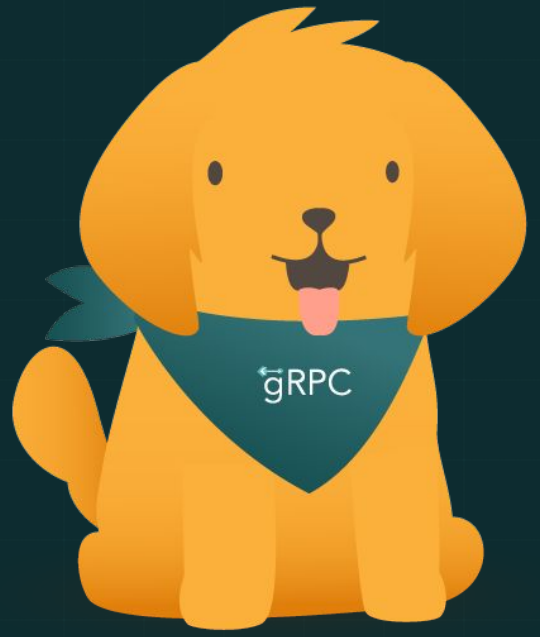
Scaling AI: gRPC as transport backbone for Enterprise MCP



Pawan Bhardwaj
gRPC Maintainer
Sr. Software Engineer, Google

What is gRPC ?

- A cutting-edge, open source, high-performance state-of-the-art **R**emote **P**rocedure **C**all (RPC) framework.
- **gRPC speaks your language**.
- **gRPC runs on your OS**.
- **gRPC is performant**.
- **gRPC is open sourced**.
- **gRPC is extensible with plugins**.

gRPC has been widely embraced

Discord

ENGINEERING & DEVELOPERS

HOW DISCORD STORES TRILLIONS OF MESSAGES

Our data services sit between the API and our ScyllaDB clusters. They contain roughly one gRPC endpoint per database query and intentionally contain no business logic. The big feature our data services provide is request coalescing. If multiple users are requesting the

reddit

Reddit's move to gRPC (self.RedditEng)
submitted 2 years ago by SussexPondPudding Lisa O'Cat

Spotify®

Adopting gRPC at Spotify

JFokus 2019

VOLVO

Dev tips

gRPC API for flexible automotive development

NETFLIX

Netflix Technology Blog · Follow
Published in Netflix TechBlog · 8 min read · Sep 3, 2021

By *Alex Borysov*, *Ricky Gardiner*

At Netflix, we heavily use gRPC for the purpose of backend to backend communication. When we process a request it is often beneficial to know

LinkedIn

LinkedIn-wide migration from Rest.li to gRPC. gRPC will offer better performance, support for more programming languages, streaming, and a robust open source community.

CISCO

Systems administrators in enterprise companies who are constantly upgrading, repurposing, and managing thousands of switches, listen up! Starting with release 17.7 of Cisco IOS XE, is support for a suite of Google Remote Procedure Call (gRPC)-based microservices that will simplify and lighten your workloads.

Square

FEBRUARY 26, 2015 | 1 MINUTE READ

gRPC — cross-platform open source RPC over HTTP/2

Collaborating with Google on open sourcing gRPC

salesforce

How gRPC is enabling Salesforce's unified interoperability strategy

BROADCOM®

Enterprise requirement and bottlenecks for MCP Adoption

What Enterprise Users need from MCP Transport

Requirement and bottlenecks

Bandwidth Saturation

Json-RPC strings consume heavy bandwidth, leading to inefficient network utilization.

Performance Overhead

Parsing highly verbose dynamic text payloads forces continuous high CPU utilization at scale.

Security Constraints

No clear native support for security at the transport layer, complicating enterprise compliance.

Mesh Integration

Requires a sidecar (Envoy) to hook into cloud native routing patterns, increasing architectural complexity.

Migration Challenge: MCP Tools

Migration of existing infrastructure using RPCs (gRPC) to use JSON-RPC is a significant bottleneck.

gRPC as the Transport Backbone

Transforming Enterprise Requirements into High-Performance, Cloud Native Infrastructure

Core Foundations of gRPC

Protobuf

Strict binary serialization ensures build-time type safety while slashing payload bandwidth and parsing overhead

HTTP/2

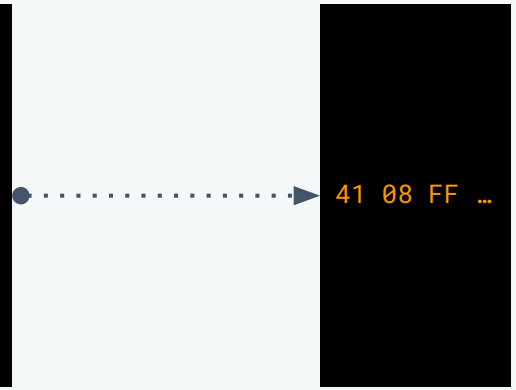
Native multiplexing and bi-directional streaming optimize high-concurrency agentic workflow over a single reusable connection.

Mesh/Security

Industrial-strength infrastructure provide native mTLS security and seamless integration with cloud-native service meshes.

gRPC Serializes Data Using Protobuf


```
service RouteGuide {
  rpc getFeature (Point) returns (Feature)
  {}
}

messages Point {
  int32 latitude=1;
}
```



What's Good 😊

- **Strict Contract Safety at Build Time.**
- **Ultra-dense Binary format**
- **Polyglot Code Generation**
- **Reduced Parsing Overhead**
- **Single Source of Truth**

Tradeoff 😕

- **Non-Human Readable Format**
- **Additional dependency**
- **Strict vs Dynamic Schemas**

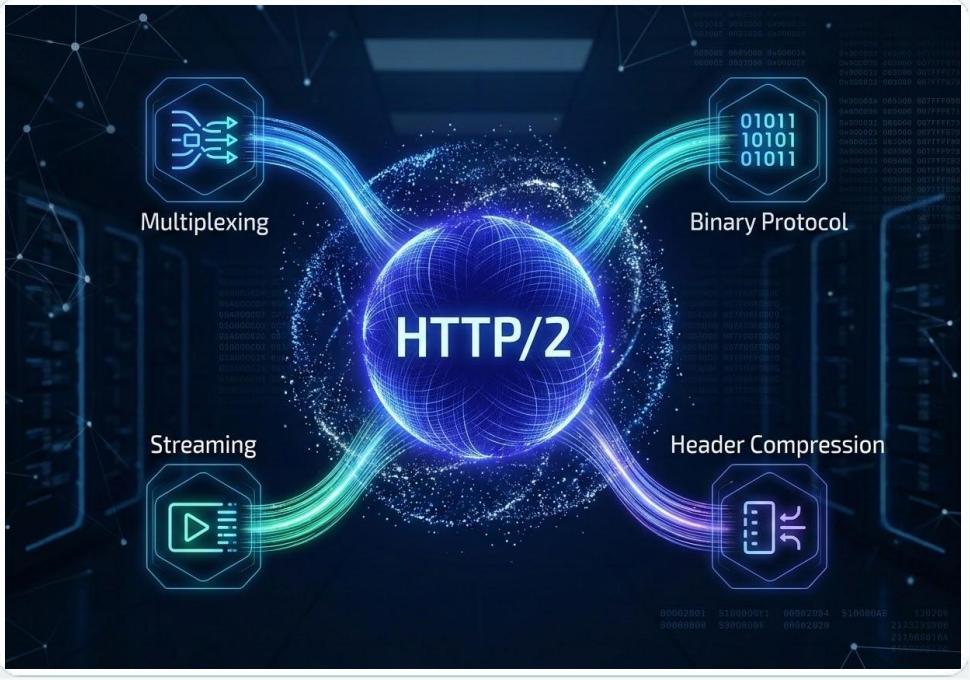
gRPC Built on HTTP/2

What's Good 😊

- Native Multiplexing
- Header Compression
- Bidirectional Streaming
- Deadline propagation

Tradeoff 😕

- Limited legacy support
- Inspection Complexity



Enterprise Infrastructure

Proxyless Mesh, Native Security, and Observability

Proxyless Mesh

Natively supports xDS API and integrates with industry-standard control planes (like Istio), enabling sidecarless discovery and load balancing without sidecar overhead.

Native Security (mTLS & SPIFFE)

Provides out-of-the-box support for mTLS and integrates with SPIFFE to ensure cryptographically verifiable identities for MCP servers and clients.

Universal Observability

Native integration with OpenTelemetry for unified metrics, tracing, and logging across the entire infrastructure stack.

Migration & Standards

Facilitates migration of existing RPC infrastructure to MCP tools using the same backbone used by production micro-services.

Implementation

Formalizing the protocol, Scaling the SDK, and Enabling a Pluggable Ecosystem

Mapping the Spec: mcp-grpc-transport-proto

Repository: https://github.com/GoogleCloudPlatform/mcp-grpc-transport-proto

Canonical Source of Truth

Will be used for all language implementations, serving as the central reference for the gRPC transport protocol.

Synchronized Evolution

Updates are driven by the MCP transport workgroup to maintain strict parity with the latest specification releases.

Native gRPC Service

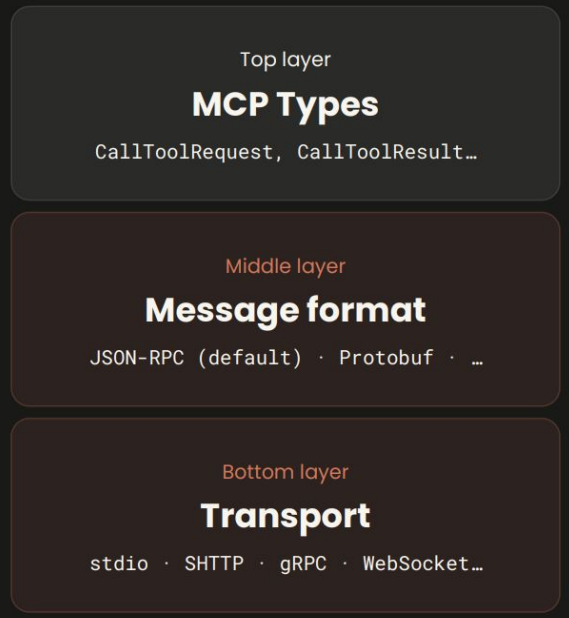
A first-class service Mcp with dedicated methods (ListResources, CallTool, GetPrompt) instead of a JSON-RPC wrapper.

Direct Semantic Mapping

MCP primitives map directly to typed Protobuf messages, ensuring contract safety without text-based overhead.

CHANGES

Pluggable transports



"Pluggable transports" usually means swapping both bottom layers.

gRPC with protobuf over JSON-RPC makes no sense — you'd smuggle JSONRPC as a string and lose protobuf's whole value proposition.

Solution: Dispatcher pattern — swap both layers

Reference: Path to V2 for MCP SDKs - Max Isbey, Anthropic

CHANGES

The Dispatcher pattern

Draft PR · design settled

Reference: Path to V2 for MCP SDKs - Max Isbey, Anthropic

MCP gRPC Python Implementation

Repository: https://github.com/GoogleCloudPlatform/mcp-grpc-transport-py

Active Development (WIP)

Will provide pluggable transport implementation for MCP Python, currently in progress.

Dispatcher-Based Architecture

Based on dispatchers defined in official transport; facilitates adding MCP endpoints to existing gRPC services.

Standard Evolution

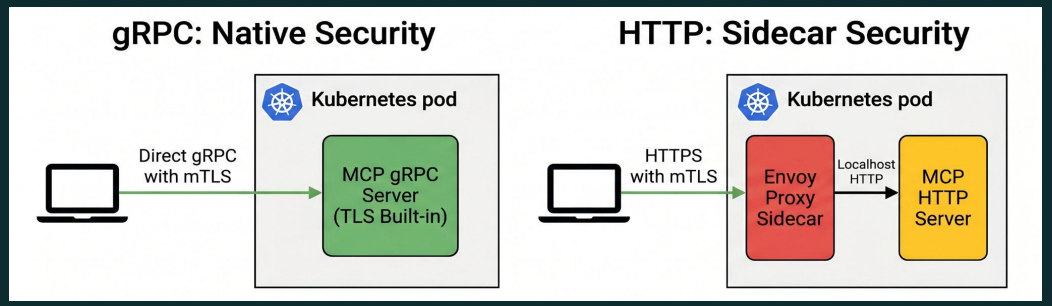
Built in lockstep with mcp-grpc-transport-proto and mcp-python to maintain strict parity with MCP specs.

Unified Developer Experience

Identical tool creation for JSON-RPC and gRPC. Simply swap JSONRPCDispatcher for GrpcDispatcher.

Demo

Topology



Identical Implementation: Tool creation logic is shared across HTTP and gRPC deployments.

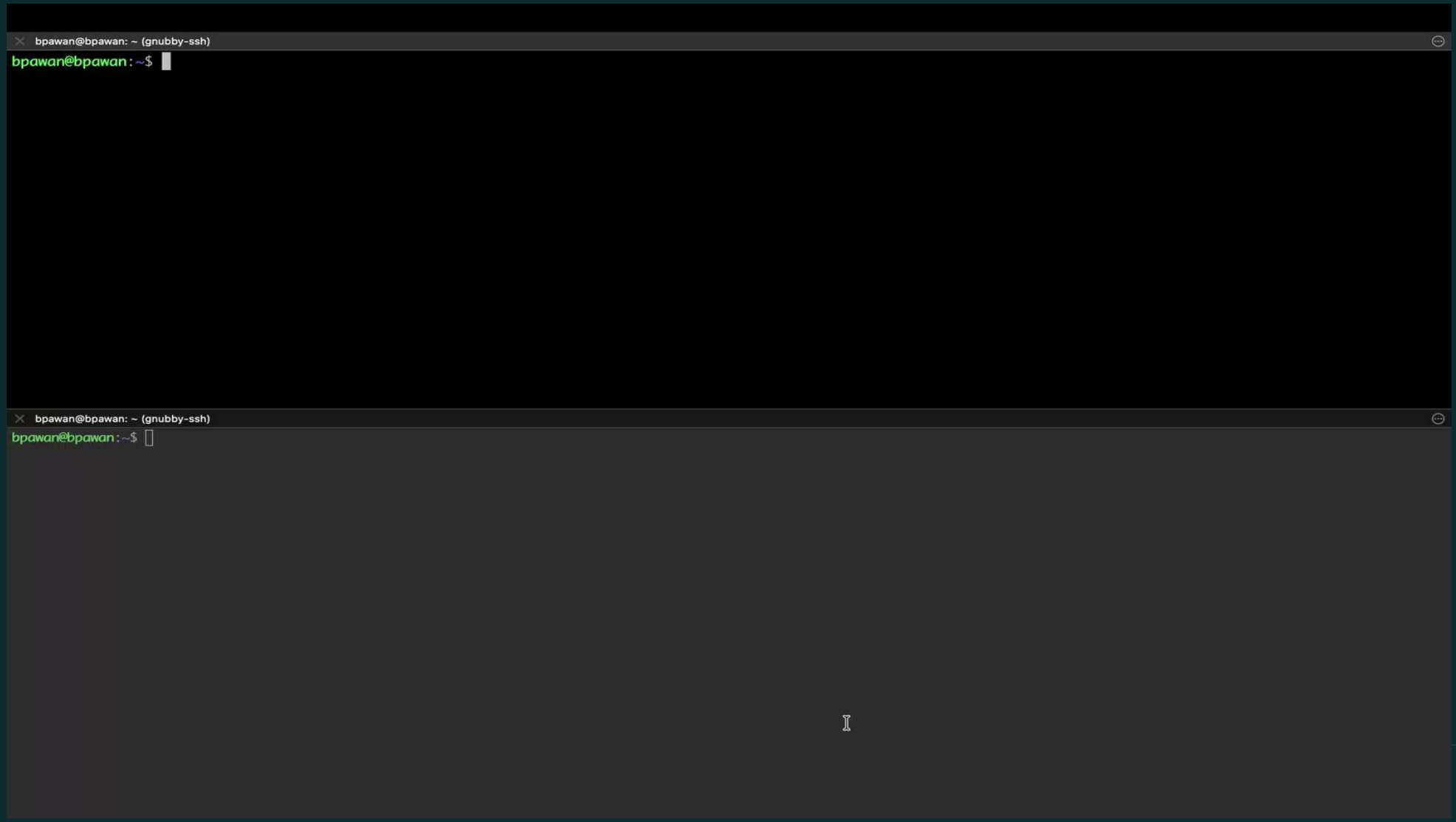
Modular Dispatchers:

- **JSON-RPC:** Included natively in the core mcp package.
- **gRPC**: Distributed via the mcp-grpc-transport specialized package.

Plug-and-Play: Simply inject the desired Dispatcher during session initialization.

```python
from mcp.server.runner import ServerRunner
from mcp_grpc_transport import GrpcServerDispatcher

# 1. Initialize your high-level MCPServer
app = MCPServer("log-query-grpc-server")

# 2. Define your tools using decorators
@app.tool(description="Query system logs with multiple filters.")
async def query_logs(
    query: str,
    level: str = "INFO",
    limit: int = 10,
    services: Optional[List[str]] = None,
    include_metadata: bool = True
) -> str:
    """Query system logs with filters."""
    # Implementation logic...
    return f"Results for '{query}' at level {level}"

# 3. Simple pattern using ServerRunner and GrpcServerDispatcher
async def main():
    # The ServerRunner acts as the per-connection orchestrator.
    # It bridges the MCPServer instance to the gRPC dispatcher.
    runner = ServerRunner(
        server=app._lowlevel_server,
        dispatcher=GrpcServerDispatcher()
    )

    # Start the receive loop and handle the initialization handshake
    await runner.run()

if __name__ == "__main__":
    anyio.run(main)
```

Demo


bpawan@bpawan: ~ (gnubby-ssh)
bpawan@bpawan:~$
bpawan@bpawan: ~ (gnubby-ssh)
bpawan@bpawan:~$

What's Next

Moving forward together

Roadmap

- **SEP-2598: Pluggable Transports (Under Review)**: https://github.com/modelcontextprotocol/modelcontextprotocol/pull/2598
- **Language Support**: Python under-development, will be follow up with other
- **Expansion beyond Python**
- **Conformance test for gRPC Transport**
- **Migration Playbook for existing gRPC services to tools and Changing Transports.**

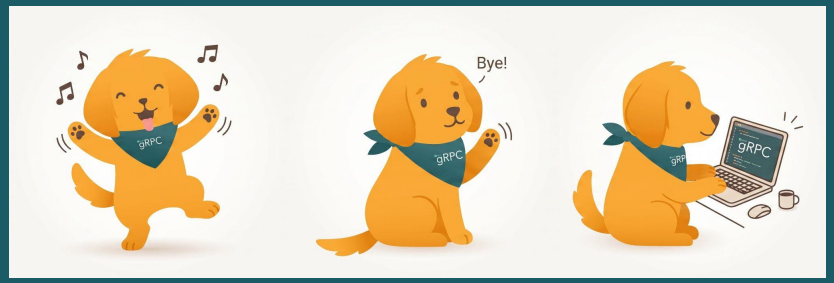
Thank You !!

Site : grpc.io
YouTube Channel: youtube.com/@grpcio

gRPC Conf North America : 3 September, 2026
gRPC Conf Bangalore: To be announced

Meetup
meetup.com/grpcio
meetup.com/grpc-bangalore

Join the mailing list
groups.google.com/g/grpc-io
groups.google.com/g/grpc-io-announce



Feedback

Scaling AI: gRPC as Transport Backbone for Enterprise MCP