Network Management Working Group
Internet-Draft
Intended status: Informational
Expires: 4 January 2026

Y. Yang
Q. Wu
Huawei
D. Lopez
Telefonica
N. R. Moreno
Deutsche Telekom
L. Tailhardat
Orange Research

3 July 2025

Applicability of MCP for the Network Management
draft-yang-nmrg-mcp-nm-00

Abstract

The application of MCP in the network management field is meant to develop various rich AI driven network applications, realize intent based networks management automation in the multi-vendor heterogeneous network environment. This document discusses the applicability of MCP to the network management in the IP network that utilizes IETF technologies. It explores operational aspect, key components, generic workflow and deployment scenarios. The impact of integrating MCP into the network management system is also discussed.

Table of Contents

1. Introduction

The Model Context Protocol (MCP) provides a standardized way for LLMs to access and utilize information from different sources, interact with tools, making it easier to build AI applications that can interact with external LLM and network management systems.

MCP has been seen as rapid adoption internet technology. The application of MCP in the network management field is meant to develop various rich AI driven network applications, realize intent based networks management automation in the multi-vendor heterogeneous network environment. By establishing standard interfaces for tool encapsulation, intent translation, and closed-loop execution within the network management system, MCP enables AI Agents to have:

* Unified operation abstraction through normalized MCP tool definitions

* Seamless LLM integration via the structured protocol

* Automation Execution Ability

This document discusses the applicability of MCP to the network management plane in the IP network that utilizes IETF technologies. It explores operational aspect, key components, generic workflow and deployment scenarios. The impact of integrating MCP into the network management system will also be discussed.

2. Terminology & Notation Conventions

The following terms are used throughout this document:

2.1. MCP

* *MCP Protocol*: MCP is an open standard designed to facilitate communication between LLMs and external data sources or tools.

* *MCP Host*: The entity initiating the LLM request.

* *MCP Client*: A built-in module within a host, specifically

designed for interaction with the MCP server.

* *MCP Server*: A dedicated server that interacts with MCP clients and provides tools.

* *CLI*: Command Line Interface

2.2. Others

* *LLM*: Large Language Model

* *NETCONF*: Network Configuration Protocol [RFC6241]

* *RESTCONF*: RESTful Network Configuration Protocol [RFC8040]

* *SNMP*: A Simple Network Management Protocol [RFC2576]

3. Overview of key challenges for the network management

In large scale network management environment, a large number of devices from different vendors need to be uniformly managed, which can lead to the following issues or challenges:

3.1. Inconsistent YANG Model Support

Different vendors implement different YANG models (standard or proprietary), leading to:

* Lack of uniform data structures for configuration/state retrieval.

* Requirement for vendor-specific adaptations in automation scripts.

Also IETF standard device models has slow adoption. Similar device models are defined in Openconfig or other SDOs, therefore the current YANG device models ecosystem is fragmented.

3.2. Partial or Non-Standard Network Management Protocols Implementations

Some vendors only partially support standard Network management protocols, and proprietary extensions may break interoperability. Other vendors might choose non-stanard network management protocol or telemetry protocol such as gnmi [I-D.openconfig-rtgwg-gnmi-spec], grpc [I-D.kumar-rtgwg-grpc-protocol]. A significant number of network operators continue to rely on legacy network management mechanisms such as SNMP.

3.3. YANG Models Lack Integration with Open APIs

Today, open API has been widely adopted by the northbound interface of OSS/BSS or Network orchestrators while YANG data models have been widely adopted by the northbound interface of the network controller/ orchestrator or the southbound interface of the network controller/ orchestrator. However Open API ecosystem and YANG model ecosystem are both built as silo and lack integration or mapping between them.

4. Architecture Overview

The LLM model with MCP support and its ability to comprehend diverse complex requirements and deliver corresponding functionalities, is well-suited for large scale multi-vendor network management environments, effectively addressing the aforementioned challenges in Section 3. Therefore, we have introduced the MCP protocol in the network management environments for building an intelligent network management and control platform.

4.1. Encapsulating Device Operations into MCP Tools

* Objective: Standardize device operations into modular, reusable tools.

* Implementation:

  - Tool Abstraction: Vendor-specific commands are wrapped into discrete MCP Tools with uniform schemas.

  - Tool Registry: A centralized database stores MCP Tools with metadata (e.g., names, descriptions, parameters).

* Benefits:

  - Eliminates manual translation of commands across different vendors

  - Enabling the plug-and-play integration of new device types.

4.2. LLM for Intent-to-Tool-Request Translation

* Objective: Allow AI models (such as Claude) to understand natural language commands and trigger operations.

* Workflow:

  - Intent Recognition: The LLM first analyzes the user’s natural language query to identify：

    o The user’s intent or goal

    o Required actions or operations

    o Entities, parameters, and constraints mentioned

    o Context from previous interactions

  - Tool Discovery and Toolchain Generation: The LLM access tool descriptions provided by MCP servers, and matches the identified intent with available tools.

  - Parameter Extraction and Mapping: The LLM maps natural language references to structured parameter names and extracts relevant information from the user query.

  - Structured Invocation Generation: The LLM generates properly formatted tool calls following MCP’s protocol.

* Benefits:

  - Bridge natural language to tool invocation requests in a fixed format, then return this request to the client, enabling the client to properly parse the request.

4.3. Closed-Loop Automation Execution Workflow

* Objective: Realize the closed loop of "voice/text commands → automatic execution".

* A general workflow is as follows:

  - User Input Submission: An operator submits a natural language request to the MCP client. And The MCP client forwards this request to the LLM.

- LLM Intent Processing: The LLM parses the input, identifies the operational intent, and forwards a structured request to the MCP client, which queries the MCP Server to retrieve the available tools. The information would include the functional description, required parameters of tools.

- LLM Toolchain Decision:

  - The LLM evaluates the context and if tools are required, select and sequence tools.

  - The decision is sent back to the MCP Client and then MCP Client will execute tools via server.

- Tool Execution: The MCP Server executes the translated commands on target devices and returns results to the client.

- Result Aggregation & Feedback: The MCP Client collates tool outputs (success/failure logs) and forwards them to the LLM for summarization.

* Benefits:

  - Tools safely retry/rollback.

  - Full traceability of LLM decisions and tool executions.

4.4. Example

Take multi-vendor network management as an example, the MCP server is deployed locally on the network controller, and the tools are integrated into the MCP server. The server provides the following registered tool descriptor information:

Tools description: It describes the name, use, and parameters of tools.

Tools implementation: MCP implementation describes how the tools are invoked.

See Tool descriptor information example as follows:

```
# Tool Descriptor
[
  {
    "name": "batch_configure_devices",
    "description": "Batch Configure Network Devices",
    "parameters": {
      "type": "object",
      "properties": {
        "device_ips": {"type": "array", "items": {"type": "string"}, "description": "Devi
ce IP List"},
        "commands": {"type": "array", "items": {"type": "string"}, "description": "CLI Se
quence"},
        "credential_id": {"type": "string", "description": "Credential ID"}
      },
      "required": ["device_ips", "commands"]
    }
  },
  {
    "name": "check_device_status",
    "description": "Check the Status of Network Devices",
    "parameters": {
      "type": "object",
      "properties": {
        "device_ip": {"type": "string"},
```

```
        "metrics": {"type": "array", "items": {"enum": ["cpu", "memory", "interface"]}}
      }
    }
  }
]
# Tool Implementation
from netmiko import ConnectHandler
from mcp_server import McpServer

app = FastAPI()
server = McpServer(app)

#Connection Pool Management
devices = {
    "192.168.1.1": {"device_type": "VendorA-XYZ", "credential": "admin:XYZ@password"},
    "192.168.1.2": {"device_type": "VendorB-ABC","credential":"admin:ABC@passowrd"}
     ....
}

@server.tool("batch_configure_devices")
async def batch_config(device_ips: list,commands: list,credential_id: str):
    results = {}
    for ip in device_ips:
        conn = ConnectHandler(
            ip = ip,
            username = devices[ip]["credential"].split(':')[0],
            password = devices[ip]["credential"].split(':')[1],
            device_type = devices[ip]["device_type"]
        )
        output = conn.send_config_set(commands)
        results[ip] = output
    return {"success": True, "details": results)

@server.tool("check_device_status")
async def check_status(device_ip: str, metrics: list):
    status = {}
    if "cpu" in metrics:
        status["cpu"] = get_cpu_usage (device_ip)
    if "memory" in metrics:
        status["memory"] = get_memory_usage(device_ip)
    return status
```

Suppose a user submits a request (via the client) such as "Configure OSPF Area 0 with process ID 100 for all core switches in the Beijing data center," the MCP client retrieves the necessary tooling descriptor information from the MCP server and forwards it along with the request to the LLM. The LLM determines the appropriate tools and responds in JSON format as follows:

```
{
"method": "batch_configure_devices",
"params": {
   "device_ips":["192.168.10.1",....,"192.168.10.10"],
   "command": [
     "router ospf 100",
     "network 192.168.0.0 0.0.255.255 area 0"
   ]
 }
}
}
```

The MCP server executes the network management operation in JSON format and returns the results to the MCP client, which forwards them to the LLM. The LLM parses the response, generates a natural-language summary, and sends it back to the client for final presentation to the user. See natural lanauge summary example as

follows:

```
{
  "192.168.10.1": "Configure Successfully, take 2.3 seconds",
  "192.168.10.2": "Error: no response from the device",
}
```

5. Deployment Considerations

This section describes MCP deployment requirements for network management environments, followed by implementation scenarios. Key architectural requirements include:

* Function-Specific MCP Servers: To maintain proper architecture and performance with growing tool volumes, servers should be categorized by network management function. Typical categories include network log analysis, device configuration management, energy consumption management, and security operations.

* Secure and Scalable Architecture: The architecture must:

  - Enforce strict access controls limiting MCP operations to authorized AI models and users

  - Scale efficiently with increasing network device counts while maintaining performance

* Automated Workflows: MCP implementations should support LLM-coordinated automation of:

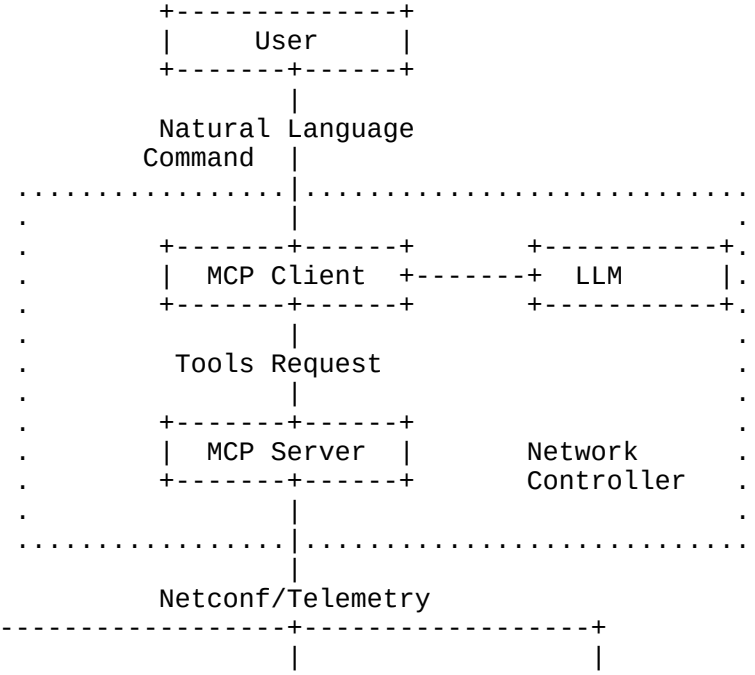
  - Real-time monitoring and diagnostics

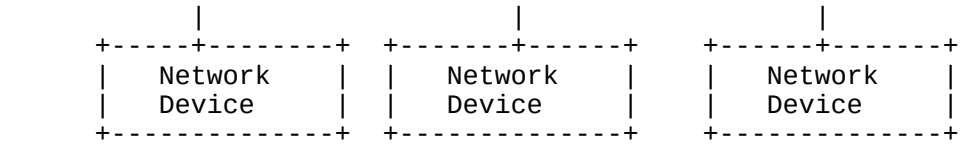
  - Fault remediation workflows

  - Other common management operations to reduce operator workload

While these core requirements apply universally, operational characteristics vary based on deployment location (on-premises vs. remote). The following subsections detail these deployment scenarios.

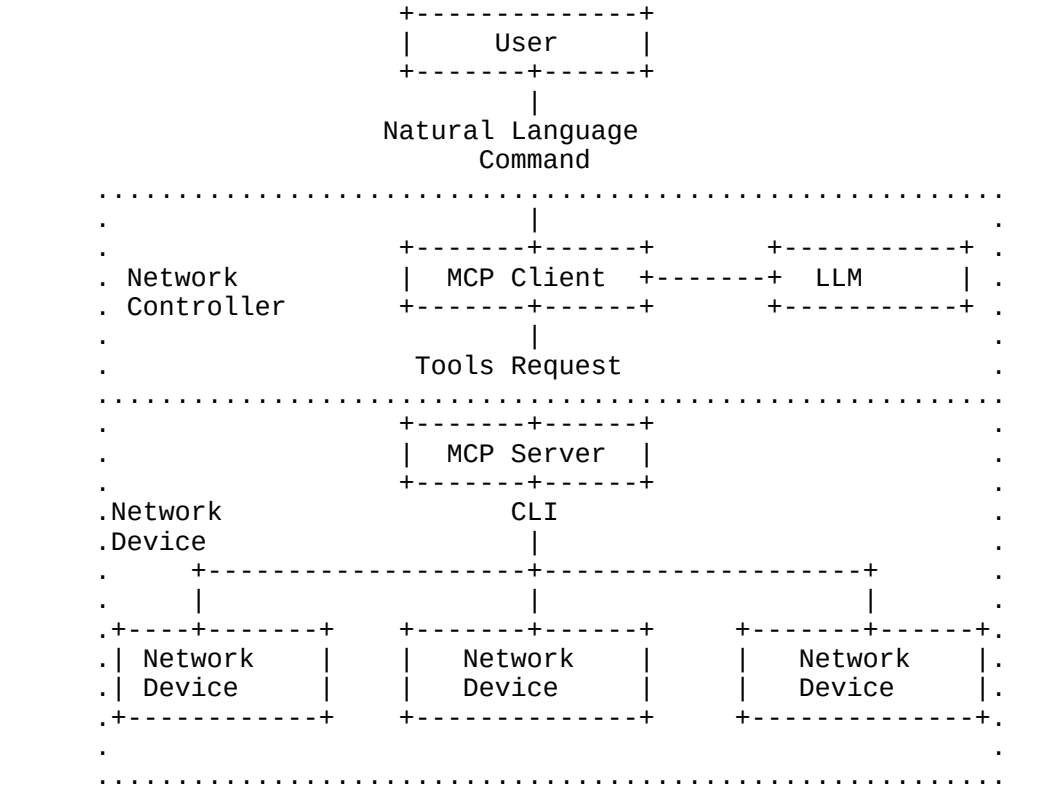
5.1. MCP hosted within the Network Controller





In this senario, the MCP server is deployed within the network controller, which could potentially be a cloud environment. The MCP server acts as a protocol converter, transforming NETCONF's XML/YANG models into JSON-RPC 2.0 format for AI system. - Scope: The MCP Server, colocated with the MCP Client and LLM model, is hosted within a cloud environment, the network devices stay as it is. - Key Characteristics: - Centralized Management: A single MCP Client instance can manage all network devices in geographically dispersed network. - Scalability: Cloud-native scaling accommodates dynamic tool registry updates and high request volumes.

5.2. MCP Server Hosted Within the Network Device



In this senario, the MCP server is deployed within the network devices. The MCP server acts as a protocol converter, transforming CLI into JSON-RPC 2.0 format for AI system. - Scope: The MCP server is colocated with network devices. The MCP Client operates in a cloud environment, requesting distributed MCP Server via public/ private APIs. - Key Characteristics: - Low Latency: Direct access to network devices minimizes tool execution delays

6. Impact of integrating MCP on Network Management

	MCP Hosted Within the Network Controller	MCP Server Hosted Within Network Device
Management Protocol	No impact,reuse existing NM Protocols	1.Protocol for Context Management 2 Including approval mechanisms where human input is required.

		3.Coexist with NM proto in case not all devices support MCP
Management Tools	Use internal tools and LLMs within the controller for managing context and decision making	Need to ensure right tools and background info in the network device
Task Management	Works with pre-structured goal driven tasks. Tasks are usually designed and pre-defined by client	Same Rule Apply
Stateful Management	Yes, Agents can retain context from previous interaction, enabling continuity in long term task or conversation	Yes Same Rule Apply

6.1. MCP Hosted Within the Network Controller

* Pro

  - Resource utilization efficiency: Controllers usually have stronger computing and storage resources, which can better support the operation of MCP Server and will not have a significant impact on the performance of the network equipment itself.

  - Security:

    o Security mechanisms can be implemented centrally on the controller, and the overall security can be improved through unified authentication, authorization and audit mechanisms.

    o Reduces the risk of equipment being exposed to the network and reduces the possibility of being attacked.

  - Protocol adaptability:

    o Communicating with devices through the NETCONF protocol can better be compatible with existing devices and protocols, reducing the need for equipment modification.

    o NETCONF protocol has wide support and mature tool chains in the industry, which is easy to develop and maintain.

* Con

  - Latency and real-time performance:

    o Since management instructions need to be forwarded through the controller, latency may increase and real-time performance may be affected.

    o For some scenarios with extremely high real-time requirements, it may not meet the requirements.

  - Protocol conversion complexity:

    o The MCP protocol needs to be converted to the NETCONF protocol, which increases the complexity and development cost of protocol conversion.

- It is necessary to deal with compatibility and consistency issues between different protocols.

6.2. MCP Server Hosted within the Network Device

* Pro

  - The protocol stack simplification:

    - If you deploy the MCP Server directly on the network device, you can skip the NETCONF protocol layer and manage the device directly through MCP. This reduces the complexity of protocol conversion and simplifies the overall architecture.

    - It reduces the development and maintenance costs caused by protocol adaptation, especially when the device manufacturer supports the MCP protocol.

  - Real-time performance and response speed:

    - The MCP Server is directly deployed on the device, which reduces the transmission latency in the middle and can respond to management instructions faster, which is suitable for scenarios with high real-time requirements.

* Con

  - Device Resource Consumption:

    - Network devices usually have limited resources (CPU, memory, etc.). Deploying MCP Server may occupy a large amount of resources, affecting the normal operation of the device.

    - It is necessary to optimize and expand the hardware and software resources of the device, which increases the complexity of the device.

  - Security and Management Complexity:

    - Each device needs to manage the security of the MCP server separately (such as authentication, authorization, audit, etc.), which increases the complexity of management.

    - Each device needs to independently deploy and maintain the MCP Server, which increases the operation and maintenance cost.

  - Incompatible with Legacy devices:

    - Legacy devices do not have the ability to support MCP servers and still need NETCONF to implement network configuration. This makes it impossible for the network to form a unified control mechanism.

7. IANA Considerations

This document has no IANA actions.

8. Security Considerations

The MCP protocol needs to consider scenarios where either the client or server encounters issues, such as crashes. If one or both parties go offline during communication, the entire process may remain stuck waiting for messages, potentially leading to an infinite loop. Furthermore, certain tool operations may be interrupted, and some

irreversible network management operations could be affected.

Due to network latency, some operations might not return in time, yet from the user's perspective, these operations may appear either unexecuted or failed. If the user then initiates another tool request to the server, problems may occur.

For complex network management workflows, while LLM's tool invocation process may generally function correctly, issues can arise in the details. Users must verify each LLM operation to prevent unintended hazardous actions.

9. References

9.1. Normative References

[RFC2576] Frye, R., Levi, D., Routhier, S., and B. Wijnen, "Coexistence between Version 1, Version 2, and Version 3 of the Internet-standard Network Management Framework", RFC 2576, DOI 10.17487/RFC2576, March 2000, <https://www.rfc-editor.org/rfc/rfc2576>.

[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>.

[RFC8040] Bierman, A., Bjorklund, M., and K. Watsen, "RESTCONF Protocol", RFC 8040, DOI 10.17487/RFC8040, January 2017, <https://www.rfc-editor.org/rfc/rfc8040>.

9.2. Informative References

[I-D.kumar-rtgwg-grpc-protocol] Kumar, A., Kolhe, J., Ghemawat, S., and L. Ryan, "gRPC Protocol", Work in Progress, Internet-Draft, draft-kumar-rtgwg-grpc-protocol-00, 8 July 2016, <https://datatracker.ietf.org/doc/html/draft-kumar-rtgwg-grpc-protocol-00>.

[I-D.openconfig-rtgwg-gnmi-spec] Shakir, R., Shaikh, A., Borman, P., Hines, M., Lebsack, C., and C. Morrow, "gRPC Network Management Interface (gNMI)", Work in Progress, Internet-Draft, draft-openconfig-rtgwg-gnmi-spec-01, 5 March 2018, <https://datatracker.ietf.org/doc/html/draft-openconfig-rtgwg-gnmi-spec-01>.

Contributors

Guanming Zeng
Huawei
China
Email: zengguanming@huawei.com

Authors' Addresses

Yuanyuan Yang
Huawei
China
Email: yangyuanyuan55@huawei.com

Qin Wu
Huawei

China
Email: bill.wu@huawei.com

Diego Lopez
Telefonica
Spain
Email: diego.r.lopez@telefonica.com

Nathalie Romo Moreno
Deutsche Telekom
Germany
Email: nathalie.romo-moreno@telekom.de

Lionel Tailhardat
Orange Research
France
Email: lionel.tailhardat@orange.com