution or Fallback Substitution is triggered, simply re-running the agent is insufficient. The CICP must issue a state-truncation command that removes the contaminated upstream output from the downstream agent's context window before the intervention primitive is applied. Formally:

$$\mathbf{S_v(t+1) = S_v(t) \setminus O_u^{fau3ty}}$$

where $S_v(t)$ is the memory state of the downstream agent at time t, and O_u^{fau3ty} is the specific contaminated output from the upstream causal agent. By explicitly pruning contaminated tokens from downstream agents' context windows before applying any intervention primitive, the CICP ensures that the cascading anomaly is completely flushed from the system's latent memory. This operation

ISSN: 2096-3246
Volume 58, Issue 04, 2026

mirrors the isolation boundaries enforced by a classic distributed circuit breaker and is essential for preventing re-contamination on the subsequent execution cycle.

Table 2: Intervention Primitives — Trigger Conditions and Cost Class

Primitive	Trigger Condition	Cost Class	Failure Type Addressed
Selective Re-execution	High CRS, transient deviation, retryable	Medium	Transient output error
Propagation Blocking	Anomalous node, downstream not yet consumed	Low	Imminent cascade spread
Agent Quarantine	Persistent low confidence (≥2 executions)	Low	Systemic agent failure
Fallback Substitution	Quarantined agent, fallback available, critical step	Low-Med	Critical path degradation
Consensus Override	No reliable fallback, step critical	High	Unresolvable single-agent failure

4.4 Cost-Aware Decision Layer

Intervention should be proportional to failure severity. The Failure Severity Score $FSS(v) = CRS(v) \cdot propagation\_depth(v) \cdot (1 - c_{terminal})$ captures urgency: high CRS, deep cascade propagation, and degraded terminal output jointly elevate severity. The Intervention Cost Model (ICM) assigns a latency and resource cost to each primitive based on computational overhead and downstream disruption scope. The selection policy is:

$$\mathbf{i^* = \arg\min_i ICM(i) \quad subject\ to \quad FSS > \theta^{I_n^t}}$$

This ensures that low-severity anomalies do not trigger expensive interventions while high-severity failures receive adequate response even when intervention costs are elevated. The ICM parameterization is deployment-specific: a real-time conversational system with a 200ms response budget assigns different cost weights than a batch analytics pipeline with a 30-second tolerance.

4.5 Theoretical Approaches to Real-Time Confidence Estimation

A foundational assumption of the CICP architecture is the availability of the confidence score c(v) for each executing node. In deployments relying on secondary large language models as evaluators, the additional inference cost introduces latency that can violate real-time operational constraints. To address this cold-start confidence estimation problem without external model overhead, the CICP is designed to accommodate intrinsic, mathematically derived proxies for output reliability.

Logit-Entropy Bounding: The confidence c(v) can be derived from the mean predictive entropy of the token distribution generated by the agent. High entropy across critical semantic tokens correlates strongly with hallucination or inter-agent misalignment, triggering a sub-threshold c(v) value without any external evaluation call. Formally, agents whose output token distributions exhibit high Shannon entropy over semantically load-bearing positions are assigned reduced confidence scores, allowing the FPG to flag potential cascade sources before downstream consumption.

Semantic Consistency Latent Projection: For agents utilizing Retrieval-Augmented Generation (RAG), c(v) can be approximated by calculating the cosine distance between the latent vector of the agent's generated output and the latent vector of the retrieved source context. A large angular distance between generation and retrieval embeddings indicates semantic drift — a condition correlated with factual inaccuracy and misalignment propagation risk.

While exhaustive empirical validation of these proxy mechanisms across varied pipeline architectures remains a direction for future work, bounding c(v) using information-theoretic metrics ensures the CICP maintains its $O(|V_s^{ag}|)$ real-time monitoring budget without incurring the latency overhead of

ISSN: 2096-3246
Volume 58, Issue 04, 2026

external model-based evaluation. Both proxies are computable within the agent's existing inference pass, making them consistent with the real-time operational constraints established in Section 5.

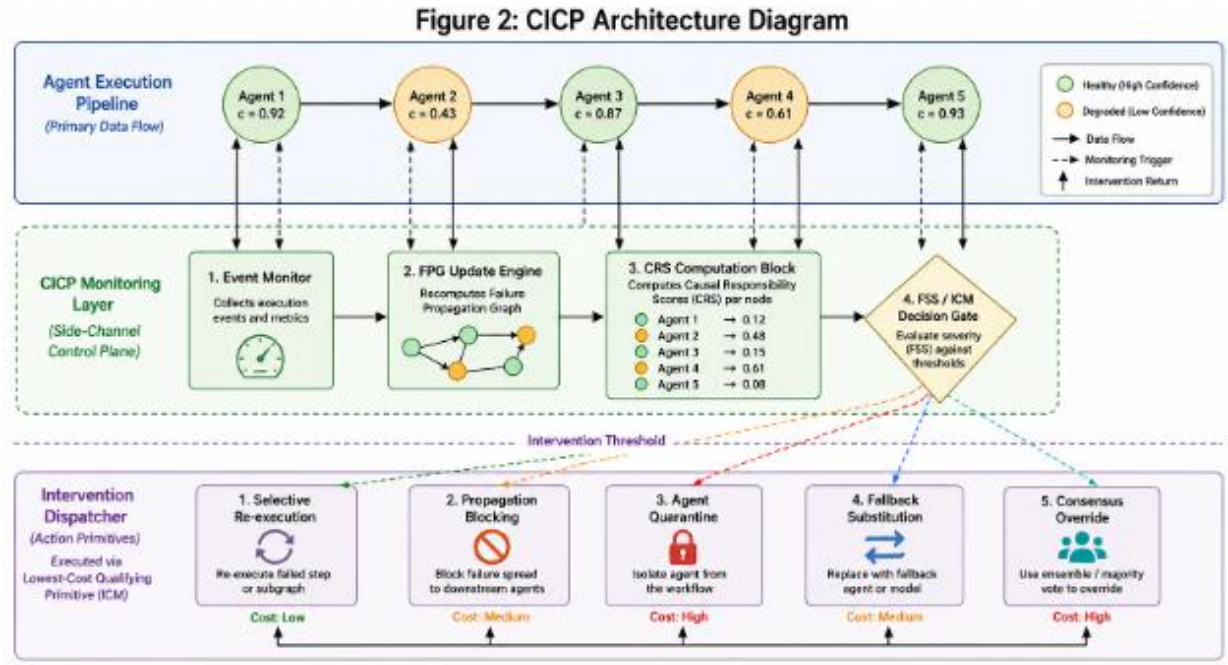

Figure 2: CICP Architecture Diagram — monitoring layer receiving agent execution events, FPG update cycle, CRS computation block, FSS/ICM decision gate, and intervention dispatcher routing to five primitive handlers.

5. CONTROL LOOP INTEGRATION AND REAL-TIME CONSTRAINTS

The five CICP components integrate into a continuous monitoring-attribution-intervention loop that operates concurrently with the execution pipeline. Four phases execute on each agent completion event. **Monitoring:** the agent's confidence score c(v) is updated and the FPG is refreshed. **Detection:** the FPG is scanned for nodes crossing θ^d and new propagation paths are identified. **Attribution:** if a failure subgraph exists, CRS values are calculated for all nodes within it. **Decision and Dispatch:** FSS is assessed; if it surpasses the intervention threshold, the lowest-cost qualifying primitive is selected and dispatched.

The computational profile of this loop is favorable for real-time deployment. FPG update is O(degree(v)) per event — bounded by local graph structure. Detection is $O(|V_s^{ag}|)$ — bounded by failure subgraph size. CRS computation is $O(|V_s^{ag}| + |E_s^{ag}|)$ — a graph traversal over the failure subgraph. Decision is O(|P|) = O(5). Total overhead scales with failure subgraph size, not total pipeline size, preserving performance when failures are localized — the common case in well-designed pipelines. Zhang et al. [11] report performance gains of approximately 4.8–14.2% in MetaGPT and MaAS from targeted feedback-driven agent correction, corroborating that agent-level intervention yields measurable system-level improvement within real-time operating constraints.

The Erlang/OTP supervision tree model [1] offers the clearest architectural precedent for the CICP's design philosophy. Supervisor processes apply topology-aware restart strategies without modifying worker execution logic — the exact separation the CICP implements for agent pipelines. The circuit breaker's three-state model [2] maps directly to agent states in the CICP: a healthy agent is in closed state, an anomalous agent under monitoring is in half-open state, and a quarantined agent is in open state. The CICP's intervention selection policy implements an analogous topology-aware recovery strategy, choosing primitives whose operational scope matches the dependency reach of the causal agent. Lovén et al. [16] demonstrate that hierarchical dependency graph structures produce stable resource allocation convergence in real-time AI service contexts, confirming that the structural properties making CRS computation tractable in tree-structured pipelines also support broader runtime governance.

ISSN: 2096-3246
Volume 58, Issue 04, 2026

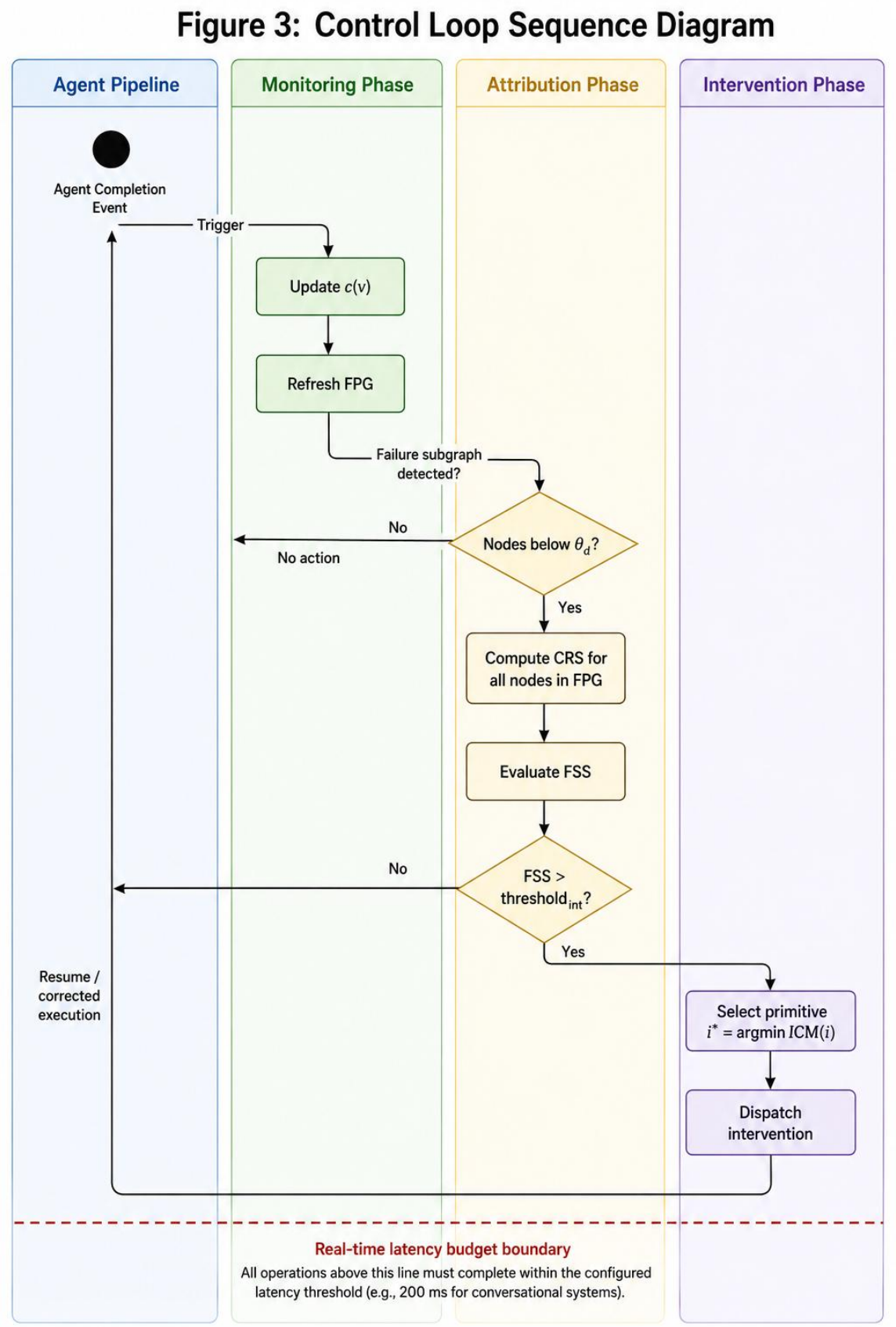

Figure 3: Control Loop Sequence Diagram — event timeline showing agent completion trigger, monitoring phase, FPG update, detection pass, conditional attribution branch, FSS evaluation, primitive selection, and dispatch to intervention handler.

6. DISCUSSION

The CICP closes a specific architectural gap: the absence of failure governance between agents. Execution frameworks handle what agents do and when; the CICP governs whether what they produce is reliable and what to do when it is not. This separation mirrors the separation between application logic and infrastructure monitoring in mature software systems — a design principle familiar to practitioners who have built on Erlang/OTP supervision trees or designed microservice resilience layers at enterprise scale. The CICP does not replace orchestration frameworks; it extends them with a control layer that can be instantiated as a side-channel observer without modifying underlying agent logic.

Practical implementation surfaces three non-trivial requirements. Confidence score generation is the most pressing: most current orchestration frameworks do not annotate agent outputs with reliability estimates natively. The theoretical proxies described in Section 4.5 — logit-entropy bounding and semantic consistency latent projection — offer deployable starting points that avoid external evaluation overhead, but integrating confidence scoring as a first-class pipeline concern remains a prerequisite for FPG operation. Graph state persistence requires a store that supports read and write at execution frequency; for long-running pipelines with many agents, this store grows proportionally to pipeline complexity. CRS computation over large failure subgraphs, while theoretically tractable at $O(|V_s^{ag}| + |E_s^{ag}|)$, may impose non-trivial latency in pathological cases where failures are widespread rather than localized.

Two explicit scope limitations bound the current framework. The CICP assumes observable agent outputs with estimable confidence — an assumption that fails for agents whose outputs are opaque, binary, or require domain-specific evaluation to assess. It also assumes a known or inferrable pipeline topology, which does not hold for dynamically composed systems where the agent graph is constructed or modified at runtime. Pham et al. [17] caution that causal inference accuracy in microservice contexts degrades when graph assumptions diverge from actual system dynamics — a limitation that applies equally to the FPG-based attribution model and that users should validate against their specific pipeline structures. Joshua [18] demonstrates that chaos engineering methods can proactively expose failure modes in LLM-based multi-agent systems under simulated fault conditions before production deployment, suggesting that systematic resilience testing should accompany any CICP deployment.

7. CONCLUSION

Multi-agent AI systems have reached a level of deployment complexity where execution-centric design is no longer sufficient. Cascading failures are not random malfunctions — they are structural consequences of compositional pipeline architectures that lack failure governance. The empirical record confirms this: failure rates between 41% and 86.7% across production frameworks are not attributable to model capability limits but to the absence of any layer for governing inter-agent output reliability. As long as orchestration frameworks treat agent outputs as trusted inputs by default, cascading failures will remain a predictable consequence of scale.

The Causal Intervention Control Plane provides the foundational components of a control layer for multi-agent AI systems: a Failure Propagation Graph that models how degradation spreads through agent dependencies, including a formally specified confidence propagation equation bounded by edge sensitivity weights; a Causal Responsibility Score that quantifies which agents are causally responsible for observed failures; five intervention primitives — now augmented with explicit Contextual State Rollback semantics — that address distinct failure classes at varying cost; theoretical real-time confidence estimation mechanisms that eliminate cold-start dependency on external evaluators; and a cost-aware decision policy that ensures corrective actions are proportionate to failure severity and remain within real-time operating constraints. Together, these components enable multi-agent systems to detect failures as they emerge, attribute them to specific sources before full cascade development, and apply targeted corrective action without full pipeline disruption.

Empirical validation on production pipelines constitutes the most important direction for future work: measuring attribution accuracy, intervention latency, and recovery effectiveness against baselines in real enterprise deployment contexts will test the formal assumptions underlying the FPG, CRS, and ICM formulations. The second priority is extending the FPG model to dynamic graph topologies where agent composition changes at runtime, requiring incremental graph update mechanisms and attribution methods robust to structural change. Empirical validation of the logit-entropy and semantic consistency confidence proxies across diverse agent architectures represents a third distinct research direction opened by the formalization in this paper. These directions will determine whether the CICP moves from theoretical foundation to deployed infrastructure.

REFERENCES

1. F. Cesarini and S. Vinoski, Designing for Scalability with Erlang/OTP (O'Reilly, 2016). Available: https://dl.acm.org/doi/10.5555/3055893
2. M. T. Nygard, Release It! 2nd ed. (Pragmatic Bookshelf, 2018). Available: https://pragprog.com/titles/mnee2/
3. D. Wang, Z. Chen, Y. Fu, Y. Liu, and H. Chen, "Incremental Causal Graph Learning for Online Root Cause Analysis," ACM SIGKDD 2023, pp. 2269-2278. https://doi.org/10.1145/3580305.3599392
4. T. Wang, G. Qi, and T. Wu, "KGroot: A Knowledge Graph-Enhanced Method for Root Cause Analysis," Expert Systems with Applications, vol. 255, p. 124679, 2024. https://doi.org/10.1016/j.eswa.2024.124679
5. Y. Wang, W. Wu, J. Wang, and Q. Wang, "From Flat Logs to Causal Graphs: Hierarchical Failure Attribution for LLM-based Multi-Agent Systems," arXiv:2602.23701, 2026.
6. M. Mohammad, "Resilient Microservices: A Systematic Review of Recovery Patterns, Strategies, and Evaluation Frameworks," arXiv:2512.16959, 2025.
7. M. Cemri et al., "Why Do Multi-Agent LLM Systems Fail?" arXiv:2503.13657, 2025.
8. J. Huang et al., "On the Resilience of LLM-Based Multi-Agent Collaboration with Faulty Agents," in Proc. 42nd ICML, 2025.
9. G. Liu, V. Pandey, N. Motee, and C. Somarakis, "Incremental Risk Assessment for Cascading Failures in Large-Scale Multi-Agent Systems," arXiv:2604.06024, 2026.
10. H. Zhang et al., "GraphTracer: Graph-Guided Failure Tracing in LLM Agents," arXiv:2510.10581, 2025.
11. G. Zhang et al., "AgenTracer: Who Is Inducing Failure in the LLM Agentic Systems?" arXiv:2509.03312, 2025.
12. Z. G. Wang, "AgentTrace: Causal Graph Tracing for Root Cause Analysis in Deployed Multi-Agent Systems," arXiv:2603.14688, 2026.
13. A. Bonagiri et al., "CausalFlow: Causal Attribution and Counterfactual Repair for LLM Agent Failures," arXiv:2605.25338, 2026.
14. D. Di Gioia, "Cascade-Aware Multi-Agent Routing: Spatio-Temporal Sidecars and Geometry-Switching," arXiv:2603.17112, 2026.
15. Z. Luo et al., "Auditing Cascading Risks in Multi-Agent Systems via Semantic-Geometric Co-evolution," arXiv:2603.13325, ICLR 2026 Workshop.
16. L. Lovén et al., "Real-Time AI Service Economy: A Framework for Agentic Computing Across the Continuum," arXiv:2603.05614, 2026.
17. L. Pham, H. Ha, and H. Zhang, "Root Cause Analysis for Microservice System Based on Causal Inference: How Far Are We?" in Proc. 39th IEEE/ACM ASE, 2024.
18. O. S. Joshua, "Assessing and Enhancing the Robustness of LLM-based Multi-Agent Systems Through Chaos Engineering," arXiv:2505.03096, 2025.