Is Your Virtual Machine Really Ready-to-go with Istio?

Kailun Qin, Intel
Haoyuan Ge

#IstioCon

Quick Summary (from Google Cloud Next ’19 [1])

VM works on Istio!

[1] Istio Service Mesh for VM Native, Chris Crall, Jianfei Hu, Google Cloud Next ‘19

Why Add VMs to the Mesh?

- = Why Service Mesh?
  - More services = more complexity
  - Need consistent policy enforcement
  - Need consistent metrics aggregation
- Traffic management
  - Load balancing for VMs, failover, A/B testing, modern rollouts for VM services
- Security
  - Enforce the same policies in the same way, across compute environments
- Observability
  - See VM metrics alongside containers
- Extensibility

Why Should Istio Support VMs

- ≈ Why VMs?
  - Technical reasons
    - Better known security controls
    - Better isolation (of resources, fault domains etc.)
    - Compatibility (non-Linux, unikernels)
  - Business reasons
    - Legacy applications
    - Deterministic workloads with strong requirements
- For Istio
  - What is Istio? A service mesh. But more: an open service platform!
  - More use cases!
  - (Consul, Kuma...)

Emerging Use Cases

Telco & Edge Computing

Artifical Intelligence

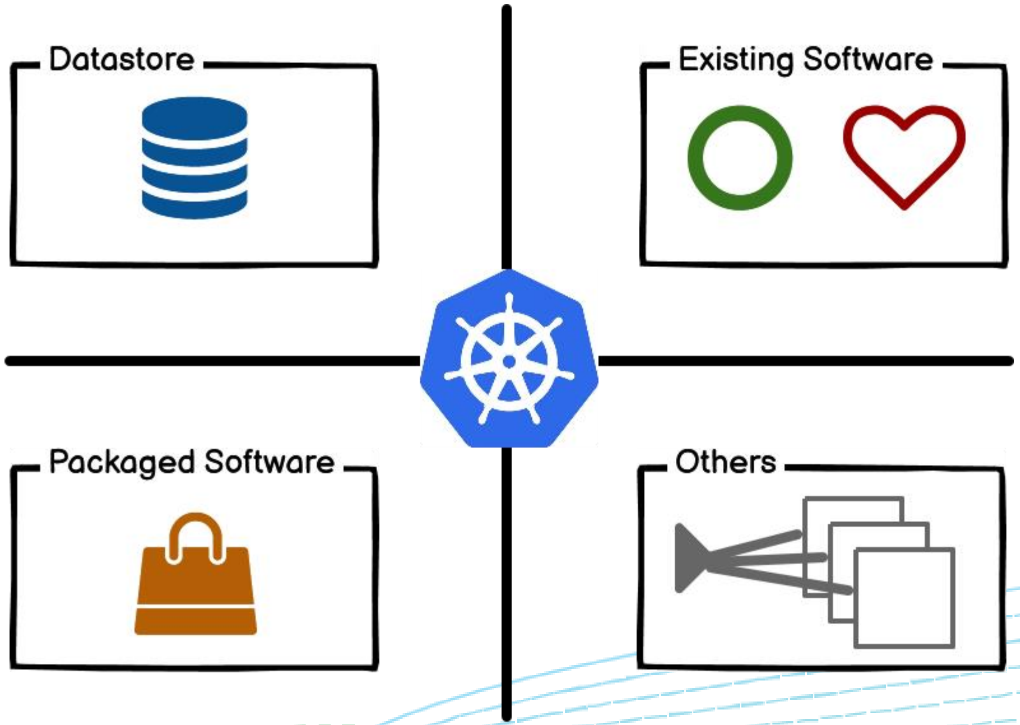
Next-gen Financial Services

Block Chain

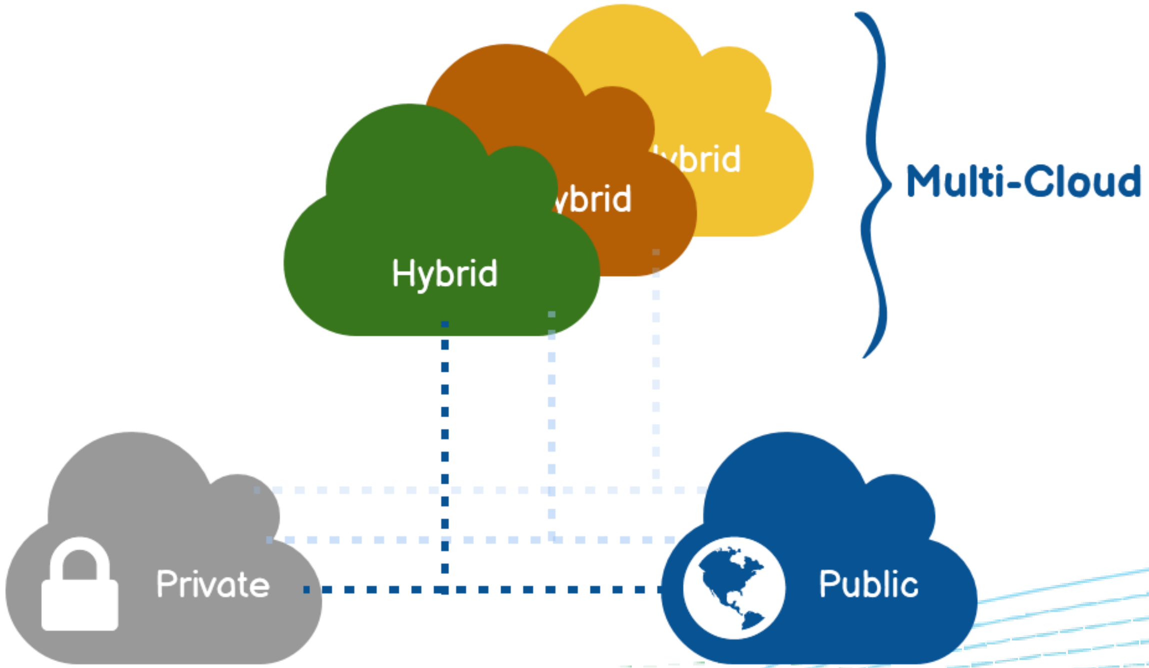
AR/VR

Legacy Scenarios

- Stateful applications
  - Data store
- Legacy software
  - Financial services
  - Enterprise/Workshop applications
  - Hard to lift and shift
- Packaged software
  - Non-Linux
  - unikernels
- Domain specific workloads
  - Network Functions (NFV)

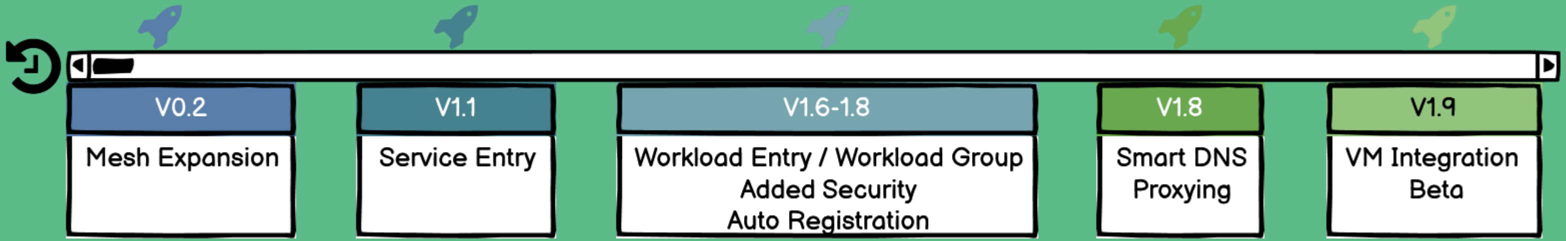


Hybrid and Multi Clouds



Istio VM Integration is?

A Tumultuous Odyssey...

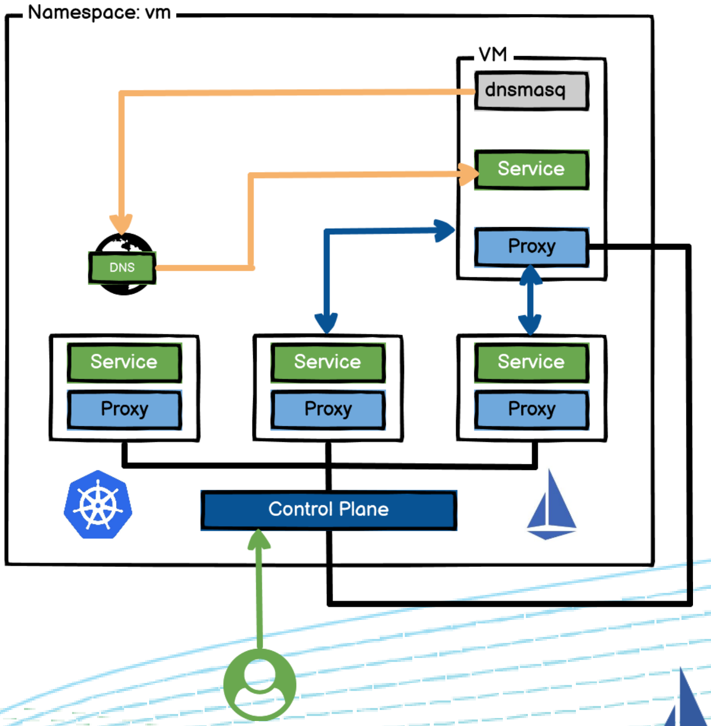


#IstioCon

[1] Istio 1.8: A Virtual Machine Integration Odyssey, Jimmy Song

V0.2 Mesh Expansion

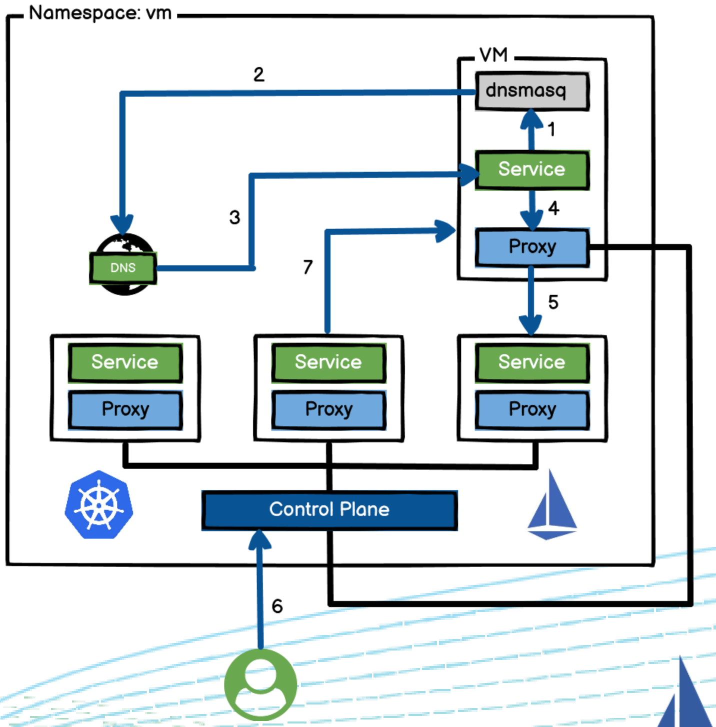
- Prerequisites
  - IP connectivity to the endpoints in the mesh
  - Istio control plane services (Pilot, Mixer, CA) accessible from the VMs
  - (optional) Kubernetes DNS server accessible from the VMs
- Onboard steps
  - Setup Internal Load Balancers (ILBs) for Kube DNS, Pilot, Mixer and CA
  - Generate configs for VMs, incl. `cluster.env`, DNS config, Istio authN secrets etc.
  - Setup dnsmasq, Istio components in the VM and verify functionality
  - Configure sidecar interception; restart Istio and manually register the services running



V0.2 Mesh Expansion (cont.)

- Traffic flow (VM -> Container)
  1. Dnsmasq accepts DNS queries
  2. Access the built-in Kube DNS (exposed by ILB)
  3. Obtain the Cluster IP resolved
  4. Traffic intercepted by the sidecar proxy
  5. xDS
     - Traffic forwarded to ingress in the mesh
- Traffic flow (Container -> VM)
  1. Manual registration

  istioctl -n onprem register mysql 1.2.3.4 3306



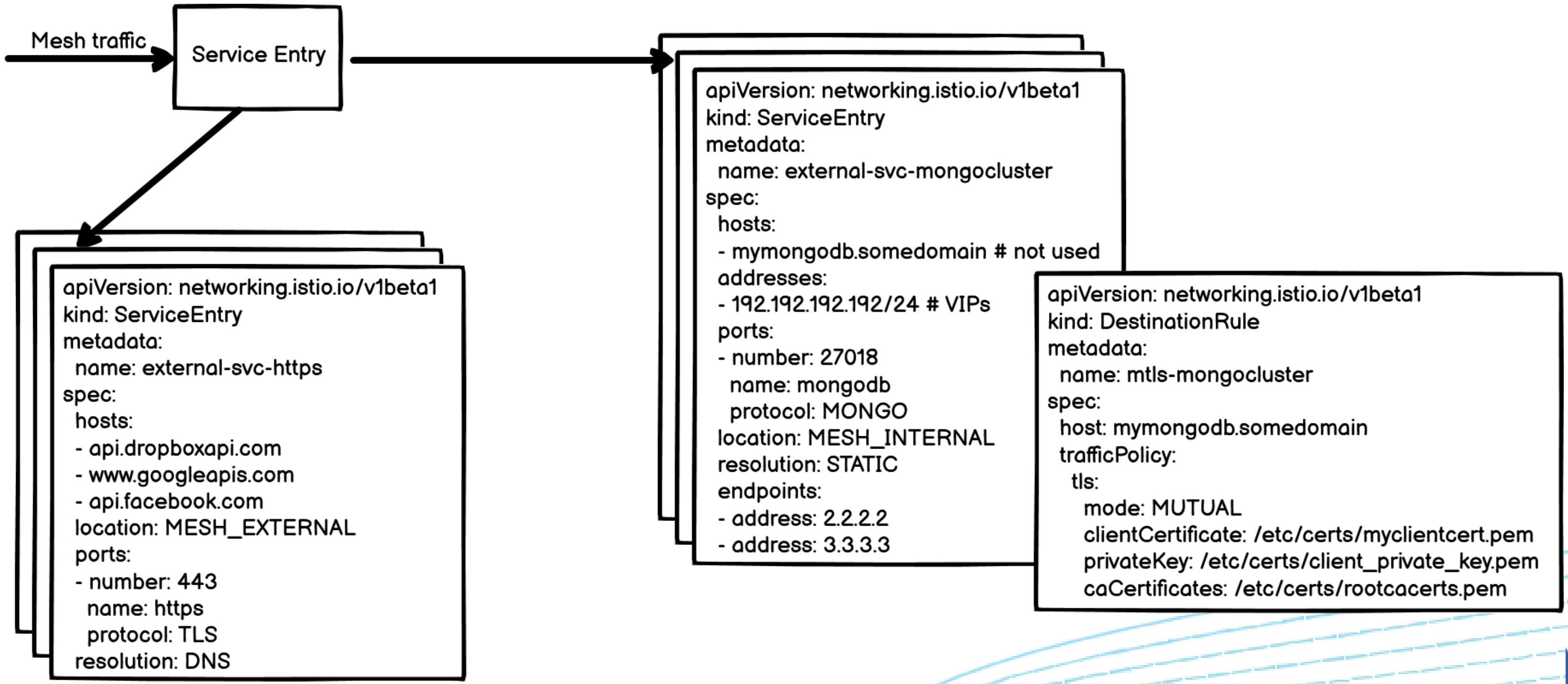
V1.1 Introducing Service Entry

Service Entry v.s. Service v.s. Endpoints

- Service Entry
  - An entry that Istio maintains internally
  - Describing the properties of a service, internal/external to the mesh
    - DNS name
    - VIPs, ports, protocols
    - Endpoints
  - After adding, sending traffic to the service as if it was a service in your mesh
    - Traffic redirect and forward
    - Retry, timeout, fault injection, mtls policies
    - VM service, multicluster Istio mesh support

- Service + Endpoints
  - Usually for internal traffic
  - ExternalName
    - Service <-> DNS name
  - External IPs

V1.1 ServiceEntry

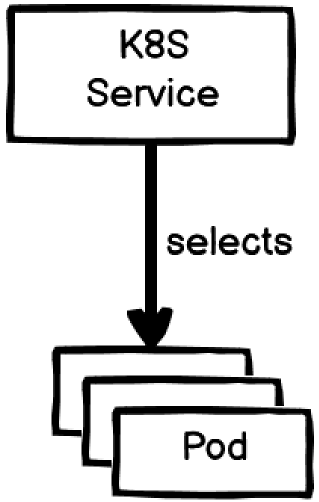


```yaml
apiVersion: networking.istio.io/v1beta1
kind: ServiceEntry
metadata:
  name: external-svc-https
spec:
  hosts:
  - api.dropboxapi.com
  - www.googleapis.com
  - api.facebook.com
  location: MESH_EXTERNAL
  ports:
  - number: 443
    name: https
    protocol: TLS
  resolution: DNS
```

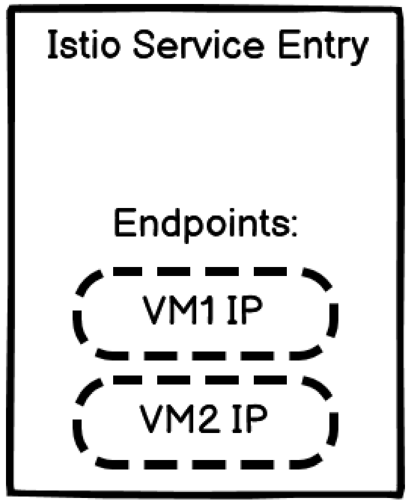
```yaml
apiVersion: networking.istio.io/v1beta1
kind: ServiceEntry
metadata:
  name: external-svc-mongocluster
spec:
  hosts:
  - mymongodb.somedomain # not used
  addresses:
  - 192.192.192.192/24 # VIPs
  ports:
  - number: 27018
    name: mongodb
    protocol: MONGO
  location: MESH_INTERNAL
  resolution: STATIC
  endpoints:
  - address: 2.2.2.2
  - address: 3.3.3.3
```

```yaml
apiVersion: networking.istio.io/v1beta1
kind: DestinationRule
metadata:
  name: mtls-mongocluster
spec:
  host: mymongodb.somedomain
  trafficPolicy:
    tls:
      mode: MUTUAL
      clientCertificate: /etc/certs/myclientcert.pem
      privateKey: /etc/certs/client_private_key.pem
      caCertificates: /etc/certs/rootcacerts.pem
```

V1.6-1.8 Better VM Workload Abstraction



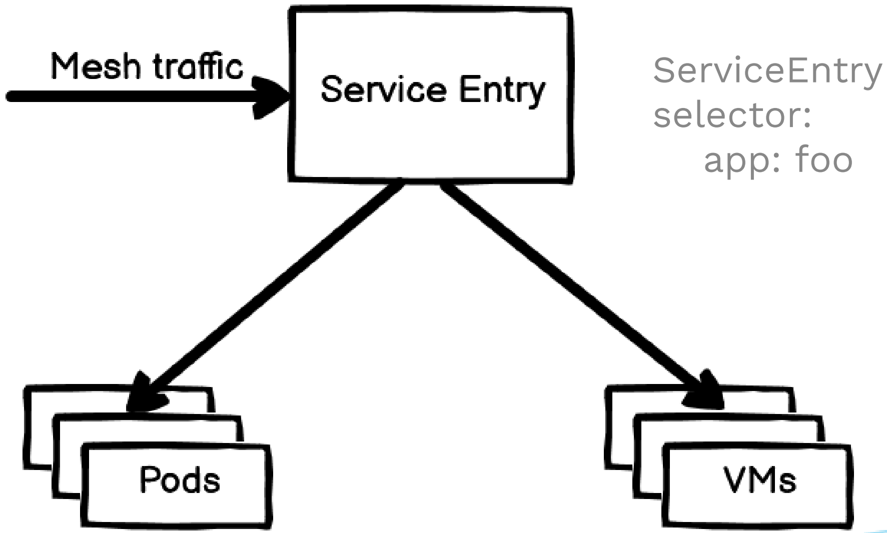
A K8s Service and Pods
Two separate object with distinct lifecycles



Before Workload Entry, a single Istio Service Entry object combined the lifecycles of both the service and the workloads implementing it, w/o giving a first-class representation for the workloads themselves

V1.6-1.8 Better VM Workload Abstraction

Item	**Kubernetes**	**Virtual Machine**
Basic schedule unit	Pod	WorkloadEntry
Component	Deployment	WorkloadGroup
Service registry and discovery	Service	ServiceEntry



ServiceEntry
selector:
    app: foo

K8s Pods
labels:
    app: foo
    class: pod

Istio Workload Entries
labels:
    app: foo
    class: vm

#IstioCon

V1.6-1.8 Better VM Workload Abstraction

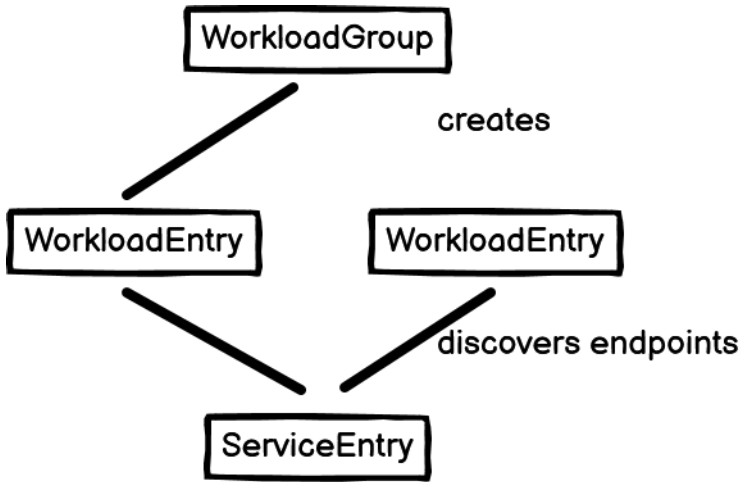
- Workload Entry
  - single non-Kubernetes workload
  - mTLS using service account
  - work with an Istio ServiceEntry
- Workload Group
  - a collection of non-K8s workloads
  - metadata and identity for bootstrap
  - mimic the sidecar proxy injection
  - automate VM registration
  - health/readiness check

V1.7 VM Support with Added Security

- Secure bootstrapping process
  - Automate provisioning a VM's mesh identity (certificate)
    - based on a platform-specific identity
    - w/o a platform-specific identity
      - using a short-lived K8s service account token
- Automatic certificate rotation
- Validation of the proxy’s status for VM-based workloads

V1.8 VM Auto Registration

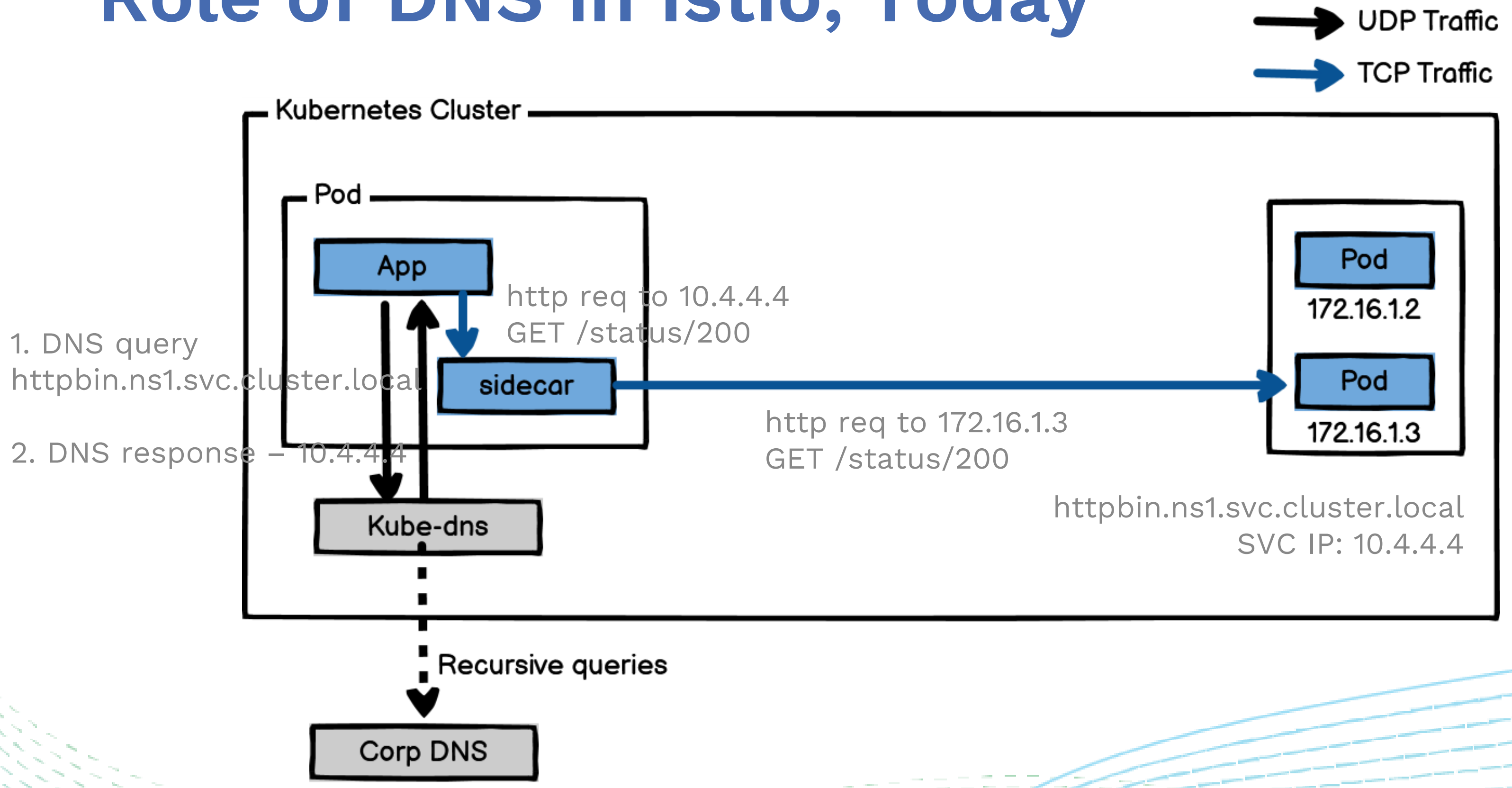
- Experimental
- Auto-scaling
- Automatically add a WorkloadEntry for a VM instance that connects with a valid identity token
- All we have to do is
    - specify a new WorkloadGroup with a template (to create WorkloadEntry)
    - create a ServiceEntry (to select specific workloads)



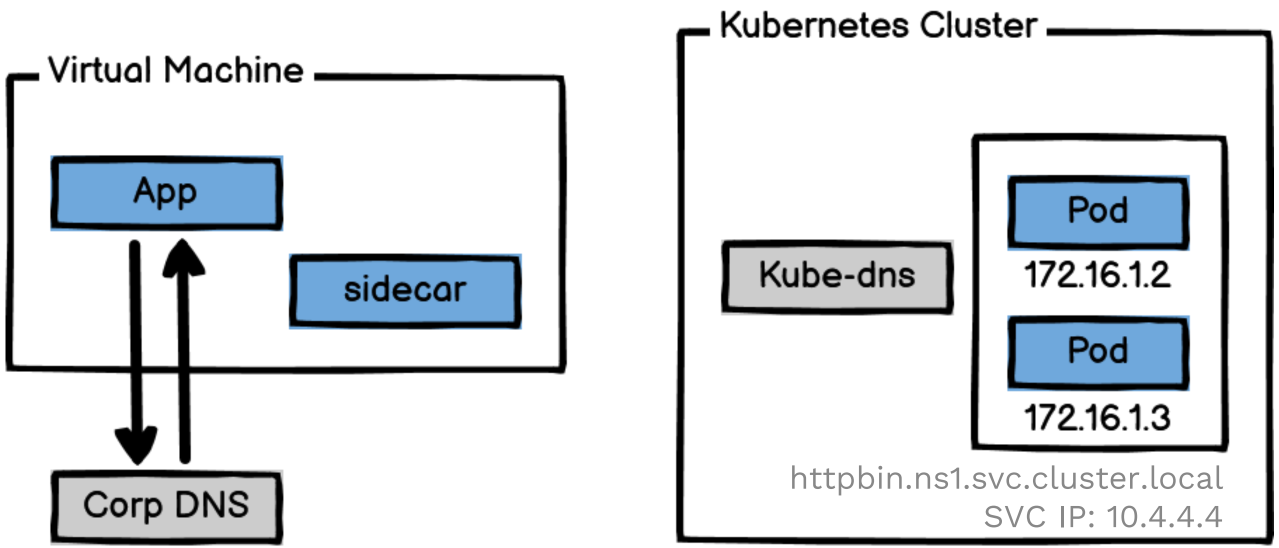
What Else Did Not Solve?

- VM access to K8s services
  - needs convoluted workarounds
  - exposes security risks
- External TCP services without VIPs
  - no way to distinguish traffic listening on the same port
  - workaround: `resolution: NONE`
- Resolving DNS for services in remote clusters

Role of DNS in Istio, Today



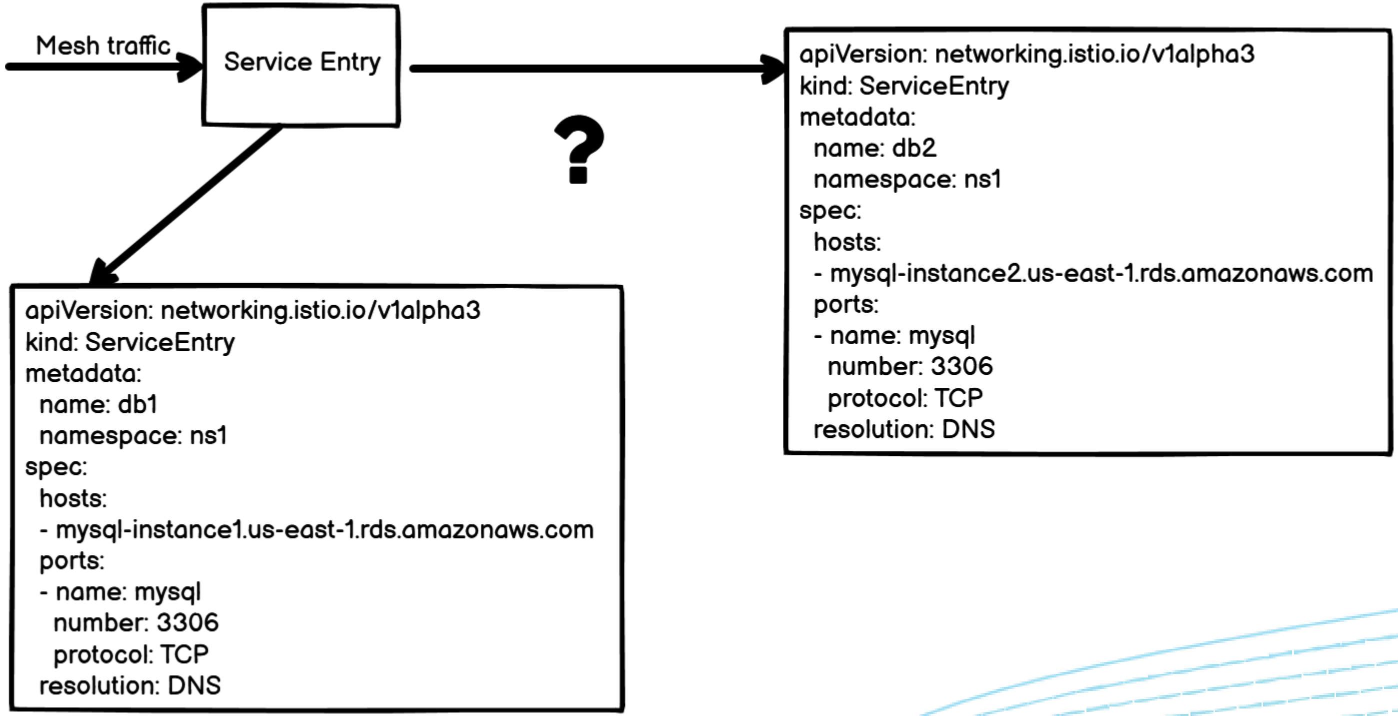
DNS Issues on VMs accessing K8s SVCs



1. DNS query for httpbin.ns1.svc.cluster.local

2. DNS response – no such host

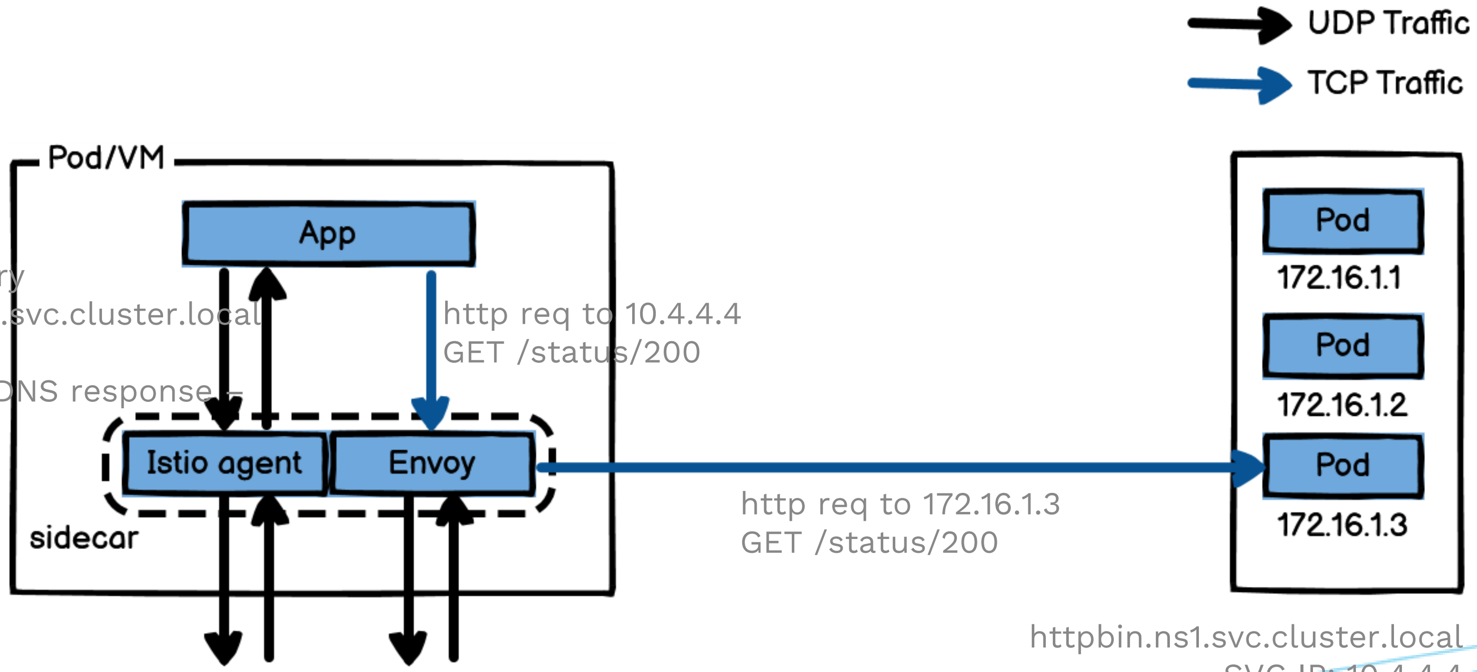
DNS Issues on ext-TCP SVCs without VIPs



```
apiVersion: networking.istio.io/v1alpha3
kind: ServiceEntry
metadata:
  name: db1
  namespace: ns1
spec:
  hosts:
  - mysql-instance1.us-east-1.rds.amazonaws.com
  ports:
  - name: mysql
    number: 3306
    protocol: TCP
  resolution: DNS
```

```
apiVersion: networking.istio.io/v1alpha3
kind: ServiceEntry
metadata:
  name: db2
  namespace: ns1
spec:
  hosts:
  - mysql-instance2.us-east-1.rds.amazonaws.com
  ports:
  - name: mysql
    number: 3306
    protocol: TCP
  resolution: DNS
```

Smart DNS Proxying



1. DNS query httpbin.ns1.svc.cluster.local

2. Cached DNS response – 10.4.4.4

http req to 10.4.4.4
GET /status/200

http req to 172.16.1.3
GET /status/200

httpbin.ns1.svc.cluster.local
SVC IP: 10.4.4.4

DNS queries to the system configured name servers. Envoy does not use the agent's DNS cache.

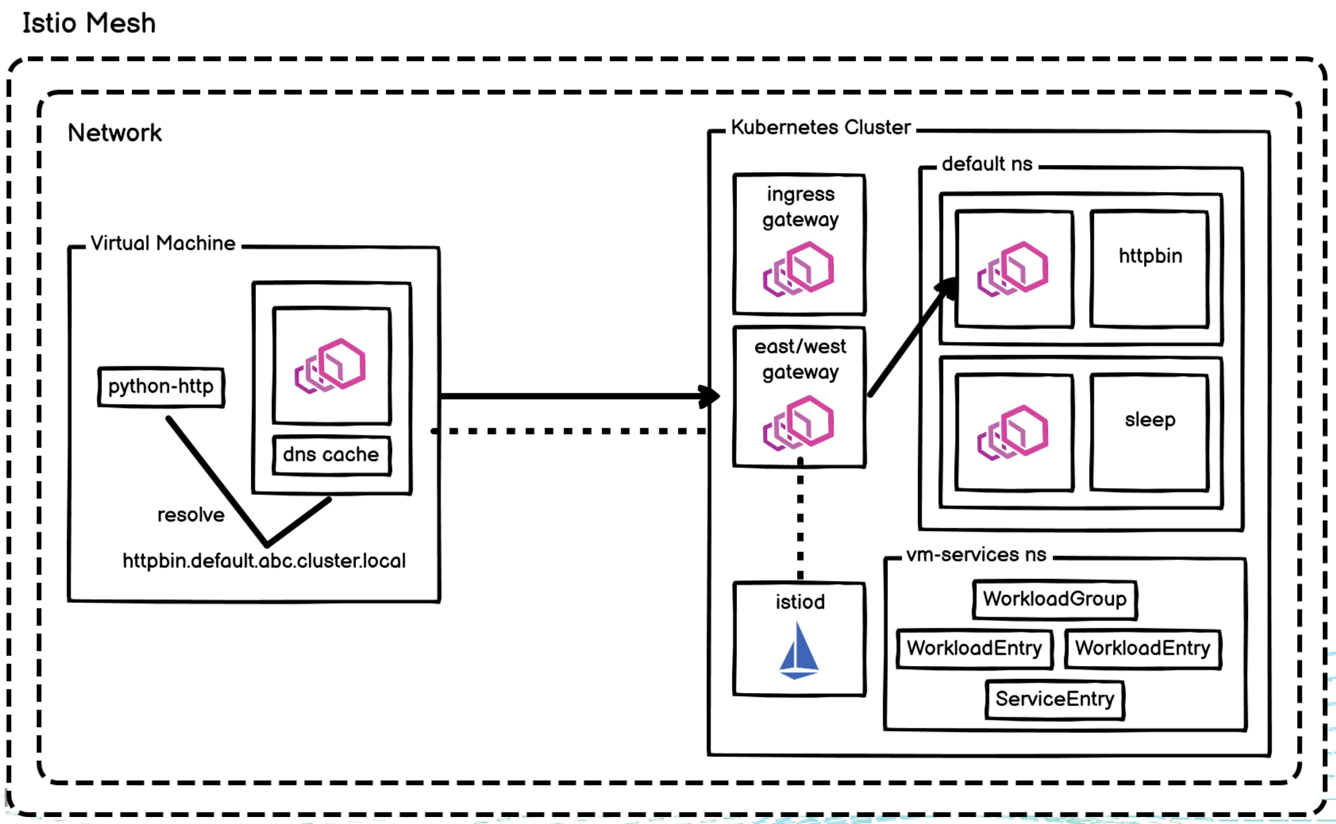
V1.8 Smart DNS Proxy: A Step Further

- Taking control of DNS!
    - VMs to Kubernetes integration
    - Reduced load on your DNS servers w/ faster resolution
    - Automatic VIP allocation where possible
    - Multicluster DNS lookup

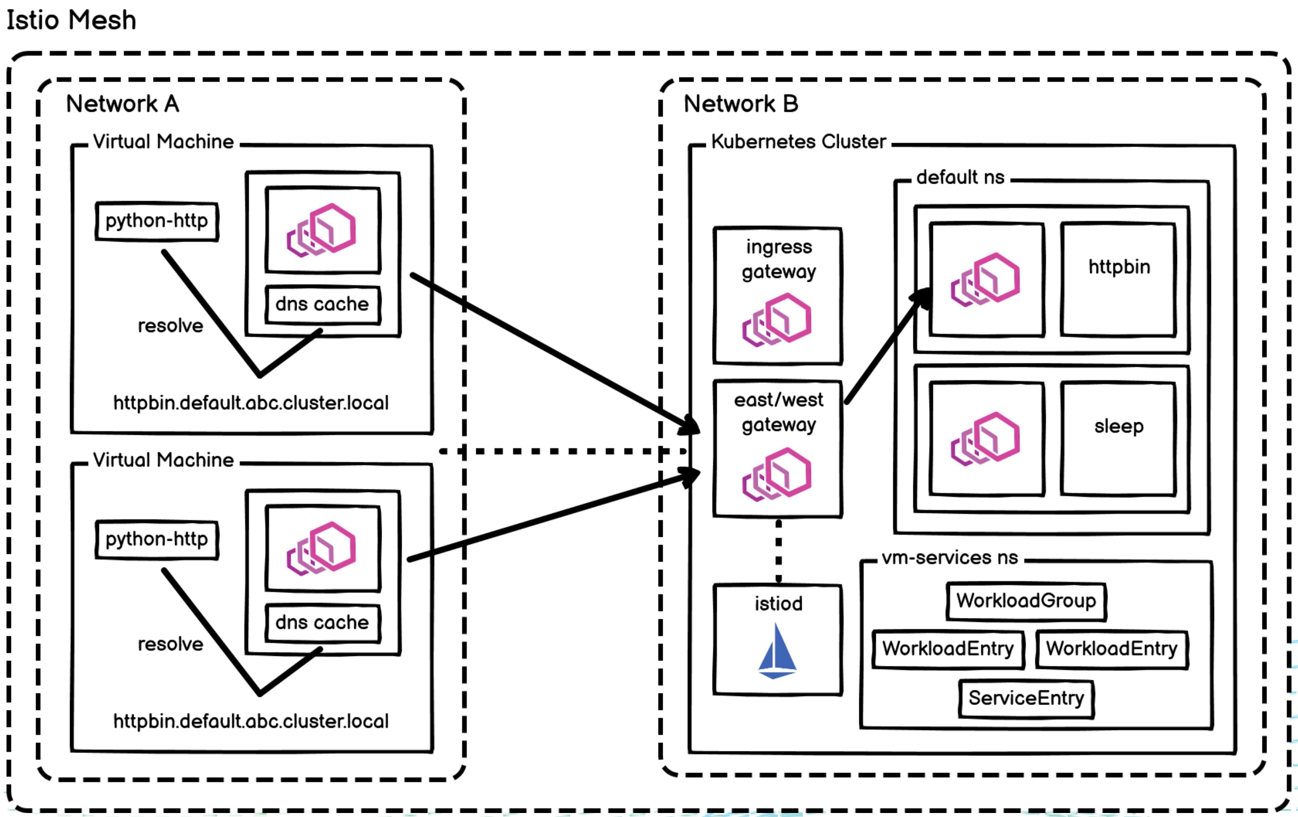
V1.9 VM Integration, Beta!

- DNS_AUTO_ALLOCATE
  - Decoupled from DNS_CAPTURE
- Documents available
  - Virtual Machine Installation to get started.
  - Virtual Machine Architecture to learn about the high level architecture of Istio's virtual machine integration.
  - Debugging Virtual Machines to learn more about troubleshooting issues with virtual machines.
  - Bookinfo with a Virtual Machine to learn more about connecting virtual machine workloads to Kubernetes workloads.

VM Support – Single Network



VM Support – Multiple Networks



Current State of VM Support

- Traffic flow
  - VM connects up to the Istio control plane through a Gateway
  - WorkloadEntry created
    - VM sidecar is made aware of all services in the cluster
  - DNS name resolved
    - gets routed through the gateway to the service
- The data plane traffic
    - Single network
      - direct communication w/o requiring intermediate Gateway
    - Multiple networks
      - all goes though the Gateway
      - via L3 networking (if enhanced performance is desired)

Istio VM integration seems closer to be production ready?

Should we expect more?
And what do we need else?

Why We Expect More? A Closer Look...

- Example use case: Telco & Edge computing
  - where VMs play a crucial role now and later
  - where service mesh is a key paradigm for solving challenges [1]
    - Traffic steering (network slicing)
    - Fault injection (resilience of the app)
    - Circuit detection and outlier detection (reliability) etc.
    - Pervasive security (via mtls)
    - Extensibility (to cherry pick extensions)

Key Drivers [1]

Latency Sensitive Solutions

Privacy and Security

Right Sized Bandwidth

New Business Outcomes

[1] Service Mesh use cases for Telco and Edge – Google, ServiceMeshCon NA 2020

What Do We Need Else to Augment Istio?

- Strong security and privacy guarantees
    - Confidentiality, integrity and privacy protection for sensitive data
    - Strong isolation for multi-vendor services
    - End-to-end security! (not just between middle boxes)
- High performance networking
    - Much higher multi-Gbps peak data speeds
    - Ultra low latency
    - And of course, reduce overheads introduced!
- High availability
- CapEx, OpEx

Security & Usability Limitations

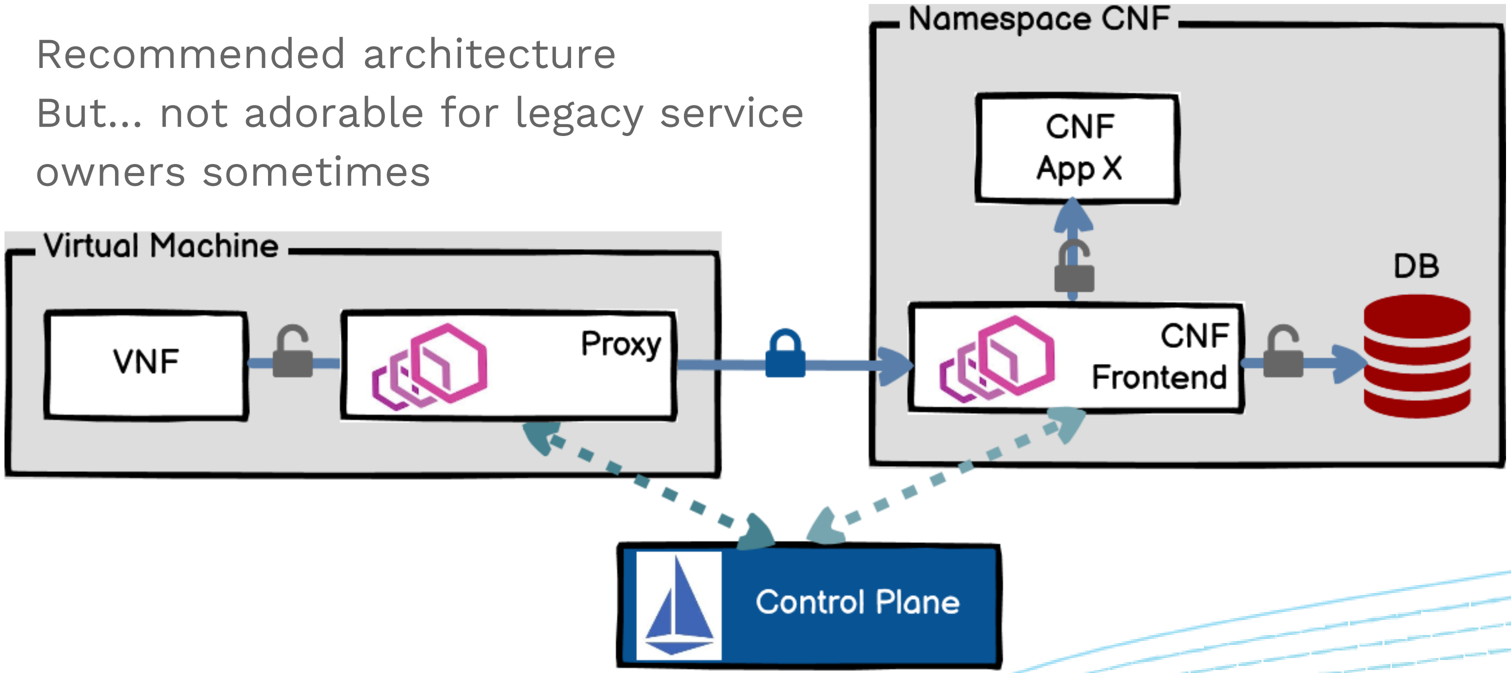
- VM cert provisioning (exp. for on-prem VMs)
  - Alternative opts
  - Current: Fetch and exchange a k8s token for a bootstrap certificate, then place that bootstrap certificate on the VM
    - Dependency on K8s API server
    - Requires creating an RBAC impersonation rule for each user
    - Private key and CSR generation limited to Istio agent (no support of other provisioner tools and HSM incompatible)
    - Limitations to audit (proactively secure)
- VM cert extensibility
  - No support for workload certificate attributes

Security & Usability Limitations (cont.)

- Access management: CNI needs improvements
  - Much required to avoid escalated Pod privileges
  - No support for smart DNS proxying (yet...)
- Further security middle boxes support
  - Deep packet inspection (DPI)
  - DDoS defense
  - Firewall
- Lack dedicated gateway support (architectural changes)
  - No separating out the gateway used for untrusted user traffic from the internal mesh traffic
  - One of the viable solutions to communicate between Legacy VNFs and new CNFs
- Need a stricter security model for end-to-end key protection

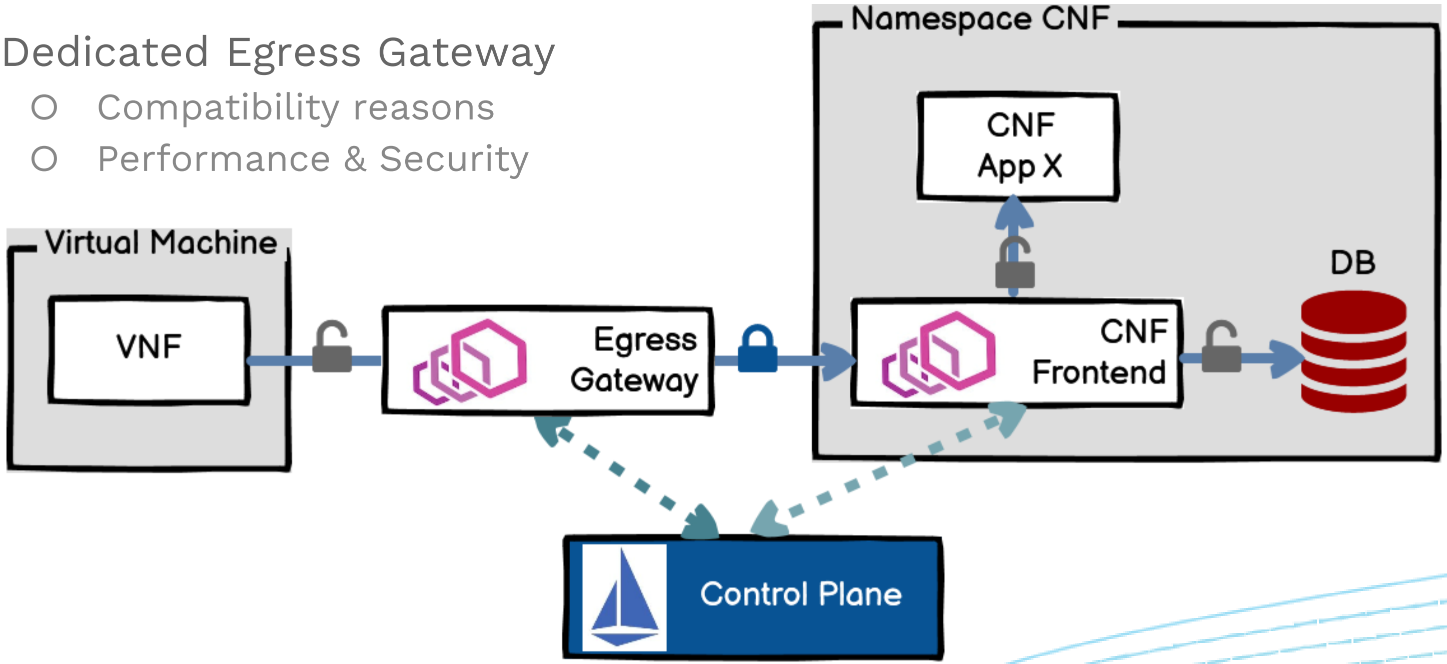
Legacy VNF ⇔ CNF: Option 1

- Recommended architecture
- But... not adorable for legacy service owners sometimes



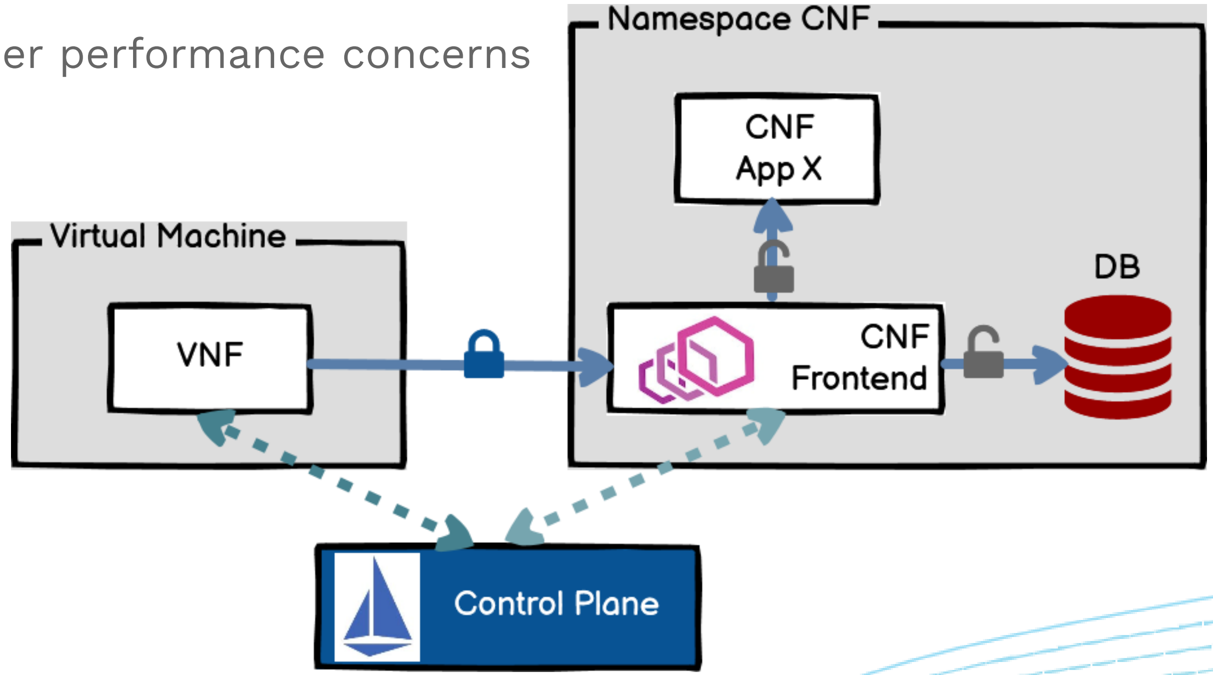
Legacy VNF ⇔ CNF: Option 2

- Dedicated Egress Gateway
  - Compatibility reasons
  - Performance & Security



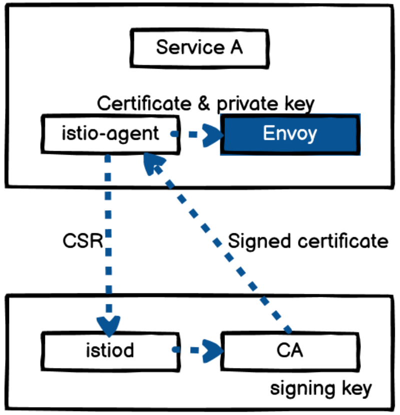
Legacy VNF ⇔ CNF: Option 3

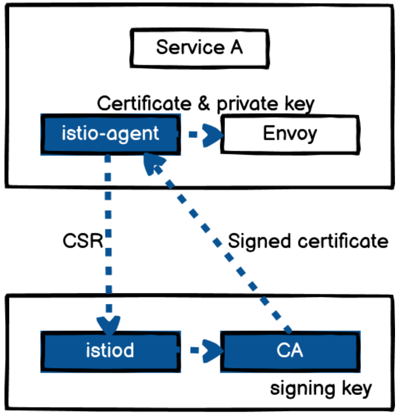
- Further performance concerns



End-to-end Key Protection

- SDS (Secret Discovery Service)
- A stricter security model
  - Protections for inline components & workflows
  - Trust model augmentation
    - Impersonating
    - Secret clear in memory
    - Secret persistence
- Key protection
  - Private key for TLS
  - Signing key
  - ...





Performance Limitations

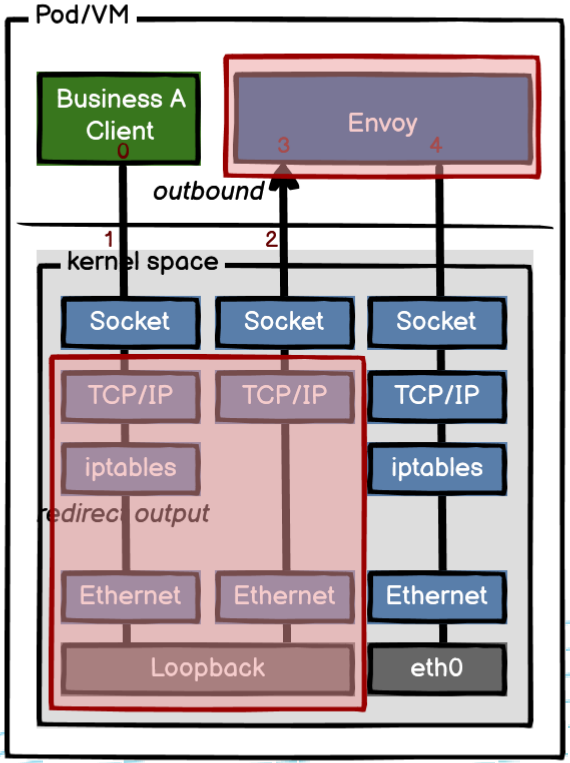
- Some not just limited on VMs, but
  - need to be extended to VMs
  - and much more demanding for some VM use cases (w/ strict requirements)
- No first-class support for VM Multiple Networks
  - All traffic goes though the Gateway
  - Need to setup L3 networking if enhanced performance is desired
- Overheads introduced
- No high performance data path support
  - Multi-Gbps bandwidth
  - Ultra low latency

Performance Limitations: Solutions

- Software techniques
  - (eBPF-based) TCP/IP stack bypass
  - HTTP/3 & QUIC
- Hardware acceleration technologies
  - SRIOV/DPDK
  - Networking/Security offloading
- Hybrid solutions using SW-HW co-designs

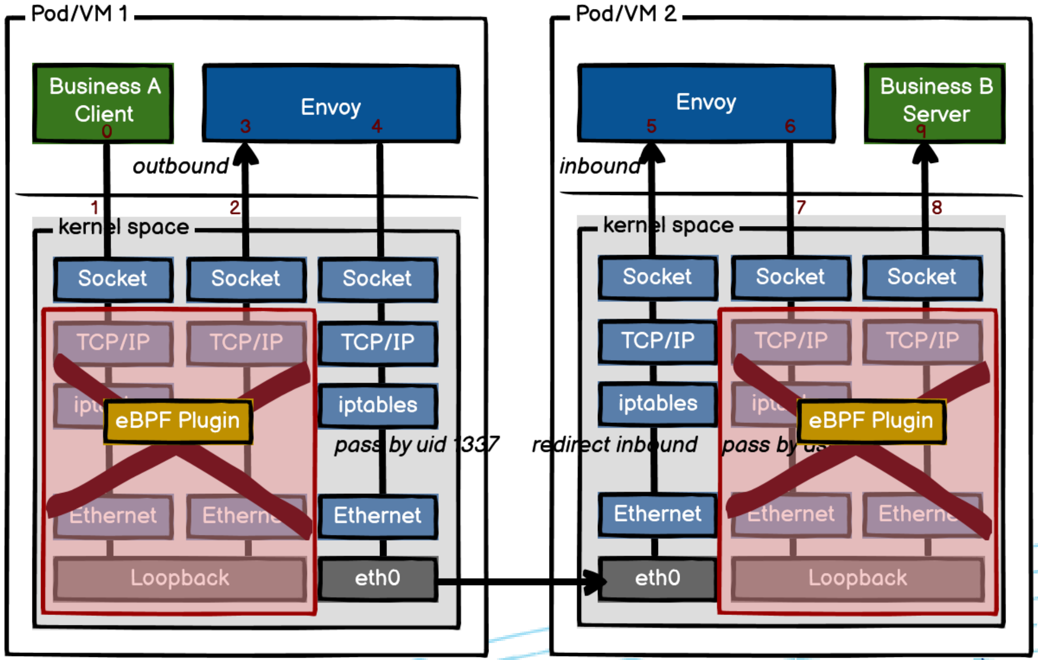
Latency Analysis

- ~3ms P90 latency added
  - Istio v1.6
  - More for VM usage
- Hotspots
  - 1 ⇔ 2
  - 3 ⇔ 4: 30%~50%
- Others
  - Latency between Pods
  - Latency introduced by C/S



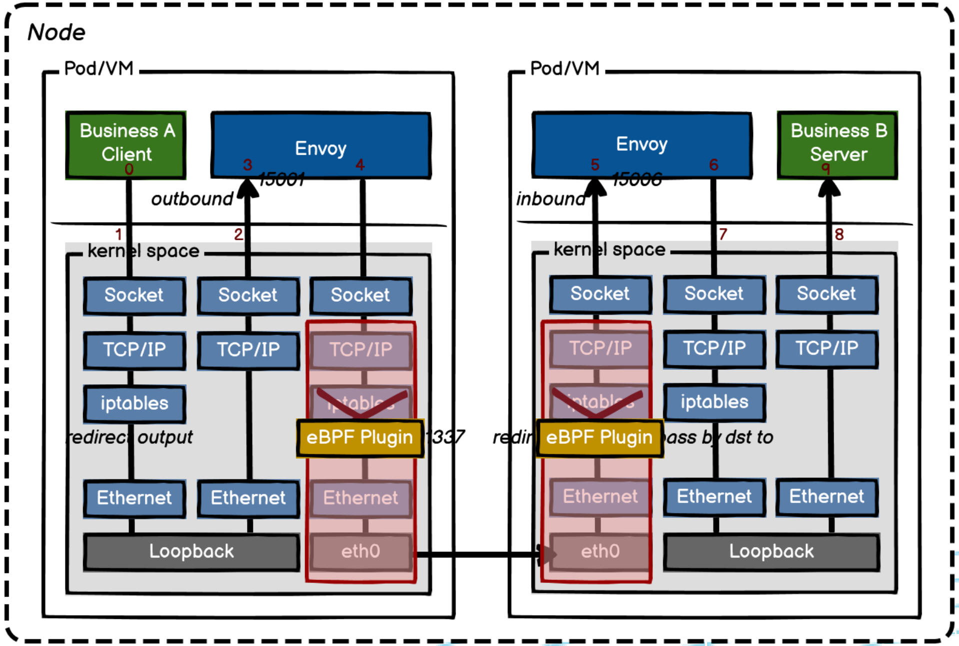
(eBPF-based) TCP/IP Stack Bypass

- eBPF
  - In-kernel virtual machine
  - Running user code in kernel space safety
  - Tracing, security
  - Networking
- Hooks
  - sock_ops
    - Construct map
  - sk_msg_md
    - Match & redirect
- ~5% improvements



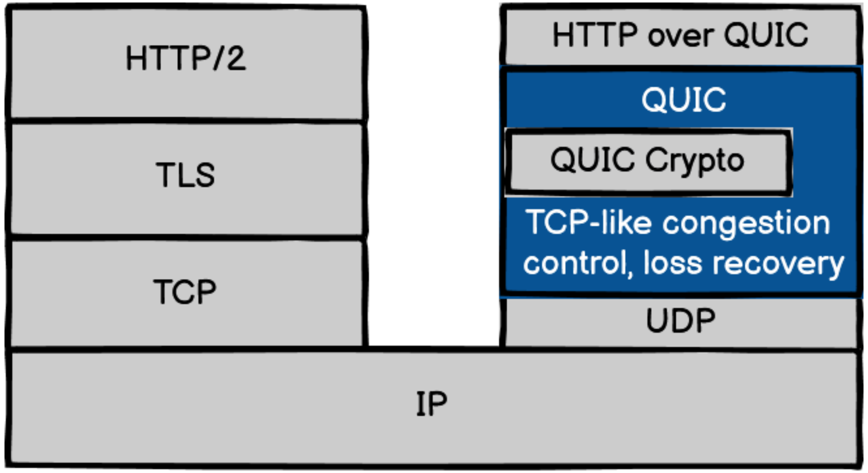
TCP/IP Stack Bypass (cont.)

- Leverage eBPF
- Target Pod/VMs on the same node
- Use case: edge computing
  - Limited number of nodes
  - More traffic across Pod/VMs on the same node



QUIC

- A new transport protocol
- A little like TCP + TLS, but build on top of UDP
  - Uses UDP like TCP uses IP
  - Adds connections, resends and flow control on top
  - Provides independent streams
    - Extremely similar to HTTP/2, but in transport layer
- Improvements
  - TCP head of line blocking
  - Faster handshakes
  - Earlier data
  - Connection-ID
  - More encryption, always



[1] Http3 Full Stack Fest, Daniel Stenberg

HTTP/3

- HTTP/3 = HTTP over QUIC
- Application protocol over QUIC
- HTTP – same but different
  - HTTP/1 in ASCII over TCP
  - HTTP/2 – binary multiplexed over TCP
  - HTTP/3 –binary over multiplexed QUIC
- Faster!
  - Handshakes
  - Early data
  - Independent streams

	HTTP/2	**HTTP/3**
Transport	TCP	QUIC
Streams	HTTP/2	QUIC
Clear text version	Yes	**No**
Independent streams	No	**Yes**
Header compression	HPACK	QPACK
Server push	Yes	Yes
Early data	In theory	**Yes**
0-RTT Handshake	No	**Yes**

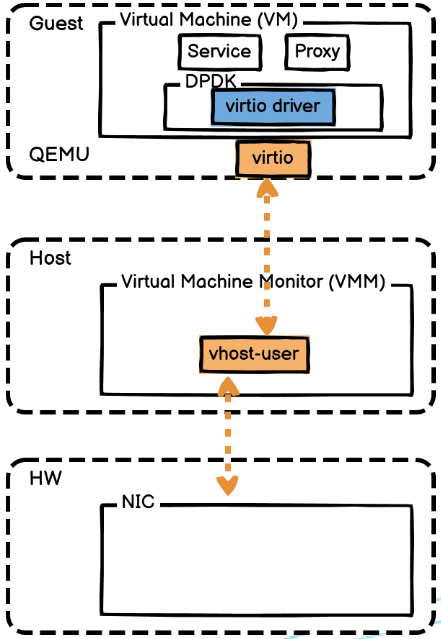
[1] Http3 Full Stack Fest, Daniel Stenberg

QUIC & HTTP/3 Support in Istio

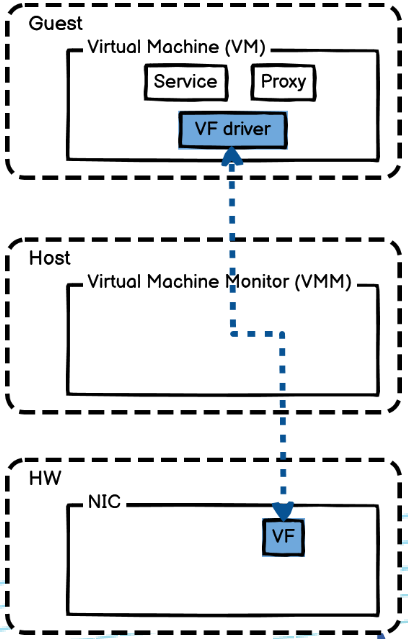
- Should take Gateway as the 1st step
  - Less engineering effort
  - Particularly valuable for some VM user scenarios
- Limitations
  - Envoy QUIC support in early stages
    - Security
      - Both the downstream and upstream need to be trusted
    - Stability (quite a few issues/broken functionalities)
      - Concurrency limitations
    - Lack of docs etc.

VM High Performance Networking

- VM ⇔ Host IO interface
  - Relay
    - DPDK
  - Passthrough
    - SRIOV
- SRIOV
  - Single Root I/O Virtualization
- SIOV
  - Scalable I/O Virtualization
- SRIOV => SIOV



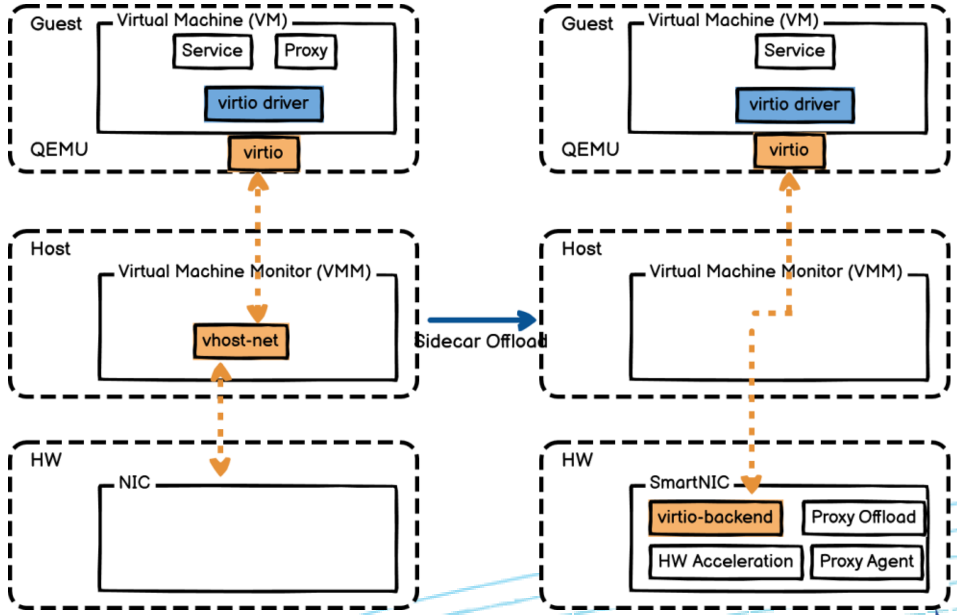
relay w/ DPDK



SRIOV

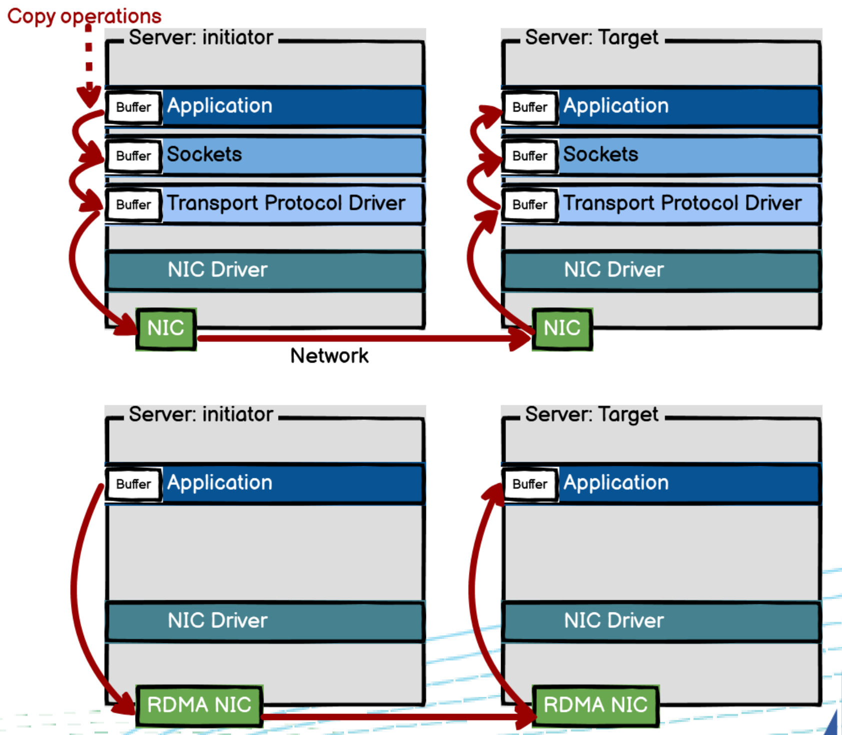
SmartNIC – Sidecar Offload

- Ultimate goal
  - Proxyless services (for high performance)
- Offload
  - Traffic management
  - Security (DDoS defense...)
- HW acceleration
  - Crypto
  - Rule matching
- Further isolation w/ host
- CapEx, OpEx



RDMA (Remote Direct Memory Access)

- Advance transport protocol (same layer as TCP and UDP)
- Main features
  - Remote memory r/w semantics in addition to send/receive
  - Kernel bypass / direct user space access
  - Transport fully offloaded to the NIC HW
  - Zero-copy operation
  - Secure, channel based IO
- Application advantage
  - Low latency
  - High bandwidth
  - Low CPU consumption
- Istio: cross-node Proxy to Proxy kernel bypass w/ HW acceleration



Quick Summary, Today

Istio is ready-to-go for VM native.
And should/will be ready for MORE!

Thank you!

Github:
@kailun-qin
@harryge00