Network Management
Internet-Draft
Intended status: Informational
Expires: 21 April 2026

Y. Cui
Tsinghua University
M. Xing
L. Zhang
Zhongguancun Laboratory

18 October 2025

A Framework for LLM Agent-Assisted Network Management with Human-in-the-Loop

draft-cui-nmrg-llm-nm-01

Abstract

This document defines an interoperable framework that facilitates collaborative network management between Large Language Models (LLMs) agents and human operators. The proposed framework introduces an enhanced telemetry module, an LLM agent decision module, and interaction data models between human operators and LLM agent-driven systems, and workflows to enforce human oversight. The approach ensures compatibility with existing network management systems and protocols while improving automation and decision-making capabilities in network operations.

About This Document

This note is to be removed before publishing as an RFC.

The latest revision of this draft can be found at https://xmzzyo.github.io/draft_llm_nm/draft-cui-nmrg-llm-nm.html. Status information for this document may be found at https://datatracker.ietf.org/doc/draft-cui-nmrg-llm-nm/.

Discussion of this document takes place on the Network Management Research Group mailing list (mailto:nmrg@irtf.org), which is archived at https://mailarchive.ietf.org/arch/browse/nmrg. Subscribe at https://www.ietf.org/mailman/listinfo/nmrg/.

Source for this draft and an issue tracker can be found at https://github.com/xmzzyo/draft_llm_nm.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Copyright Notice

Table of Contents

1. Introduction

1.1. Motivation

Traditional network automation systems often face challenges in handling unanticipated scenarios and managing complex multi-domain data dependencies. Large Language Models (LLMs) offer advanced multimodal data comprehension, adaptive reasoning, and generalization capabilities, making them a promising tool for network management and autonomous network[TM-IG1230]. However, full automation remains impractical due to risks such as model hallucination, operational errors, and a lack of accountability in decision-making[Huang25]. This document proposes a structured framework that integrates LLMs into network management through human-in-the-loop collaboration, leveraging their strengths while ensuring oversight, reliability, and operational safety.

1.2. Problem Statement

Network management faces significant challenges, including the complexity of multi-vendor configurations, the real-time correlation of heterogeneous telemetry data, and the need for rapid responses to dynamic security threats. LLMs offer a promising approach to addressing these challenges through their advanced multimodal data understanding and adaptive reasoning capabilities. However, the direct application of LLMs in network management introduces several technical considerations. These include the need for semantic enrichment of network telemetry to enhance LLM comprehension, a decision execution mechanism with confidence-based escalation, and auditability of LLM-generated decisions through provenance tracking. Addressing these requirements is critical to integrating LLM agents effectively into network management workflows while maintaining reliability, transparency, and interoperability.

2. Terminology

2.1. Acronyms and Abbreviations

* LLM: Large Language Model
* RAG: Retrieve Augmented Generation
* MCP: Model Context Protocol
* A2A: Agent-to-Agent Protocol

3. Framework Overview

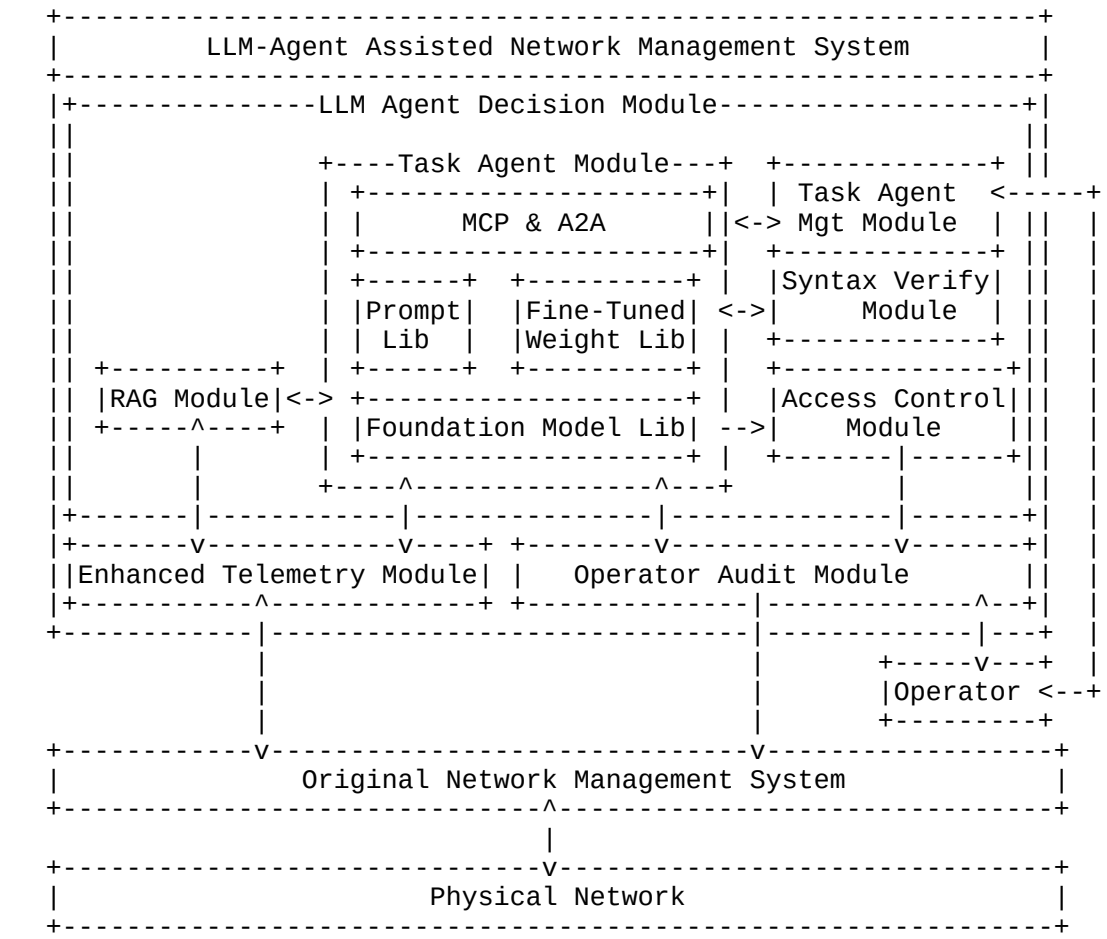

Figure 1: The LLM-Assisted Network Management Framework

The proposed framework is shown in Figure 1, highlighting the key components of LLM agent-assisted network management. The human operator can generate a specific task agent, e.g., fault analysis or topology optimization, using the task agent management module by specifying foundation model, prompt, and adaptation fine-tuned parameters[Hu22]. The enhanced telemetry module improves the semantics of raw telemetry data from original network management system, providing supplementary information to the LLM agent decision module for more informed decision-making. After the decision-making, the generated configuration parameters are validated against the YANG model and enforced with access control rules. The operator audit module provides a structured mechanism for human oversight of LLM-generated configurations, and the configuration can be issued to the original network management system for deployment once the operator approves.

3.1. Enhanced Telemetry Module

The Enhanced Telemetry Module improves the semantics of raw telemetry data, providing supplementary information to the LLM agent decision module for more informed decision-making. Telemetry data retrieved from network devices via NETCONF[RFC6241], e.g., in XML format, often lacks field descriptions, structured metadata, and vendor-specific details. Since this information is not included in the pre-trained knowledge of LLMs, it can lead to misinterpretation and erroneous reasoning. To address this limitation, an external knowledge base should be introduced to store YANG model schema, device manuals, and other relevant documentation. The Enhanced Telemetry Module functions as middleware between the network management system and the external knowledge base. Through its southbound interface, it retrieves NETCONF data from the NETCONF client of existing network management system. Through its northbound interface, the module queries the external knowledge base for the corresponding YANG model or device manual. To enhance semantic richness, the Enhanced Telemetry Module processes the retrieved data by simplifying formatted content (e.g., removing redundant or closing XML tags) and appending XML path and description information from the YANG tree to the relevant fields. This approach ensures that the LLM has access to structured, contextually enriched data, improving its ability to analyze and reason about network telemetry.

3.2. LLM Agent Decision Module

3.2.1. RAG Module

The pre-trained LLM may not encompass specific task requirements or vendor-specific knowledge. To address this kind of limitation, the Retrieve-Augmented Generation (RAG)[Lewis20] approach is widely used. This module retrieves relevant information from operator-defined sources, such as device documentation and expert knowledge, and integrates it with the Enhanced Telemetry Module to obtain YANG model schema. The retrieved textual data is stored in a database, either as raw text or in a vectorized format for efficient search and retrieval. For a given task context, the module retrieves relevant knowledge from the database and incorporates it into the input context, improving the understanding and response accuracy of LLM.

3.2.2. Task Agent Module

To execute a specific task, such as traffic analysis, traffic optimization, or fault remediation, a corresponding task agent must be created. A task agent consists of a selected LLM foundation model, an associated prompt, and fine-tuned weights.

* Foundation Model Library. Operators must select an appropriate foundation model based on the specific task requirements. Examples include general-purpose models such as GPT-4, LLaMA, and DeepSeek, as well as domain-specific models fine-tuned on private datasets. Since foundation models are trained on different datasets using varying methodologies, their performance may differ across tasks.

* Fine-Tuned Weight Library. For domain-specific applications, fine-tuned weights can be applied on top of a foundation model to efficiently adapt it to private datasets. One commonly used approach is to store the fine-tuned weights as the difference between the original foundation model and the adapted model, which can largely reduce storage requirements. The Fine-Tuned Weights Module supports the selection and loading of an appropriate foundation model along with the corresponding fine-tuned weights, based on the selection of operators. This ensures flexibility in leveraging both general-purpose and domain-specific knowledge while maintaining computational efficiency.

* Prompt Library. For each task, it is essential to accurately define the task description, the format of its inputs and outputs. These definitions are stored in a structured prompt library. When an operator instantiates a task, the corresponding prompt, including placeholders for contextual information, is automatically retrieved. operator inputs and device data are then incorporated into the prompt at the designated placeholders, ensuring a structured and consistent interaction with the language model.

3.2.3. Task Agent Communication Module

Task agent can interact with external tools (e.g., Python scripts, Batfish or ILP solver) to acquire additional knowledge and perform specific actions. The Model Context Prtocol (MCP)[mcp] enables task agents to securely and efficiently interact with external tools and services. MCP defines a standardized interface for tool invocation, data exchange, and state synchronization between the agent and external systems. MCP consists of two primary components:

* MCP Client: Embedded within the task agent, the MCP client is responsible for:
  - Serializing agent-generated tool invocation requests into structured MCP messages.
  - Managing authentication and session tokens for secure tool access.
  - Handling timeouts, retries, and error propagation from tool responses.
  - Injecting tool outputs back into the agent context as structured observations.
* MCP Server: Hosted alongside or within external tools, the MCP server:
  - Exposes a well-defined set of capabilities via a manifest (e.g., tool_name, tool_description, input_schema, output_schema, authentication_method).
  - Validates incoming MCP requests against the tool’s schema and permissions.
  - Executes the requested operation and returns structured results in a deterministic format.

- Supports streaming for long-running operations (e.g., iterative optimization or real-time telemetry polling).

In many complex scenarios, multiple task agents may collaborate to achieve a shared network management objective. The Agent-to-Agent Protocol (A2A)[a2a] is a coordination protocol that enables multiple task agents to exchange information, delegate subtasks, negotiate resource usage, and synchronize execution states in a distributed network management environment. A2A ensures semantic interoperability, temporal consistency, and conflict avoidance across heterogeneous agents. Key design principles of A2A include:

* Decentralized orchestration: No central controller is required; agents coordinate peer-to-peer using shared intents and commitments.

* Intent-based messaging: Communication is driven by high-level intents (e.g., “optimize latency for flow X”) rather than low-level commands, allowing flexibility in implementation.

* State-aware handoffs: Agents share partial execution states (e.g., intermediate results, constraints, confidence scores) to enable context-preserving collaboration.

Additionally, A2A integrates with the Task Agent Management Module to dynamically spawn or terminate agents based on collaboration needs, ensuring scalability and resource efficiency.

3.2.4. Task Agent Management Module

The Task Agent Management Module is responsible for the creation, update, and deletion of task agents. This module ensures that each agent is appropriately configured to align with the intended network management objective.

The agent lifecycle operations are defined as follows:

1. Creation of Task Agents. A task agent is instantiated in response to an operator request, an automated policy trigger, or a higher-level orchestration workflow. The creation process involves the following steps:

    * Intent Parsing: The module parses the high-level intent (e.g., “remediate BGP flapping on router R5”) to identify the required task type, target network scope, and performance constraints.

* Agent Template Selection: Based on the parsed intent, the module selects a pre-registered agent template from the Prompt Library.

* Resource Allocation: The module allocates necessary compute resources (CPU/GPU, memory) and instantiates the LLM runtime environment.

* Context Initialization: The agent is initialized with: Network context (e.g., device inventory, topology from Enhanced Telemetry), security credentials, and session ID and logging context for auditability.

* Registration: The newly created agent is registered with its metadata, status (“initializing”), and heartbeat endpoint.

2. Update of Task Agents. Task agents may require updates due to changing network conditions, model improvements, or revised operator policies. Updates should be performed in a non-disruptive manner whenever possible:

   * Configuration Update: Operators or automated controllers may modify agent parameters (e.g., optimization thresholds, output verbosity).

   * Model or Weight Swapping: If a newer fine-tuned weight version becomes available, the module can hot-swap the adapter weights while preserving the agent’ s execution state, provided the base foundation model remains compatible.

   * State Preservation: During updates, the module snapshots the agent’ s working memory (e.g., conversation history, intermediate plans) and restores it post-update to maintain task continuity.

3. Deletion of Task Agents. Task agents are terminated when their assigned task completes, when an unrecoverable error occurs, or upon an explicit teardown request. The deletion process ensures proper resource reclamation and audit compliance:

   * Graceful Shutdown: The module issues a termination signal, allowing the agent to complete any pending operations (e.g., commit configuration changes, MCP calls, A2A communications).

   * State Archival: The final agent state, including input context, generated actions, and performance metrics, is serialized and stored in the Audit Log for replayability and compliance.

* Resource Deallocation: Compute resources (GPU memory, threads) are released, and MCP sessions are invalidated.

* Deregistration: The agent entry is removed, and its lifecycle event is logged.

By providing structured, auditable, and policy-governed lifecycle management, the Task Agent Management Module enables scalable and trustworthy deployment of LLM Agent-driven network automation.

3.3. Config Verification Module

3.3.1. Syntax Validation Module

To ensure correctness and policy compliance, LLM-generated configurations must pass the YANG schema validation steps before being queued for human approval. This module ensures that only syntactically correct configurations are presented for operator review, thereby reducing errors and enhancing network reliability.

3.3.2. Access Control Module

Although the Configuration Verify Module can guarantee the syntactic correction, LLMs may generate unintended or potentially harmful operations on critical network devices, it is essential for operators to enforce clear permission boundaries for LLM task agent to ensure security and operational integrity. The Network Configuration Access Control Model defined in [RFC8341] provides a framework for specifying access permissions, which can be used to grant access to LLM task agents. This data model includes the concepts of users, groups, access operation types, and action types, which can be applied as follows:

* User and Group: Each task agent should be registered as a specific user, representing an entity with defined access permissions for particular devices. These permissions control the types of operations the LLM is authorized to perform. A task agent (i.e., user) is identified by a unique string within the system. Access control can also be applied at the group level, where a group consists of zero or more members, and a task agent can belong to multiple groups.

* Access Operation Types: These define the types of operations permitted, including create, read, update, delete, and execute. Each task agent may support different sets of operations depending on its assigned permissions.

* Action Types: These specify whether a given operation is permitted or denied. This mechanism determines whether an LLM request to perform an operation is allowed based on predefined access control rules.

* Rule List: A rule governs access control by specifying the content and operations a task agent is authorized to handle within the system.

This module must enforce explicit restrictions on the actions an LLM is permitted to perform, ensuring that network configurations remain secure and compliant with operational policies.

3.3.3. Feedback Module

As the generated configuration might not be always aligned with YANG model schema, access control, or bussiness constrains, they need further modification to meet the multi-dimensional requirements. The Feedback Module is used to provide semantic-rich feedback (e.g., represented in structure text) and hints to LLM agent to improve the generated configuration.

3.4. Operator Audit Module

The Operator Audit Module provides a structured mechanism for human oversight of LLM-generated configurations before deployment. The output from the LLM Decision Module should include both the generated configuration parameters and an associated confidence score. The configuration parameters are validated for compliance with the YANG model and are subject to Access Control Rules enforcement. The confidence score, e.g., ranging from 0 to 100, serves as a reference for operators to assess the reliability of the generated configuration. Each audit process must track the input context (e.g., input data, RAG query content, model selection, configuration files) and the corresponding output results. The auditing steps are as follows:

* Result Verification: The operator verifies the LLM-generated output to ensure alignment with business objectives and policy requirements.

* Compliance Check: The operator ensures that the LLM output adheres to regulatory standards and legal requirements.

* Security Verification: The operator examines the output for potential security risks, such as misconfigurations or vulnerabilities.

* Suggestions and Corrections: If issues are identified, the operator documents the findings and proposes corrective actions.

Upon completing the audit, the system maintains an audit decision record to ensure traceability of operator actions. The audit record includes the following information:

* Timestamp of the audit action

* LLM Task Agent ID associated with the action

* Operator decisions, including approval, rejection, modification, or pending status

* Executed command reflecting the final action taken

* Operation type (e.g., configuration update, deletion, or execution)

This structured approach ensures that all LLM-generated configurations undergo rigorous human review, maintaining operational accountability, and security.

The Operator Audit Module is enhanced with Explainability functionality to improve transparency and trust in LLM-assisted decision-making. This enhancement ensures that operators can understand the reasoning behind LLM-generated configurations, verify their reliability, and take corrective actions when necessary.

To achieve Explainability, the module provides a structured breakdown of the decision-making process by capturing key influencing factors, justification logs, and confidence scores. Each LLM-generated configuration must include a detailed rationale explaining why a particular decision was made. For example, if the system recommends increasing bandwidth allocation, the decision log should indicate whether this was due to high latency detected in telemetry, SLA thresholds being exceeded, or other contributing factors.

Additionally, the audit process incorporates counterfactual analysis, allowing operators to assess alternative outcomes. For instance, the system may indicate that if no action is taken, packet loss is expected to increase by 20% in the next ten minutes. This capability enhances operational decision-making by providing a comparative assessment of different actions.

To further ensure reliability, the module includes a mechanism to detect potential biases or hallucinations in LLM-generated outputs. If an LLM agent decision is based on incomplete or uncertain data,

the system flags it accordingly, allowing operators to make informed judgments. For example, if real-time telemetry data is insufficient, the system may indicate that the confidence in a particular recommendation is low.

By integrating these Explainability features, the Operator Audit Module strengthens human oversight, ensuring that LLM-generated configurations align with business policies, security requirements, and regulatory standards. This structured approach enhances accountability and mitigates risks associated with automated decision-making.

4. Use Cases

4.1. DDoS Intelligent Defense

Distributed Denial of Service (DDoS) attacks remain a critical operational threat. While conventional systems use rate-limiting and signature matching, they often fail to adapt in real-time or generate fine-grained filtering rules based on multi-dimensional telemetry data.

This use case demonstrates how the LLM agent-assisted framework enables intelligent filtering rule generation and secure deployment with human oversight.

1. *Telemetry Collection and Semantic Enrichment* The Enhanced Telemetry Module retrieves real-time traffic statistics and interface metrics from network devices using NETCONF. These raw telemetry data are semantically enriched by referencing YANG models and device-specific documentation to generate a context-rich dataset suitable for LLM processing.

2. *DDoS Filtering Task Instantiation* The operator initiates a ddos-mitigation task. The Task Agent Module selects a security-specialized foundation model and task-specific prompt.

   It first analyzes the following conclusions: - Interface GigabitEthernet0/1 receiving sustained traffic > 100,000 pps - 95% of incoming packets are TCP SYN - Top source prefixes identified as: IP1/24, IP2/24

   The RAG Module supplements the analysis with: - Cisco ACL syntax documentation - Prior incident response templates

3. *LLM-Generated Firewall Configuration Output* The LLM agent reasons that a TCP SYN flood is underway and generates the following ACL-based filtering policy:

```
ip access-list extended BLOCK-DDOS
deny tcp IP1 0.0.0.255 any
deny tcp IP2 0.0.0.255 any
permit ip any any

interface GigabitEthernet0/1
ip access-group BLOCK-DDOS in
```

This configuration includes both source-based filtering and a global SYN rate-limiter. It is passed to the Config Verify Module for syntax validation and to the Access Control Module for permission checks.

1. *Operator Audit and Decision* The Operator Audit Module provides the following metadata:

* *Task Agent ID*: ddos-mitigation-task-01

* *Confidence Score*: 85/100

* *RAG Context*: Cisco IOS ACL Syntax, Internal Threat List v5

* *Input Summary*:

  - Affected Interface: GigabitEthernet0/1

  - Malicious Sources: IP1/24, IP2/24

The human operator performs the audit with the following steps:

* *Result Verification*: ACL syntax matches Cisco IOS 15.6 standards.

* *Compliance Check*: Confirmed that both prefixes appear in the enterprise blacklist.

* *Security Review*: Low false-positive risk for blanket deny of SYN traffic from identified prefixes.

* *Audit Record*:

  - Timestamp: 2025-07-16T17:03:00Z

  - Decision: _Approved_

  - Action: Apply via NMS interface

- Operator Note: “Confirmed match with active threat list. Monitor for collateral traffic drops.”

The configuration is deployed through the network management system, completing the defense workflow with human oversight and traceability.

This simplified case shows how the LLM agent can assist in quickly generating actionable filtering rules in response to DDoS patterns, while adhering to operational safeguards.

4.2. Traffic Scheduling and Optimization

In large-scale networks, dynamic traffic scheduling is required to adapt to fluctuating loads, ensure QoS, and meet SLA requirements. Traditional methods may lack responsiveness or cross-domain visibility.

The proposed framework enables intelligent, context-aware traffic scheduling through LLM agent-based reasoning. The following illustrates the process:

1. *Telemetry Data Acquisition* The Enhanced Telemetry Module gathers link utilization, queue occupancy, and delay metrics from multiple routers. The semantic enrichment process tags each metric with human-readable labels from the YANG model, including path topology and policy tags (e.g., gold/silver/bronze service classes).

2. *Optimization Task Execution* An operator initiates a traffic scheduling task. The Task Agent Module selects a foundation model fine-tuned on traffic engineering datasets and uses a structured prompt to describe current constraints: high utilization on core links L1-L3, SLA violations for gold-class VoIP traffic.

3. *Sample Configuration Output* The LLM agent suggests adjusting RSVP-TE path metrics to reroute gold traffic via underutilized backup paths:

```
policy-options {
    policy-statement reroute-gold {
        term gold-traffic {
            from {
                community gold-voip;
            }
            then {
                metric 10;
                next-hop [IP];
            }
        }
    }
}
protocols {
    rsvp {
        interface ge-0/0/0 {
            bandwidth 500m;
        }
    }
}
```

This configuration is syntax-validated and checked against the Access Control Module to ensure that traffic engineering policy updates are permitted for this task agent.

1. *Operator Audit and Decision* The Operator Audit Module presents:

* *Confidence Score*: 91/100

* *Audit Metadata*:

  - Task Agent: traffic-opt-te-v2

  - Input Context: “Link L1 (95% utilization), gold-voip SLA latency breach”

  - Recommendation: “Increase TE path metric to reroute VoIP via L2 backup path”

The operator reviews:

* *Result Verification*: Simulates expected path shift via NMS.

* *Compliance Check*: Confirms SLA routing rules permit TE adjustment.

* *Security Review*: Confirms backup link L2 is secure and isolated.

* *Final Action*: _Approved with modification_: “Set bandwidth cap to 400m on backup to avoid overuse.”

The revised configuration is stored and forwarded to the network management system for application.

This use case illustrates how the framework enables LLM agents to propose adaptive, policy-compliant traffic engineering strategies while maintaining operator control, traceability, and auditability.

5. IANA Considerations

This document includes no request to IANA.

6. Security Considerations

* Model Hallucination: A key challenge is that, without proper constraints, the LLM agent may produce malformed or invalid configurations. This issue can be mitigated using techniques such as Constrained Decoding, which enforces syntactic correctness by modeling the configuration syntax and restricting the output to conform to predefined rules during the generation process.

* Training Data Poisoning: LLMs can be trained on malicious or biased data, potentially leading to unintended behavior or security vulnerabilities. To mitigate this risk, LLMs should be trained on curated, high-quality datasets with rigorous validation and filtering processes. Periodic retraining and adversarial testing should also be conducted to detect and correct anomalies before deployment.

7. References

7.1. Normative References

[RFC6241] Enns, R., Ed., Bjorklund, M., Ed., Schoenwaelder, J., Ed., and A. Bierman, Ed., "Network Configuration Protocol (NETCONF)", RFC 6241, DOI 10.17487/RFC6241, June 2011, <https://www.rfc-editor.org/rfc/rfc6241>.

[RFC8341] Bierman, A. and M. Bjorklund, "Network Configuration Access Control Model", STD 91, RFC 8341, DOI 10.17487/RFC8341, March 2018, <https://www.rfc-editor.org/rfc/rfc8341>.

7.2. Informative References

[a2a] "Announcing the Agent2Agent Protocol (A2A)", July 2025, <https://developers.googleblog.com/en/a2a-a-new-era-of-agent-interoperability/>.

[Hu22] Hu, E. J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., Wang, L., and W. Chen, "LoRA Low-Rank Adaptation of Large Language Models", n.d..

[Huang25] Huang,, L., Yu, W., Ma, W., Zhong, W., Feng, Z., Wang, H., Chen, Q., Peng, W., Feng, X., Qin, B., and T. Liu, "A Survey on Hallucination in Large Language Models Principles, Taxonomy, Challenges, and Open Questions", n.d..

[Lewis20] Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., Kttler, H., Lewis, M., Yih, W.-t., Rocktschel, T., and S. Riede, "Retrieval-augmented generation for knowledge-intensive NLP tasks", n.d..

[mcp] "Model Context Protocol", July 2025, <https://modelcontextprotocol.io/introduction>.

[TM-IG1230] McDonnell, K., Machwe, A., Milham, D., O' Sullivan, J., Niemller, J., Varvello, L. F., Devadatta, V., Lei, W., Xu, W., Yuan, X., and Y. Stein, "Autonomous Networks Technical Architecture", February 2023.

Acknowledgments

We thanks Shailesh Prabhu from Nokia for his contributions to this document.

Appendix

Appendix A.1 Data Model

This section defines the essential data models for LLM agent-assisted network management, including the LLM agent decision response and human audit records.

LLM Response Data Model

The LLM agent decision module should respond with the generated configuration parameters along with an associated confidence score. If the LLM is unable to generate a valid configuration, it should return an error message accompanied by an explanation of the issue.

```
module: llm-response-module
  +--rw llm-response
     +--rw config?         string
     +--rw confidence?     uint64
     +--rw error-reason?   enumeration
```

The LLM response YANG model is structured as follows:

```
module llm-response-module {
  namespace "urn:ietf:params:xml:ns:yang:ietf-nmrg-llmn4et";
  prefix llmresponse;
  container llm-response {
    leaf config {
      type string;
    }
    leaf confidence {
      type uint64;
    }
    leaf error-reason {
      type enumeration {
        enum unsupported-task;
        enum unsupported-vendor;
      }
    }
  }
}
```

Human Audit Data Model

This data model defines the structure for human audit operations and record-keeping. It facilitates collaborative decision-making by allowing LLMs to generate actionable insights while ensuring that human operators retain final operational authority.

```
module: human-audit-module
  +--rw human-audit
     +--rw task-id?      string
     +--rw generated-config?   string
     +--rw confidence?         int64
     +--rw human-actions
        +--rw operator?          string
        +--rw action?            enumeration
        +--rw modified-config?   string
        +--rw timestamp?         yang:date-and-time
```

The human audit YANG model is structured as follows:

```
   module human-audit-module {
     namespace "urn:ietf:params:xml:ns:yang:ietf-nmrg-llmn4et";
     prefix llmaudit;
     import ietf-yang-types { prefix yang; }

     container human-audit {
       leaf task-id {
         type string;
         }
       leaf generated-config {
         type string;
         }
       leaf confidence {
         type int64;
         }
       container human-actions {
         leaf operator {
           type string;
           }
         leaf action {
           type enumeration {
             enum approve;
             enum modify;
             enum reject;
             }
           }
         leaf modified-config {
           type string;
           }
         leaf timestamp {
           type yang:date-and-time;
           }
       }
     }
   }
```

Authors' Addresses

Yong Cui
Tsinghua University
Beijing, 100084
China
Email: cuiyong@tsinghua.edu.cn
URI: http://www.cuiyong.net/

Mingzhe Xing
Zhongguancun Laboratory
Beijing, 100094
China
Email: xingmz@zgclab.edu.cn

Lei Zhang
Zhongguancun Laboratory
Beijing, 100094
China
Email: zhanglei@zgclab.edu.cn