KubeCon CloudNativeCon

THE LINUX FOUNDATION

OPEN SOURCE SUMMIT

AI_dev

Open Source GenAI & ML Summit

China 2024


KubeCon
CloudNativeCon
THE LINUX FOUNDATION
OPEN SOURCE SUMMIT
AI_dev
Open Source GenAI & ML Summit
China 2024
Revolutionizing Service Mesh with Kernel-Native sidecarless Architecture

About Me

Xin Liu
Softwar Engineer,Huawei

KMESH

- Kmesh Maintainer
- Linux Kernel Contributors

Agenda

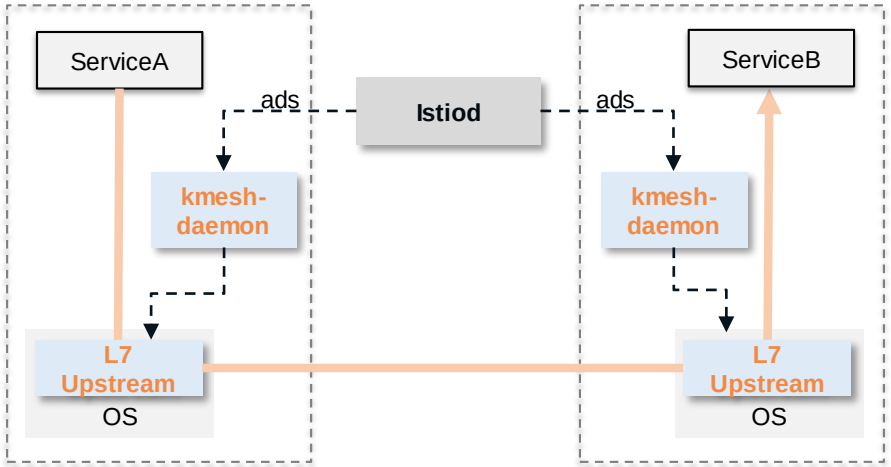
- Kmesh Overview
- How to implement L4/L7 traffic offload
- Restarting without stop the service
- High-performance non-intrusive observability

Agenda

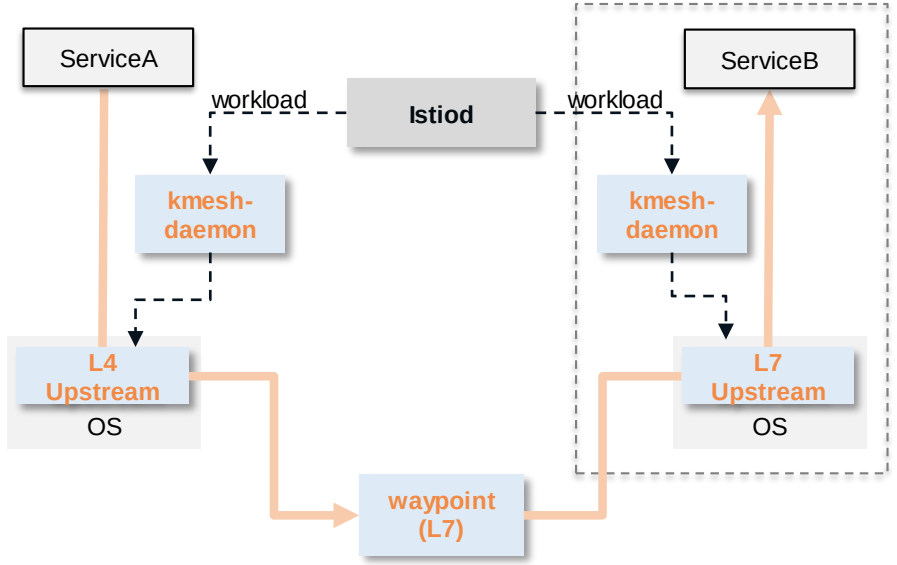
- Kmesh Overview
- How to implement L4/L7 traffic offload
- Restarting without stop the service
- High-performance non-intrusive observability

Kmesh Overview

Kmesh is a high-performance and low overhead service mesh data plane based on eBPF and programmable kernel. Kmesh brings traffic management, security and monitoring to service communication without needing application code changes. It is natively sidecarless, zero intrusion and without adding any resource cost to application container.



Kernel-Native:
L4~L7 ultimate performance



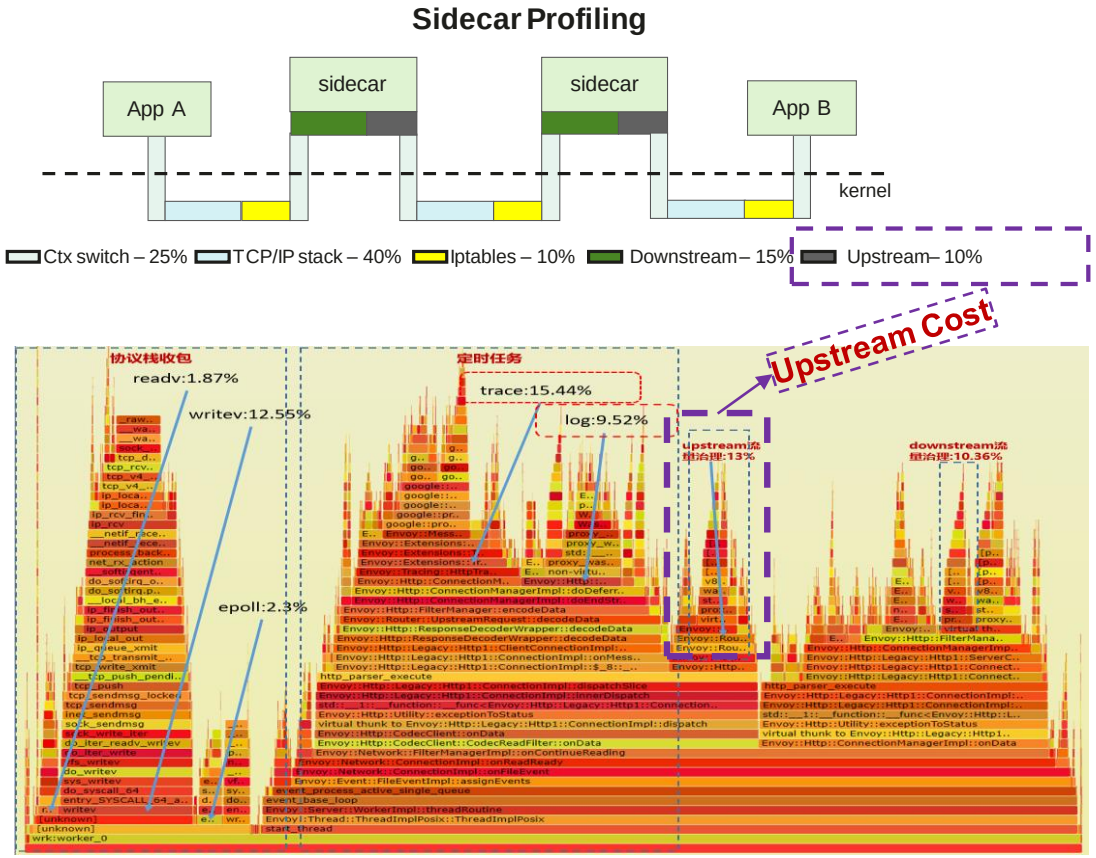
Dual Engine
L4/l7 Slicing the layers Flexible management

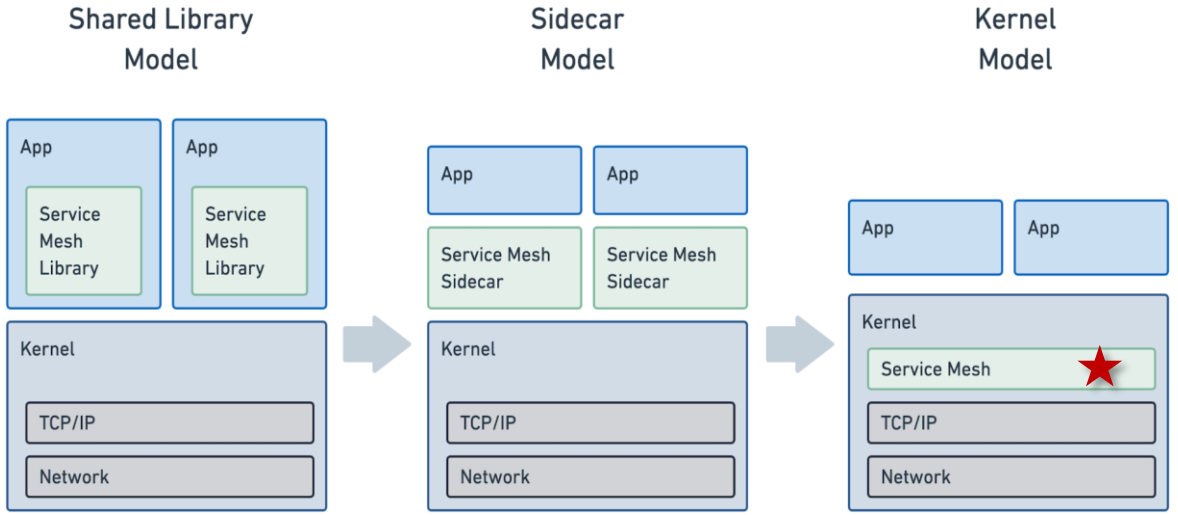
- **High Performance**
- **Low overhead**
- **High availability**
- **Security isolation**
- **Flexible management mode**
- **Seamless compatibility**

Agenda

- Kmesh Overview
- How to implement L4/L7 Upstream
- Restarting without stop the service
- High-performance non-intrusive observability

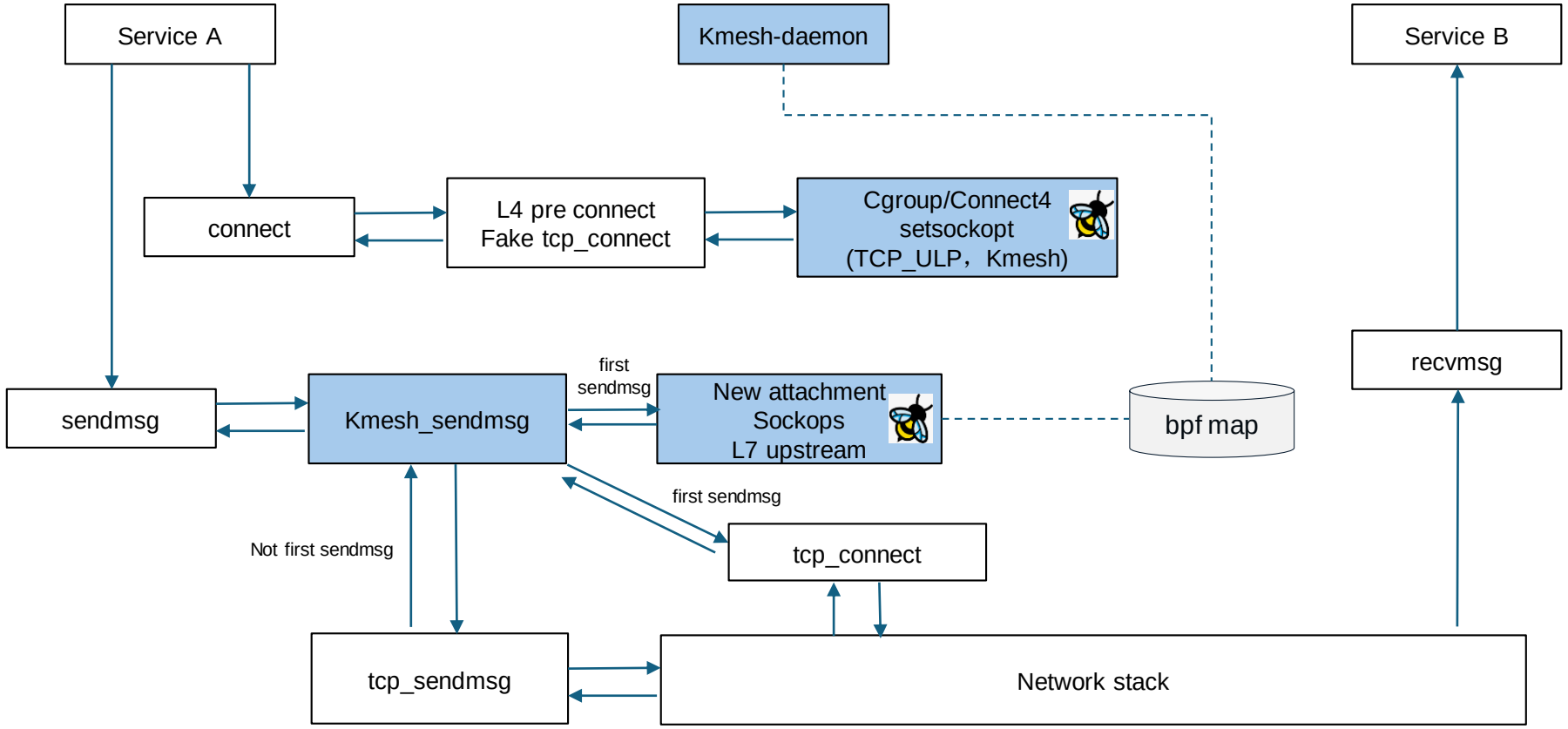
Istio sidecar performance profiling





The network time consumption shows that： **The sidecar architecture introduces a large amount of latency overhead. Upstream is only 10% of grid overhead.** Most overheads are caused by **data copy**, **two extra link setup** communications, and **context switch** scheduling.

Kernel-Native L7 Upstream



Kmesh L7 upstream

**The blue components are kmesh components.*

Kmesh-Native L7 upstream Contains multiple components:

- **Kmesh-daemo**

Configuration data delivery on the management and control plane

- **Cgroup/Connect4(ebpf):**

replace sendmsg with Kmesh_sendmsg by ULP

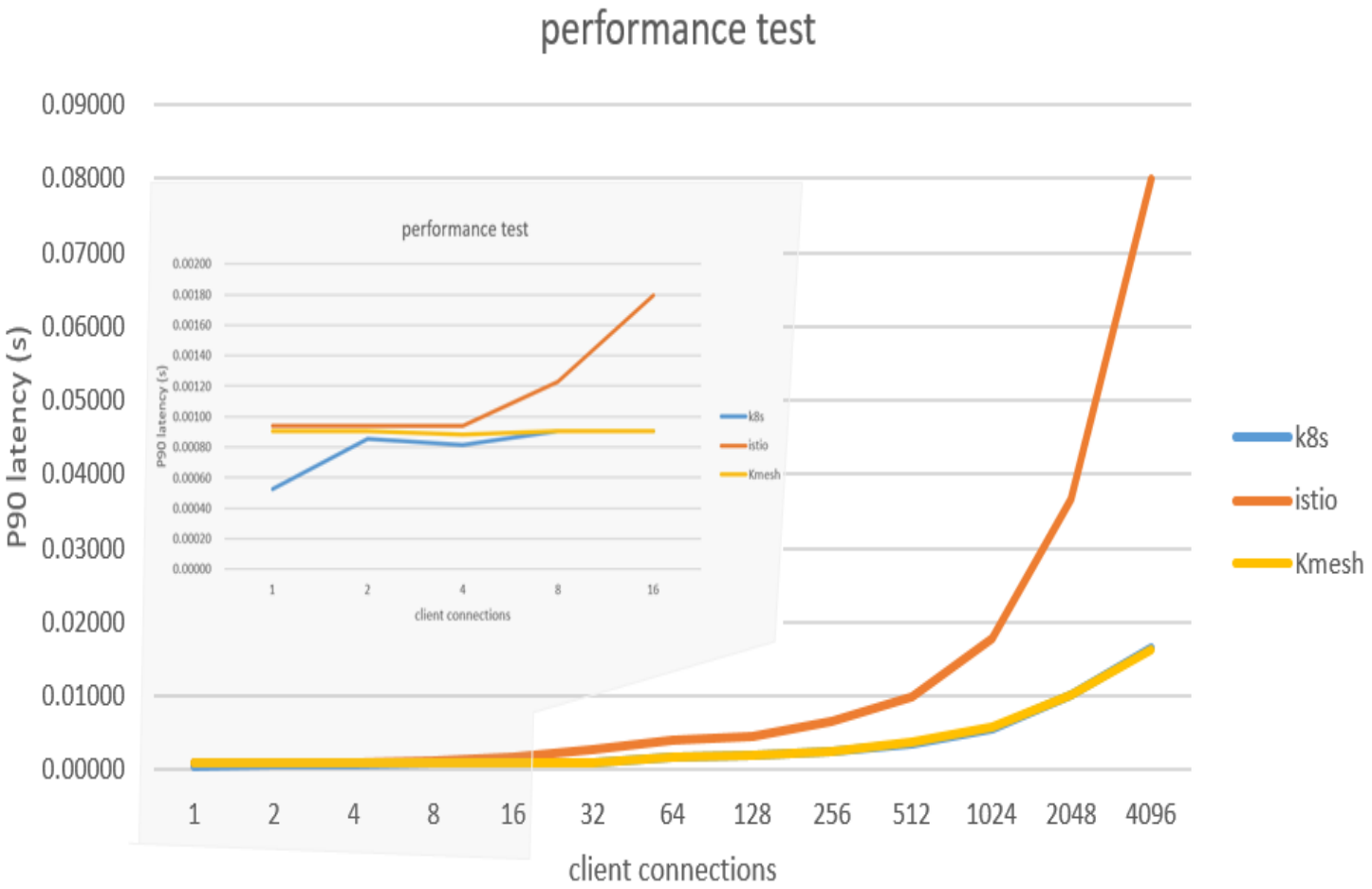
- **Kmesh sendmsg ko:**

The first sendmsg message defer connect and invoke the eBPF upstream logic.

- **Sockops(ebpf):**

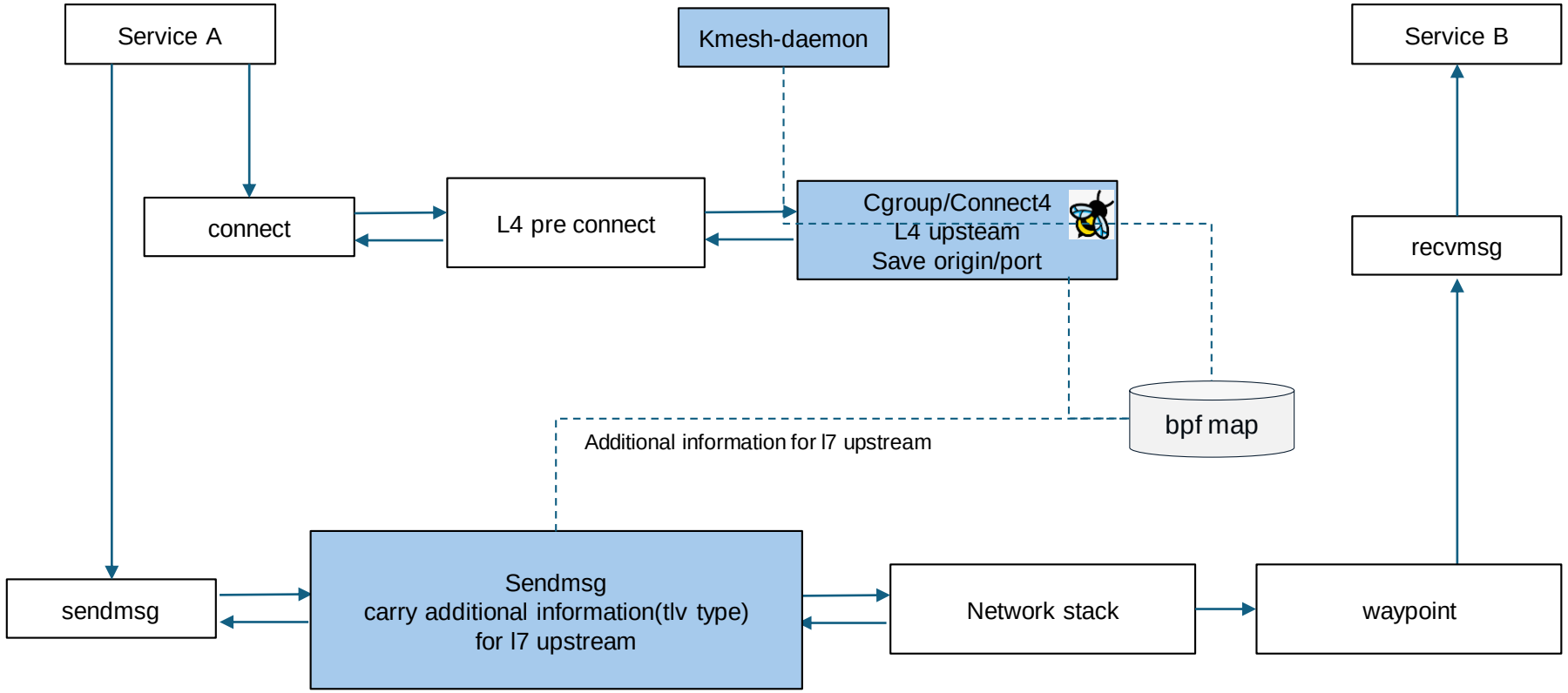
L7 upstream logic

Performance Comparation With Istio



Kernel-native: HTTP response latency almost same as baseline

Dual Engine mode



L4-waypoint upstream
Contains multiple components:

- **Kmesh-daemo**

Configuration data delivery on the management and control plane

- **Cgroup/Connect4(ebpf):**

L4 upstream logic

- **Sockops(ebpf):**

Trigger sendmsg execution.

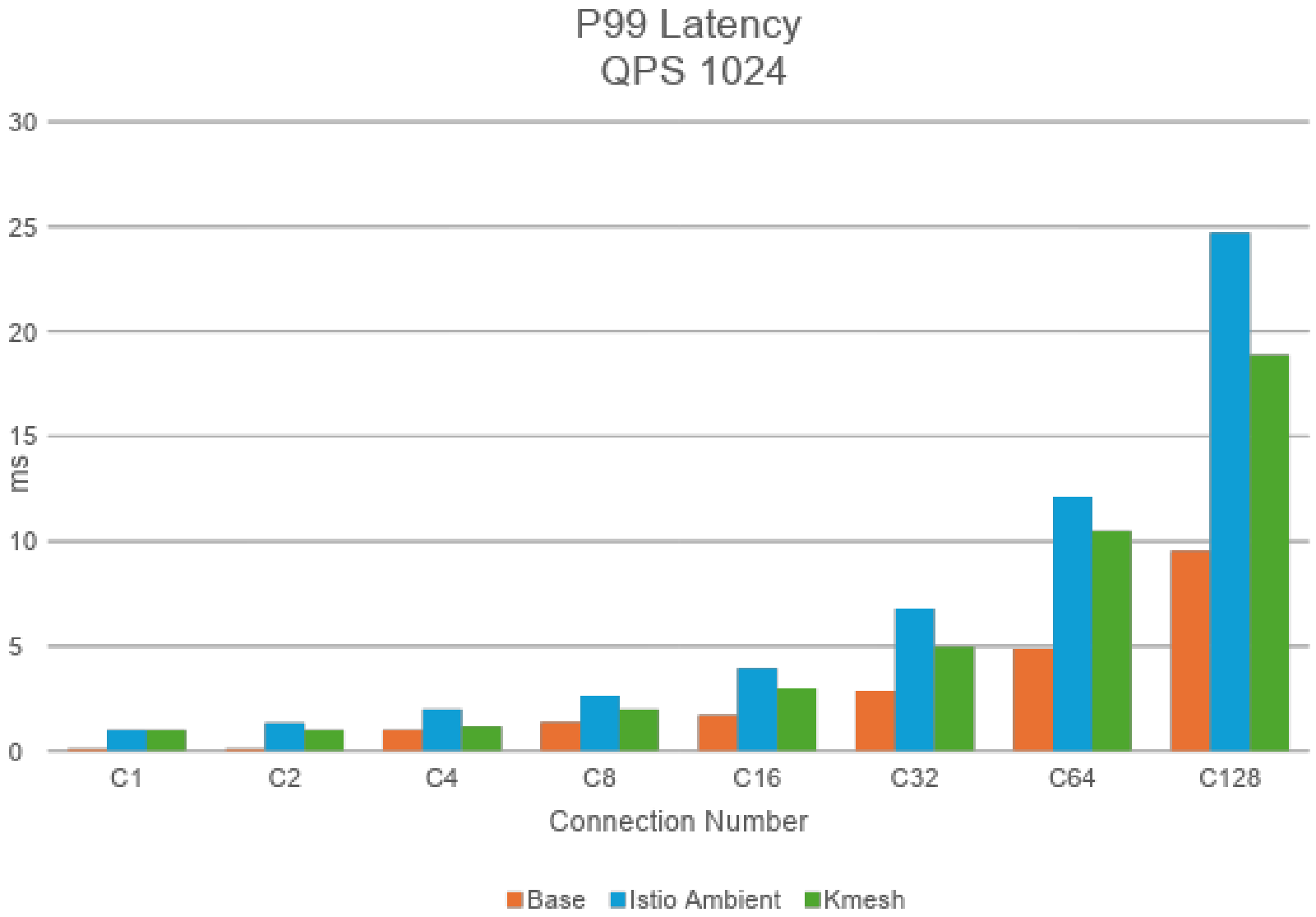
- **Sendmsg(ebpf):**

Carry additional information for l7 upstream to waypoint by tlv(type-length-value) type

Kmesh L4 upstream

**The blue components are kmesh components.*

Performance Comp with Ambient

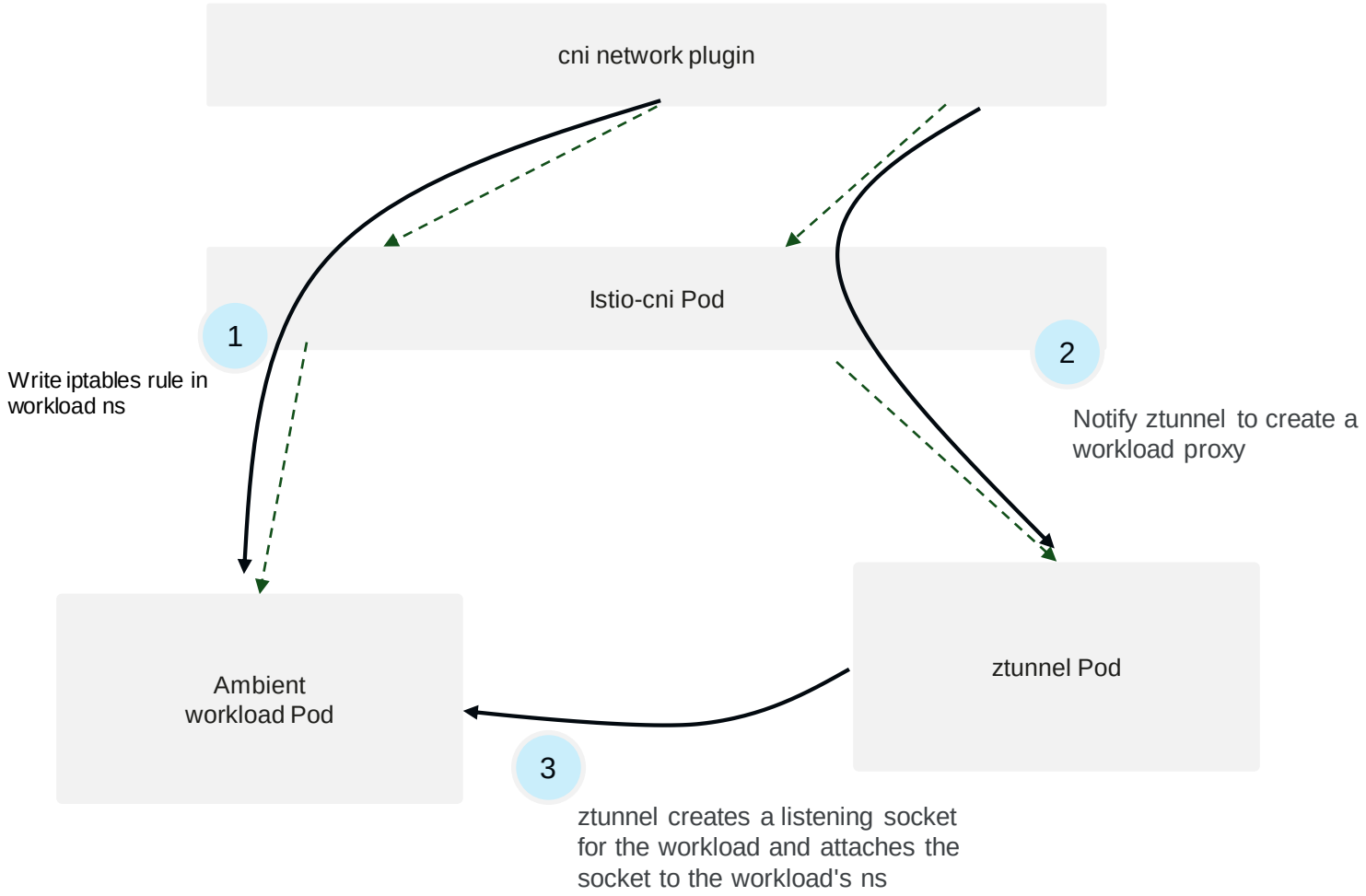


- Kmesh is faster than ambient mesh by about 25-30%
- Both are slower than base line with L7 processing

Agenda

- Kmesh Overview
- How to implement L4/L7 traffic offload
- Restarting without stop the service
- High-performance non-intrusive observability

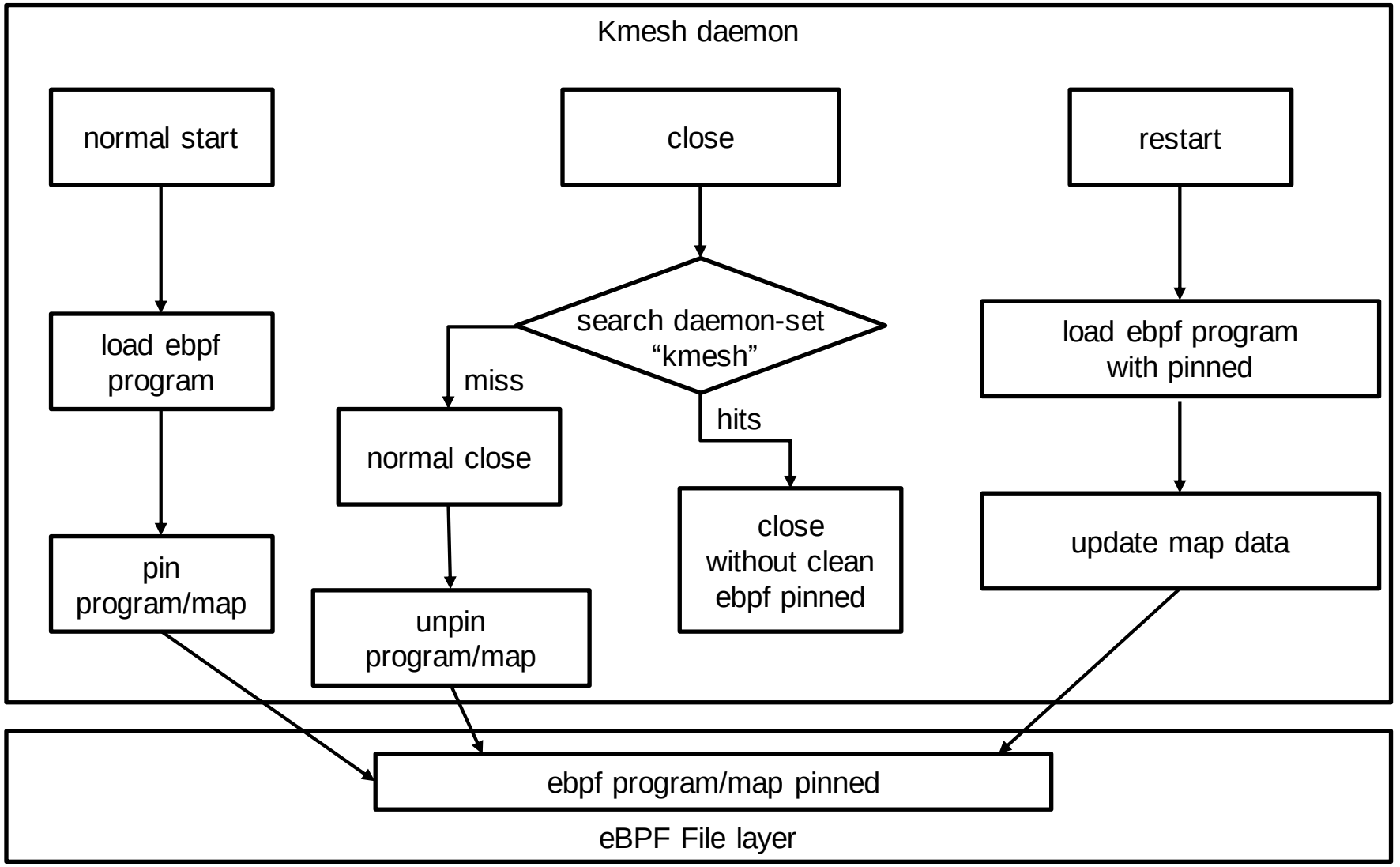
Ambient restart



Restart with connect interruption

- During the restart process, the original socket will be released, which will cause the original **connect to fail**.
- During the restart process, ztunnel **cannot provide effective services** for the newly established connection.

Restart without interruption



- The program/map is pinned to the EBPF file system through the EBPF pin mechanism
- When the Kmesh-daemon exits, check whether the Kmesh-daemon exits normally or restarts
- If the logout is normal, delete the pinned file
- If the system is restarted, the pinned file is not cleared. The system directly associates the pinned file during the next startup
- When Kmesh exits, the pinned file always exists and the service is not terminated

Agenda

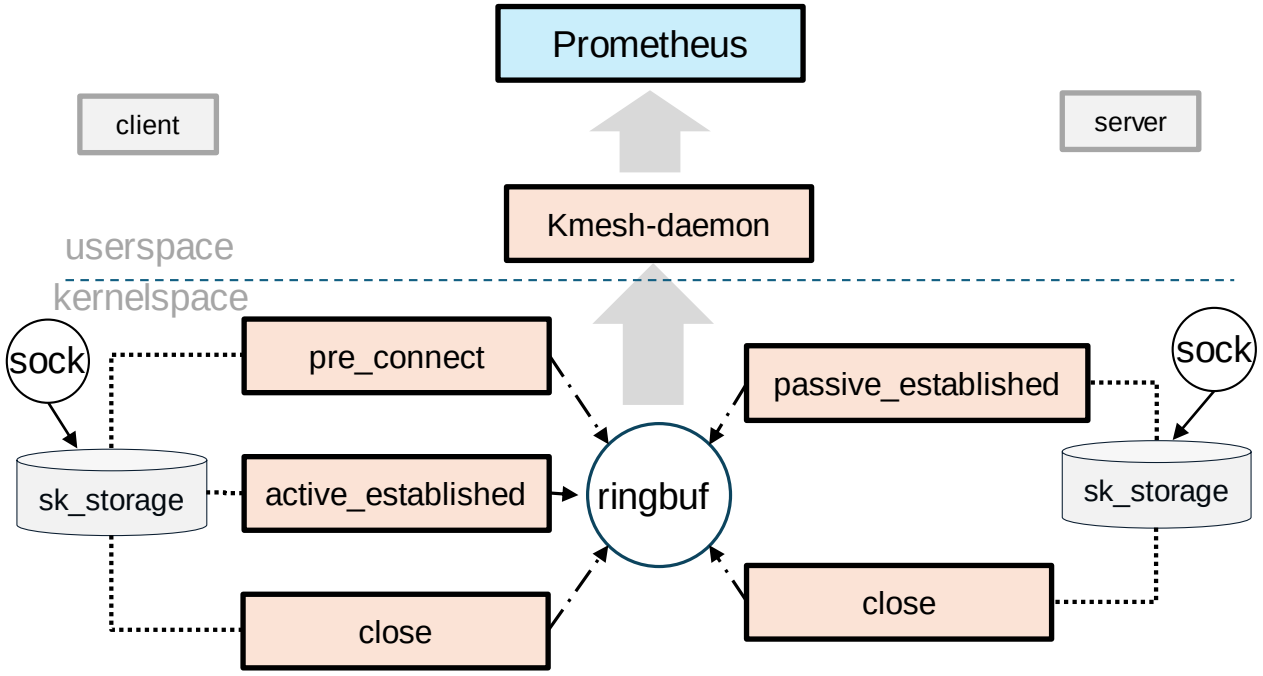
- Kmesh Overview
- How to implement L4/L7 traffic offload
- Restarting without stop the service
- High-performance non-intrusive observability

observability

Use the kernel ebpf sockops to collect the METIRC data during the connection running.

- pre_connect attachment
- active_established attachment
- passive_established attachment
- close attachment

More information can be collected in the future to support more native platforms.



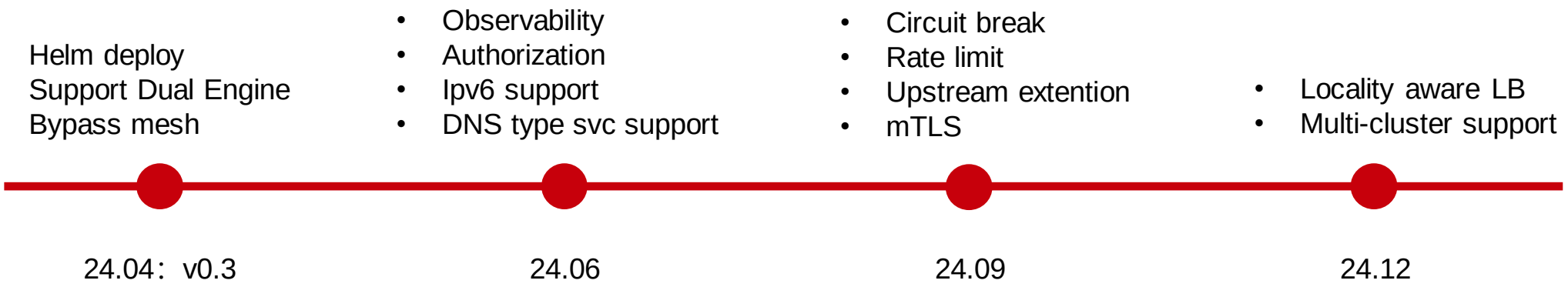
advantage:

1. E2E observability base on ebpf
2. Low cpu cost. < 5%
3. More low-level data collection. Socket link level data collection.

Agenda

- Kmesh Overview
- How to implement L4/L7 traffic offload
- Restarting without stop the service
- High-performance non-intrusive observability

Kmesh Roadmap



QA

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

扫码进入Kmesh技术交流群
或添加k8s2222回复Kmesh