Stable by Design: A Control-Theoretic Account of AI-Driven Self-Healing Remediation Loops

Ajin Baby

cloudandsre.com

2026-09-01 · v0.3 · cloudandsre.com/research/stable-by-design-remediation-loops/

Abstract

"Self-healing" infrastructure—systems that detect degradation and remediate it autonomously—is increasingly driven by AI agents that decide *what* corrective action to take. The reliability conversation around these systems focuses on whether each individual decision is correct. We argue this misses the more dangerous failure class: a remediation system is a **closed feedback loop**, and feedback loops can be individually-correct yet *collectively unstable*—oscillating (scale up, then down, then up), overshooting (over-remediating), or running away (positive feedback amplifying the very problem it attacks). Control theory has well-developed tools for exactly these phenomena, and they have been applied to deterministic autoscalers, where hysteresis and cooling periods are standard. But AI-driven remediation introduces a controller that is **probabilistic, has variable and large dead-time (reasoning latency), and is frequently one of several agents acting on shared actuators**—conditions under which classical stability intuitions break. We provide the first control-theoretic account of AI-driven self-healing loops. We contribute (1) a model of the **Remediation Control Loop (RCL)** and a taxonomy of its instability modes (oscillation, overshoot, runaway, loop interaction); (2) stability-oriented design constructs—**hysteresis gates, action damping, settling-time cooldowns, dead-time-aware gating, and a loop-interaction matrix**—with informal stability conditions adapted to a probabilistic controller; and (3) a simulation study measuring oscillation, overshoot, settling time, and MTTR with and without these constructs. The study confirms the framework's three hypotheses: at fixed gain, growing dead-time alone flips a stable loop (1/20 runs oscillating, MTTR ≈ 15 steps) into rail-to-rail oscillation that never settles (20/20 runs at 4× the dead-time); hysteresis and damping alone do not repair a dead-time-driven instability, while the temporal constructs do—and **dead-time-aware gating restores stability at roughly one-third the MTTR cost of a fixed cooldown** (≈40 vs. ≈117 steps) because it adapts its hold to the controller's actual dead-time; and two individually-stable loops sharing an actuator destabilize each other until an actuator mutex held through the correction's settling window restores single-loop behavior. A fourth sweep maps the frontier itself: the settle/diverge boundary follows a **delay–gain product law** — the critical product $g \cdot L$ stays within 1.35–1.8 across a 6× gain range, where L is dead-time plus sensor delay — the deterministic boundary is one dead-time step wide, and controller stochasticity does not shift it so much as **blur it into an unreliable band in both directions**, motivating an explicit engineering margin ($g \cdot L \leq 1.2$) at which every sampled configuration settled at every stochasticity level. The thesis: self-healing systems must be **stable by design**, not merely correct per decision, and control theory is the right lens—suitably adapted to an LLM-in-the-loop controller.

Keywords: self-healing infrastructure, control theory, feedback stability, autonomous remediation, AIOps, oscillation, hysteresis, site reliability engineering.

---

1. Introduction

A self-healing system observes a symptom, decides on a remedy, and acts—then observes again. This is, structurally, a control loop, and the autonomous-operations literature has mostly examined the *decide* box: is the chosen remediation correct? But anyone who has operated an autoscaler knows the failure that bites hardest is not a single wrong decision; it is a *loop that will not settle*. A scaler that adds capacity, sees load drop, removes capacity, sees load rise, and adds capacity again—each decision locally reasonable, the aggregate behavior a destructive oscillation. Deterministic autoscaling learned this lesson and codified it: hysteresis, cooling periods, dead-bands are standard precisely to prevent flapping [1,2,3]. Control theory names and tames these effects—convergence, oscillation avoidance, overshoot mastery are its core concerns [5,10].

AI-driven self-healing reopens the problem in a harder form. The controller is now an LLM agent (or several), and three properties that classical control assumes are violated:

1. **The control law is probabilistic.** The same symptom can yield different actions across invocations; the controller is not a fixed transfer function, so its gain is not constant and its stability is not analytically obvious.
2. **Dead-time is large and variable.** Reasoning latency inserts a substantial, fluctuating delay between observation and action. Control theory is unambiguous that dead-time is destabilizing—it is the classic cause of oscillation—and AI controllers have *more* of it, and less predictably, than the rule-based controllers that already needed hysteresis.
3. **Multiple loops share actuators.** In agentic operations it is common to have several agents (or an agent and a legacy autoscaler) acting on overlapping resources, creating *interacting* control loops that can beat against one another even when each is individually stable—the multi-controller instability that the distributed-systems flapping literature documents for resources oscillating against each other [1].

No prior work, to our knowledge, analyzes the *stability* of AI-driven remediation loops as control systems. We provide that analysis. Contributions:

1. **The Remediation Control Loop (RCL) model and instability taxonomy** (§4): oscillation, overshoot, runaway, and loop interaction, each defined for a probabilistic controller with variable dead-time.
2. **Stability-oriented design constructs** (§5): hysteresis gates, action damping (rate-limited actuation), settling-time cooldowns, dead-time-aware gating, and a loop-interaction matrix for coordinating multiple agents—with informal stability conditions analogous to gain/phase margins.
3. **A measured simulation study** (§6) quantifying oscillation amplitude, settling time, overshoot, and MTTR with and without the constructs, on a released companion simulator.
4. **A measured stability frontier** (§6 E4): the greedy loop's settle/diverge boundary mapped across the gain × dead-time plane at three controller-stochasticity levels — a delay–gain product law with an explicit engineering margin, and the finding that stochasticity blurs the boundary rather than shifting it.

This paper is the rigorous, theory-grounded companion to pattern surveys of self-healing infrastructure: where those catalog *what* remediations exist, we analyze *whether the loops that execute them settle*.

2. Background: Loops, Not Decisions

MAPE-K and feedback control. Autonomic computing's Monitor-Analyze-Plan-Execute-over-Knowledge loop is explicitly a feedback controller [4], and a body of work applies control theory to runtime self-adaptation to obtain convergence and avoid oscillation and overshoot [5,8,9]. **Autoscaling as the canonical case.** Reactive autoscalers use

hysteresis/cooling to prevent flapping, accepting resource misalignment as the price of stability [2,3]; learning-based scalers wrestle with the same oscillation [6,7]. The lesson generalizes: *any* loop that actuates on a delayed, noisy signal can oscillate. AI remediation is such a loop, with worse delay and a non-stationary control law.

3. Related Work

Control-theoretic self-adaptation. Decades of work formalize self-adaptive systems as control loops with stability guarantees [4,5,8,9,11]; we inherit its vocabulary (gain, dead-time, settling time, hysteresis) and adapt it to an LLM controller. **Autoscaling stability and flapping.** Hysteresis/cooling in reactive autoscaling [2,3], learning-based autoscaling oscillation [6], and decentralized flapping detection/elimination among resources oscillating against each other [1] establish the instability modes we generalize. **AIOps remediation.** Autonomous-remediation systems and harnesses evaluate decision quality but not loop stability; our framing is orthogonal and composable with them. **Agent-level reliability frameworks.** Recoverability and containment metrics characterize single-agent failure; loop stability is a *system-level* property those metrics do not capture—an agent can be individually recoverable yet embedded in an unstable loop.

4. The Remediation Control Loop and Its Instabilities

4.1 The RCL. We model a self-healing system as a feedback loop: a *plant* (the managed system) with state $x(t)$; a *sensor* (telemetry) producing a delayed, noisy measurement; a *controller* (the AI agent) mapping measurements to actions with a probabilistic, non-stationary control law and dead-time τ (reasoning + tool latency); and an *actuator* (the remediation) altering the plant. Reliability of *individual* controller decisions is necessary but not sufficient for loop stability.

4.2 Oscillation (flapping). The loop repeatedly reverses a remediation because the measurement lags the action: the controller acts, the effect is not yet visible, it acts again, then over-corrects in reverse. Probabilistic actuation can *worsen* this (inconsistent action magnitudes) or, paradoxically, *mask* it (randomized actions look like noise rather than a clean oscillation), making it harder to detect.

4.3 Overshoot (over-remediation). The controller applies too large a correction relative to the dead-time, driving the plant past the target before feedback arrives—e.g., draining too many nodes, scaling too aggressively—then must correct back.

4.4 Runaway (positive feedback). The remediation amplifies the symptom: restarting a service that is failing *because* of restart storms; scaling a system whose bottleneck is downstream, increasing pressure. The loop's gain is effectively positive and the system diverges. This is the most dangerous mode because each step looks like "trying harder."

4.5 Loop interaction. Multiple controllers (agents, or an agent plus a legacy autoscaler) act on shared actuators with no coordination, beating against each other—the multi-controller flapping documented for resources oscillating against one another [1]. Each loop can be individually stable and the *coupled* system unstable.

5. Stable-by-Design Constructs

For each instability we specify a construct and an informal stability condition adapted to a probabilistic, high-dead-time controller.

5.1 Hysteresis gates (vs. oscillation). Require the symptom to cross *separated* engage/disengage thresholds before reversing a remediation, with the gap sized to the measurement delay. Informal condition: the hysteresis band must

exceed the peak-to-peak measurement noise *plus* the expected plant movement over one dead-time τ; otherwise the loop can reverse within its own blind window.

5.2 Action damping (vs. overshoot). Rate-limit actuation magnitude per loop iteration (small, incremental corrections) so a single decision cannot move the plant further than feedback can confirm within τ. Informal condition: per-step action magnitude × plant gain ≤ the correction observable within one dead-time. This is the loop analogue of a learning rate.

5.3 Settling-time cooldowns (vs. premature re-action). After acting, hold for a settling period ≥ the plant's response time + τ before permitting another action, so the controller observes the *settled* effect rather than the transient. Distinct from hysteresis: hysteresis governs *direction reversal*; cooldown governs *re-action timing*.

5.4 Dead-time-aware gating (vs. variable τ). Because AI controllers have large, variable reasoning latency, make the loop *aware* of its own dead-time: discount or suppress actions when telemetry is older than τ (the controller would be acting on the past), and widen damping/hysteresis as measured τ grows. This is the control-theoretic counterpart to the trust-calibration idea—here the untrusted input is *time*, not content.

5.5 Loop-interaction matrix (vs. interaction). Maintain an explicit map of which loops actuate which resources; before acting, a controller checks the matrix and yields, coordinates, or defers when another loop owns or is mid-correction on the same actuator. Informal condition for stability of the coupled system: no two active loops share an actuator without an ordering/lock—mutual exclusion on actuators as the minimal coordination primitive.

These constructs are individually familiar from autoscaling; the contribution is (i) recognizing that an LLM controller *needs them more*, for principled reasons (non-stationary gain, large variable dead-time), and (ii) the dead-time-aware and loop-interaction constructs, which target the specifically agentic failure modes. §6 E4 turns the underlying stability condition itself — how much correction a loop may commit per unit of delay — into a measured frontier with an explicit margin.

6. Simulation Study

Methodology. A discrete-time simulator of the RCL (companion artifact [12]): a bounded scalar plant holding a setpoint inside [0, 200], hit at step 20 by a persistent step disturbance (+40; +25 in coupled scenarios); a sensor reporting the plant two steps late with Gaussian noise; and a proportional remediation controller with *stochastic* action magnitude (±30% lognormal) and *stochastic* dead-time (lognormal around mean τ)—the probabilistic, high-latency controller of §1. Action direction is always correct given the measurement, so decisions are *individually correct by construction*; every pathology below is a loop property. Runs are 400 steps, 20 seeds per configuration; the qualitative claims are pinned by the artifact's 13-test suite.

Metrics. Oscillation fraction (runs with ≥3 action-direction reversals), mean reversals, mean peak-to-peak amplitude after the disturbance, overshoot past the setpoint, settling rate, and MTTR (steps until the plant stays within ±5 of setpoint for 20 consecutive steps).

E1 — dead-time alone destabilizes (H1 confirmed). With gain fixed at 0.3—chosen so the loop is stable at τ = 1—we sweep only dead-time:

Table 1 — Greedy baseline vs. dead-time (20 seeds per row).

dead-time τ	oscillating	mean reversals	mean amplitude	settled	MTTR
1	1/20	1.4	54.6	20/20	14.7
2	16/20	3.2	66.6	20/20	30.0
4	20/20	40.8	127.1	0/20	—
8	20/20	43.9	197.0	0/20	—
16	20/20	48.8	200.0	0/20	—

The control law never changes; only its latency does. The loop passes from stable through ringing ($\tau = 2$: oscillates but still settles) into sustained rail-to-rail oscillation that never settles—the classic dead-time instability, now exhibited by a controller whose every individual decision is directionally correct.

E2 — which constructs actually repair it (H2 confirmed, sharpened). At $\tau = 8$ (the LLM regime), constructs isolated and combined:

Table 2 — Constructs at $\tau = 8$ (20 seeds per row).

configuration	oscillating	settled	overshoot	MTTR
baseline (greedy)	20/20	0/20	50.0	—
hysteresis + damping only	20/20	0/20	50.0	—
cooldown alone	0/20	20/20	0.0	117.1
dead-time-aware gating alone	0/20	20/20	2.7	39.9
all four constructs	0/20	20/20	0.3	117.0

Two findings sharpen H2. First, the *amplitude-domain* constructs (hysteresis, damping) cannot by themselves repair a *time-domain* instability: they shave reversals and amplitude, but the loop still never settles. The *temporal* constructs are each individually sufficient. Second, **dead-time-aware gating buys stability at roughly one-third the recovery cost of a fixed cooldown** (MTTR 39.9 vs. 117.1): a cooldown must be sized for the worst case and holds that long every time, while dead-time-aware gating holds exactly as long as the controller's actual (variable) dead-time. This is direct evidence that the specifically agentic construct targets the specifically agentic failure mode. Composing all four is safe but inherits the most conservative construct's MTTR—the conservatism cost of §7, measured.

E3 — loop interaction (H3 confirmed). An "agent" (gain 0.8, $\tau = 6$, jittered) and a "legacy autoscaler" (gain 0.6, $\tau = 2$, acting every third step) each run with all single-loop constructs; each is individually stable (20/20 settled; MTTR 25.3 and 20.9). Coupled on the shared actuator with no coordination, they jointly over-correct: mean overshoot rises from 1.3 to 8.9 and only 13/20 runs settle. Notably this interference is *not* flapping—each loop stays nearly reversal-free while the *pair* drives the plant past the target or parks it off-setpoint, which is why single-loop oscillation detectors would miss it. With the interaction matrix (actuator mutex), the coupled system reproduces single-loop behavior exactly (20/20 settled, overshoot 1.3) at a coordination cost of ~4 MTTR steps. One implementation subtlety the simulation surfaced: a mutex over only the *in-flight* window is insufficient, because the second loop acts on telemetry that does not yet reflect the first loop's landed correction; the lock must extend through the correction's settling window—precisely the "mid-correction" ownership that §5.5 specifies.

E4 — the stability frontier (new in v0.3). E1 sweeps dead-time at one gain; E4 maps the boundary itself. For each (gain, dead-time) cell we run the greedy loop (no constructs) for 20 seeds and record the fraction that settle; the critical

dead-time $\tau^*(g)$ is the largest dead-time at which ≥ 90% of seeds settle (and every shorter one does). Let $L = \tau + \tau_s$ denote the full loop delay — dead-time plus sensor delay ($\tau_s = 2$ throughout).

Table 3 — the deterministic frontier (magnitude and dead-time jitter 0; 20 seeds/cell).

gain	τ^*	$L^* = \tau^* + \tau_s$	critical product $g \cdot L^*$
0.10	12	14	1.40
0.15	8	10	1.50
0.20	6	8	1.60
0.30	3	5	1.50
0.45	1	3	1.35
0.60	1	3	1.80
0.90	—	—	unstable at $\tau = 1$ ($g \cdot L = 2.7$)

Three findings. **The frontier is a delay–gain product law.** Across a 6× range of gains the critical product $g \cdot L^*$ stays within 1.35–1.8 — $g \cdot L \approx 1.5$ at the sweep's integer dead-time resolution — while gain varies 6× and critical dead-time 12×. The law has a one-line mechanism: a greedy loop acting every step commits ≈ $g \cdot e$ of correction per step for L steps before the first lands, so it stacks $g \cdot L$ error-multiples of correction blind; overshoot begins at $g \cdot L > 1$ (E1's ringing onset at $g \cdot L = 1.2$ is visible in Table 1 as 16/20 oscillating-but-settling) and successive half-cycles amplify beyond $g \cdot L \approx 2$. **The deterministic cliff is one dead-time step wide.** Across the entire 70-cell deterministic plane, no cell settles in between: every cell is 20/20 or 0/20 except a single 19/20 boundary cell. **Stochasticity blurs the frontier instead of shifting it.** At the paper's E1 jitter (0.3), four cells settle only partially; at heavy jitter (0.6), fifteen do — and the blur runs in both directions: a deterministically-stable marginal cell degrades ($g = 0.6$, $\tau = 1$, $g \cdot L = 1.8$: 20/20 → 6/20) while a deterministically-divergent cell at the same product occasionally settles ($g = 0.3$, $\tau = 4$: 0/20 → 9/20). Within the band $g \cdot L \approx 1.35$–1.8, settling under jitter ranged from 25% to 100% with no monotone pattern — the band is not merely derated, it is unreliable.

The engineering consequence is a margin rule rather than a boundary rule: **every sampled cell with $g \cdot L \leq 1.2$ settled in every seed at every stochasticity level.** For an LLM controller — whose gain is not a design constant and whose dead-time is a distribution — the operating point belongs at $g \cdot L \leq 1.2$: commit no more than ~120% of one observed error's worth of correction across the interval the loop cannot yet see, and put a measured p95 of reasoning latency, not its mean, into the product as L.

7. Discussion and Threats to Validity

Simulation vs. production. A simulator abstracts the plant; the stochastic-controller abstraction approximates an LLM's variability and is not the LLM itself. We treat the §6 study as establishing the *mechanisms*; production validation is future work. **Parameter sensitivity.** E4 measures the E1 boundary directly: a delay–gain product law with critical product 1.35–1.8 across a 6× gain range. The constant is plant-specific — it depends on disturbance size, the noise floor, and the settle band — and is not a portable number; the product *structure*, the sharpness of the deterministic cliff, and the both-directions blur under stochasticity are the portable findings. **Conservatism cost.** Stability constructs slow the loop; over-damping raises MTTR. §6 measures the trade directly—a fixed cooldown pays ~3× the recovery time of dead-time-aware gating for the same stability—and the right operating point is service-specific (and connects to the cost-reliability frontier of related work). **Non-stationarity.** A probabilistic controller's "gain" is only loosely defined; our stability conditions are engineering conditions, not formal proofs — E4 tightens the central one empirically (a measured frontier

with a margin), but a proof for the non-stationary controller remains open. **Composability.** The constructs compose with decision-quality and containment work rather than replacing it. **Nested loops.** The remediation loop is increasingly itself governed: an authority-management layer adjusts how much the controller may do, based on its adjudicated track record. That layer is an outer control loop on the same plant, and it inherits this paper's failure modes at its own timescale — the sibling budget analysis [13] measures the governance-layer oscillation that appears when its hysteresis (asymmetric fast-down/slow-up authority transitions) is removed (16.1 vs. 1.4 transitions, 2.5× the realized harm), and shows the governance loop's own sensor dead-time (the adjudication pipeline) to be exactly additive to its detection latency. The cascade rule applies across the pair: the outer loop must run slower than the inner loop settles.

8. Conclusion and Future Work

Self-healing infrastructure has been evaluated one decision at a time, but its characteristic catastrophes—flapping, over-remediation, runaway—are properties of the *loop*, not the decision. Control theory has long understood such loops; what is new is a controller that is probabilistic, slow, and rarely alone. We modeled the AI-driven Remediation Control Loop, catalogued its instabilities, and specified stable-by-design constructs—hysteresis, damping, cooldowns, dead-time-aware gating, and actuator coordination—with informal stability conditions tuned to an LLM-in-the-loop. The §6 simulation confirms all three hypotheses and adds a design-relevant asymmetry: temporal constructs repair what amplitude constructs cannot, and the dead-time-aware variant does so at a third of the recovery cost of a fixed cooldown. E4 then maps the boundary those constructs defend — a delay–gain product law, sharp for a deterministic controller and blurred in both directions for a stochastic one — and yields a concrete operating margin, $g \cdot L \leq 1.2$, at which every sampled configuration settled. The simulator is released alongside this paper. Next steps: formalize stability margins for non-stationary controllers, and validate in production self-healing systems. The headline for practitioners is unchanged from a century of control engineering, now pointed at AI operations: **make the loop stable by design; a system of individually-correct decisions can still shake itself apart.**

References

(Author lists and identifiers verified against arXiv/DOI metadata and live URLs, 2026-09-01.)

[1] A. Vaca and F. Milano. Decentralized Approach to Detect and Eliminate Flapping Phenomena due to Flexible Resources. arXiv:2511.02497, 2025. [2] F. Shaikh, G. Reali, and M. Femminella. Mitigating Temporal Blindness in Kubernetes Autoscaling: An Attention-Double-LSTM Framework. arXiv:2603.28790, 2026. [3] E. Rutten, S. Cerf, R. Bleuse, V. Reis, and S. Perarnau. Sustaining Performance While Reducing Energy Consumption: A Control Theory Approach. arXiv:2107.02426, 2021. [4] J. O. Kephart and D. M. Chess. The Vision of Autonomic Computing. *IEEE Computer* 36(1), 2003. [5] J. L. Hellerstein, Y. Diao, S. Parekh, and D. M. Tilbury. *Feedback Control of Computing Systems.* Wiley-IEEE Press, 2004. [6] H. Arabnejad, C. Pahl, P. Jamshidi, and G. Estrada. A Comparison of Reinforcement Learning Techniques for Fuzzy Cloud Auto-Scaling. arXiv:1705.07114, 2017. [7] W. Feng, R. Xiao, Z. Li, H. Yu, G. Sun, L. Luo, M. Guizani, Q. Ho, et al. Learning In Chaos: Efficient Autoscaling and Self-Healing for Multi-Party Distributed Training. arXiv:2505.12815, 2025. [8] A. Filieri, M. Maggio, et al. Software Engineering Meets Control Theory. In Proc. SEAMS 2015. [9] S. Shevtsov, M. Berekmeri, D. Weyns, and M. Maggio. Control-Theoretical Software Adaptation: A Systematic Literature Review. *IEEE Transactions on Software Engineering* 44(8), 2018. [10] K. J. Åström and R. M. Murray. *Feedback Systems: An Introduction for Scientists and Engineers.* 2nd ed., Princeton University Press, 2021. [11] R. de Lemos et al. Software Engineering for Self-Adaptive Systems: A Second Research Roadmap. In *Software Engineering for Self-Adaptive Systems II*, LNCS 7475, Springer, 2013. [12] A. Baby. remediation-stability-sim: A Discrete-Time Simulator for AI-Driven Remediation Loop Stability. Companion artifact, 2026. https://github.com/ajinb/remediation-stability-sim (public, Apache-2.0). [13] A. Baby. Error Budgets for Autonomy:

SLO-Driven Authority Management for Autonomous AI Operators. Preprint, cloudandsre.com, 2026. https://cloudandsre.com/research/error-budgets-for-autonomy/