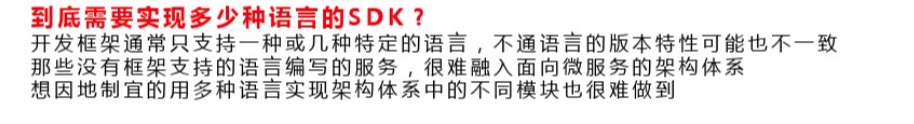
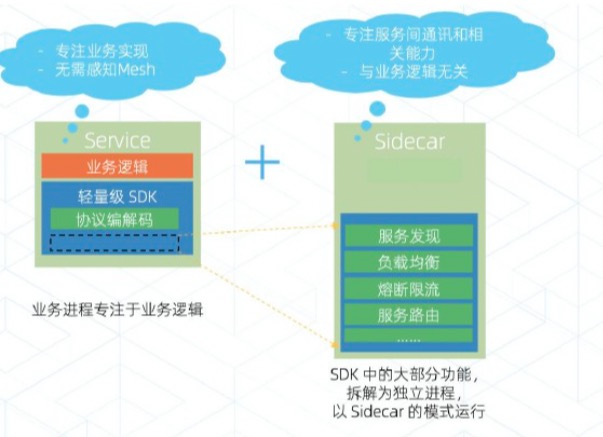
## 定义





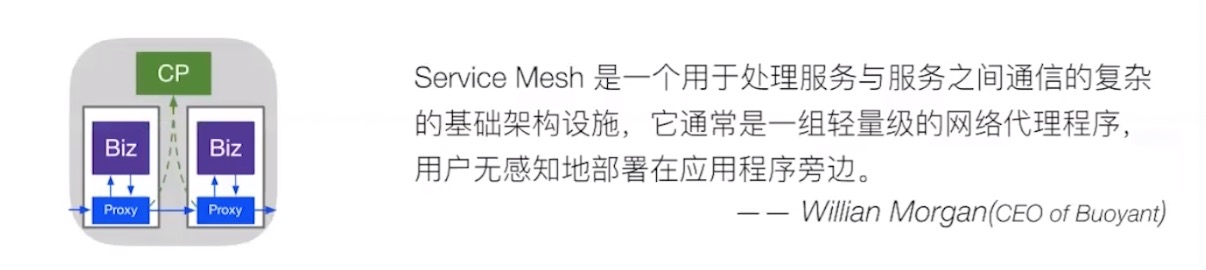
框架升级也很麻烦

将服务治理下沉到基础设施



控制面和执行者 执行者注入 流量劫持 , 降低通信性能 稳定性将会取决一mesh 管理执行者

2016年才开始研究的，现在已经开始商用。



和应用程序部署在一起容器的轻量级代理.

处理服务和服务之间或者服务与中间件之间的代理.

而且这些轻量级代理有一个统一的cp控制面来控制,这样的话通过通过cp就可以随时操作这个代理,那么我们通过代理就可以控制服务相关的东西.

也可以在mesh上做监控啥的,反正就很好,相当与给所有的service都加了一个统一管理的代理.

还可以在mesh上面做流量控制,熔断什么.

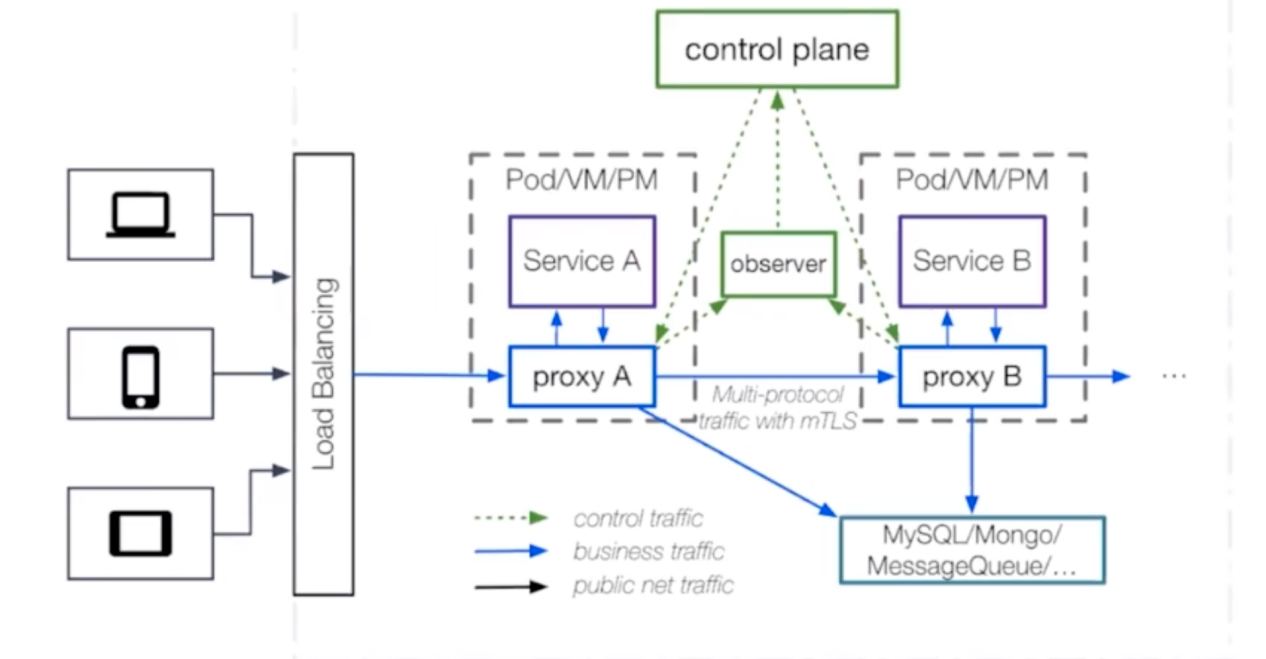
还以把kite给集成到mesh上,从而可以统一控制kite的版本.

还有各种中间件的配置,mysql的配置你要去mysql的平台,redis的配置要去redis的平台,有mesh的话可以把这些配置都给你代理起来,统一在mesh的cp去操作.

还可以兼容各种协议.

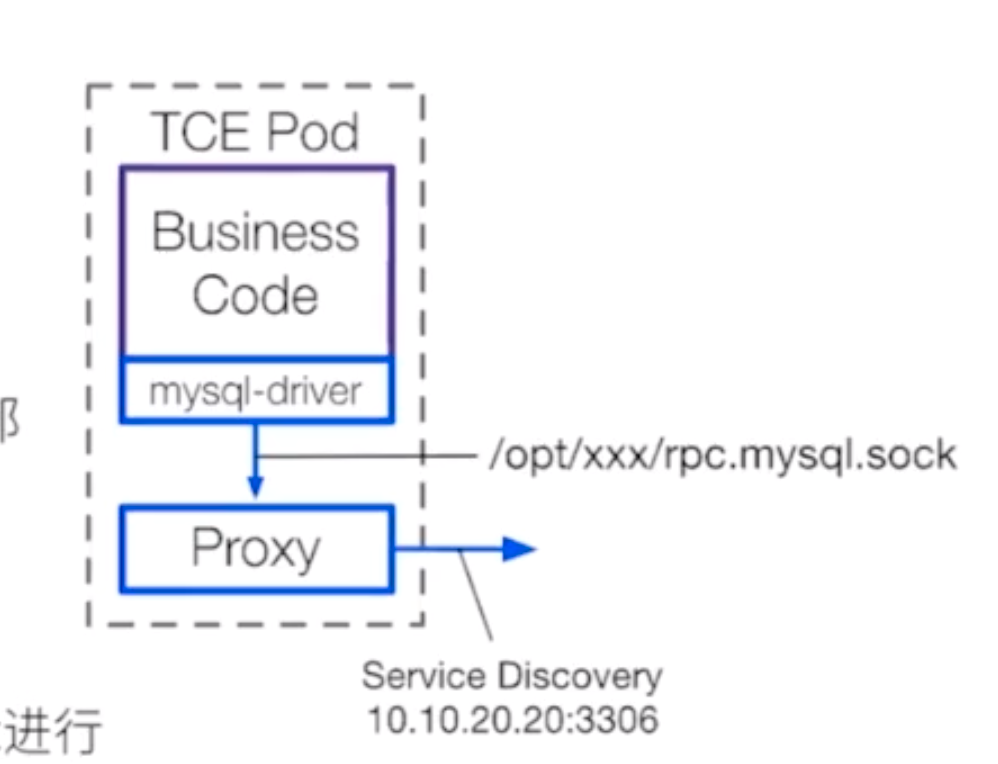
所谓的mesh劫持服务的流量进行统一控制.

## 流程



全权代理service的流量.

可以看到,kite不再监听真正对外的端口了,而是通过一个域套接字和mesh链接,而mesh再去监听真正的网络端口.



对于服务对mysql的请求,mesh是这样劫持的,他和mysql哪个包(mysql-driver)商量好,他们走mesh通过域套接字链接到mesh,而mesh再去链接真正的mysql.



微服务的缺点。

1，语言碎片化，

语言用的很多，框架也很多。

2，框架更新困难

比如框架调用链增减字段，或者框架有bug，想把所有用这个框架的人的版本都更新了很困难。

3，协议多样化

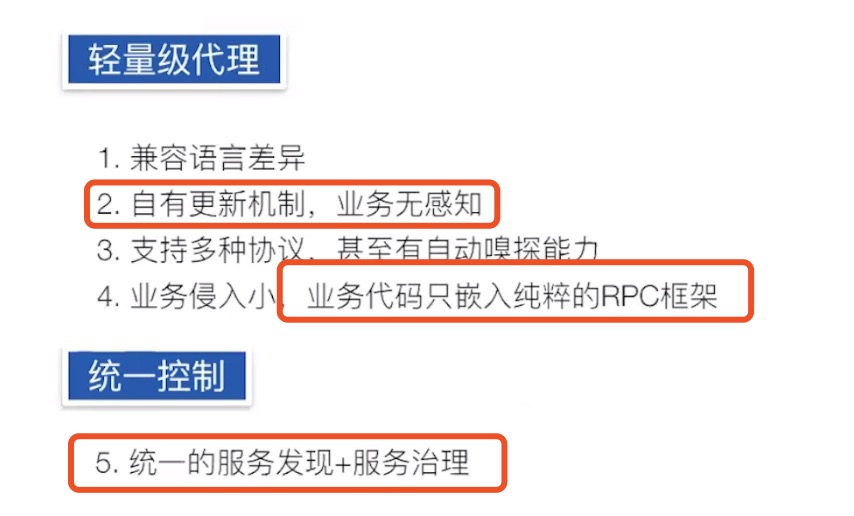
4，业务入侵性强

业务自己服务治理，会影响性能，不够统一。

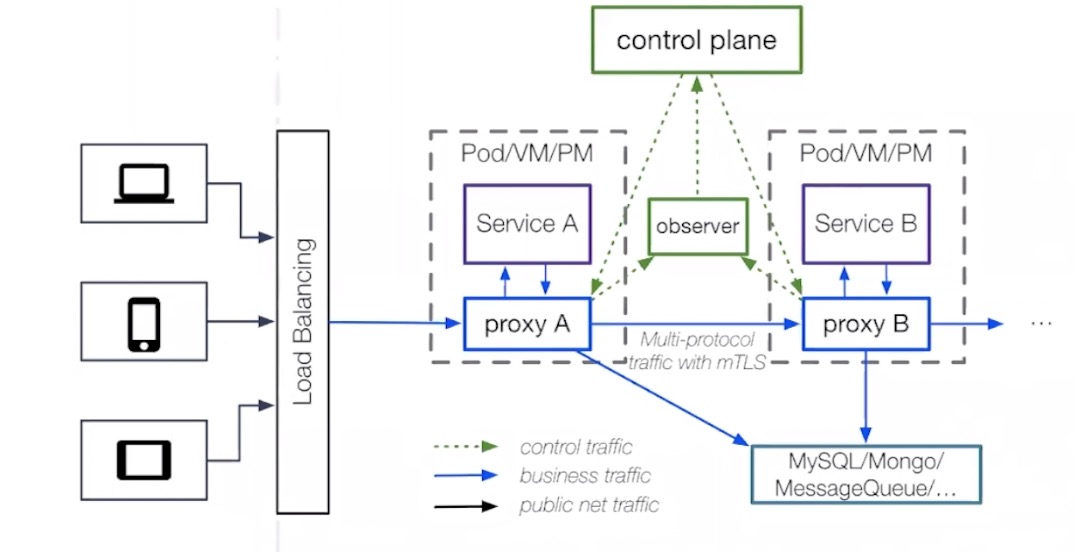
5， 服务治理不对齐

各种mysql redis什么的配置啊，tmq的配置啊，不能统一管理。

微服务这种小而自治的东西会让不同的服务的一些标准不够统一。



这些都在mesh的控制面中来操作。



所谓apigateway 就是在nginx和api服务之间统一加了一个apigateway服务,这个可以统一管理调度熔断配置等操作.