Network Working Group
Internet-Draft
Intended status: Informational
Expires: 15 August 2026

J. Yan
X. Zhang
ZTE

11 February 2026

Applicability of A2A Protocol for Network Management Agents
draft-yan-a2a-device-agent-applicability-01

Abstract

The evolution of network management towards autonomic operation requires the deployment of AI agents at various hierarchical layers, including directly on network elements. This transformation shifts network devices from passively managed resources to autonomous entities capable of local decision-making and collaborative problem-solving.

This document discusses the applicability of the Agent-to-Agent (A2A) Protocol to the network management plane, specifically for communication between Controller Agents (CAs) and Device Agents (DAs). This indicates that the inherent characteristics of Device Agents necessitate the adoption of the agent-to-agent communication paradigm. The document further explores generic workflows, deployment scenarios, and the relationship of A2A with existing network management protocols like NETCONF, RESTCONF, gNMI,and the Model Context Protocol (MCP).

Table of Contents

1. Introduction

The management of large-scale networks is undergoing a significant transition, moving from centralized, imperative control to distributed, intent-based autonomic operation. A key enabler of this shift is the deployment of AI Agents on network elements. These Device Agents (DAs) transform network devices from passive, managed resources into autonomous entities that can perceive their local environment, make decisions, and act to achieve goals delegated by a higher-level Controller Agent (CA).

This transformation necessitates a re-evaluation of the communication protocols utilized. Traditional network management protocols are typically based on a client-server, request-response model where a controller directly manipulates data on a device. While this model remains effective for certain interactions, it may not fully support the collaborative needs of autonomous agents.

This document examines the applicability of the Agent-to-Agent (A2A) Protocol[A2A-SPEC] for communication between Controller Agents and Device Agents. The objectives of this document are to:

* Explain why CA-DA interactions often align with an "agent-to-agent" model, rather than exclusively a "controller-to-device" or "agent-to-tool" paradigm.

* Explore how the features of the A2A protocol, such as task management, stateful sessions, and peer-to-peer collaboration, are well-suited to the needs of autonomic network management.

* Describe generic workflows and deployment scenarios for A2A in the network management plane.

* Clarify the relationship between A2A and other relevant protocols, including NETCONF [RFC6241], gNMI, and the Model Context Protocol (MCP).

This document aims to provide a conceptual framework for applying the base A2A protocol to the network management domain.

2. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

This document uses the following terms:

Agent-to-Agent (A2A) Protocol: An open standard protocol for enabling secure communication and collaboration between autonomous AI Agents, as specified in [A2A-SPEC].

Controller Agent (CA): An AI Agent operating in a network controller or domain management system.

Device Agent (DA): An AI Agent deployed on or embedded within a network element, possessing a degree of autonomy.

Agent-to-Tool (A2T) Paradigm: A communication model where an agent invokes a tool, which is a passive entity that performs a specific, well-defined function and returns a result.

Agent-to-Agent (A2A) Paradigm: A communication model where autonomous agents collaborate as peers to achieve a shared or delegated goal.

3. The Shift from Managed Device to AI Agent

3.1. Three-Layer Agent Architecture

The deployment of AI Agents in network management typically follows a three-layer hierarchical model. This architecture is illustrated in Figure 1.

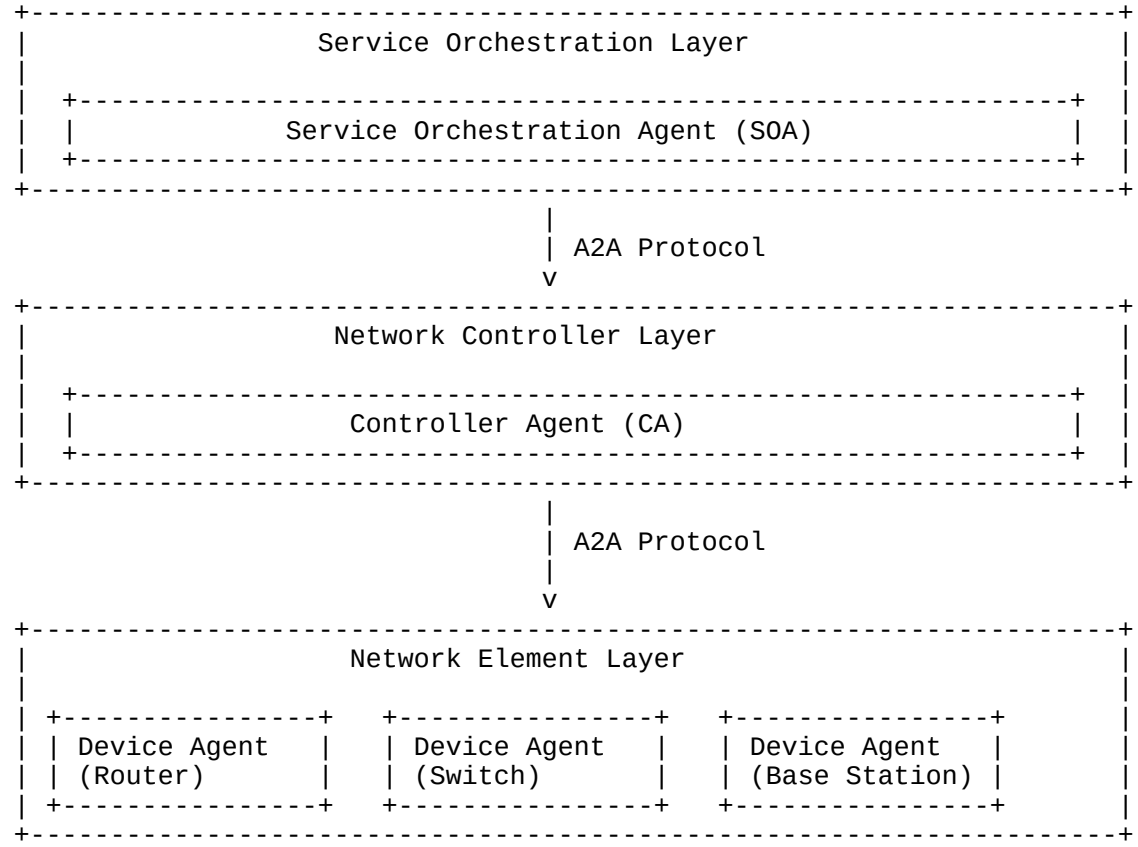

Figure 1: Three-Layer Agent Architecture

3.2. Limitations of Traditional Management Paradigms

Traditional network management protocols like SNMP, NETCONF, and gNMI are built on a paradigm where the network element is a passive repository of state and configuration data. Intelligence resides solely in the controller, which exerts imperative control through explicit operations:

* Imperative Control: The controller specify the exact sequence of commands to achieve a desired state.

* Request-Response Model: The device only acts in response to a direct request from the controller.

* Limited Abstraction: The controller requires knowledge of device-specific details.

* High Latency for Local Events: Responding to a local event (e.g., a link failure) requires a round-trip to the controller, which may be too slow for time-critical applications.

In dynamic, large-scale networks, these limitations hinder scalability, resilience, and advanced automation.

3.3. Characteristics of a Device Agent

Integrating a DA redefines a network element as an intelligent, autonomous entity with:

* Autonomy: Independent decision-making and action on delegated goals (e.g., maintaining link uptime via rerouting).

* Statefulness: Retention of context, history, and learned behaviors for informed decisions.

* Proactiveness: Initiation of communications, notifications, or remediations based on observations.

* Goal-Oriented Behavior: Planning and reasoning to achieve high-level objectives, beyond mere command execution.

These traits differentiate DA interactions from those with traditional devices.

4. Agent-to-Agent vs. Agent-to-Tool Paradigm

Distinguishing agents from tools is essential for paradigm selection. Both A2A and A2T have roles in network management and can complement each other based on autonomy levels.

4.1. The Agent-to-Tool Model

In an Agent-to-Tool (A2T) model, an agent interacts with a tool. A tool is a passive component that:

* Performs a specific, well-defined function.

* Is typically stateless.

* Does not have its own goals or intentions.

* Responds only when invoked.

MCP exemplifies this, standardizing function calls and responses. It suits simple interactions, including with advanced agents for imperative tasks, or wrapping legacy interfaces like NETCONF.

4.2. The Agent-to-Agent Model

In A2A, agents collaborate as peers. As outlined in Section 3.3, agents are autonomous, stateful, proactive, and goal-driven. Interactions encompass:

* Delegation of tasks and goals.

* Negotiation and collaboration.

* Sharing of context and state.

* Long-lived, multi-turn interactions.

* Asynchronous and proactive notifications.

Protocols designed for this model, such as the A2A Protocol, provide primitives for task lifecycle management, stateful conversations, and peer-to-peer discovery.

4.3. Why Device Agents Require an A2A Paradigm

Given the characteristics of a Device Agent (Section 3.3), a DA is typically an "agent" rather than a "tool." The A2A paradigm leverages these capabilities effectively for collaborative scenarios.

Exclusive A2T use might constrain proactiveness and intent delegation, reducing interactions to basic calls. However, A2T can serve as a building block within an A2A framework—for example, a DA might expose certain functions via A2T while engaging in broader collaboration via A2A. This hybrid approach allows for flexibility, ensuring that simpler operations can use efficient A2T mechanisms without forgoing the benefits of A2A for more complex, intent-driven tasks.

Therefore, while communication between a Controller Agent and a Device Agent is often best served by an agent-to-agent interaction, incorporating A2T elements can enhance efficiency in specific contexts. For examples of how A2A enables agent-to-agent collaboration in network management, see Section 6, particularly Section 6.1 and Section 6.2.

5. A2A Protocol Applicability in Network Management

The A2A Protocol’s core features align well with CA-DA needs.

5.1. Task-Oriented Collaboration

Network management operations are naturally expressed as tasks (e.g., service configuration, fault diagnosis). The A2A protocol’s task management primitives are well-suited to these requirements:

* Task Creation: A CA can delegate a high-level task to a DA (e.g., "optimize energy usage") without specifying the implementation details.

* Task Lifecycle: The DA manages the task through its lifecycle (PENDING, WORKING, COMPLETED/FAILED), providing the CA with clear status visibility.

* Asynchronous Execution: Long-running tasks, such as a firmware upgrade, can be managed asynchronously. The CA can create the task and check on its status later, without maintaining a persistent session for the duration of the task.

* Artifacts: The DA can return structured artifacts upon task completion, such as diagnostic reports or verification of a configuration change.

5.2. Stateful, Long-Running Interactions

Many network management operations are often long-running, involve multiple steps, produce incremental results, or require human intervention. A2A provides mechanisms for managing such asynchronous interactions, ensuring that clients receive updates effectively, whether they remain continuously connected or operate in a more disconnected fashion.

* Stable context_id: Every message carries a globally unique context identifier. Both the Controller Agent and Device Agent can resume the dialogue at any time without re-explaining the original intent. This is what makes the multi-hour energy-optimization dialogue possible.

* Long-lived bidirectional streams: When continuous connectivity is desired, either party may open a WebSocket or gRPC stream (the “Task Stream” defined in the A2A specification). The Device Agent can push telemetry, intermediate results, or questions in real time.

* Push notifications for disconnected scenarios: For the common case where the Controller Agent is not permanently connected, the Device Agent pushes TaskStatusUpdateEvent, TaskArtifactUpdateEvent, or arbitrary messages to a pre-registered webhook. This is the mechanism used in Appendix A and is the recommended pattern for long-running network management tasks.

5.3. Autonomous Operation and Proactive Notification

The peer-to-peer nature of A2A allows a DA to be proactive.

* Sending unsolicited TaskStatusUpdateEvent when an autonomous decision is taken (e.g., sleeping or waking a line card).

* Pushing intelligent, analysed notifications instead of raw alarms.

* Asking for clarification or proposing alternatives when the original intent cannot be fulfilled with current local constraints.

* Delivering final results or artefacts via push notification when the task completes, even if the Controller Agent has been offline for hours.

5.4. Dynamic Capability Discovery

The A2A Agent Card provides a standardized mechanism for a DA to advertise its capabilities.

* Service Discovery: A CA can discover a DA and its endpoint.

* Capability Negotiation: A CA can inspect the DA’s Agent Card to understand what skills it possesses, what data models it understands, and its level of autonomy.

* Extensibility: This allows for vendor-specific extensions and new capabilities to be added over time without changing the protocol itself.

6. Generic Workflows and Deployment Scenarios

6.1. Intent-Based Configuration

This workflow shifts from imperative command execution to intent-based delegation:

1. The CA formulates a high-level intent.

2. The CA creates an A2A task and delegates it to the relevant DA(s), expressing the intent in the message body, potentially using a combination of natural language and structured data.

3. The DA receives the task, parses the intent, and plans the necessary low-level configuration steps (e.g., configure IPsec, QoS, routing).

4. The DA executes the steps, handling any local errors or dependencies.

5. Upon completion, the DA reports the task status and provides verification (e.g., tunnel status, traffic counters) as a task artifact.

6.2. Autonomous Fault Remediation

This workflow demonstrates the value of DA autonomy.

1. The DA performs continuous monitoring of local device health based on its local context awareness.

2. The DA detects a fault (e.g., a primary link fails).

3. Based on pre-delegated policy from the CA, the DA autonomously executes a local remediation action (e.g., reroutes traffic to a backup link) within milliseconds.

4. After taking action, the DA proactively sends an A2A message to the CA, informing it of the event, the action taken, and the current status.

5. The CA receives the notification, updates its network-wide view, and can initiate further actions if needed (e.g., creating a maintenance ticket).

6.3. Deployment Models

Property	On-Box	Off-Box
Description	DA runs on device hardware, in NOS or container/VM.	DA on adjacent hardware, managing via NETCONF/gNMI as proxy.
Advantages	Lowest latency; direct state access	Legacy support; easier scaling
Considera-tions	Resource limits; update challenges	Added latency; proxy complexity

Figure 2: DA Deployment Models

7. Relationship to Other Protocols

7.1. NETCONF, RESTCONF, and gNMI

A2A complements these protocols; it does not replace them. They operate at different levels of abstraction.

* A2A is for task-based, intent-driven collaboration between agents.

* NETCONF/gNMI are for structured, data-oriented configuration and telemetry exchange between a manager and a device.

A Device Agent will often utilize NETCONF or gNMI as internal "tools" to interact with the underlying device’s configuration datastore and hardware.

7.2. Relationship to Other Protocols

A2A and MCP address different needs and are also complementary.

* A2A is for agent-to-agent collaboration, emphasizing autonomy and proactiveness.

* MCP is for agent-to-tool interaction, suitable for passive, function-specific calls.

In the context of network management:

* The CA-DA interaction is primarily A2A for collaborative tasks.

* A CA might use MCP to interact with a legacy device whose NETCONF interface has been wrapped as an MCP "tool".

* A DA might internally use MCP to call its own local tools (e.g., a ping utility, a NETCONF client).

This complementarity enables a gradual transition, where A2T via MCP handles basic operations, while A2A manages higher-level autonomy.

8. Security Considerations

Applying the A2A model to network management introduces important security considerations. Since DAs have a degree of autonomy, the trust relationship between a CA and a DA is critical.

* Authentication: CAs and DAs MUST mutually authenticate each other before any interaction. The use of mTLS is strongly RECOMMENDED.

* Authorization and Policy: The scope of a DA’s autonomy MUST be strictly defined by policies delegated from the CA. A DA MUST NOT be allowed to take actions that violate these policies.

* Auditability: All significant actions taken by a DA MUST be logged and reported to the CA for auditing and accountability.

9. IANA Considerations

This document has no IANA actions.

10. Acknowledgments

TBD.

11. References

11.1. Normative References

[A2A-SPEC] "Agent-to-Agent Protocol Specification", <https://a2a-protocol.org/dev/specification/>.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

11.2. Informative 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/info/rfc6241>.

Appendix A. Example Use Case: Network Energy Efficiency

This appendix provides a concrete, though simplified, example of how the A2A protocol can be applied to a network energy efficiency scenario. The JSON examples are simplified for readability and are compliant with the A2A Protocol Specification v0.3.0.

A.1. Scenario Description

A network operator wants to reduce the power consumption of its edge routers during off-peak hours (e.g., 01:00 to 06:00) without impacting service availability. The high-level business intent is: "Reduce energy consumption by at least 15% during the night while ensuring link capacity is always sufficient for demand."

The Controller Agent (CA) translates this intent into a policy for its Device Agents (DAs). The policy allows DAs to autonomously put underutilized line cards or ports into a low-power (sleep) state and wake them up when traffic demand increases.

A.2. Device Agent Capability Advertisement

The Device Agent on an edge router advertises its energy-saving capabilities via its A2A Agent Card. It exposes a skill named "network.energy.optimize".

```
{
  "protocol_version": "0.3.0",
  "name": "Edge Router 01 DA",
  "description": "Device Agent for edge router energy management",
  "version": "1.2.0",
  "supported_interfaces": [
    {
      "url": "https://da-edge-router-01.example.com/a2a",
      "protocol": "HTTP+json"
    }
  ],
  "capabilities": {
    "streaming": true,
    "push_notifications": true
  },
  "default_input_modes": ["text/plain", "application/json"],
  "default_output_modes": ["text/plain", "application/json"],
  "skills": [
    {
      "id": "skill-energy-opt",
      "name": "network.energy.optimize",
      "description": "Autonomously manages device power state. Accepts time_window (start
/end) and min_power_reduction_pct as parameters.",
      "tags": ["energy", "optimization", "power-management"],
      "examples": [
        "Reduce energy by 15% during 01:00-06:00"
      ]
    }
  ]
}
```


A.3. Interaction Flow

The following steps illustrate the A2A interaction between the CA and DA.

Step 1: CA Delegates the Task

The CA initiates a task to delegate the energy-saving goal to the DA. It sends a ‘SendMessage‘ request.

```
POST /v1/message:send HTTP/1.1
Host: da-edge-router-01.example.com
Content-Type: application/json
Authorization: Bearer token
A2A-Version: 0.3

{
  "message": {
  "message_id": "msg-01",
    "role": "user",
    "parts": [
      {
        "text": "Execute skill network.energy.optimize with time_window from 01:00:00 to
06:00:00 and min_power_reduction_pct of 15%"
      }
    ]
  },
  "configuration": {
    "pushNotificationConfig": {
      "url": "https://ca.example.com/a2a/webhook",
      "authentication": {
        "schemes": ["Bearer"],
      }
    }
  }
}
```

Step 2: DA Acknowledges the Task

The DA receives the message, validates the parameters, and creates the task. It responds with a ‘SendMessageResponse‘ including the task details. The DA generates a new ‘context_id‘ for this interaction.

```
HTTP/1.1 200 OK
Content-Type: application/a2a+json

{
  "task": {
    "id": "task-energy-01",
    "contextId": "ctx-01",
    "status": {
      "state": "submitted",
      "timestamp": "2025-12-05T01:00:00Z"
    }
  }
}
```

Step 3: DA Performs Autonomous Action

At 02:15, the DA's internal monitoring detects that traffic on line card 3 has been below 5% for the last 30 minutes. Based on its internal logic for achieving the 15% power reduction goal, it decides to put the line card into a low-power state.

Step 4: DA Proactively Notifies the CA

After successfully putting the line card to sleep, the DA proactively informs the CA of the action taken.In this simplified example, the DA sends a message to the CA. In a production implementation, this could be delivered via A2A's push notification mechanism (TaskStatusUpdateEvent) or streaming (Subscribe to Task).

```
POST https://ca.example.com/a2a/webhook HTTP/1.1
Host: ca.example.com
Content-Type: application/a2a+json
Authorization: Bearer shared-secret-token
X-A2A-Notification-Token: da-edge-router-01-token

{
  "statusUpdate": {
    "taskId": "task-energy-01",
    "contextId": "ctx-01",
    "status": {
      "state": "working",
      "timestamp": "2025-12-05T02:15:00Z",
      "message": {
        "role": "agent",
        "parts": [
          {
            "text": "Autonomous action taken for task task-energy-01: Line card 3 placed
in low-power state due to low utilization. Current power savings: 18%."
          }
        ]
      }
    },
    "final": false
  }
}
```

Step 5: DA Reacts to Changing Conditions

At 05:30, traffic demand begins to increase. The DA's predictive traffic model forecasts that existing active line cards will exceed 80% utilization within the next 10 minutes. To prevent potential congestion, it autonomously wakes up line card 3.

Step 6: DA Sends Another Proactive Notification

The DA again informs the CA of its reactive, autonomous action.

```
POST https://ca.example.com/a2a/webhook HTTP/1.1
Host: ca.example.com
Content-Type: application/a2a+json
Authorization: Bearer shared-secret-token
X-A2A-Notification-Token: da-edge-router-01-token

{
  "statusUpdate": {
    "taskId": "task-energy-01",
    "contextId": "ctx-01",
    "status": {
      "state": "working",
      "timestamp": "2025-12-05T05:30:00Z",
      "message": {
        "role": "agent",
        "parts": [
          {
            "text": "Proactive action taken for task task-energy-01: Line card 3 awakened
 to meet anticipated traffic demand. Current power savings: 5%."
          }
        ]
      }
    },
    "final": false
  }
}
```

Step 7: DA Completes the Task

At the end of the time window (06:00), the DA concludes the optimization task and sends a final message to indicate the task is complete.

```
POST https://ca.example.com/a2a/webhook HTTP/1.1
Host: ca.example.com
Content-Type: application/a2a+json
Authorization: Bearer shared-secret-token
X-A2A-Notification-Token: da-edge-router-01-token

{
  "statusUpdate": {
    "taskId": "task-energy-01",
    "contextId": "ctx-01",
    "status": {
      "state": "completed",
      "timestamp": "2025-12-05T06:00:00Z",
      "message": {
        "role": "agent",
        "parts": [
          {
            "data": {
              "final_savings_percent": 16.2,
              "actions_taken": ["line_card_3_slept", "line_card_3_woken"]
            }
          }
        ]
      }
    },
    "final": true
  }
}
```

A.4. Key Observations from the Example

* Intent Delegation: The CA delegates a high-level goal ("optimize energy") without needing to know the specifics of how the DA will achieve it (which ports to sleep, when, etc.).

* Autonomy: The DA makes its own decisions based on local context (traffic load) to achieve the delegated goal.

* Proactiveness: The DA initiates communication to inform the CA of significant autonomous actions, providing visibility without requiring constant polling from the CA.

* Task-Oriented: The entire interaction is framed within a long-running task, which provides a clear context and lifecycle for the operation.

Authors' Addresses

Jinjie Yan
ZTE Corporation
Email: yan.jinjie@zte.com.cn

Xiaoqiu Zhang
China Mobile
Email: zhangxiaoqiu@chinamobile.com