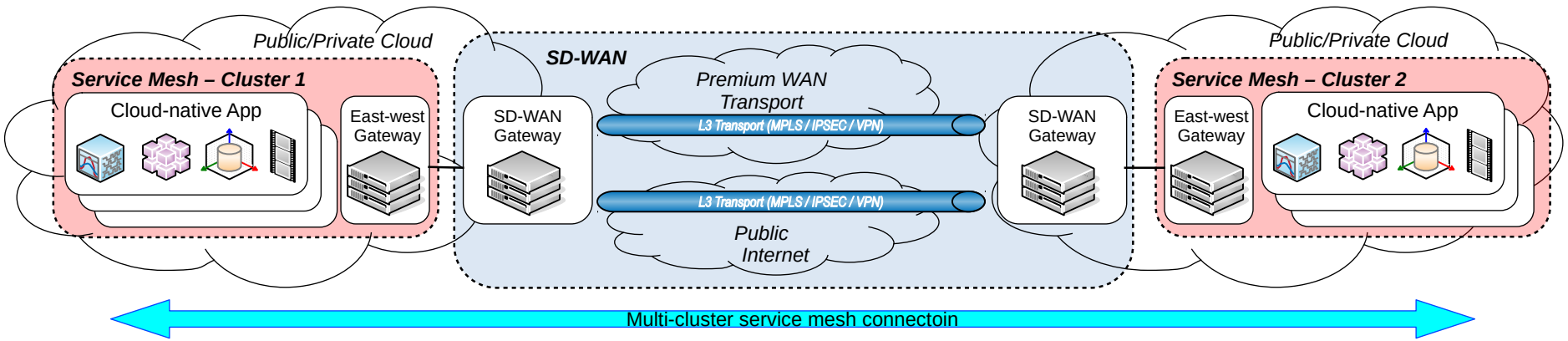
A Multi-cluster service mesh east-west gateway for SD-WAN

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



User stories

- enforce 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 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
- enforce the L4/L7 traffic management policies and SD-WAN priorities on the egress EW gateway, the ingress EW gateway merely proxies the service traffic to the corresponding endpoints in the cluster
- reuse official Kubernetes APIs, namely the Multi-cluster Services API (KEP-1645) for service export/import and the Kubernetes Gateway API for L4/L7 policies
- implement the whole thing on top of a standard Kubernetes gateway implementation

Service exports

- services have to be explicitly exported from the hosting cluster to allow access from other clusters

```
apiVersion: mc-wan.l7mp.io/v1alpha1
kind: ServiceExport
metadata:
  name: payment
  namespace: secure
spec:
  http:
    rules:
      - matches:
          - method: GET
            path: { type: PathPrefix, value: /payment }
        backendRefs:
          group: mc-wan.l7mp.io
          kind: WANPolicy
          name: sd-wan-priority-high
      - matches:
          - path: { type: PathPrefix, value: /stats }
        backendRefs:
          group: mc-wan.l7mp.io
          kind: WANPolicy
          name: sd-wan-priority-low
```

- service exports consist of 2 parts: the L4/L7 policies at the ingress side plus the SD-WAN priority to apply to matching traffic

Service imports and WAN policies

- 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
- service exports and imports can associate the requested WAN policy for a service

```
apiVersion: mc-wan.l7mp.io/v1alpha1
kind: WANPolicy
metadata:
  name: sd-wan-priority-high
spec:
  tunnel: business
  port: 31111
```

Mechanics

- the spec contains a description of how to map the above CRDs to actual Kubernetes APIs: https://github.com/l7mp/mc-wan

```
apiVersion: gateway.networking.k8s.io/v1beta1
kind: HTTPRoute
metadata: { name: payment-secure, namespace: mc-wan }
spec:
  parentRefs:
  - name: payment-secure
  hostnames:
    - payment-secure.mcw.svc.cluster.local
    - payment.secure.svc.cluster.local
  rules:
   - matches: [ method: GET, path: { type: PathPrefix, value: /payment} ]
     filter: { urlRewrite: { hostname: payment.secure.svc.cluster.local }}
     backendRefs:
       - name: mc-wan-cluster-1-target
         namespace: mc-wan
         port: 31111
         weight: 1
   - matches: [ path: { type: PathPrefix, value: /stats } ]
     filter: { urlRewrite: { hostname: payment.secure.svc.cluster.local }}
     backendRefs:
       - name: mc-wan-cluster-1-target
         namespace: mc-wan
         port: 31112
         weight: 1
```

TODO

- the plan is now to configure the gateways manually to play with the idea
- iterate on the CRDs and the mechanics
- and finally write an actual Kubernetes operator to automate this: this will be a massive undertaking