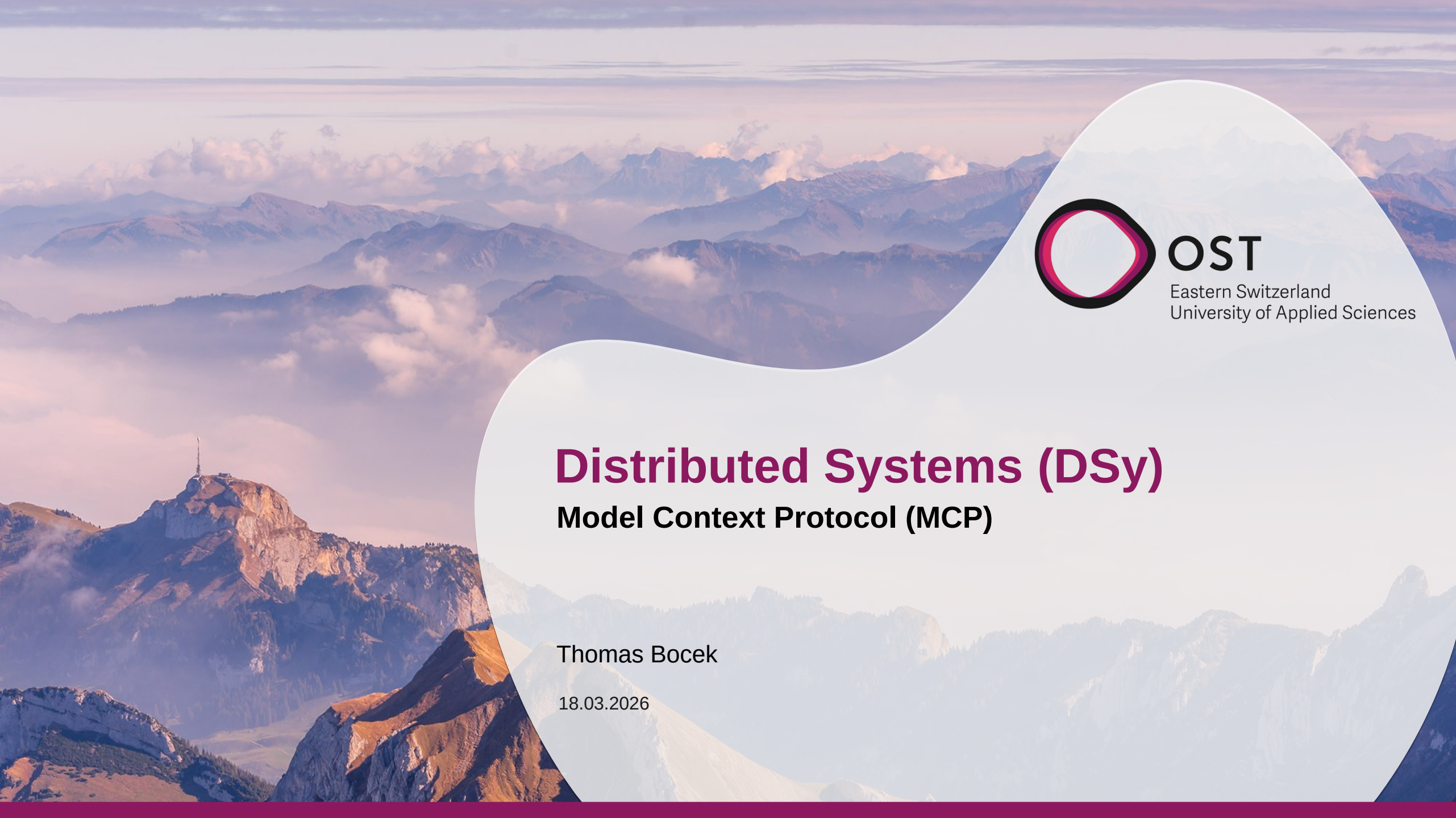


OST
Eastern Switzerland
University of Applied Sciences

Distributed Systems (DSy)

Model Context Protocol (MCP)

Thomas Bocek

18.03.2026

Learning Goals

- Lecture 6 (Model Context Protocol)
  - Explain what MCP is, why it exists, and how tool calling works (tools/list, tools/call, JSON-RPC)
  - Describe the MCP message flow between user, client, LLM, and MCP server
  - Discuss challenges of running MCP in the browser and how WebMCP addresses them

    → more about AI at OST:
    - AI Applications
    - AI Foundations
    - Applied AI with Deep Learning
    - Generative AI

Model Context Protocol (MCP)

- By Anthropic, November 2024, open-source
  - Claude Opus / Sonnet
- Donated to Linux Foundation (AAIF), 2025
  - No single company controls it [link]
- Connects AI agents to external tools, data sources, and services
  - Universal protocol
  - Implement once, works everywhere
- Before MCP: every AI app needed a custom connector per tool (N×M problem)
  - Doesn't scale, fragile, duplicated effort
- Born from a single dev's frustration copy-pasting between Claude Desktop and his IDE [link]
  - Not a corporate strategy, solving an annoying workflow problem
- Adoption [link]
  - OpenAI March 2025, Google in April 2025, Microsoft May 2025

MCP API

- tools/list
  - JSON-RPC 2.0

```
→ {"jsonrpc":"2.0","id":1,"method":"tools/list"}

← {"jsonrpc":"2.0","id":1,"result":{"tools":[
    {"name":"measure_endpoint","description":"Measure
     response time of a URL", "inputSchema"
      {"type":"object","properties":
       {"url":{"type":"string"}}}}]}}
```

  - MCP client asks "what can you do?"
  - MCP server returns array of tools
  - Each tool: name, description, input JSON Schema

- tools/call
  - JSON-RPC 2.0

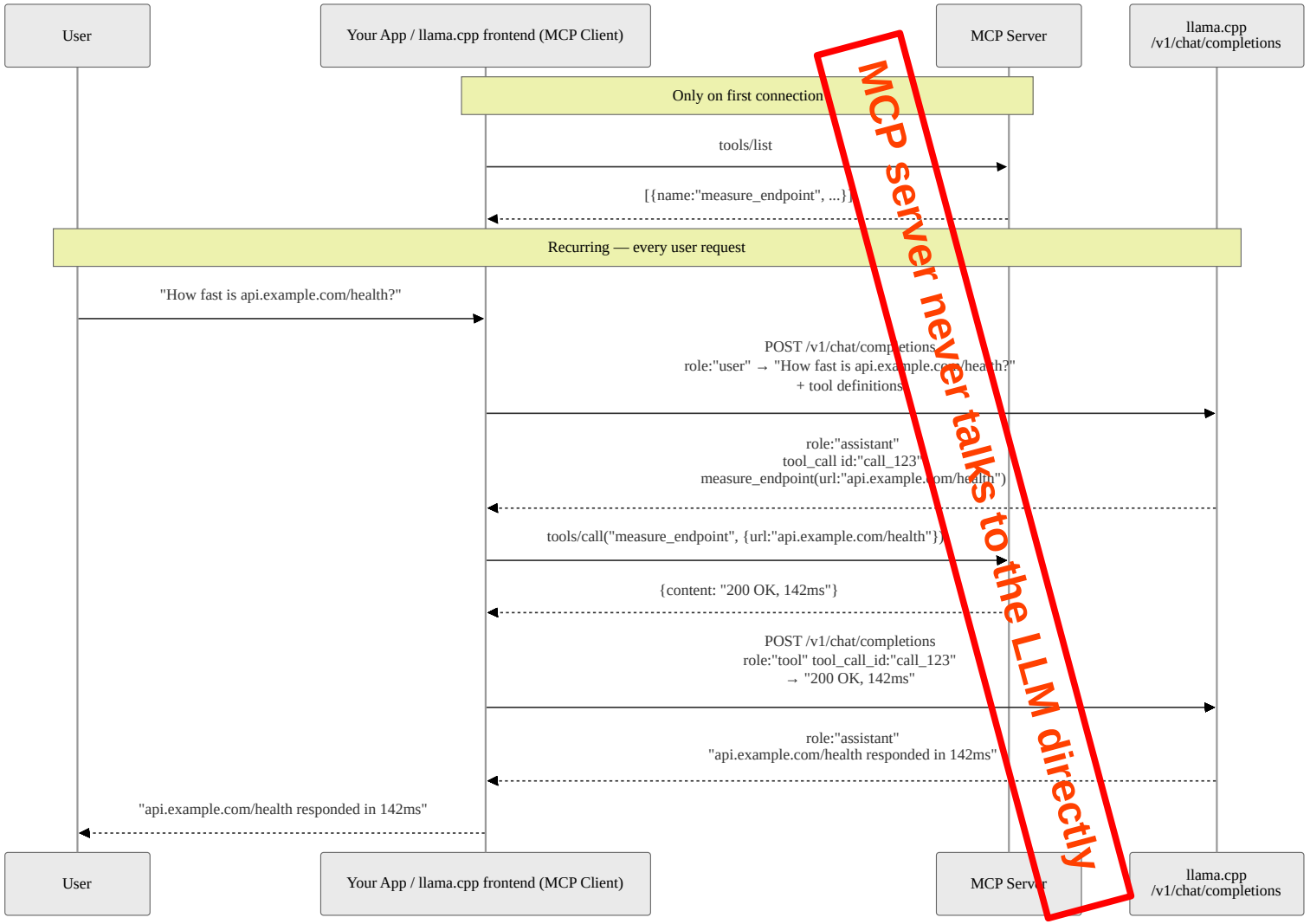
```
→ {"jsonrpc":"2.0","id":2,"method":"tools/call",
   "params":{"name":"measure_endpoint","arguments"
            {"url":"api.example.com/health"}}}

← {"jsonrpc":"2.0","id":2,"result":{"content"
    [{"type":"text","text":"200 OK, 142ms"}]}}
```

  - MCP client: "run this tool with these arguments"
  - MCP server executes, returns result
  - Result is an array of content blocks (text, image, etc.)

OST

MCP Sequence Diagram



Simple MCP Server

- Example in golang
  - 1 tool: measure_endpoint
  - LLM decides when to call, what to call, what parameters
    - Every LLM calls needs to add available tools:

```
{"model": "my-model",
 "messages": [
   {"role": "user", "content": "Check api.example.com and
db.example.com"}
 ],
 "tools": [{
   "type": "function",
   "function": {
     "name": "measure_endpoint",
     "description": "Measure response time of a URL",
     "parameters": {
       "type": "object",
       "properties": {"url": {"type": "string"}},
       "required": ["url"]}}}]}
```

```
{"choices": [{"message": {
   "role": "assistant",
   "content": null,
   "tool_calls": [
     {"id": "call_1", "type": "function",
      "function": {"name": "measure_endpoint", "arguments": "{\"url\":\"api.example.com\"}"}},
     {"id": "call_2", "type": "function",
      "function": {"name": "measure_endpoint", "arguments": "{\"url\":\"db.example.com\"}"}}
   ]
}}]}
```

- Reply from LLM
- Roles must follow: user → assistant (tool_call) → tool (result) → assistant (answer)
- llama.cpp
  - LLM inference in C/C++
  - Frontend in Svelte

OST

Simple MCP Server

- OpenAI chat API is stateless, also tool calls
- 2nd call to the LLM, now with tool results

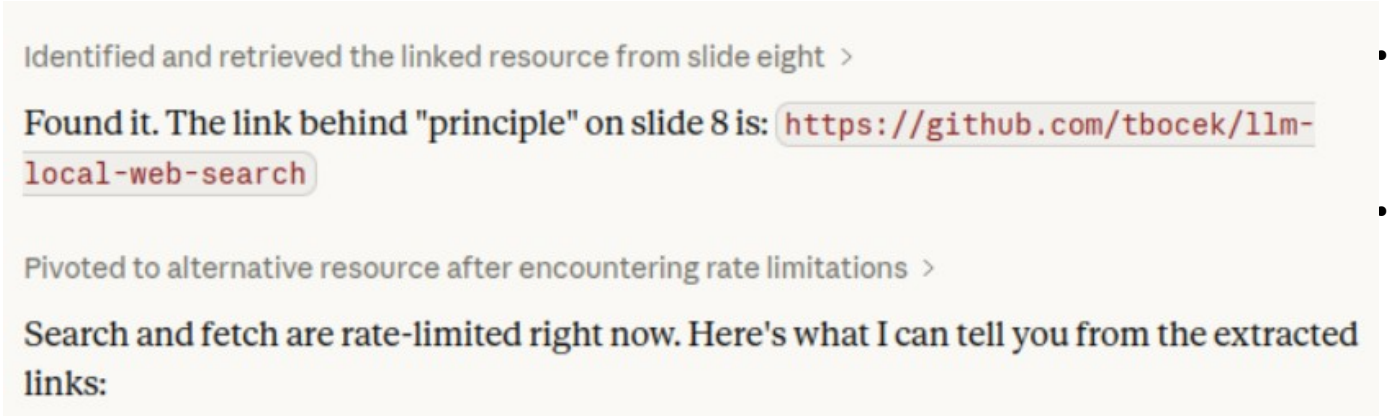
```
{"model": "my-model",
 "messages": [
   {"role": "user", "content": "Check api.example.com and
db.example.com"},
   {"role": "assistant", "content": null, "tool_calls": [
     {"id": "call_1", "type": "function",
      "function": {"name": "measure_endpoint", "arguments": "{\"url\":
\"api.example.com\"}"}},
     {"id": "call_2", "type": "function",
      "function": {"name": "measure_endpoint", "arguments": "{\"url\":
\"db.example.com\"}"}}
   ]},
   {"role": "tool", "tool_call_id": "call_1", "content": "200 OK,
142ms"},
   {"role": "tool", "tool_call_id": "call_2", "content": "200 OK,
89ms"}
 ]
}
```

- tool_call_id must match — LLM generates it, you echo it back
- Multiple tool calls? Return multiple role:"tool" entries, each with its own tool_call_id
- Wrong ID or role = error or hallucinated result

OST

MCP in the Browser?

- Sample principle
  - Send available tools + query
  - Get tool calls name/arguments
  - Resend available tools + query + result from tool call
- Problem:



- Firefox extension that intercepts fetch() calls to /v1/chat/completions
  - Injects a client_web_search tool into the request, no MCP server needed, exetension is the MCP server
  - Solves the "could not fetch" problem
- The Future: WebMCP
  - W3C draft (Feb 2026), navigator.modelContext browser API
  - Websites register tools natively, e.g., search engine could expose search(), a news site getArticle()
    - Extension?

OST