KubeCon CloudNativeCon

THE LINUX FOUNDATION

OPEN SOURCE SUMMIT

AI_dev

Open Source GenAI & ML Summit

China 2024

KubeCon

CloudNativeCon

THE LINUX FOUNDATION

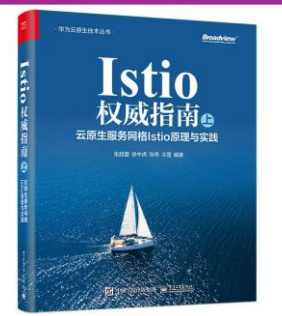
China 2024

A New Choice for Istio Data Plane:

Reshape Sidecarless ServiceMesh with eBPF and Programmable kernel

About Me

Zhonghu Xu
Huawei Cloud



KMESH

- CNCF TAG Network Tech Lead
- Istio Steering Committee Member
- Istio, Kmesh Maintainer
- Kubernetes Member

Agenda

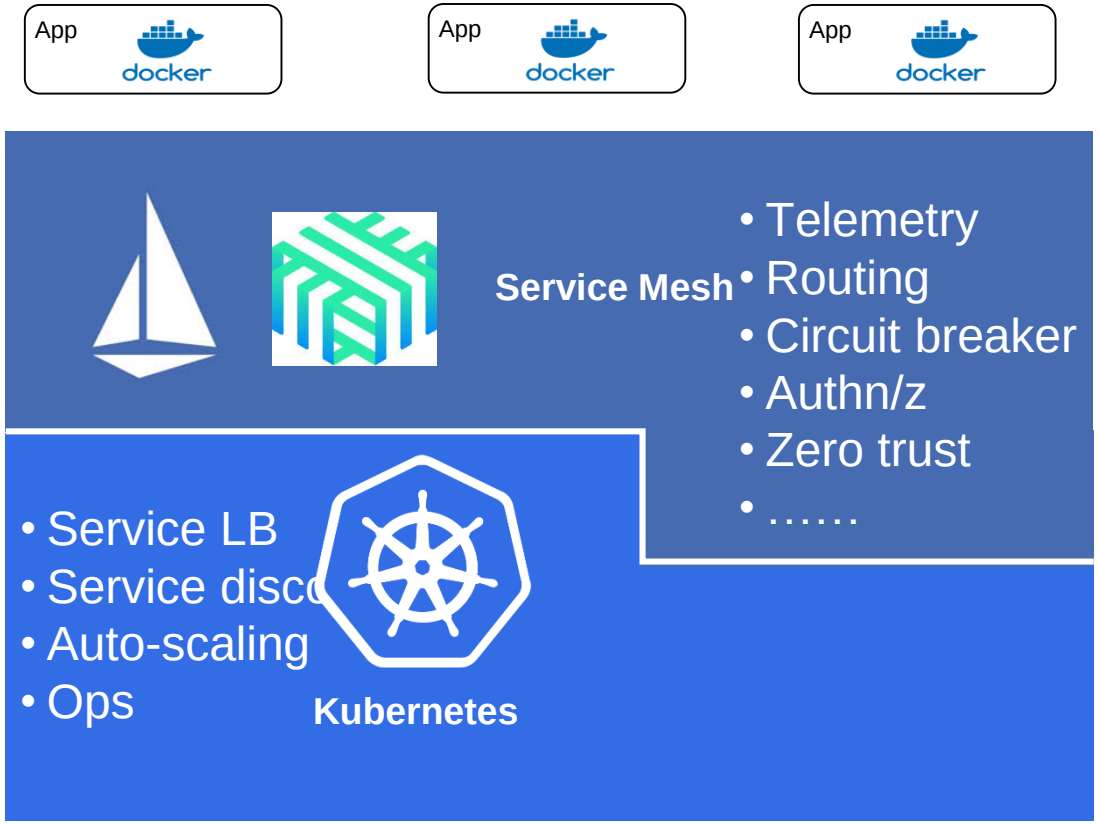
- **Service Mesh Background**
- **Why Kmesh**
- **Kmesh Key Features**
- **Future of Service Mesh**

Agenda

- **Service Mesh Background**
- Why Kmesh
- Kmesh Key Features
- Future of Service Mesh

What is Service Mesh

Service Mesh is an infrastructure layer that handles communication between services. It is commonly used in conjunction with microservices architecture to provide features such as service discovery, load balancing, circuit breaking, monitoring, tracing, and security.



Core Concepts:

1.Non-intrusive Sidecar Injection, injecting a sidecar container into the application's pod without affecting the application itself. It is agnostic to the application's programming language.

2.Declarative API: Service Mesh exposes a northbound API using Kubernetes Custom Resource Definitions (CRDs). This API is fully declarative and standardized.。

3.xDS Dynamic Config Update: The data plane and control plane communicate using the xDS gRPC, supporting sub-pub updates。

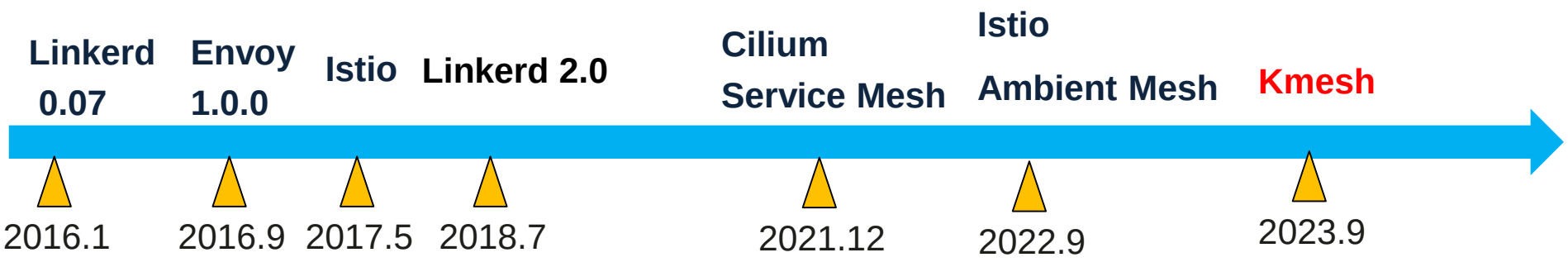
Key Features:

1.Service & Traffic Manage: Circuit breaking, fault injection, rich load balancing algorithms, rate limiting, health checks, canary releases, blue-green deployments, etc.

2.Observability: Provides application-level monitoring, distributed call chains, access logs, and more.

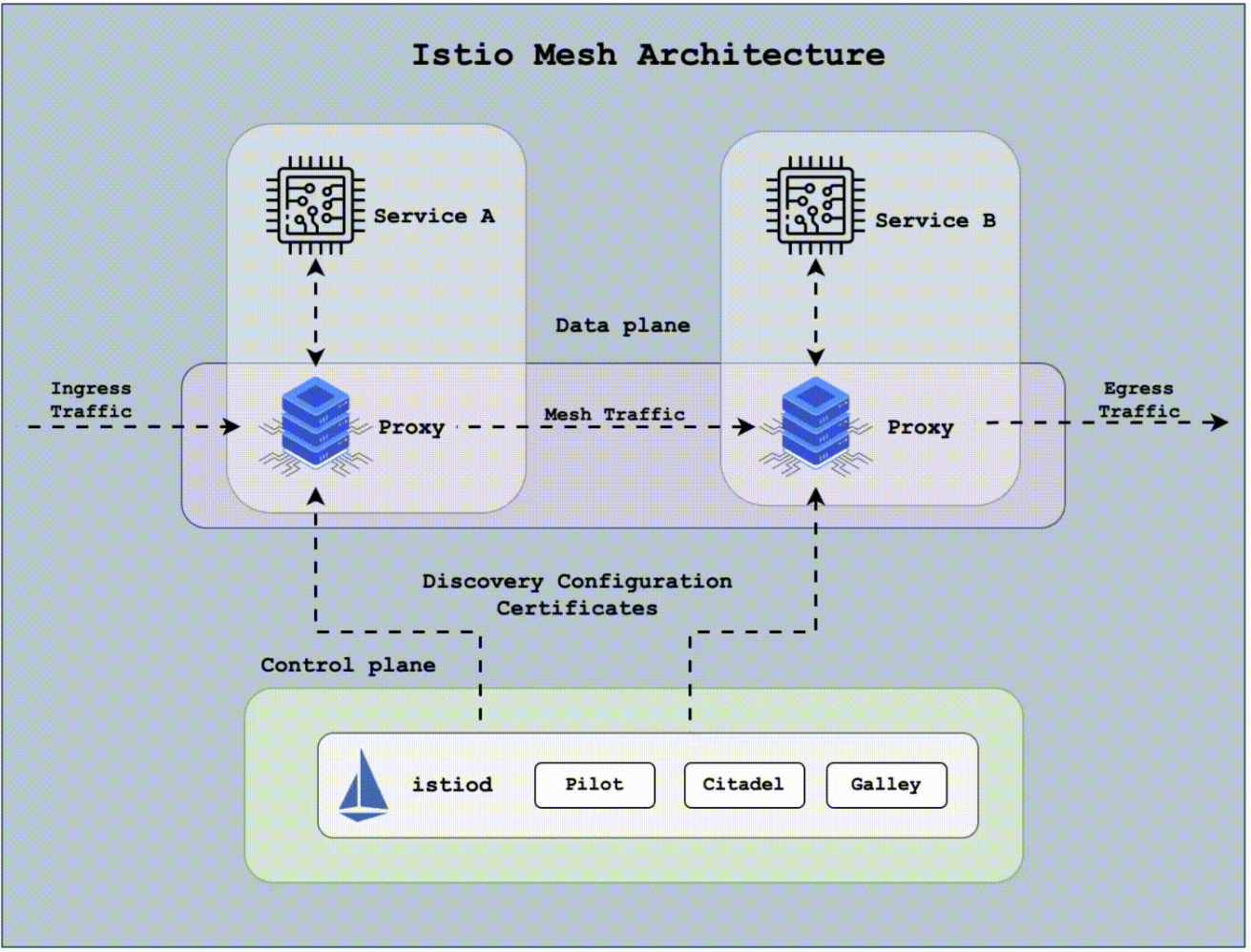
3.Secure Encryption: Helps enterprises run applications in zero-trust networks through security measures such as mTLS (Mutual Transport Layer Security), authentication, authorization,

Retrospect of SM History



1. Service Mesh evolved from sidecar to sidecarless
2. Performance and resource overhead are main concerns
3. Istio is not the only choice
4. L4 traffic management offloading is a trend
5. L7 traffic management can diversifies

Sidecar Model



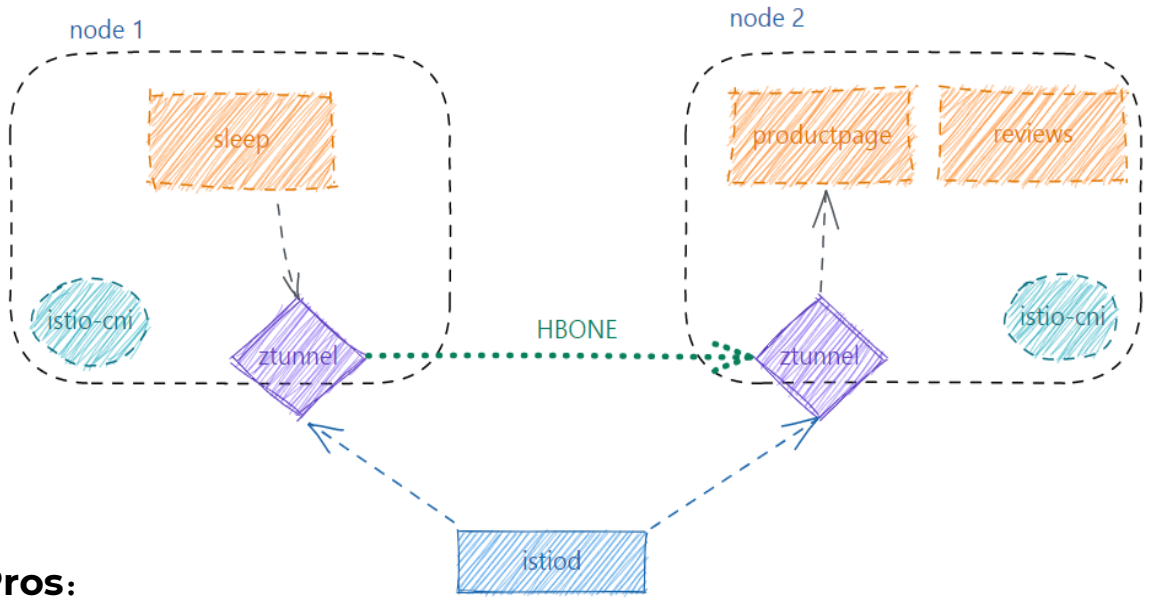
Characteristics:

1. App and Sidecar belongs to a network namespace
2. Inbound/Outbound interceped through iptables
3. Default mTLS，all traffic encrypted
4. Accesslog, metrics, traces generated by Sidecar

Cons:

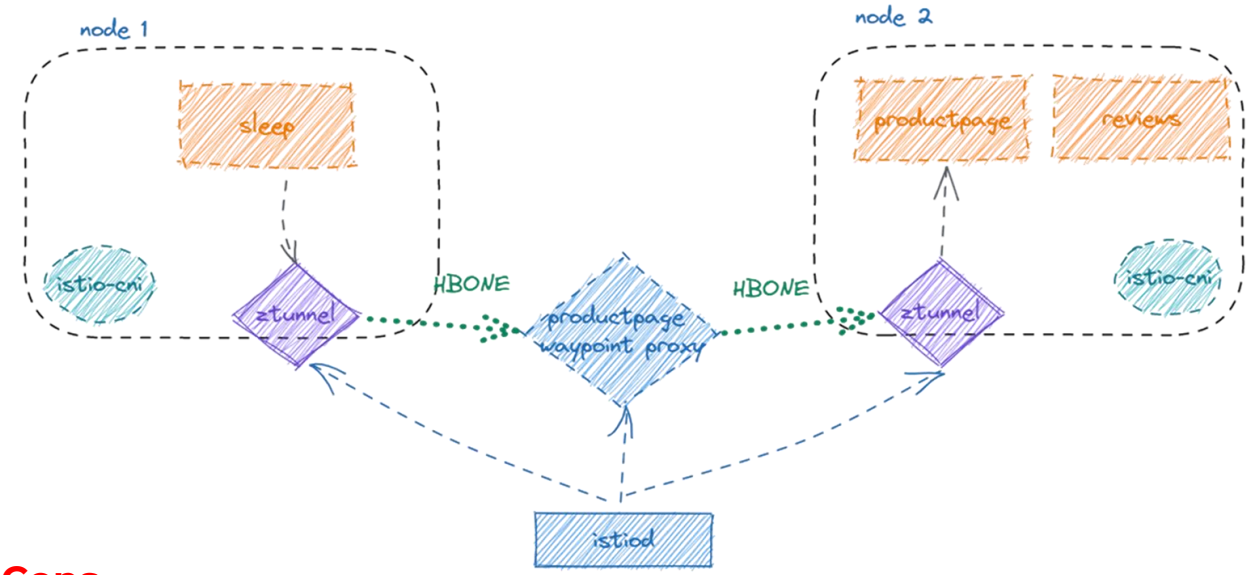
1. Application lifecycle bound with Sidecar，
2. Sidecar resource overhead
3. Connection numbers increased on the path，so request latency is neglectable

Sidecarless：Istio Ambient



Pros:

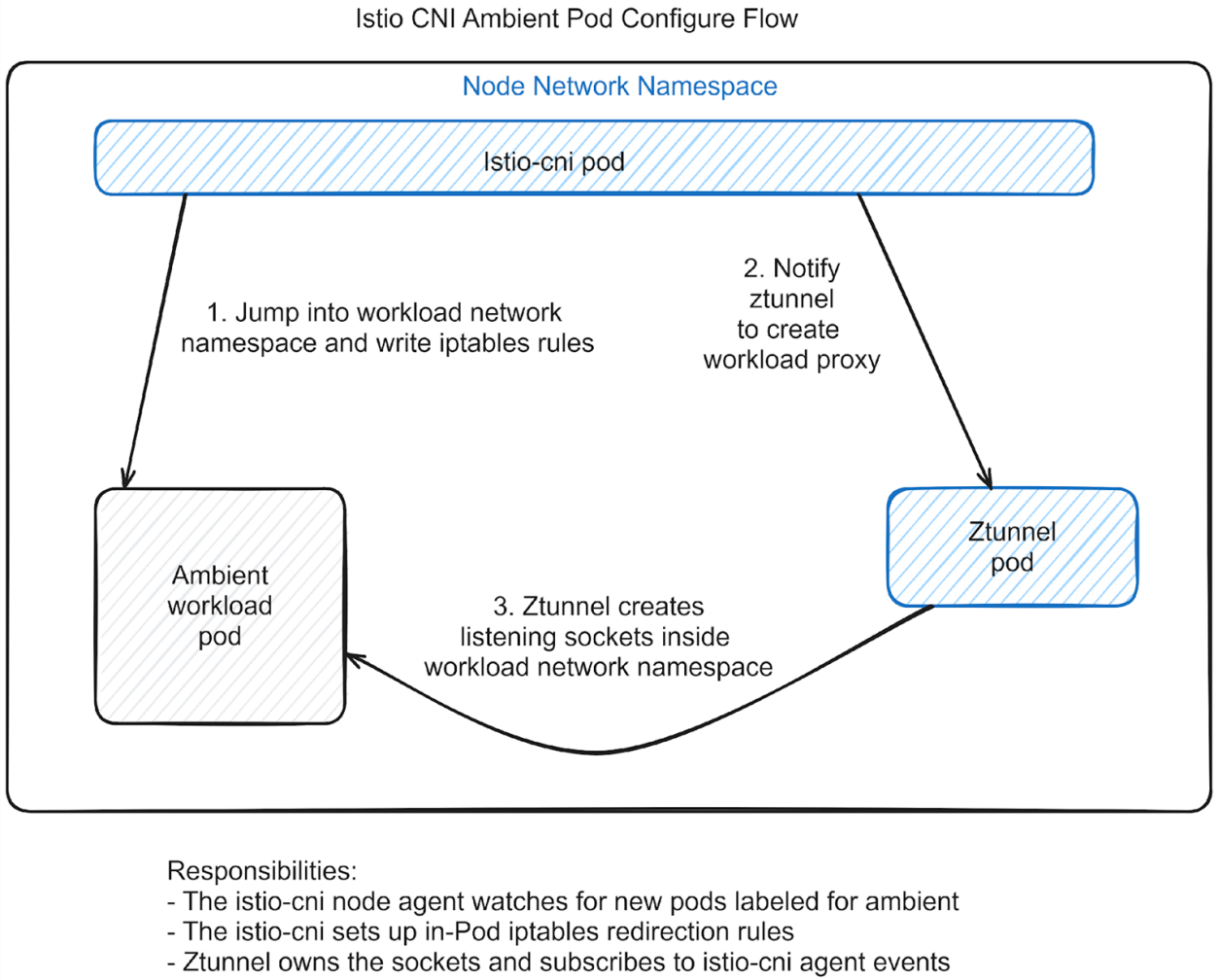
- Traffic management offload from sidecar to ztunnel + waypoint
- Slicing management：more flexible
- Ztunnel is responsible for L4 forwarding， auth，light-weight
- Waypoint is for L7 management，automatically deoloyed at a granularity of namespace、service or pod
- Resource overhead is low compared with sidecar



Cons:

- Ztunnel is a kind of userspace proxy per node，interrupting service communication during upgrade or restart
- Increased tcp connection number from one to four
- App traffic interception complex, complex usually means unreliability

Istio Ambient – traffic interception



https://istio.io/latest/docs/ambient/architecture/traffic-redirection/#istios-in-pod-traffic-redirection-model

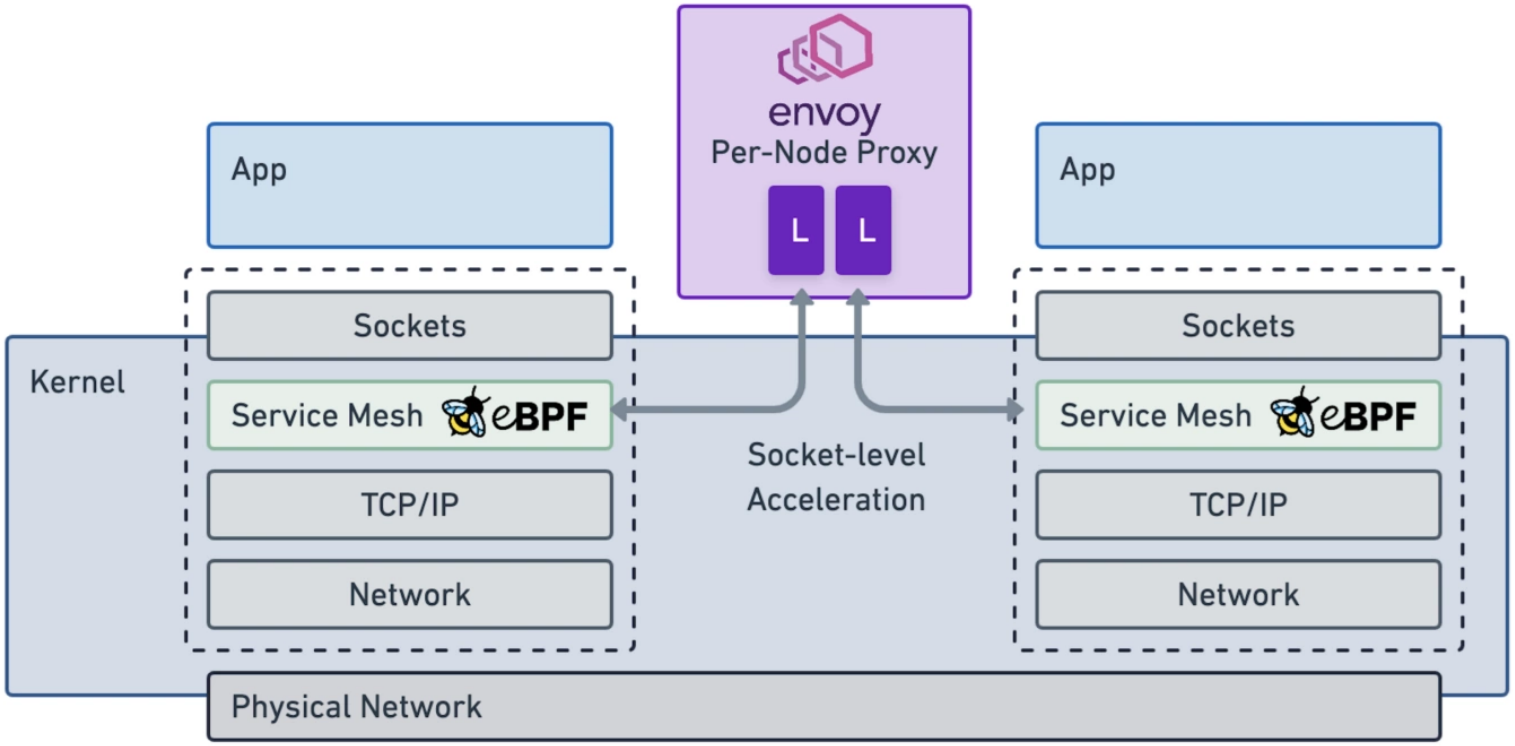
In Pod Redirection:

to be compatible with main CNI, like cilium calico

Cons:

- Istio-cni and ztunnel works in collaboration and are stateful, so if one works abnormally, then the pod can not even startup
- Ztunnel restart/upgrade is a breaking behavior, all node traffic is broken

Sidecarless: Cilium Service Mesh



Key Features：

1.Sidecarless：L3、L4 management based on eBPF, L7 traffic management is based on shared per-node proxy

2.eBPF high performance，accelerate with Sockmap, interception without going theough tcp/ip stack

3.Supported API: Gateway API， CiliumNetworkPolicy， CiliumEnvoyConfig， CiliumClusterwideEnvoyConfig

Cons：

1.Single point of failure, blast radius is node level same as istio ambient

2.Config API is complex and not friendly to use.

3.Tightly coupled with cilium CNI

Agenda

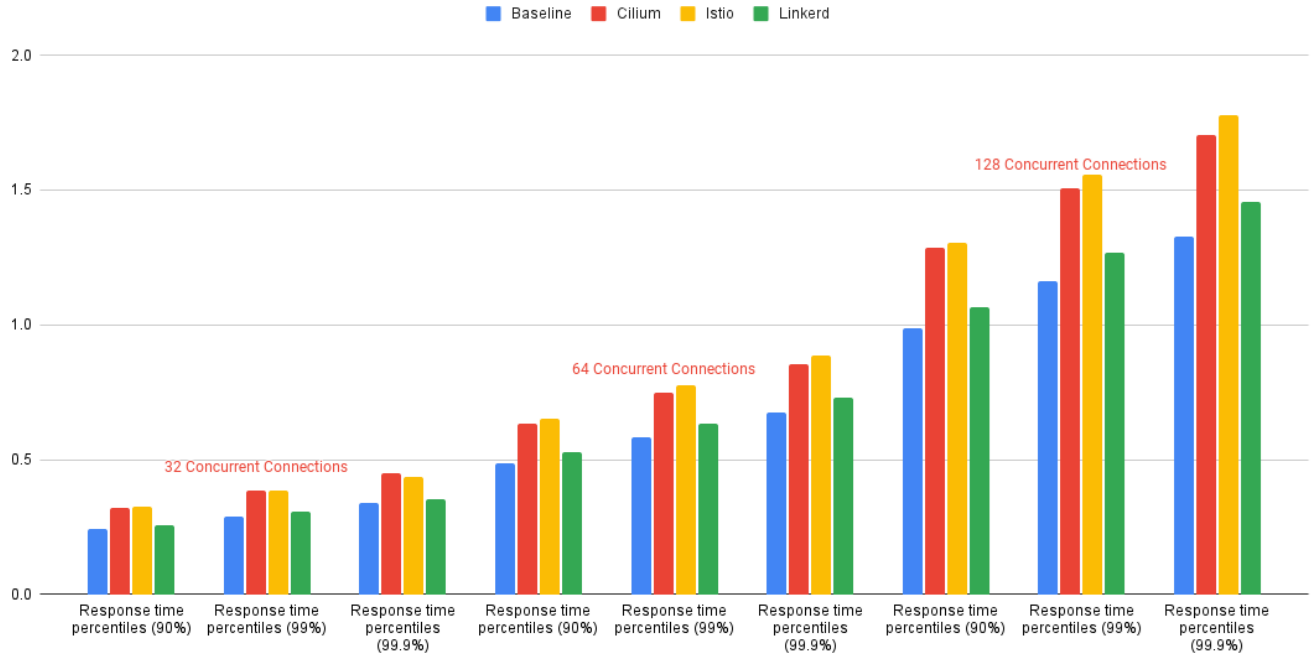
- Service Mesh Background
- **Why Kmesh**
- Kmesh Key Features
- Future of Service Mesh

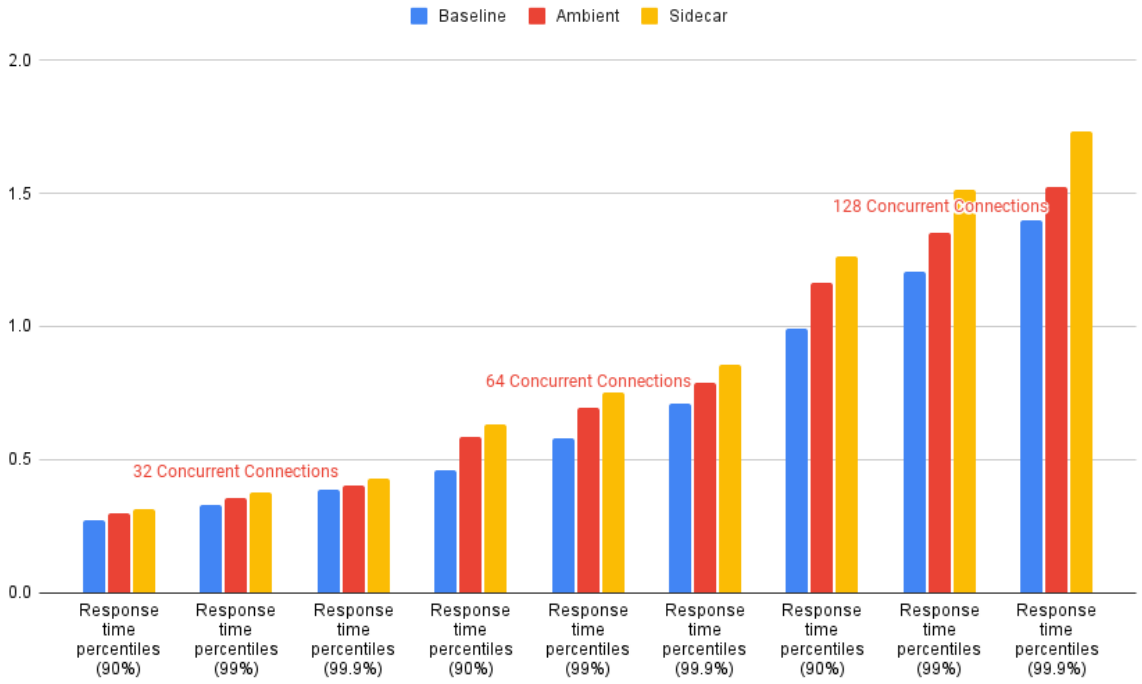
What is the ideal service mesh infra

Our thinking:

- Service mesh should return back to the essence of cloud native requirements, achieving an **application-transparent, high-performance, low-noise** service mesh infrastructure.
- High-reliability: in flight traffic should not be influenced during infra upgrade
- Do not occupy users' cpu/memory resources, or as less as possible
- Flexible deploy mode, some users may want managed proxy, some others may want auto-manage within local cluster, some may even want customized remote proxy.

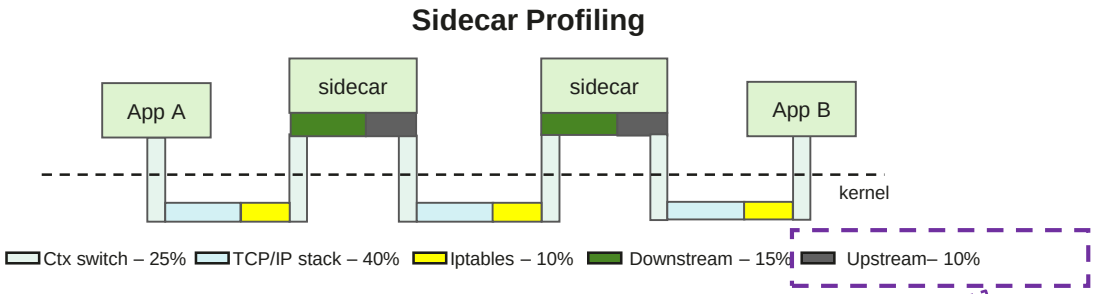
Performance Analysis

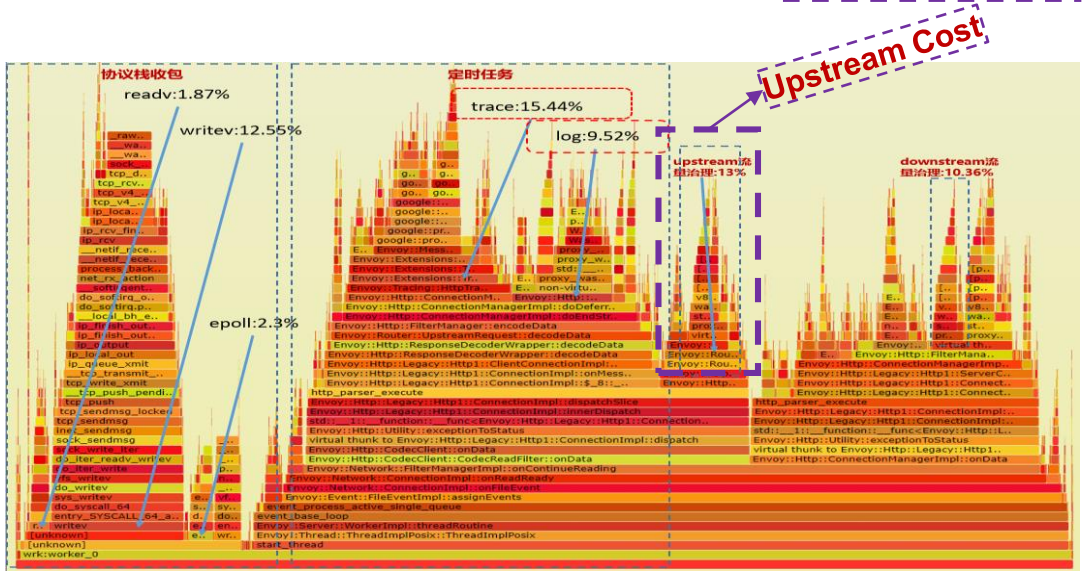




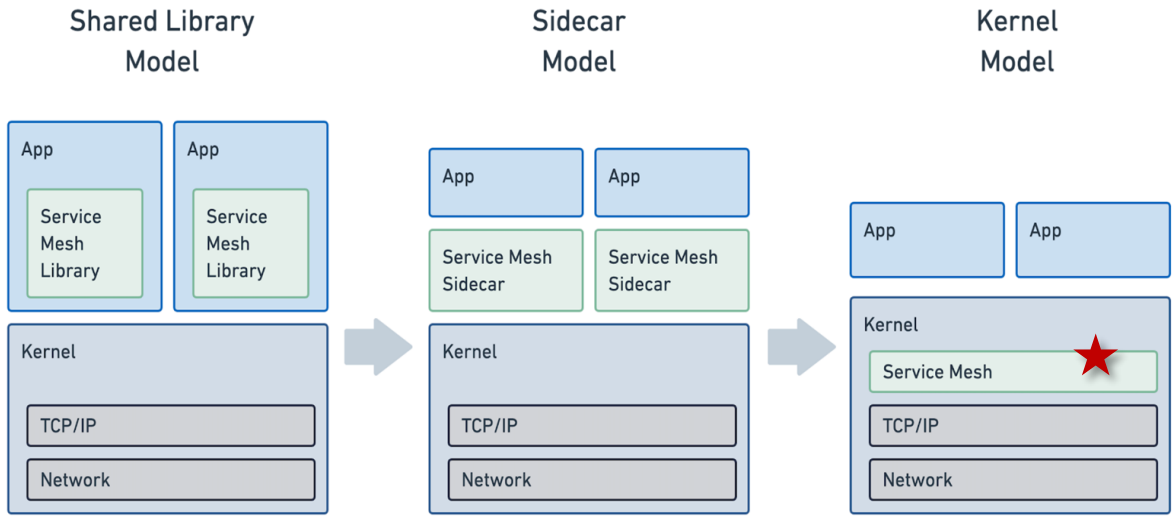
1. Istio is slower than the baseline by 25-35%.
2. Istio Ambient is slower than the baseline by 15%
3. Cilium slower then the baseline by 20-30%.
4. Linkerd is slower then the baseline by 5-10%

Istio sidecar performance profiling

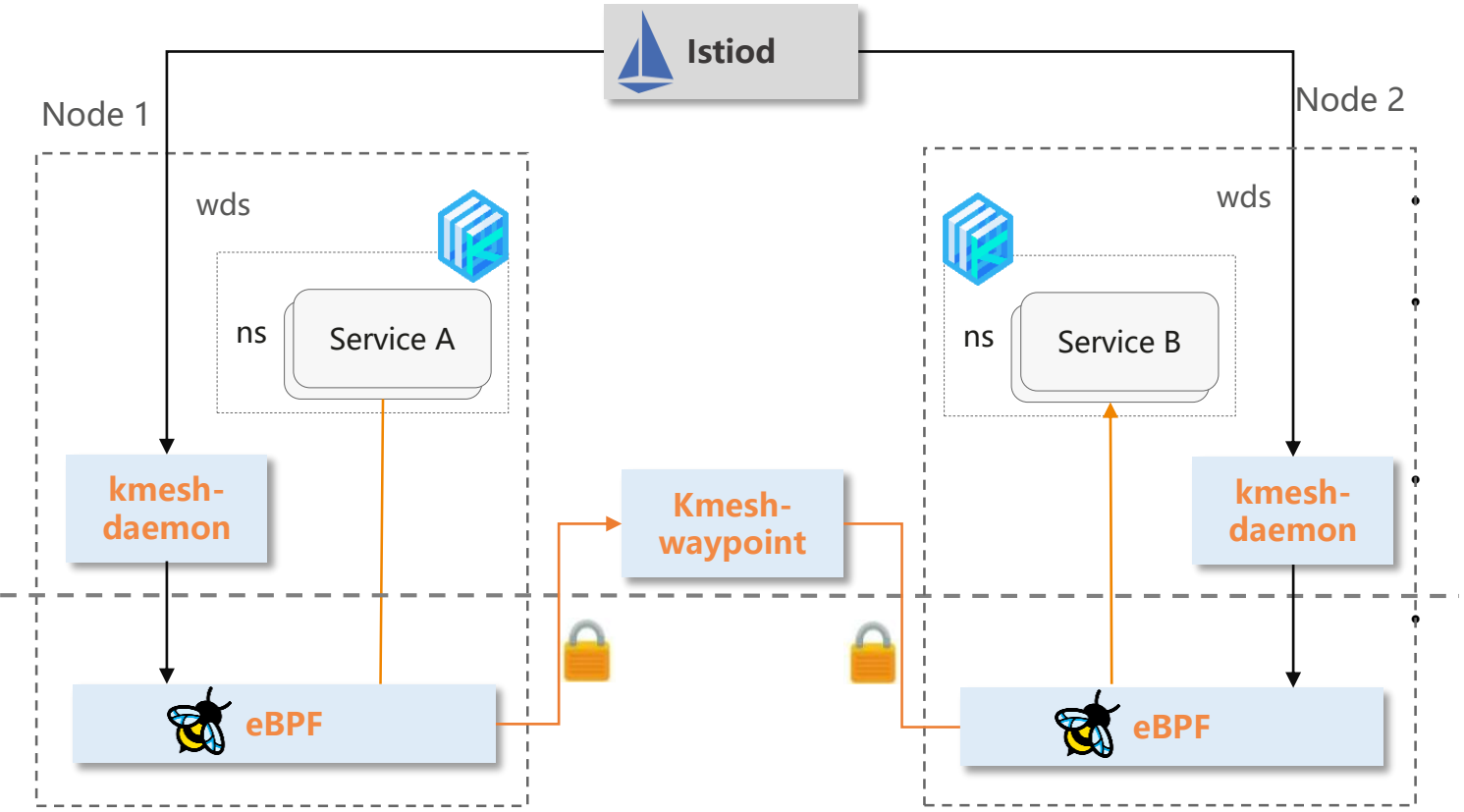




网格耗时分布可以看出：**sidecar架构引入大量时延开销，流量编排只占网格开销的10%**，大部分开销在数据拷贝、多出两次的建链通信、上下文切换调度等。



Kmesh: kernel-native sidecarless



- **High Performance**

  Kernel-native L7 management，reducing forwarding latency by 60%

  Dual Engine L7 management, reducing latency by 30% compared with Ambient

  Application bootstrap improved by 40%

- **Low overhead**

  No sidecar，resource consumption down by 70%；

- **High availability**

  Kernel traffic management does not terminate connections, and Kmesh component upgrade and restart do not effect existing service connections；

- **Security isolation**

  BPF-based VM security and cgroup-level governance isolation are supported；

- **Flexible management mode**

  In addition to the kernel native management, Kmesh also supports slicing L4 and L7 management for isolation. The kernel eBPF program and waypoint component process L4 and L7 traffic respectively

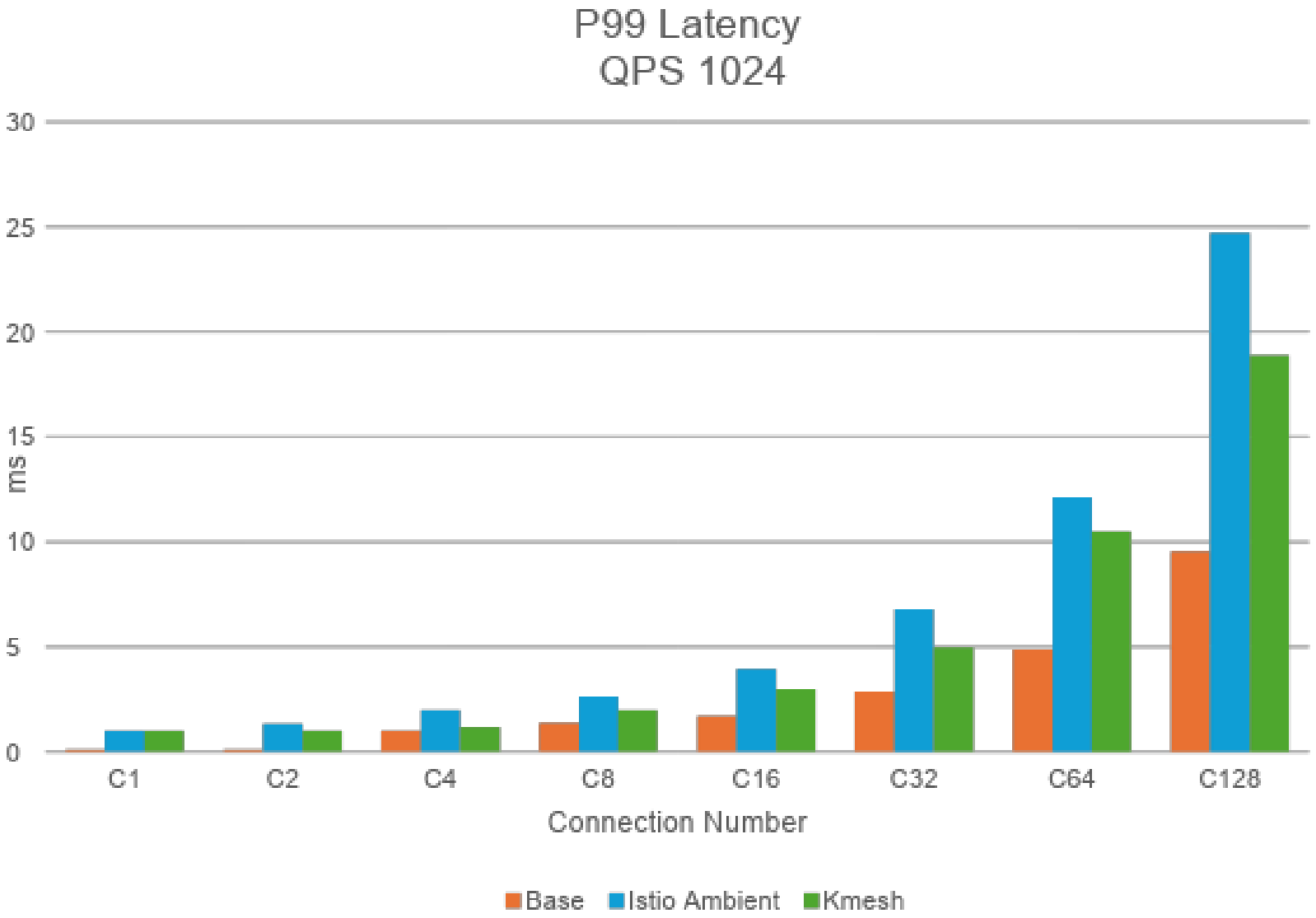
- **Seamless compatibility**

  seamlessly integrate with xDS protocol. Support both Istio APIs and Gateway APIs ；

Performance Comparation With Istio

Kernel-native: HTTP response latency almost same as baseline

Performance Comp with Ambient



- Kmesh is faster than ambient mesh by about 25-30%
- Both are slower than base line with L7 processing

Advantages of Kmesh

1. Offloading traffic management into kernel, application pod is totally decoupled.
2. No connection termination, reducing connection numbers on the flow path

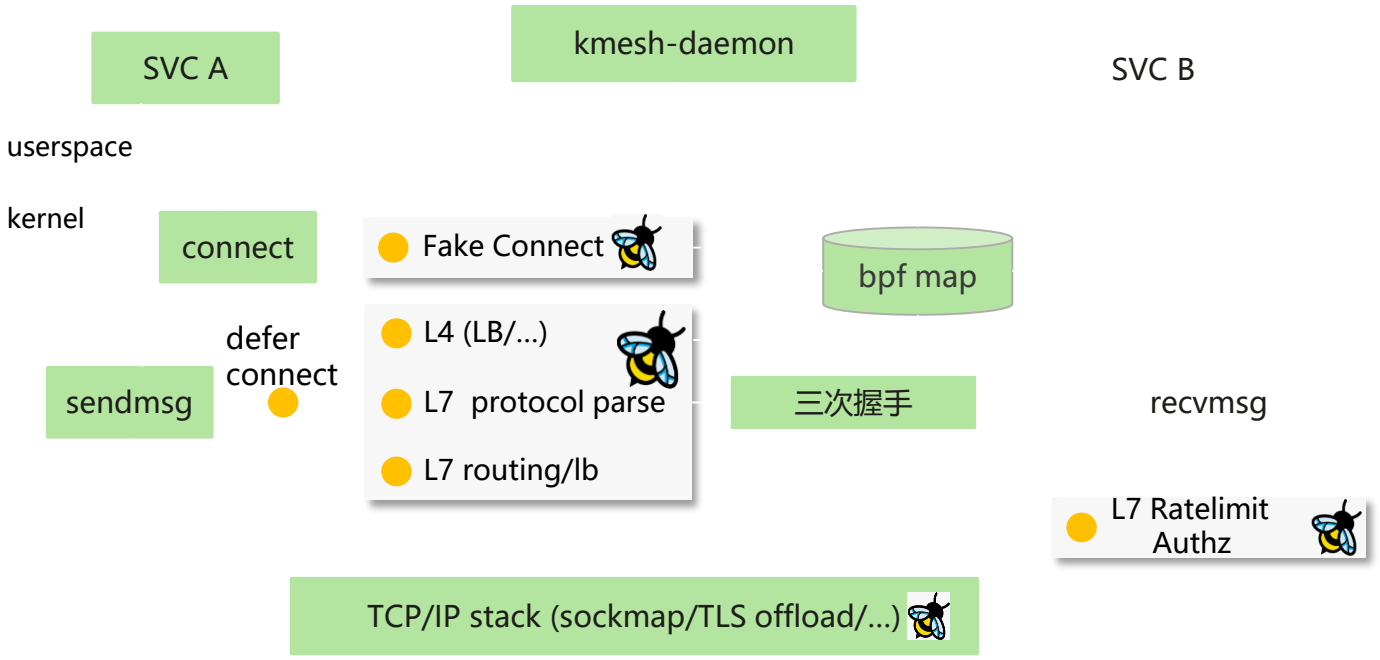
Kmesh Kernel Native	Kmesh Dual Engine	Istio Sidecar	Istio Ambient
1	2	3	4

3. High performance, low latency, low resource consumption
4. High-reliability: in flight traffic should not be influenced during infra upgrade
5. No occupying users' cpu/memory resources, only a daemon set used to control bpf prog

Agenda

- Service Mesh Background
- Why Kmesh
- **Kmesh Key Features**
- Future of Service Mesh

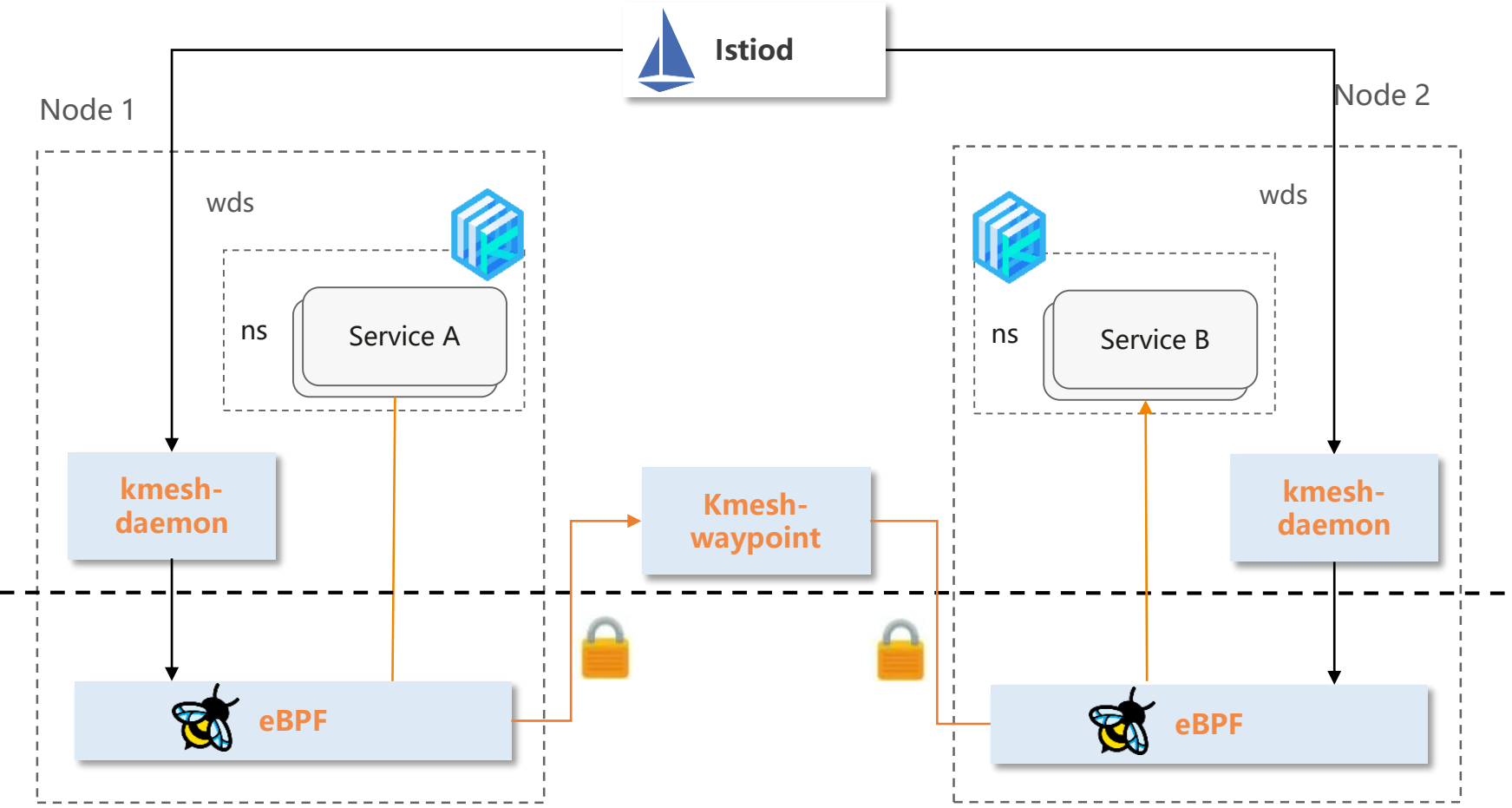
Kernel Native L7 Management



Traffic Orchestration：

- Based on Fake TCP Connect 、Defer Connect，implement traffic manage fundamental；
- Based on ko, implement HTTP protocol parse
- Based on ebpf，implement L7 traffic routing， advanced loadbalancing；

Dual Engine: Slicing the layers

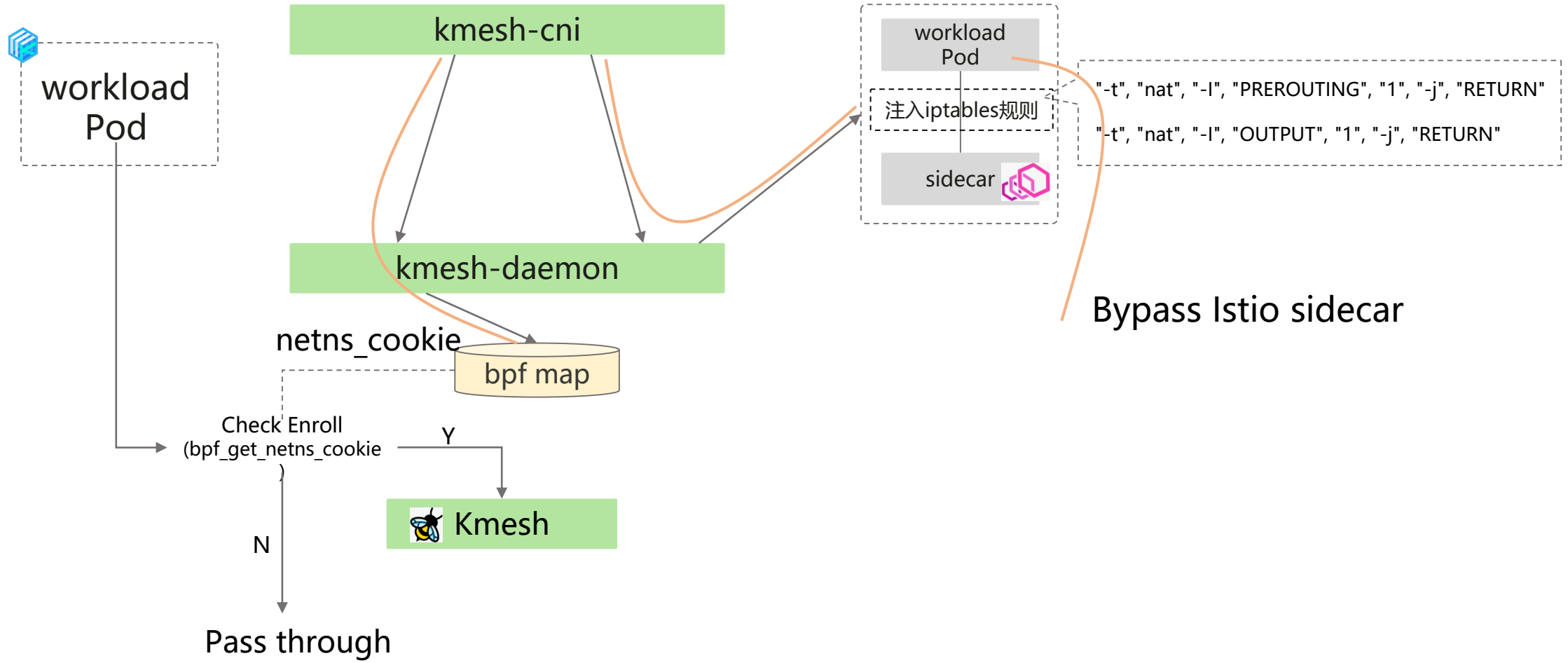


- **L4 Processing:**

L4 redirecting, authorization, TCP metrics and access log.

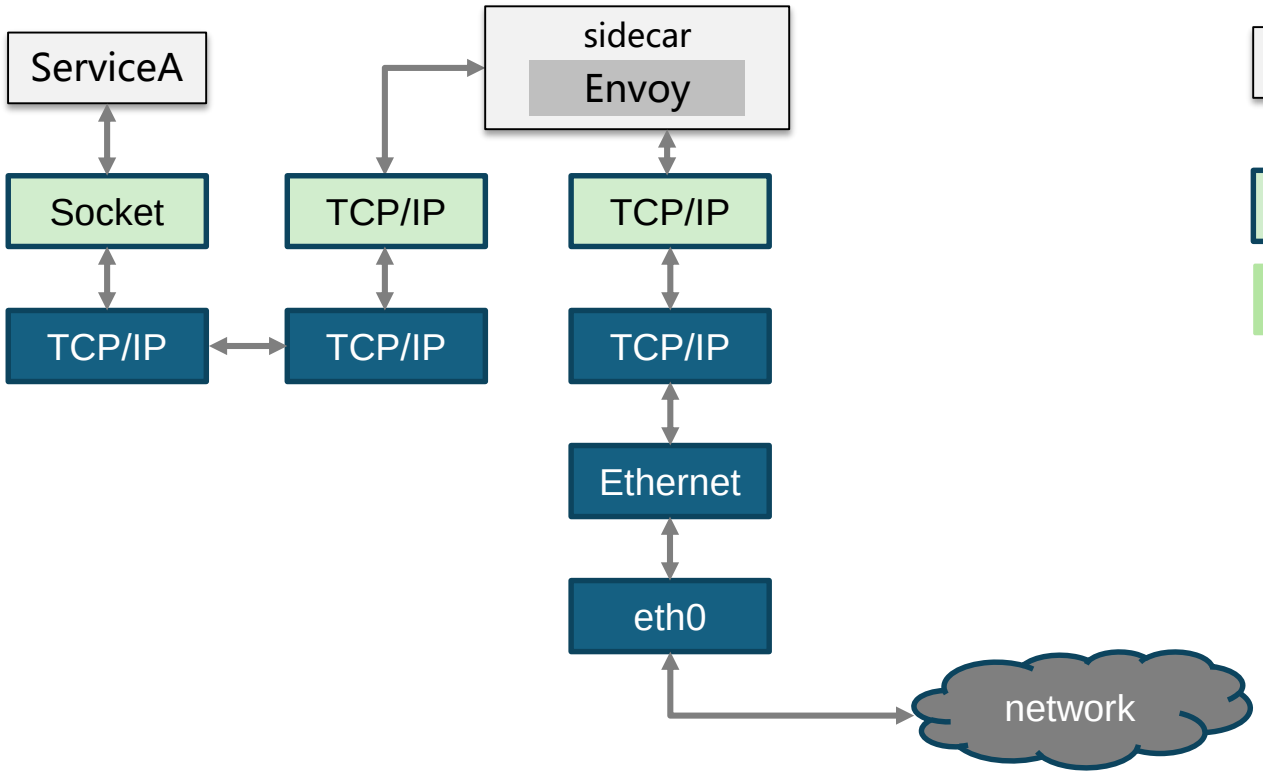
- **L7 Processing:**
  - Traffic Management, HTTP routing, loadbalancing, circuit breaking, rate limiting, fault injection, retry, timeouts, ...
  - Security: rich authorization policies
  - Obsevability: HTTP metrics, Access Logging, Tracing

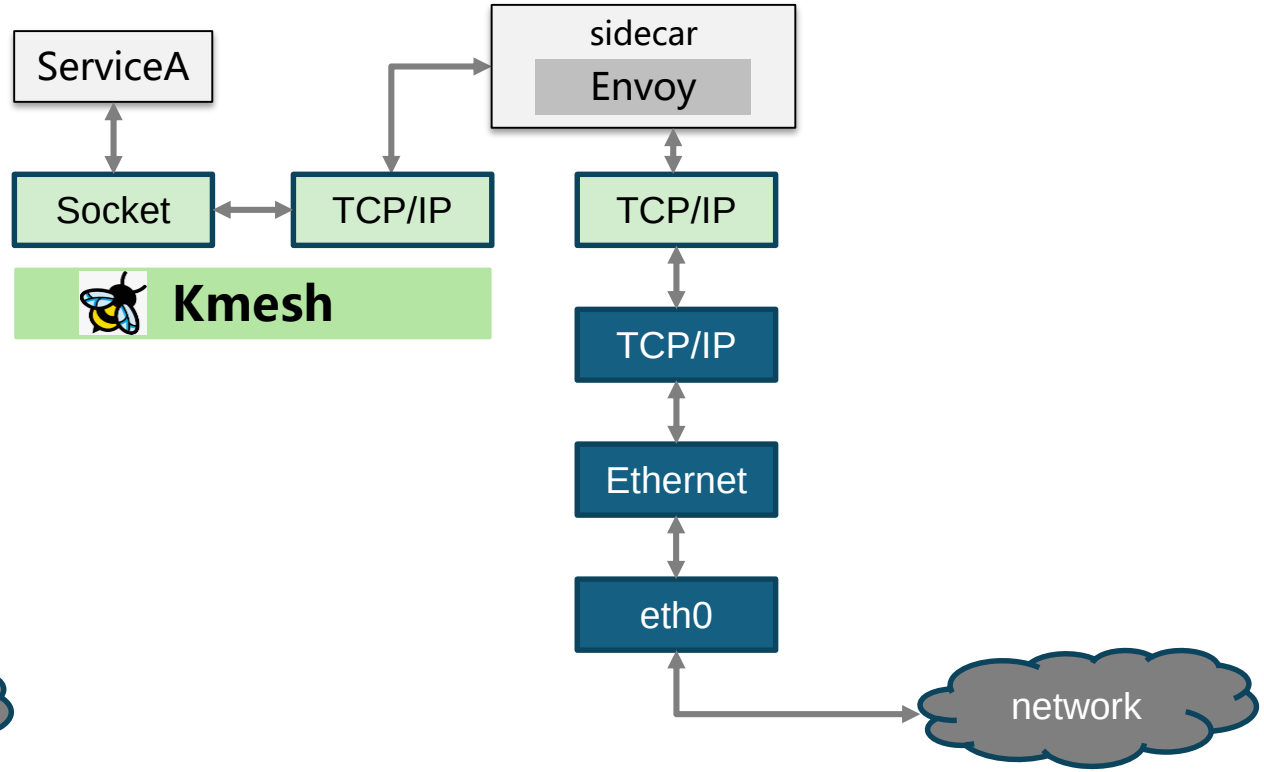
UX: Bypass Service Mesh



Bypass Kmesh traffic management

Istio Sidecar Interception Acceleration





Agenda

- Service Mesh Background
- Why Kmesh
- Kmesh Key Features
- **Future of Service Mesh**

Is Sidecarless the Future

Sidecarless

offoading： kmesh, etc

Performance

- Low resource overhead
- Low latency

Resilience

- Decoupling workload and infra
- Zero Intrusion
- No breaking traffic during upgrade

Next Hop

Sidecarless is the future arch，eBPF is imaginative，one day more complex L7 processing can be done with eBPF.

Currently Kernel is able to do simple HTTP traffic management

QA

KubeCon CloudNativeCon China 2024 | THE LINUX FOUNDATION OPEN SOURCE SUMMIT | AI_dev

Aug 22–Day 02

11:00 - 11:35

Level 1 | Hung Hom Room 5

Revolutionizing Service Mesh with Kernel-Native Sidecarless Architecture

(Xin Liu,Software Engineer, Huawei)

https://github.com/kmesh-net/kmesh

Kmesh Wechat Group
或添加k8s2222回复Kmesh