Tool-Calling LLM Agents Under Realistic MCP Faults: Scale Does Not Purchase Fault Tolerance

Antonio Ken Iannillo[1] Joshua Segun Owotogbe[2] Roberto Natella[3] Francesco Avallone[4] Kristina Kudryavtseva[1] Indika Kumara[2]

[1]SnT, University of Luxembourg [2]Tilburg University [3]Gran Sasso Science Institute [4]University of Naples Federico II

SEDAN SERVICES AND DATA MANAGEMENT

Interdisciplinary Centre for Security, Reliability and Trust

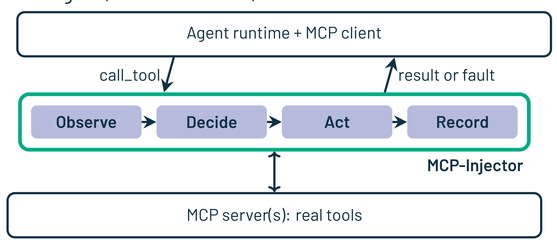
Agents trust their tools

Tool-calling agents act on the world through the **Model Context Protocol (MCP)**. Benchmarks assume every tool call returns the right answer, on time. Real MCP servers do not: they return errors, hang, and send malformed or wrong results.

Which agent designs degrade gracefully?
Does a larger backbone buy fault tolerance?

MCP-Injector

A drop-in middleware on the MCP tool-call path. No change to the agent, the MCP client, or the servers.



- **Six failure modes.** Content: minor mistake, incorrect output, invalid output, explicit error. Timing: delayed response, silent drop.
- **Three persistence regimes:** permanent, transient, intermittent.
- **Deterministic fault schedules:** seeded, every fired fault logged.
- **Non-intrusive:** with faults off, no metric changes (18 paired tests, all Bonferroni $p \geq 0.05$).

Faults the ecosystem reports

We mined public MCP server repositories and classified each failure report by **mode × persistence**.

1,774 MCP server repositories

103,166 issue and PR threads

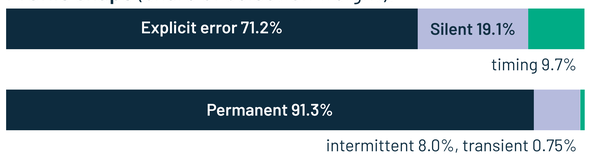
57,240 classified threads

14,829 failure reports

18-entry fault profile

GPT-4o classification; two annotators on 117 threads, Cohen's $\kappa = 0.93$ (mode), 0.80 (persistence).

Profile shape (share of selection weight):



Calibrated to the observed incident distribution: 18 selection weights and 6 intermittent activation probabilities come from the data. **Hand-set:** 30 s delay, perturbation intensities, deactivation 0.3, transient activation 1. GitHub issues over-represent persistent failures, so we treat this as a **stress profile**.

1,350 cells, fully crossed

2 backbones gpt-oss 20B, 120B × **3 stacks** MCP-Bench, OpenAI SDK, LangChain × **3 conditions** no proxy, OP0, OPP

× 25 MCP-Bench tasks × 3 seeds. **OP0**: proxy on, no faults. **OPP**: calibrated profile. Paired OPP vs. OP0 on each (task, seed); LLM judge scores 0 to 10.

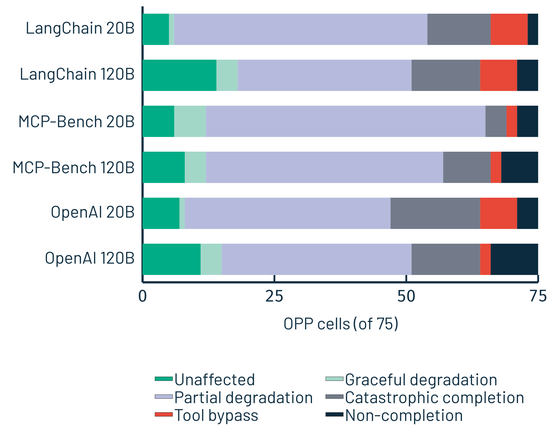
1 Faults cut quality by 60%

8.08 → 3.23

task-completion score out of 10, OP0 → OPP

Paired drop −4.85 points (95% CI [−5.22, −4.48]). All nine judge metrics fall **54 to 68%**; on runs that complete under both conditions, still −57%. Strongly task-dependent: live APIs −97%, short computations −19%.

2 Agents keep going, worse



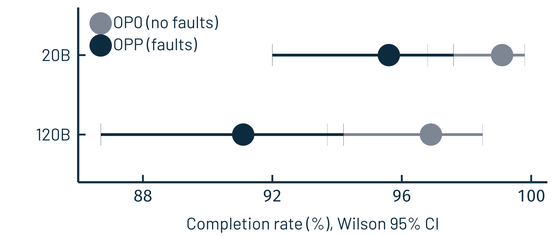
Partial degradation dominates (56% of cells): the run completes, with a much worse answer.

Tool bypass: plausible answers, no tool evidence

27 of 450 OPP cells (6%) answer well while barely using tools. A unit-conversion run computed everything from memory: fulfillment 9.8, tool appropriateness 1.2.

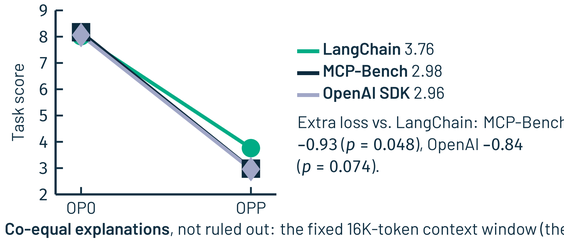
It happens **without faults too** (21 OP0 cells): faults expose a baseline behavior, and the judge rewards it. Counting only grounded answers, fulfillment falls 63%, not 58%.

3 Scale does not purchase fault tolerance



- **Resilience ratio** (OPP/OP0 score): model-size effect $\beta = -0.009$, 95% CI [−0.39, 0.37], $p = 0.96$.
- The 120B completes **fewer** tasks under faults (91.1% vs. 95.6%), but half the gap is already there without faults (96.9% vs. 99.1%).
- Fault-attributable drop: −5.8 vs. −3.6 points; the difference is not distinguishable from zero.

4 The agent stack matters (suggestive)



Extra loss vs. LangChain: MCP-Bench −0.93 ($p = 0.048$), OpenAI −0.84 ($p = 0.074$).

Co-equal explanations, not ruled out: the fixed 16K-token context window (the 120B writes longer outputs) and tool-call volume (MCP-Bench sees 8.6 fault events per cell vs. 1.8 to 2.3). One model family; transient and intermittent faults barely fired (9 of 1,905 events).

Capability benchmarks can hide operational brittleness. Test agents with protocol-aware fault injection.

Code, data, analysis:
doi.org/10.5281/zenodo.23573765

We are hiring!

CHAOS-MAS, funded by FNR: chaos engineering for multi-agent, tool-using LLM systems.
Open: one **PhD** and one **postdoc** position in the SEDAN group at SnT, University of Luxembourg.

Project and positions
how to apply

NeurIPS 2026, Main Track · antonioken.iannillo@uni.lu · Compute: SURF allocation EINF-16024

gitlab.com/uniluxembourg/snt/sedan/papers/mcp-injector