SERVICE MESH

Service Mesh: The Next Step in Networking for Modern Applications

Bruce Davie
Posted July 23, 2019

0 Comments

By Bruce Davie, CTO, Asia Pacific & Japan

What's New in the World of Networking

As I'm currently preparing my breakout session for VMworld 2019, I've been spending plenty of time looking into what's new in the world of networking. A lot of what's currently happening in networking is driven by the requirements of modern applications, and in that context it's hard to miss the rise of service mesh. I see service mesh as a novel approach to meeting the networking needs of applications, although there is rather more to it than just networking.

There are about a dozen talks at VMworld this year that either focus on service mesh or at least touch on it – including mine – so I thought it would be timely to comment on why I think this technology has appeared and what it means for networking.

To be clear, there are a lot of different ways to implement a service mesh today, of which Istio – an open-source project started at Google – is probably the most well-known. Indeed some people use Istio as a synonym for service mesh, but the broader use of the term rather than a particular implementation is my focus here.

The Emergence of Service Mesh

My first exposure to the concepts of service mesh came about two years ago during the Future:NET conference. Thomas Graf gave a presentation on the Cilium project, and to motivate the work he showed why traditional network-level security isn't sufficient for securing the communication paths among microservices. His example showed how one microservice could be connected to another with all the security enforced within the network, using firewalls for example. The problem with such a security model is that the entire API of a microservice is exposed to any other service that is allowed to connect to it. Given that the API is most likely implemented with TLS (Transport Layer Security) to authenticate endpoints and encrypt the traffic, there's really no way for network-layer devices to enforce any sort of policy at the API layer.

One might conclude that the obvious answer is to implement this sort of control inside the application, but that has its own set of drawbacks. As summarised by Chenxi Wang, this would lead to lots of redundant code, implemented by many teams in different languages – with

implications for efficiency, performance, and security. Furthermore, there's value in exposing a common set of controls on this application-level communications mesh, so that meaningful policies can be set in a consistent way across all microservices.

At last year's Future:NET, Louis Ryan, one of the Istio project leaders, gave another memorable talk on service mesh, including some interesting reasons for its adoption. Imagine you wanted to add service mesh capabilities to an existing piece of code. As Louis pointed out, a good analogy to opening up an old piece of code is opening up an old can of *surströmming* – a sort of fermented fish. YouTube is full of cautionary tales on this topic. The point here is that there are real advantages to adding service mesh capabilities to your applications non-intrusively.

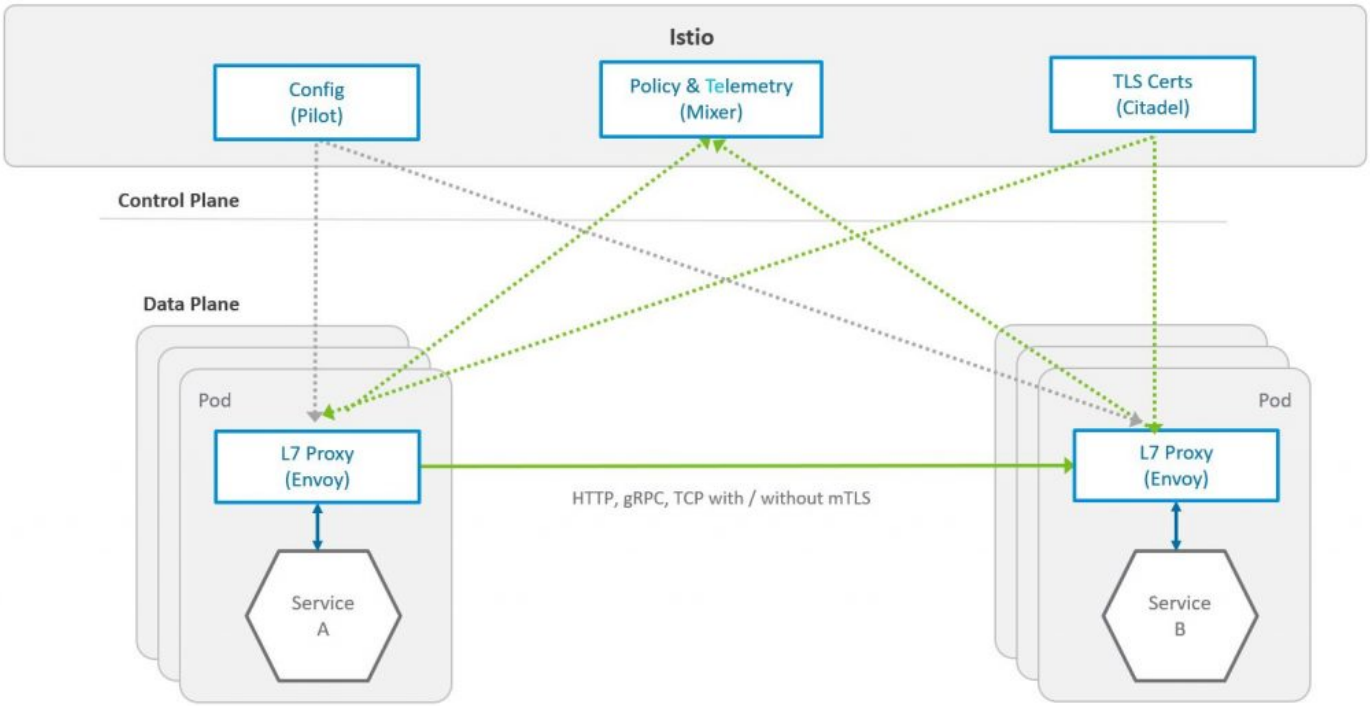
Problems Service Mesh Solves

Service mesh is the emerging solution to the preceding problems. It provides a uniform way to:

- Implement application-aware security policy
- Traffic management, monitoring, and load balancing
- A range of other useful functions among communicating microservices

Service Mesh and Network Virtualisation

There are considerable similarities between service meshes and network virtualisation overlays such as that implemented by NSX Data Center. Like NSX, a service mesh has a centralised control plane and a distributed data plane (see figure). The data plane component of a service mesh is called a *sidecar proxy*, of which Envoy is the most well-known example. Istio is a control plane that integrates with Envoy.



There are some key differences between a network virtualisation system like NSX Data Center and a service mesh (explored in detail here) — especially how close they sit to the application. NSX Data Center processes network traffic as it enters and leaves a container or VM – which means that it's not close enough to the application for things like API-granularity access control. Conversely, the sidecar proxy of a service mesh can be inserted right into the same Kubernetes pod in which the microservice is running, for example, so it can provide application-level services. Service mesh recognizes that the end point of a communication is a service, not just a machine or a device.

As service mesh becomes more mainstream, I expect that plenty of enterprises will be looking for easy ways to introduce service mesh capabilities to their infrastructure. NSX Service Mesh is a new product that aims to meet the needs of those enterprises. Because there is so much rapid innovation in the open source world around service mesh, we expect open source efforts – like Istio and Envoy – to be central to our work, just as Open vSwitch has been central to our development of network virtualisation. Simplifying adoption of these open source projects will be key for enterprises, and an area of focus for us.

Service Mesh Resources

There are several blog posts already about NSX Service Mesh:

- Introducing NSX Service Mesh
- How Istio, NSX Service Mesh, and NSX Data Center Fit Together
- This is also well worth a read: "Which Service Mesh Should I Use?"

I'm looking forward to spending a bit more time on this topic during my session at VMworld, and if all goes to plan we'll have a live demo as well. I hope to see you in San Francisco.