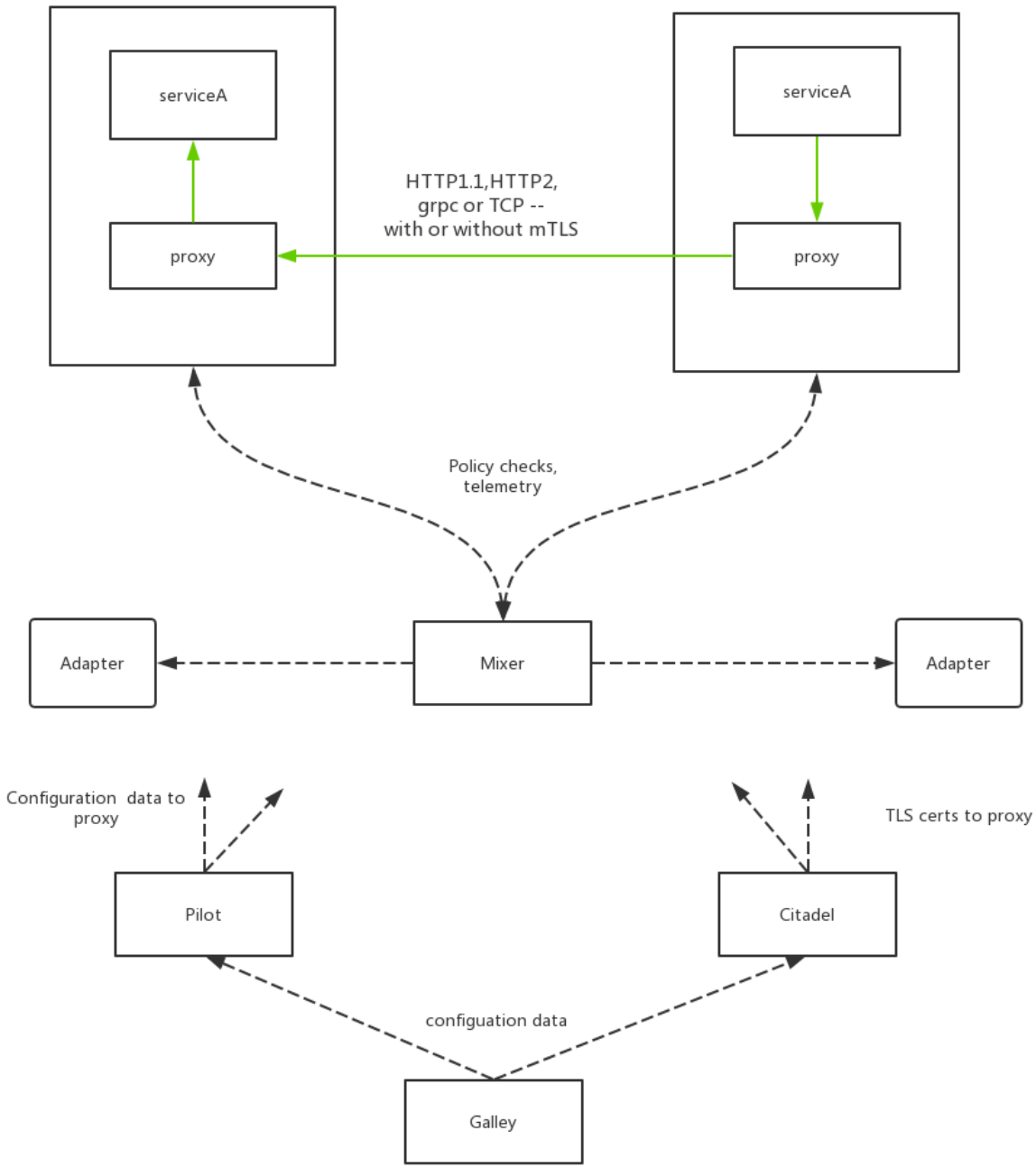
istio考试

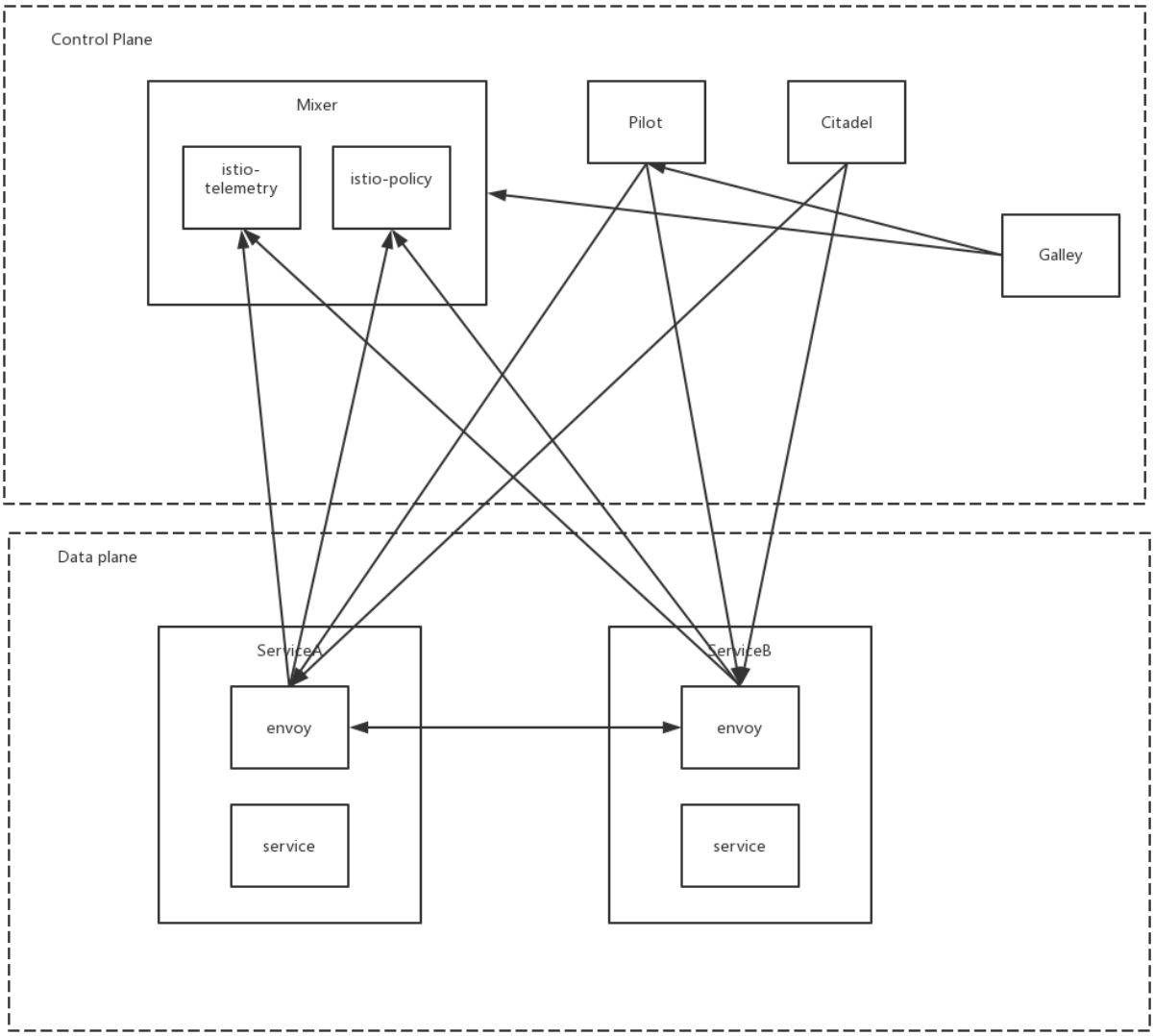
- 这里我对istio的理解基于istio1.1.7

1. 问题1

1.1 画出istio的架构图



1.2 画出组件交互图



1.3 写出envoy和mixer,pilot,auth的交互方式(push还是定时)

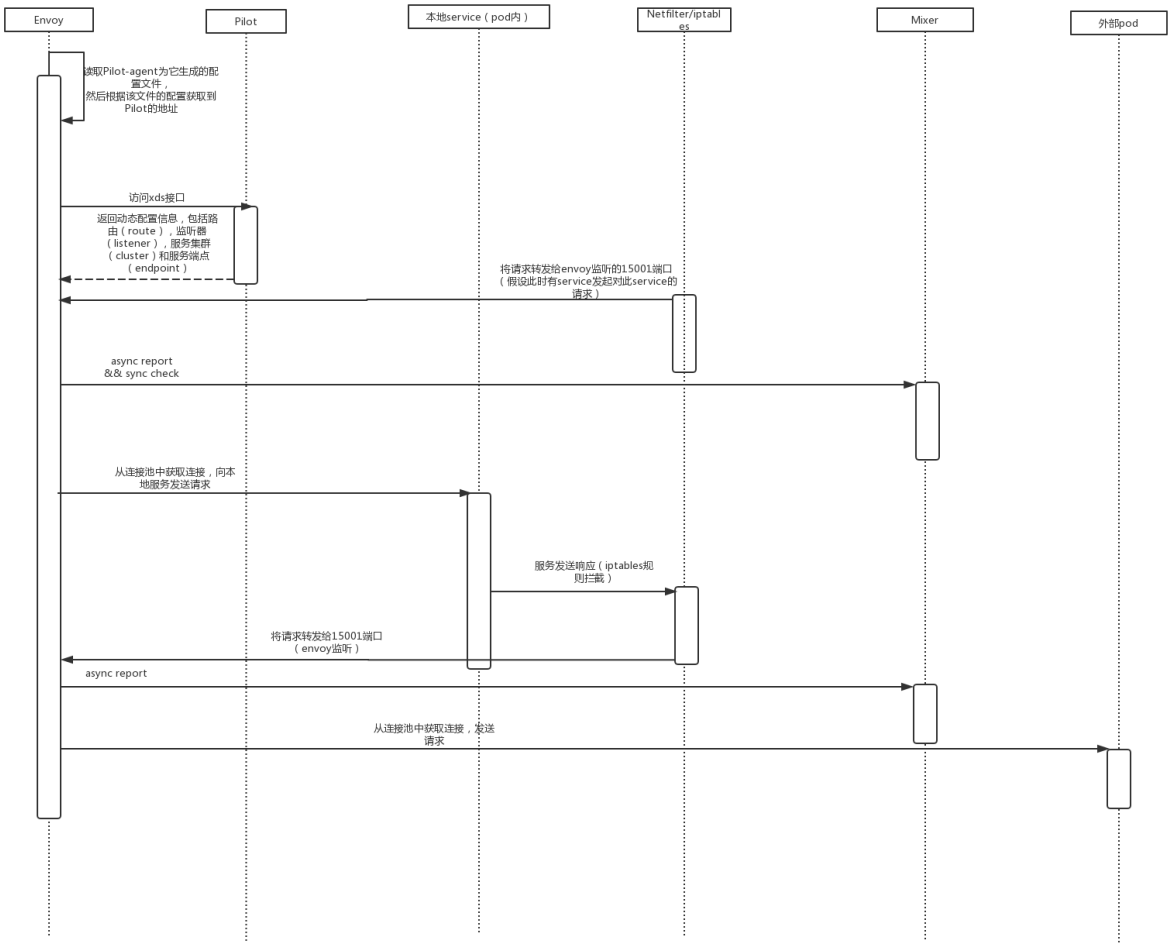
- mixer - envoy, 定时(check,quota,report request)
- pilot - envoy, pilot push data to envoy
- auth(Citadel)- envoy, auth(Citadel) push data(TLS certs etc) to envoy

2. 问题2

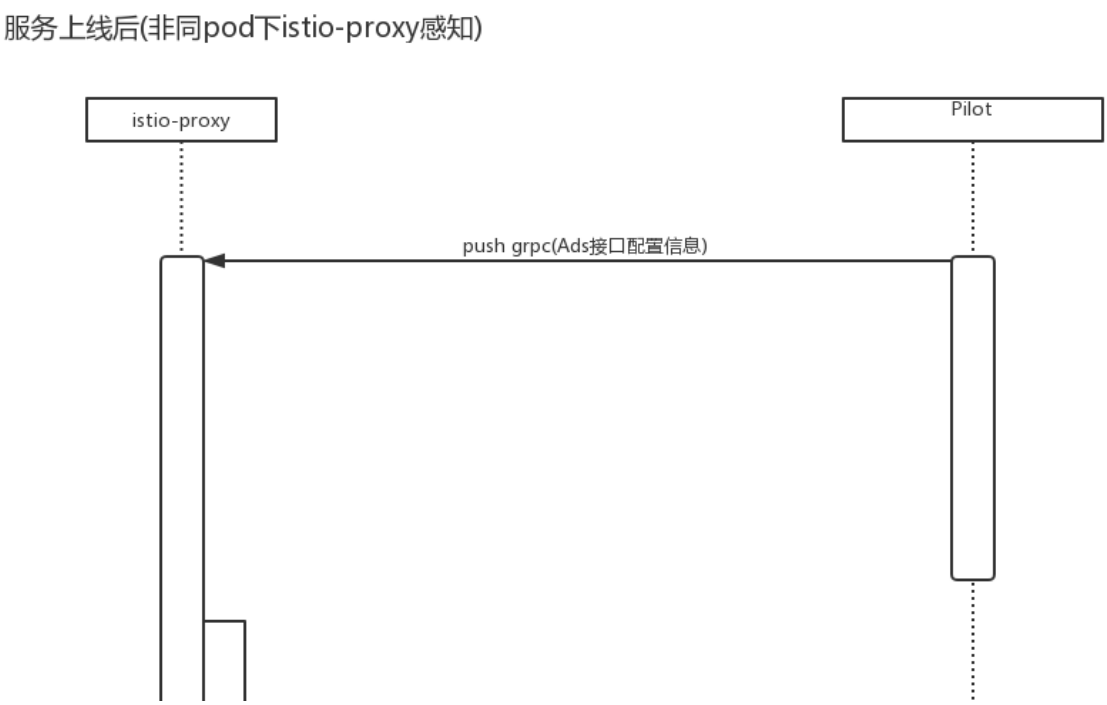
2.1 服务上被客户端所感知的时间序列图

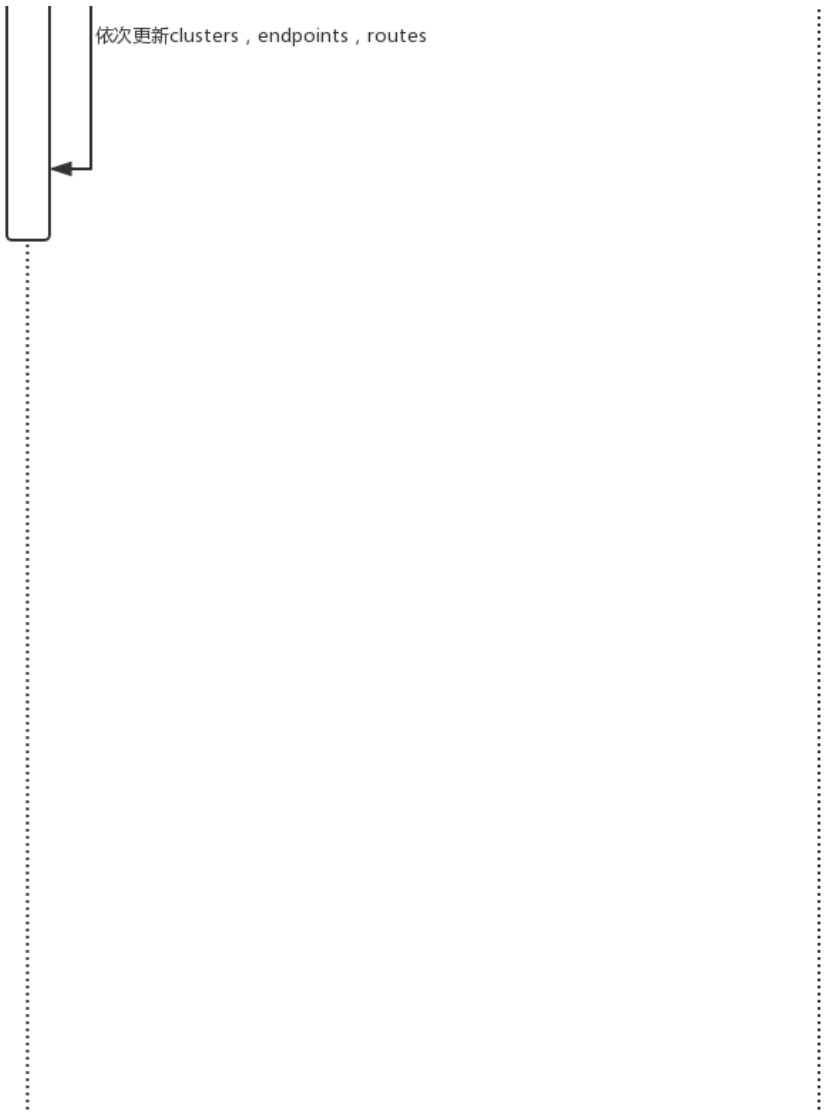
- 服务上线后该服务下的istio-proxy（envoy）时序图

服务上线后（所在pod Envoy视角）



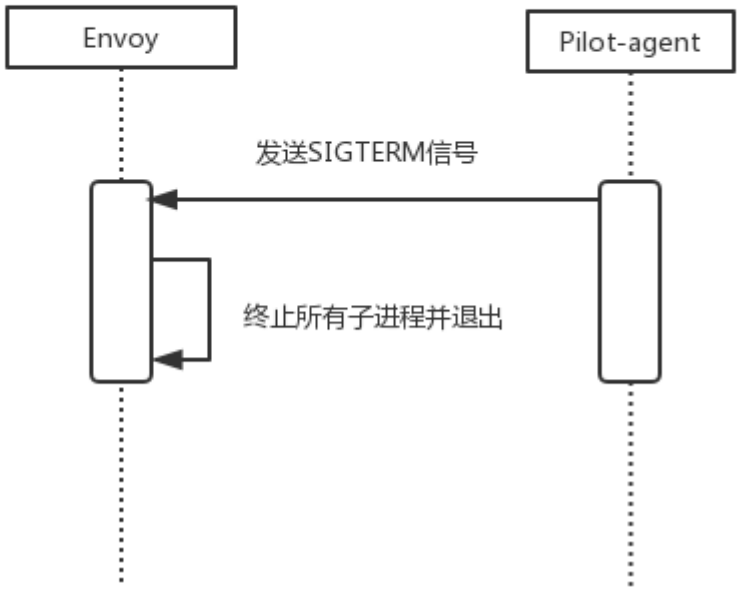
- 服务上线后其他service的envoy时序图



- 

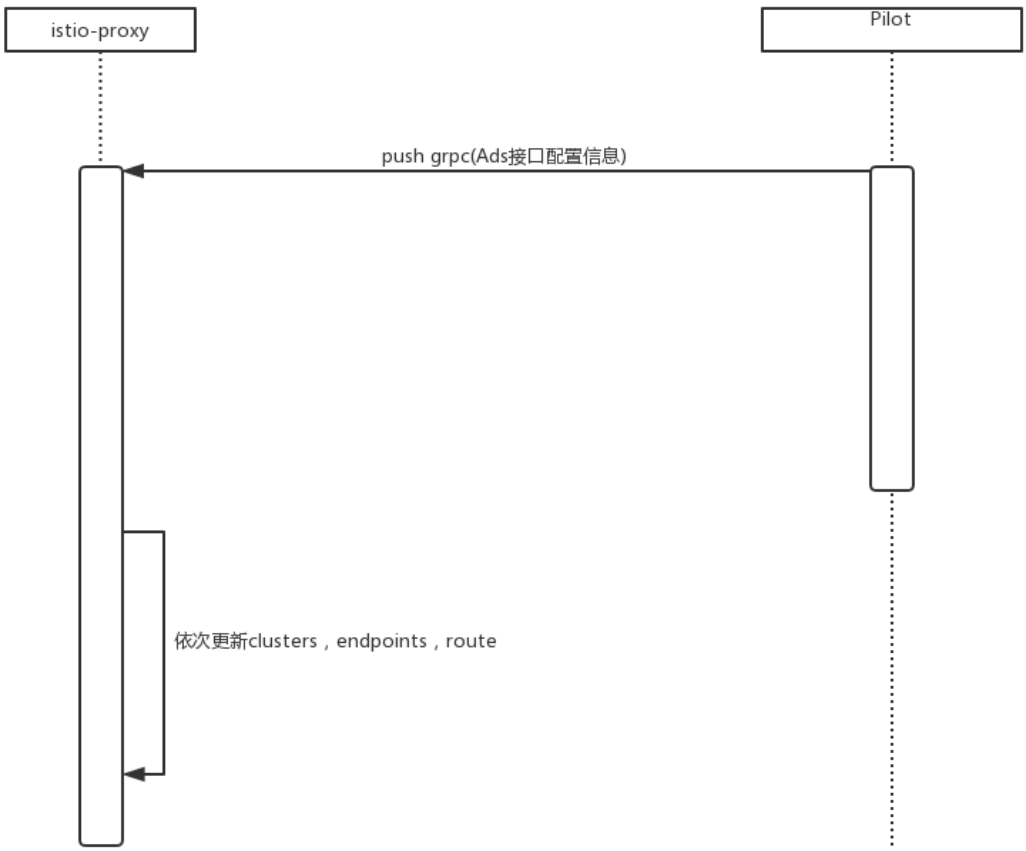
- 服务下线后该服务下的istio-proxy（envoy）时序图

- 服务下线后（所在pod Envoy视角）



- 服务下线后其他service的envoy时序图

服务下线后(非同pod下istio-proxy感知)



3. 问题3

3.1 分别说出istio对于http,grpc，tcp支持有哪些不同和相同（原理）

- 相同
  - istio对http，grpc，tcp的支持都是基于envoy的监听器实现的
  - Istio可以为任何TCP连接（L4）提供路由，mTLS和授权。（包括HTTP,GRPC L7）
- 不同
  - 对于HTTP，HTTP / 2和gRPC（L7）还支持其他功能（例如：基于路径的路由，标头，每个请求的监视延迟，每个方法调用的监视等）

4. 问题4

4.1 整理一个完整的istiodemo,包含根据地域进行灰读，流量降级，熔断，故障注入，jwt的认证，日志收集，prometheus监控（演示）

根据地域灰度...测试环境在同一个数据中心没法做真实的根据地域灰度。

思路如下：

1. 将新版本部署到对应区域的node上（selector）
2. 定义规则，将subset路由到第一步的pod上

```
---
apiVersion: networking.istio.io/v1alpha3
kind: DestinationRule
metadata:
  name: {{ projectName }}
  namespace: java-service
spec:
  host: {{ projectName }}
  subsets:
  - name: normal
    labels:
      release: normal
  {%- for version in allVersion %}
  - name: {{ version.branchIdentify }}
    labels:
      release: {{ version.branchIdentify }}
  {%- endfor %}

---
apiVersion: networking.istio.io/v1alpha3
kind: VirtualService
metadata:
  name: {{ projectName }}
  namespace: {{ namespace }}
spec:
  hosts:
  - {{ projectName }}.{{ namespace }}.svc.cluster.local
  http:
  {%- for version in allVersion %}
  - match:
    - headers:
        x-version:
          regex: "^(.*?;)?({{ version.gray }})(;.*)?$"
    route:
    - destination:
        host: {{ projectName }}.{{ namespace }}.svc.cluster.local
        subset: {{ version.branchIdentify }}
  {%- endfor %}
  - route:
    - destination:
        host: {{ projectName }}.{{ namespace }}.svc.cluster.local
        subset: normal
```

5. 问题5

5.1 能不能做到pod挂掉5s, istio能够感知到？如果能，请说明为什么？并附上相关文档，如果不能，请解释原因

能。原因如下：

- pilot通过watch来检测kubernetes apiserver的资源变化（pod挂掉对应节点上的kubelet会每隔1s向docker查询各容器的状态，并将查询的结果发给kube-apiserver），并于客户端envoyb保持一个grcpc长连接来g更新ads对应数据https://istio.io/docs/reference/commands/pilot-discovery/
- envoy发起主动健康检查（默认时间间隔为100s，可以修改源码/admin_util.go中healthCheckInterval，不过这样会产生大量的主动检查流量），异常pod不可用envoy会将其标记为不健康（这里存在一个恐慌机制panic，当超过恐慌阈值会将所有服务实例重新加到负载均衡列表，默认阈值为上游集群实例的50%），更新客户端负载均衡 http://www.servicemesher.com/envoy/intro/arch_overview/health_checking.html

非官网文档：

- Service Mesh深度学习系列part3—istio源码分析之pilot-discovery模块分析（续）
- kubernetes-troubleshooting-book

5.2 能不能排出网络抖动引起的节点驱逐？如果能怎么做，列出配置

识别网络抖动类别：

- dns抖动
- ...

解决方案：

- 查看kube-dns状态日志与节点日志