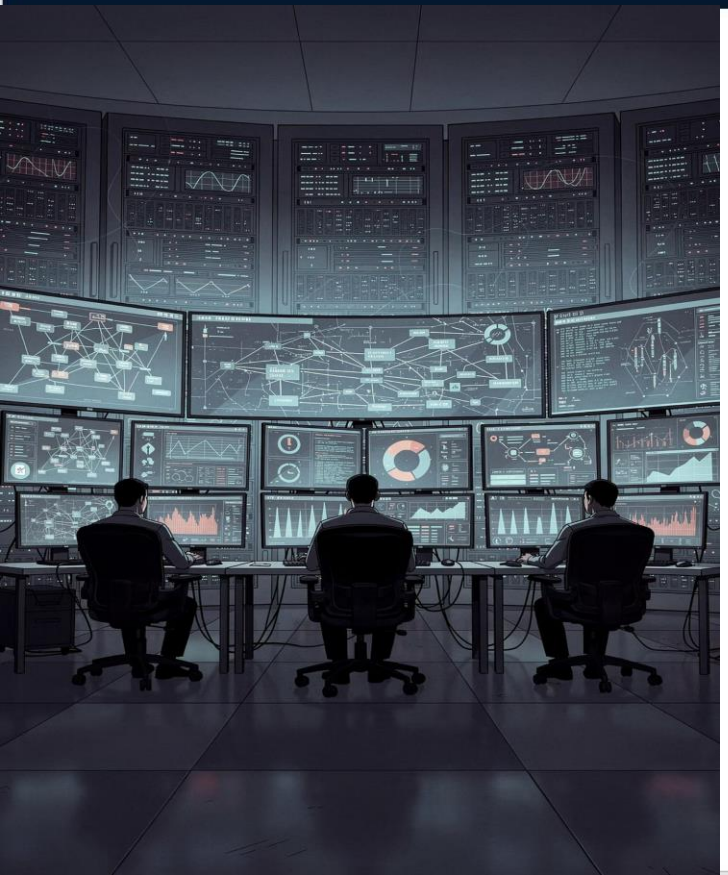
MCP Dev Summit Bengaluru



When MCP Meets Reality

Performance, Latency, and the Hidden Cost of AI Orchestration in Enterprises

Lessons from Network Troubleshooting and AI-Driven Remediation · MCP Dev Con

Partha Sarthy | Applied Materials

THE PROBLEM

Why MCP Is Exciting

MCP Dev Summit Bengaluru

Traditional Operations

1. Engineer
2. Telemetry + Logs
3. Config + Topology
4. Root Cause Analysis

AI-Driven Operations

1. Engineer
2. AI Agent
3. MCP
4. Enterprise Tools

MCP standardises access to enterprise capabilities — turning manual orchestration into automated intelligence.

USE CASE

The Network Troubleshooting Scenario

MCP Dev Summit Bengaluru

"Investigate why interface ge-0/0/0 is down."

Engineer → NOC Portal → AI Agent

Single natural-language query triggers full investigation

Telemetry MCP Server

Interface stats, error counters, optical levels

Config MCP Server

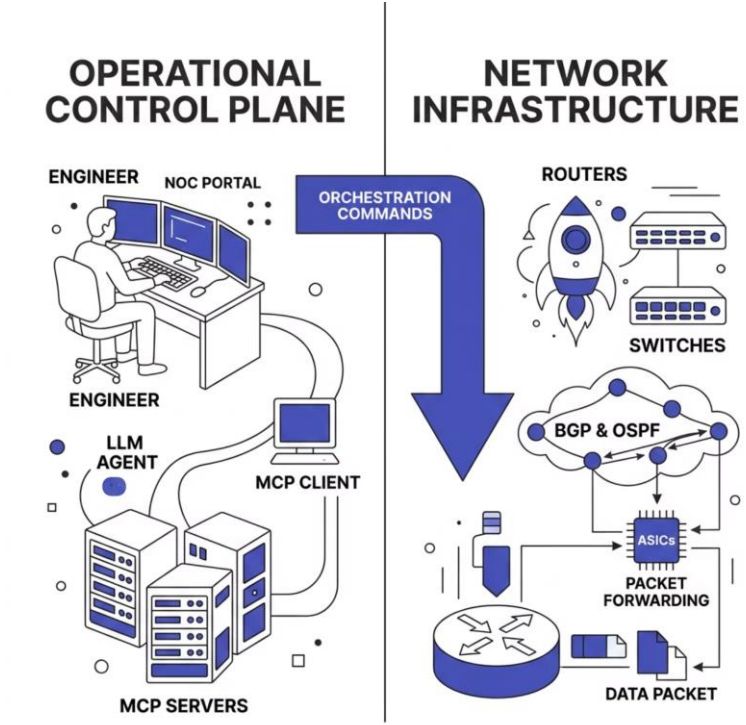
Recent config changes, rollback history

Topology + Incident Servers

Neighbour state, BGP sessions, open tickets

ARCHITECTURE

Understanding MCP Placement



MCP operates *above* the network

It is an orchestration layer — never an inline forwarding component.

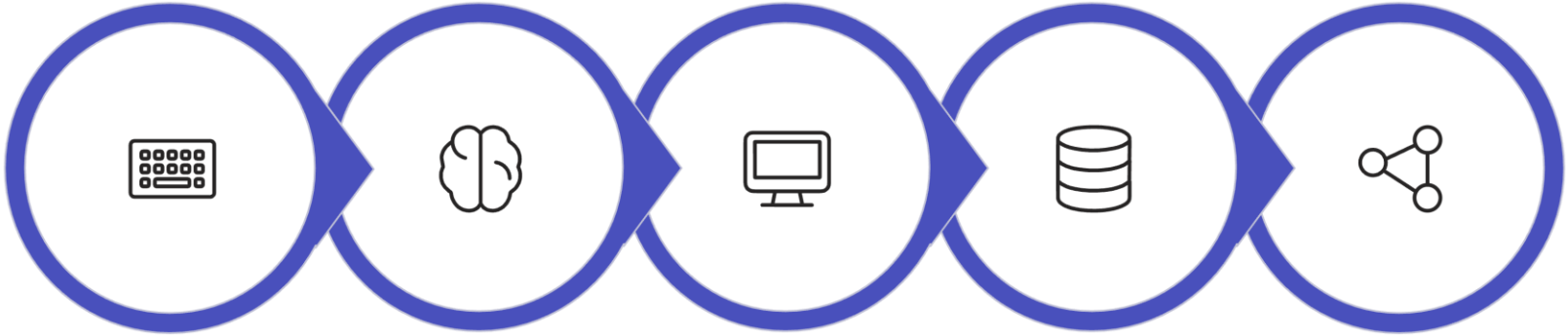
> ⚠ **MCP does NOT run inside the router.** The control plane and execution plane are categorically separate.

- Control plane: orchestrate, diagnose, remediate
- Data plane: route, forward, switch — at line rate

PIPELINE

Formal MCP Execution Pipeline

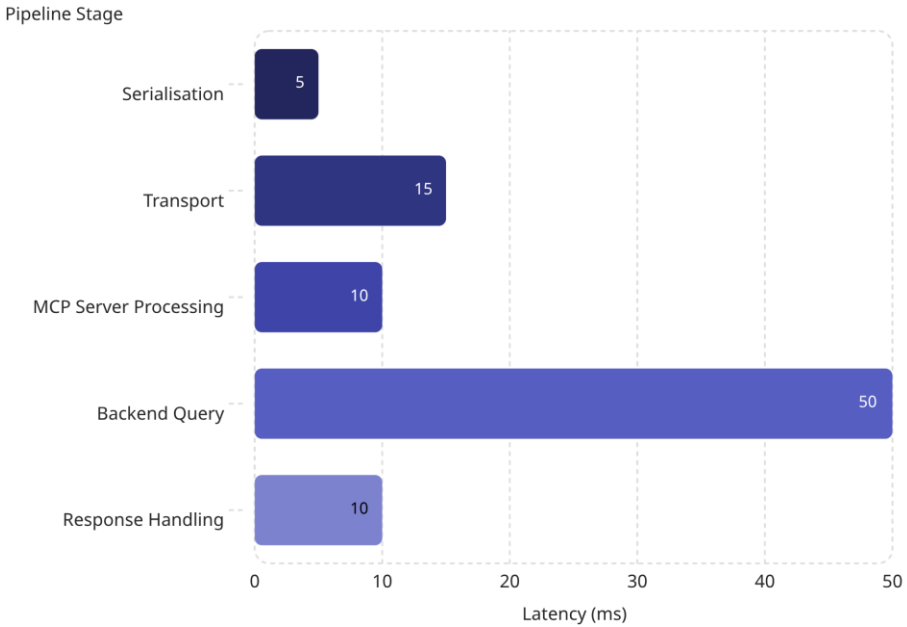
MCP Dev Summit Bengaluru



Every tool call has a measurable cost. Each hop in this pipeline contributes latency before the agent can reason on the result.

LATENCY

Where Latency Comes From



Single call: ~90 ms

Acceptable in isolation. Dangerous at scale.

×5 Sequential	×10 Sequential
450 ms	900 ms

⚠ Latency amplifies with every additional tool call in a sequential workflow.

OPTIMISATION

Sequential vs. Parallel Tool Invocation

MCP Dev Summit Bengaluru

❌ Sequential — 450 ms

1. **Telemetry** ~90 ms
2. **Config** ~90 ms
3. **Topology** ~90 ms
4. **Incidents** ~90 ms

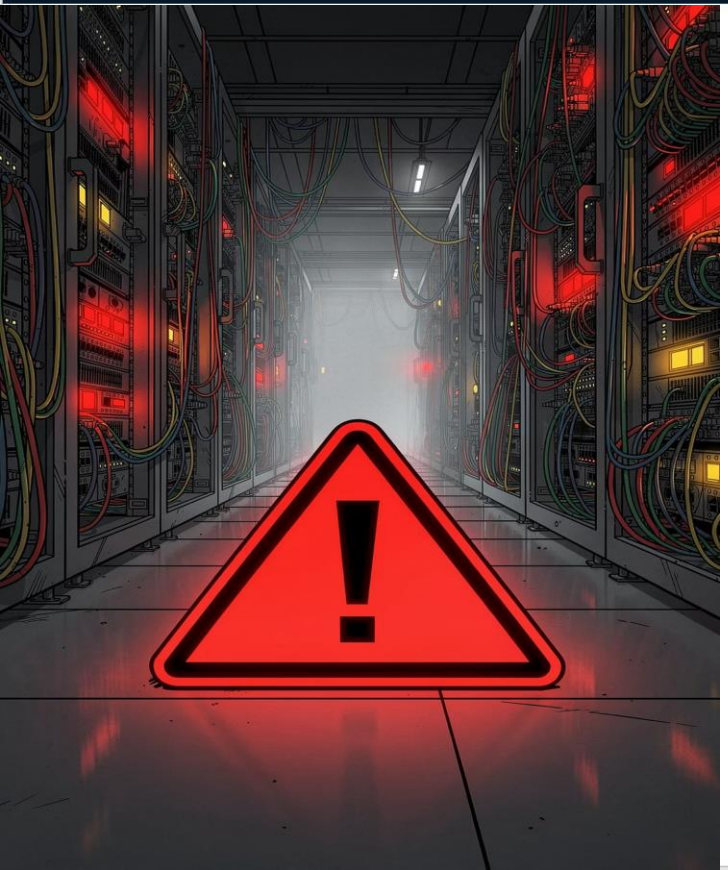
✅ Parallel — ~100 ms

All four tools execute concurrently via `asyncio.gather()`
4.5× latency reduction — architecture choice, not protocol tuning.

CRITICAL MISTAKE

MCP Must Never Enter the Data Plane

MCP Dev Summit Bengaluru



✅ MCP Belongs Here

- Investigation
- Diagnosis
- Recommendation
- Remediation

❌ Never This

1. Packet Arrives
2. MCP Tool Call
3. LLM Reasoning
4. Forward Packet

"Nobody would place an LLM in the packet forwarding path." MCP belongs in decision-making workflows, not forwarding loops.

Partha Sarthy | Applied Materials

FAILURE MODES

Production Failure Taxonomy

MCP Dev Summit Bengaluru

Latency Amplification

Sequential calls compound 90 ms into seconds

Data Movement Overhead

Large payloads serialised and deserialised per hop

Schema Drift

Tool contracts change; agents break silently

Observability Gaps

No tracing across LLM → MCP → backend hops

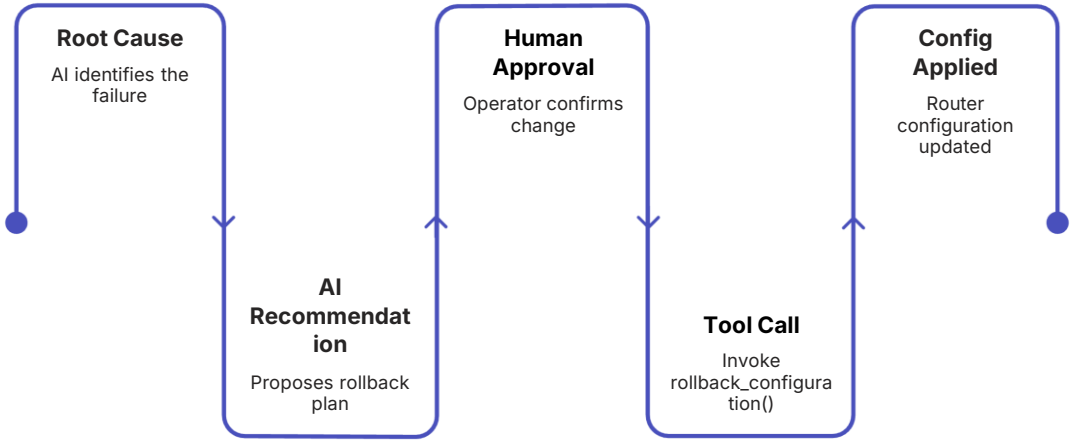
Boundary Violations

MCP invoked in latency-sensitive execution paths

REMEDIATION

Network Remediation Agent

MCP Dev Summit Bengaluru



Human-in-the-loop by design

The agent **recommends**. The operator **approves**. Infrastructure **executes**.

AI Agent

Determines root cause, proposes rollback

Approval Gate

Human operator confirms before any change

MCP Tool

`rollback_configuration()` via NETCONF

DESIGN PRINCIPLE

The Control Plane Principle

MCP Dev Summit Bengaluru

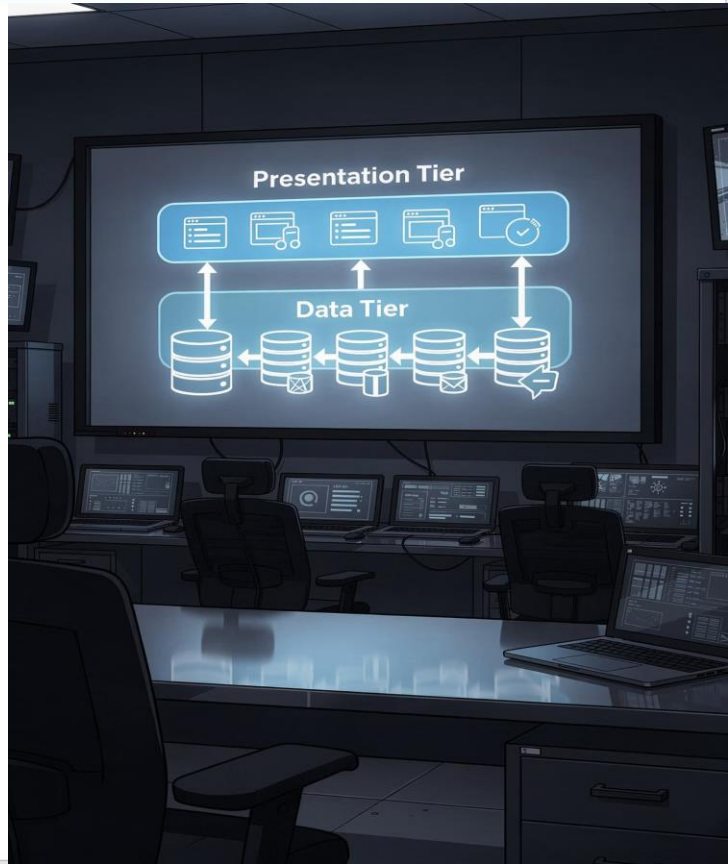
Operational Control Plane

- LLM Agent — reasoning
- MCP — orchestration
- Diagnostics — evidence
- Remediation — intent

Execution Plane

- Routing — BGP, OSPF
- Forwarding — ASICs, line rate
- Telemetry — gNMI, streaming
- Hardware — chips, optics

"BGP computes routes. ASICs forward packets. MCP orchestrates operational intelligence."



Partha Sarthy | Applied Materials

TOOL REGISTRY

Network Troubleshooting MCP Tool Registry

MCP Dev Summit Bengaluru

`get_interface_telemetry()`

Admin/oper status, error counters, optical power — sourced from gNMI / Prometheus

`get_recent_config_changes()`

Last N commits with diff and author — sourced from NETCONF / change management DB

`get_topology_neighbors()`

BGP/OSPF adjacency state, LLDP neighbours — sourced from topology service

`rollback_configuration()`

Revert to previous known-good config — requires human approval gate

MCP Servers expose capabilities. AI agents discover and compose them into investigative workflows.

PROTOCOL

MCP JSON-RPC Request

MCP Dev Summit Bengaluru

```
{
  "jsonrpc": "2.0",
  "id": 101,
  "method": "tools/call",
  "params": {
    "name": "get_interface_telemetry",
    "arguments": {
      "device_id": "R1",
      "interface_name": "ge-0/0/0"
    }
  }
}
```

What's happening here

01

MCP Client serialises intent

JSON-RPC 2.0 envelope wraps the tool call

02

Transport delivers to server

HTTP / stdio / SSE depending on deployment

03

Server routes to handler

Tool name resolves to registered function

PROTOCOL

MCP JSON-RPC Response

MCP Dev Summit Bengaluru

```
{
  "jsonrpc": "2.0",
  "id": 101,
  "result": {
    "device_id": "R1",
    "interface_name": "ge-0/0/0",
    "admin_status": "up",
    "oper_status": "down",
    "input_errors": 0,
    "output_errors": 0
  }
}
```

The server returns facts.

Not conclusions. Not remediation steps.

> The AI agent performs all reasoning. MCP Servers are **evidence providers**, not decision engines.

`oper_status: "down"` with zero errors signals a physical or upstream issue — the agent determines which.

IMPLEMENTATION

Troubleshooting Analyser Logic

MCP Dev Summit Bengaluru

```
class TroubleshootingAnalyzer:

  async def investigate_interface(
    self, device_id, interface_name
  ):
    telemetry, config, topology = await asyncio.gather(
      self.telemetry_tool(device_id, interface_name),
      self.config_tool(device_id),
      self.topology_tool(device_id)
    )
    return self.determine_root_cause(
      telemetry, config, topology
    )
```

Key design decisions

→ Parallel execution

`asyncio.gather()` — all three tools fire simultaneously

→ Reasoning stays in the agent

`determine_root_cause()` — LLM synthesises evidence

→ MCP Servers stay dumb

Servers fetch and return — no analysis logic

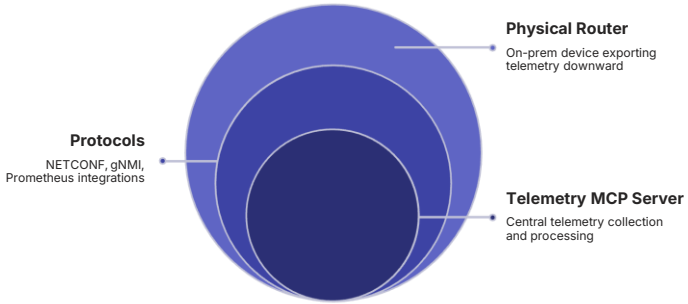
IMPLEMENTATION

MCP Server Implementation

MCP Dev Summit Bengaluru

```
@mcp.tool()
async def get_interface_telemetry(
  device_id: str,
  interface_name: str
):
  data = telemetry_backend.query(
    device_id, interface_name
  )
  return {
    "admin_status": data.admin_status,
    "oper_status":  data.oper_status,
    "input_errors": data.input_errors,
    "output_errors": data.output_errors
  }
```

Thin adapter pattern



The MCP Server exposes enterprise capabilities over a standard protocol. Existing infrastructure is **unchanged**.

OBSERVABILITY

Design Observability from Day One

MCP Dev Summit Bengaluru

Distributed Tracing

Correlate LLM reasoning spans → MCP tool calls → backend queries with a single trace ID

Per-Tool Latency Metrics

Instrument each tool invocation — identify which backend is the bottleneck

Schema Version Enforcement

Version every tool contract; alert on drift before agents break silently in production

Audit Trails

Every remediation action logged with agent reasoning, approval identity, and timestamp

KEY TAKEAWAYS

What to Carry Forward

MCP Dev Summit Bengaluru

1 **MCP standardises enterprise capability access**

One protocol — telemetry, config, topology, remediation

2 **Every tool call has measurable cost**

90 ms per hop · sequential chains hit seconds fast

3 **Parallelise early, not after**

`asyncio.gather()` is your first performance lever

4 **MCP belongs in the control plane**

Orchestration, diagnosis, remediation — never forwarding

5 **Observability is not optional**

Trace every hop from intent to infrastructure change

Call to Action

MCP Dev Summit Bengaluru

"Deploy MCP deliberately, not everywhere."

Match protocol scope to operational scope. The control plane is where MCP thrives.

Partha Sarthy | Applied Materials

References

1. **MCP & JSON-RPC request response examples**

   https://modelcontextprotocol.io/, https://www.jsonrpc.org/

2. **Open Telemetry**

   https://opentelemetry.io/

3. **NETCONF**

   https://datatracker.ietf.org/doc/html/rfc6241

4. **gNMI**

   https://github.com/openconfig/reference/blob/master/rpc/gnmi/gnmi-specification.md/

5. **Google's The Tail at Scale**

Questions?

MCP Dev Summit Bengaluru