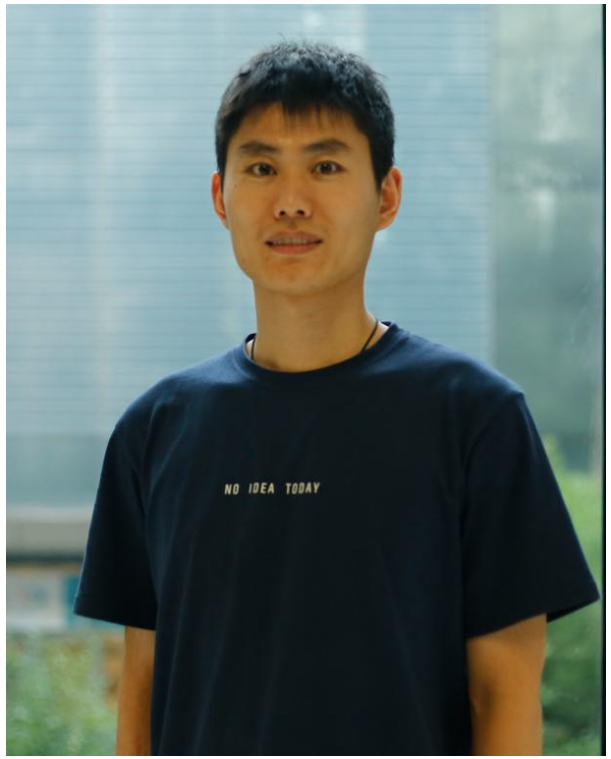
IstioCon

Kmesh: Architectural Innovation for a Brand-New Performance Experience

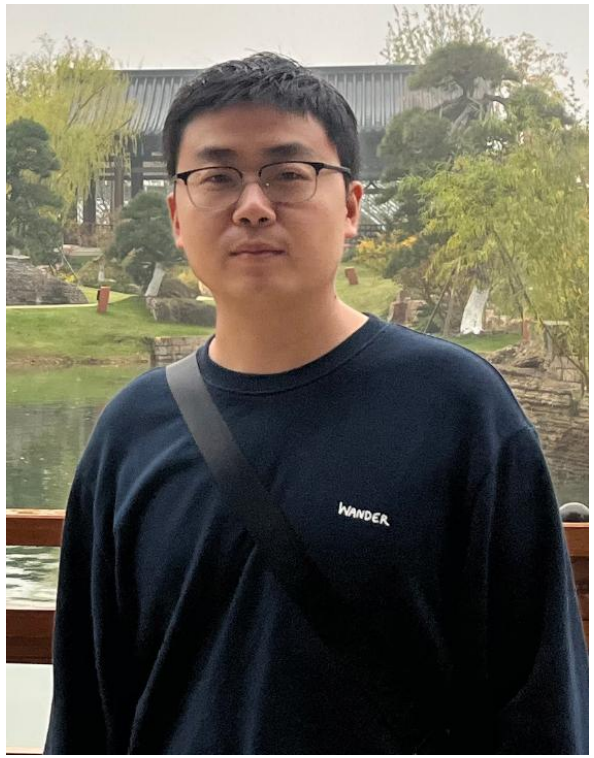
Songyang Xie & Zhonghu Xu, Huawei

About Us

IstioCon



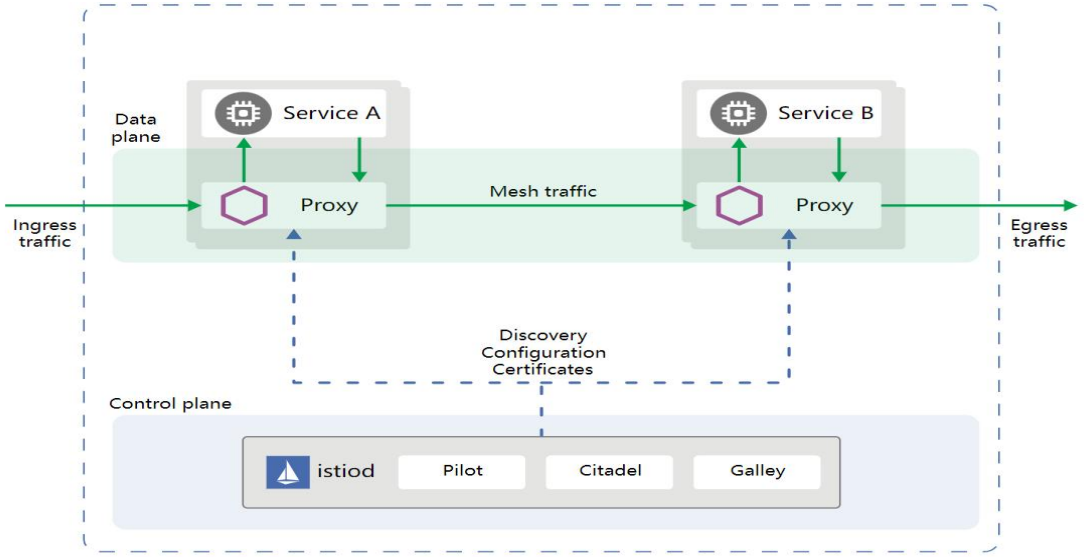
- Zhonghu Xu
- Principal engineer, Huawei Cloud
- Istio steering committee member, maintainer
- Kubernetes member & core contributor
- Google Open Source Peer Bonus Award
- Co-Authored《云原生服务网格Istio》& 《Istio权威指南》



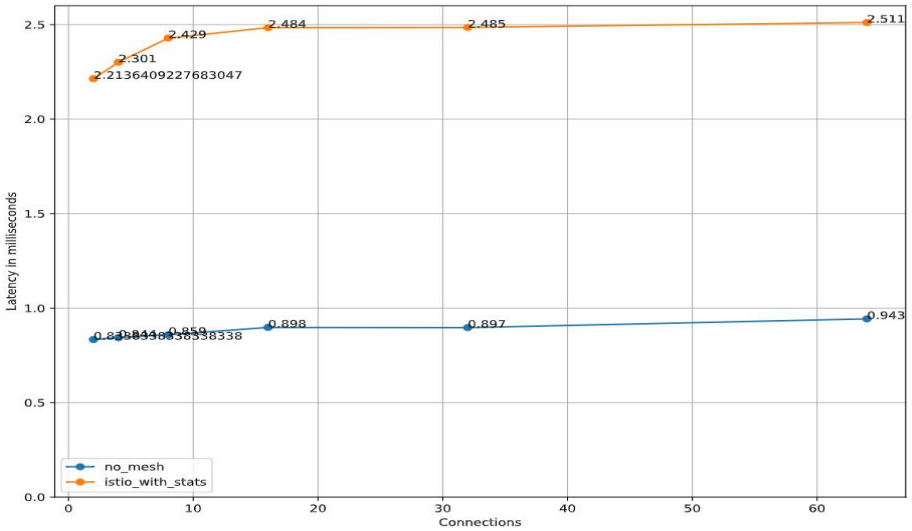
- Songyang Xie
- Senior Engineer, Huawei
- openEuler high-performance network sig maintainer
- openEuler ebpf sig maintainer
- Kmesh member & core contributor

Service Mesh: Resource and Latency Overhead Challenges

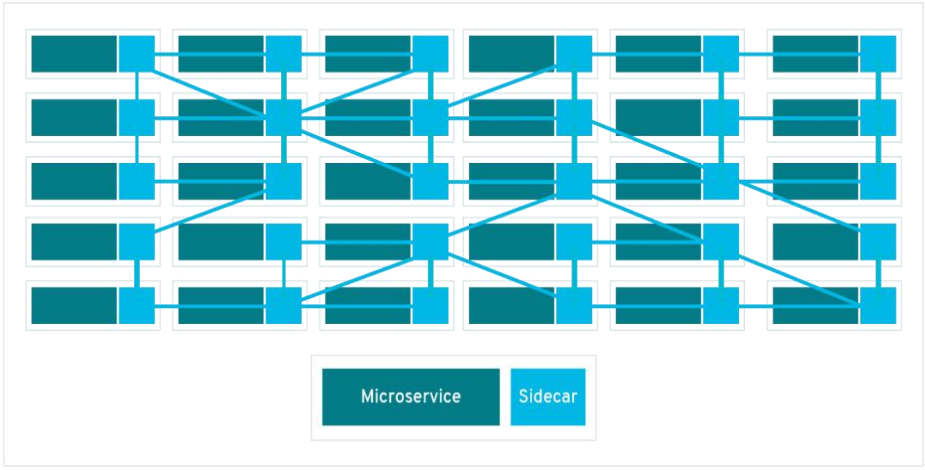
IstioCon



Traffic path



P99 latency vs client connections

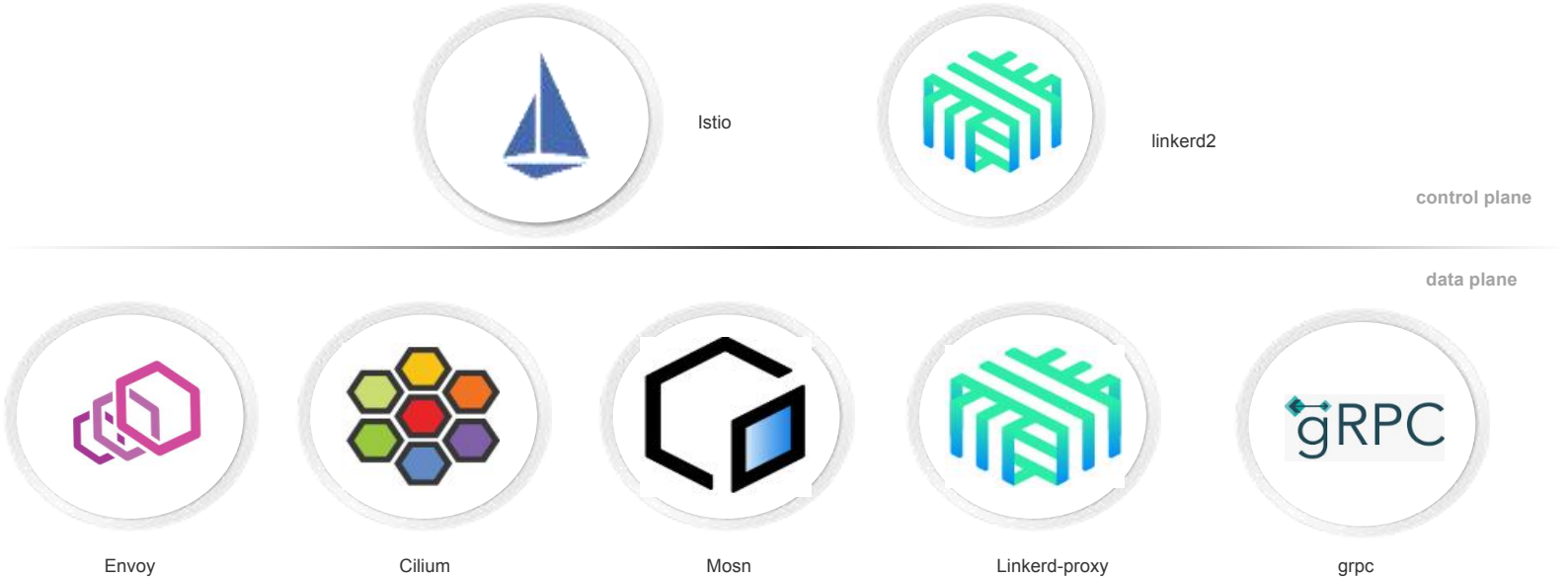


Deployment View

- **CPU and memory:** In Istio 1.19, a proxy consumes about 0.5 vCPUs per 1,000 requests per second.
- **Latency:** The two proxies add about 1.31 ms and 1.58 ms to the 90th and 99th percentile latency.

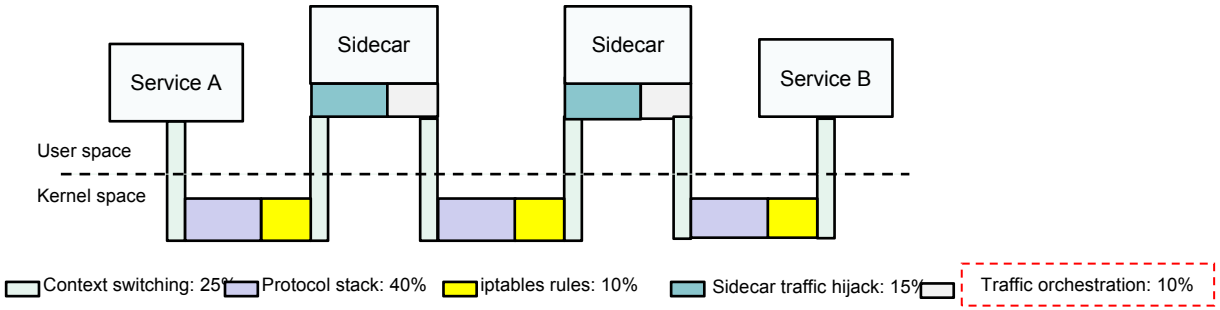
Reference: https://istio.io/latest/docs/ops/deployment/performance-and-scalability/

Community Exploration

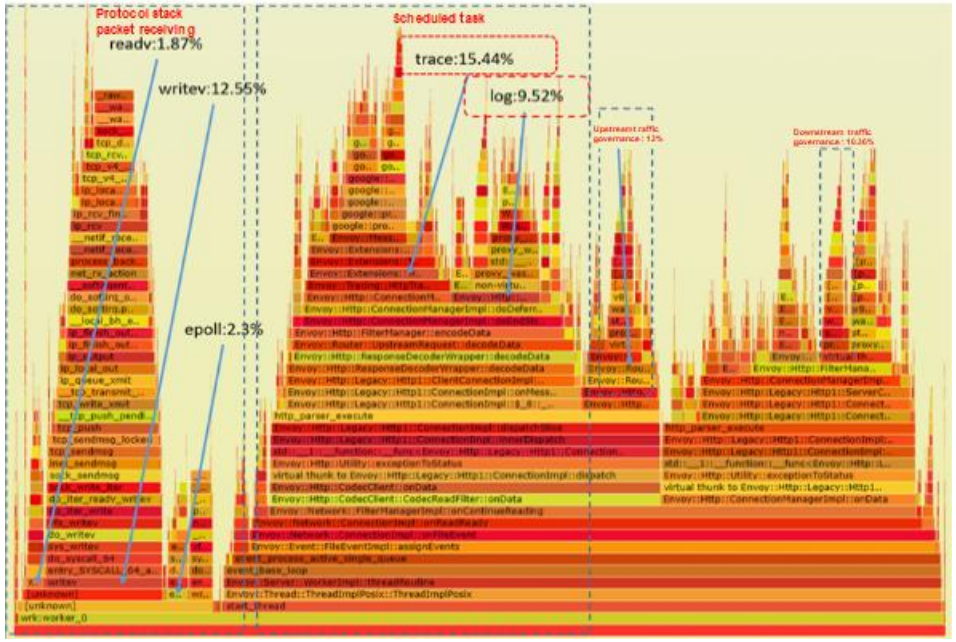


sockmap | L4 ebpf | sidecarless

Performance Analysis & Our Thoughts

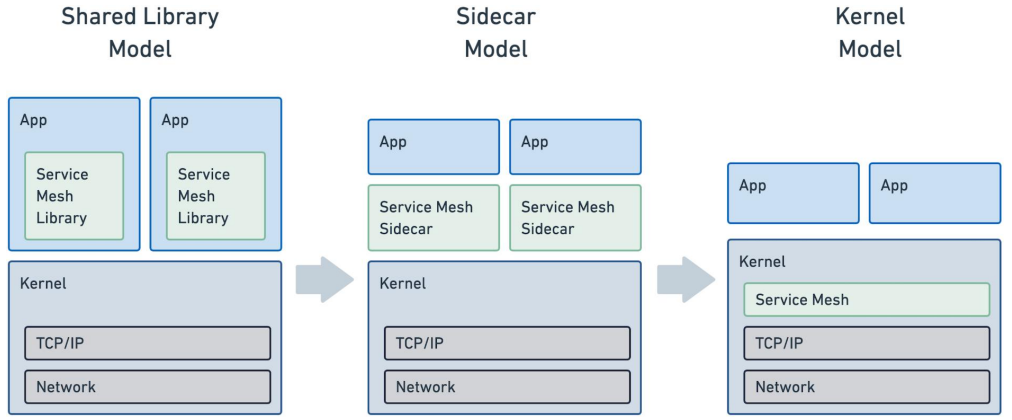


Time consumption distribution of the Sidecar pattern



Performance analysis:
According to the time consumption distribution, the Sidecar pattern introduces high latency overhead. **Traffic orchestration accounts for only 10% of the total overhead**, while most of the overhead lies in data copy, two redundant link setup communications, and context switching.

Our thoughts:
A transparent, efficient, and low-overhead service mesh infrastructure is needed to support a high-performance data plane for a Sidecarless service mesh.



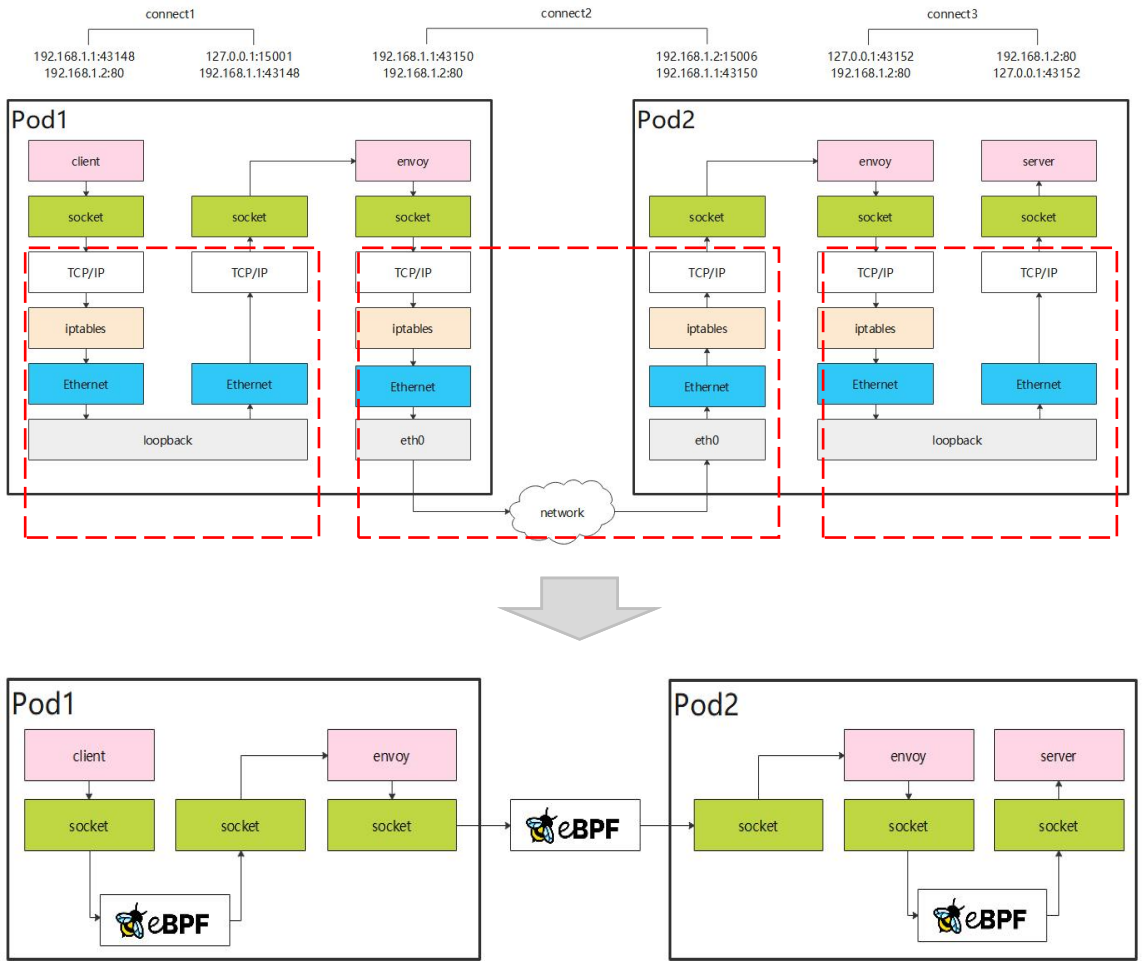
Our Approaches

IstioCon

- Service mesh data plane acceleration based on sockmap
- High-performance Sidecarless service mesh data plane at layer 4 to layer 7 based on the programmable kernel

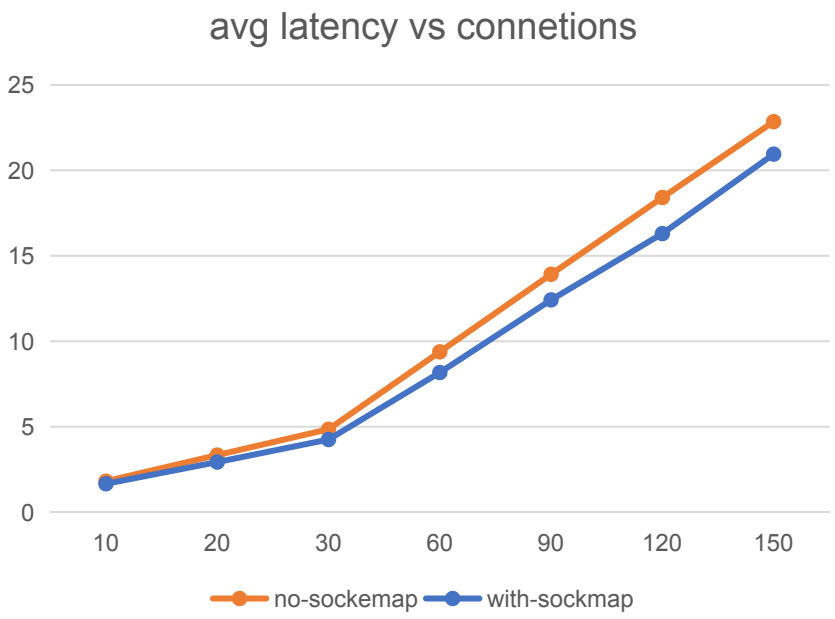
Sockmap: Accelerating Service Mesh Data Planes by Over 15%

IstioCon



Key technologies:

- Redirection of cross-socket data flows using sockmap
- Optimized kernel protocol stack paths

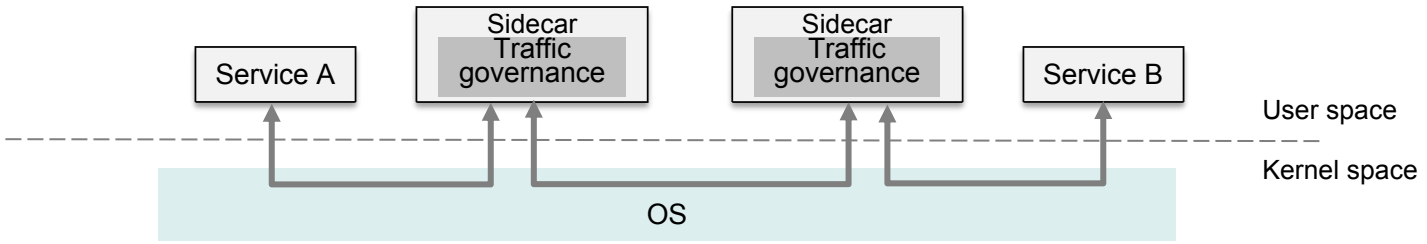


Latency performance improved by 15+%

Offload: Traffic Governance Offloaded to the OS

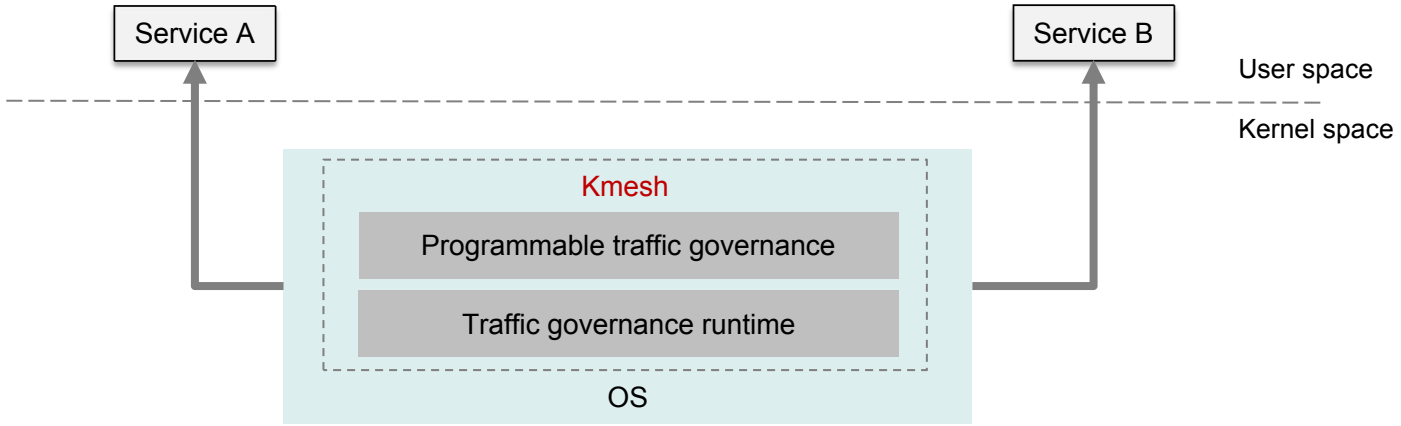
Proxy pattern

The data plane introduces additional latency, failing to meet the requirements of latency-sensitive applications.



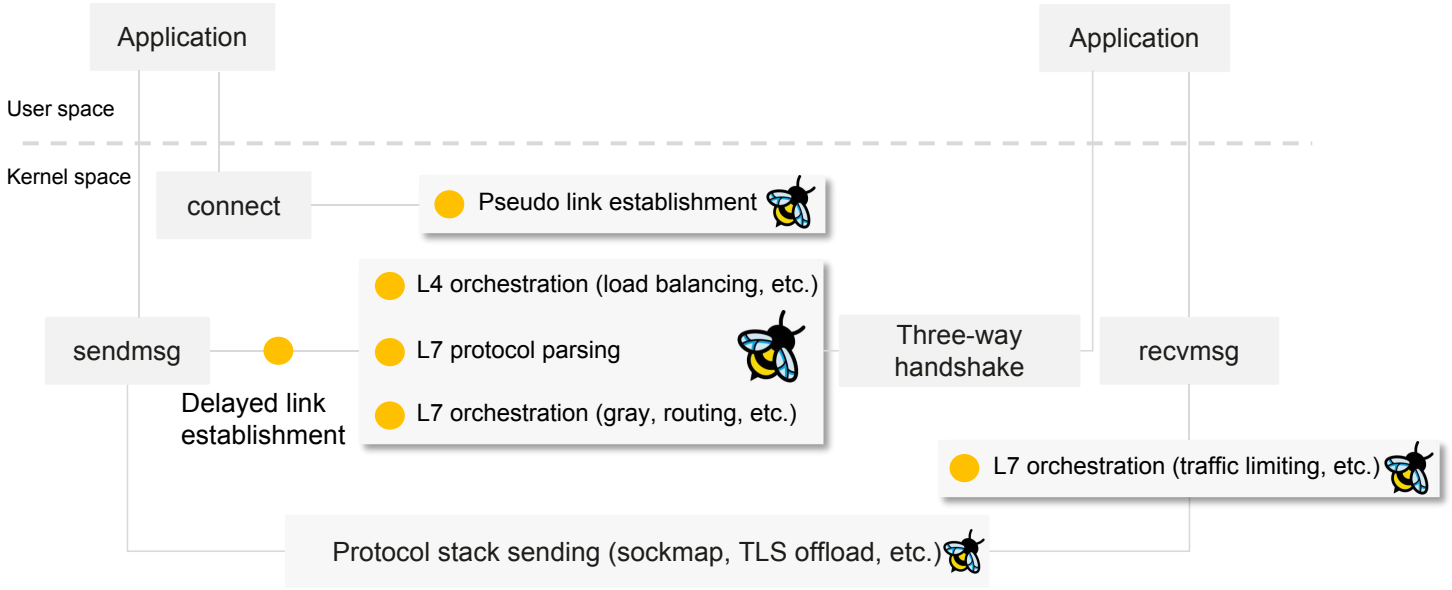
Kmesh

Based on the programmable kernel, Kmesh offloads traffic governance to the OS and shortens the traffic path from three hops to one.



- ✓ No context switching
- ✓ No data copy
- ✓ No proxy communication

High Performance: OS-Native Traffic Orchestration at Layer 4 to Layer 7



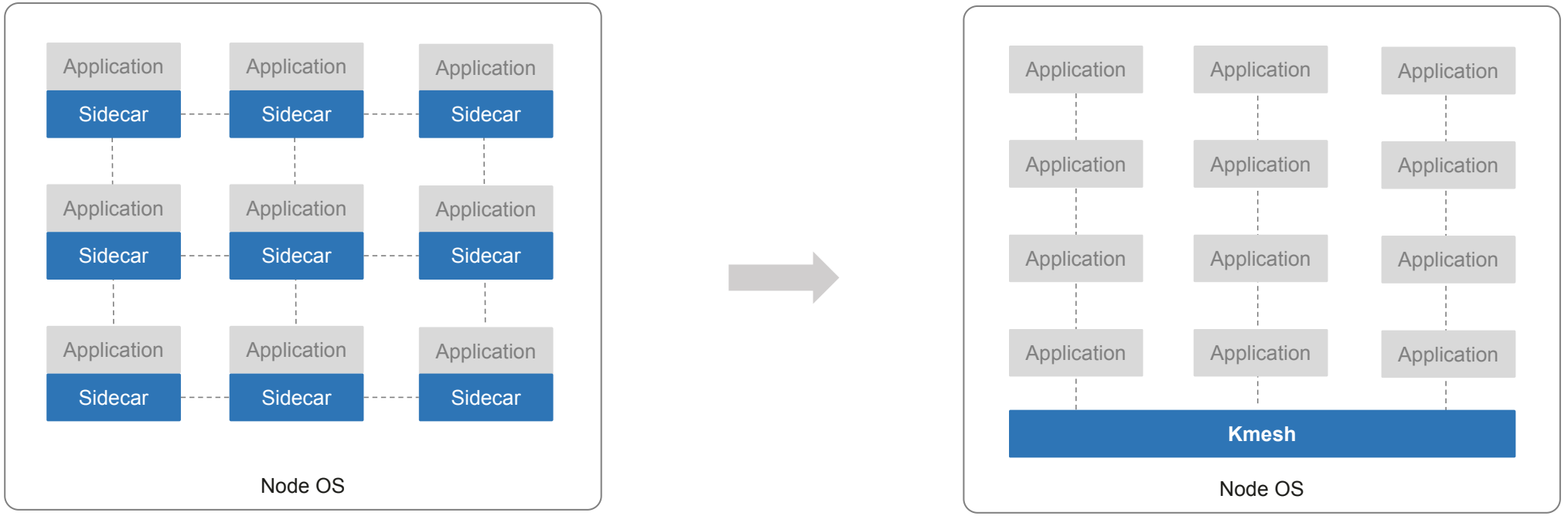
Kmesh traffic orchestration runtime

Traffic orchestration runtime:

- Based on technologies such as pseudo link establishment and delayed link establishment, Kmesh implements traffic orchestration at L4 to L7 in the OS.
- Kmesh builds a programmable full-stack traffic orchestration runtime in the kernel protocol stack using eBPF.

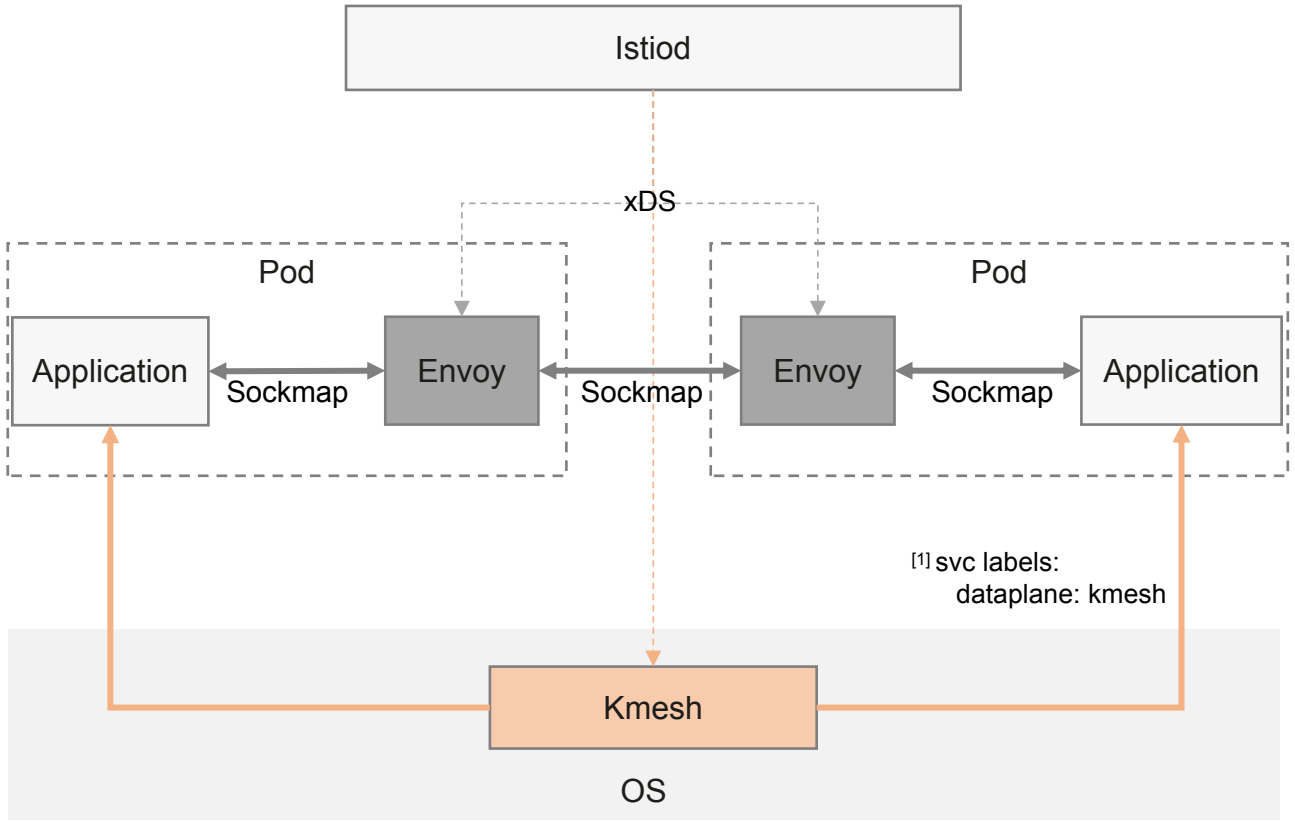
Low Overhead: 70% Lower Resource Overhead on the Data Plane

IstioCon



Smooth Compatibility: Collaborate With the Existing Data Plane

IstioCon



Smooth compatibility

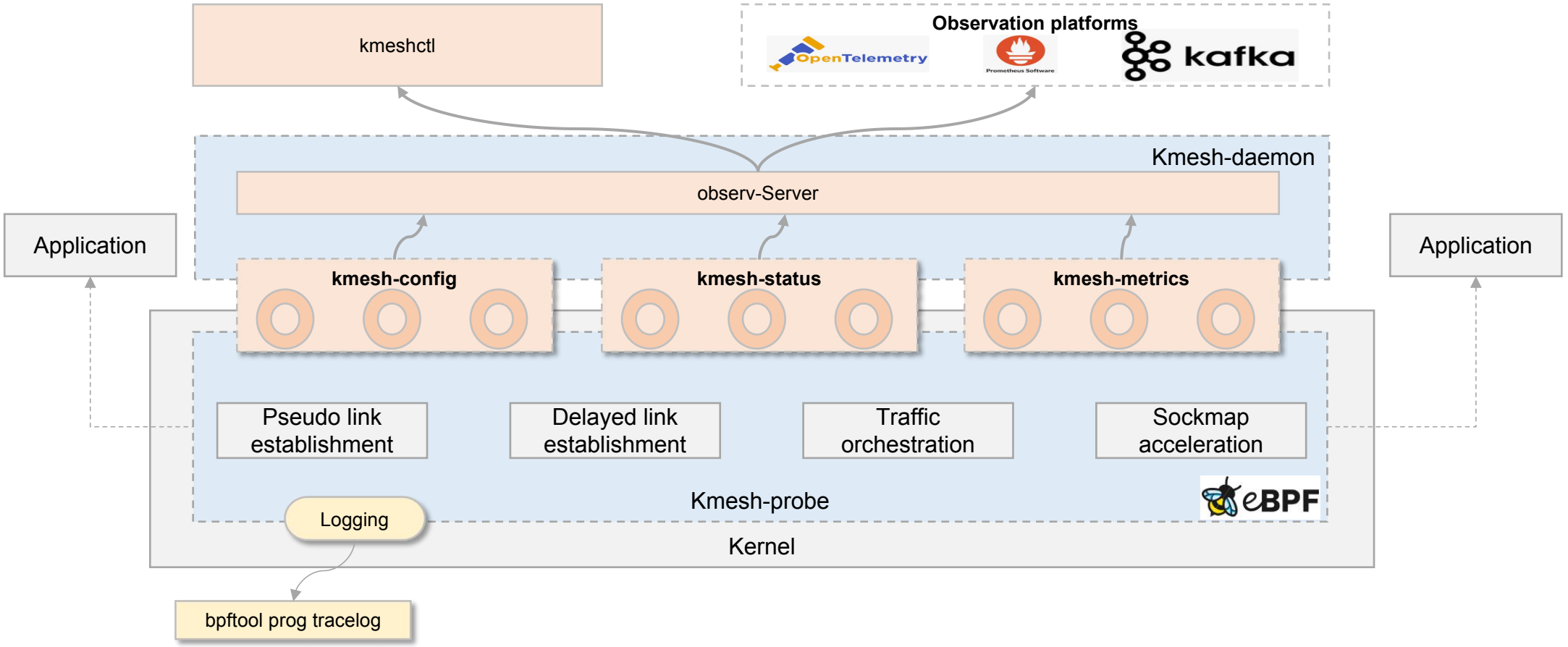
- Automatically connects to the service mesh control plane (Istiod).
- Supports xDS traffic orchestration protocols.

Collaborative governance

- Co-works with existing service mesh data planes. [1]
- Accelerates forwarding for existing service meshes by replacing iptables with sockmap.

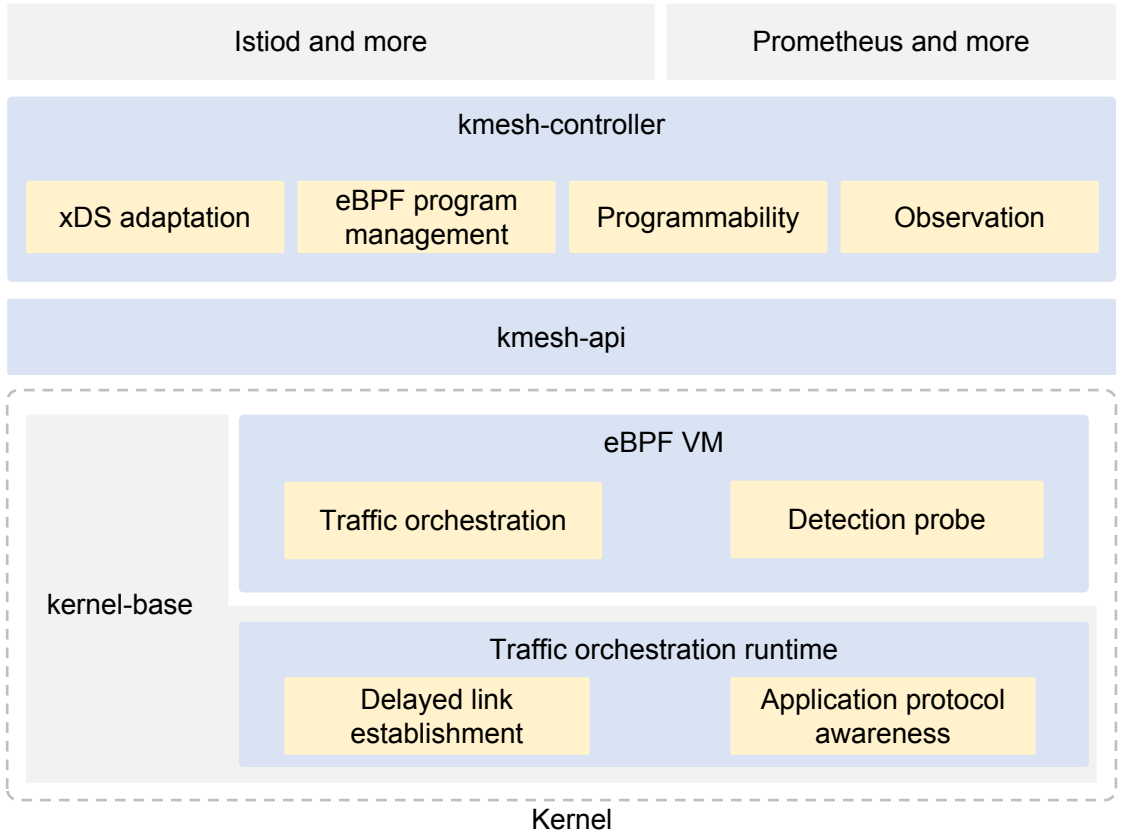
Full-Stack Visualization: Visualized Traffic Governance Full-Stack*

IstioCon



- ✓ End-to-end metric collection
- ✓ Low-level fine-grained observation
- ✓ eBPF-based low-load probe
- ✓ Interconnection with mainstream observation platforms

Kmesh Capability Overview



Smooth compatibility

- Application-unaware traffic governance
- Automatic interconnection with Istio

High performance

- **60%** lower service mesh forwarding delay
- **40%** faster service startup

Low overhead

- **70%** lower service mesh resource overhead

Open ecosystem

- xDS protocol compatibility

Security isolation

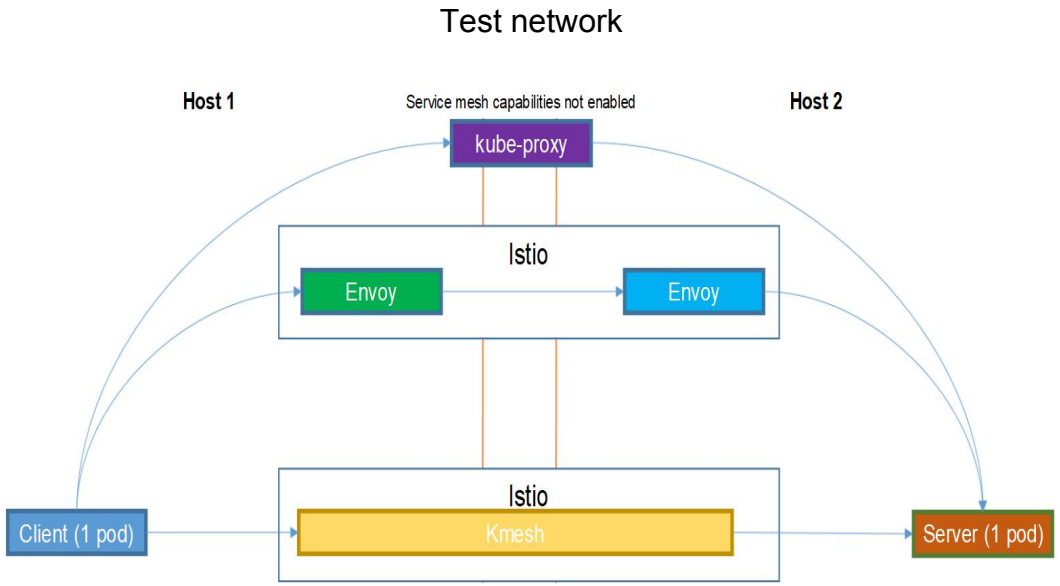
- eBPF VM security
- Cgroup-level orchestration isolation

Full-stack visualization

- End-to-end metric collection*
- Interconnection with mainstream observation platforms*

* In planning

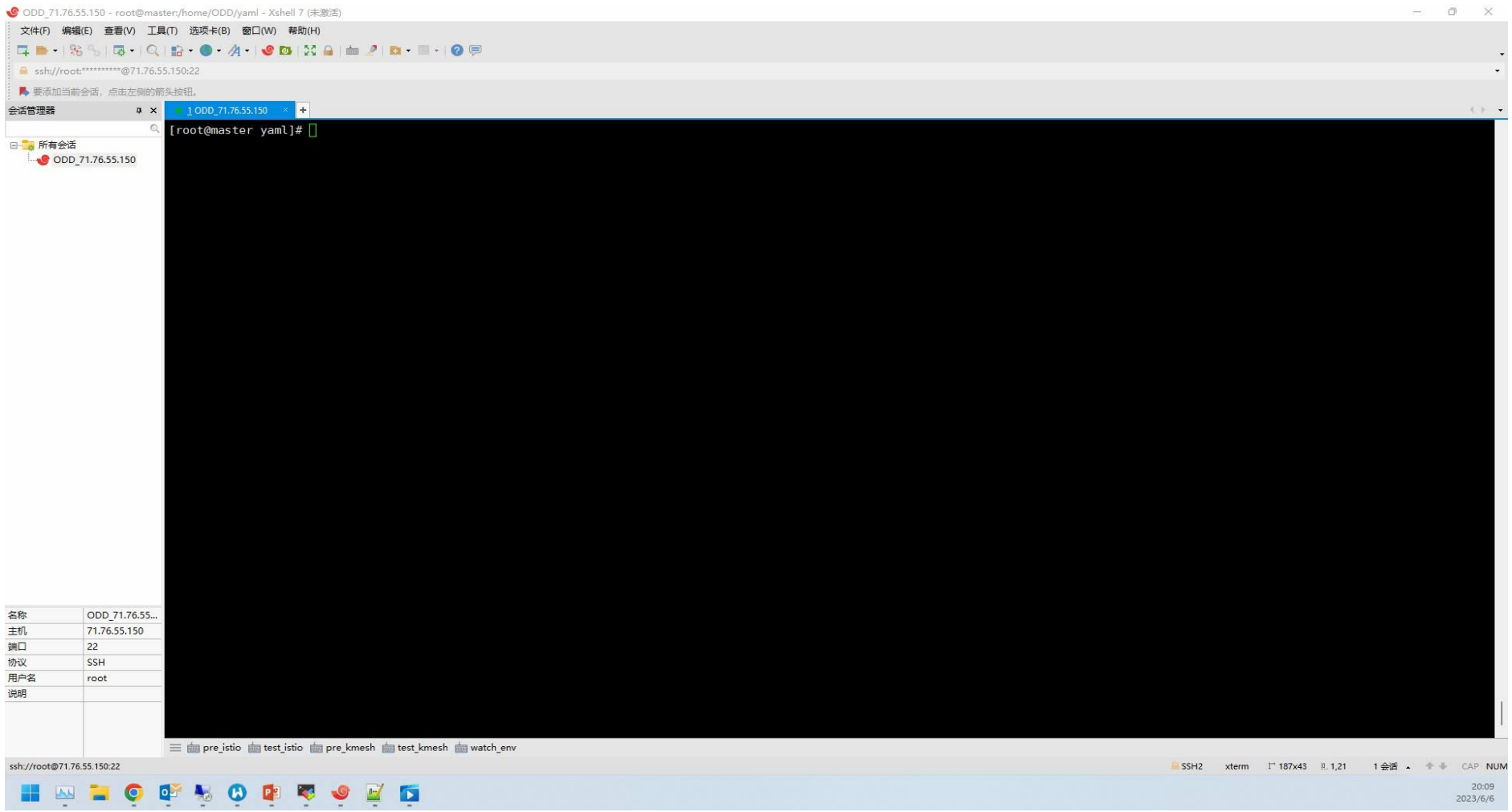
Performance



Layer 7 routing (protocol parsing + routing + load balancing)

Demonstration

IstioCon



Roadmap

IstioCon

- Traffic orchestration runtime
- HTTP 1.1 support
- Routing
- Grayscale release
- Load balancing

Sep 2023

Multi-service mesh collaboration

Oct 2023

Observability

Dec 2023

Circuit breaker and traffic limiting

Mar 2024

Mutual TLS governance

Sep 2024

Community

IstioCon

https://kmesh.net

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

technical exchange group

KMESH

IstioCon

Thank you