JOURNAL OF INTERNATIONAL CRISIS AND RISK COMMUNICATION RESEARCH
ISSN: 2576-0017
2026, VOL 9, NO 1

Agentic AI For Network Management: Autonomous Troubleshooting And Configuration Through MCP Servers

Vivek Koodakkara Shanmughan

Independent Researcher, USA.

Abstract

Enterprise networks have grown much more complex, with distributed network topologies, hybrid cloud infrastructure, and on-the-fly configuration. Customary approaches to operationalizing network infrastructure have been reactive and inefficient. Collocating agentic artificial intelligence with the Model Context Protocol allows the development of a scalable system for self-managing networks, where clever agents observe, analyze, plan, execute, and verify via common interfaces. As a replacement for manual operational troubleshooting and configuration processes, the architecture employs automated reasoning cycles for real-time anomaly detection, root cause analysis, and remediation. The Model Context Protocol exposes network functions as clean, typed, and validation-friendly interfaces that allow the use of artificial intelligence agents in the architecture to interoperate with the underlying network infrastructure, enabling a reliable semantic understanding of routing protocols, security policies, interface states, and performance indicators. Pre-execution validation, policy-based verification, rollback, and audit logging are supported as part of the architecture for secure, accountable automated operations. Intent-driven configuration management is especially noteworthy: Natural language requirements are converted into validated network configurations. This reduces time-to-deployment of complex operations across multiple devices and, at the same time, minimizes human error. Stateful processing abstractions, network-wide synthesis techniques, and declarative models enable the diagnosis of problems by separating symptoms from root causes while preserving network-wide invariants across a distributed set of network elements. This enables network healing, compliance enforcement, and operational agility at a scale not achievable by human analysts and administrators. Organizations using agentic network management can achieve orders of magnitude improvement in service availability and operational efficiency, and infrastructure agility, while enabling network and service architects to focus on design rather than implementation. Self-healing networks represent the next generation of enterprise networking infrastructure, where smart automation deals with operational complexity with precision and speed that fundamentally changes the economics and capabilities of network operations.

Keywords: Agentic Artificial Intelligence, Network Management Automation, Model Context Protocol, Intent-Driven Configuration, Autonomous Troubleshooting.

1. Introduction

Enterprise networks have become distributed and are hybridly connected to various cloud networks and thousands of configuration points to consider. The report on Cisco Global Networking Trends explains that

organizations are experiencing new levels of complexity as they introduce artificial intelligence in their networks. Complexity of the network has also been cited by enterprise customers as being one of the most significant issues in the operation of scaled-up digital infrastructure [1]. Traditional Network Management Systems are mainly reactive and need human intervention to troubleshoot and/or reconfigure infrastructure components. It can cause more lengthy outages, configuration drift, and high costs because network outages may negatively affect the productivity of the enterprise. Network downtimes can be expensive, and estimates of downtime vary by industry sector, although outages usually become the most expensive operation costs to a business when both direct costs (lost productivity and lost income) and indirect costs (tarnished reputation and customer dissatisfaction) are combined [2]. The emergence of agentic AI is transforming the character of network operations radically. New systems are agentic AI systems, which independently make their decisions and are able to perform actions using structured tools. Via the addition of the Model Context Protocol, agentic AI systems can act as an independent entity that is capable of operating networks with unmatched accuracy, consistency, and awareness in real time. This transformation will enable organizations to drop the reactive firefighting approach to the intelligent coordination of network resources.

2. Architectural Foundation: MCP Servers as Network Interfaces

An architectural basis for autonomous network management is the Model Context Protocol server, which provides an open abstraction layer between the AI agents and the network. MCP servers expose clean, typed, and validated interfaces to network operations, causing the client never to parse unstructured command line interface (CLI) output, nor work through vendor APIs. Network update is a well-studied problem in software-defined networking. As the size and complexity of networks increase, the question of how to write network updates in a strong way becomes more relevant. In particular, inside updates, programmers need abstractions that give certain guarantees about update semantics. For example, if a network operator updates their network, they want to be sure that, at no point during the update, packets are forwarded through a mix of the old and new configurations (packet consistency) and that all packets from a single flow are either entirely under the old or new configuration (per-flow consistency). The protocol has provided a consistent semantic for interfaces, virtual LANs, routing protocols, access control lists, VPN tunnels, and health metrics. The work on the consistent network updates has established that several coordination mechanisms are required to achieve consistency in distributed networking environments, especially when multiple switches are updated and forwarding correctness is maintained when a network is being updated [4]. The MCP architecture implements these principles via conventional version control, schema validation, and transaction semantics to ensure every interaction of the AI agent with the network infrastructure is safe. This allows AI agents to learn generalizable network management skills that can be shared across vendor platforms and deployment scenarios in order to remove the vendor-specific complications that have historically obstructed network automation.

Table 1: MCP Server Network Interface Capabilities [3, 4]

Interface Function	Description	Key Benefits
Device Status Retrieval	Real-time collection of operational state information from network elements	Provides comprehensive visibility into hardware health, software versions, and operational parameters
Routing Table Export	Extraction of forwarding information base and routing protocol state	Enables analysis of path selection, convergence behavior, and routing policy effectiveness
Configuration Validation	Syntax checking, policy compliance verification, and conflict detection	Identifies errors before deployment, preventing configuration-induced outages
Rollback Triggers	Automated restoration of previous configuration states	Enables rapid recovery from problematic changes with minimal service disruption

Change Application	Coordinated deployment of configuration updates across multiple devices	Ensures atomic updates that maintain network consistency during transitions
Health Metrics Collection	Gathering of performance indicators, including latency, throughput, and error rates	Supports baseline establishment, anomaly detection, and capacity planning

3. Agentic Reasoning Cycle: From Observation to Execution

The agentic AI systems pursue a process of five development phases. They will autonomously and networked work with less or no human intervention. In the observation stage, an AI agent monitors the health of a network, its present settings and performance using queries at MCP server to establish the normal state of the network and the circumstance that shows the presence of an abnormal scenario. Pattern recognition methods are employed during analysis to detect the root cause of the anomalies that have been identified. Stateful abstractions also support more advanced features in a network that would not exist with stateless implementations [5]. These stateful functionalities enable the agent to trace connection states, sequence numbers and counters across a number of devices that provide the full visibility of how the network is being operated. During the planning phase, candidate remediations are created based on the thinking about policy constraints and the operational state of the agent. The agent involves network-wide synthesis methods which generate valid configurations automatically by a high-level policy specification [6]. Network-wide synthesis will map logical network properties into the actual per-device configurations in such a way that the synthesized network behavior satisfies the policy of all traffic patterns and failure scenarios. The execution phase implies that the plan that is selected is implemented within the functions of the corresponding MCP server. Pre-execution checks, dry-run simulators, and automatic checkpointing of executing code are also part of this stage to be able to switch back in case of undesirable side effects. The last procedure, a verification is to help know whether the desired outcome has been achieved. This is achieved by testing the convergence of behavior of the control plane and the data plane forwarding. This is an open ended reasoning loop. This loop has a temporal and spatial scale of single parameter setting all the way to multi-device orchestration, and becomes increasingly complicated as an increasing number of past incidences are taken into account.

Table 2: Agentic AI Reasoning Cycle Phases [5, 6]

Phase	**Primary Functions**	**Outputs**
Observe	Continuous monitoring of telemetry data, collection of configuration states, and establishment of behavioral baselines	Normalized metrics, state snapshots, deviation alerts
Analyze	Pattern recognition across multiple data sources, correlation of events, and root cause identification	Diagnostic hypotheses, causal relationships, and impacted components
Plan	Generation of remediation alternatives, evaluation against constraints, and selection of optimal strategy	Ranked action proposals, impact assessments, and risk evaluations
Execute	Configuration deployment, coordinated updates, checkpoint creation, transaction management	Applied changes, rollback points, execution logs

Verify	Post-change validation, performance comparison, secondary issue detection	Confirmation reports, residual problem identification, and knowledge base updates

4. Autonomous Troubleshooting: Reducing MTTD and MTTR

Using agentic AI for network troubleshooting turns the reactive incident-management process customarily conducted by human staff into immediate action planning. This considerably improves service availability. For instance, in case of packet loss or delay spikes, the AI agent can proactively retrieve health information from the MCP server and compare the current health metrics with historical baselines to detect positive/negative deviations and take appropriate actions. AI-powered monitoring can detect anomalies, predict failures, and respond to intrusions far more quickly than customary monitoring based on fixed thresholds with manual operator analysis. AI can also be more effective in detecting patterns that correspond to a range of different problems. For example, it can detect unusual patterns at different layers of the network or when the network gradually degrades in ways that are not obvious to human operators. This system relies on correlation to identify the difference between a symptom and its cause, in order to avoid the situation seen by operators frequently, where symptoms such as error messages are dealt with as the underlying failures or misconfigurations that may continue to generate them. In the case of performance, the agent would look for the cause in the network topology based on routing table stability, protocol state machines, interface error rates, and resource usage. The distributed nature of the modern network poses difficulties in fault localization as faults may be with components in different administrative domains, or with components from different vendors. When an agent detects a misconfiguration or a failing component, it recommends a corrective action and provides transparent reasoning of the diagnostic process, the cause of the problem, the action taken, and the expected result. This transparency allows human oversight and approval of interventions in high-consequence situations, or complete automation of remediation, if the agent can show consistent reliability in past interactions and can display appropriate understanding of the situation.

Table 3: Autonomous Troubleshooting Capabilities [7, 8]

Capability Domain	**Automated Functions**	**Operational Impact**
Anomaly Detection	Real-time comparison of observed metrics against statistical baselines and learned patterns	Identifies performance degradation, protocol failures, and resource exhaustion before user impact
Root Cause Analysis	Correlation of symptoms across network layers, elimination of secondary effects, isolation of failure points	Reduces diagnostic time and prevents ineffective remediation of downstream symptoms
Remediation Synthesis	Generation of corrective actions based on failure type, validation against policies, and impact prediction	Produces safe, compliant solutions that address underlying problems rather than masking symptoms
Autonomous Execution	Coordinated application of fixes across affected devices, creation of rollback checkpoints, and verification of outcomes	Enables rapid resolution without human intervention for well-understood failure scenarios
Knowledge Retention	Recording of incident patterns, remediation effectiveness, and outcome correlations	Improves future diagnostic accuracy and response speed through accumulated operational experience
Escalation Management	Recognition of novel or high-risk scenarios, preparation of context for human review, and recommendation formulation	Ensures appropriate human involvement for situations outside autonomous operation boundaries

5. Intent-Driven Configuration Management: From Natural Language to Network Reality

Besides network troubleshooting after the fact, agentic AI also enables proactive, intent-based configuration management able to close the divide between business intent and technical configuration via natural language understanding and automated synthesis. Network operators specify their operational intent as natural language specifications of the intended behavior of the network, while an AI agent automatically synthesizes the corresponding configuration artifacts that realize the intended policies on the network. This has been demonstrated in the ONOS platform, where distributed control planes implement large-scale networks with the help of abstraction layers. The use of abstract network programming allows the operator to think of network-wide policies rather than the lower-level configuration of the devices within the system [8]. The separation of concerns allows the AI agent to reason about network behavior at a higher level and automatically compile abstract policies into vendor-specific configuration language across heterogeneous device populations. The agent uses network design patterns, vendor best practices, and organizational preferences to synthesize complete sets of configurations from high-level intent. When using this option, routing protocol, security zone, quality of service, and monitoring configurations are generated by the agent and verified by the MCP server. Checking network configurations serves to verify that the configurations are correct with respect to syntax, policy, and conflict-checking rules. Network configuration correctness checkers that can process abstract network topologies can prove the network will always behave correctly in all possible scenarios [9]. This formal verification detects configuration errors that are extremely difficult to detect via testing, such as configurations that push the system into the exotic failure mode or configurations with complex interactions among distributed components. The agent takes snapshots of the state of the entire network before every configuration change, which are called automatic rollback checkpoints, so that erroneous configurations can be quickly rolled back in production. Configuration simulation is how the agent predicts the effects of proposed changes. It does this by digitally modeling the network in an identical twin environment, where a copy of the production topology, traffic patterns, and failures is simulated. This eliminates manually configuring the network, which is error-prone and requires vendor-specific, expert knowledge to understand how to use their command-line interface. It provides a safe, declarative, and intent-based model, meaning that domain experts do not have to know how to configure how to achieve an intent on a specific vendor platform or specific software version.

Table 4: Intent-Driven Configuration Capabilities [9, 10]

Configuration Aspect	Automated Processes	Value Delivered
Natural Language Processing	Parsing of operational requirements, extraction of semantic intent, and ambiguity resolution	Eliminates the need for command-line expertise, making configuration accessible to domain experts
Policy Synthesis	Translation of high-level objectives into device-specific configurations across vendor platforms	Ensures consistent implementation of business requirements across heterogeneous infrastructure
Formal Verification	Mathematical proof of correctness, invariant preservation checking, and reachability analysis	Provides guarantees that configurations will behave as intended under all traffic and failure scenarios
Simulation Testing	Digital twin modeling, traffic pattern replay, failure injection, performance prediction	Identifies problematic configurations before production deployment through comprehensive scenario testing
Coordinated Deployment	Atomic multi-device updates, dependency ordering, rollback coordination, and consistency maintenance	Eliminates configuration windows where network behavior is undefined or inconsistent

Compliance Enforcement	Continuous validation against regulatory requirements, corporate policies, and security standards	Prevents configuration drift and ensures ongoing adherence to governance frameworks

Conclusion

With agentic artificial intelligence and Model Context Protocol (MCP) servers, enterprise networks will shift from human-driven reactive operating models to automated self-healing and self-optimizing ones. As networks grow in complexity, agents at scale will follow. Agent intelligence will provide reliable, consistent, and generic interfaces for safe consumption between agentic operations and the underlying network infrastructure through full validation, policy compliance, and audit capabilities. This reasoning cycle enables diagnosis and the synthesis of corrective action orders of magnitude faster and more accurately than current human-based processes. It resolves the error-prone translation from business requirements to technical implementation with natural language to vendor-neutral and validated configuration specification, and through autonomic adaptation to populations of heterogeneous devices. Formal verification is also used to guarantee that autonomous reconfiguration of a network's configuration preserves properties such as reachability, isolation, and performance guarantees, even when the underlying network infrastructure is being automatically reconfigured to meet different sets of operational requirements. Companies adopting smart automation data center architectures have experienced dramatic increases in service availability, operational cost, and infrastructure agility, while moving technical staff from tactical troubleshooting to calculated architecture, capacity planning, and policy work. The result of these approaches leads to continuous learning, when systems develop an increasingly advanced model of network behavior and optimized responses based on experience gained across different types of failures and traffic patterns. Reinforcing policy enforcement mechanisms allows autonomous systems to maintain compliance with organizational and regulatory requirements using role-based access controls, graduated autonomy, and detailed logging for auditability and operator trust. In operational deployments, these technologies will drive a model shift in the network from one that is human-controlled to one that is self-optimizing with a higher level of automation, improving network performance and reliability in known use cases while deferring new or risky use cases to a human expert. In a highly competitive landscape, first movers will not only realize meaningful cost advantages they will also positively impact service quality, accelerate innovation cycles, and provide greater business agility. In short, this is not simply improved efficiency of network management; it is a rethink of how businesses design, provision, and run the critical infrastructure that supports the digital economy.

References

[1] Cisco Systems, "2024 Global Networking Trends Report". [Online]. Available: https://www.cisco.com/site/us/en/solutions/networking/global-networking-trends/index.html

[2] John Moore, "The cost of downtime and how businesses can avoid it," SearchDataBackup, 2025. [Online]. Available: https://www.techtarget.com/searchdatabackup/feature/The-cost-of-downtime-and-how-businesses-can-avoid-it

[3] Mark Reitblatt et al., "Abstractions for network update," SIGCOMM '12: Proceedings of the ACM SIGCOMM 2012 conference on Applications, technologies, architectures, and protocols for computer communication, 2012. [Online]. Available: https://dl.acm.org/doi/10.1145/2342356.2342427

[4] Ratul Mahajan and Roger Wattenhofer, "On Consistent Updates in Software Defined Networks". [Online]. Available: https://conferences.sigcomm.org/hotnets/2013/papers/hotnets-final108.pdf

[5] Mina Tahmasbi Arashloo, et al., "SNAP: Stateful Network-Wide Abstractions for Packet Processing," SIGCOMM '16: Proceedings of the 2016 ACM SIGCOMM Conference. [Online]. Available: https://dl.acm.org/doi/10.1145/2934872.2934892

[6] Ahmed El-Hassany et al., "Network-wide Configuration Synthesis," arxiv>cs>arXiv:1611.02537, 2016. [Online]. Available: https://arxiv.org/abs/1611.02537

[7] Pratik Patel, "How to Optimize Network Performance and User Experience with AI-Powered Monitoring," Motadata, 2025. [Online]. Available: https://www.motadata.com/blog/how-to-optimize-network-performance-and-user-experience-with-ai-powered-monitoring/
[8] Pankaj Berde et al., "ONOS: towards an open, distributed SDN OS," HotSDN '14: Proceedings of the third workshop on Hot topics in software defined networking, 2014. [Online]. Available: https://dl.acm.org/doi/10.1145/2620728.2620744
[9] Ryan Beckett et al., "Network configuration synthesis with abstract topologies," PLDI 2017: Proceedings of the 38th ACM SIGPLAN Conference on Programming Language Design and Implementation, 2017. [Online]. Available: https://dl.acm.org/doi/10.1145/3062341.3062367
[10] Ari Fogel et al., "A General Approach to Network Configuration Analysis," 12th USENIX Symposium on Networked Systems Design and Implementation (NSDI '15), 2015. [Online]. Available: https://www.usenix.org/system/files/conference/nsdi15/nsdi15-paper-fogel.pdf