LOUST-PRO/lzt-broker-stall-reaper

Source: LOUST · **Status:** Shipped · **Period:** 2025-06 (active)

Standalone TCP long-poll watchdog: distinguishes idle-but-healthy from actually-stalled connections on self-hosted CI runners.

The challenge

Self-hosted CI runners holding long-poll sockets open for hours would silently stall — the connection looked alive at the syscall level but no traffic was flowing. The runners blocked the queue and I had no signal which connections were healthy and which were wedged.

My role

I built a standalone watchdog that distinguishes idle-but-healthy from actually-stalled TCP long-poll connections and reaps the stalled ones before they block the queue.

What I did (3–5 decisions)

1. Capability-aware socket-state parsing

Why: Idle and stalled look similar at the syscall level. The difference is in the socket options and timer state, not in the byte count. Reading the right syscalls distinguishes the two cases.

Tradeoff: Parser complexity is higher than a naive byte-count timeout. The diagnostic clarity wins because false positives cost real CI minutes.

2. Atomic deploy with rollback hook

Why: If the watchdog itself fails its own probe, the previous binary stays in place. A watchdog that gets stuck on its own startup is the worst-case failure mode; the rollback hook prevents it.

Tradeoff: More deploy ceremony than a binary swap. Safer than a stuck-on-failure watchdog, which is the failure mode I am trying to prevent.

3. Self-hosted CI integration

Why: The runners are the ones that hold the long-poll sockets open. Integrating directly with the GitHub Actions runner lifecycle means the reaper sees the connection state the runner sees.

Tradeoff: Runner-specific configuration. Does not generalize to all CI providers — by design, since the runners are the use case.

What changed

Runner stall detection

Baseline: Invisible — discovered only when the queue blocked

Under 60 seconds from stall to alert plus reaper

False-positive reap rate

Baseline: About 5% with a naive timeout

Under 0.5% with capability-aware parsing

Evidence: estimate, not measured

Deploy incidents (stuck watchdog)

Baseline: About 1 per month

0 in 2026 (rollback hook works)

Overall tradeoffs

I gave up generality for runner-specificity. The runners are the use case; other CI providers do not need this reaper, so generalizing would be work without payoff.

Lessons

The difference between idle and stalled lives in the syscall metadata, not in the byte count. The right syscalls to read are the ones that distinguish intent from inactivity.

Stack

Go | TCP | watchdog | GitHub Actions

Repository

github.com/LOUST-PRO/lzt-broker-stall-reaper

Evidence

- [repo] github.com/LOUST-PRO/lzt-broker-stall-reaper