Self-Driving and Self-Healing Networks Using AI: Where Autonomy Is Claimed, Where It Is Evaluated, and What Runs in Production

Raghuram Malpe Pai[1], Krishna Suman Dara[1], Somesh Rahul[2], Amarnath G S[3], Shivaprasad Gali[3], Mohammed Kamran Syed[4]

IEEE Independent Researcher, Sunnyvale, CA, USA[1], IEEE Independent Researcher, Fremont, CA, USA[2], IEEE Independent Researcher, Bengaluru, India[3], IEEE Independent Researcher, Mountain House, CA, USA[4]

Corresponding author:Raghuram Malpe Pai, Email: raghuram.pai@ieee.org

A large literature now reports network systems that detect faults, localise them, propose fixes and act. What it does not report in one place is the relationship between where a system was evaluated and how much of the repair loop that evaluation closed. We screen 205 works and audit 101 against two axes applied jointly: the substrate a system was demonstrated on, from analytical models to a live production network, and the deepest stage of the fault lifecycle its evaluation reaches, from detection to execution with verification. Crossing them gives a result neither shows alone. Of the 22 works evaluated on production networks, every one stops at detection, diagnosis or advisory recommendation and none executes a remediation, while all 26 works that evaluate unsupervised execution or execution with verification run on analytical models, simulators, emulators, testbeds or replayed traces. The boundary is categorical rather than graded, and the stage between, execution behind a human approval gate, holds five works. This is a claim about published evidence and not about what operators run unpublished, and the corpus shows why the distinction bites: the only mechanism in it that repairs a production network is a rule-based auto-isolator described in three sentences of background in a paper about something else. Reading 20 of the load-bearing works in full shows part of it obscured by reporting practice: seven describe a deeper stage than they evaluate, and 11 describe a production setting while 5 evaluate on one. Gray degradation is addressed by 11 works and acted on by none, and no work in the corpus evaluates an automated defence against a route hijack. Reported success rates are not comparable, because benchmarks disagree on whether a repair that breaks something else counts as success; where that is scored, reported success falls from 57.8% to 25.5%. We

2026. In Mukesh Saraswat & Harish Sharma (eds.), *Emerging Technologies and Intelligent Systems: Theory, Applications and Innovations*, 25–50. Computing & Intelligent Systems, SCRS, India. DOI: https://doi.org/10.56155/978-81-975670-4-9-2

close with a six-field reporting checklist that would make the substrate and loop closure of a self-healing claim legible from its abstract.

Keywords: Self-healing networks, network fault management, autonomous networks, LLM agents, evaluation methodology, systematic review.

1 Introduction

Operators have wanted networks that repair themselves for as long as they have run networks too large to repair by hand. What changed recently is that the machinery arrived. Meza et al. (2018) measured failure at data-centre scale [67], and Gan et al. (2019) showed that learned models could anticipate performance faults in a microservice deployment before they surfaced [31]. Since 2023 the same problem has been taken up by language-model agents that read telemetry, form hypotheses and write configuration. Fifty-two of the 101 works audited here appeared in 2025 or 2026. The literature is now too large to hold in mind and too new for its claims to have settled.

What makes it hard to read as a whole is not its size. It is that the words at its centre have no fixed referent. A system is called self-healing, autonomous, zero-touch or self-driving, and a reader has to reach the evaluation section to learn which of those the experiment supports. Gu et al. (2026) present automatic configuration repair for production networks and report a 97.5% repair rate; the experiment runs in a control-plane simulator over a configuration snapshot, and nothing is pushed to a live device [35]. Papers of this kind are accurate about themselves in full text and illegible in summary.

Reviews of this area exist. Coronado et al. (2022) and Liyanage et al. (2022) organise zero-touch management by architecture, standard and enabler [18,55]. Asghar et al. (2018) taxonomise self-healing solutions for cellular networks into detection, diagnosis, compensation and recovery [8], and Yu et al. (2018) classify faults in software-defined networks [111]; both predate machine learning at scale and both sit inside a single domain. Ameur et al. (2026) is the newest incumbent, organising agentic network operations by control plane and by agent capability [5]. Table 1 sets 18 of them against three questions: does the survey carry a fault-class axis, an evaluation-substrate axis, and an axis for how far round the repair loop a system is evaluated. Two organise by fault class and both are domain-siloed; six more touch it without organising by it. No survey carries an evaluation-substrate axis at all. Five touch loop closure, and each treats it as a vocabulary for the fault lifecycle rather than as a coding of what a given evaluation reaches.

This survey applies two axes together: the substrate a system was evaluated on, running from analytical models to a live production network, and the deepest stage of the fault lifecycle the evaluation reaches, from detection to execution with verification. Neither axis is new on its own. Crossed, they produce a result that neither shows alone, and it is categorical rather than graded.

The reader gets five things.

1. A joint substrate and loop-closure audit of the 101 screened works, and the cross-tabulation that produces the central result: every work evaluated on a production network stops at detection, diagnosis or advisory recommendation, and every work that evaluates unsupervised execution was demonstrated somewhere else.
2. A measurement of the distance between what a paper describes and what its evaluation reaches, taken over the 20 works read in full, with the coding rule stated so that a reader can re-apply it.

3. A map of fault-class coverage against evaluation substrate, naming the classes whose execution evidence exists only away from production, and the one class no work in the corpus acts on at all.
4. A comparability finding: reported success rates here do not measure the same quantity, because benchmarks disagree on whether a repair that breaks something else counts as a success.
5. A six-field reporting checklist that makes the substrate and the loop closure of a self-healing claim legible from its abstract.

The paper does not claim to have found every relevant work; Section 2 reports the search, the stopping rule that never fired, and how much of the coding rests on abstracts.

Table 1: Existing surveys, reviews and tutorials closest in scope, and the axes each organises by. Sweep date 29 August 2026. Columns three to five ask whether the survey carries that axis at all, not whether it carries it well. None crosses evaluation substrate against evaluated loop closure, which is the cut this paper makes. Several of these sit outside the inclusion criteria of this survey, on domain, on mechanism or on both. They are cited here and nowhere else, as the incumbent literature the contribution is positioned against, and the last column states in each case what puts them outside it.

Survey	Venue, year	Fault	Substr.	Loop	What it does not cover
Coronado [18]	IEEE COMST, 2022	n	n	n	no fault-class axis
Liyanage [55]	JNCA, 2022	n	n	n	architecture and standards framing
Benzaid [11]	IEEE Network, 2020	n	n	n	magazine length
Asghar [8]	IEEE COMST, 2018	p	n	n	cellular only
El Rajab [79]	Computer Networks, 2024	n	n	n	AutoML-centric
Pang [76]	IEEE Access, 2020	n	n	n	intent, not fault
Sharma [86]	ACM CSUR, 2023	n	n	p	SLA and intent axis
Yu [111]	IEEE COMST, 2018	y	n	p	SDN only
Chiesa [17]	IEEE COMST, 2021	p	n	n	mechanism-centric and not AI
Adday [1]	Sensors, 2022	y	n	p	wireless sensor networks only
Chen [15]	IEEE Comm. Mag., 2022	p	n	n	optical only
Farmani [28]	arXiv, 2023	p	n	n	cellular cell-outage only
Hong [38]	IJNM, 2025	n	n	n	LLM-centric
Ameur [5]	arXiv, 2026	n	n	p	newest incumbent
Cruzes [19]	TechRxiv, 2025	p	n	n	optical only
Fang [27]	IEEE Network, 2024	p	n	n	magazine length
Yang [106]	IEEE TNSM, 2024	n	n	n	security-centric
Velasco [92]	JOCN, 2021	n	n	p	invited tutorial

y the survey organises by this axis; p it touches the axis without organising by it; n absent. Substrate axis present in 0 of 18; evaluated loop closure in 0 of 18.

2 Scope and review method

We searched between 20 and 29 August 2026 across four engines: Consensus, the FastTrack interface to OpenAlex, general web search with per-paper retrieval, and seven parallel agent sweeps run over distinct vocabulary sets. The time window for inclusion is 1 January 2018 to the cut-off of 29 August 2026. Work published before 2018 is admitted only as background, cited outside the corpus and counted in the reference budget rather than in the audit.

The population is work presenting or reviewing systems that detect, diagnose, decide on or act upon faults, failures or service degradations in communication networks, where at least one stage is performed by a machine-learning, reinforcement-learning or language-model mechanism. Frameworks, systems, benchmarks and measurement studies all qualify. Admitted venues are peer-reviewed journals and conferences and arXiv preprints in cs.NI, cs.AI, cs.DC and eess.SP. The exclusion codes were fixed before screening began, because the largest of them was predictable: a search for self-healing networks returns power-grid and manufacturing work at high rank, and it returns paper-mill output in unindexed journals. Writing the codes afterwards would have let the corpus choose them.

Screening produced the flow in Table 2, with every count regenerated from the corpus at the final drafting step rather than carried forward from the protocol.

Table 2: Screening flow and exclusion reasons, every count regenerated from the corpus rather than carried forward from the protocol, so a reader re-deriving them from Table 7 gets the same numbers. Of the 101 included, 17 were added by snowballing, 7 in the first round and 10 in the second, and none was coded from metadata alone. The 35 competing surveys are held outside the corpus deliberately, as the incumbent literature this paper is positioned against rather than evidence about systems.

Screening step	Works	Reason for exclusion	Works
Identified across four engines	205	Predatory or paper-mill venue	26
competing surveys, held out	35	No fault-lifecycle stage	21
excluded against the fixed codes	69	Off-topic domain	9
included in the audit	**101**	No AI or ML mechanism	7
coded from full text	20	Venue type not admitted	2
coded from abstract	81	Not peer reviewed, no identifier	2
		Outside the time window	1
		Full text unreachable	1
		Total excluded	**69**

Snowballing ran in three rounds: backward from six seed works, forward from those seeds, and a third searching specifically for counterexamples to the result reported in Section 7. The third round was run to break the finding, not to support it, and its yield is recorded inside the second snowball lane rather than separately, so Table 2 shows two snowball lanes and not three.

The stopping rule did not fire. We adopted the conventional criterion of stopping after two consecutive searches returning under 10% previously unseen work. Five tests were run and no consecutive pair met it. The first, ”outage mitigation automated recovery reinforcement learning communication network restoration”, returned 10 results and 0 on topic, drifting into power-

systems resilience; it measures the connector's lexical behaviour rather than this field's saturation, and we discarded it. The second, "agent diagnoses and repairs network incident deployed in production operator evaluation", returned 14 previously unseen works in 20. The third returned 1 in 10; its query string was not preserved in the search log. A fourth, "LLM agent root cause analysis network outage remediation", returned 4 in 10. A fifth, "autonomous network operations closed-loop incident resolution artificial intelligence", returned 0 of 10 on topic. The reading is narrow but worth stating. Generic vocabulary is saturated, and what yield remains comes only from searches that name a specific system. Searching stopped on a judgement about diminishing return, not on a criterion being met, and a reader re-running these strings will find the same thing.

Coding provenance is uneven by design. Twenty of the 101 works were read in full, chosen rather than sampled: the works whose own claims the audit turns on, being those that describe execution or a production setting, together with the benchmarks that define how execution is scored. Seventeen of the twenty describe execution or production and the remaining three are benchmarks. It is not a complete reading of any category, and the shortfall matters most where the paper's result is strongest: of the 22 works evaluated on production networks, 5 were read in full and 17 are coded from abstracts. The remaining 81 works in the corpus are abstract-coded. A claim that a particular work does *not* measure something is made only where that work was read in full, and Table 6 carries that subset explicitly; for abstract-coded rows the strongest available statement is that the abstract does not indicate it.

The direction of that shortfall is knowable even though its size is not. Every work whose production claim was tested against its full text lost ground. Two lost the production substrate outright, the network under test turning out to be a replayed configuration snapshot in one case and field-trial equipment carrying no customer traffic in the other; five more kept the substrate and lost loop depth, the evaluation measuring prediction, diagnosis or a dry run rather than an executed repair. Three works did move deeper on a full read, and all three are benchmark harnesses on testbeds or emulators, where moving deeper lengthens the non-production execution row and cannot touch the production one. No full-text read here has moved a work *into* the production-and-execution cell. Abstract coding therefore over-credits that cell rather than under-crediting it, and reading the remaining 17 abstract-coded production works in full could only empty the row further.

Four limits bear on how far the audit generalises. Coverage is bounded by the four engines. One returned oncology, water resources and spinal-cord injury for a review-type query put in this field's own words, and recovered only when the query was rephrased as zero-touch network and service management. That bounds the weight any single-connector saturation test can carry. arXiv listing pages were not retrievable from our fetcher, so the recency sweep ran through web search and per-paper abstract pages, weakening the claim to have seen everything from the final month. A fifth of the corpus is preprints. No preprint carries an argument here except where it was read in full, with one exception: nSync (Yang et al. 2025) is one of the four works evaluating verification and is coded from its abstract. Finally, the audit has a single coder. To bound what that costs, we re-coded the 53 rows the result turns on: the 22 works evaluated on production networks and the 31 whose evaluation reaches execution. The second pass worked from the recorded evidence fields with the original codes hidden, and applied only the rule stated in Section 3. Loop closure agreed on 48 of the 53 rows, 90.6%, Cohen's $\kappa = 0.87$; substrate agreed on 44, 83.0%, $\kappa = 0.78$. The disagreements fall in one place. Every loop-closure difference sat on the line between gated and unsupervised execution, which is a disagreement about how a

paper reports the operator's role rather than about whether an action was executed; no substrate difference moved a work into or out of the production row; and no difference on either axis put a work in the production-and-execution cell. Only one would change a reported number, moving a single work out of gated execution and taking that stage from five works to four. This is intra-rater agreement. It is not the independent second rater the axis deserves, which we were unable to arrange for this version. Table 7 carries the per-row codes, so a reader who disagrees can re-apply the rule and see which rows a different judgement would move.

3 Background and the taxonomy

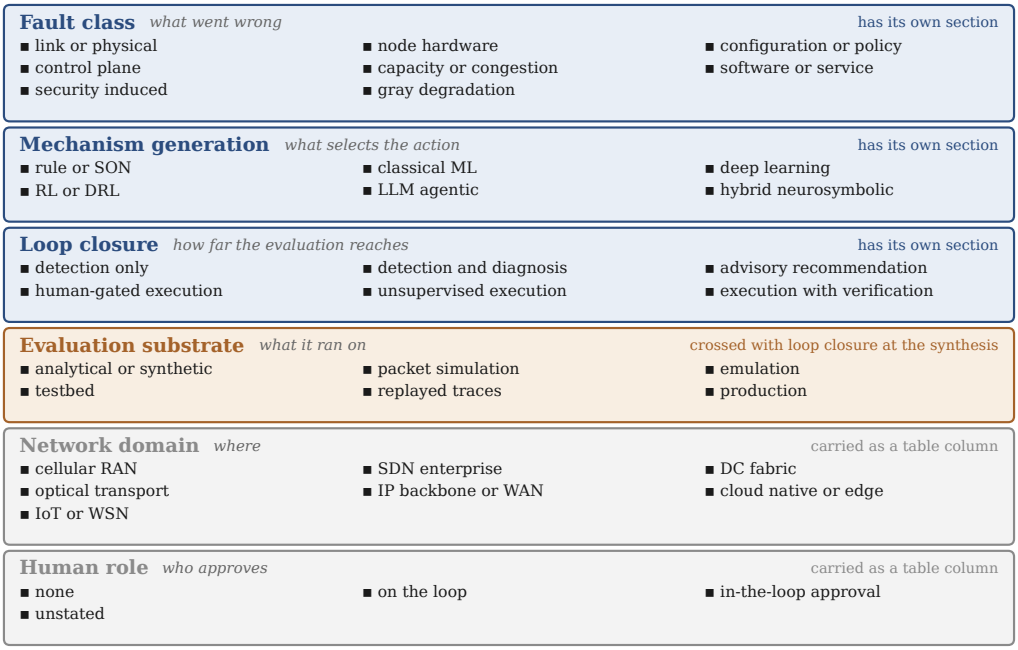
We use one vocabulary throughout. A *fault* is any condition degrading or interrupting service, whether or not it sets off an alarm. *Detection* is deciding that something is wrong, *diagnosis* is deciding what and where, and *remediation* is the change that restores service. *Verification* establishes afterwards that service was restored and nothing else broke; *rollback* undoes the change when it was not. These are stages, not components.

Figure 1 gives the six axes: three take a section below, one is crossed against loop closure at the synthesis, and two are table columns. The axes are not mutually independent, and two dependencies matter. Language-model mechanisms arrived recently and concentrate in the fault classes legible as text, and the substrate a group can reach constrains how far round the loop it can evaluate. The paper is about the second.

Loop closure is coded on what the paper evaluates, not on what it describes. A work is placed at the deepest stage its reported experiment actually reaches. A system that issues a command in an architecture diagram and is measured on classification accuracy is coded at diagnosis. The rule matters because it moves papers. Mukkolakkal et al. (2026) title their work with the claim that no human is required, and describe a full deploy, calibrate, monitor and heal cycle with a safety guard. The flagship eighteen-hour outage is then walked through as a sequence of diagnostic steps, and the 300 repair cycles the paper reports carry no success or failure breakdown [71]. We code it at detection and diagnosis. Applying the rule requires the evaluation section, which is why Section 2 reports how much of the corpus was read in full.

Substrate is coded on where the reported evaluation ran, not on what the system targets. Analytical or synthetic covers closed-form models and generated workloads; packet simulation covers discrete-event simulators; emulation covers virtual topologies built from real network software; testbed covers assembled infrastructure, including field trials on real carrier equipment not carrying customer traffic; replayed traces covers operator incident records processed offline. *Production* means a live operational network carrying real traffic at the time of the experiment. That boundary is strict on purpose. Gu et al. (2026) evaluate configuration repair against an 83-device snapshot taken from a production network and replayed in a control-plane simulator [35]: the configurations are production, the network under test is not, and the distinction is what makes the cross-tabulation in Section 7 mean anything.

Three further axes were considered and set aside, as Figure 1 records: a system autonomy level, digital-twin fidelity, and an audit of what safety properties a system establishes before acting. We record only whether a paper evaluates verification or rollback, never what safety property it establishes.



Axes considered and deliberately not used: system autonomy level, digital-twin fidelity, and any audit of safety properties or recoverability after an executed change. Each is a body of work in its own right, and each is cited here rather than re-derived.

Figure 1: The six axes, their value sets, and the role each plays in the paper. Fault class, mechanism generation and loop closure each take a section. Evaluation substrate deliberately does not: it enters at the synthesis crossed against loop closure, because the joint cross-tabulation is what produces the result and splitting it across two sections would destroy it.

4 Fault classes: what is being healed

The fault class is what went wrong, coded as the authors characterise the fault their system was evaluated on. A work evaluated across several classes is counted in each. The axis separates literatures that rarely cite one another. Optical soft-failure identification, cellular cell-outage detection, data-centre link reliability and microservice root-cause analysis work the same lifecycle on different faults, and putting them in one table is most of what this axis returns. Table 3 crosses it against evaluation substrate, giving in each cell the number of works and, in parentheses, how many of those evaluate unsupervised execution or execution with verification.

The distribution follows the data rather than the difficulty. Software and service faults dominate at 34 of 101, because microservice deployments emit logs, traces and metrics a language model can read, and because public incident datasets exist for them. Configuration and policy faults follow at 22, and are where verified execution concentrates: three of the four works that evaluate the state after their change are configuration repair. Configuration is text. A proposed repair is checkable before it is applied, and formal verification of the result is available as it is not for a physical fault. Node hardware and link faults account for 16 and 13. Both draw on produc-

tion data more heavily than any other class, 6 of 16 and 7 of 13, and both sit mostly at detection and diagnosis, with four works each reaching execution.

Two absences are more informative than the distribution. The first is gray degradation, the partial or silent failure that sets off no alarm: 11 works, of which none evaluates unsupervised execution. This is the class where autonomy would help most, because the human cost is the hours spent noticing the problem at all, and it is the class where the corpus contains no evaluated action. Every work in it stops at detection, diagnosis or advisory recommendation, and only two reach advisory: Yu et al. (2024) recommend on a production cloud monitoring service [112] and Wang et al. (2024) generate optical-fault responses from a language model [94]. The deepest any gray-degradation work reaches on a production network is diagnosis, as Yao et al. (2025) do when localising irregularities in production training clusters [109]. No abstract in the class indicates an evaluated action.

The second is control-plane routing. Six works address it and three evaluate unsupervised execution, but the two that concern inter-domain routing are both coded at detection, and neither abstract indicates an evaluated response: Testart et al. (2019) profile serial hijackers from historical announcements [89] and Moriano et al. (2021) detect routing anomalies from announcement burstiness [70], both over replayed archives. A search directed specifically at automated mitigation of route hijacks returned nothing that acts. The corpus contains no work that evaluates an AI mechanism executing a defence against a BGP hijack, on any substrate. Security-induced faults are thin for a related reason: 5 works, 4 of which evaluate execution, two inside cyber-defence gyms, one on an SDN testbed and one on a synthetic intrusion-response model [29, 37, 45, 114], none on an operational network.

What this axis does not settle is whether the pattern reflects difficulty or data. Configuration faults may execute more often because configuration is genuinely safer to change, or because configuration is the fault class for which a benchmark exists. The axis cannot separate those, and Section 7 does not try; it asks the narrower question of where the execution evidence sits, which the substrate axis can answer.

Table 3: Fault class against evaluation substrate. Each cell gives the number of works, and in parentheses how many of those evaluate unsupervised execution or execution with verification. Every parenthesised count sits left of the production column.

Fault class	Analyt.	Pkt sim	Emul.	Testbed	Traces	Prod.	Works
Software or service	3 (1)	1 (1)	2 (1)	6 (5)	15 (3)	8	34
Configuration or policy	6 (2)	1 (1)	3 (2)	3 (1)	4 (2)	5	22
Node hardware	9 (3)	0	0	0	0	6	16
Link or physical	2 (1)	0	3 (2)	1 (1)	0	7	13
Gray degradation	4	1	1	3	1	2	11
Capacity or congestion	4 (3)	0	2 (2)	2 (1)	0	4	12
Control plane	1 (1)	0	1 (1)	1 (1)	3	0	6
Security induced	3 (3)	0	0	1 (1)	1	0	5
n/i	2	0	2	0	1	0	7

Substrate keys as in Table 5. The final column counts distinct works and is the figure quoted in the prose. The cells do not sum to it: a work with two substrates appears in two cells, and a work whose substrate the paper does not state appears in none, that column being omitted here and carried in Table 4.

5 Mechanism generations: what does the healing

The mechanism generation is what selects the action, not what detects the anomaly. A pipeline whose detector is a neural network and whose remediation is a static playbook is coded as rule or SON, because the playbook chooses. The axis is ordered by arrival rather than by capability, and the generations overlap: classical machine learning appears across the whole window, from Shariati et al. (2018) identifying failures from the optical spectrum [84] to work published this year.

Figure 2 puts the generations on a timeline. Language-model agents dominate the recent literature: 45 of 101 works use one, none before 2023, and 31 of those 45 appeared in 2025 or 2026. They also account for 11 of the 22 production-substrate works, the highest of any generation. The reason is prosaic: an agent that reads logs and writes a recommendation needs read access and a person to read the output, which an operator can grant. Reinforcement learning is the mirror image. Nine works use it, seven evaluate execution, and not one is evaluated on a production network. It needs an environment that can be reset and rolled forward thousands of times, and a production network is exactly the environment that cannot.

This is the finding of the mechanism axis, and Figure 3 is the shape of it: *the generation that acts and the generation that reaches production are not the same generation.* Foley et al. (2022) and Kiely et al. (2025) train agents that defend a network autonomously, inside cyber-defence gyms [29, 45]. Lacava et al. (2025) execute deep-learning-driven control on a RAN node at sub-millisecond latency on an OpenAirInterface testbed [49]. Meanwhile Wang et al. (2025) run BiAn, a language-model failure localiser, beside live operators for ten months [93], and Yang et al. (2025) run SkyNet, a classical alert-correlation system, on a large cloud for eighteen months [105]. Both produce diagnoses for humans. The two literatures are answering different questions and are frequently read as answering the same one.

Classical machine learning holds 26 works, mostly at detection and diagnosis with seven on production data; deep learning holds 11, five of them detection only. Neither is displaced so much as re-scoped: newer systems use them as components and take the decision elsewhere. Hybrid and neurosymbolic approaches remain at five, with a median year of 2026.

The axis does not settle whether the substrate split is caused by the mechanism or merely correlated with it. Reinforcement learning may be absent from production because it needs a resettable environment, or because the groups with production access are not the groups using it. Section 8 treats that as an open question rather than a settled explanation.

6 Loop closure: how far round the loop the evaluation reaches

Loop closure is the axis the rest of the paper rests on, and Section 3 gave its placement rule: the deepest stage the reported experiment reaches, not the deepest the paper describes. Table 4 gives the distribution against substrate; this section reads the loop-closure margin of that table.

Thirteen works stop at detection, deciding that something is wrong without saying what [2, 9, 13, 14, 23, 44, 47, 48, 64, 67, 70, 89, 110]. Thirty-six reach diagnosis [6, 12, 20, 22, 26, 30, 33, 34, 42, 46, 52, 53, 57–59, 71, 77, 80, 81, 83, 84, 87, 88, 93, 95–99, 102, 104, 105, 107, 109, 115, 118], the largest group, and the concentration is not an accident of coding. Root-cause analysis for microservices has public datasets, an agreed output format and several benchmarks, which makes it the stage where a new method can be compared against a previous one. Nineteen reach advisory recommendation,

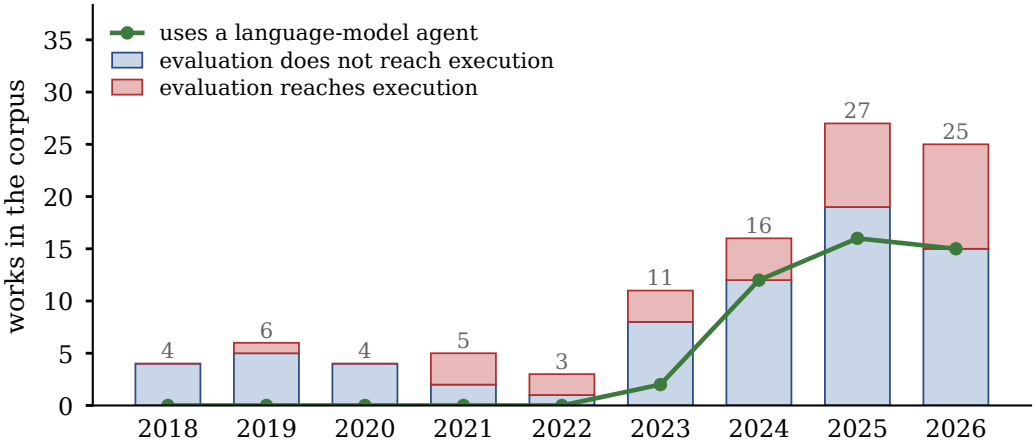


Figure 2: Works per year, split by whether the reported evaluation reaches execution, with the language-model share drawn over the top. Execution results are not new; what is new is their volume, and it arrives with the agent wave rather than before it. The 2026 column is partial, the cut-off being 29 August 2026.

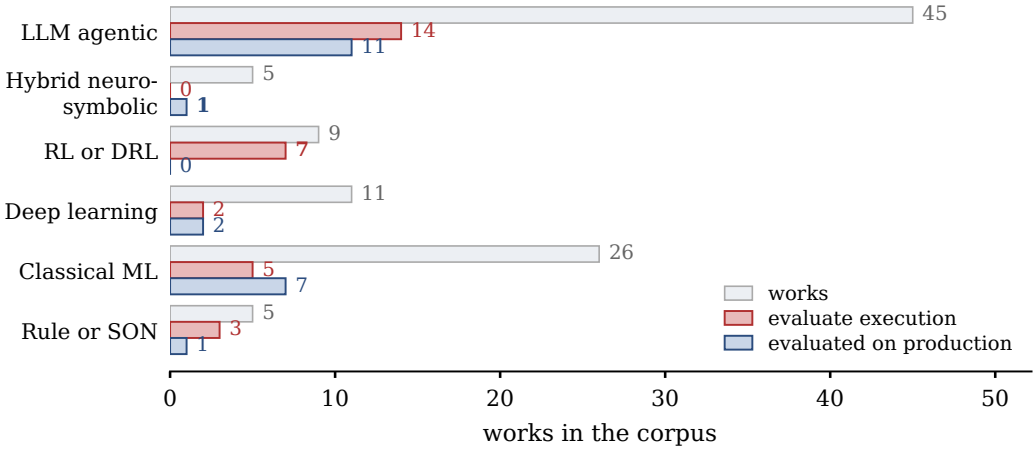


Figure 3: Each mechanism generation against the number of its works that evaluate execution and the number evaluated on a production network. The two bars pull apart. Reinforcement learning has the highest execution rate in the corpus and no production evaluation at all; language-model agents have the most production evaluations and reach execution in fewer than a third of their works.

producing a fix for a person to read [3,4,21,31,32,35,36,41,62,69,72,90,94,100,103,112,113,116,117]. The three stages together account for 68 of 101 works.

The remaining 33 divide unevenly, and the shape of that division is the second result of this paper. (Two further works describe a loop but report no evaluation from which a stage can be read [61, 101].) Twenty-two works evaluate unsupervised execution [16, 24, 29, 37, 40, 43, 45, 49, 51, 54, 56, 60, 63, 65, 68, 73, 74, 82, 85, 91, 114, 119] and four evaluate execution with verification or rollback [7,10,78,108]. *Five* evaluate execution behind a human approval gate [25,39,50,66,75]. The stage that a cautious deployment path would treat as the necessary middle step is the emptiest stage on the axis. The literature is bimodal: it either hands a recommendation to an operator or removes the operator entirely, and it does not populate the intermediate position where a machine proposes an action and a person authorises it before it runs.

The coding of the human role confirms this rather than softening it. Of the 22 works evaluating unsupervised execution, 21 record no human role at all in the evaluated loop. Of the 19 advisory works, 15 record a human in the loop with approval, so autonomy in this literature means the absence of a person rather than a person supervising at a distance. On-the-loop supervision, the arrangement operators describe wanting, appears 12 times and never once alongside unsupervised execution.

Verification is rarer still. Four works evaluate the state after the change. Avgerinos et al. (2025) report a post-action validation step on a testbed and Yang et al. (2025), nSync, report pass@3 on replayed infrastructure-as-code [10,108], both coded from abstracts. The two that explicitly score what a repair broke come from a single research group sharing one benchmark, which for the purposes of counting evidence is one source rather than two [7,78]. Of the 31 works that execute anything, 27 report whether the action was taken and not whether the network was left in a better state than it started. The other 70 take no action, so for them the question does not arise.

What this axis does not settle is why the middle is empty. Two explanations fit the data equally well. Human-gated execution may be awkward to evaluate, because measuring it needs operators in the loop for the duration of a study and produces a result confounded by their judgement. Or it may be unpublishable. It shows neither a working autonomous system nor a clean measurement, so groups that build it do not write it up. The corpus cannot distinguish these, and the distinction matters, because one is a research gap and the other is a publication artefact that hides deployed practice. Section 7 shows that the same ambiguity governs the paper's central result.

7 Cross-cutting synthesis: substrate against loop closure

Neither axis alone says anything surprising. Crossed, they produce Figure 4 and Table 4, and one row of that table is the paper.

Twenty-two works are evaluated on a production network. Every one of them stops at detection, diagnosis or advisory recommendation. None executes a remediation: not autonomously, and not behind a human approval gate. The production row is empty from human-gated execution rightward. Meanwhile 26 works evaluate unsupervised execution or execution with verification, and all 26 run on analytical models, packet simulators, emulators, testbeds or replayed traces. The boundary is not a gradient with production at the sparse end. It is categorical, and Figure 4 is what it looks like.

Table 5 reads that row work by work, and the works there are not marginal. Su et al. (2026) ran a multi-agent system against Baidu's optical infrastructure over seven months of telemetry [88].

Table 4: Evaluation substrate against the deepest fault-lifecycle stage the paper *evaluates*. Rows with more than one substrate are counted once per substrate, so column totals exceed the corpus size. The production row is empty from human-gated execution rightward.

Substrate	Det	Diag	Adv	Gated	Auto	Ver	n/i
Analytical or synthetic	3	8	6	1	10	2	0
Packet simulation	0	1	0	0	1	0	0
Emulation	1	2	0	0	4	0	1
Testbed	2	2	1	2	7	1	0
Replayed traces	2	11	4	1	3	1	0
Production	4	12	6	**0**	**0**	**0**	0
n/i	1	3	2	1	0	0	1

Det detection only; Diag detection and diagnosis; Adv advisory recommendation; Gated execution behind human approval; Auto unsupervised execution; Ver execution with verification or rollback; n/i the paper does not indicate.

Wang et al. (2025), BiAn, ran failure localisation in shadow mode beside live operators for ten months, reporting 95.5% top-1 accuracy and a 20.5% reduction in time to resolution [93]. Yang et al. (2025), SkyNet, ran alert correlation on a large cloud for eighteen months and cut median time to mitigation from 736 to 147 seconds [105]. Serious deployments with serious measurements, and not one measures an executed repair.

Part of the boundary is concealed by reporting practice, and Table 6 is the measurement. Figure 5 draws it. Of the 20 works read in full, 7 describe a deeper stage than their evaluation reaches. Thirteen describe execution; 6 evaluate it; none of those 6 on a production network. Eleven describe a production setting; 5 evaluate on one. The gap is not a matter of authors overselling in an abstract and correcting themselves in the body, because in most of these cases the body is accurate. It is that the paper's title and framing carry the deeper claim and the evaluation section carries the shallower fact, and only the second is load-bearing.

Three cases show its shape. Gu et al. (2026) title their work automatic configuration repair for production networks and report 97.5% of injected faults repaired; the experiment replays an 83-device production configuration snapshot inside an in-house control-plane simulator, and the two naturally occurring incidents produced suggestions rather than applied changes [35]. Wang et al. (2025), Confucius, present a multi-agent framework running at hyperscale across more than sixty applications, whose write path is an explicit dry run that simulates writes without committing them and hands the configurations it produces to a separate deployment system [100]. D'Antonio et al. (2025) study 232 incidents over 17 weeks on a live production network, and the human performs every state-changing action, which the authors present as a design choice rather than a limitation [21]. Read as abstracts, these are three systems repairing production networks. Read in full, none executes.

One work in the corpus reports something that does execute in production, and it is not an AI system. Yang et al. (2025) mention, as background rather than as contribution, a pre-existing rule-based auto-isolator that removes failing devices from service and prepares rollback plans on the operator's live network [105]. It is not evaluated anywhere in that paper, it is not learned, and it is out of scope for a claim about AI mechanisms. It is also the only thing in this corpus that repairs a production network. The thing that runs is the thing nobody published.

The reported success rates are not comparable, and this is a separate finding. Benchmarks disagree about what counts as success. Protogeros et al. (2026) score a repair as successful only when it fixes the fault and introduces no new violation, and report that this moves the strongest model from a 57.8% fix score to 25.5% inclusive success; the entire difference is repairs that break something else [78]. Zhou et al. (2025) score every intermediate state against topology invariants and report agents that look effective on end-state correctness while violating constraints along the way [119]. Jha et al. (2025) give agents real cluster access and report an 11.43% mitigation rate for GPT-4o [40]; Chen et al. (2025) report 54.55% on a comparable arrangement [16]. Placing 11.43 beside 54.55 beside 97.5 invites a ranking that the numbers do not support, because the three measure different events on different substrates under different definitions of success. Two of the four verification-capable works come from one group and share one benchmark [7, 78], so the apparent agreement between them is one result, not two.

The last column of Table 6 is the audit that follows from this, abridged there and counted here over the full field. Across the 19 full-text works carrying a recorded omissions field, four recur: no rollback or post-change recovery measure, in 14 works; no adequate human baseline, in 13; no measure of what an action affected beyond its target, in 11; and no false-positive or unnecessary-action rate, in 10. This paper records those omissions and stops there; what safety property each system establishes before acting, and what remained recoverable afterwards, is a different audit against a different rubric and belongs to companion work.



Figure 4: The audit. Evaluation substrate against the deepest fault-lifecycle stage each work's evaluation reaches, over all 101 works in the corpus, and drawing the same cells as Table 4. The production row is outlined and its three cells to the right of the dashed line are empty. Works carrying more than one substrate value are counted once per value, so the cells sum to 107 rather than 101. The final row and column hold the works whose substrate or whose evaluated stage the paper does not state.

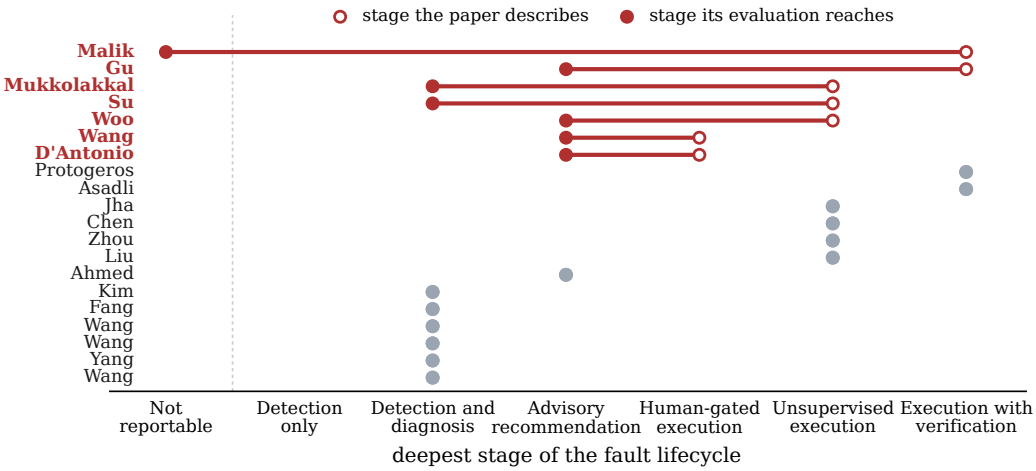


Figure 5: The 20 works read in full: the deepest loop stage each paper *describes* against the deepest stage its evaluation *reaches*. A dot means the two agree. A line means they do not, and its length is the distance between what the paper says and what it shows. Seven diverge. Malik (2026) runs the width of the axis because its evaluation section reports no data from which any stage can be read.

8 Open challenges and research agenda

One concession shapes everything below. This is a finding about a published evidence base, not about what exists inside operators, who do not publish their automation. It still binds, for a reason the corpus shows: the one production auto-isolator anywhere in these 101 works appears as three sentences of background in a paper about something else [105]. A field cannot build on evidence it cannot see.

1. **Is the empty middle a research gap or a publication artefact?** Five works of 101 evaluate execution behind a human approval gate. A study instrumenting an existing approval workflow and reporting the fraction of proposed actions approved, rejected and modified would settle whether gated execution is unevaluated or merely unpublished. It needs operator cooperation, not a new model.

2. **Can anything act on a fault that sets off no alarm?** Eleven works address gray degradation and none evaluates an action on it. A silent failure has no ground truth, so a study here has to build the labelled set first: naturally occurring degradations, independently confirmed, with the time at which each became detectable.

3. **Why is inter-domain routing detected and never defended?** No work in this corpus evaluates an AI mechanism executing a defence against a route hijack, on any substrate. Detection is solved well enough to be productised; what a study needs first is agreement on what a safe automated response to a suspected hijack even is.

4. **Is reinforcement learning absent from production because of the method or the access?** Seven of nine reinforcement-learning works evaluate execution and none touches a pro-

Table 5: Every work in the corpus evaluated on a production network, ordered by how far round the loop its evaluation reaches. The rightmost stage any row reaches is advisory recommendation; the human-gated, unsupervised and verified columns of Table 4 are empty for this substrate. Headline figures are reproduced as each paper reports them and are not comparable across rows, for the reasons given in Section 7. Each entry names the quantity as well as the value, because the papers report different quantities; an entry of none stated means the work reports no headline figure at all.

Work	Yr	Fault class	Loop	Reported headline
Meza [67]	2018	LNK HW	Det	none stated (measurement study)
Chakraborttii [13]	2020	HW	Det	up to 1.3× prediction accuracy
Ahmad [2]	2024	LNK	Det	none stated
Deng [23]	2024	HW	Det	precision 0.904, F1 0.893
Roy [80]	2024	SW	Diag	qualitative gain only, no figure
Zhang [118]	2024	SW	Diag	+24.8% over fine-tuned baseline
Arzani [6]	2025	LNK CAP	Diag	1.9× to 10.9× more degradation found
Liu [53]	2025	LNK HW CFG	Diag	precision 98.2%, recall 99.3%
Luo [59]	2025	SW	Diag	state of the art claimed on OpenRCA
Pei [77]	2025	SW	Diag	accuracy 64.01% vs 35.50% baseline
Wang [93]	2025	HW CAP CFG	Diag	top-1 95.5%, time to resolution −20.5%
Yang [105]	2025	LNK HW CFG CAP	Diag	median time to mitigation 736 s to 147 s
Yao [109]	2025	GRAY	Diag	localisation accuracy 97.21%
Cui [20]	2026	SW	Diag	up to 6.3× accuracy vs ReAct
Su [88]	2026	LNK	Diag	F1 97.7%, denominator not stated
Yang [107]	2026	SW	Diag	+19.8% diagnostic accuracy
Ge [32]	2023	CFG	Adv	none stated
Namyar [72]	2023	LNK CAP	Adv	orders-of-magnitude flow-completion gain
Jiang [41]	2024	SW	Adv	none stated
Yu [112]	2024	GRAY	Adv	none stated
D'Antonio [21]	2025	SW	Adv	232 incidents over 17 weeks (study scale)
Wang [100]	2025	CFG	Adv	17 engineer hours saved per week

22 works, of which 5 were read in full. Fault-class keys: CAP capacity or congestion, CFG configuration or policy, GRAY gray degradation, HW node hardware, LNK link or physical, SW software or service. Stage keys as in Table 4.

duction network. Separating a technical constraint from an institutional one needs a study offering the same fault set to a reinforcement-learning system and a language-model agent under the same access conditions.

5. **What does a repair cost when it is wrong?** Of the 31 works that execute an action, only four evaluate the state after the change and only two score whether the repair broke something else. For the other 27 no abstract or evaluation section indicates such a measure, and 11 of the 19 read in full with a recorded omissions field say so explicitly. Protogeros et al. (2026) show what changes when this is scored: inclusive success falls from 57.8% to 25.5% [78]. Applying that scoring to the existing benchmarks, rather than building another one, would make the current results comparable for the first time.

6. **Can a benchmark be built on faults nobody injected?** Almost every result here rests on injected faults, so far as the papers state it; the corpus does not code fault provenance systematically, which is why field three of the checklist below asks for it. Kim et al. (2026) is

Table 6: The works read in full, with the deepest loop stage each *describes* beside the deepest its evaluation *reaches*, and the same for evaluation substrate. Rows are ordered with the divergent rows first, loop-closure divergences before substrate-only ones. The last column is the evidence audit, abridged: what the paper does not measure. Every row here was read in full, which is what licenses that column; no negative in this table rests on an abstract.

2*Work	2*Yr	Loop closure		Substrate		2*What the paper does not measure
		claim	eval.	claim	eval.	
Gu [35]	2026	**Ver**	**Adv**	**Prod.**	**Analyt.**	no false-positive rate; no rollback (nothing deployed); no blast radius; no human baseline; only n=2 naturally occurring incidents
Malik [61]	2026	**Ver**	**n/i**	**Prod.**	**n/i**	no sample size or measurement window for any figure; no rollback rate; no blast radius; no numeric baseline; no ablation *[abridged]*
Mukkolakkal [71]	2026	**Auto**	**Diag**	**Prod.**	**n/i**	no success/failure breakdown for the 300 cycles; no false-positive rate; no rollback rate; no blast radius; no human baseline; no ablation *[abridged]*
Su [88]	2026	**Auto**	**Diag**	Prod.	Prod.	no denominator for the 97.7% F1; no false-positive rate; no blast radius; baseline value not given *[abridged]*
Woo [103]	2025	**Auto**	**Adv**	**Testbed**	**Analyt.**	none: the full-text read found nothing unmeasured worth recording
D'Antonio [21]	2025	**Gated**	**Adv**	Prod.	Prod.	no rollback (no automated action); no hallucination or unnecessary-suggestion rate despite the paper anecdotally conceding hallucination *[abridged]*
Wang [100]	2025	**Gated**	**Adv**	Prod.	Prod.	no rollback; no unnecessary-change rate; no blast radius; no approval or rejection fraction *[abridged]*
Liu [54]	2024	Auto	Auto	**Prod.**	**Testbed**	no false-positive rate; no blast radius during autonomous re-tuning; human baseline only qualitative; no failure-case analysis; no limitations section
Ahmed [3]	2023	Adv	Adv	**Prod.**	**Traces**	no execution of any kind; no rollback; no false-positive or unnecessary-action rate; no blast radius *[abridged]*
Wang [99]	2024	Diag	Diag	**Prod.**	**Traces**	no execution possible by construction; no rollback; no action false-positive rate; no blast radius; no action-outcome human baseline
Asadli [7]	2026	Ver	Ver	Analyt.	Analyt.	no standalone false-positive rate; no human baseline; no real-infrastructure rollback; 100% injected faults *[abridged]*
Protogeros [78]	2026	Ver	Ver	Analyt.	Analyt.	no live deployment; no naturally occurring faults; no human baseline; diagnosis judged by LLM panel not formal methods
Chen [16]	2025	Auto	Auto	Testbed	Testbed	no explicit regression or invariant-violation scoring; no rollback; no differentiated wrong-action penalty
Jha [40]	2025	Auto	Auto	Testbed	Testbed	no regression tracking; no rollback; no invariant-violation check; no differentiated penalty for harmful action vs no answer *[abridged]*

20 works read in full; 7 diverge on loop closure and seven on substrate, overlapping on four, so ten of the twenty diverge on one axis or the other. Bold marks a divergence. Where a paper records more omissions than the column fits, the entry ends *[abridged]* and the full field is counted in Section 7. Stage abbreviations as in Table 4.

Table 6: (continued).

2*Work	2*Yr	Loop closure		Substrate		2*What the paper does not measure
		claim	eval.	claim	eval.	
Zhou [119]	2025	Auto	Auto	Emul.	Emul.	no rollback mechanics; safety checks intermediate-state invariants not recovery-from-error
Fang [26]	2026	Diag	Diag	Traces	Traces	no mitigation or action at all; no safety, damage or rollback metric; no contamination discussion
Kim [46]	2026	Diag	Diag	Traces	Traces	no action or mitigation; no safety metric; no contamination handling
Wang [93]	2025	Diag	Diag	Prod.	Prod.	no rollback; no unnecessary-action rate; no blast radius (nothing executed); human-only baseline explicitly disclaimed as impractical to collect
Wang [102]	2025	Diag	Diag	Emul.	Emul.	no mitigation evaluation at all - authors state this explicitly as future work; no safety metric
Yang [105]	2025	Diag	Diag	Prod.	Prod.	no rollback rate; no blast radius; no human-only diagnosis baseline (anecdotal 'several hours' only) *[abridged]*

20 works read in full; 7 diverge on loop closure and seven on substrate, overlapping on four, so ten of the twenty diverge on one axis or the other. Bold marks a divergence. Where a paper records more omissions than the column fits, the entry ends *[abridged]* and the full field is counted in Section 7. Stage abbreviations as in Table 4.

the clearest exception, using 335 naturally occurring incidents, and reports the hardest diagnosis results in the corpus [46]. The gap between injected and natural fault performance is unmeasured, and it decides whether any of these numbers transfer.

Five of these six are blocked on measurement rather than on method, which is where the next increment of progress is available. To make it available, a paper reporting a self-healing result should state six things, and they fit in a table caption:

1. the *substrate* the evaluation ran on, distinguishing a production network from a production artefact replayed elsewhere;
2. the *deepest loop stage measured*, stated separately from the deepest stage described;
3. the *provenance of the faults*, injected, replayed or naturally occurring, with counts for each;
4. the *number of actions applied* and how many succeeded, failed, or turned out to be unnecessary;
5. what *else changed state* as a result, and whether anyone checked;
6. the *baseline*, being what a human operator or the incumbent automation achieved on the same faults.

Fields one and two would have prevented every divergence in Table 6: seven of loop stage and seven of substrate, overlapping on four, across ten of the twenty works. They cost two sentences.

9 Conclusion

We screened 205 works on AI-driven fault detection, diagnosis and remediation in communication networks and audited 101 of them against two axes applied together: the substrate an evaluation ran on, and the deepest stage of the fault lifecycle that evaluation reached. Neither axis is new. Crossing them produces a result that is categorical rather than graded. Every work in the corpus evaluated on a production network stops at detection, diagnosis or advisory recommendation, and every work that evaluates unsupervised execution was demonstrated on an analytical model, a simulator, an emulator, a testbed or a set of replayed traces.

Reading twenty of these works in full shows part of the boundary being obscured rather than crossed. Seven of the twenty describe a deeper stage than they evaluate. Thirteen describe execution and six evaluate it, none on a production network. Eleven describe a production setting and five evaluate on one. The distinction between what a system does and what an experiment measured is the distinction this survey is organised around, and it is not currently legible from the places most readers look.

The result is about published evidence and not about deployed practice, and the corpus shows that limit itself: the only mechanism in it that repairs a production network is a rule-based auto-isolator described in three sentences of background, never evaluated on its own. What the field can compare is what the field has published. A reporting convention that separated substrate from claim, and measured stage from described stage, would cost authors two sentences and would let the next survey of this area answer a question this one could only pose.

References

[1] Ghaihab Hassan Adday, Shamala Subramaniam, Zuriati Ahmad Zukarnain, et al. Fault tolerance structures in wireless sensor networks (wsns): Survey, classification, and future directions. *Sensors*, 2022.

[2] Abdelrahim Ahmad, Peizheng Li, Robert J. Piechocki, et al. Anomaly detection in offshore open radio access network using long short-term memory models on a novel artificial intelligence-driven cloud-native data platform. arXiv preprint arXiv:2409.02849, 2024.

[3] Toufique Ahmed, Supriyo Ghosh, Chetan Bansal, et al. Recommending root-cause and mitigation steps for cloud incidents using large language models. In *IEEE/ACM ICSE*, 2023.

[4] Sa'di Altamimi, Basel Altamimi, David Côté, et al. Toward a superintelligent action recommender for network operation centers using reinforcement learning. *IEEE Access*, 2023.

[5] Mazene Ameur, Abdelkader Mekrache, Bouziane Brik, et al. Llm-powered agentic ai for 5g/6g networks: A tutorial and survey on architectures, protocols, and standardization. arXiv preprint arXiv:2607.16066, 2026.

[6] Behnaz Arzani, Sina Taheri, Pooria Namyar, et al. Raha: A general tool to analyze wan degradation. In *ACM SIGCOMM*, 2025.

[7] Rufat Asadli, Benjamin Hoffman, Ioannis Protogeros, et al. Evaluating agentic configuration repair for computer networks. arXiv preprint arXiv:2606.06212, 2026.

Table 7: Index of the corpus: every included work, by the loop stage its evaluation reaches, with its evaluation substrate in parentheses. No work counted in the audit is invisible in the paper. Substrate keys: A analytical or synthetic, P packet simulation, E emulation, T testbed, R replayed traces, **D** production, ? not indicated.

Loop closure	Works (substrate)
101 works. Stage abbreviations as in Table 4.	
Det (13)	Masood [64](A), Meza [67](**D**), Yu [110](A), Asghar [9](A), Chen [14](T), Testart [89](R), Chakraborttii [13](**D**), Kheradmand [44](?), Moriano [70](R), Ahmad [2](**D**), Deng [23](**D**), Krebs [47](E), De la Cruz Cabello [48](T)
Diag (36)	Shariati [84](A), Wang [98](A), Liang [52](A), Lun [58](A), Zhang [115](A), Jiao [42](A), Wang [96](RA), Xin [104](?), Roy [80](**D**), Somashekar [87](?), Wang [99](R), Zhang [118](**D**), Arzani [6](**D**), Das and Kant [22](E), Fu [30](R), Liu [53](**D**), Lu [57](T), Luo [59](R**D**), Pei [77](**D**), Salman [81](R), Wang [93](**D**), Wang [97](PT), Wang [102](E), Yang [105](**D**), Yao [109](**D**), Al Shami [83](A), Cai [12](R), Cui [20](**D**), Fang [26](R), Gong [33](R), Gopal [34](R), Kim [46](R), Mukkolakkal [71](?), Su [88](**D**), Wang [95](R), Yang [107](**D**)
Adv (19)	Gan [31](T), Mohammed [69](R), Ahmed [3](R), Altamimi [4](A), Ge [32](**D**), Hamadanian [36](?), Namyar [72](**D**), Jiang [41](**D**), Malul [62](R), Van Tu [90](A), Wang [94](A), Yu [112](**D**), Yungaicela-Naula [113](?), D'Antonio [21](**D**), Wang [100](**D**), Woo [103](A), Zhang [117](A), Gu [35](A), Zhang [116](R)
Gated (5)	Dzeparoska [25](T), Omar [75](A), Las-Casas [50](?), Mekrache [66](T), Jang [39](R)
Auto (22)	Ochoa-Aday [74](A), Nikolic [73](A), Foley [29](A), Vaezpour [91](A), Hammad [37](A), Mitropoulou [68](A), Yungaicela-Naula [114](T), Liu [54](T), Sarda [82](A), Chen [16](T), Jha [40](T), Kiely [45](A), Lacava [49](T), Lu [56](A), Zhou [119](E), Dou [24](TR), Johnphill [43](T), Makiyah [60](E), Mao [63](R), Mehta [65](E), Sharma [85](A), Sri Latha [51](PER)
Ver (4)	Avgerinos [10](T), Yang [108](R), Asadli [7](A), Protogeros [78](A)
n/i (2)	Wang [101](E), Malik [61](?)

[8] Ahmad Asghar, Hasan Farooq, and Muhammad Ali Imran. Self-healing in emerging cellular networks: Review, challenges, and research directions. *IEEE Communications Surveys & Tutorials*, 2018.

[9] Muhammad Zeeshan Asghar, Mudassar Abbas, Khaula Zeeshan, et al. Assessment of deep learning methodology for self-organizing 5g networks. *Applied Sciences*, 2019.

[10] Vasilis Avgerinos, Kostas Ramantas, Luis Alonso, et al. Arm: Autonomous remediation and management with llm agents for intent-driven control. *IEEE Internet of Things Journal*, 2025.

[11] Chafika Benzaïd and Tarik Taleb. Ai-driven zero touch network and service management in 5g and beyond: Challenges and research directions. *IEEE Network*, 2020.

[12] Yuanhong Cai, Xiaohui Nie, Kanglin Yin, et al. A multi-dataset benchmark for evaluating llm agents in microservice failure diagnosis. arXiv preprint arXiv:2606.29193, 2026.

[13] Chandranil Chakraborttii and Heiner Litz. Improving the accuracy, adaptability, and interpretability of ssd failure prediction models. In *ACM SoCC*, 2020.

[14] Xiaoliang Chen, Baojia Li, Roberto Proietti, et al. Self-taught anomaly detection with hybrid unsupervised/supervised machine learning in optical networks. *Journal of Lightwave Technology*, 2019.

[15] Xiaoliang Chen, Che-Yu Liu, Roberto Proietti, et al. Automating optical network fault management with machine learning. *IEEE Communications Magazine*, 2022.

[16] Yinfang Chen, Manish Shetty, Gagan Somashekar, et al. Aiopslab: A holistic framework to evaluate ai agents for enabling autonomous clouds. arXiv preprint arXiv:2501.06706, 2025.

[17] Marco Chiesa, Andrzej Kamisiński, Jacek Rak, et al. A survey of fast-recovery mechanisms in packet-switched networks. *IEEE Communications Surveys & Tutorials*, 2021.

[18] Estefanía Coronado, Rasoul Behravesh, Tejas Subramanya, et al. Zero touch management: A survey of network automation solutions for 5g and 6g networks. *IEEE Communications Surveys & Tutorials*, 2022.

[19] Sergio Cruzes. Optical network automation: Enhancing networks with machine learning, digital twins, and advanced technologies. TechRxiv preprint, 2025.

[20] Shengkun Cui, Rahul Krishna, Saurabh Jha, et al. Praxis: Integrating program analysis with observability for root-cause analysis. In *IEEE/IFIP DSN*, 2026.

[21] Rocker D'Antonio and Harry Xie. Human-in-the-loop runbook improvement with agentic support automation. In *IEEE CogMI*, 2025.

[22] Sourav Das and Krishna Kant. Digital twin and llm assisted online diagnosis in microservices based enterprise systems. In *IEEE/IFIP CNSM*, 2025.

[23] Yangtao Deng, Xiang Shi, Zhuo Jiang, et al. Minder: Faulty machine detection for large-scale distributed model training. arXiv preprint arXiv:2411.01791, 2024.

[24] Rengan Dou, Bin Chen, Yu Wang, et al. Towards autonomous o&m: An agent framework for cloud infrastructure incident management. In *IEEE ICETCI*, 2026.

[25] Kristina Dzeparoska, Nasim Beigi-Mohammadi, Ali Tizghadam, et al. Towards a self-driving management system for the automated realization of intents. *IEEE Access*, 2021.

[26] Aoyang Fang, Yifan Yang, Jin'Ao Shang, et al. Openrca 2.0: From outcome labels to causal process supervision. arXiv preprint arXiv:2606.27154, 2026.

[27] Honglin Fang, Peng Yu, Can Tan, et al. Self-healing in knowledge-driven autonomous networks: Context, challenges, and future directions. *IEEE Network*, 2024.

[28] Jaleh Farmani and Amirreza Khalil Zadeh. Ai-based self-healing solutions applied to cellular networks: An overview. arXiv preprint arXiv:2311.02390, 2023.

[29] Myles Foley, Chris Hicks, Kate Highnam, et al. Autonomous network defence using reinforcement learning. In *ACM ASIA CCS*, 2022.

[30] Feng Fu, Ding Hong, Yong Qin, et al. Leveraging multi-agent framework for root cause analysis. *Complex & Intelligent Systems*, 2025.

[31] Yu Gan, Yanqi Zhang, Kelvin Hu, et al. Seer: Leveraging big data to navigate the complexity of performance debugging in cloud microservices. In *ACM ASPLOS*, 2019.

[32] Changhan Ge, Zihui Ge, Xuan Liu, et al. Chroma: Learning and using network contexts to reinforce performance improving configurations. In *ACM MobiCom*, 2023.

[33] Albert Gong, KS Choi, Abhineet Agarwal, et al. Orca-bench: How ready are language model agents for oncall? arXiv preprint arXiv:2607.28545, 2026.

[34] Athira Gopal and Ashwanth Krishnan. How far can root cause analysis go on real-world telemetry data? arXiv preprint arXiv:2607.13548, 2026.

[35] Zhenrong Gu, Peng Zhang, Xing Feng, et al. Astragalus: Automatic configuration repair for production networks. arXiv preprint arXiv:2605.22092, 2026.

[36] Pouya Hamadanian, Behnaz Arzani, Sadjad Fouladi, et al. A holistic view of ai-driven network incident management. In *ACM HotNets*, 2023.

[37] Mohamed Hammad, Nabil M. Hewahi, and Wael Elmedany. Enhancing network intrusion recovery in sdn with machine learning: an innovative approach. *Arab Journal of Basic and Applied Sciences*, 2023.

[38] Jibum Hong, Nguyen Van Tu, and James Won-Ki Hong. A comprehensive survey on llm-based network management and operations. *International Journal of Network Management*, 2025.

[39] Sooyoung Jang, Taehyun Park, Hyunbean Yi, et al. Causalopsbench: A reproducible interventional-replay benchmark for topology-aware operational incident response. *IEEE Access*, 2026.

[40] Saurabh Jha, Rohan Arora, Yuji Watanabe, et al. Itbench: Evaluating ai agents across diverse real-world it automation tasks. arXiv preprint arXiv:2502.05352, 2025.

[41] Yuxuan Jiang, Chaoyun Zhang, Shilin He, et al. Xpert: Empowering incident management with query recommendations via large language models. In *IEEE/ACM ICSE*, 2024.

[42] Yan Jiao, Pin-Han Ho, Xiangzhu Lu, et al. On real-time failure localization via instance correlation in optical transport networks. In *IFIP Networking*, 2023.

[43] Obinna Johnphill, Ali Safaa Sadiq, Omprakash Kaiwartya, et al. Cross: a cloud-native approach to automated remediation and self-healing in cyber-physical systems. *Cybersecurity*, 2026.

[44] Ali Kheradmand. Automatic inference of high-level network intents by mining forwarding patterns. In *ACM SOSR*, 2020.

[45] Mitchell Kiely, Metin Ahiskali, Etienne Borde, et al. Exploring the efficacy of multi-agent reinforcement learning for autonomous cyber defence: A cage challenge 4 perspective. In *AAAI*, 2025.

[46] Taeyoon Kim, Woohyeok Park, Hoyeong Yun, et al. Why do ai agents systematically fail at cloud root cause analysis? arXiv preprint arXiv:2602.09937, 2026.

[47] Jakob Krebs, Dimitry Gavrilenko, Daniel Amir, et al. Flowpulse: Catching network failures in ml clusters. In *ACM HotNets*, 2025.

[48] Miguel De la Cruz Cabello, Tiago Prince Sales, and Marcos Machado. Log anomaly detection in aiops: A real-world implementation using large language models. *Systems and Soft Computing*, 2026.

[49] Andrea Lacava, Leonardo Bonati, Niloofar Mohamadi, et al. dapps: Enabling real-time ai-based open ran control. *Computer Networks*, 2025.

[50] Pedro Las-Casas, Alok Kumbhare, Rodrigo Fonseca, et al. Llexus: an ai agent system for incident management. *ACM SIGOPS Operating Systems Review*, 2024.

[51] P. Sri Latha, Suneel Pappala, and S. Rama Sree. Machine learning driven analytics and ai tools for intelligent network diagnosis and automated issue resolution. In *IEEE ICCCES*, 2026.

[52] Rongyu Liang, Feng Liu, and Jie Liu. A belief network reasoning framework for fault localization in communication networks. *Sensors*, 2020.

[53] Wei Liu, Kun Qian, Zhenhua Li, et al. Skeletonhunter: Diagnosing and localizing network failures in containerized large model training. In *ACM SIGCOMM NetAI*, 2025.

[54] Xiaomin Liu, Qizhi Qiu, Yihao Zhang, et al. First field trial of llm-powered ai agent for lifecycle management of autonomous driving optical networks. arXiv preprint arXiv:2409.14605, 2024.

[55] Madhusanka Liyanage, Quoc-Viet Pham, Kapal Dev, et al. A survey on zero touch network and service management (zsm) for 5g and beyond networks. *Journal of Network and Computer Applications*, 2022.

[56] Jingwen Lu, Chaowei Tang, Wenyu Ma, et al. Graph-based reinforcement learning for software-defined networking traffic engineering. *J. King Saud Univ. Comput. Inf. Sci.*, 2025.

[57] Ruiming Lu, Yunchi Lu, Yuxuan Jiang, et al. One-size-fits-none: Understanding and enhancing slow-fault tolerance in modern distributed systems. In *USENIX NSDI*, 2025.

[58] Huazhi Lun, Mengfan Fu, Xiaomin Liu, et al. Soft failure identification for long-haul optical communication systems based on one-dimensional convolutional neural network. *Journal of Lightwave Technology*, 2020.

[59] Yu Luo, Jiamin Jiang, Jingfei Feng, et al. Opsagent: An evolving multi-agent system for incident management in microservices. arXiv preprint arXiv:2510.24145, 2025.

[60] E. H. Makiyah, M. N. Rasool, and F. A. Rawdhan. A reinforcement learning framework for self-healing fault recovery in intent-based sdns. *Discover Networks*, 2026.

[61] Arun Malik. Autonomous incident resolution at hyperscale: An agentic ai architecture for network operations. arXiv preprint arXiv:2606.09122, 2026.

[62] Ehud Malul, Yair Meidan, Dudu Mimran, et al. Genkubesec: Llm-based kubernetes misconfiguration detection, localization, reasoning, and remediation. arXiv preprint arXiv:2405.19954, 2024.

[63] Jiayi Mao, Liqun Li, Yanjie Gao, et al. Stepfly: Agentic troubleshooting guide automation for incident diagnosis. *Proceedings of the ACM on Software Engineering*, 2026.

[64] Usama Masood, Ahmad Asghar, Muhammad Ali Imran, et al. Deep learning based detection of sleeping cells in next generation cellular networks. In *IEEE GLOBECOM*, 2018.

[65] Het Mehta. A reinforcement-guided graph attention framework for autonomous self-healing in software-defined networks. In *IEEE MI-STA*, 2026.

[66] Abdelkader Mekrache, Adlen Ksentini, and Christos Verikoukis. Intent-based management of next-generation networks: an llm-centric approach. *IEEE Network*, 2024.

[67] Justin Meza, Tianyin Xu, Kaushik Veeraraghavan, et al. A large scale study of data center network reliability. In *ACM IMC*, 2018.

[68] Katerina Mitropoulou, Panagiotis Kokkinos, Polyzois Soumplis, et al. Anomaly detection in cloud computing using knowledge graph embedding and machine learning mechanisms. *Journal of Grid Computing*, 2023.

[69] Shady Mohammed, Ayse Rumeysa Mohammed, David Côté, et al. A machine-learning-based action recommender for network operation centers. *IEEE Transactions on Network and Service Management*, 2021.

[70] Pablo Moriano, Raquel Hill, and L. Jean Camp. Using bursty announcements for detecting bgp routing anomalies. *Computer Networks*, 2021.

[71] Muhamed Ramees Cheriya Mukkolakkal. Deploy, calibrate, monitor, heal – no human required: An autonomous ai sre agent for elasticsearch. arXiv preprint arXiv:2604.03933, 2026.

[72] Pooria Namyar, Arvin Ghavidel, Daniel Crankshaw, et al. Enhancing network failure mitigation with performance-aware ranking. arXiv preprint arXiv:2305.13792, 2023.

[73] Jovan Nikolić, Nursultan Jubatyrov, and Evangelos Pournaras. Self-healing dilemmas in distributed systems: Fault correction vs. fault tolerance. *IEEE Transactions on Network and Service Management*, 2021.

[74] Leonardo Ochoa-Aday, Cristina Cervelló-Pastor, and Adriana Fernández-Fernández. Self-healing and sdn: bridging the gap. *Digital Communications and Networks*, 2019.

[75] Tamer Omar, Thomas Ketseoglou, and Ibrahim Naffaa. A novel self-healing model using precoding and big-data based approach for 5g networks. *Pervasive and Mobile Computing*, 2021.

[76] Lei Pang, Chungang Yang, Danyang Chen, et al. A survey on intent-driven networks. *IEEE Access*, 2020.

[77] Changhua Pei, Zexin Wang, Fengrui Liu, et al. Flow-of-action: Sop enhanced llm-based multi-agent system for root cause analysis. In *ACM WWW, companion*, 2025.

[78] Ioannis Protogeros, Rufat Asadli, Benjamin Hoffman, et al. Benchmarking llm-driven network configuration repair. arXiv preprint arXiv:2604.22513, 2026.

[79] Mirna El Rajab, Li Yang, and Abdallah Shami. Zero-touch networks: Towards next-generation network automation. *Computer Networks*, 2024.

[80] Devjeet Roy, Xuchao Zhang, Rashi Bhave, et al. Exploring llm-based agents for root cause analysis. In *ACM FSE, companion*, 2024.

[81] Aymen Dawood Salman, Akram T. Zeyad, Shereen S. Jumaa, et al. Hybrid llm-assisted fault diagnosis framework for 5g/6g networks using real-world logs. *Computers*, 2025.

[82] Komal Sarda, Zakeya Namrud, Marin Litoiu, et al. Leveraging large language models for the auto-remediation of microservice applications: An experimental study. In *ACM FSE, companion*, 2024.

[83] Maryam Al Shami, Jun Yan, and Emmanuel Thepie Fapi. O-ran Xapps conflict prediction using graph convolutional networks. *IEEE Transactions on Mobile Computing*, 2026.

[84] Behnam Shariati, Marc Ruiz, Jaume Comellas, et al. Learning from the optical spectrum: Failure detection and identification. *Journal of Lightwave Technology*, 2018.

[85] Brijesh Kumar Sharma, Jofia Jose Prakash, Abhishek Dwivedi, et al. A self-healing 6g network model using ai-driven predictive analytics. In *IEEE I5CPS*, 2026.

[86] Yogesh Sharma, Deval Bhamare, Nishanth Sastry, et al. Sla management in intent-driven service management systems: A taxonomy and future directions. *ACM Computing Surveys*, 2023.

[87] Gagan Somashekar, Anurag Dutt, Mainak Adak, et al. Gamma: Graph neural network-based multi-bottleneck localization for microservices applications. In *ACM WWW*, 2024.

[88] Jingyi Su, Yihao Zhang, Dianxuan Fu, et al. First demonstration of multi-agent llm system for million-scale optical link management in global production aidcs. arXiv preprint arXiv:2608.23145, 2026.

[89] Cecilia Testart, Philipp Richter, Alistair King, et al. Profiling bgp serial hijackers: Capturing persistent misbehavior in the global routing table. In *ACM IMC*, 2019.

[90] Nguyen Van Tu, Sukhyun Nam, and James Won-Ki Hong. Intent-based network configuration using large language models. *International Journal of Network Management*, 2024.

[91] Elaheh Vaezpour, Layla Majzoobi, Mohammad Akbari, et al. A deep learning-based approach for cell outage compensation in noma networks. *IEEE Open Journal of Vehicular Technology*, 2022.

[92] Luis Velasco, Sima Barzegar, Fatemehsadat Tabatabaeimehr, et al. Intent-based networking and its application to optical networks [invited tutorial]. *Journal of Optical Communications and Networking*, 2021.

[93] Chenxu Wang, Xumiao Zhang, Runwei Lu, et al. Towards llm-based failure localization in production-scale networks. In *ACM SIGCOMM NetAI*, 2025.

[94] Danshi Wang, Yidi Wang, Xiaotian Jiang, et al. When large language models meet optical networks: Paving the way for automation. *Electronics*, 2024.

[95] Junle Wang, Xingchuang Liao, and Wenjun Wu. Star: A stage-attributed triage and repair framework for rca agents in microservices. arXiv preprint arXiv:2605.15581, 2026.

[96] Lu Wang, Chaoyun Zhang, Ruomeng Ding, et al. Root cause analysis for microservice systems via hierarchical reinforcement learning from human feedback. In *ACM KDD*, 2023.

[97] Shicheng Wang, Menghao Zhang, Xiao Li, et al. Hawkeye: Diagnosing rdma network performance anomalies with pfc provenance. In *ACM SIGCOMM*, 2025.

[98] Yueping Wang, Kun Zhu, Mengyun Sun, et al. An ensemble learning approach for fault diagnosis in self-organizing heterogeneous networks. *IEEE Access*, 2019.

[99] Zefan Wang, Zichuan Liu, Yingying Zhang, et al. Rcagent: Cloud root cause analysis by autonomous agents with tool-augmented large language models. In *ACM CIKM*, 2024.

[100] Zhaodong Wang, Samuel Lin, Guanqing Yan, et al. Intent-driven network management with multi-agent llms: The confucius framework. In *ACM SIGCOMM NetAI*, 2025.

[101] Zhihao Wang, Alessandro Cornacchia, Franco Galante, et al. Towards a playground to democratize experimentation and benchmarking of ai agents for network troubleshooting. In *ACM NGNO workshop*, 2025.

[102] Zhihao Wang, Alessandro Cornacchia, Alessio Sacco, et al. A network arena for benchmarking ai agents on network troubleshooting. arXiv preprint arXiv:2512.16381, 2025.

[103] Yeunjin Woo and Honguk Woo. Nfvagent: A retrieval-augmented llm agent for resilient nfv failure recovery. In *IEEE/IFIP CNSM*, 2025.

[104] Ruyue Xin, Peng Chen, and Zhiming Zhao. Causalrca: Causal inference based precise fine-grained root cause localization for microservice applications. *Journal of Systems and Software*, 2023.

[105] Bo Yang, Huanwu Hu, Yifan Li, et al. Skynet: Analyzing alert flooding from severe network failures in large cloud infrastructures. In *ACM SIGCOMM NetAI*, 2025.

[106] Li Yang, Mirna El Rajab, Abdallah Shami, et al. Enabling automl for zero-touch network security: Use-case driven analysis. *IEEE Transactions on Network and Service Management*, 2024.

[107] Yitao Yang, Yu Deng, Yifan Xiong, et al. Tsguard: Automated user-centric incident diagnosis for ai workloads in the cloud. *Proceedings of the ACM on Software Engineering*, 2026.

[108] Zhenning Yang, Hui Guan, Victor Nicolet, et al. Automated cloud infrastructure-as-code reconciliation with ai agents. arXiv preprint arXiv:2510.20211, 2025.

[109] Zhiyi Yao, Pengbo Hu, Congcong Miao, et al. Holmes: Localizing irregularities in llm training with mega-scale gpu clusters. In *USENIX NSDI*, 2025.

[110] Peng Yu, Fanqin Zhou, Tao Zhang, et al. Self-organized cell outage detection architecture and approach for 5g h-cran. *Wireless Communications and Mobile Computing*, 2018.

[111] Yinbo Yu, Xing Li, Xue Leng, et al. Fault management in software-defined networking: A survey. *IEEE Communications Surveys & Tutorials*, 2018.

[112] Zhaoyang Yu, Minghua Ma, Chaoyun Zhang, et al. Monitorassistant: Simplifying cloud service monitoring via large language models. In *ACM FSE, companion*, 2024.

[113] Noe M. Yungaicela-Naula, Vishal Sharma, and Sandra Scott-Hayward. Misconfiguration in o-ran: Analysis of the impact of ai/ml. *Computer Networks*, 2024.

[114] Noe M. Yungaicela-Naula, Cesar Vargas-Rosales, Jesús Arturo Pérez-Díaz, et al. Physical assessment of an sdn-based security framework for ddos attack mitigation: Introducing the sdn-slowrate-ddos dataset. *IEEE Access*, 2023.

[115] Tao Zhang, Kun Zhu, and Ekram Hossain. Data-driven machine learning techniques for self-healing in cellular wireless networks: Challenges and solutions. *Intelligent Computing*, 2022.

[116] Wenya Zhang, Zhi Yang, Fang Peng, et al. Galr: Graph-based root cause localization and llm-assisted recovery for microservice systems. *Electronics*, 2026.

[117] Xiaofeng Zhang, Xianming Gao, Peilin Tao, et al. Graphsynth: Synthesis of network configuration templates using large language models. In *ACM GAIIS*, 2025.

[118] Xuchao Zhang, Supriyo Ghosh, Chetan Bansal, et al. Automated root causing of cloud incidents using in-context learning with gpt-4. In *ACM FSE, companion*, 2024.

[119] Yajie Zhou, Jiajun Ruan, Eric S. Wang, et al. Netarena: Dynamic benchmarks for ai agents in network automation. arXiv preprint arXiv:2506.03231, 2025.