Approved Too Late: Verdict Staleness in LLM-Guarded Self-Adaptive Systems

Ilai Shraga
University of Cambridge
Cambridge, United Kingdom
is628@cam.ac.uk

Roei Eshel
Maccabim-Re'ut High School
Modi'in-Maccabim-Re'ut, Israel
roeieshel@gmail.com

Lior Gorelik
The Open University of Israel
Raanana, Israel
golior32@365.openu.ac.il

***Abstract*—A large language model (LLM) guardrail for a self-adaptive system (SAS) may issue an approval that is correct at check time but stale by actuation. This creates an Execute-stage time-of-check to time-of-use (TOCTOU) hazard. We study verdict freshness: whether a guardrail verdict remains valid when used. We distinguish three quantities that answer different questions: all-candidate verdict change under fixed-action replay, oracle-labeled approval expiry on recorded closed-loop trajectories, and judge-conditioned use-time invalidity. Across five reproducible SAS environments, all-candidate verdict-change rates span 5.3–48.4% at a common replay shift of eight simulator steps. We introduce the Freshness-Bounded Shield (FBS), which estimates each approval's validity horizon from its safe-side margin and recent feature volatility, without an explicit plant-dynamics model. Using fixed settings documented in the artifact, FBS reduces oracle-labeled approval-expiry rates from 3.4–24.7% to 0–1.8% at the same shift. A separate audit of four LLM judges finds nonzero judge-conditioned use-time invalidity in every approval stream. We formulate a freshness contract: every approval must be correct at check time and remain valid at use time.**

***Index Terms*—self-adaptive systems, large language model guardrails, verdict freshness, time-of-check to time-of-use (TOCTOU), runtime assurance**

I. Introduction

An Execute-stage guardrail can return a correct verdict on the context–action pair it checks and still authorize an action that is inadmissible when applied. In a self-adaptive system (SAS), a controller observes the plant, proposes an action, and relies on a large language model (LLM) guardrail to approve or reject it before actuation. The plant continues to evolve while the verdict is computed and delivered. By actuation, the checked context may no longer describe the current plant state, and the action may no longer be admissible. This creates an assurance gap between check-time correctness and use-time validity.

We frame this gap as an Execute-stage TOCTOU-style hazard [1], [2] within the MAPE-K (Monitor–Analyze–Plan–Execute over a shared Knowledge base) loop [3], [4] (Fig. 1). Ordinary closed-loop evolution can invalidate an already-issued approval even without an adversary. Approval latency sets the duration of the exposure window, while plant and predicate dynamics determine whether validity is lost. The central question is not only whether the guardrail returns a correct verdict for context $x_t = (o_t, h_t)$ and action a_t at check time, but whether that verdict remains valid for (x_{t+K}, a_t) at actuation. We call validity at use time *verdict freshness* and its loss *verdict staleness*. Its validity therefore depends on when it is used.

The hazard extends beyond LLMs to any delayed Execute-stage approver whose verdict is grounded in an evolving plant state (Sec. V). LLM guardrails motivate our study because they insert a model-serving step between planning and actuation, making approval latency part of the control-loop timing. We evaluate the hazard using fixed-action replay over fixed-seed trajectories from five reproducible SAS environments spanning IoT network adaptation, simulator tuning for edge–cloud resource management, architectural self-healing, adaptive object detection, and backend job dispatch: DeltaIoT [5], SimTune [6], mRUBiS [7], Switch [8], and SimDex [9] (Secs. III and III-F).

We make three contributions. *(i) Concept:* we identify and formalize an Execute-stage TOCTOU-style hazard in SASs: a semantic approval grounded in an observed plant state can lose validity before actuation under endogenous closed-loop evolution. *(ii) Measurement:* we hold each proposed action fixed and re-evaluate its admissibility against later recorded contexts, distinguishing all-candidate verdict change, oracle-labeled approval expiry on recorded closed-loop trajectories, and judge-conditioned use-time invalidity. All-candidate verdict change is nonzero in all five environments, and a descriptive audit of four LLM judges finds nonzero judge-conditioned use-time invalidity in every audited approval stream. *(iii) Mitigation:* we formulate a freshness contract and instantiate it with FBS, a lightweight post-verdict gate that estimates each approval's validity horizon from its safe-side margin and recent feature volatility without an explicit plant-dynamics model. At the common replay shift $K{=}8$, under fixed settings documented in the artifact, FBS reduces oracle-labeled approval-expiry rates in all five environments. In the audited DeltaIoT cell, it reaches 0% expiry while matching the *reference-open* (`e_open`) baseline's reported mean reward.

II. Related Work: What Becomes Stale?

An Execute-stage verifier must answer two questions: is the proposed action acceptable when checked, and does that approval remain valid when applied? Related work usually addresses check-time acceptability, assigns freshness to a

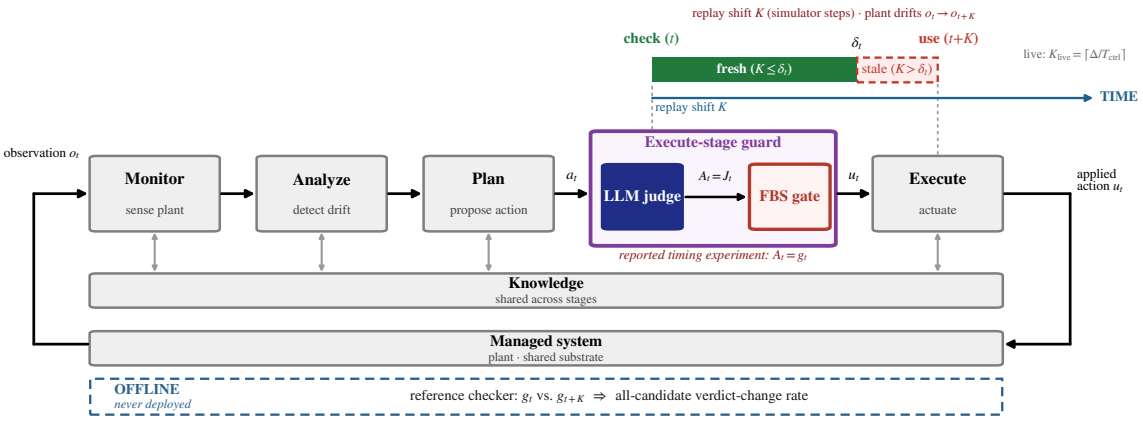

Fig. 1. Pipeline view of Execute-stage verdict freshness in the MAPE-K loop. The upper path depicts the intended Judge+FBS deployment: the LLM judge evaluates check-time context $x_t = (o_t, h_t)$ and candidate action a_t, and supplies upstream approval $A_t = J_t$. The reported oracle-labeled timing experiment instead sets $A_t = g_t$ to isolate temporal expiry from check-time judge error. During the check–use window, the plant evolves to the later recorded context x_{t+K}. For an upstream-approved candidate, FBS applies a_t as u_t only if its safe-side margin is positive and replay shift K does not exceed the estimated validity horizon δ_t; otherwise, it applies the fallback. A live deployment conservatively discretizes end-to-end elapsed time as $K_{\text{live}} = \lceil \Delta / T_{\text{ctrl}} \rceil$. Used only offline, the deterministic reference checker evaluates the same candidate at x_t and x_{t+K}; disagreement contributes to the all-candidate verdict-change rate.

different runtime object, or provides a complementary intervention mechanism.

Runtime LLM guardrails constrain conversational behavior or classify prompts and responses at inspection time [10], [11]. The GAP benchmark identifies a distinct execution gap: text-level safety need not transfer to the resulting tool call [12]. In SAS research, a recent roadmap maps potential LLM roles across MAPE-K, while iLLM-TSC reviews and may revise an action proposed by an RL policy [13], [14]. These works address check-time assessment; when such checks authorize execution, they do not estimate how long the resulting approval remains valid under subsequent plant evolution.

Freshness and delay work attaches time to information or adaptation tactics rather than approvals. Age of Information characterizes the age of received status information [15], while tactic-volatility work addresses variation in tactic latency or cost [16]. TOCTOU provides the closest structural analogue: classical work examines mutable state between check and use, and recent LLM-agent and browser-agent work revalidates external state before dispatch [1], [2], [17]. These lines share the check–use separation, but the revalidated object is information, external state, or authority evidence rather than an Execute-stage semantic approval whose validity changes under endogenous closed-loop plant evolution.

Runtime assurance provides the closest mitigation architecture. Simplex and model-predictive shielding switch to a backup or filter candidate actions based on current or predicted safety [18], [19]. FBS follows the same broad intervention pattern but uses a different trigger: it compares an approval's age with a validity horizon estimated from the signed safe-side margin and recent feature volatility, without an explicit plant-dynamics model. We are not aware of prior work that estimates such a plant-dependent horizon for an already-issued Execute-stage semantic approval under endogenous closed-loop evolution. These approaches are complementary reference points rather than direct baselines; comparing them under matched dynamics-model, verifier-call, and fallback assumptions remains future work.

III. Methodology

Our replay asks one question: if the deterministic reference checker deems candidate action a_t, grounded in context $x_t = (o_t, h_t)$, admissible at check time, does the same candidate remain admissible at replay use time $t + K$? To answer it, we hold a_t fixed, take the later recorded context $x_{t+K} = (o_{t+K}, h_{t+K})$ from the same episode, and compare the two verdicts of the same deterministic reference checker (the *oracle*). The shifted history is reconstructed from the recorded prefix ending at $t+K$ and contains no later information. This is a fixed-action relabeling audit: it evaluates the candidate against later recorded contexts rather than reconstructing the counterfactual trajectory that delaying, rejecting, or replacing a_t would induce (Sec. VI).

We use three related quantities with different conditioning sets. The *all-candidate verdict-change rate* averages reference-label changes before conditioning on a method's pass set. The directional *oracle-labeled approval-expiry rate* is computed, for each method, among candidates that method passes and that are reference-admissible at check time in the corresponding recorded experiment. The descriptive LLM audit separately reports *judge-conditioned use-time invalidity* within each judge's own approval set; this quantity may include check-time judge error as well as temporal expiry.

A. System, Threat Model, and Clock Model

At step t, the monitor emits observation o_t, the controller proposes action a_t, and an Execute-stage LLM guardrail returns verdict $J_t \in \{0, 1\}$ on (x_t, a_t), where $x_t = (o_t, h_t)$ includes a short history and $J_t{=}1$ denotes approval. In replay, the same candidate is re-evaluated against x_{t+K}. An approval's

grounding age is measured from acquisition of o_t, not from verdict issuance, and therefore includes all latency between observation and actuation. The threat model excludes prompt injection and deliberate adversarial state manipulation; the hazard is ordinary closed-loop plant evolution during that interval.

We distinguish wall-clock latency Δ from replay shift K (Fig. 1). Because the simulators are step-driven, K counts each simulator's own control steps and is the experimental knob. Let Δ denote the end-to-end elapsed time from acquisition of o_t to actuation and let T_{ctrl} denote the control period. Replay represents age through integer observation shifts, so we conservatively discretize a latency realization as

$$K_{\text{live}} = \left\lceil \frac{\Delta}{T_{\text{ctrl}}} \right\rceil .$$

Thus, any positive remainder beyond an integer number of control periods is assigned to the next age step. For example, a 2 s latency and a 0.25 s control period yield $K_{\text{live}}=8$. Variable serving latency induces a distribution over K_{live}, not a single fixed age.

We sweep $K \in \{1, 2, 3, 5, 8, 10\}$ and use a common replay shift of eight simulator steps for the headline cross-environment comparison (Table II and Fig. 2). This does not imply a common physical duration: step semantics and control periods differ across environments. For each K, an index is eligible only when $t+K$ remains within the same episode. The sparse grid covers short, intermediate, and longer observation shifts while keeping the replay campaign tractable. Because the eligible set can shrink with K, cross-shift curves are descriptive and may reflect both age and episode-boundary cohort changes (Sec. VI).

B. Reproducible Environments

Table I summarizes the five environments. DELTAIOT is a multi-hop IoT network whose adaptations adjust per-link communication settings under interference. SIMTUNE tunes simulator parameters for edge–cloud resource-management workloads. mRUBIS is a component-based marketplace exemplar for architectural self-healing. SWITCH switches among object-detection model variants as load changes. SIMDEX dispatches jobs across backend workers. The lineup is not intended to make any individual benchmark representative; it applies one replay protocol across systems with different domains, plant dynamics, and operational predicates.

C. Replay-Based Freshness Metrics

Let $g(a_t, x_t; \theta) \in \{0, 1\}$ be the binary verdict of the deterministic reference checker, where $g=1$ denotes reference admissibility. Let $\mathcal{T}_K$ contain the logged indices for which $t+K$ remains in the same episode, and define $g_t = g(a_t, x_t; \theta)$ and $g_{t+K} = g(a_t, x_{t+K}; \theta)$. The all-candidate trajectories use candidates generated by each environment's logged controller, so the resulting rates reflect its controller-induced state, history, and action distribution. For $t \in \mathcal{T}_K$,

$$\text{change}(t, K) = \mathbf{1}[g_t \neq g_{t+K}] . \tag{1}$$

TABLE I
THE FIVE REPRODUCIBLE SAS ENVIRONMENTS. "ADMISSIBLE" GIVES THE DIRECTION OF THE SELECTED SCALAR PREDICATE RELATIVE TO THRESHOLD θ; THESE PREDICATES PROVIDE OPERATIONAL EXPERIMENTAL LABELS RATHER THAN COMPLETE SAFETY SPECIFICATIONS. SWITCH USES A PER-INSTANCE CALIBRATED THRESHOLD ("CALIB."). THRESHOLDS ARE HELD CONSTANT ACROSS COMPARED METHODS AND SEEDS.

Env	Domain	Gate feature	Admissible	θ
DELTAIOT	IoT network	`packet_loss`	$z \leq \theta$	0.10
SIMTUNE	Edge–cloud	`rt_err`	$z \leq \theta$	2.0
mRUBIS	Self-healing	per-step utility	$z \geq \theta$	27.0
SWITCH	Adaptive vision	`utility`	$z \geq \theta$	calib.
SIMDEX	Job dispatch	`delay_avg`	$z \leq \theta$	5.0

The all-candidate verdict-change rate is the empirical mean of (1) over $\mathcal{T}_K$, before conditioning on any evaluated pass set. It includes both expiry ($g_t{=}1,\ g_{t+K}{=}0$) and recovery ($g_t{=}0,\ g_{t+K}{=}1$). The term *all-candidate* describes the denominator; the rate remains conditional on the logged distribution.

For a method M, let $S_t^M \in \{0, 1\}$ indicate that M passes candidate a_t in the corresponding recorded experiment. Its directional oracle-labeled approval-expiry rate is

$$R_{\text{exp}}(M, K) = \Pr\big[g_{t+K}{=}0 \mid S_t^M{=}1,\ g_t{=}1\big] . \tag{2}$$

This quantity asks: among candidates passed by M that were reference-admissible at check time, what fraction are inadmissible at replay use time? Recovery is a missed opportunity rather than an unsafe stale approval. Paired seeds align environment randomness across compared methods; the metric does not assume that their candidate streams remain identical after fallback interventions.

The descriptive LLM audit uses a different denominator. For judge verdict J_t, it reports

$$R_{\text{judge}}(J, K) = \Pr[g_{t+K}{=}0 \mid J_t{=}1] . \tag{3}$$

Because (3) does not additionally condition on $g_t{=}1$, it may include both candidates already inadmissible at check time and candidates whose admissibility expires later. It therefore measures judge-conditioned use-time invalidity, not temporal expiry alone.

The LLM judge, reference checker, and FBS play different roles. The judge supplies the check-time semantic verdict. The deterministic reference checker is an offline measurement device, never deployed in the loop, that labels admissibility at check and replay use time. FBS decides whether an upstream approval remains within an estimated temporal horizon; it does not re-judge the candidate's semantics. Calling the checker an oracle is shorthand for its measurement role, not a claim that it captures every safety-relevant property. The conceptual contribution concerns the lifetime of a semantic approval, whereas the main timing experiment uses deterministic operational labels and FBS uses a scalar proxy; the separate LLM audit does not constitute end-to-end Judge+FBS evaluation.

D. Freshness-Bounded Shield

Each adapter exposes a scalar state feature $z_t = f(o_t)$ used by the freshness gate. The offline reference checker may additionally use a_t and h_t; accordingly, FBS is a lightweight proxy for the selected scalar predicate rather than a certificate for the full checker. Define the signed safe-side margin as $m_t = \theta - z_t$ when admissibility requires $z \leq \theta$, and $m_t = z_t - \theta$ when it requires $z \geq \theta$. Thus, $m_t \geq 0$ on the admissible side and $m_t > 0$ strictly inside the boundary. The estimated horizon is

$$\delta_t = \min\left(\delta_{\max},\ \frac{\max(m_t, 0)}{\max(\mathrm{vol}_t, \varepsilon)}\right), \qquad \mathrm{vol}_t = \alpha\,|z_t - z_{t-1}| + (1-\alpha)\,\mathrm{vol}_{t-1}, \tag{4}$$

with α=0.1 fixed across all five environments. At this value, the contribution of an observed change to the EMA decays by half after $\ln(1/2)/\ln(0.9) \approx 6.6$ simulator steps. The margin measures distance to the scalar boundary, and vol_t is an exponential moving average of recent absolute feature changes. Their ratio is used as a heuristic time-to-boundary estimate. The floor $\varepsilon > 0$ prevents a momentarily static feature from receiving an unbounded horizon, while $\delta_{\max}$ caps the horizon. The artifact fixes ε, $\delta_{\max}$, initialization, and update order.

Let $A_t \in \{0, 1\}$ denote the upstream approval. In a deployed Judge+FBS composition, A_t is the judge verdict; in the oracle-labeled timing experiment, $A_t = g_t$. The candidate passes if and only if A_t=1, $m_t > 0$, and $K \leq \lfloor \delta_t \rfloor$. Otherwise, the environment-specific fallback is applied.

The rule is motivated by a bounded-drift argument for the selected scalar predicate. If $|z_{s+1} - z_s| \leq b$ at every step, then the feature moves by at most Kb over K steps, and $Kb \leq m_t$ is sufficient to avoid crossing an inclusive threshold. FBS substitutes the smoothed recent change vol_t for the unknown worst-case bound b. Because vol_t is not an upper bound, the resulting horizon is a heuristic rather than a safety certificate: an abrupt change can outpace the estimate and invalidate a passed approval. This is a possible failure mode of the rule, not evidence that it caused any particular residual observed below. The signed margin also prevents a large distance on the inadmissible side from producing a long lifetime; if $m_t \leq 0$, FBS invokes the fallback.

E. Methods Compared

Each compared variant is evaluated under paired seeds in its corresponding recorded closed-loop experiment. The *reference-open* (`e_open`) baseline applies every candidate with positive upstream approval and performs no freshness check. In the oracle-labeled timing experiment, $A_t = g_t$, so it passes every candidate deemed admissible by the reference checker at check time. FBS (`fbs_last_safe`) uses the same upstream approval source in that experiment and additionally rejects candidates whose estimated horizon has expired. Thus, the reported `e_open`–FBS comparison isolates freshness gating; it is not an end-to-end evaluation of an LLM-generated approval stream. `margin_only` retains only boundary-distance information, while `vol_only` retains only recent-volatility information. `e_deadline` rejects every approval beyond a fixed age regardless of context and appears only in the DELTAIOT safety–utility audit. Exact ablation rules, deadline, parameter values, fallback definitions, and pass frequencies are recorded in the artifact.

F. Reproducibility and Traceability

All five environments pass a two-run reproducibility test under fixed seeds. Reproduction is bit-exact where supported and tolerance-bounded for JVM-backed environments whose upstream simulators reproduce floating-point outputs within a fixed numerical tolerance rather than bit for bit. Each adapter imports directly from the corresponding upstream simulator, and a SHA-256 manifest pins the artifacts and generated traces. Closed-loop comparisons use n=30 paired seeds per environment and method (n=60 for SIMDEX). Decision-level outcomes are aggregated into one rate per seed, and methods are compared using paired same-seed Wilcoxon signed-rank tests [20]. The artifact records the SciPy version, the exact `alternative`, `zero_method`, `method`, and continuity-correction settings, the treatment of tied and zero differences, the handling of seeds with no passed candidates, the prompts, decoding settings, model identifiers, and per-seed outputs.

IV. RESULTS

We address three research questions. **RQ1:** How does all-candidate verdict change vary across environments and replay shifts? **RQ2:** At K=8, does FBS reduce oracle-labeled approval expiry on recorded closed-loop trajectories, and, in the audited DELTAIOT cell, what do the ablation and safety–utility audit reveal about over-blocking? **RQ3:** What judge-conditioned use-time invalidity remains in concrete LLM approval streams?

All three analyses re-label admissibility at replay use time, but they condition on different sets. The curves in Fig. 2 report all-candidate verdict change over eligible logged candidates. Table II and the K=8 markers in Fig. 2 report method-specific approval expiry among passed candidates that are reference-admissible at check time in each corresponding experiment. Fig. 3 instead reports $R_{\mathrm{judge}}(J, K)$ within each judge's own approval set. The direct mitigation comparison is therefore *reference-open* (`e_open`) versus FBS within the oracle-labeled timing experiment. It isolates freshness gating and should not be interpreted as an end-to-end comparison of LLM approval streams; numerical gaps across the three metric families are not effect estimates.

A. RQ1: How Does Verdict Change Vary?

All-candidate verdict change is nonzero in every environment. At the common replay shift K=8, it ranges from 5.3% in SIMDEX to 48.4% in DELTAIOT, a roughly ninefold spread (Fig. 2). Because a simulator step need not represent the same physical duration across environments, this is a common integer shift rather than a common wall-clock age. Plant dynamics, predicate structure, and the logged candidate

distribution determine how often the reference verdict changes within each environment's shift.

The curves exhibit four descriptive shapes. DELTAIOT and SIMTUNE are *early-saturating*: their rates are already near 48% at $K{=}1$ and vary little over the sampled ages. This pattern is consistent with frequent boundary crossings or weak temporal persistence, but the curves alone do not identify the cause. mRUBIS is *age-sensitive*: its rate rises overall from 16.5% at $K{=}1$ to 32.7% at $K{=}8$, despite local non-monotonicity. SWITCH is approximately *flat*, while SIMDEX is *low-base* with small local variation. These labels summarize observed shapes rather than establish underlying mechanisms.

B. RQ2: Does FBS Reduce Approval Expiry?

At $K{=}8$, FBS reports a lower oracle-labeled approval-expiry rate in all five environments, reducing the *reference-open* (`e_open`) range of 3.4–24.7% to 0–1.8% (Fig. 2 and Table II). The observed FBS-to-`e_open` rate ratios range from 0 to 0.18, equivalent to relative reductions of 82–100%. All five paired per-seed comparisons yield raw $p{<}0.001$ under the configured Wilcoxon signed-rank tests. Because each rate conditions on the candidates passed by the corresponding method, a lower expiry rate alone does not determine whether the reduction is selective or results from more aggressive rejection.

C. Ablation: Freshness Signals and Over-Blocking

The single-signal variants isolate the two inputs to (4), but approval expiry alone cannot identify over-blocking because each variant induces a different pass set. `margin_only` uses only distance to the admissibility boundary and reaches 0% approval expiry in every environment; the expiry table alone cannot determine whether this reflects selective gating or conservative rejection. `vol_only` uses only recent feature movement and remains close to *reference-open* (`e_open`) in DELTAIOT, SIMTUNE, and SWITCH; for example, their rates are 23.9% and 24.0% on DELTAIOT. On SIMDEX, `vol_only` matches the full-FBS rate of 0.6%. These patterns motivate combining boundary distance and recent motion, but do not by themselves establish a safety–utility benefit.

The DELTAIOT safety–utility audit at $K{=}8$ provides complementary evidence in one cell (Table III). `margin_only` reaches 0% approval expiry but reduces reported mean episode reward from the *reference-open* (`e_open`) baseline's -960 to -1050, while `e_deadline` reduces it to -1200. FBS reaches 0% approval expiry while matching the reported reference-open mean reward of -960. Thus, in this audited cell, FBS reduces expiry without the mean-reward loss observed for the two more conservative alternatives. This does not establish the same trade-off in other environments, at other replay shifts, or under other fallback designs.

D. RQ3: Use-Time Invalidity in LLM Approval Streams

Every audited judge stream has a nonzero observed use-time invalidity rate. We replayed approval streams from four LLM judge backends—`qwen2.5:0.5b`, `llama3.2:1b`, Claude Haiku, and `gpt-4o-mini`—under the same observation-shifting protocol. Each judge receives (x_t, a_t) and the environment's operational admissibility predicate and returns an approve/reject verdict.

At $K{=}8$ in the early-saturating DELTAIOT setting, use-time invalidity within each backend's own approval set ranges from 11.5% to 36.8% (Fig. 3). Because each judge induces a different approval set and the audit does not additionally condition on reference admissibility at check time, these rates may combine check-time judge error with temporal expiry. They characterize exposure within each stream; they do not isolate temporal expiry, rank judge quality, or provide a direct comparison with FBS, which is evaluated separately. The full backend-by-environment matrix is provided in the artifact. End-to-end Judge+FBS evaluation on the same judge-generated stream remains future work.

TABLE II
ORACLE-LABELED APPROVAL-EXPIRY RATE (%) AT $K{=}8$ ON RECORDED CLOSED-LOOP TRAJECTORIES. EACH VALUE CONDITIONS ON CANDIDATES PASSED BY THE CORRESPONDING VARIANT AND REFERENCE-ADMISSIBLE AT CHECK TIME. LOWER VALUES ALONE DO NOT ESTABLISH A BETTER SAFETY–UTILITY TRADE-OFF.

Variant	DELTAIOT	SIMTUNE	mRUBIS	SWITCH	SIMDEX
Ref.-open (`e_open`)	24.0	24.7	13.8	6.8	3.4
`margin_only`	0.0	0.0	0.0	0.0	0.0
`vol_only`	23.9	24.0	8.9	6.7	0.6
FBS	0.0	0.0	1.8	0.0	0.6

TABLE III
SAFETY–UTILITY AUDIT ON DELTAIOT AT $K{=}8$. APPROVAL EXPIRY CONDITIONS ON CANDIDATES PASSED BY THE METHOD AND REFERENCE-ADMISSIBLE AT CHECK TIME; REWARD IS THE REPORTED MEAN EPISODE REWARD OVER PAIRED SEEDS.

Method	Expiry (%)	Mean reward
Ref.-open (`e_open`)	24.0	-960
FBS	0.0	-960
`margin_only`	0.0	-1050
`vol_only`	23.9	-965
`e_deadline`	0.0	-1200

V. DISCUSSION

For deployment, check-time judge accuracy and median verifier latency are not enough. The latency distribution determines the grounding-age distribution, while signed margin and recent feature volatility help characterize exposure to plant drift [15], [21]. Runtime monitoring should track the distributions of K_{live}, m_t, and vol_t, together with fallback frequency and task utility. Periodic offline replay audits should report oracle-labeled approval expiry separately. Judge-conditioned use-time invalidity is a different quantity because it conditions on each judge's own approval set and may include check-time error.

These requirements define a *freshness contract*: an approval records its checked context and action, grounding time, expiry or revalidation rule, and a fallback justified for the deployment. FBS implements one version of this contract using signed mar-

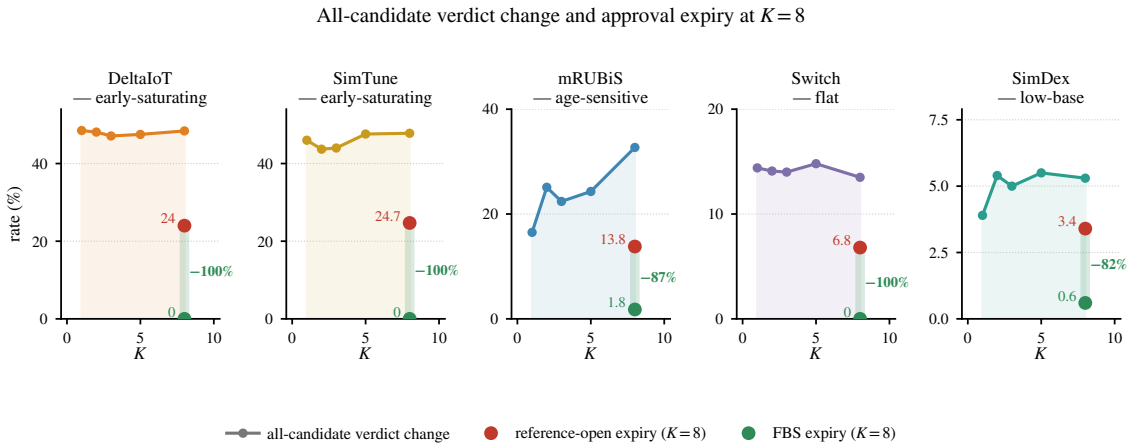

Fig. 2. All-candidate verdict change across replay shift K, with method-specific oracle-labeled approval expiry at $K=8$. Each curve reports the all-candidate verdict-change rate for one environment. The red and green markers report approval-expiry rates for *reference-open* (`e_open`) and FBS in their corresponding recorded experiments. Curves and markers represent different events and denominators; curve–marker gaps are not effect estimates. The direct mitigation comparison is between the paired red and green outcomes. K counts each simulator's own steps, not a common wall-clock duration.

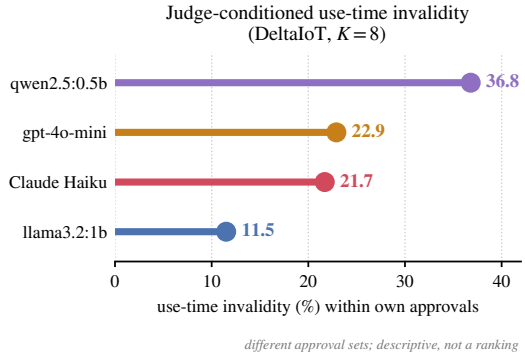

Fig. 3. Judge-conditioned use-time invalidity in four LLM approval streams on DELTAIOT at $K=8$. Each marker is computed within that judge's own approval set and may include both check-time judge error and temporal expiry. The observed rates range from 11.5% to 36.8%; because the approval sets differ, marker heights characterize separate streams rather than rank judge quality.

gin and recent feature volatility, without an explicit dynamics model; other deployments may use different mechanisms.

Several complementary mechanisms can reduce the hazard. Lower verifier latency narrows but does not eliminate the check–use window; a dynamics model can support prediction of future safety or recoverability [19], while revalidation immediately before dispatch can check a more current state [17]. FBS instead expires an earlier approval without another verifier call, but its horizon is heuristic rather than a safety certificate. Its value depends on the fallback: a no-op or held action is not inherently safe. Because approval-expiry rates condition on each method's pass set, they should be reported with fallback frequency and task utility [18]. The DELTAIOT audit shows one favorable safety–utility point, not a general guarantee.

Once z_t, θ, and age are available, FBS adds $O(1)$ scalar work and no additional model call; this excludes deployment-specific feature extraction, timestamping, and fallback execution. The freshness requirement is not LLM-specific, but applying FBS elsewhere requires a meaningful scalar margin, a usable online volatility estimate, trustworthy age measurement, and a justified fallback. End-to-end evaluation of Judge+FBS on the same approval stream remains future work.

VI. LIMITATIONS

This study evaluates all-candidate verdict change and oracle-labeled approval expiry in five step-driven simulators at a sparse set of replay shifts K, and evaluates FBS under fixed settings ($\alpha=0.1$; the remaining parameters are fixed in the artifact). Equal K values denote equal simulator-step shifts, not equal physical durations. The findings apply to the evaluated environments, scalar predicates, logged candidate distributions, replay grid, and configuration, not to a particular deployment. A deployment-specific estimate requires measured end-to-end latency traces, their discretization into K_{live}, and evaluation on the target plant and controller-induced action distribution. Sensitivity to FBS parameters and unsampled ages remains unmeasured. Because the eligible replay cohort shrinks near episode boundaries as K increases, cross-age curves may also reflect cohort changes.

Replay is a fixed-action relabeling audit, not an intervention-consistent causal simulation: it evaluates a_t on later recorded contexts rather than reconstructing the trajectory induced by delaying, rejecting, or replacing it. The reference checker supplies deterministic labels for selected operational predicates but is not a complete safety oracle. FBS uses a scalar state feature as a proxy and therefore does not certify every dependency of the full checker. Utility and fallback trade-offs are tested only on DELTAIOT at $K=8$. Judge-conditioned rates may mix check-time error with temporal expiry; check-time LLM accuracy and same-stream Judge+FBS composition are not evaluated. Finally, FBS replaces an unknown worst-case drift

bound with a smoothed recent-change estimate and depends on trustworthy age measurement and a justified fallback. It is a proof-of-concept heuristic, not a certified safety guarantee.

VII. Conclusion

An Execute-stage approval is not a timeless Boolean. It can lose validity as the plant evolves away from the context on which it was based. Across five reproducible, fixed-seed SAS environments, the all-candidate verdict-change rate is nonzero and spans 5.3–48.4% at the common replay shift $K{=}8$. This roughly ninefold spread shows that age alone does not determine the observed rate; plant dynamics, predicate structure, and the logged candidate distribution also matter.

Under fixed settings documented in the artifact, FBS reduces oracle-labeled approval-expiry rates on recorded closed-loop trajectories in all five environments. In the audited DeltaIoT cell at $K{=}8$, it reaches 0% approval expiry while matching the *reference-open* (`e_open`) baseline's reported mean reward. The separate LLM audit finds nonzero judge-conditioned use-time invalidity in all four approval streams, although it does not isolate temporal expiry from check-time judge error.

FBS remains a proof-of-concept heuristic. Deployment-specific evaluation and calibration, together with end-to-end Judge+FBS evaluation on the same approval stream, remain future work. The broader contribution is the *freshness contract*: semantic approval at check time must be paired with an explicit validity-at-use rule and a justified fallback on expiry. An Execute-stage assurance mechanism that establishes only check-time correctness answers only half of the safety question.

References

[1] M. Bishop and M. Dilger, "Checking for race conditions in file accesses," *Computing Systems*, vol. 9, no. 2, pp. 131–152, 1996.

[2] D. Lilienthal and S. Hong, "Mind the Gap: Time-of-check to time-of-use vulnerabilities in LLM-enabled agents," arXiv preprint arXiv:2508.17155, 2025.

[3] J. O. Kephart and D. M. Chess, "The vision of autonomic computing," *Computer*, vol. 36, no. 1, pp. 41–50, Jan. 2003.

[4] IBM Corporation, "An architectural blueprint for autonomic computing," IBM Corporation, Autonomic Computing White Paper, Jun. 2005, third edition.

[5] M. U. Iftikhar, G. S. Ramachandran, P. Bollansée, D. Weyns, and D. Hughes, "DeltaIoT: A self-adaptive internet of things exemplar," in *Proceedings of the 12th IEEE/ACM International Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS)*, 2017, pp. 76–82.

[6] S. Tuli, G. Casale, and N. R. Jennings, "SimTune: Bridging the simulator reality gap for resource management in edge-cloud computing," *Scientific Reports*, vol. 12, no. 1, p. 19158, Nov. 2022.

[7] T. Vogel, "mRUBiS: An exemplar for model-based architectural self-healing and self-optimization," in *Proceedings of the 13th International Conference on Software Engineering for Adaptive and Self-Managing Systems (SEAMS)*, 2018, pp. 101–107.

[8] A. Marda, S. Kulkarni, and K. Vaidhyanathan, "SWITCH: An exemplar for evaluating self-adaptive ML-enabled systems," in *Proceedings of the 19th International Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS)*, 2024, pp. 143–149.

[9] M. Kruliš, T. Bureš, and P. Hnětynka, "Simdex: A simulator of a real self-adaptive job-dispatching system backend," in *Proceedings of the 17th Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS)*, 2022, pp. 167–173.

[10] T. Rebedea, R. Dinu, M. N. Sreedhar, C. Parisien, and J. Cohen, "NeMo guardrails: A toolkit for controllable and safe LLM applications with programmable rails," in *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*. Singapore: Association for Computational Linguistics, Dec. 2023, pp. 431–445.

[11] H. Inan, K. Upasani, J. Chi, R. Rungta, K. Iyer, Y. Mao, M. Tontchev, Q. Hu, B. Fuller, D. Testuggine, and M. Khabsa, "Llama Guard: LLM-based input-output safeguard for human-AI conversations," arXiv preprint arXiv:2312.06674, 2023.

[12] A. Cartagena and A. Teixeira, "Mind the GAP: Text safety does not transfer to tool-call safety in LLM agents," arXiv preprint arXiv:2602.16943, 2026.

[13] J. Li, M. Zhang, N. Li, D. Weyns, Z. Jin, and K. Tei, "Generative AI for self-adaptive systems: State of the art and research roadmap," *ACM Transactions on Autonomous and Adaptive Systems*, vol. 19, no. 3, pp. 13:1–13:60, 2024.

[14] A. Pang, M. Wang, M.-O. Pun, C. S. Chen, and X. Xiong, "iLLM-TSC: Integration reinforcement learning and large language model for traffic signal control policy improvement," arXiv preprint arXiv:2407.06025, 2024.

[15] R. D. Yates, Y. Sun, D. R. Brown, III, S. K. Kaul, E. Modiano, and S. Ulukus, "Age of information: An introduction and survey," *IEEE Journal on Selected Areas in Communications*, vol. 39, no. 5, pp. 1183–1210, May 2021.

[16] J. Palmerino, Q. Yu, T. Desell, and D. E. Krutz, "Improving the decision-making process of self-adaptive systems by accounting for tactic volatility," in *Proceedings of the 34th IEEE/ACM International Conference on Automated Software Engineering (ASE)*, 2019, pp. 949–961.

[17] L. Jiang, Z. Liu, H. Luo, and Z. Lin, "Atomicity for agents: Exposing, exploiting, and mitigating TOCTOU vulnerabilities in browser-use agents," arXiv preprint arXiv:2603.00476, 2026.

[18] U. Mehmood, S. Sheikhi, S. Bak, S. A. Smolka, and S. D. Stoller, "The black-box simplex architecture for runtime assurance of autonomous CPS," in *NASA Formal Methods*, ser. Lecture Notes in Computer Science, vol. 13260. Cham: Springer, 2022, pp. 231–250.

[19] S. Li and O. Bastani, "Robust model predictive shielding for safe reinforcement learning with stochastic dynamics," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2020, pp. 7166–7172.

[20] F. Wilcoxon, "Individual comparisons by ranking methods," *Biometrics Bulletin*, vol. 1, no. 6, pp. 80–83, 1945.

[21] J. Dean and L. A. Barroso, "The tail at scale," *Communications of the ACM*, vol. 56, no. 2, pp. 74–80, Feb. 2013.