Network Service Mesh

Intro

Nikolay Nikolaev
Some thoughts on Cloud Native Networking
  - Good Example: K8s Networking API
Story Time: Marsha and the Multi-cloud Application
How Network Service Mesh works at a High Level
More Resources/Get Involved
Some Thoughts on Cloud Native Networking
Cloud native technologies empower organizations to build and run scalable applications in modern, dynamic environments such as public, private, and hybrid clouds. Containers, service meshes, microservices, *immutable infrastructure*, and declarative APIs exemplify this approach.

These techniques enable *loosely coupled* systems that are resilient, manageable, and observable. Combined with robust automation, they allow engineers to make high-impact changes frequently and predictably with *minimal toil*.

The Cloud Native Computing Foundation seeks to drive adoption of this paradigm by fostering and sustaining an ecosystem of open source, vendor-neutral projects. We democratize state-of-the-art patterns to make these innovations accessible for everyone.
Minimal Toil Networking

- **Minimal Conceptual Toil**
  - No Interfaces/Routes/Subnets as concepts
  - Conceptualize as ‘Network Services’
    - Intersection of Connectivity/Security/LoadBalancing/NAT/etc

- **Minimal Consumption Toil**
  - Ask for what you want by name:
    - secure-intranet-connectivity
    - manufacturing-partner-network
    - marshas-app-connectivity
Example: K8s Networking:
- K8s Network Conceptually:
  - Connectivity - L3 between all Pods
  - Security - Network Policies
  - Load Balancing - Services/Endpoints
  - But mostly Intra-cluster
- Consumption:
  - It’s just there
  - Network Policies/Services are easy
Loose Coupling

• App Microservices are loosely coupled to each other:
  ○ Allows lego block assembly of complex patterns from simple primitives
  ○ Flexibility

• Historically, Networking is **strongly coupled**
  ○ Networking is defined at the level of
    ■ Cluster
    ■ Datacenter
    ■ VPC
    ■ Etc
  ○ Coarse Granularity - many workloads get the same ‘Network Service’ based on where they **run**, not what they need.
    ■ You may have fine tuning, but only the same ones everyone in that domain gets
Loose Coupling

Example: K8s Networking:
- K8s API loosely coupled to implementation
  - Many CNI plugins
- Strongly coupled to cluster
  - Usually one CNI per cluster
- Single Edge for entire cluster
  - Or possibly multiple clusters
- Coarse granularity
  - Realistically all workloads in cluster or none
## Immutable Infrastructure

- Pods/Network Services use, rather than modify, infrastructure
- Unprivileged

<table>
<thead>
<tr>
<th>CNI</th>
<th>K8s</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Network Functions</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>HW Resources</th>
</tr>
</thead>
</table>
Story Time
Marsha and the multi-cloud application
Meet Marsha. Marsha is building an app that has a lot of **multi-cloud** and **hybrid** cloud aspects.
Marsha’s app has workloads running in public K8s clusters, private K8s clusters, legacy VMs, and bare metal on prem servers.
Each of those domains has its own intra-networking.
Marsha doesn’t want to connect the clusters/VIMs/DC networks. That’s too course a granularity.
What Marsha really needs is Connectivity between the workloads in her app, wherever they are.
Enter Network Service Mesh. Network Service Mesh uses ‘vWires’ to connect individual Pods/Workloads to a Network Service that provides her desired Connectivity/Security.
The ‘marshas-app-connectivity’ provides the correct Connectivity/Security/other services for **her** app.
Marsha directs her Pods to consume Network Service:

```yaml
apiVersion: v1
type: Pod
metadata:
  name: marsha-pod-1
annotations:
  ns.networkservicemesh.io: marsha-app-connectivity
```
Loose Coupling

Breaks strong coupling of networking to cluster/VIM/DC network.
Day 0 - K8s admin enables Network Service Mesh on cluster
  - `helm install nsm`

Day 1 - Network Service Deployed to K8s cluster
  - `helm install marshas-app-connectivity`

No change to underlying K8s
Works with any CNI
Currently working in our CI/CD on:
- GKE
- AKS
- EKS
- Vanilla K8s on VMs/bare metal
- Kind
How the Magic Works
What is NSM

• A Network Service definition
• A gRPC API to describe, publish and consume Network Service(s)
• A distributed control plane with minimum shared state
• A concrete Kubernetes based implementation
  • Runtime interface injection/removal for Pods. Orthogonal to CNI
  • Leverage etcd as a central shared storage through CRDs
  • Use Kubernetes `DaemonSet` to provision local node agents
  • VPP as a base forwarding component
How NSM works
```yaml
apiVersion: apps/v1
kind: Deployment
spec:
template:
spec:
containers:
- name: alpine-img
  image: alpine:latest
  command: ['tail', '-f', '/dev/null']
metadata:
  name: my-app
  annotations:
    ns.networkservicemesh.io: service?label=value
```
Network Service Manifest

```
apiVersion: networkservicemesh.io/v1
kind: NetworkService
metadata:
  name: secure-intranet-connectivity
spec:
  payload: IP
  matches:
    - match:
        sourceSelector:
          app: firewall
        route:
          - destination:
              destinationSelector:
                app: vpn-gateway
        route:
          - destination:
              destinationSelector:
                app: firewall
```

- **Describe the type**
  NetworkService

- **The name of the service is** secure-intranet-connectivity

- **Match the service request labels for** app=firewall

- **Find an endpoint that implements** secure-intranet-connectivity and is labeled app=vpn-gateway

- **Wildcard sourceSelector**
Resources/Get Involved
Resources

www.networkservicemesh.io

github.com/networkservicemesh/networkservicemesh

networkservicemesh.io/community/
• Tuesday, June 25th 14:00-16:00 VMware booth G8
• Wednesday, June 26th 11:30 CNCF Answer bar
Network Service Mesh

Nikolay Nikolaev - nnikolay@vmware.com