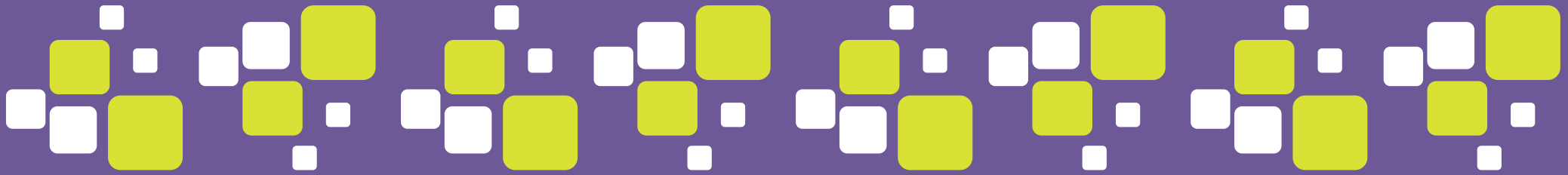
FLUIDOS-MESH

Webinar #1 - June 12th 2025 - CNIT, Univ. Rome Tor Vergata, Andrea Detti



FLUIDOS extensions

- Introduction of service mesh functionality
- Microservice offload and request routing for edge cloud multi-cluster scenarios

Why a Service Mesh

- **Built-in Observability:**
  Automatic collection of metrics, logs, and traces across services—without code changes.

- **Advanced Load Balancing:**
  Beyond random load balancing—enable L7 locality-aware and advanced load-balancing policies such as Least Outstanding Request.

- **Connection Persistence:**
  Enable long-lived connections and sticky sessions across microservices.

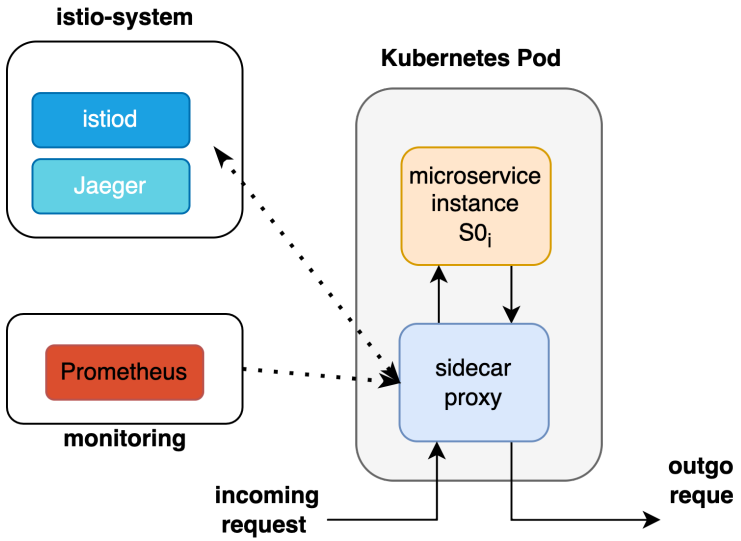
- **Zero-Trust Security with TLS:**
  Transparent mutual TLS (mTLS) between workloads ensures encryption, identity verification, and access control at the service level.

Why Istio Over Liqo for Multi-Cluster Mesh

- **Liqo.io is the multi-cluster glue of FLUIDOS**
- **Istio Multi-Cluster Native Support**
- **Liqo Protocol Agnostic Connectivity:**
  Liqo operates at L3, supporting any protocol (TCP, UDP, etc.), unlike Istio's L7 focus limited to HTTP/gRPC.
- **Liqo Full Support for Federated Applications:**
  Liqo provides Kubernetes resource sharing among clusters, not just service-to-service communication.
- **Liqo Tenant Resource Isolation:**
  Liqo ensures resource reservation per tenant—missing in traditional Istio setups.

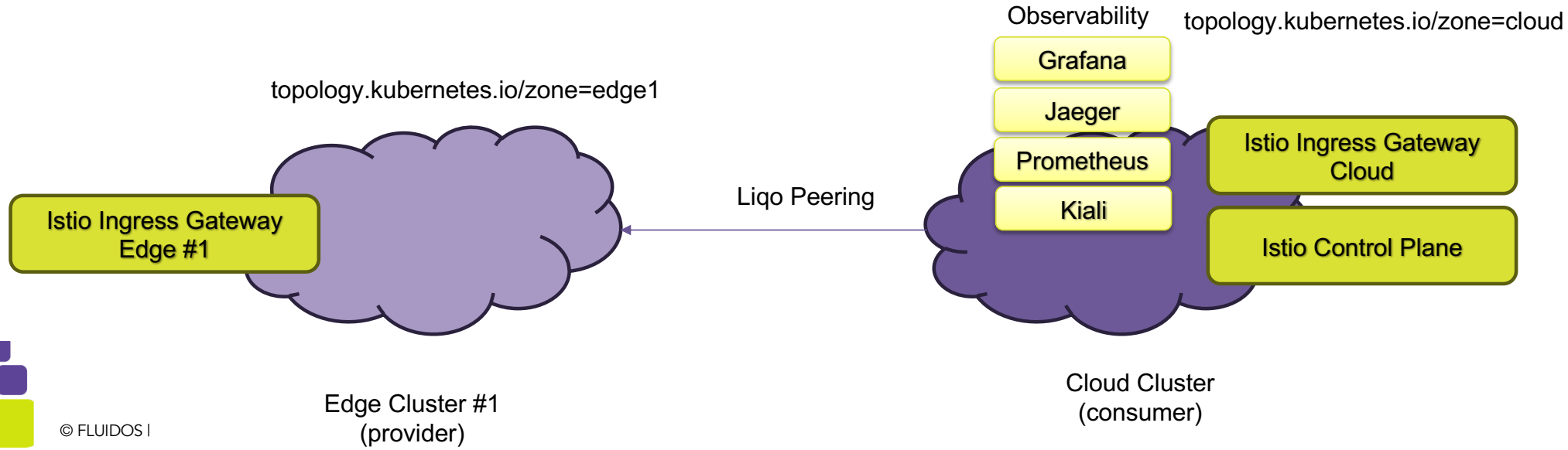
Inside Istio Service Mesh

- **Sidecar Injection:**
  Envoy proxies are automatically injected into Pods to intercept traffic.
- **Transparent Traffic Interception:**
  All inbound and outbound requests are captured for fine-grained control.
- **Control Plane Management:**
  Istiod configures sidecars with policies for load balancing, security, and observability.
- **Built-in Telemetry:**
  Metrics (Prometheus) and traces (Jaeger) are collected seamlessly from sidecar proxies.
- **L7 Load Balancing:**
  Smart traffic routing based on service-level information.
- **Ingress Gateways:**
  Manage external access to the mesh in a secure, observable way.

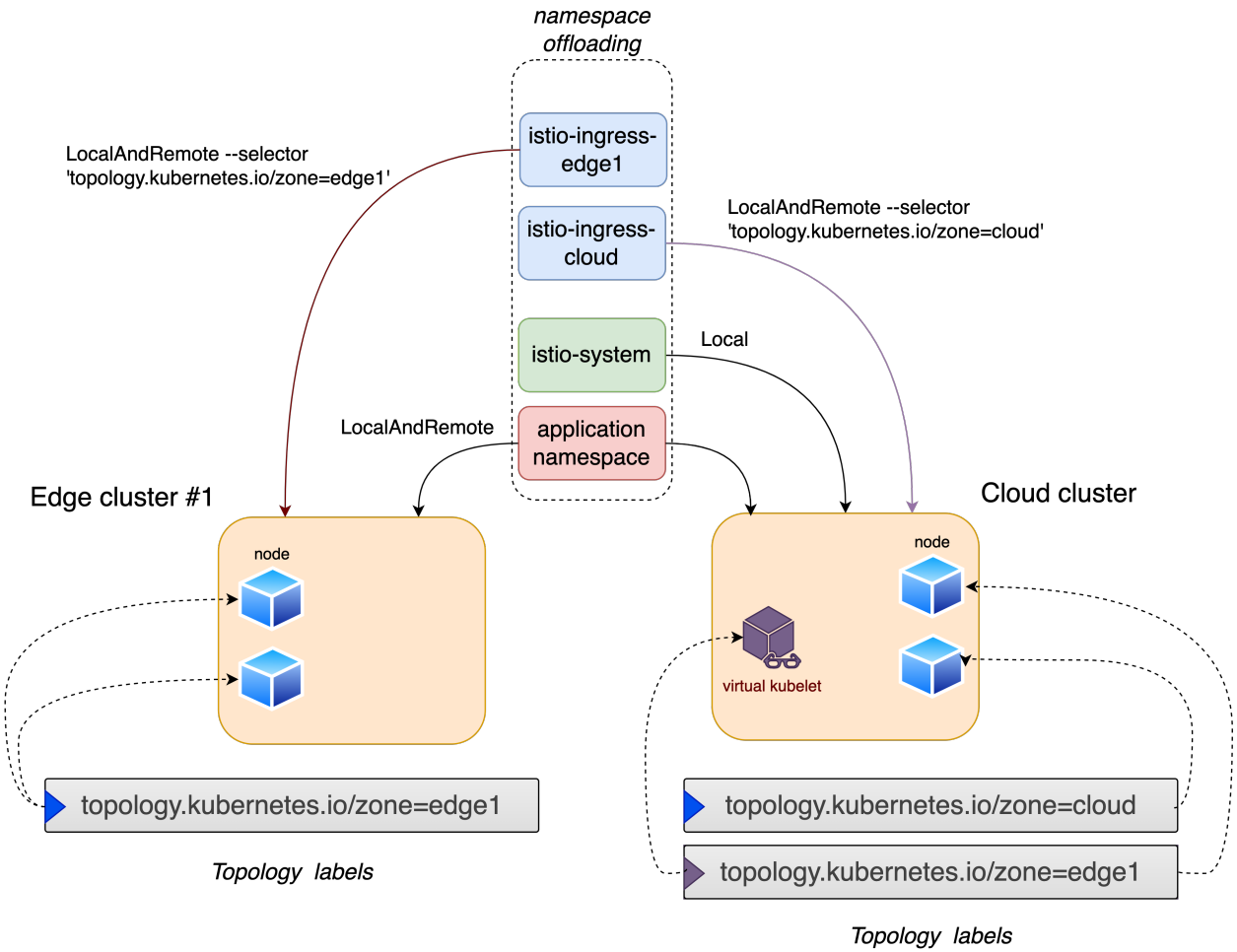


Istio for FLUIDOS edge-computing applications

- **Centralized Control Plane:**
  Istiod and observability tools (Prometheus, Grafana, Jaeger) run in the consumer (main) cluster.
- **Distributed Ingress Gateways:**
  Separate Istio-ingress gateways at edge and cloud clusters to manage local access points.
- **Locality-Aware Load Balancing:**
  Prioritize traffic to local Pods, reducing latency and inter-cluster traffic.

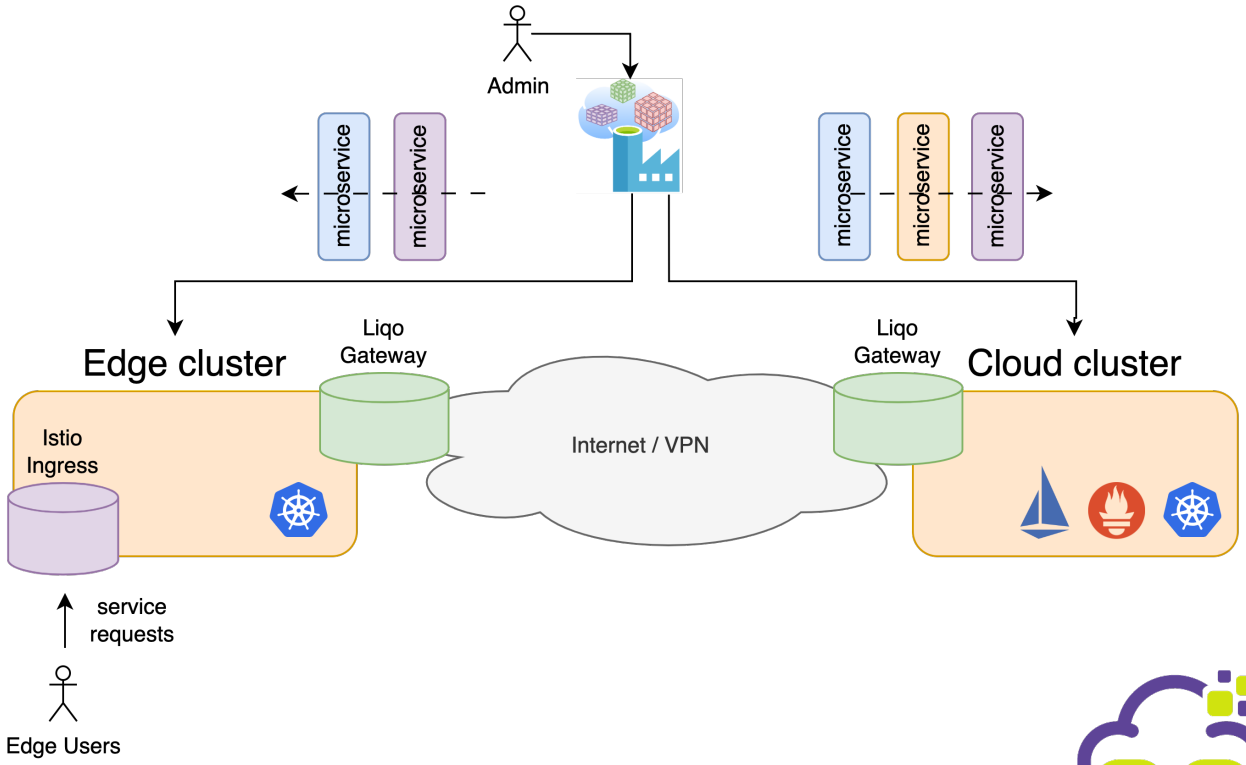


Namespace offloading



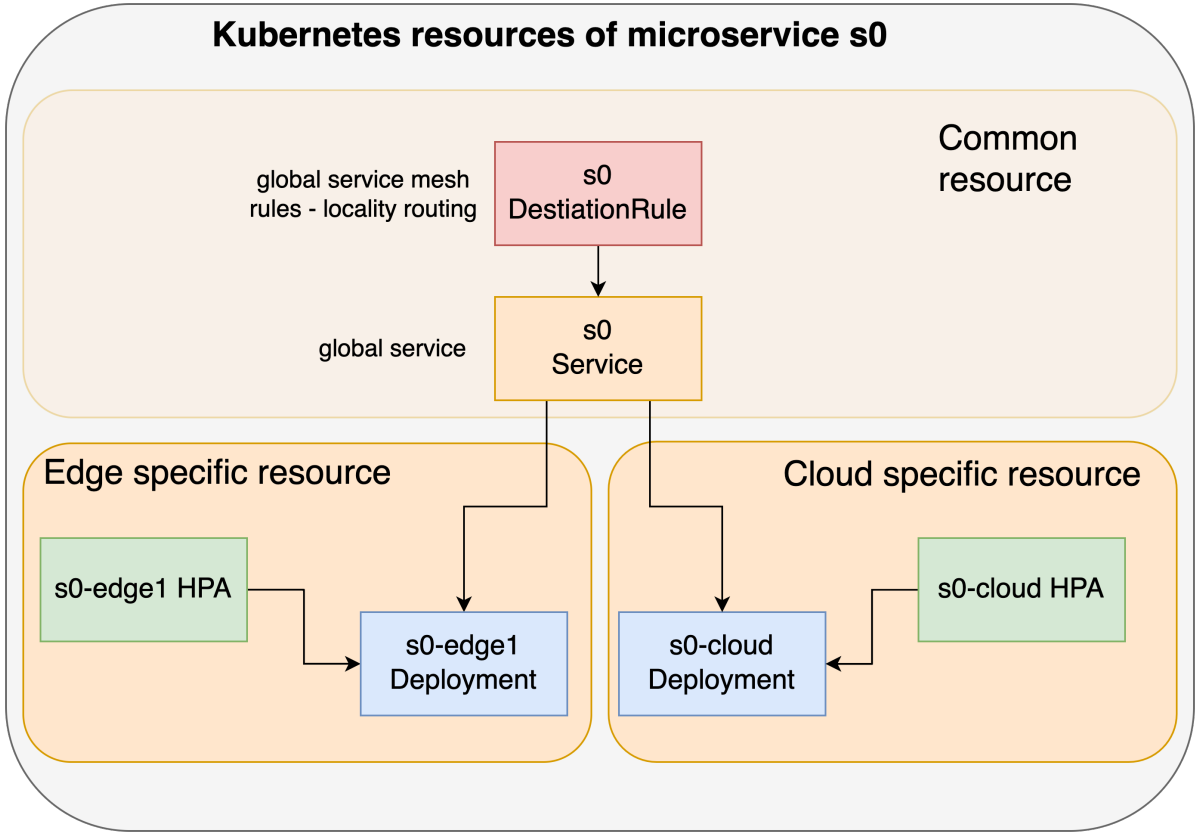
Cloud-Edge Microservice Distribution

- **Cloud as the Anchor:**
  The full microservice set runs in the cloud, ensuring central reliability and availability.
- **Edge-Specific Offloading:**
  Each edge hosts only the latency-critical subset of microservices.
- **Adaptive Deployment:**
  The microservice subset varies per edge site based on user proximity and application requirements.



k8s/Istio packaging

- **Two Deployments:**
  Same containerized microservice, placed in distinct locations (s0-local, s0-remote1) via node topology affinity.
- **Independent Autoscaling:**
  One HPA per Deployment to adapt to local traffic independently.
- **Unified Service Access:**
  A single Kubernetes Service enables transparent access using a shared DNS name across locations.
- **Locality-Aware Traffic Control:**
  A single Istio DestinationRule ensures traffic is routed to the nearest available instance.
  - **No Back-and-Forth Traffic Loops**
    avoiding inefficient and fragile random paths across clusters.

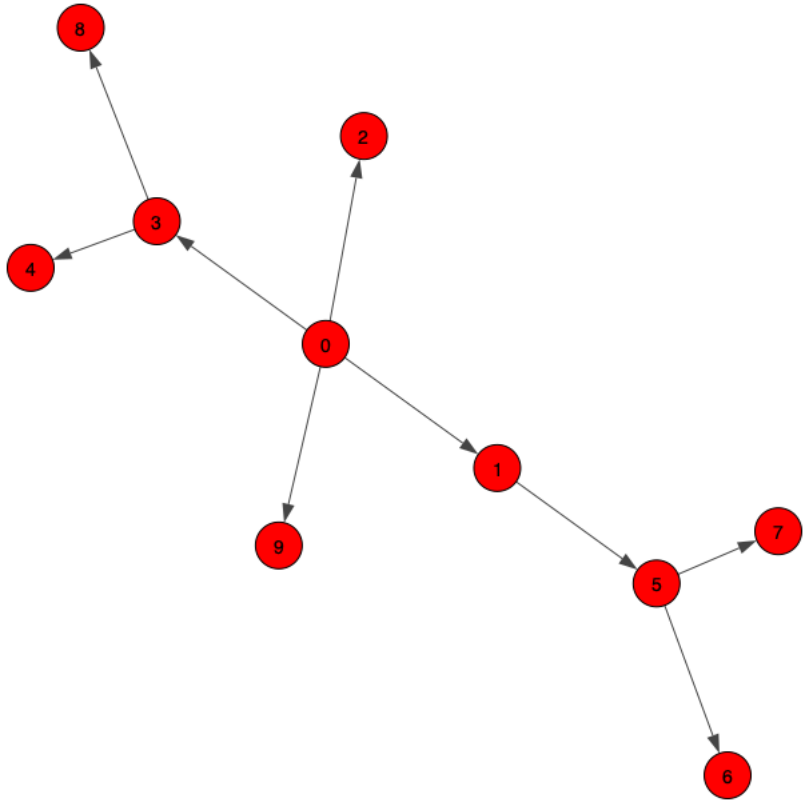


FLUIDOS

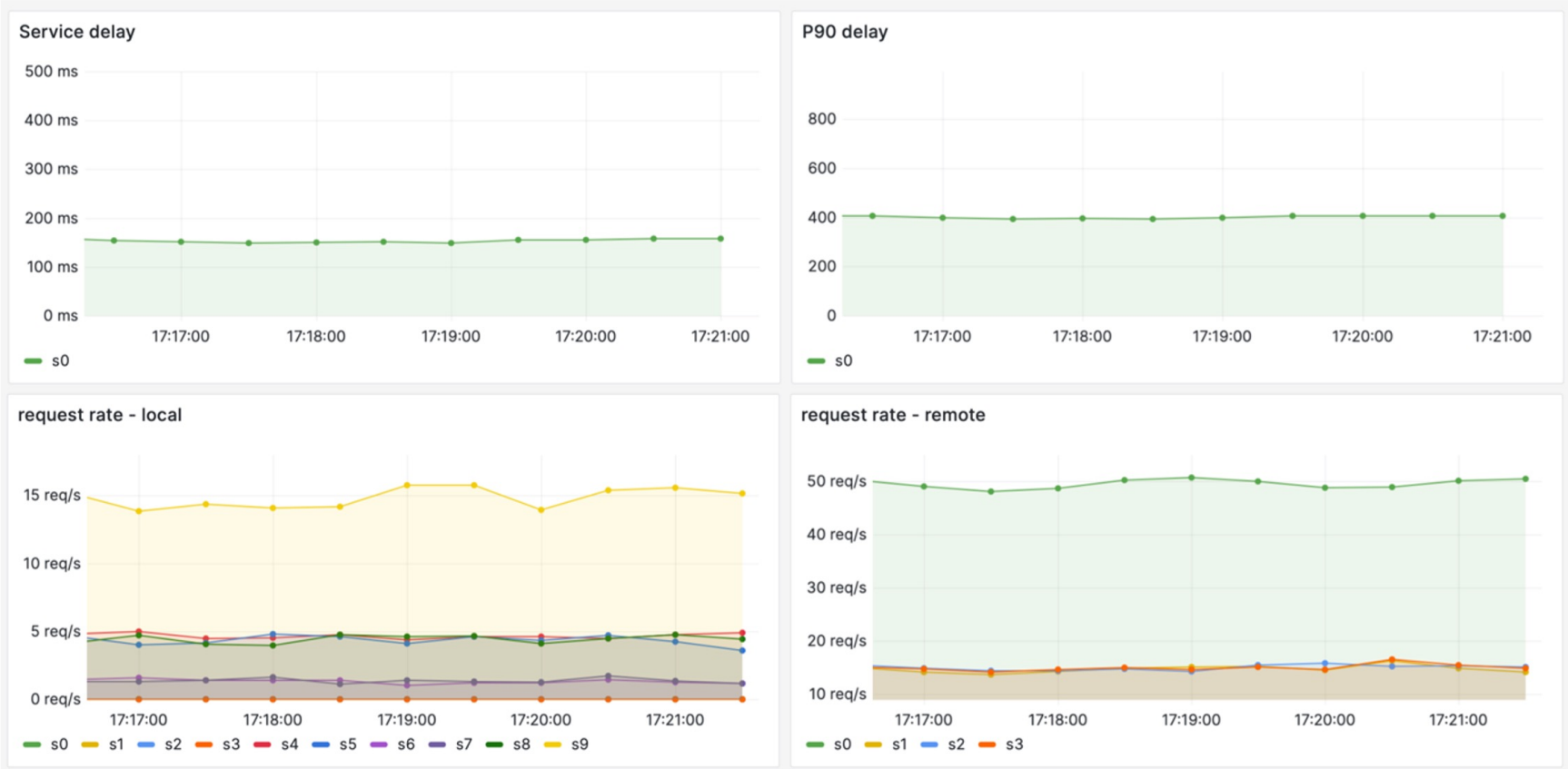
Edge computing results

Testbed: benchmark microservice app

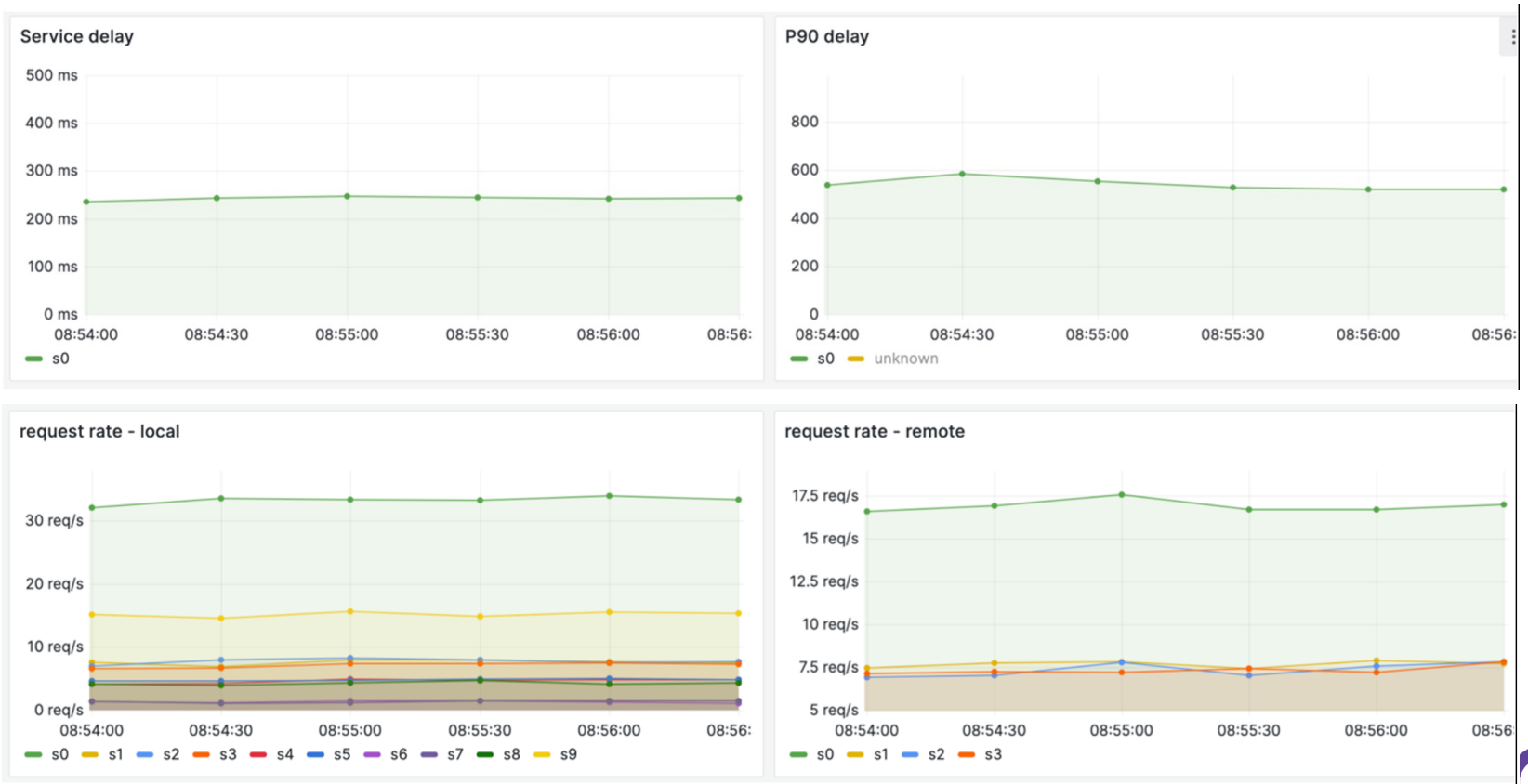
- µBench app
  - https://github.com/mSvcBench/muBench
- Call probability 0.3
- CPU stress only



Testbed: result with offloading of 0,1,2,3



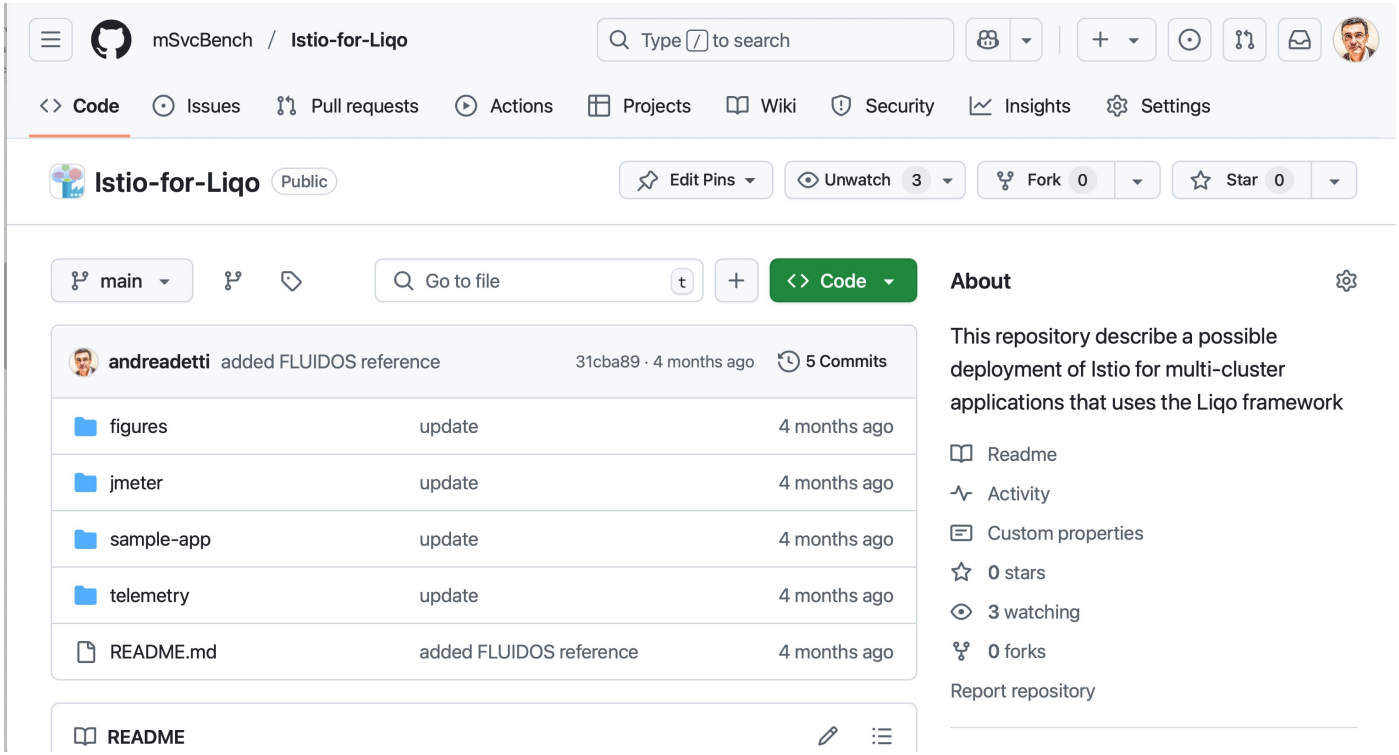
Testbed: result with offloading of 0,1,2,3



Edge-computing without Fluidos-Mesh

Reference

https://github.com/mSvcBench/Istio-for-Liqo



How did FluidosMesh improve FluidOS ?

- **Scalable Deployment of Istio in Multi-Cluster Topologies**
  We present a systematic methodology for deploying the Istio service mesh across federated Kubernetes clusters orchestrated by FluidOS. This approach ensures transparent service discovery, secure inter-cluster communication, and unified observability.

- **Optimized Microservice Packaging for Edge-Aware Routing**
  We propose a microservice packaging and deployment strategy leveraging native Kubernetes and Istio capabilities (e.g., DestinationRule, VirtualService). The goal is to minimize user-perceived latency by placing microservices closer to the edge and enforcing locality-aware traffic policies.