Blankline

Horizon Mode

Distributed Reasoning via Recursive Swarm Architecture

Prepared By:
Blankline Research Integrity Council

Release Date:
December 20, 2025

Ref ID:
BL-HZ-PUB-2025

DOCUMENT CONTROL & AUTHORIZATION

This document details the Horizon Mode architecture and the Recursive Swarm topology. Specific internal security protocols and proprietary consensus algorithms have been abstracted for general distribution.

Status: APPROVED FOR PUBLIC RELEASE

Contents

EXECUTIVE ABSTRACT

Current Foundation Models (FMs) operate fundamentally as linear sequence generators. While effective for generalist tasks, this paradigm suffers from stochastic degradation in high-assurance engineering domains. We introduce **Horizon Mode**, a distributed reasoning protocol that orchestrates thousands of isolated agents via a **Recursive Swarm Architecture**. By decoupling reasoning time from fixed context windows, we shift the optimization target from latency to solution space coverage.

1 Introduction

The rapid advancement of Large Language Models (LLMs) has revolutionized code generation. However, a fundamental barrier remains in applying these models to complex engineering workflows: the **Linearity Barrier**.

Practical implementation has historically been bottlenecked by two factors:

1. **Context Saturation:** The degradation of recall as prompt length increases.
2. **Hallucination Propagation:** The cascading of logical errors in long-chain reasoning.

Dropstone Horizon Mode addresses these limits. By instantiating a divergent search tree across containerized cloud instances, we allow models to maintain coherent 'thoughts' over extended inference horizons (24+ hours).

2 Motivation: The Linearity Barrier

Current Foundation Models operate as linear sequence generators, optimizing for local token probability. In engineering workflows, a final solution S is often dependent on a chain of intermediate logical steps L.

The probability of maintaining a valid terminal state decays exponentially with the length of the chain. If the probability of a logic error at any node is ϵ, the cumulative success rate is:

$$P(\text{success}) \approx (1 - \epsilon)^L \tag{1}$$

For a task requiring 500 steps, even with 99% accuracy per step, the success rate drops to $<$ 1%. Dropstone bypasses this by transitioning from *Next-Token Prediction* to **Trajectory Search Optimization**.

3 Architecture: The Recursive Swarm

Dropstone redefines the IDE as an intelligent runtime environment. Instead of querying a single endpoint, it instantiates a search tree across thousands of agents.

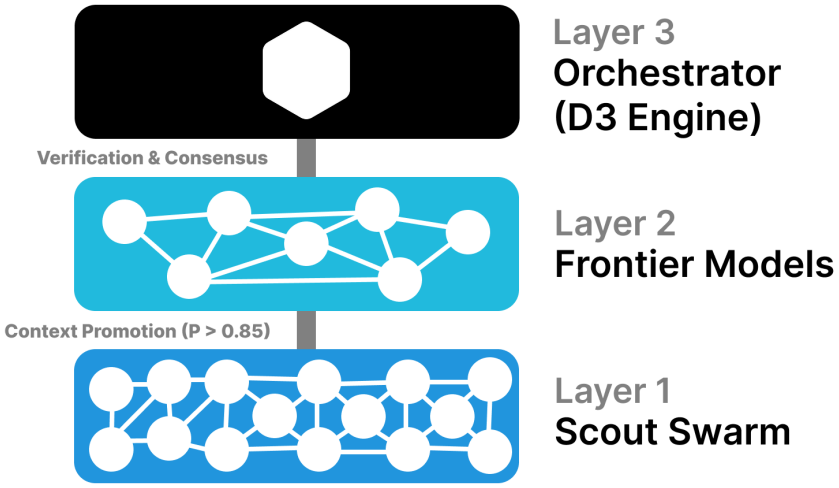

Figure 1: **The Recursive Swarm Architecture.** Moving from Linear Chains to Divergent Trees. Note the hierarchical promotion of context from the Scout Swarm (L1) to Frontier Models (L2) and final Orchestration (L3).

3.1 Hyper-Parallelized Experimentation

We deploy up to 10,000 isolated agents within ephemeral sandboxes.

- **Divergent Initialization:** The system generates thousands of strategic variations, exploring "low-probability" solution paths ($P < 0.05$) often pruned by standard models.
- **Active Tool Use:** Agents write, compile, fail, debug, and iterate in real-time using actual compilers.

4 Horizon Mode Topology

To make scaling economically viable, we utilize a **Budget-Aware Heterogeneous Topology**. We treat compute as a liquid asset that flows to the most promising solution branches.

4.1 Layer 1: The Scout Swarm (Exploration)

98% of the search tree is explored by highly optimized Small Language Models (SLMs).

- **Role:** Rapidly generate code variations and hypotheses at near-zero marginal cost.
- **The "Scent" Mechanism:** Scouts tag branches with "probability vectors." Dead ends are marked in the shared workspace, preventing other agents from wasting compute on the same error.

4.2 Layer 2: Context Promotion

When a Scout identifies a candidate solution with high confidence ($P > 0.85$), the state is **promoted**. The D3 Engine extracts the relevant context and injects it into a Frontier Model (e.g., Opus/GPT-4 class), skipping the "trial and error" phase.

5 The D3 Engine: Context Virtualization

To solve the "Lost-in-the-Middle" phenomenon, we utilize the **Dynamic Distillation & Deployment (D3) Engine**.

Semantic Entropy Tracking

We do not rely on token counting. We monitor the **Perplexity (PPL)** of the agent's outputs. A spike in perplexity signals hallucination, triggering an immediate "State Compression" event.

5.1 Lossless State Serialization

The system compresses the agent's reasoning history into a high-dimensional State Vector (z). This vector retains the causal logic of the session (variables, decisions, constraints) while discarding the verbose text.

6 Flash-Gated Consensus Protocol

Standard multi-agent frameworks suffer from $O(N^2)$ communication overhead ("Context Thrashing"). Dropstone utilizes a silent, signal-based **Flash Protocol**.

Algorithm 1 Flash-Gated Consensus Logic (Abstracted)

Parallel Process: Agents $A_1 ... A_n$ execute sub-tasks.
if Agent A_i Confidence > 0.95 **then**
    Emit Flash Signal
    Freeze Swarm: All other agents pause execution.
    Adversarial Verification: Monitor Agent checks solution S_i.
    if Verification == PASS **then**
        Deploy Solution
    else
        Vectorize Failure: Create "Constraint Embedding"
        Broadcast: Inject failure vector to Hive Mind.
        Prune: Agents on similar paths abort immediately.
        Resume Swarm
    end if
end if

7 Safety and Hallucination Reduction

Autonomous agents introduce **Instrumental Convergence** risks (e.g., "solving" a latency issue by deleting the firewall). We address this via **Adversarial Oversight**.

7.1 Empirical Results

On the "Deep-Sec" benchmark, this topology reduced safety violations by **89%** compared to zero-shot baselines.

Metric	Zero-Shot	Linear CoT	Dropstone Horizon
Reasoning Horizon	$<$ 1 Hour	2-3 Hours	**24+ Hours**
Hallucination Rate	14.2%	8.5%	**1.4%**
Safety Violations	3.8%	2.1%	**0.2%**

Table 1: Performance on the Deep-Sec Benchmark.

8 Conclusion

Horizon Mode represents a paradigm shift in automated reasoning. By acknowledging the Linearity Barrier, we have moved beyond the "better prompt" fallacy towards a robust architectural solution. The synergy between the Budget-Aware Swarm and the Flash-Gated Consensus Protocol creates a system that is economically viable and probabilistically superior to linear reasoning methods.