LINEMAN wongnai

gRPC xDS Load Balancing

Dev Mountain Tech Festival
2022-03-19

Contents

- What is gRPC
- Envoy & Universal data plane API
- xDS
- What’s coming next

Who is this talk for

- What is gRPC « Dev
- Envoy & Universal data plane API « Ops
- xDS « Ops
- What's coming next « Ops

About Me

- Manatsawin Hanmongkolchai
- Senior Architect @ LINE MAN Wongnai
- **We're hiring!**
  - https://careers.lmwn.com

gRPC

What is gRPC

- It's API definition language - think Swagger
- But the ecosystem is heavily defined by Google
  - So you get usable client & server codegen in many languages - PHP, Python, Ruby, JavaScript, C#, Java, Go

What is gRPC

```
syntax = "proto3";

package sms.v2;

service SMSService {
  rpc SendSMS(SendSMSRequest) returns (SendSMSResponse);
}

message SendSMSRequest {
  string phone_no = 1;
  string message = 2;
}
```

What is gRPC

```go
func main() {
  conn, _ := grpc.Dial("sms:3000", …)

  client := smsv2.NewSMSServiceClient(conn)

  resp, err := client.SendSMS(context.TODO(), &smsv2.SendSMSRequest{
     PhoneNo: "+6620000000",
     Message: "hello world",
  })
}
```

Why gRPC

1. You get models generated in many languages
2. Errors are handled as exceptions (no need for `raise_for_error()`)

```
export interface SendSMSRequest {
   /**
    * Phone number in E164 format
    */
   phoneNo: string;
   message: string;
}
```

Why gRPC

1. You get models generated in many languages
2. Errors are handled as exceptions (no need for `raise_for_error()`)

```
class SendSMSRequest(Message):
    phone_no: typing.Text = ...
    "Phone number in E164 format"
    message: typing.Text = ...
    def __init__(
        self,
        *,
        phone_no: typing.Text = ...,
        message: typing.Text = ...,
    ) → None: ...
```

Why gRPC

3. It simplify team communication

Why gRPC

4. It allow schema first development

Envoy

and the Universal Data Plane API

What is Envoy

- It's reverse proxy originally created by Lyft (now hosted by CNCF)
- Now used by Istio, Consul Connect, Ambassador, Gloo, Contour

Why Envoy

The `ngx_http_api_module` module (1.13.3) provides REST API for accessing various status information, configuring upstream server groups on-the-fly, and managing key-value pairs without the need of reconfiguring nginx.

> This module is available as part of our commercial subscription.

How do you config Envoy?

- Load config from YAML file

How do you config Envoy?

- Load config from YAML file
- Load config from gRPC server
  - Config changes can be sent via gRPC stream

How do you config Envoy?

- So yes, you can codegen Envoy schema

```
routeConfig := &routev3.RouteConfiguration{
  Name: targetHostPortNumber,
  VirtualHosts: []*routev3.VirtualHost{
     {
        Name:    targetHostPort,
        Domains: []string{fullName, targetHostPort, targetHostPortNumber, svc.Name},
        Routes: []*routev3.Route{{
           Name: "default",
           Match: &routev3.RouteMatch{
              PathSpecifier: &routev3.RouteMatch_Prefix{},
           },
```

xDS

- Envoy config protocol is called xDS
- xDS will be the base of Universal Dataplane Platform - one config format for L4/L7 data plane

xDS

- ?DS stands for
  - **Cluster Discovery Service** (CDS) - nginx `upstream` object
  - **Endpoint Discovery Service** (EDS) - nginx `upstream` items
  - **Listener Discovery Service** (LDS) - nginx `server` blocks
  - **Route Discovery Service** (RDS) - nginx `location` blocks
  - **Aggregated Discovery Service** (ADS)
  - etc.

Recap

- Envoy is reverse proxy like nginx
- Envoy have configuration API (xDS)
  - Load balancing pool, listeners, routes
- xDS is going to be universal dataplane API

gRPC xDS

where I no longer care about service mesh

Problem with gRPC

Cool event, but not public

https://speakerdeck.com/whs/a-date-with-grpc

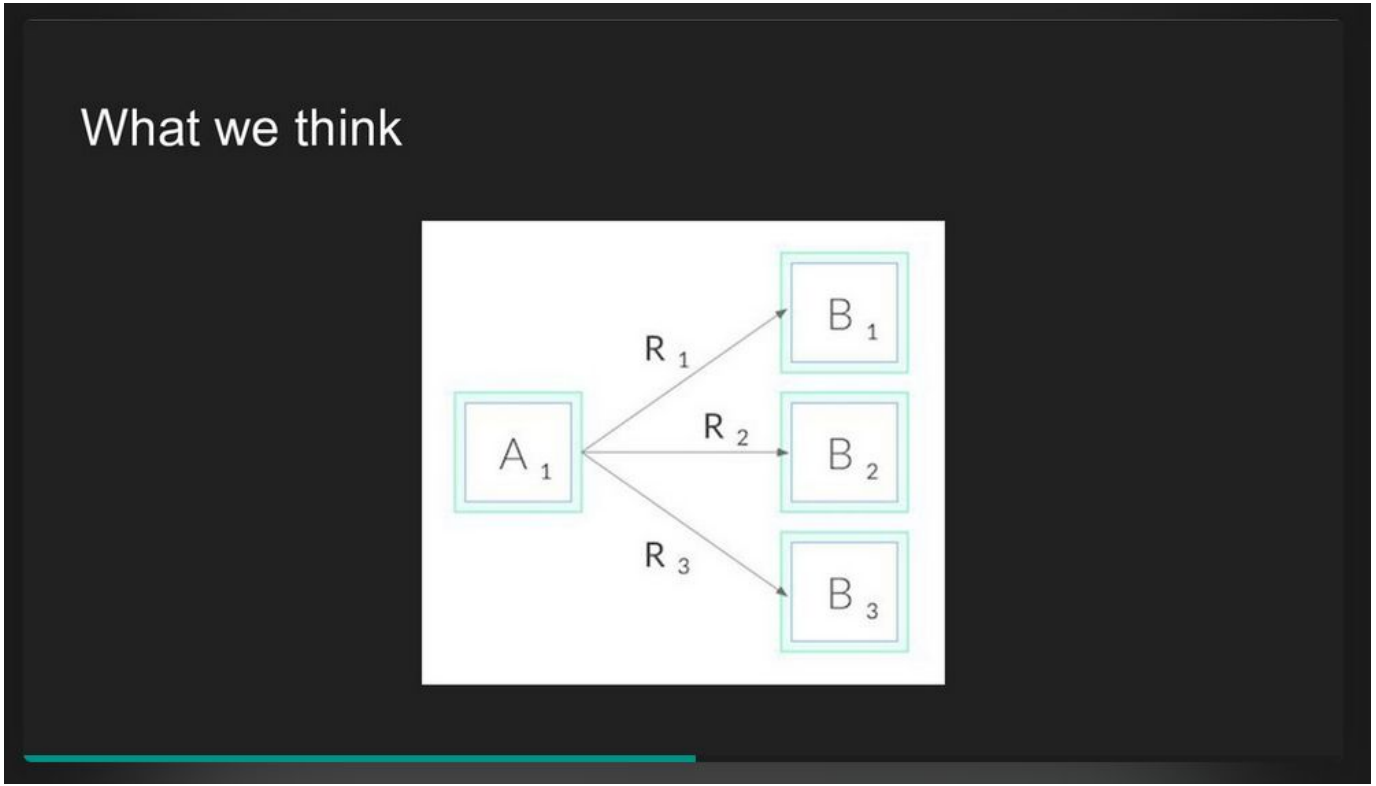
Problem with gRPC

Kubernetes service balance
connections, not calls

https://speakerdeck.com/whs/a-date-with-grpc

wongnai

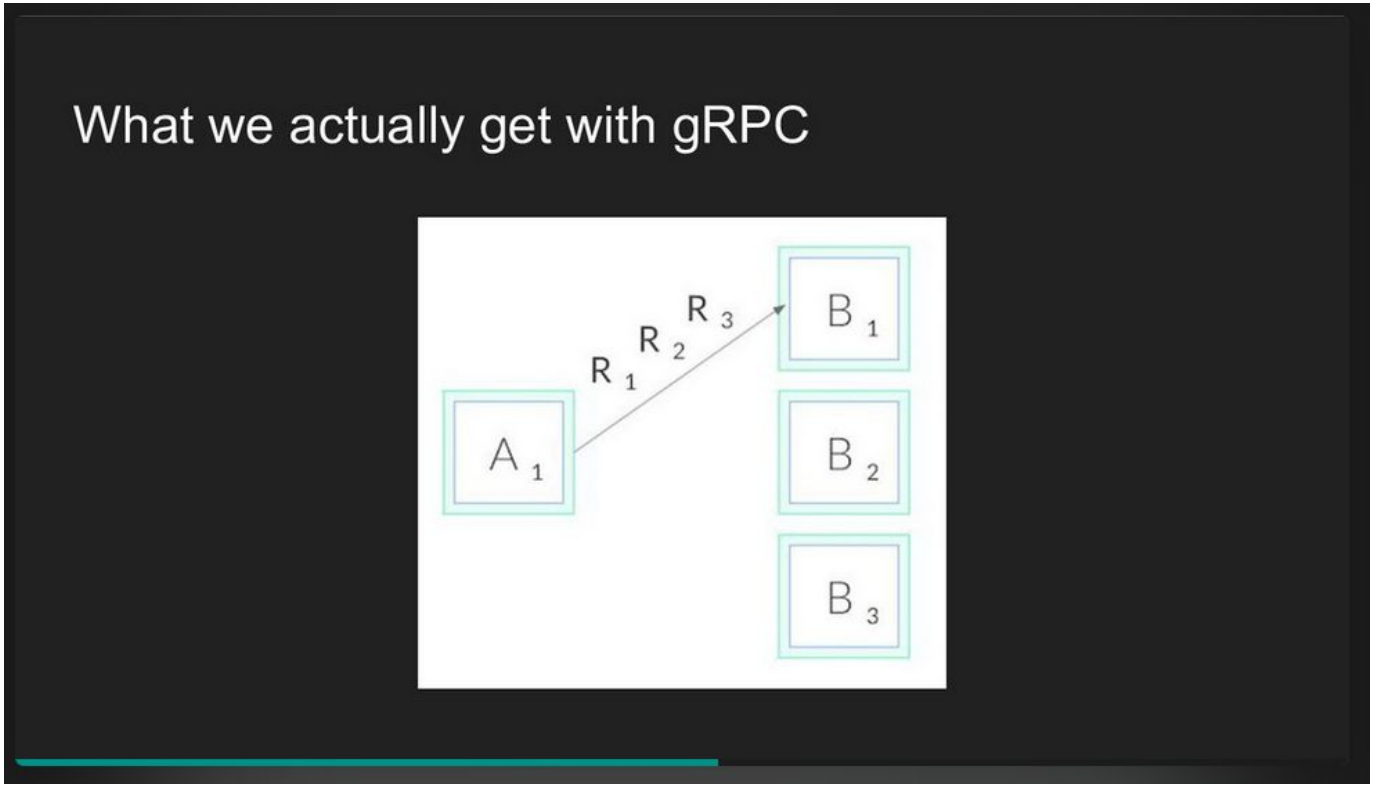
Problem with gRPC



https://speakerdeck.com/whs/a-date-with-grpc

wongnai

Problem with gRPC



https://speakerdeck.com/whs/a-date-with-grpc

wongnai

Problem with gRPC

What we actually get with gRPC

Pods

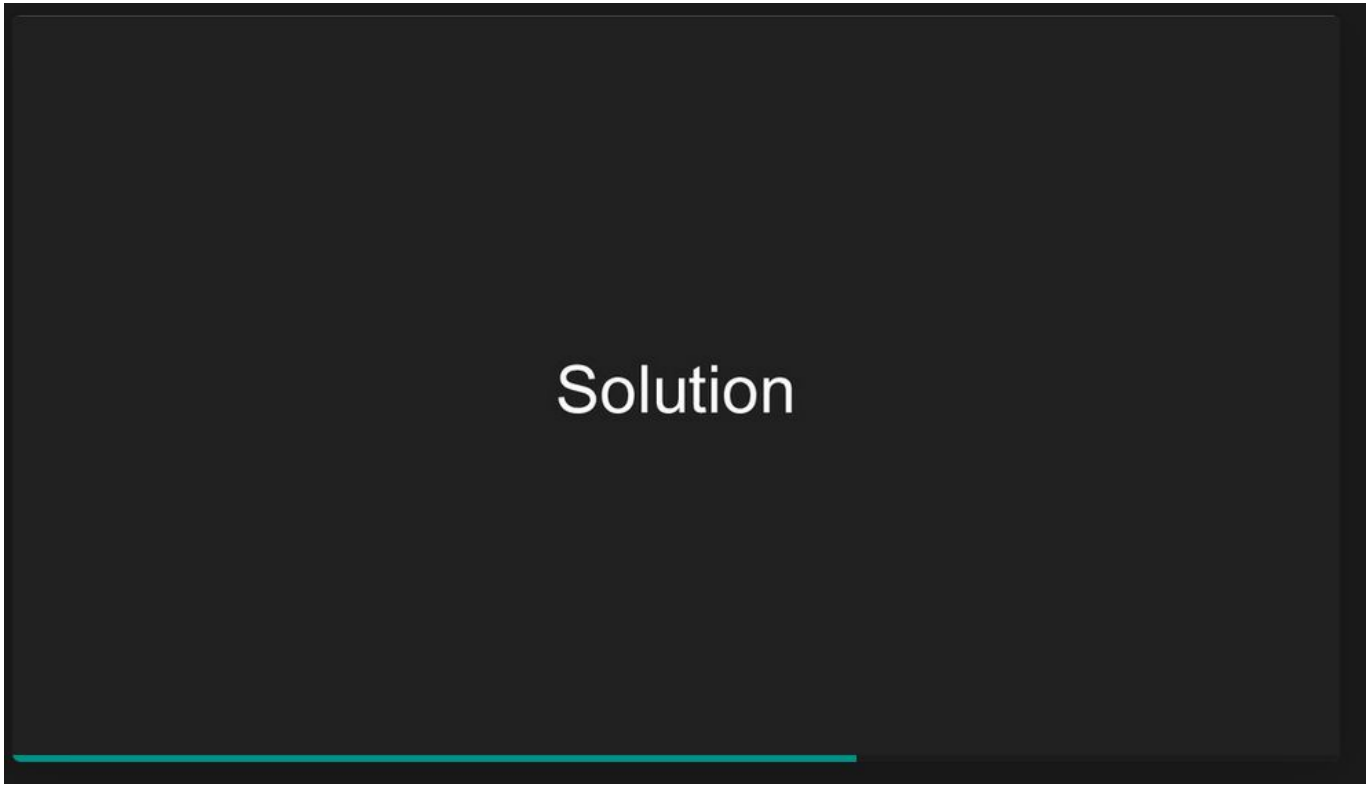
voting

Name	Node	Status	Restarts	Age	CPU (cores)	Memory (bytes)
voting-676c65b4d9-xs-	gke-thomas-dev-default-pool-e8682290-5kjd	Running	0	6 minutes	0	44 Mi
voting-676c65b4d9-tpr	gke-thomas-dev-default-pool-e8682290-c51q	Running	0	6 minutes	0	44.246 Mi
voting-676c65b4d9-72	gke-thomas-dev-default-pool-e8682290-qrpt	Running	0	2 hours	0	42.777 Mi
voting-676c65b4d9-nv	gke-thomas-dev-default-pool-e8682290-3k50	Running	0	2 hours	0.01	43.402 Mi
voting-676c65b4d9-srl	gke-thomas-dev-default-pool-	Running	0	2 hours	0	43.090 Mi

https://speakerdeck.com/whs/a-date-with-grpc

wongnai

Problem with gRPC



https://speakerdeck.com/whs/a-date-with-grpc

wongnai

Problem with gRPC

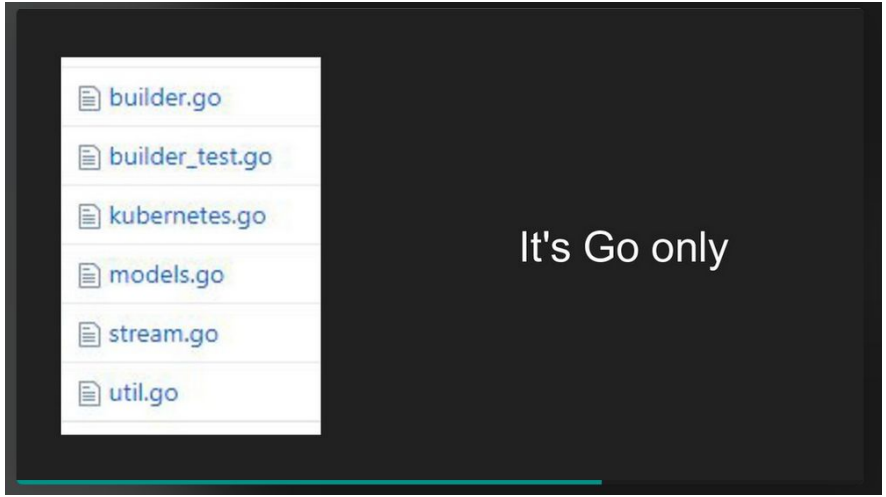
1. gRPC client side load balance

https://speakerdeck.com/whs/a-date-with-grpc

wongnai

2022 Update

- You still could do it…
  - My team did try it
- But it’s deprecated
- And you have to do it per-language



Problem with gRPC

2. Read Kubernetes Blog

https://speakerdeck.com/whs/a-date-with-grpc

wongnai

Problem with gRPC

gRPC Load Balancing on Kubernetes without Tears

Wednesday, November 07, 2018

Author: William Morgan (Buoyant)

Many new gRPC users are surprised to find that Kubernetes's default load balancing often doesn't work out of the box with gRPC. For example, here's what happens when you take a simple gRPC Node.js microservices app and deploy it on Kubernetes:

Pods

voting

Name	Node	Status	Restarts	Age	CPU (cores)	Memory (bytes)
voting-676c65b4d9-xs-	gke-thomas-dev-default-pool-e8682290-5kjd	Running	0	6 minutes	0	44 Mi
voting-676c65b4d9-tpr	gke-thomas-dev-default-pool-e8682290-c51q	Running	0	6 minutes	0	44.246 Mi
voting-676c65b4d9-72	gke-thomas-dev-default-pool-e8682290-qrpt	Running	0	2 hours	0	42.777 Mi
voting-676c65b4d9-nv	gke-thomas-dev-default-pool-e8682290-3k50	Running	0	2 hours	0.01	43.402 Mi
voting-676c65b4d9-srl	gke-thomas-dev-default-pool-e8682290-k21j	Running	0	2 hours	0	43.090 Mi
voting-676c65b4d9-tj6	gke-thomas-dev-default-pool-e8682290-3k50	Running	0	2 hours	0	43.535 Mi
voting-676c65b4d9-tpj	gke-thomas-dev-default-pool-e8682290-5kjd	Running	0	2 hours	0	43.398 Mi
voting-676c65b4d9-8fl	gke-thomas-dev-default-pool-e8682290-65g4	Running	0	2 hours	0	44.074 Mi

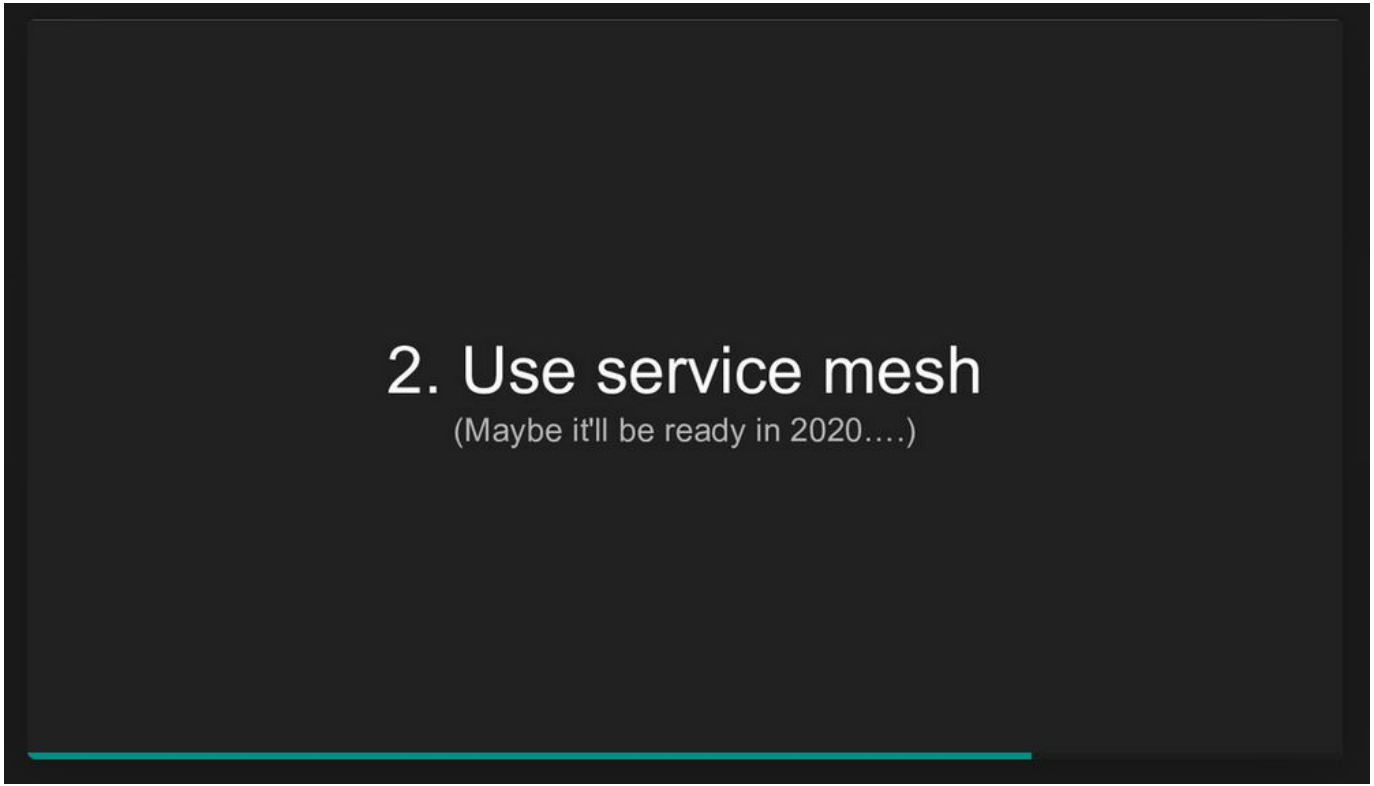
While the `voting` service displayed here has several pods, it's clear from Kubernetes's CPU graphs that only one of the pods is actually doing any work—because only one of the pods is receiving any traffic. Why?

In this blog post, we describe why this happens, and how you can easily fix it by adding gRPC load balancing to any Kubernetes app with Linkerd, a CNCF service mesh and service sidecar.

https://speakerdeck.com/whs/a-date-with-grpc

wongnai

Problem with gRPC



https://speakerdeck.com/whs/a-date-with-grpc

wongnai

2022 Update

- Istio didn't work out well for us
  - Both Wongnai & LINE MAN team tried, too much tuning works for unclear losses & gains
- Linkerd didn't work out well for us
  - More operational overhead
  - Insufficient load testing

2022 Update

- But, option 3 is coming in 2022

xDS is coming to gRPC !!

- gRPC is going to support xDS without sidecar
- Client side load balancing for all languages, fault injection, routing, circuit breaking, mTLS

How to add gRPC xDS to your application

- Python
  1. Upgrade to grpcio v1.36.0
  2. There is no 2

How to add gRPC xDS to your application

- Go
  1. `import _ "google.golang.org/grpc/xds"`
  2. There is no 2

How to add gRPC xDS to your application

- Node (@grpc/js only)

```
1. import * as grpcJsXds from '@grpc/grpc-js-xds'
2. grpcJsXds.register()
```

Config

- Where does the config comes from?

xDS server

LMWN's xDS server for Kubernetes

- https://github.com/wongnai/xds
- We're running it in production since Feb 2022
  - www.wongnai.com/cooking is mostly powered by xDS

xDS server

- Use 5% CPU & 100MB to support 100 pods
- ~2% application CPU overhead to handle config changes
- Handle only config, not data - not a point of failure

xDS server

- Cons: gRPC xDS support is buggy currently
  - C-based (Python), Go is stable
  - Node.js has a race condition
  - Java has another race condition

xDS server

- What you still need service mesh: **observability**
  - But you can already integrate OpenTelemetry for that

Future

where the mastermind show themself

The mastermind

- Why is Google building xDS v2 & gRPC xDS?

The mastermind

Traffic Director

Enterprise-ready traffic management for open service mesh.

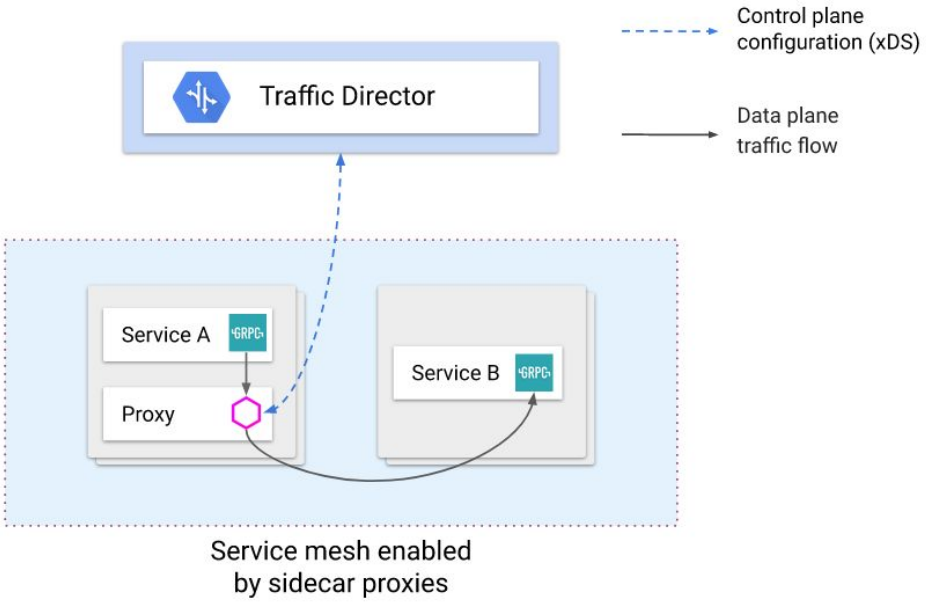
Try it free

View documentation for this product.

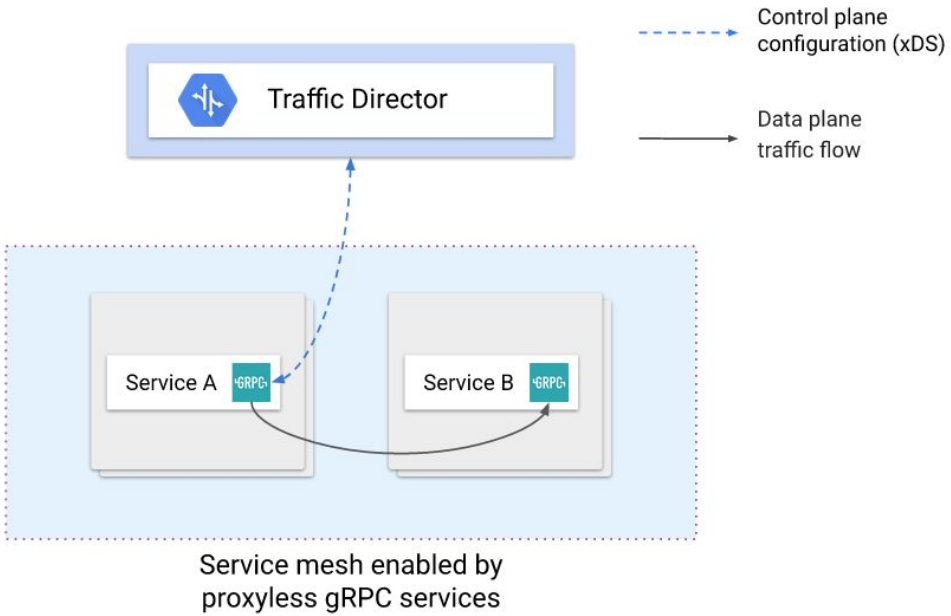
Toil-free traffic management for your service mesh

Service mesh is a powerful abstraction that's become increasingly popular to deliver microservices and modern applications. In a service mesh, the service mesh data plane, with service proxies like Envoy, moves the traffic around and the service mesh control plane provides policy, configuration, and intelligence to these service proxies. Traffic Director is GCP's fully managed traffic control plane for service mesh. With Traffic Director, you can easily deploy global load balancing across clusters and VM instances in multiple regions, offload health checking from service proxies, and configure sophisticated traffic control policies. Traffic Director uses open xDS APIs to communicate with the service proxies in the data plane, which ensures that you are not locked into a proprietary interface.

Traffic director



Traffic director



Google Cloud Load Balancer (Preview)

Modes of operation

You can configure External HTTP(S) Load Balancing in the following modes:

- **Global external HTTP(S) load balancer.** This is a global load balancer that is implemented as a managed service on Google Front Ends (GFEs). It uses the open-source Envoy proxy to support advanced traffic management capabilities such as traffic mirroring, weight-based traffic splitting, request/response-based header transformations, and more. This load balancer is currently in **Preview**.
- **Global external HTTP(S) load balancer (classic).** This is the classic external HTTP(S) load balancer that is global in Premium Tier but can be configured to be regional in Standard Tier. This load balancer is implemented on Google Front Ends (GFEs). GFEs are distributed globally and operate together using Google's global network and control plane.
- **Regional external HTTP(S) load balancer.** This is a regional load balancer that is implemented as a managed service on the open-source Envoy proxy. It includes advanced traffic management capabilities such as traffic mirroring, weight-based traffic splitting, request/response-based header transformations, and more. This load balancer is currently in **Preview**.

Google Cloud Load Balancer (Preview)

Product	Load balancing scheme	IP Protocol options
Global external HTTP(S) load balancer	EXTERNAL_MANAGED	TCP
Global external HTTP(S) load balancer (classic)	EXTERNAL	TCP
Regional external HTTP(S) load balancer	EXTERNAL_MANAGED	TCP
SSL proxy load balancer	EXTERNAL	TCP
TCP proxy load balancer	EXTERNAL	TCP
Network load balancer	EXTERNAL	TCP, UDP, or L3_DEFAULT
Internal TCP/UDP load balancer	INTERNAL	TCP or UDP
Internal HTTP(S) load balancer	INTERNAL_MANAGED	TCP
Traffic Director	INTERNAL_SELF_MANAGED	TCP

My questions to the future

- Once Istio control plane become SaaS, what is the next step for Istio
- What's the grand scheme for Envoy & xDS on Google Cloud?

Q & A

Get this slide deck - **speakerdeck.com/whs**
LMWN is hiring! - careers.lmwn.com

Slides