CNCF Project Focus

Episode #3

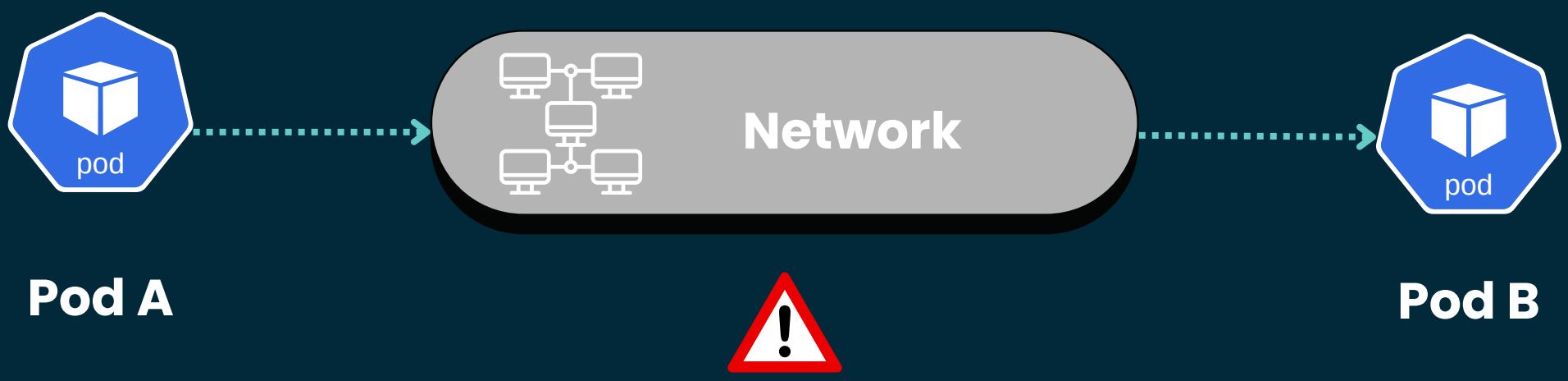
Cilium

Kernel-Native Kubernetes Networking

Christian Dussol

The hidden network overhead

Every Pod-to-Pod call in your Kubernetes cluster



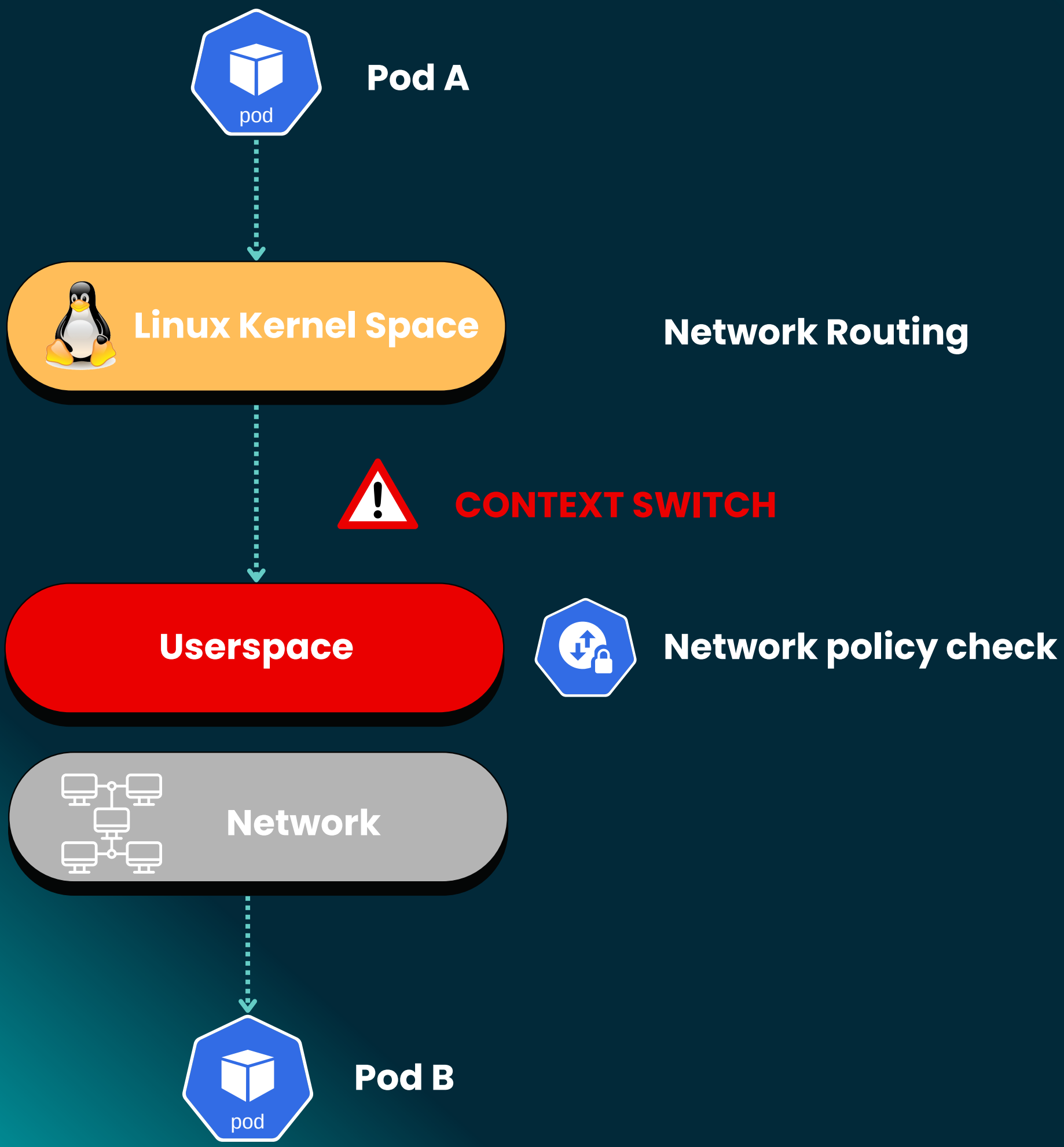
+ 40% Latency

Slower responses

Higher Cost

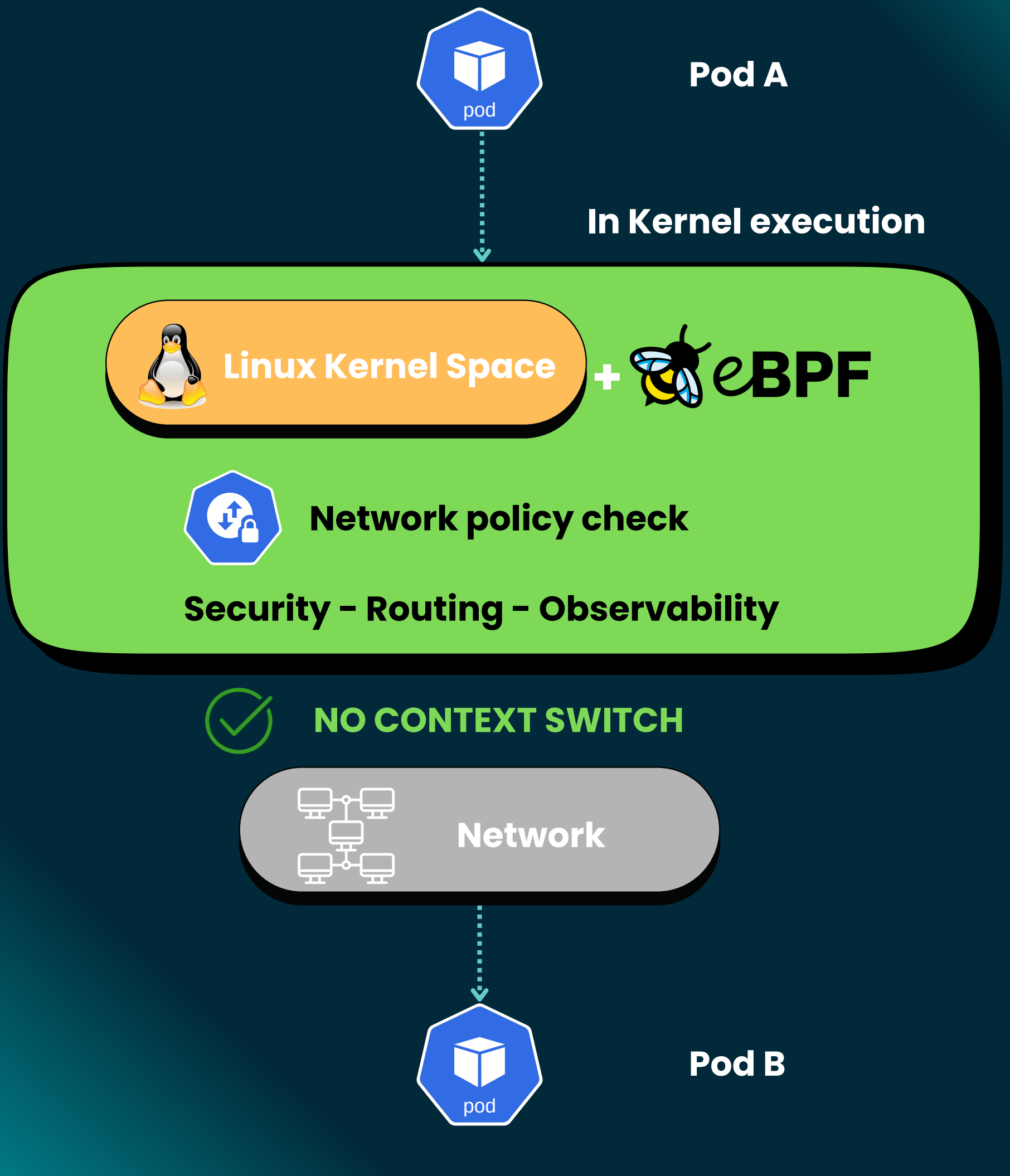
The userspace overhead

Pod-to-Pod communication



Solution : zero context switch

Network policy check happens in Kernel with eBPF



What is Cilium ?

CNCF Graduated project (October 2023)

Core Idea

eBPF-based networking, observability and security

What is eBPF

Extended Berkeley Packet Filter

- Run code in Linux kernel
- Safe, fast, no kernel mods
- Network, security, observability

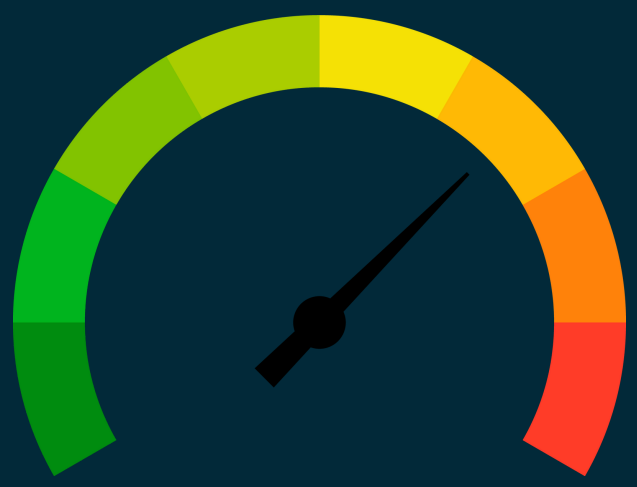
Adopted by

Google aws Microsoft NETFLIX

10+ years of innovation (2014+)

USE CASE 1 : Performance

Kernel-native networking eliminates userspace overhead



- **75% latency reduction vs traditional CNI**
- **85% CPU reduction for network policies**
- **XDP (eXpress Data Path) for 10M+ pps**
- **Efficient connection tracking (conntrack)**

USE CASE 2 : security

Identity-based security with L3-L7 enforcement

- **Identity-based policies (not IP-based)**

  → Services identified by Kubernetes labels
  → IP changes don't break policies

- **API-aware security (L7 filtering)**

  → Allow: POST /api/v1/payment
  → Deny: GET /api/v1/admin

- **Transparent encryption**

  → WireGuard or IPsec

Identity-based security

IP-Based policies ✗

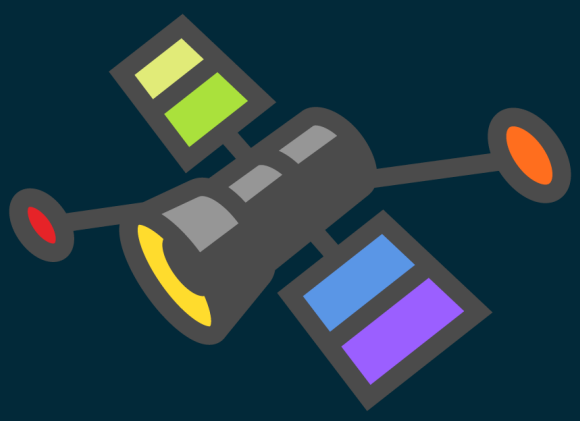
- Fragile (Breaks on pod restart)
- Complex (Manage 1000s of IPs)
- Insecure (IPs can be spoofed)

Identity-Based Policies ✓

- Resilient (Survives pod changes)
- Simple (Label-based rules)
- Secure (Cryptographic identity)

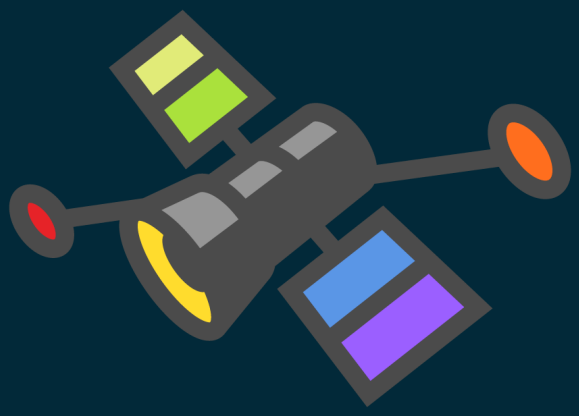
USE CASE 3 : Observability

Deep network visibility with Hubble (eBPF-powered)



Hubble

- Real-time network flow visualization
- Service dependency mapping
- DNS query monitoring
- Metrics: latency, throughput, errors
- Zero instrumentation overhead



Hubble

default | No service selected | View options | Update in 19s

default

spaceship
default
Ingress | Egress
class:tiefighter
io.cilium.k8s.policy.cluster:default
io.cilium.k8s.policy.serviceaccount:default
org:empire

spaceship
default
Ingress | Egress
class:xwing
io.cilium.k8s.policy.cluster:default
io.cilium.k8s.policy.serviceaccount:default
org:alliance

http:// **deathstar**
default
Ingress | Egress
80 TCP · HTTP
class:deathstar
io.cilium.k8s.policy.cluster:default
io.cilium.k8s.policy.serviceaccount:default
org:empire

Filter labels key=val, ip=0.0.0.0, dns=google.com | Flows | Policies | All Statuses | HTTP Status | Columns

Source Pod Na...	Source Service	Destination Pod...	Destination Ser...	Destination IP	Destination Port	Destination L7 I...	Status	Last Seen
tiefighter	class=tiefighter...	deathstar-5b7489bc...	10.0.1.42 *default*	10.0.1.42	TCP:80		forwarded	2 minutes ago
deathstar-5b7489bc...	class=deathstar...	tiefighter	10.0.2.42 *default*	10.0.2.42	TCP:56086		forwarded	2 minutes ago
xwing	class=xwing *default*	deathstar-5b7489bc...	10.0.2.64 *default*	10.0.2.64	TCP:80		forwarded	2 minutes ago

Explore my learning toolkit

github.com/christian-dussol-cloud-native/cilium/

📚 Educational GitHub Repository

Complete hands-on tutorial, including:

- ✅ **Quick start**
  Cluster setup, Cilium installation & verification
- ✅ Network policies (L3/L4/L7)
- ✅ Hubble observability
- ✅ Kyverno governance
- ✅ Payment API demo (PCI-DSS compliant)