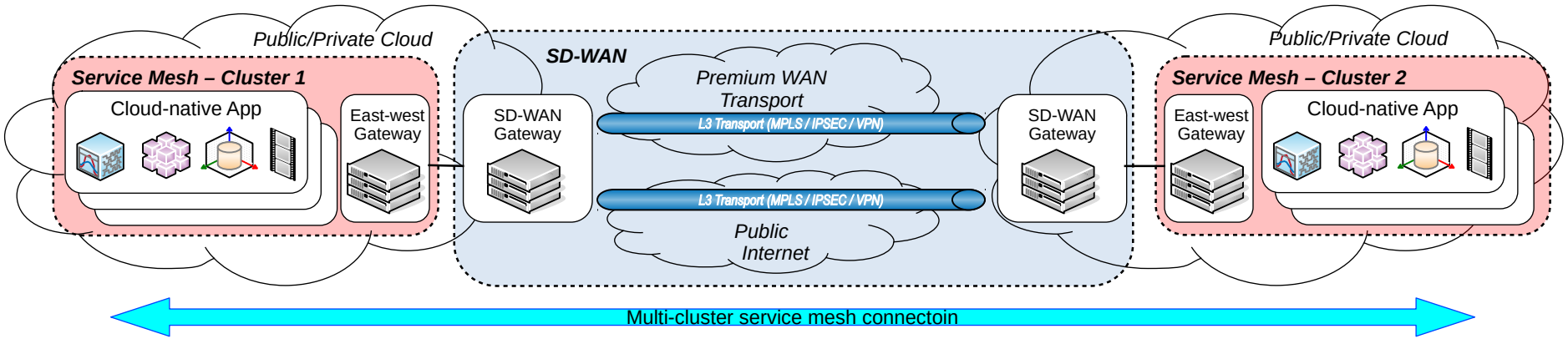
A Multi-cluster SD-WAN East-west Gateway

Petra Kalocsai, Tamás Lévai, Gábor Rétvári (BME)

Goals

- build an east-west (EW) gateway to seamlessly interconnect two or more service-mesh clusters over an SD-WAN fabric for a consistent end-to-end user experience
- integrate the L4/L7 traffic management policies on the service-mesh side with the L3/L4 policies on the SD-WAN interconnect
- end-to-end observability and the security across the service mesh and the SD-WAN segments
- demos!



User stories

- service-level SD-WAN policies: the service owner wants all accesses to the `payment.secure` HTTP service (port 8080) to be mapped to the high-prio SD-WAN tunnel
- L7 traffic management: same, but now only GET queries to the API endpoint `/payment` should map to the high-prio tunnel
- resiliency: automatic failover between SD-WAN tunnels
- monitoring: end-to-end observability

Principles

- separate traffic flows intended to be sent over different SD-WAN tunnels on the egress EW gateway, in order for the SD-WAN to be able to apply the proper SD-WAN priority
- services between clusters are exported/imported using the Multi-cluster Services API (KEP-1645) CRDs, extended with a set of annotations to control SD-WAN routing policies
- L7 policies can be applied on top of the service-level exports/imports to control the way traffic egresses from, and ingresses into, the cluster

Service-level traffic management

- associate an SD-WAN policy with distinct Kubernetes service
- no way to impose additional L7-level SD-WAN policies on top (yet)
- basic model will be extended to a fully-fledged L7 model later
- separate CRDs to define WAN policies, service exports and service imports

WAN policies

- each global service can have a WAN policy associated with it, defined in a cluster-global CRD named `WANPolicy.mcw.l7mp.io`
- format is unspecified for now; below is a sample to define 2 policies
  - high-priority tunnel with stringent SLAs associated with the port 31111
  - high-priority tunnel with less stringent SLAs associated with the port 31112

```
apiVersion: mcw.l7mp.io/v1alpha1
kind: WANPolicy
metadata:
name: sd-wan-priority-high
spec:
  tunnel: business
  port: 31111
  sla:
    jitter-ms: 50
    latency-ms: 100
    loss-percent: 1
```

Service exports

- services have to be explicitly exported from the hosting cluster to allow access from other clusters
- the below will export the service called `payment` in the `secure` namespace over the high-prio SD-WAN tunnel

```
apiVersion: multicluster.k8s.io/v1alpha1
kind: ServiceExport
metadata:
  name: payment
  namespace: secure
  annotations:
    mcw.l7mp.io/mc-wan-policy: sd-wan-priority-high
```

Service imports

- exported services are automatically imported into all clusters in which the service's namespace exists (including the exporting cluster)
- ServiceImport CRDs are automatically created by the controller

```
apiVersion: multicluster.k8s.io/v1alpha1
kind: ServiceImport
metadata:
  name: payment
  namespace: secure
spec:
  type: ClusterSetIP
  ports:
  - name: http
    protocol: TCP
    port: 8080
```

Demo!

L7 traffic management

- up to this point, SD-WAN policies could be applied to individual Kubernetes services only
- the below shows how to add L7 traffic management policies by reusing the ServiceImport/ServiceExport CRs
- we assume that the `payment.secure` service is exported over both the high- (`payment-high-prio`) and the low-priority tunnel (`payment-low-prio`)

Client-side L7 HTTP request routing

- goal is to route requests to different SD-WAN tunnels depending on the HTTP headers
- in the below, only GET requests to the `/payment` HTTP path would be routed to the high-priority tunnel, everything else should go over low-prio tunnel

```
apiVersion: networking.istio.io/v1beta1
kind: VirtualService
metadata:
  name: payment
  namespace: secure
spec:
  hosts:
  - payment.secure.svc.clusterset.local
  http:
  - match: [ uri: {prefix: "/payment" }, method: { exact: GET } ]
    route:
    - destination:
        host: payment-high-prio.secure.svc.clusterset.local
  - route:
    - destination:
        host: payment-low-prio.secure.svc.clusterset.local
```

Demo!

Command line tool (WiP!)

- there is a proof-of-concept command line tool to automate service imports and service exports

```
# export a service
mcwanctl --context $CTX1 export payment/secure --wan-policy=high
# get status
mcwanctl --context $CTX1 status payment/secure
# import
mcwanctl --context $CTX2 import payment/secure --ingress-gw=<GW_IP_ADDRESS>
```

- ingress Gateways already done
- Petra is working on it!

Further plans

- play with the spec, see pros and cons in practice
- find new ways to map the semantics to existing Kubernetes APIs
- implement the command line tool
- implement a fully-fledged multi-cluster API operator to automatically reconcile ServiceImports/ServiceExports
- integrate with the `l7mp/stunner` Kubernetes Gateway project