An Introduction to Service Meshes

Platform Labs Seminar, Nov 13, 2018
Zack Butcher
Agenda

- The Motivating Problem
- Shape of the Solution Space
- Dive Into Istio
The Problem

- Overarching goal: provide value to customers (so they pay you)
The Problem

- Overarching goal: provide value to customers (so they pay you)
- Many ways to achieve value
The Problem

- Overarching goal: provide value to customers (so they pay you)
- Many ways to achieve value

Value for Customers

- Features
- Pricing
- Support
The Problem

- Overarching goal: provide value to customers (so they pay you)
- Many ways to achieve value
- Get features to users
  - Iterate faster
  - Provision infra to serve demand
The Problem

- Get features to users
  - Iterate faster
  - Provision infra to serve demand
The Problem

- Get features to users
  - Iterate faster
  - Provision infra to serve demand
- New architectures and deployment methods => new requirements

Diagram:
- Development Velocity
- Scalability
- Microservice Architecture
- Orchestrators (Mesos, K8s)
- Observability
- Management
- Service Identity
- Connectivity
The Problem

- New architectures and deployment methods => new requirements
- “Modern Architecture”
The Problem

- New architectures and deployment methods => new requirements
- “Modern Architecture”
- Service Mesh feature space

Diagram:
- Development Velocity
- Scalability
- Microservice Architecture
- Orchestrators (Mesos, K8s)
- Observability
- Management
- Service Identity
- Connectivity
IT’s shift to a modern distributed architecture has left enterprises unable to connect, monitor, manage, or secure their services in a consistent way.
The Problem

modern distributed architecture

container based services
deployed into dynamic environments
composed via the network
The Problem

IT’s shift to a modern distributed architecture has left enterprises unable to connect, monitor, manage, or secure their services in a consistent way.
Connect

Get the network out of the application.

- Service Discovery
- Resiliency
  - retry, circuit breaking, timeouts, lame ducking, etc.
- (Client Side) Load Balancing
Monitor

Understand what’s actually happening in your deployment.

- Metrics
- Logs
- Tracing
Control where and how requests flow, and which requests are allowed.

- Fine grained traffic control
  - L7, not L4!
  - Route by headers, destination or source ID, etc
- Policy on requests
  - Authn/z, rate limiting, arbitrary policy based on L7 request metadata
Elevate security out of the network.

- (L7) Workload Identity
  - IP:port is not an identity
  - Reachability != Authorization
- Service-to-Service Authn/z
These aren’t new Problems!

- Enterprise Service Bus
  - Handles similar concerns
  - Architecture mean scaling is hard
  - Tend to introduce unintended coupling
  - Business logic wound up in the bus
- “Smart endpoints, dumb pipes”
These aren’t new Problems!

- Libraries and Frameworks
  - Ad hoc; per organization, language
    - e.g. Spring, Hystrix, Finagle; Stubby/gRPC, Thrift, Twirp
  - Expensive
  - Invasive
  - Hard to change due to deployment model
The goal of a service mesh is to move this functionality out of the application so application developers don’t need to worry about it.

- Consistency across the fleet
- Centralized control
- Fast to change (update config to affect change, not redeploy)
- Language Agnostic
Istio is a platform to connect, monitor, manage, and secure services consistently.
How Istio Works

A call B
1. Deploy a proxy (Envoy) beside your application (“sidecar deployment”)
2. Deploy Galley to configure the rest of the Istio control plane
3. Deploy Pilot to configure the sidecars
How Istio Works

3. Deploy Mixer to get telemetry and enforce policy

![Diagram showing how Istio works with Mixer, Envoy, Galley, and Pilot]
4. Deploy Citadel to assign identities and enable secure communication
How Istio Works
How Istio Works

Envoy

A

Envoy

B

Galley

Pilot

Mixer

Citadel
How Istio Works
How Istio Works

- A
  - Envoy

- B
  - Envoy

- Galley
- Pilot
- Mixer
- Citadel

Policy
How Istio Works

A

Envoy

B

Envoy

Galley
Pilot
Mixer
Citadel
How Istio Works

A

B

Envoy

Response

Galley  Pilot  Mixer  Citadel
How Istio Works

A
Envoy

B
Envoy

Galley
Pilot
Mixer
Citadel

Telemetry
**Architecture**

**Pilot**: Control plane to configure and push service communication policies.

**Envoy**: Network proxy to intercept communication and apply policies.

**Mixer**: Policy enforcement with a flexible plugin model for providers for a policy.

**Citadel**: Service-to-service auth[n,z] using mutual TLS, with built-in identity and credential management.

**Galley**: Configuration validation, distribution*

*not yet, but upcoming in 1.1
Closing Thoughts

● Clear parallels between SDN (L2/L3) and service mesh (L7)
● Clear parallels between features provided by TCP/IP at network layer and service mesh at application layer
● Both are part of the march towards a higher level of abstraction across the industry
Thanks!

Istio:
● https://istio.io
● https://github.com/istio
● @IstioMesh on Twitter

Me: @ZackButcher on Twitter