gRPC Conf 2020

A Simplified Service Mesh with gRPC

Srini Polavarapu
Engineering Manager at Google

gRPC Is Awesome!

- High performance, open source and standards-based.
- Feature rich:
  - Connection mgmt, request multiplexing, bi-di streaming and flow control.
  - Deadlines, cancellation and metadata.
  - Pluggable, interceptors and more.
- Multi-language, multi-platform.
- High industry adoption.
- Works great with Protocol Buffers.

Awesome framework for microservices based applications.

Service Meshes Are Cool!

gRPC Conf 2020

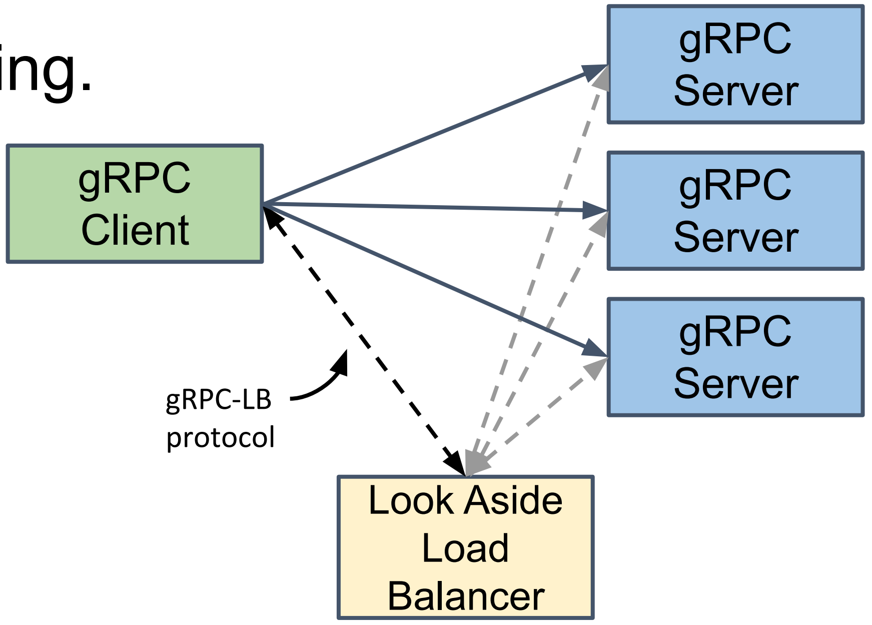
- ➢ Service discovery - Service lookup by name.
- ➢ Traffic Management - Request routing and load balancing.
- ➢ Security - Authentication and authorization between services.
- ➢ Observability - Metrics, monitoring, logging and debugging.

Solves complexities of microservices based architecture.

Istio - A popular service mesh solution

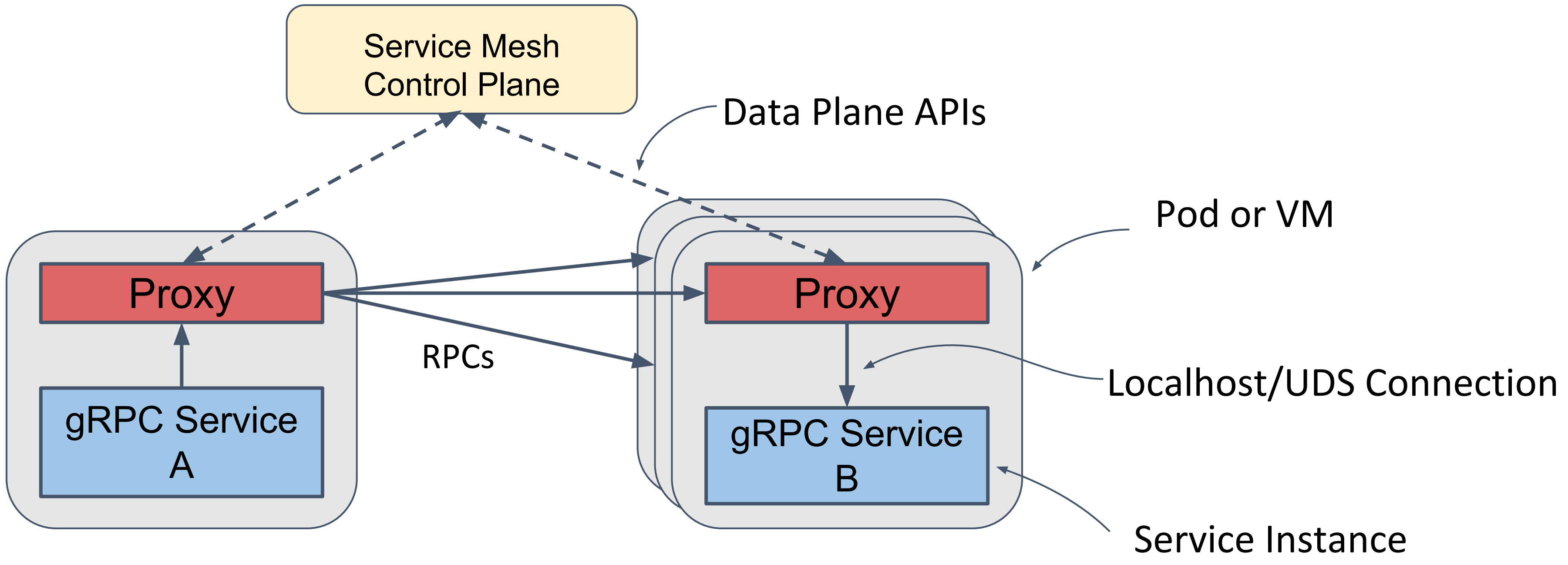
gRPC + Service Mesh

- No native service mesh integration in gRPC.
- Comes with only a DNS name resolver.
- Only pick-first and round-robin built-in load balancing.
- Anything more requires implementing your own:
  - Resolver and balancer plugins.
  - gRPC-LB server.



Next gRPC evolution - service mesh integration!

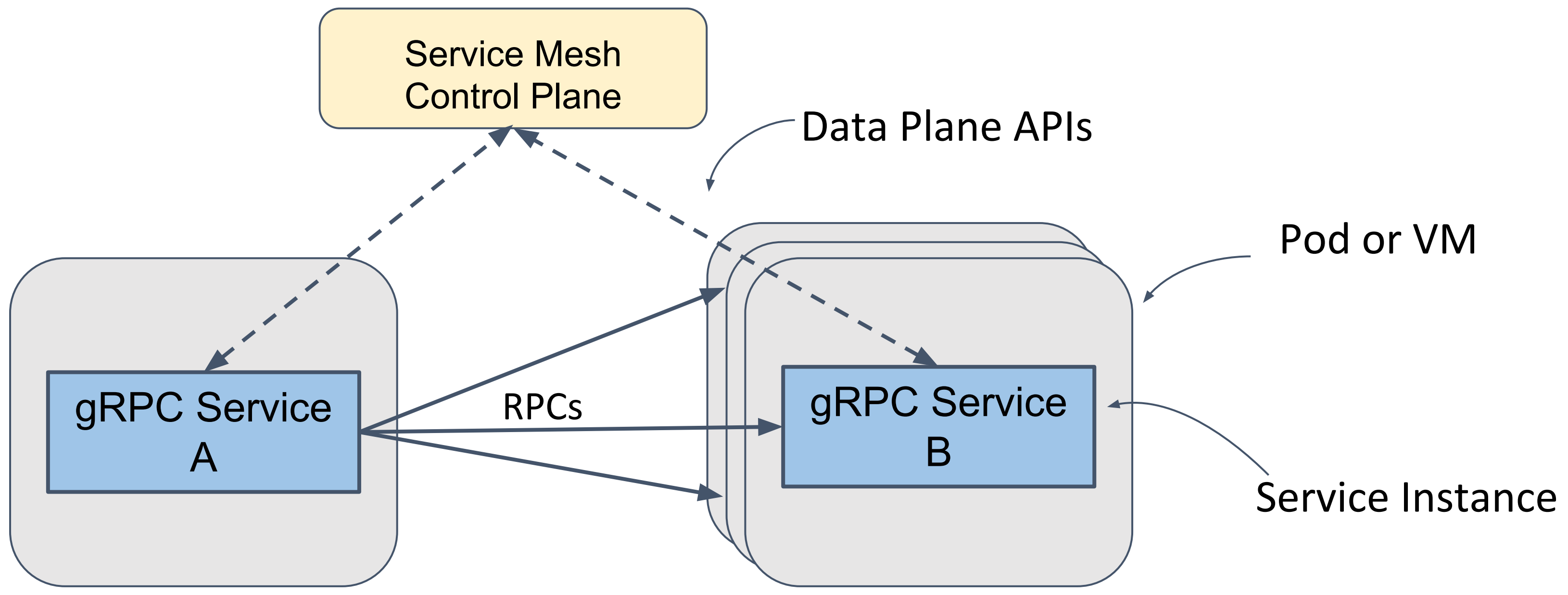
Without Service Mesh Integration



- ➢ Sidecar proxies get service mesh policies from the control plane.
- ➢ gRPC applications use DNS lookup and send requests to the virtual IP of the service.
- ➢ Requests are intercepted by sidecar proxies which apply service mesh policies and route accordingly.

Proxyless gRPC Service Mesh!

gRPC Conf 2020



- ➢ gRPC applications get service mesh policies directly from the control plane.
- ➢ No sidecar proxies. Services talk to each other directly.

Which Service Mesh?

- ➢ Choose the right data plane APIs - APIs between mesh control plane and the proxies.
- ➢ Attributes: Open, extensible, fits gRPC architecture, strong community support and widely used.
  - ○ Works with any control plane that supports such data plane APIs.
  - ○ Helps prevent vendor lock-in.

Winner - xDS APIs - the wildly popular Envoy proxy’s data plane APIs.

- ➢ Istio and several other open source and proprietary service meshes use xDS APIs and Envoy proxy.
- ➢ Evolving into Universal Data Plane APIs.

What is xDS?

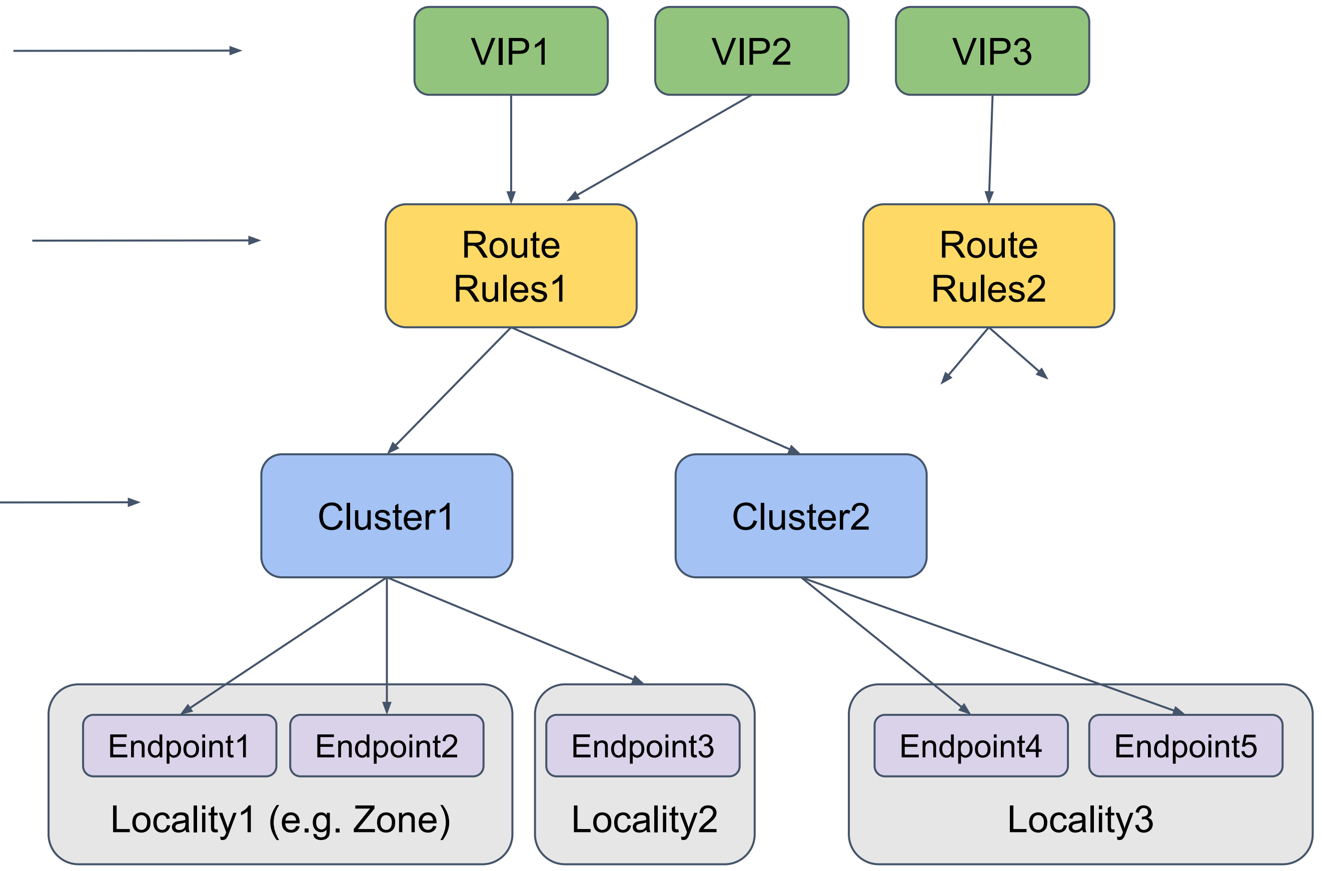
- It’s all about discovering!
- (x)Discovery Service - Listener, Route, Cluster, Endpoint, Health, Secret etc.

Listener Discovery Service
Service VIP(IP:Port) configuration

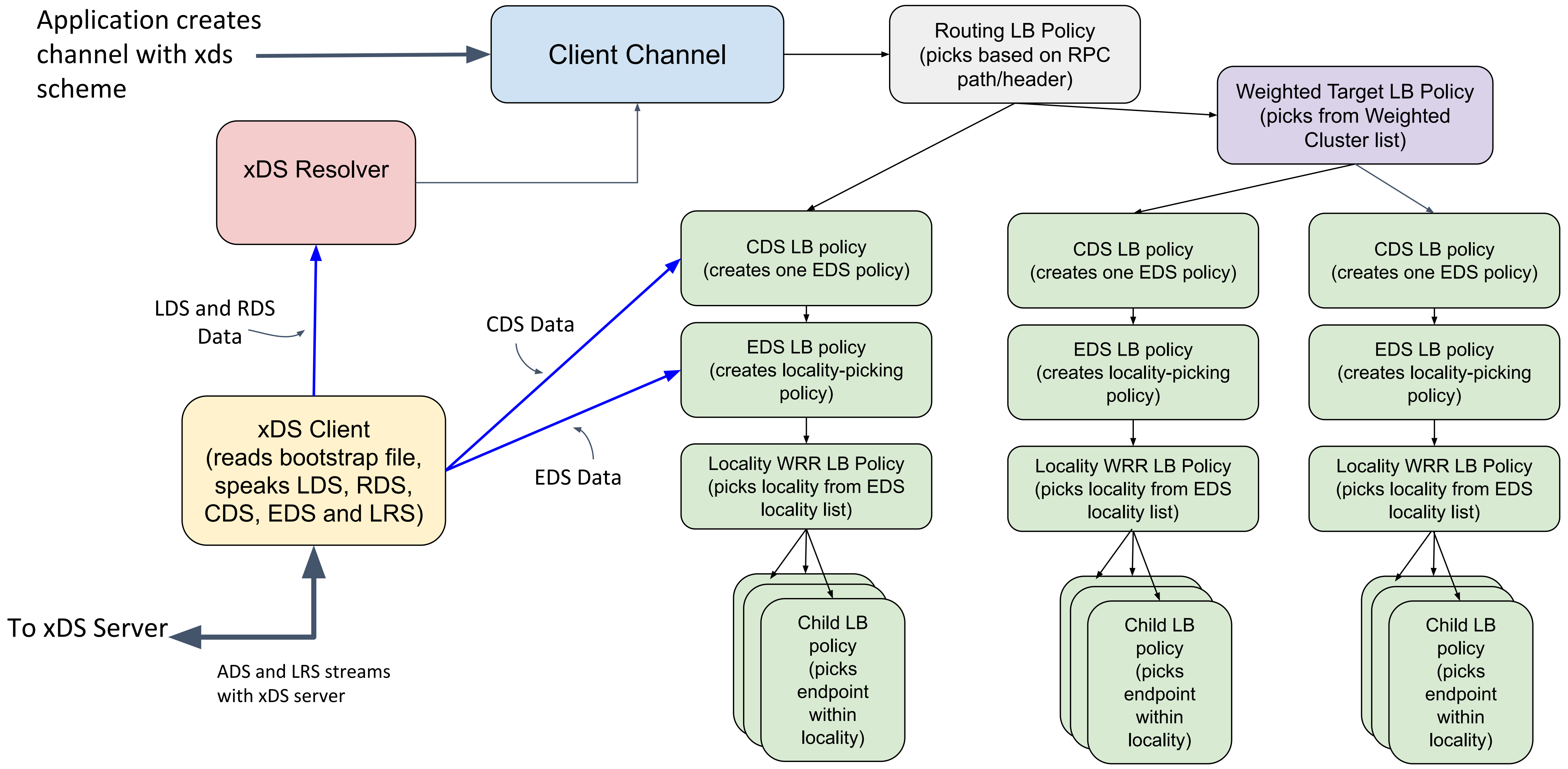
Route Discovery Service
Route matching rules and actions configuration

Cluster Discovery Service
Cluster (Backend Service) configuration

Endpoint Discovery Service
Prioritized and weighted list of localities and endpoints



xDS in gRPC


Application creates channel with xds scheme
Client Channel
Routing LB Policy (picks based on RPC path/header)
Weighted Target LB Policy (picks from Weighted Cluster list)
xDS Resolver
LDS and RDS Data
CDS Data
EDS Data
xDS Client (reads bootstrap file, speaks LDS, RDS, CDS, EDS and LRS)
To xDS Server
ADS and LRS streams with xDS server
CDS LB policy (creates one EDS policy)
EDS LB policy (creates locality-picking policy)
Locality WRR LB Policy (picks locality from EDS locality list)
Child LB policy (picks endpoint within locality)
CDS LB policy (creates one EDS policy)
EDS LB policy (creates locality-picking policy)
Locality WRR LB Policy (picks locality from EDS locality list)
Child LB policy (picks endpoint within locality)
CDS LB policy (creates one EDS policy)
EDS LB policy (creates locality-picking policy)
Locality WRR LB Policy (picks locality from EDS locality list)
Child LB policy (picks endpoint within locality)

Enabling xDS In gRPC

- Build a gRPC channel with ‘xds’ resolver scheme.
  - Example: `ManagedChannelBuilder.forTarget("xds:///foo.myservice")`
- Provide a bootstrap file with xDS server address, credentials and node info.
- Set GRPC_XDS_BOOTSTRAP env variable to the bootstrap file.

That’s it!

- The scheme is per channel - Easy to migrate and mix’n’match proxied and proxyless deployment.

Why Go Proxyless?

gRPC Conf 2020

- ➢ Higher performance, efficiency and scalability.
- ➢ Eases migration of gRPC applications to a service mesh.
- ➢ Simplified network without traffic interception.
- ➢ Avoid potential bottlenecks.
- ➢ No lifecycle management of proxies.
- ➢ Works with containers as well as VMs.
- ➢ End-to-end security (when available) and observability.

Get the most out of your investment in gRPC.

Many cloud scale companies use this model.

On The Flip Side

- Feature gap.
    - But, active development going on.
- Ecosystem around Envoy.
    - But, gRPC has interceptors and OpenCensus integration.
- Must recompile applications.
    - Easy to mix proxy and proxyless deployments.
- Limited languages.
    - C++, Java, Go, Python, PHP, Ruby and C#.
    - Languages wrapping C-Core get it for free.

Current Status

Released in v1.30.0

- ➢ xDS client with LDS, RDS, CDS and EDS.
- ➢ Load reporting via LRS.
- ➢ Weighted locality picking and round robin endpoint LB within the locality.

What's next?

- ➢ Route matching with path and headers field.
- ➢ Traffic splitting between weighted clusters.
- ➢ More features like timeout, circuit breaking, fault injection and retries.
- ➢ gRPC server side xDS integration.
- ➢ Security features like service-to-service mTLS.
- ➢ Migrate to v3 APIs.

Demo

Now a demo using Traffic Director, Google Cloud's managed control plane for service mesh. Traffic Director uses xDS to communicate with gRPC clients.

Resources

- gRFC on xDS load balancing design
- gRFC on xDS traffic splitting and routing design
- xDS features in gRPC by release
- Envoy xDS APIs
- Universal Data Plane APIs
- Data plane vs. control plane
- Concepts and terminology
- Traffic Director

Questions or feedback?

gRPC
Conf 2020