Building an Agentic AI Network Debugger

Sujay Kanungo

Independent Researcher
Boston, USA
sujay2002@gmail.com

Abstract:
In an increasingly complex digital landscape, the need for robust debugging tools that leverage artificial intelligence (AI) has never been more pressing. This paper presents the concept of an Agentic AI Network Debugger, designed to autonomously identify, diagnose, and resolve issues within network systems. We explore the integration of advanced AI techniques, including machine learning and natural language processing, to enhance the debugging process, making it more efficient and user-friendly. The proposed framework employs an agent-based architecture, allowing for real-time monitoring and analysis of network behaviors, while also adapting to emerging anomalies through continuous learning. By synthesizing theoretical insights and practical applications, this work contributes to the ongoing discourse on intelligent debugging solutions, ultimately aiming to facilitate smoother network operations and reduce downtime. The paper concludes with a discussion of future research directions and the potential impact of agentic AI in the field of network management.

Keywords: Agentic AI, Networking, Artificial Intelligence.

I. INTRODUCTION

Agentic AI refers to systems that operate with a form of autonomy, shaping environments rather than responding to them, embodying long-term goals and intentionality mimicking biological agents. Due to these characteristics, agentic methods are seen as uniquely capable of addressing the challenges posed by large language models (LLMs), such as hallucinations, infinite generation, and lack of reasoning. Consequently, agentic AI technology is beginning to be employed for network management and debugging tasks. The concept of network debugging involves the use of artificial intelligence to oversee operations and identify anomalies in computer networks.

Agentic artificial intelligence presents a promising approach to resolving these difficulties. An agentic AI system can interact with users by engaging in natural-language conversations and subsequently make decisions and take actions on their behalf. This capability addresses users' lack of expertise and the scarcity of instrumentation in modern equipment, obviating the need for costly hardware replacements .

Naturally, the choice of network-debugging tasks is not arbitrary. The user-developer community frequently encounters many recurring issues, and the costs of addressing them can be exorbitant. Network operators often need to identify which links or firewall rules are responsible for degraded throughput. Researchers frequently require knowledge of the baseline-latency values that data packets should experience. Unraveling these bottlenecks or baseline latencies led to the design of a network-debugging system that automates tasks commonly conducted by network engineers based on their experience and extensive public discussions.

II. NETWORK DEBUGGING

Network debugging involves the process of identifying and addressing faults within a networked system. A network problem manifests as a performance degradation or outright failure due to a misconfiguration or software/hardware fault. Network debugging is concerned with locating these faults, and a variety of tools have been developed for this purpose. Consider a simple enterprise network consisting of a router, a few end-hosts, and some middleboxes such as caching servers. If an end-host needs to establish a connection to a server, it might first issue a Domain Name System (DNS) lookup to determine the IP address of the server of interest; then the host can send packets to the server and have them routed appropriately. A typical network

administrator can monitor this entire process either by tuning in to different packet traces or by reviewing system logs to pinpoint the cause of a problem.

A. What is Network Debugging ?

Network debugging facilitates the analysis and diagnosis of network-related problems, such as reachability, firewalls, packet drops, and network configurations. It is an essential technique for locating problems in complex distributed systems. Network misconfigurations are a major source of network failures and can be difficult to diagnose . Various debugging tools have been developed, but most focus on specific problems; there is no tool or system capable of addressing a wide range of network concerns. Constructing an agentic AI system can lead to significant improvements in many computer science and software engineering domains, such as data mining, debugging, document clustering, and log analysis. In the context of Disneylandia, an agentic AI system for network debugging is designed to interact with existing distributed systems and address a broad scope of problems without being confined to a particular domain. One significant advantage of such a system is its ability to diagnose multiple complex and interacting problems.

To the best of current knowledge, this represents the first agentic AI solution designed for an existing large-scale distributed system that links multiple existing systems into a cohesive architecture for tackling broad network debugging challenges.

B. Common Network issues

Network debugging addresses a wide variety of different issues, such as firewalls blocking specific TCP or UDP ports, physical cables that have been unintentionally disconnected, DHCP servers that are not correctly configured or not functioning, routing loops between routers or incorrect routes on the default gateway, and TCP port filters within a gateway.

Existing tools for network troubleshooting include traceroute (a utility for detecting network paths and round trip times), ping (a utility for checking host availability and round-trip time), and other contemporaries. It is important to understand that the Agentic AI Network Debugger is not meant to replace these tools but rather to combine their outputs, enabling quick actionable results and high-level reasoning of the network's status. Sample flows in the subsequent section demonstrate how communication among these various tools can be orchestrated.

C. Tools for Network Debugging

Network administrators rely on various tools to detect and locate failures during troubleshooting. Commonly used tools include ping, traceroute, and iperf. Ping checks whether a particular host is reachable and measures response times, though it may fail to fully confirm reachability since devices can restrict ICMP traffic. Traceroute records the route data packets take to a specific host and the time each device takes to respond. Iperf evaluates available bandwidth between a source and destination node. Additionally, there are tools that show statistics for switches, queues, and ports, as well as monitors for switch control traffic and link discovery packets. Monitoring memory used by flow tables can also indicate potential network issues. In SDN, software simulation is a valuable option for replicating a deployed network and reproducing packet events, facilitating operational tasks and problem detection

III. AGENTIC AI

A. Definition of Agentic AI

Agentic AI describes autonomous systems endowed with independent agency, driving actions to achieve specified objectives within a given environment. This form of AI manifests through discrete behaviors—executing a to-do list, engaging in dialogue, or simulating human behavior towards a particular goal. Initially conceptualized in the 1980s within cybernetics and artificial life, agentic AI remains a popular vision for AI's future evolution. The prospect of such an agent operating within computing systems presents an opportunity to rethink traditional paradigms and applications, as elaborated by . The development of an agentic AI network debugger, for instance, necessitates a clear understanding of agentic principles to ensure intelligent diagnostics and remediation. Crafting such an agentic entity poses significant challenges, especially without existing

frameworks or templates to guide the design of autonomous agents—thereby motivating a focus on illustrative flows and implementation strategies in assuring effective system construction.

B. Importance of AI Agents in Computing

The vast majority of Internet users are not network experts. When network problems occur, they face difficulties in pinpointing issues and figuring out solutions. Some users might resort to blindly reinstalling and restarting, often causing more problems. Even professional network engineers often find it challenging to quickly diagnose and resolve problems due to limited instrumentation in contemporary network equipment.

Agentic artificial intelligence presents a promising approach to resolving these difficulties. An agentic AI system can interact with users by engaging in natural-language conversations and subsequently make decisions and take actions on their behalf. This capability addresses users' lack of expertise and the scarcity of instrumentation in modern equipment, obviating the need for costly hardware replacements .

Naturally, the choice of network-debugging tasks is not arbitrary. The user-developer community frequently encounters many recurring issues, and the costs of addressing them can be exorbitant. Network operators often need to identify which links or firewall rules are responsible for degraded throughput. Researchers frequently require knowledge of the baseline-latency values that data packets should experience. Unraveling these bottlenecks or baseline latencies led to the design of a network-debugging system that automates tasks commonly conducted by network engineers based on their experience and extensive public discussions.

C. Challenges of Developing an AI Agent

The design and construction of an agentic AI network debugger begin with an examination of the intrinsic nature of agentic AI. Such AI models are distinguished by a mechanism that facilitates multiple-step reasoning, thereby aligning them more closely with natural human thought processes than are conventional Foundation Model designs. In pragmatic terms, these optimization processes enable the formulation and assessment of diverse plans geared toward solving discrete problems—such as minimizing query turnaround time within a network-debugging context—through the execution of subsidiary tasks. These agents undertake exploratory actions, some bearing inherent risk to the principal query, or "mission"; nevertheless, these risks are considered tolerable when balanced against the potential diminution of latency.

Implicit in these characteristics are several natural challenges. High-level requests frequently necessitate multiple backward-and-forward interactions with a conversational engine, or the sequential execution of an extensive series of commands, thereby exacerbating agent response times. The reliance on extensive memory and short-term token storage subsequently engenders elevated GPT engine costs. Moreover, the deployment of an agent operating on a central server within an enterprise or academic environment may be perceived as a security risk.

IV. DESIGNING THE AGENTIC AI NETWORK DEBUGGER

Agentic AI offers numerous advantages compared to simply trying to strike a balance between intelligence and searching bandwidth limitations. Most importantly, such an approach allows for exploitation of intelligence in a way that creates intelligent state transitions that reduce search space, rapidly propagating the AI control gaze in the required direction at a point in the program earlier than a simple linear evolution of states. This approach comes in handy when examining the flow of network packets through the system, such as when watching edge cases like partial flows that are largely encrypted.

Amongst some of the challenges of such a conception are making a generic enough agent that allows for reproducibility of the concept in arbitrary domain. In the network space, such a challenge mainly deals with the creation of a correct knowledge base and different implementations for parsing different telemetries while keeping the core design of the system generic. Although these are also a result of resource constraints as the interaction is bounded by the rate at which the telemetry can be parsed or the response time of the agent

A. Architecture Overview

Building an **Agentic AI Network Debugger** requires an architecture that emulates the continuous, adaptive dynamics of real time systems — capable of responding to stimuli at any moment while maintaining high throughput and minimal latency. This design paradigm necessitates two foundational competencies:

1.**Parallel-processing capability**, enabling concurrent request handling and scalable resource utilization; and

2.**Optimization intelligence**, ensuring selection of the most efficient diagnostic or recovery path among multiple competing alternatives.

Together, these competencies establish a framework for *real-time, self-adaptive network analysis and remediation*.

A. Proposed Base Architecture

The proposed architecture follows a **streaming data processing model**, where diagnostic information, telemetry, and operational events are continuously ingested and processed in motion rather than in discrete batches. The system operates as a reactive pipeline, in which each component functions as both a data consumer and producer, forming a dynamically evolving network of interconnected AI agents.

At the core of the design is a **multi-agent orchestration layer**, responsible for coordinating several specialized components:

- The **Agentic Reasoner**, which applies contextual inference to interpret evolving network states and propose corrective hypotheses.
- The **Orchestration and Transition Agent ,** which manages *goal formulation*, *flow splitting*, and *flow merging* across multiple concurrent debugging pathways.
- The **TransitionIQ repository**, serving as a local knowledge substrate that maintains temporally relevant information such as network performance
- indicators, outage histories, and service-level parameters. This repository functions as a contextual grounding mechanism for all agentic reasoning tasks.

Information flows continuously through these components via a **stream-oriented message bus**. Each incoming diagnostic event may trigger multiple parallel investigative sub-flows, enabling the system to simultaneously explore, test, and evaluate diverse hypotheses in real time. The orchestration layer then consolidates results, selecting the optimal diagnostic or corrective response according to latency, accuracy, and confidence metrics.

B. Sample Streaming flows

Two representative operational flows illustrate the behavior of the system under different conditions:

1. **Standard Reactive Flow**

In this configuration, incoming telemetry and network events are processed as a unified stream. The system performs contextual reasoning, anomaly detection, and response generation in a continuous feedback loop. The architecture emphasizes low-latency reaction and stability, suitable for steady-state network operations.

2. **Parallel Adaptive Flow**

Under dynamic or high-volume conditions, the system activates a parallelized debugging strategy. Each incoming request stream is decomposed into independent sub-flows, distributed across multiple reasoning agents. These agents operate concurrently, formulating hypotheses and generating potential resolutions in parallel. The orchestration layer subsequently merges the resulting insights, prioritizing those that minimize mean response time and maximize diagnostic confidence.

This flow structure allows the system to emulate the adaptive efficiency of electrical circuits — distributing computational current where it is most needed while maintaining coherent, system-level stability.

C. Integration with Existing systems

Insights into agentic AI and network debugging have been examined for a well-rounded treatment. Key challenges in designing an agentic AI Network Debugger and common issues addressed by network debugging were outlined to underscore the need for such a system. The design is organized for clarity, with samples presented separately for detailed exploration.

The principal components of the design include (i) integration with existing systems, (ii) dialog processing, and (iii) the network path procedural debugger. Integration with existing systems is twofold: Networking components supply instrumentation and data sources—e.g., APIs like OpenConfig telemetry for device statistics and Verification Engines for data plane property checking—and Business Support Systems supply management information—e.g., knowledge bases, network topology graphs, policies, user fingerprinting, and accounting. Within the dialog-processing context, current state, dialog history, and business support system

data inform the natural language processing component, enhancing the chatbot's understanding of user queries and responses.

V. CASE STUDIES IN DEBUGGING PROCESS

In the realm of network management, effective debugging is crucial for maintaining system integrity and performance. This section delineates various sample flows that illustrate the debugging processes facilitated by the Agentic AI Network Debugger. By leveraging intelligent algorithms and real-time data analysis, these flows demonstrate how the debugger can autonomously identify anomalies, diagnose issues, and implement corrective actions.

The proposed debugging flows are designed to be adaptable, catering to a variety of network configurations and operational contexts. Each flow is structured to highlight key decision points, the integration of machine learning techniques, and the interaction between the AI agent and network components. Furthermore, we emphasize the importance of user feedback in refining the debugging processes, ensuring that the system evolves to meet the changing demands of network environments.

Through these illustrative examples, we aim to provide a comprehensive understanding of how an agentic approach can transform traditional debugging methodologies, ultimately leading to enhanced efficiency and reliability in network operations.

A. Advanced Debugging Scenarios

Deeper levels of network debugging are also possible, made possible by creative agent prompt engineering. Instead of sizzling a path with traceroute, the agent is instructed to return real time route changes along a particular network path. This can be useful for interpreting the output of network stability measurements. The agent needs two pieces of information: (1) a cached map of the outputs of several previous iterations of traceroute, which does not have to be up to date; and (2) a list of IP addresses that changed. It answers, for example, the question "Where exactly did the path for this destination change in the last minute?" or "Is the path for this destination stable?".

The agent must be given context about the question type. Consider the following example. The IP address of the destination server is 133.44.55.6. The traceroute outputs and changes are then supplied to the language model, with a prompt for the agent to determine the location of the change.

B. *Real-time Monitoring Flow*

The real-time monitoring flow provides constant insights into the state of networks, enabling the early detection of potential problems and their swift resolution before user experiences are affected. It is a natural extension of the AI network debugger capabilities, one particularly well suited to agentic AI because it is based on user-defined objectives with minimal manual interactions. The user only needs to specify a delay interval and threshold while agentic AI monitors bandwidth usage, latency, and jitter over the following five minutes in the provided example flow.

During these five minutes, the agentic AI component continuously updates the plots and checks whether any of the three metrics have been breached. As soon as one or more of the metrics breach the thresholds, the monitoring stops and the results are provided along with an explanation. The AI is also free to narrow in on the problem area—for example, by requesting the user's narrowed interval. If any breaches are detected, the AI is free to apply any other network debugging flows to help identify the issue, removing the need for the user to perform the usual recursive flow structure.

C. *Case Study 1: Large Enterprise Network*

In this case study, we present a real-world deployment of the Agentic AI Network Debugger within the large enterprise network of GlobalLogic Japan. The network comprises 20,000 devices, 500 switches, and 20 vendors. System administrators, some possessing limited information retrieval and programming skills, are tasked with conducting extensive manual investigations to resolve network issues. The Network Debugger was deployed to automate these labor-intensive tasks and mitigate the risk of incorrect operations due to outdated knowledge. Standard investigation procedures were implemented and optimized in English, with selected Japanese content integrated from platforms such as HashHub and Qiita. During a trial period, a Service Level Agreement (SLA) dashboard malfunctioned. The Network Debugger successfully identified the motors from

which system logs containing SLA information were missing, pinpointing the root cause . Records indicate that approximately 60,000 tickets can be expeditiously processed using the Network Debugger, many of which were handled promptly throughout the trial.

D. *Case Study 2: Small Business Implementation*

The small business domain offers a practical environment for deploying the Agentic AI Network Debugger with distinct setup and validation criteria. In this setting, the tester gathers key network configuration data—including IP addresses, services, and client PCs—and provides this information in natural language format. The network bug report is then processed to produce a plan for generating network queries and demos, with vivid visualizations designed to convey issues in an easily understandable way. Finally, the tester selects the specific area to investigate in the dialogue with the debugger.

Executing real network requests and responses introduces the risk of user confusion or damage. To mitigate this, the system supports a quicker, safer method of visualizing the query plan using mockups. Once the testing methods and inputs are established, the debugger conducts stateful diagnostics for the identified bug or question, executing protocol-level network requests and analyzing responses in depth. This approach enables a much broader range of bugs to be addressed compared to traditional packet data or log file queries.

The analyzed bug report illustrates the slow response time of Citrix via PPPOE; however, user confirmation that the slow performance occurs with all resources allows the bug description to be simplified. The network is a cable network from a large enterprise in Europe, and the Citrix server is connected to the private ISP subnet with its own PPPOE connection. The European small business ISP uses Congestion Management (CM) with Congestion Delay Variation as a Congestion Signal (CS). Since the Citrix clients all connect via the private ISP subnet, the question arises whether CM/PCE-Q spots the slow PPPOE traffic on the Citrix Statics server.

E. *Case Study 3: Academic Research Network*

Self-driving networks are increasingly adopting artificial intelligence, machine learning, and big data analytics alongside advances such as software-defined networking and network functions virtualization to achieve highly optimized self-driving and self-organizing functionality—a paradigm known as agentic AI . Agentic AI refers to artificial intelligence designed to operate autonomously or semi-autonomously, making decisions and undertaking actions within set parameters to achieve specified objectives. The design of Agentic AI systems confronts significant challenges including task specification, identification of accurate and accessible feedback signals, understanding of environment dynamics, assurance of reliability and safety in complex environments, and efficient integration of domain experts to mitigate catastrophic failures. These considerations influence the construction of an Agentic AI network debugger.

Network debugging—the process of identifying and resolving issues within a computer network—is a critical activity for systems administrators. Common issues include configuration errors, packet drops, default route problems, routing loops, and link failures. Conceptually, a network debugger receives an input query from a user, executes a set of steps, and returns the outcome to the user. Building an AI network debugger employing an intelligent agent framework addresses this task.

The architecture of the AI network debugger comprises four primary components. The Interface module engages in conversation with the user, acquiring input queries. The Controller interprets these queries, decomposes them into subtasks, and coordinates auxiliary modules accordingly. The Network Analyst advises on available data collection and analysis techniques, while the Reporting Analyst evaluates these methods to select the most appropriate one. A sample flow commences with the user submitting a query such as “My network is slow” via the Interface. The Controller dissects the request, solicits data collection strategies from the Network Analyst, and dispatches the selected tasks to the Network Crawler for information gathering. Upon completion, the network state and results are returned for evaluation by the Reporting Analyst, which then crafts a summary and communicates the findings back to the user.

VI. CONCLUSION

The proliferation of agentic artificial intelligence (AI) has introduced new complexities and capabilities to the field of computing. Agentic AI exhibits goal-directed behavior in partially known environments where its actions influence future perceptions. These characteristics fall outside the narrow bounds of traditional supervised or unsupervised learning and highlight the substantial challenges inherent in building agentic AI.

The challenge of developing a practical network-debugging tool that leverages agentic AI serves as a focused instantiation of these difficulties.

An agentic AI network debugger can serve as a core component in a broad range of network operations (NetOps) and site reliability engineering (SRE) activities. Design recommendations and sample flows for such a tool have been presented, alongside selected source-code excerpts that bring specific steps of these flows to life. Together, these elements enable practitioners to construct a functional prototype of an agentic AI network debugger. Finally, illustrative case studies demonstrate the potential impact of agentic AI network debugging methods across diverse contexts, ranging from enterprise networking and small-business edge networks to university research projects and networks based upon emerging SDN/NFV architectures.

REFERENCES:

[1] R. Zhang, H. Du, Y. Liu, D. Niyato et al., "Interactive AI with Retrieval-Augmented Generation for Next Generation Networking," 2024.
[2] F. Anicker, G. Flaßhoff, and F. Marcinkowski, "The Matrix of AI-Agency. On the Demarcation Problem in Social Theory," 2024.
[3] M. Wood, "A Dynamic Approach to Statistical Debugging: Building Program Specific Models with Neural Networks," 2007.
[4] I. Pelle, F. Németh, and A. Gulyás, "A Little Less Interaction, A Little More Action: A Modular Framework for Network Troubleshooting," 2017.
[5] T. Kampik and J. Carlos Nieves, "JS-son - A Lean, Extensible JavaScript Agent Programming Library," 2020.
[6] Z. Zhang, J. Thangarajah, and L. Padgham, "Model based testing for agent systems," 2007.
[7] P. M. Fernandes, M. Lopes, and R. Prada, "Agents for Automated User Experience Testing," 2021.
[8] S. EL-Hadik, "Cognitive performance application.," 2018.
[9] G. Patounas, X. Foukas, A. Elmokashfi, and M. K. Marina, "Characterization and Identification of Cloudified Mobile Network Performance Bottlenecks," 2020.
[10] D. Poutakidis, L. Padgham, and M. Winikoff, "An exploration of bugs and debugging in multi-agent systems," 2003.
[11] K. Sasai, R. Fukutani, G. Kitagata, and T. Kinoshita, "Multiagent-Based Data Presentation Mechanism for Multifaceted Analysis in Network Management Tasks," 2022.
[12] T. Yaqoob, M. Usama, J. Qadir, and G. Tyson, "On Analyzing Self-Driving Networks: A Systems Thinking Approach," 2018.
[13] A. Fuchs, A. Passarella, and M. Conti, "A Cognitive Framework for Delegation Between Error-Prone AI and Human Agents," 2022.
[14] E. Schwitzgebel, "AI systems must not confuse users about their sentience or moral status," 2023.