Network Management Working Group
Internet-Draft
Intended status: Informational
Expires: 22 April 2026

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

19 October 2025

Applicability of MCP for the Network Management
draft-yang-nmrg-mcp-nm-01

Abstract

The application of MCP in the network management field is meant to refactor network management operation and network capabilities as tools and provide more agile and extensible architecture to expose these AI integration capabilities. This document discusses the applicability of MCP to the network management plane in the IP network that utilizes IETF technologies.It explores MCP for network exposure, multiple MCP server discovery generic workflow and deployment scenarios.

Status of This Memo

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

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on 22 April 2026.

Table of Contents

1. Introduction

The Model Context Protocol (MCP) decouples LLMs from tools and provides a standardized way for LLMs to access and utilize information from different data sources and tools, making it easier to build AI applications that can interact with external LLM models and software tools and enable workflows automation.

MCP has seen rapid adoption across both startups and enterprises since it announced in November 2024. Key use cases include AI coding assistants in IDEs, data analysis tools that can query databases, and productivity tools that can interact with services like Slack or Google Drive.

The application of MCP for the network management is meant to refactor network management operation and network capabilities as tools and provide more agile and extensible architecture to expose or consume these AI integration capabilities.

With integration of MCP into the network management system, it allow you develop various rich AI driven network applications, realize intent based networks management, automate workflows in the multi-vendor heterogeneous network platform. By establishing standard interfaces for tool encapsulation, intent translation, and closed-loop execution within the network management system, MCP enables the network management system 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 MCP for network exposure, multiple MCP server discovery generic workflow and deployment scenarios.

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

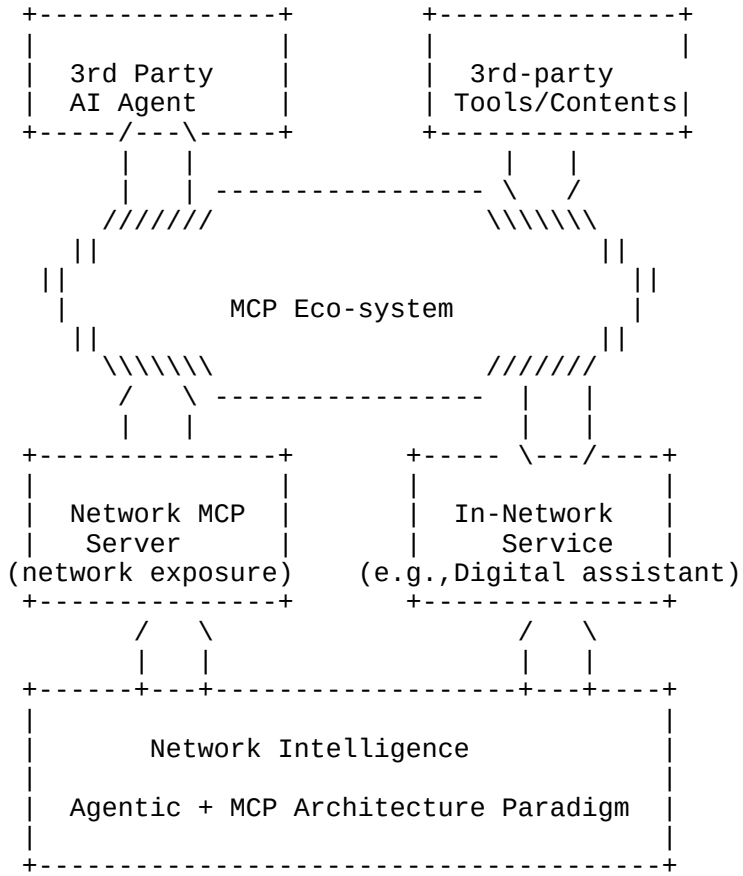
* *LLM*: Large Language Model

* *NETCONF*: Network Configuration Protocol [RFC6241]

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

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

3. The values of coupling MCP with the network management



There are 3 values for MCP coupling with the network management - Network MCP Server support * Exposing network capabilities as MCP Servers for 3rd-party AI Agents & applications

* In Network Service
  - Empowering network services by leveraging MCP ecosystem tools & contents
  - Network service as a unified interface to various and rich capabilities for network users
* Network Management Intelligence
  - Making network architecture really agile and friendly for innovation
  - Decoupling network management functions into network agents and network tools

4. The high level Challenges in adopting MCP in NM

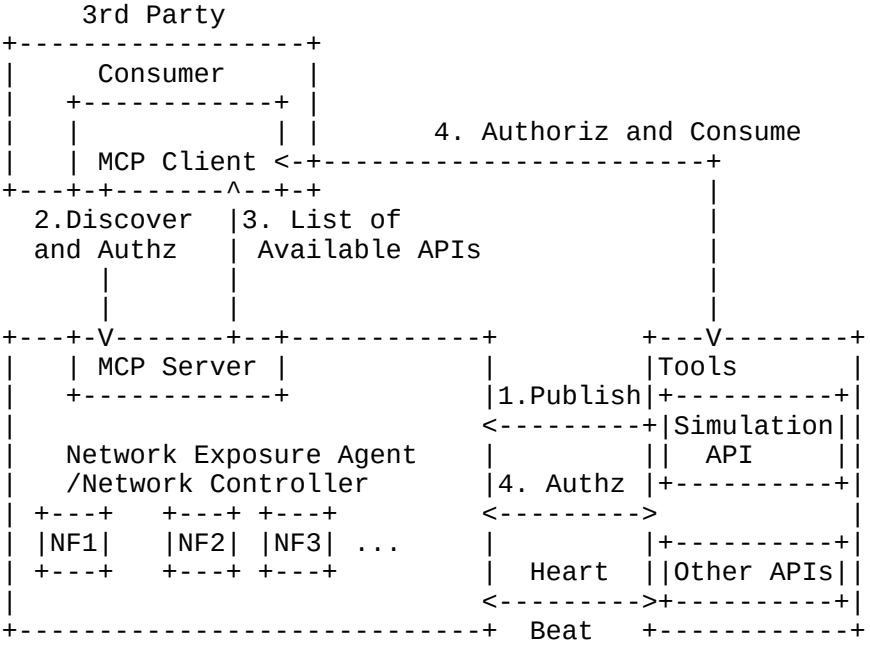
* Protocol Design
  - Error Handling o Although MCP provides basic error codes, MCP does not yet enforce a entire error-handling mechanism, and its scope is currently limited to discovery and invocation, omitting crucial aspects like tool governance, versioning, or lifecycle management.
  - Stateful o The protocol’s reliance on stateful Server-Sent Events (SSE) can create significant complexities when integrating with inherently stateless REST APIs, requiring developers to manage state externally. This can be particularly challenging for remote MCP servers due to network latency and instability, complicating load balancing and horizontal scaling efforts.
  - Context Handling o There are also concerns that multiple active MCP connections could consume significant tokens in the LLM’s context window. This can directly impact an LLM’s performance, slowing down responses and potentially hindering its ability to maintain focus and reason effectively over extended or complex interactions.
* Security Consideration
  - Malicious Actors The protocol’s ability to grant LLMs access to external systems introduces potential vulnerabilities that require careful consideration o Prompt injection, where malicious instructions embedded in user inputs or tool descriptions could lead to unintended actions by the LLM o Tool poisoning, where attackers modify tool definitions, or rug pulls (similar to tool poisoning but occurs post-installation) o Tool shadowing, where a malicious server creates a tool with the same name as a legitimate tool from another server to intercept calls
  - Security enforcement o MCP itself lacks inherent security enforcement mechanisms, relying heavily on external implementations for authentication and authorization, which were not initially well-defined within the protocol.
  - Identity Management o Determining clear identity management - whether requests originate from the end user, the AI agent, or

a shared system account - remains an area needing clearer definition.

5. MCP for Network Exposure

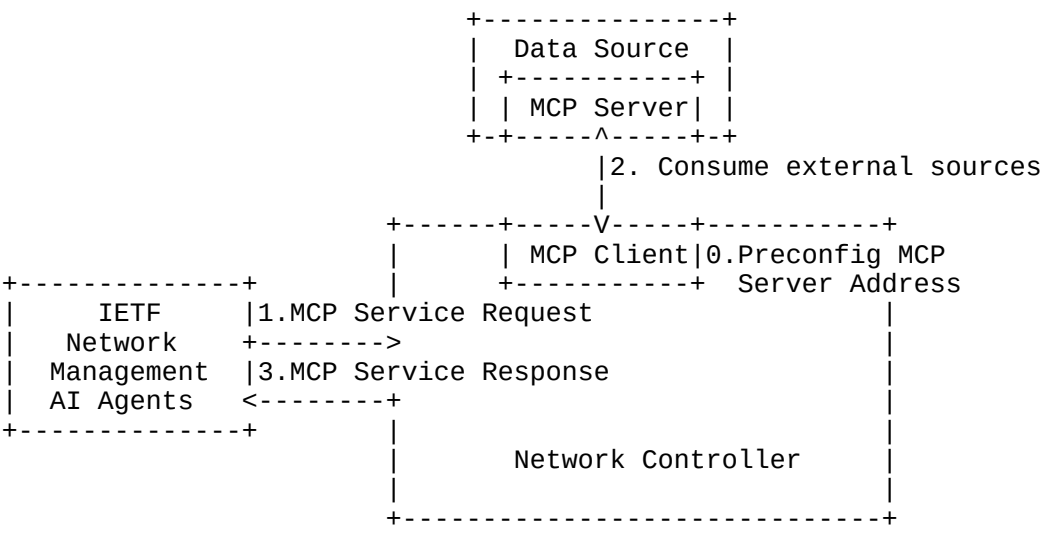
Network exposure is the process of making network capabilities, such as data and connectivity services, available to external users, applications, and developers through secure APIs. It allows for more agility and the creation of programmable networks. The MCP can be used to expose network capabilities to AI applications or consume external sources for LLMs.

5.1. The Third Party as IETF network Exposure Consumer



Step 1: External tools or data source publish a set of APIs to MCP server in the Network Controller. Step 2: MCP client send specific tools request to discover tools and MCP Server provide authorization to the MCP client. Step 3: After sucessful authorization, MCP server return API list corresponding to tools request sent by the MCP client. Step 4: MCP Client invokes tools with authorization.

5.2. The Network Controller Consumes external sources



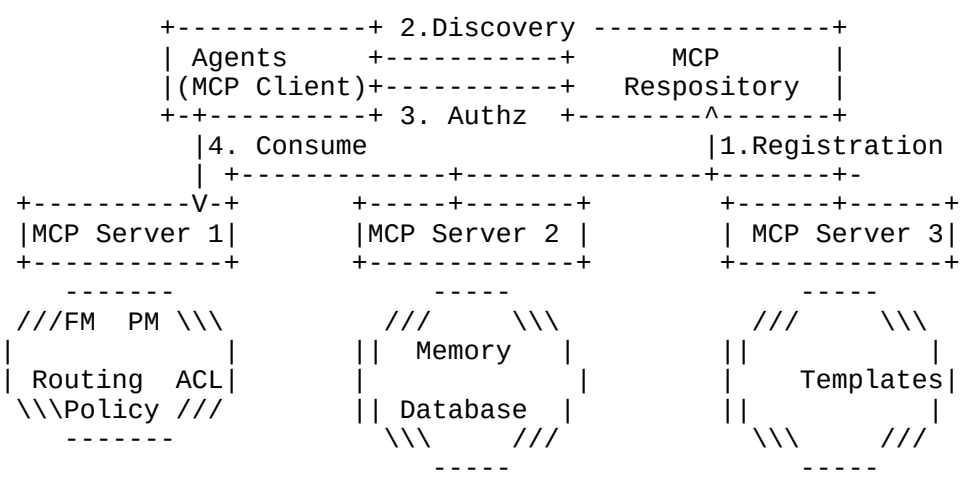
Step 0: MCP Client is preconfigured with the MCP Server address.

Step 1: IETF Network Management AI Agent sends a MCP Service Request to the MCP client within the Network Controller. Step 2: The MCP client discover tools provided by the external MCP server. Step 3: The MCP client provide available tools list to the IETF Network Management AI Agent.

6. MCP Server Discovery

The MCP Server Discovery involves clients querying servers to find available tools, resources, and functions. In case of MCP servers are distributed in different locations, MCP Respository can be established to keep track of the location of each MCP servers.

6.1. MCP core function

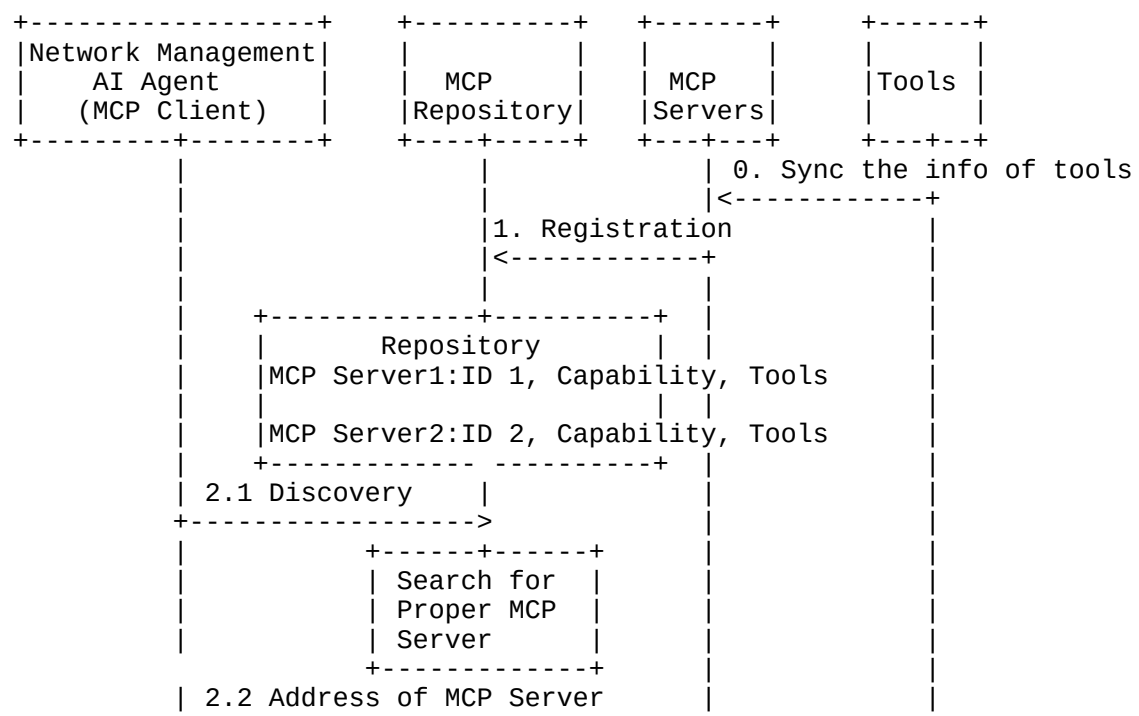


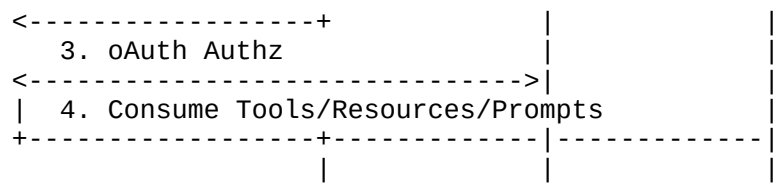
寰「 MCP Repository: * Maintain the description of MCP servers * Support MCP server discovery for Client.

寰「 MCP client: consuming the services provided by MCP servers

寰「 MCP servers: including authentication/session/policy tools, the memory/prompts used for Agents

6.2. Procedure for Multi-MCP servers Discovery





Step 0: The MCP Server syncs up on the info of tools, upon tools is added or removed, tools changes will be automatically syned up with the MCP server. Step 1: Each new MCP server will register to the centralized MCP registry. Step 2.1: MCP Client send the MCP service request to the MCP registry for specific capability. Step 2.2: The MCP registry return specific MCP server to the MCP client. Step 3: The MCP Client request authorization from the MCP server. Step 4: The MCP Client invoke specific tools with authorization.

7. Deployment Consideration in adopting MCP in the Network Management

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

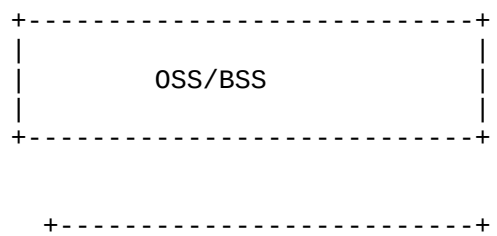
  - Fault remediation workflows

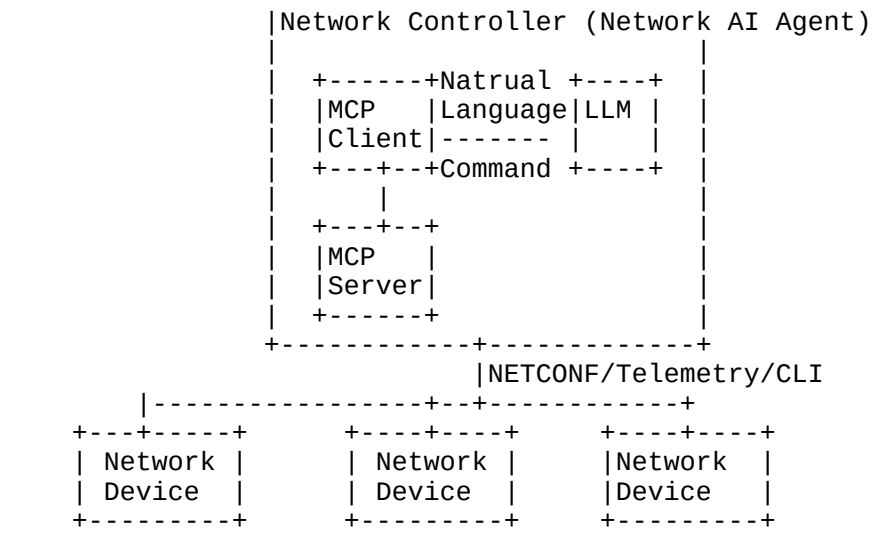
  - Other common management operations to reduce operator workload

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

7.1. MCP Communication using Inter Processing Communication

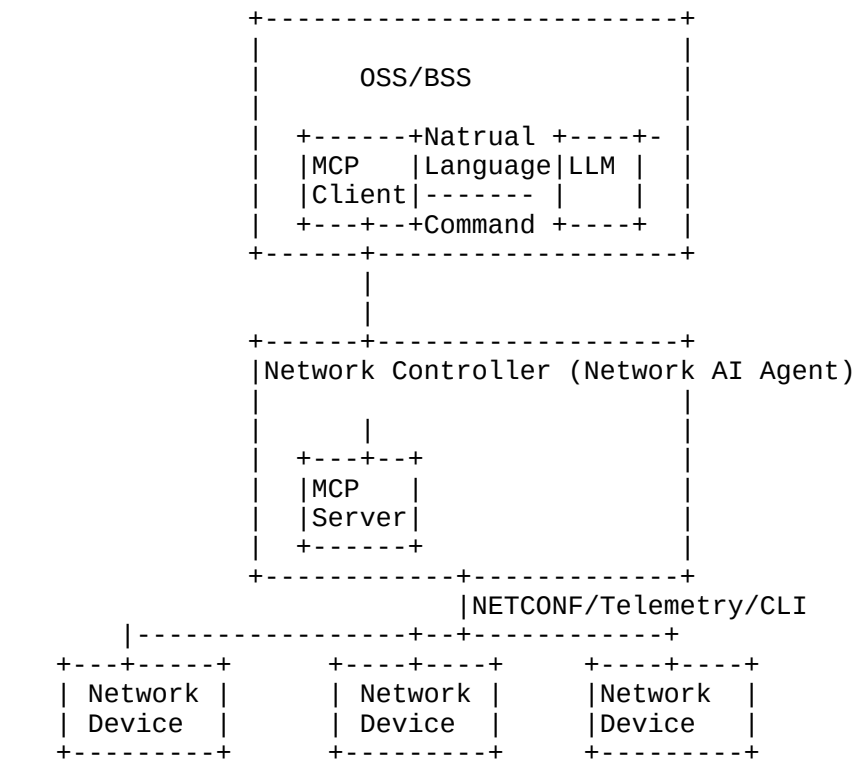
In this network scenario, both the MCP client and the MCP server are deployed within the same network controller. The MCP client communicate with the MCP server using inter processing communication. In the meanwhile, the LLM model as pre-trained model should also be deployed in the same Network controller.





7.2. The OSS and the Network Controller Communication using MCP

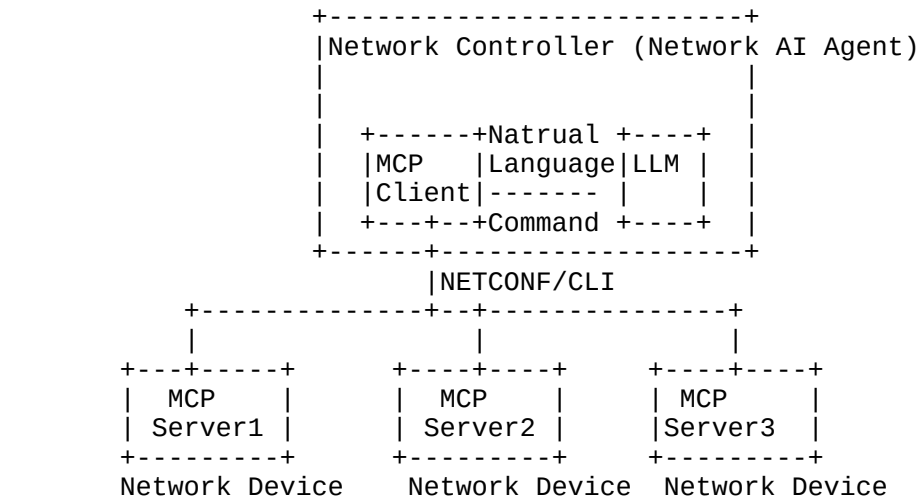
In this network scenario, the MCP client is deployed in the OSS/BSS while the MCP server is deployed in the Network Controller, the MCP client and the MCP server communicated using SSE. In the meanwhile, the LLM model can be pre-trained LLM model collocated with the MCP client or the LLM model in the Cloud.



7.3. The Network Controller and the Network Device Communication using MCP

In this network scenario, The MCP client is deployed in the network controller while the MCP server is deployed in the network devices. The MCP client communicates with the MCP server using the MCP protocol. LLM model is pre-trained model and deployed in the same Network Controller as the MCP client.

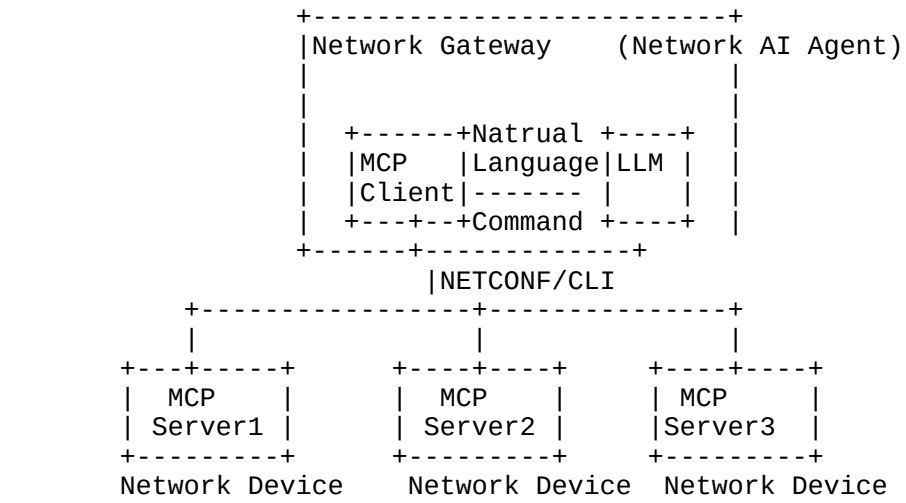
Network devices usually have limited resources (CPU, memory, etc.). Deploying MCP Server may occupy a large amount of resources, affecting the normal operation of the device.



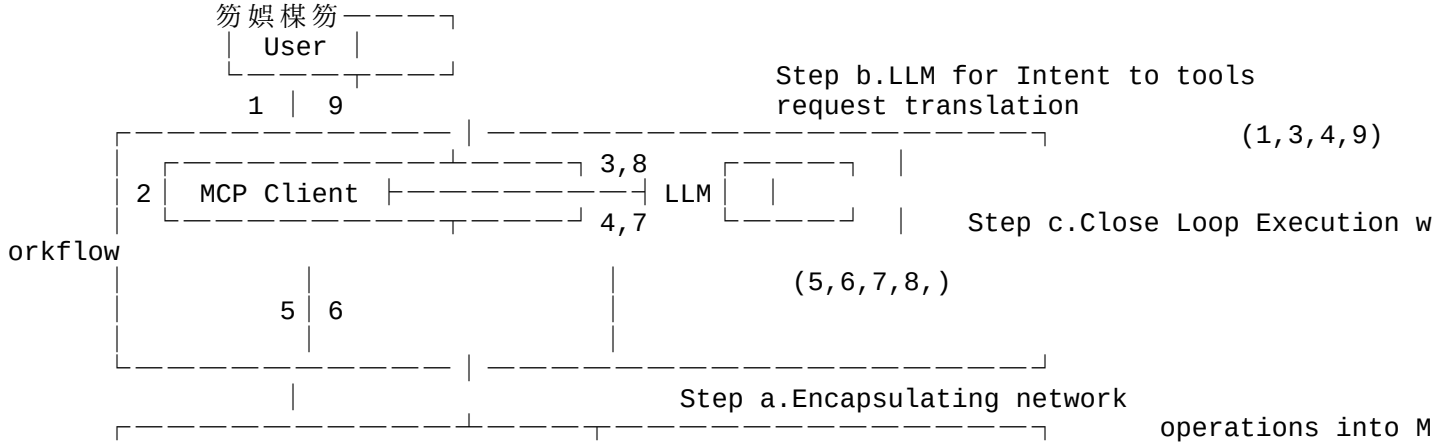
7.4. The Network Gateway and the Network Device Communication using MCP

In this network scenario, The MCP client is deployed in the network gateway device while the MCP server is deployed in each network devices. The MCP client communicates with the MCP server using the MCP protocol. LLM model is pre-trained model and deployed in the same Network gateway as the MCP client.

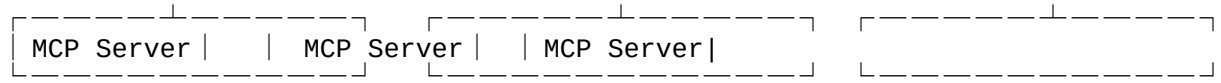
Network devices usually have limited resources (CPU, memory, etc.). Deploying MCP Server may occupy a large amount of resources, affecting the normal operation of the device.



8. MCP architecture Design for Network Management



CP tools



8.1. 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.

8.2. 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.

8.3. 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:

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

  o 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.

9. Interworking with the Network Management protocol and YANG data models

MCP can be seen as AI protocol and used to invoke AI integrated capabilities. MCP is not in the position to replace the network management and YANG data model. Instead, it can be integrated together, e.g., * Integrated MCP with CLI, allow the MCP client invoke CLI capabilities; * Integrated MCP with YANG, allow the MCP client invoke YANG interface related capabilities;

10. MCP Usage Examples

10.1. Device Configuration using MCP+CLI

In this example, MCP server is implemented by Server and expose all CLI interfaces and documentation as tools to the MCP Client. The Nework controller implement the client to interact with MCP server in the Network device.

The MCP 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 responds to the call instruction, converts it into the below CLIs of different vendors, and then the devices execute the CLIs. The results are returned to the MCP client in Json as below and are forwarded 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.

```
# Convert to CLI commands of different vendors
"commands": [
    "system-view",
    "ospf {{process_id}}",
    "area {{area_id}}",
    "network {{network_address}} {{wildcard_mask}}"
  ]

"commands": [
    "configure terminal",
    "router ospf {{process_id}}",
    "network {{network_address}} {{wildcard_mask}} area {{area_id}}",
    "end",
    "write memory"
  ]

#Feedbacks received by the MCP client of different vendors
{
    "status": "success",
    "message": "OSPF configuration applied successfully on device 192.168.10.1",
    "commands_executed": [
        "system-view",
        "ospf 100",
        "area 0.0.0.0",
        "network 192.168.10.0 0.0.0.255"
    ]
}

{
    "status": "success",
    "message": "OSPF configuration applied successfully on device 192.168.10.1",
    "commands_executed": [
        "configure terminal",
        "router ospf 100",
        "network 192.168.10.0 0.0.0.255 area 0",
        "end",
        "write memory"
    ]
}

# Natural lanauge summary of success or failure:
{
  "192.168.10.1": "Configure Successfully, take 2.3 seconds",
  "192.168.10.2": "Error: no response from the device",
```

```
}
```

11. IANA Considerations

This document has no IANA actions.

12. 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.

13. 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>.

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

Guanming Zeng
Huawei
China
Email: zengguanming@huawei.com