Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: 6 May 2026

G. Zeng
J. Mao
B. Liu
N. Geng
X. Shang
Q. Gao
Z. Li
Huawei

2 November 2025

When NETCONF Is Not Enough: Applicability of MCP and A2A for Advanced Network Management Scenarios

draft-zeng-opsawg-applicability-mcp-a2a-00

Abstract

NETCONF provides robust configuration transactions and YANG-based data models, but falls short in scenarios requiring AI-driven semantic translation, long-lived cross-domain orchestration, multi-agent consensus, rapid DevOps iteration, or delivery of large non-configuration artifacts. This document systematically analyzes the functional gaps and presents Model Context Protocol (MCP) and Agent-to-Agent (A2A) as complementary solutions. Implementation guidance and coexistence models are also provided.

Table of Contents

1. Introduction

NETCONF [RFC6241] remains the gold standard for network configuration transactions. However, five emerging scenarios expose its fundamental limitations:

* (1) AI natural-language intent
* (2) Long-flow cross-controller orchestration
* (3) multi-agent consensus
* (4) weekly DevOps release cycles
* (5) multi-modal artifact delivery

This document identifies objective gaps and specifies when and how MCP [I-D.yang-nmrg-mcp-nm] and A2A [I-D.google-agent2agent] should be engaged.

2. Gap Analysis Summary

This section enumerates the fundamental gaps between NETCONF and the advanced management scenarios introduced in Section 1. For each gap, the table below identifies:

* the missing capability,

* its root cause in NETCONF design, and

* the protocol (MCP or A2A) that natively provides it.

Gap	Root Cause in NETCONF	MCP/A2A Solution
AI Semantic Layer	XML-centric, no function registry	MCP /tools/list + JSON-Schema
Long-Flow Orchestration	No Task life-cycle or human-in-the-loop	A2A Task state machine
Multi-Agent Consensus	Client-server only; no peer negotiation	A2A AgentCard + Message
Weekly DevOps Iteration	YANG revision 6-9 months; firmware lock	MCP Tool hot-register
Large Artifact Delivery	64 kB chunk; no MIME/hash/URL	MCP/A2A Artifact (cloud URL)

Table 1

The gaps are not implementation defects but architectural invariants of RFC 6241. They become blocking only in the five advanced scenarios identified. Outside these scenarios, NETCONF continues to provide the most robust configuration transactions and should remain the south-bound protocol of choice.

3. When MCP Must Be Used

3.1. AI Natural-Language Intent

Operators increasingly expect to issue instructions in natural language: “Raise MTU to 9000 for all Beijing core switches” or “Block source 1.2.3.4 for 30 minutes”. NETCONF requires an edit-config XML blob with exact leaf names and namespaces; even experienced engineers make syntax mistakes under time pressure.

The root cause is architectural:

* XML is attribute-heavy and case-sensitive; forgotten namespaces or mismatched quotes silently fail.

* There is no machine-discoverable “function catalogue” — an LLM must rely on static prompt examples which drift as models evolve.

* Multi-vendor differences (OpenConfig vs. proprietary YANG) force the LLM to choose branches inside the XML, exploding prompt size.

MCP solves these issues with three primitives:

* /tools/list — JSON array of callable functions, each carrying human-readable description and JSON-Schema input.

* JSON-Schema — strong-typed, no namespaces, direct mapping to primitive types (string, integer, enum, array).

* JSON-RPC 2.0 — single-line request/response, easily parsed by LLM and by controller gateways.

Example MCP Tool Descriptor (simplified):

```
{
  "name": "batch_set_mtu",
  "description": "Set interface MTU on multiple devices",
  "inputSchema": {
    "type": "object",
    "properties": {
      "device_group": {"type": "string"},
      "mtu": {"type": "integer", "minimum": 576, "maximum": 9216}
    },
    "required": ["device_group", "mtu"]
  }
}
```

Figure 1

The LLM now produces:

```
{
  "jsonrpc": "2.0",
  "method": "batch_set_mtu",
  "params": {"device_group": "beijing-core", "mtu": 9000},
  "id": 1
}
```

Figure 2

3.2. Rapid Model Iteration (DevOps Week-Release)

Cloud-era value-added services must be deployed within days, not months. NETCONF's revision cycle (IETF draft → RFC: 6-9 months) and firmware upgrade windows (1 per year) are incompatible with weekly release trains. The blocking points are:

* YANG module must be burned into firmware before the first config leaf is usable;
* Controller regression suite recompiles the entire YANG tree even for a single new leaf;
* Backward-compatibility review (must not break old devices) stretches internal QA to weeks.

MCP breaks the deadlock by treating "intent" as a hot-swappable Tool rather than a permanent YANG node:

* Private YANG is compiled to JSON-Schema in the controller (milliseconds);
* Tool registers via /tools/register and is immediately callable;
* Gray-list rollout (10 % → 30 % → 100 %) and instant rollback (re-register previous Tool) are done without touching device flash.

Example: Cloud-Shield DDoS Cleaning Service

```
Private YANG (120 lines)
  +--rw start-cleanse
  | +--rw target-ip      inet:ipv4-address
  | +--rw bandwidth-Mbps uint32
  | +--rw duration-min   uint16
```

Figure 3

Compiled JSON-Schema and registered in 30 s:

```
{
  "name": "start_cleanse",
  "version": "1.0.0",
  "description": "Start DDoS cleanse for target IP",
  "inputSchema": {
    "type": "object",
    "properties": {
      "target_ip": {"type": "string", "format": "ipv4"},
      "bandwidth_Mbps": {"type": "integer", "minimum": 100, "maximum": 100000},
      "duration_min": {"type": "integer", "minimum": 5, "maximum": 1440}
    },
    "required": ["target_ip", "bandwidth_Mbps"]
  }
}
```

Figure 4

Thus MCP is mandatory for any management surface that must support weekly or daily release cycles without waiting for firmware or standards body timelines.

4. When A2A Must Be Used

4.1. Cross-Controller Long-Flow Orchestration

Maintenance windows for core-network upgrades often exceed 30 minutes and span multiple vendor domains. NETCONF provides atomic configuration on a single controller, but lacks:

* a cross-vendor task life-cycle,

* human-in-the-loop approval gates, and

* delivery of large artifacts (firmware, images, diff reports).

A2A fills these gaps with three primitives:

* Task — state machine (pending → working → completed/failed/ cancelled) persisting across agent restarts;

* Artifact — hash-signed object store (2 GB, resumable upload);

* Message — multi-round negotiation (JSON or natural language).

State	Meaning	NETCONF Equivalent
pending	Waiting for resources or approval	None (RPC is fire-and-forget)
awaiting-human-approval	Human must click approve/ cancel	None
working	Agents executing sub-tasks	edit-config (local only)
completed	All agents report success	commit
failed	Any agent reports failure	rollback-on-error
cancelled	Operator or policy cancelled	discard-changes

Table 2: A2A Task States vs. NETCONF Operations

Real-World Example: Five-City Core MTU Migration

Controllers: Huawei NCE (3 cities) + Cisco NSO (2 cities)
Devices: 312 PE routers
Window: 120 min (02:00-04:00)
Artifact: 2 GB firmware image + 40 MB diff-report

Figure 5

Step-wise A2A Flow (time-stamps):

* T+0 min: Orchestrator creates Task T100, goal="Raise MTU to 9000 on core links".

* T+5 min: Each controller Agent posts Artifact pre-check.csv (link health KPI).

* T+10 min: Orchestrator Artifact hash-verified; human approval card sent to WeChat.

* T+15 min: Engineer clicks "approve"; Task state → working.

* T+20-90 min: Controllers download 2 GB image via Artifact URL; local NETCONF edit-config issued; progress Artifacts streamed every 5 min.

* T+95 min: Last Artifact post-upgrade-verification.csv uploaded.

* T+100 min: All agents report success; Task → completed. Total human intervention: 1 click.

A2A is mandatory for any multi-vendor, multi-hour workflow that demands task persistence, human gates, and multi-gigabyte artifact delivery— scenarios where NETCONF’s single-controller, single-RPC paradigm is insufficient.

4.2. Multi-Agent Consensus (Agent-to-Agent)

Fault recovery, security mitigation and resource optimisation often require multiple autonomous agents (monitoring, security, controller, human) to reach a common decision. NETCONF’s strict client-server model provides no peer-to-peer capability advertisement, multi-round negotiation or voting primitives.

A2A introduces three building blocks:

* AgentCard — JSON-LD advertisement of skills and endpoint;

* Message — multi-round negotiation (JSON or natural language);

* Consensus Engine — policy-based scoring, voting, human-in-the-loop.

Field	Description	Example Value
id	globally unique agent identifier	monitor-sh-01
skills	array of skill objects (name, description)	{name: "threat_analyze", desc: "Return 0-10 threat score"}
endpoint	HTTPS URL for A2A messages	https://mon-sh.example:9443/a2a
authentication	mTLS + OIDC	{"type": "mTLS", "sha256": "8f66..."}

Table 3: AgentCard Mandatory Fields

Consensus Flow Example: DDoS Port Shutdown

Agents: Monitor, Security, Controller, Human
Decision: shutdown port 10/1 ?
Scoring: threat_level× 0.6 + impact× 0.4
Threshold: 5.0 → shutdown

Figure 6

Message Sequence (time-stamps):

* T+0 s: Monitor Agent posts threat_score=9.0 via Message.

* T+5 s: Security Agent confirms attack signature; score unchanged.

* T+10 s: Controller Agent posts impact=300 VPN down; computed score = 9× 0.6 + 3× 0.4 = 6.6 (> 5.0).

* T+12 s: Task state → awaiting-human-approval; WeChat card sent.

* T+135 s: Human clicks "approve".

* T+140 s: Controller Agent calls NETCONF shutdown; Artifact post-action.log uploaded.

* T+180 s: All agents report success; Task → completed.

Therefore A2A is mandatory whenever multiple autonomous agents must discover, negotiate, vote and reach a binding decision — scenarios that NETCONF's unidirectional client-server paradigm cannot emulate.

5. Coexistence Model

This section describes how MCP and A2A can be deployed _without_ forcing a redesign of the existing NETCONF ecosystem. The architecture keeps NETCONF as the configuration authority and allows _either_ controller-hosted _or_ device-hosted MCP servers — the latter avoids a central gateway bottleneck while preserving operator investment in controllers.

5.1. Design Choices at a Glance

TBD

5.2. Common Layering

TBD

5.3. Controller-Gateway Model

TBD

5.4. Device-Embedded Model

TBD

5.5. Migration Roadmap

TBD

6. Security Considerations

MCP and A2A introduce OAuth2/JWT and long-lived Tasks.

7. Normative References

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

[RFC9200] Tschofenig, H., "Guidelines for Using the Transport Layer Security (TLS) Protocol in Devices", RFC 9200, DOI 10.17487/RFC9200, March 2022, <https://www.rfc-editor.org/rfc/rfc9200>.

8. Informative References

[I-D.yang-nmrg-mcp-nm]
Yang, Y., "Applicability of MCP for the Network Management", Work in Progress, Internet-Draft, draft-yang-nmrg-mcp-nm-00, July 2025, <https://datatracker.ietf.org/doc/html/draft-yang-nmrg-mcp-nm-00>.

[I-D.google-agent2agent]
Google., "Agent-to-Agent (A2A) Protocol", Work in Progress, Internet-Draft, draft-google-agent2agent-00, May 2025, <https://datatracker.ietf.org/doc/html/draft-google-agent2agent-00>.

Authors' Addresses

Guanming Zeng
Huawei
Email: zengguanming@huawei.com

Jianwei Mao
Huawei
Email: maojianwei@huawei.com

Bing Liu
Huawei
Email: leo.liubing@huawei.com

Nan Geng
Huawei
Email: gengnan@huawei.com

Xiaotong Shang
Huawei
Email: shangxiaotong@huawei.com

Qiangzhou Gao
Huawei
Email: gaoqiangzhou@huawei.com

Zhenbin Li
Huawei
Email: robinli314@163.com